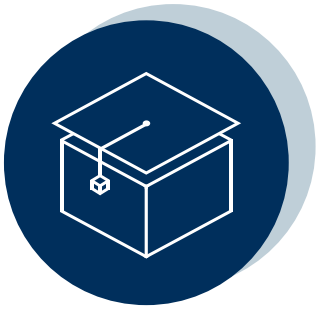

BISE Student

https://bise-student.io

BACHELOR'S THESIS

An Investigation on Online Machine Learning for Anomaly Detection in Time Series Data

Publication Date: 2024-09-16

Author
Sebastian Niklas WETTE
Darmstadt, Germany
sebastian.wette@stud.tu-darmstadt.de
0x181b0929177CD63BF1E54Df2aFf0B136d2954F10

Abstract

Concept drift in time series data poses a problem for many machine learning algorithms. Underlying shifts in the statistical properties of data lead to a decline in the performance of batch-trained models. Anomaly detection algorithms working with forecasts on the future behavior of a system suffer from these effects. Thus, adaption to concept drift is a fundamental challenge for anomaly detection systems like this, especially in quickly evolving environments. Instead of retraining models from scratch regularly, models can continuously learn and update themselves as new data arrives, a strategy known as online learning. This thesis investigates the efficacy of online machine learning in prediction-based anomaly detection for time series data under concept drift, focusing on accuracy and computational efficiency compared to batch-trained methods. The work presents a proof of concept for prediction-based anomaly detection using online learning. Furthermore, the research compares the performance of the presented approach...

Keywords: Online ML, Anomaly Detection, Time Series, Concept Drift
Methods: systematic literature review, benchmarking

Submission Date: 2024-09-09
Submission Contract: 0xC2DdBdD2b9A1A317f6976005ec62A61149F1B36c

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Hochschule Darmstadt

– Fachbereich Informatik –

An Investigation on Online Machine Learning for Anomaly Detection in Time Series Data

Abschlussarbeit zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

vorgelegt von

Sebastian Wette

Matrikelnummer: 769178

Referent	:	Prof. Dr. Andreas Heinemann
Korreferent	:	Prof. Dr. Florian Heinrichs
Abgabedatum	:	07. März 2024

ERKLÄRUNG

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Darmstadt, 07. März 2024

Sebastian Wette

ABSTRACT

Concept drift in time series data poses a problem for many machine learning algorithms. Underlying shifts in the statistical properties of data lead to a decline in the performance of batch-trained models. Anomaly detection algorithms working with forecasts on the future behavior of a system suffer from these effects. Thus, adaption to concept drift is a fundamental challenge for anomaly detection systems like this, especially in quickly evolving environments. Instead of retraining models from scratch regularly, models can continuously learn and update themselves as new data arrives, a strategy known as online learning.

This thesis investigates the efficacy of online machine learning in prediction-based anomaly detection for time series data under concept drift, focusing on accuracy and computational efficiency compared to batch-trained methods. The work presents a proof of concept for prediction-based anomaly detection using online learning. Furthermore, the research compares the performance of the presented approach with the well-known batch-trained models SARIMA and Prophet, using real-world data from Deutsche Telekom's IP Backbone, focussing on accuracy and efficiency.

Resulting measurements indicate that the online learning approach is more accurate in detecting anomalies when concept drift exists in the data. It exhibits superior adaptability to concept drift, whereas batch-trained models fail to produce adequate forecasts after a changepoint. However, batch-trained models perform better in static data environments. Lower CPU and memory usage and faster runtimes indicate the superior computational efficiency of the online learning method.

Finally, this study confirms the superiority of online learning for prediction-based anomaly detection under concept drift. It suggests potential applications in real-time systems and dynamic data environments. There are also some limitations to this approach that motivate future work. Forthcoming research should explore more diverse online learning algorithms for different use cases and address the challenges of online MLOps, namely hyperparameter tuning. Additionally, distinguishing between anomalies and concept drift remains a critical challenge, suggesting avenues for further exploration in adaptive learning strategies.

ZUSAMMENFASSUNG

Concept Drift ist für viele maschinelle Lernverfahren ein Problem. Veränderungen in den statistischen Eigenschaften der Daten resultieren bei konventionellen, auf *Batches* trainierten Modellen in Leistungseinbußen. Dies betrifft auch Algorithmen zur Anomalieerkennung, die auf Prognosen des zukünftigen Systemverhaltens basieren. Häufig wird versucht, das betroffene Modell in regelmäßigen Abständen komplett neu zu trainieren, um das Problem zu lösen. Dies ist allerdings ein rechenintensiver und komplexer Prozess. *Online Machine Learning* stellt eine fortschrittlichere Strategie dar. Dabei besitzt das Modell die Fähigkeit, sich kontinuierlich weiterzuentwickeln und selbstständig zu aktualisieren, sobald neue Daten verfügbar werden.

Diese Arbeit untersucht die Wirksamkeit von Online Learning für vorhersagebasierte Anomalieerkennung auf Zeitreihendaten, welche Concept Drift beinhalten. Der Schwerpunkt dieser Untersuchung liegt auf der Genauigkeit, Laufzeit und dem Ressourcenverbrauch im Vergleich zu den konventionellen Modellen *SARIMA* und *Prophet*. Hierfür wurde ein Prototyp für Online Learning basierte Anomalieerkennung vorgestellt und mit der Leistung von gängigen Modellen verglichen. Die Experimente für diesen Vergleich basieren auf echten Daten aus dem IP-Backbone der Deutschen Telekom.

Die Messergebnisse verdeutlichen, dass der Online Learning-Ansatz in der Anomalieerkennung eine höhere Genauigkeit aufweist, insbesondere wenn Concept Drift in den Daten auftritt. Dieser Ansatz zeichnet sich durch eine überlegene Anpassungsfähigkeit aus, wohingegen Batch-trainierte Modelle nach signifikanten Veränderungen in den Daten nicht mehr in der Lage sind, passende Vorhersagen zu treffen. In statischen Datenumgebungen hingegen erweisen sich Batch-trainierte Modelle als leistungsfähiger. Die reduzierte CPU- und Speicherauslastung sowie kürzere Ausführungszeiten des Online-Verfahrens unterstreichen dessen erhöhte Effizienz.

Diese Thesis unterstreicht die Überlegenheit des Online-Learnings in der vorhersagebasierten Anomalieerkennung unter Concept Drift, mit vielversprechenden Einsatzmöglichkeiten in Echtzeitsystemen und sich wandelnden Umgebungen. Allerdings birgt dieser Ansatz Einschränkungen, welche Raum für weitere Forschung lassen. Zukünftige Arbeiten sollten den Fokus auf die Erkundung verschiedener Online-Lernalgorithmen und die Bewältigung von Herausforderungen im Bereich Online-MLOps, insbesondere in der Feinabstimmung von Hyperparametern legen. Die Differenzierung zwischen Anomalien und Concept Drift bleibt eine zentrale Herausforderung und regt zu weiteren Untersuchungen adaptiver Lernstrategien an.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ADWIN Adaptive Windowing

AR Autoregressive

ARIMA Autoregressive Integrated Moving Average

DL Deep Learning

DT Deutsche Telekom

FN False Negatives

FP False Positives

HTM Hierarchical Temporal Memory

IoT Internet of Things

IP Internet Protocol

IT Information Technology

KDE Kernel Density Estimation

LSTMs Long Short Term Memory Networks

MA Moving Average

MAE Mean Absolute Error

ML Machine Learning

MLOps ML Operations

MSE Mean Squared Error

NAB Numenta Anomaly Benchmark

PAD PredictiveAnomalyDetection

PoC Proof of Concept

RNN Recurrent Neural Network

SARIMA Seasonal Autoregressive Integrated Moving Average

TP True Positives

THESIS

1 INTRODUCTION

Modern Information Technology (IT) ecosystems rely on anomaly detection techniques for monitoring and fault identification to help gain insights into the system's state [2]. Due to ever-growing digitalization and the continuous development of new technologies, today's environments produce large amounts of data at very high speeds. A typical example of this phenomenon can be observed with the Internet of Things (IoT) [13]. There are various approaches to anomaly detection [1], but Machine Learning (ML)-based methods stand out as the most used in real-world use-cases (see [26] for example). Their ability to efficiently process and learn from large datasets leads to this widespread adoption. However, the general use of classical ML algorithms trained on large batches of data needs to be revised to work for today's dynamically changing systems. The primary concern is the phenomenon of concept drift, which occurs when the statistical properties of the predicted target variable change over time [29]. As a result, models trained on historical data batches may become outdated, leading to decreased accuracy and effectiveness in detecting outliers because of their inability to adapt to the changes in the data [12, p. 244]. There can be various reasons for such a concept drift, one of which can be a change in the configuration of an IT system, leading to different behavior.

1.1 PROBLEM DEFINITION AND MOTIVATION

These difficulties also occur at Deutsche Telekom (DT), Germany's largest Internet service provider, which maintains a vast Internet Protocol (IP) network of various segments. One of the most essential segments is the IP Backbone, the central and high-speed core infrastructure responsible for efficiently routing and forwarding data traffic between various network segments. It serves as the main highway for data transmission within the network. The Backbone of an IP network is a complex system comprised of numerous individual machines, such as routers and switches. Network administrators and developers must carefully configure each machine to work harmoniously with the others, forming a cohesive unit that efficiently routes data traffic. They perform many configurational changes manually, but the company wants to employ automation systems increasingly in the future. Due to current automation projects within DT, network configurations are becoming increasingly automated, and behavior inside the IP Backbone changes regularly. As a result, the underlying telemetry data, in the form of time series, can also experience frequent shifts in behavior. Consequently, traditional ML models' performance can deteriorate when forecasting [29, p. 1]. Prediction-based anomaly detection techniques suffer from this problem since they utilize a

model of normal behavior to make predictions about future behavior. These forecasts undergo comparison with the actual data, and if a significant deviation occurs, the algorithm declares the data point anomalous. [1, p. 276]. Strategies like this do not perform well if the underlying predictive model is flawed due to concept drift.

Different approaches to handling concept drift have been proposed in the past [16, 29]. Some of these solutions include detecting change points in affected data and consequent model retraining. While approaches like this can lead to satisfactory results, they are complex to implement and cost additional time and computing resources. The challenges presented by concept drift, particularly within the dynamic and evolving context of DT's IP Backbone, exhibit the limitations of using traditional batch-trained ML models for anomaly detection. There is a need for a robust and dynamic anomaly detection solution that is cheap and performant. Furthermore, for many monitoring and anomaly detection applications, it is preferable to process data online, one by one, to provide an instance-based service that is accessible in real-time.

In this context, online ML emerges as a potential solution. Unlike their batch-learning counterparts, online learning algorithms are designed to incrementally update and refine their models in response to the influence of new concepts in the data [39]. This continuous learning paradigm enables these algorithms to adapt to changing distributions in data, thereby ensuring the model's sustained precision. A model can continue to learn from new examples as they come by and does not require retraining. Since there is currently little effort in using online ML for anomaly detection inside DT, this thesis aims to leverage the features of online learning for predictive anomaly detection on time series data under concept drift to counter common problems of batch-trained ML models. A key area of investigation in this thesis is how online learning approaches perform against similar state-of-the-art methods in the face of concept drift. Furthermore, this research explores the theoretical potential and the real-world challenges and limitations of online learning in this context.

1.2 OBJECTIVE AND METHODOLOGY

This thesis proposes a solution for the previously described problem. On the one hand, there is the problem of concept drift adaptation for ML-based anomaly detection, and on the other hand, there is the need for fast and cheap real-time operation of these systems. Therefore, this work develops a concept that represents a promising solution for both objectives. Specifically, the proposal combines the existing ideas of prediction-based anomaly detection with online machine learning to create a more dynamic and robust solution.

Given the challenge of adapting ML models to concept drift in anomaly detection for time series data, this thesis aims to explore the following research questions:

RQ1: How accurate is the proposed online learning approach to prediction-based anomaly detection compared to state-of-the-art techniques on time series data under concept drift?

RQ2: How does the proposed approach perform compared to state-of-the-art techniques regarding computational efficiency and resource utilization?

RQ3: What are the specific challenges and limitations when using online ML for prediction-based anomaly detection on time series data under concept drift?

Answering these questions enables the reader to decide under which circumstances online learning-based approaches are preferred over batch ones.

To address these questions, this thesis presents the following contributions. The advantages and disadvantages of the proposed approach are worked out by conducting a literature review. This approach is implemented and evaluated in a Proof of Concept (PoC) based on the online learning library *River* [20] for Python, which is the language commonly employed for ML. The prototype for the use-case of DT, introduced in Chapter 4, utilizes an online learning variant of an Autoregressive Integrated Moving Average (ARIMA) [5] model to produce accurate forecasts for the rest of the anomaly detection algorithm. Additionally, the contributed module to the River project can accommodate any forecasting model.

A benchmark on actual company data from the IP Backbone monitoring is used to compare the proposed solution to similar prediction-based approaches employed in the company (e.g., Meta's *Prophet* [41]). The benchmark primarily evaluates the accuracy and overall performance of the models, providing a clear comparison of their effectiveness in real-world applications. Besides, additional benchmarks compare both time and resource consumption.

1.3 STRUCTURE OF THE THESIS

First, there will be an introduction to fundamental literature in Chapter 2, which investigates the current approaches to anomaly detection and time series forecasting. Similarly, literature on concept drift and online learning is discussed.

Chapter 3 lays the groundwork for developing the proposed prediction-based anomaly detection solution. This chapter explores the principles of

anomaly detection, offering an overview of its fundamental concepts and approaches. It further explores time series forecasting, highlighting the ARIMA model and clarifying its relevance and application in the predictive analysis of time series data. Besides, the chapter addresses the challenge of concept drift in dynamic environments and introduces online learning as a possible solution to this problem.

Subsequently, in Chapter 4, the thesis transitions from theoretical foundations to their practical realization. It is devoted to developing a PoC and outlining the design of the proposed online learning anomaly detector.

Finally, the benchmarks, the data used for the benchmarks, and the empirical results presented in Chapter 5 are discussed in Chapter 6.

In Chapter 7, the thesis summarizes the findings and resumes the discussion of the results from the previous chapter. This summary clarifies whether the proposed concept can offer added value to the supervising company and other researchers. It also examines prospective steps, possible improvements to the approach, and future research topics.

2

LITERATURE REVIEW

This section of the paper introduces the literature to investigate the problems with anomaly detection on time series under concept drift to help design a possible solution. The featured texts primarily focus on different anomaly detection techniques, time series forecasting, concept drift, and the basics of online ML.

Researchers have published many studies and proposed various approaches regarding anomaly detection. Two seminal works guide this exploration. Chandola et al. offer a comprehensive overview of the topic in their survey on anomaly detection [11], defining the different types of anomalies and scoring techniques for detection algorithms. They explain what it means to detect outliers in data and describe the challenges of this task. They also explore different application domains of anomaly detection. Aggarwal [1] provides an in-depth analysis of different outlier detection methodologies, setting a theoretical baseline for identifying anomalies. In his work, he explains that any ML model used for anomaly detection makes assumptions about the expected behavior of data and uses these expectations to evaluate if a newly seen data point is anomalous. He dedicates a chapter to detecting anomalies in time series data using regression-based forecasting models, an essential task for this thesis. More recently, the rise of Deep Learning (DL) has also introduced a paradigm shift in anomaly detection. Pang et al. [34] addressed major problems with anomaly detection that have not been solved yet, such as performance problems with high dimensional data or high false positive rates, and explained how DL approaches might help to solve those problems. Some DL-based methods try to learn a low-dimensional feature representation, like an Autoencoder, and score anomalies by looking at the reconstruction error of new data points. This approach is similar to prediction-based anomaly detection.

A critical use case for anomaly detection exists in the context of time series data. A core concept for this purpose is time series forecasting. Chris Chatfield describes the fundamentals of time series forecasting [12]. He goes even more profound and writes about topics like multiple regression models and concept drift, or how he calls it "structural breaks" [12, p. 244]. Other essential publications include the work of De Gooijer et al. [14] and Ahmed et al. [4]. One of the most frequently used methods for predicting time series data is ARIMA modeling, or Seasonal Autoregressive Integrated Moving Average (SARIMA) for time series containing seasonality. Box and Jenkins, among others, described this model type in their standard work [9]. Chapter 3 of this thesis details the use of ARIMA and SARIMA for making precise predictions

on time series. Multiple studies evaluate ARIMA models' forecasting performance and interpretability against that of deep neural networks. Zhang and Qi found that neural networks tend to be more accurate in their forecasts but also need more complex preprocessing [46]. There are also attempts to combine ARIMA models and neural networks and create a hybrid that profits from the strengths of both models [45]. In some of its current time series-related use cases, DT employs another model called Prophet, which Meta develops and maintains. Taylor and Letham wrote a paper on the motivation behind the model and its technical implementation [41].

As previously stated, time series play an important part in anomaly detection. Blázquez-García et al. conducted a review of different approaches to anomaly detection on time series specifically [8]. Besides dealing with the different types of anomalies to detect in time series, the authors also compare prediction-based models with estimation-based models for their specific use case. They mention incremental learning techniques for prediction-based detection and point out that there is little research on this technique, even though it has potential benefits. The study by Schmidl et al. [37] presents similar findings. They present a wide range of algorithms, which they compare in real-world and synthesized benchmarks, including datasets from the Numenta Anomaly Benchmark (NAB). These experiments show that there is no single superior algorithm for anomaly detection but that the choice of algorithm depends on the specific use case. Another key takeaway is that all tested methods had problems regarding their flexibility and robustness.

Several studies on more specific use cases are similar to the one addressed in this introduction. Some of these studies refer, for example, to anomaly detection in network traffic data [3, 32]. There is also a survey by Cook et al. on detecting outliers in monitoring data from IoT systems [13]. Use cases like this allow treating the underlying data as a time series. Cook et al. emphasize the problems with nonstationarity in real-world data and the technical limitations of low-powered computing resources. In such a complex application as the IoT, processing data online rather than in batch to ensure real-time recognition is preferable, so the authors propose using incremental learning algorithms as a cheap and reliable answer.

Next, it is interesting to take a closer look at actual implementations for anomaly detection using a prediction-based approach. Malhotra et al. published a paper on their approach to using Long Short Term Memory Networks (LSTMs) for anomaly detection [31]. Such a network is a special type of Recurrent Neural Network (RNN) good at learning long-term dependencies in sequential data, e.g., time series. They use an LSTM to learn the normal behavior of a series and make predictions, to use the prediction error as a metric to identify abnormal behavior. Similarly, Laptev et al. at Yahoo proposed a framework for prediction-based anomaly detection called *Extensible Generic Anomaly Detection System* [26]. This framework consists of three sep-

arate modules. The first module performs forecasting using a model type that best fits the given data in a plug-and-play manner. The second module scores anomalies based on the prediction error of the previous model and an automatically selected threshold. Finally, the last module deals with notifications for found anomalies. Munir et al. propose a similar solution called *DeepAnT* [33]. This implementation, also designed for unsupervised anomaly detection on time series, employs DL for making forecasts on future values of a time series. It uses a Convolutional Neural Network to produce a single prediction at a time, so it performs well on data streams. Another system that relies on DL for its prediction-based anomaly detection comes from computer vision. Liu et al. use Generative Adversarial Networks to synthetically generate the expected next image of a video and compare it to the actual subsequent frame captured by the camera [28]. As in previous cases, calculating an anomaly score uses the divergence of this comparison.

All implementations above use a conventional ML model trained on a batch of data. Hence, they are susceptible to concept drift, which deserves particular attention in any scenario dealing with a continuous data stream. Some researchers call this problem "AI aging" [43]. The review by Jie Lu et al. [29] examines the problem in detail, illustrates it by example, and suggests a way of detecting it. Similarly, the survey on concept drift adaptation by Gama et al. [16] deals with the different types of concept drift and suggests multiple ways to adapt. Besides simple approaches to concept drift adaption, such as regular retraining of a batch-trained model, Adaptive Windowing (ADWIN) is a concrete example of a method for dealing with concept drift suggested by Bifet and Gavalda [7]. It keeps a sliding window of variable size, and whenever two large enough subwindows have averages that deviate a certain amount, the algorithm flags a change in the data distribution.

Online learning is another possible solution to the problem of concept drift. Shalev-Shwartz et al. authored a paper examining online regression to forecast time series data [39]. Two other essential papers on the topic are the article on ML for streaming data by Gomes et al. [18], and the survey on online learning by Hoi et al. [22], which both discuss the necessity of online ML and concrete forms of its implementation. Online variants of ARIMA models are of particular interest for the intent of this thesis. The structure of the model stays the same as in a batch-learning scenario, but the way the model's parameters are updated changes gradually. A unique form of *Gradient Descent*, called Online Gradient Descent, is used for optimization, as described in [6]. Similar learning algorithms are used by Guo et al. [19]. Although they use LSTMs for time series prediction, the optimization steps are alike. Guo et al. go even further and propose a solution called "adaptive gradient learning," which makes the learning process robust to outliers but still able to adapt to new normal patterns in the data. Anomaly detection also requires a similarly robust optimization strategy like this. In online learning, the challenge of catastrophic forgetting, where models lose earlier

knowledge when learning new information, is significant. Kirkpatrick et al. address this in their work [24], proposing solutions inspired by mechanisms of the biological brain.

Previously mentioned studies on anomaly detection mainly refer to ML models trained on batches of data. The following section reviews research papers focusing on online learning algorithms in detail. More straightforward approaches to data containing concept drift include, for example, windowing procedures. These algorithms divide the data stream into regular sections of equal size, allowing each to be processed using standard anomaly detection methods. Salehi and Rashidi explain the application of this technique to different detection methods [35]. Some real-time anomaly detection algorithms also utilize decision trees. Tan et al. argue that there is currently little work on anomaly detection for evolving data streams and propose a method called *Half Space Trees* [40], which the River Library also implements. This approach is particularly performant and resource-efficient.

While these studies are primarily theoretical, some researchers use online ML technologies in real projects. Laxhammar and Falkman employed an online learning variant of a clustering approach for detecting anomalous trajectories [27]. They found that the online paradigm enabled them to detect anomalies even as the data evolved with time. Another option they evaluate is a regular schedule of retrainings on new batches of data. A study closely related to this thesis was written by Ahmad et al. [2]. The authors identify an increased availability of streaming data and a consecutive need for detection algorithms that can deal with the characteristics of such data. They suggest using Hierarchical Temporal Memory (HTM) to continuously learn the behavior of streaming time series data. The online nature of the HTM automatically handles changes in the underlying statistics of the data. The system models the prediction errors as a Gaussian distribution, allowing for comparing any new error against this distribution. This instance is flagged as anomalous if the error falls outside a certain threshold (on the tails of the Gaussian distribution). A procedure can only perform well if the forecast model fits the data well. The study found that algorithms learning online perform particularly well in the NAB. Moreover, Saurav et al. use RNNs for this type of prediction-based detection [36] while the core concept of their approach is similar to that of Ahmad et al. However, Saurav et al. focus on a specific problem of online learners dealing with anomalies in data, comparable to Guo et al. [19]. If there is a single disruption in the data, the model will also learn this anomalous data instance and update its weights accordingly. This behavior is desirable when the model needs to adapt to the new normal behavior of the series. However, in the case of single outliers, it reduces the model's overall performance because the algorithm tries to learn the anomaly as a new pattern behavior, possibly resulting in a dire prediction for the next data point. An appropriate learning rate can limit this problem since there is more normal data than anomalies, but it can not elim-

inate it. Not learning the anomalous data instances does not work because then the adaptation to drift fails. The researchers propose using a sliding window of the latest error values to dynamically calculate the learning rate for the model weights and benchmark their approach using the NAB.

The existing literature already contains the individual components of the solution proposed in this thesis. While the paragraphs above mention existing implementations of prediction-based anomaly detection and those using online learning, there has yet to be an attempt to combine this anomaly detection method with an online ML variant of ARIMA models to provide accurate detection for time series data under concept drift.

3

TECHNICAL BACKGROUND

This thesis requires a basic understanding of techniques for anomaly detection, time series forecasting, and online ML. Additionally, concept drift is analyzed, as it poses a substantial problem for anomaly detection in time series. This chapter is structured as follows: Section 3.1 introduces the basics of anomaly detection and focuses on anomaly detection for time series data. To better understand this specific application area, the basics of time series forecasting are outlined in 3.2. Forecasting time series using ARIMA models is essential here. Section 3.3 briefly introduces the problem of concept drift in the context of time series. Finally, Section 3.4 investigates online ML as a potential solution for the problem of concept drift. There is a thorough exploration of the differences in optimization algorithms between traditional learners and online learners to see how the actual process of learning works.

3.1 ANOMALY DETECTION

3.1.1 *Definition*

One of the most widely accepted definitions of what an anomaly is comes from Hawkins, who describes them as "[...] an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism" [21]. The term *mechanism* refers to the underlying system responsible for generating the data, suggesting that anomalies represent a fundamental shift or irregularity in the system's behavior. In other words, anomalies are patterns in data that do not conform to the normal behavior of that data but instead differ from it [11, 37]. They are also called outliers, discordants, or deviants [1, p. 1]. Anomaly detection is the task of finding such anomalous instances, which generally occur less frequently than normal ones, inside of data. It is essential across many industries, and research in this field has been ongoing for many decades. Application of anomaly detection ranges from intrusion detection in networks, over fraud detection, to usage in healthcare and many more [11, 34]. A specific application area for anomaly detection is the analysis of time series that occur in many places in the industry, e.g., as telemetry data of a monitoring system [2]. Section 3.2 goes into more detail on time series and forecasting, as this thesis is related explicitly to anomaly detection on univariate time series.

3.1.2 *Types of Anomalies in Time Series Data*

There are three main types of anomalies in sequential data [11]: point anomalies, contextual anomalies, and collective anomalies, sometimes also called subsequent anomalies [8].

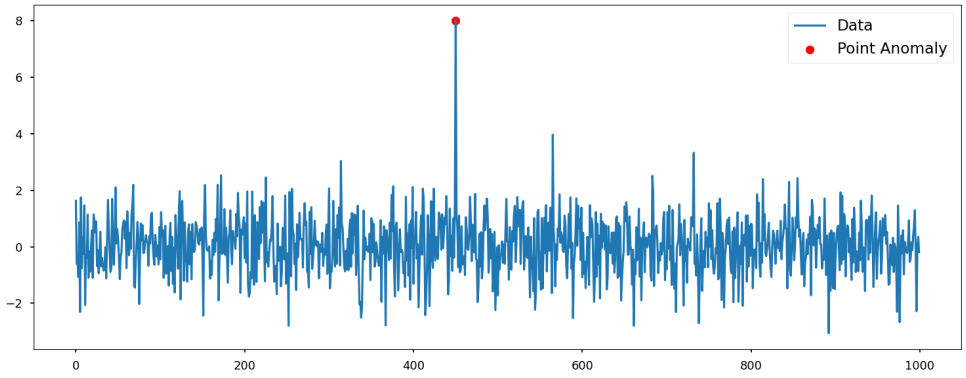

Figure 3.1: Point Anomaly in Random Noise

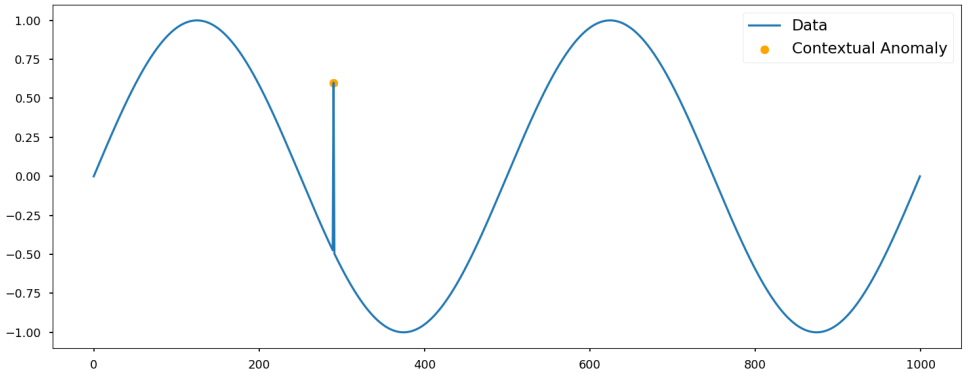

Figure 3.2: Contextual Anomaly in Sine Wave

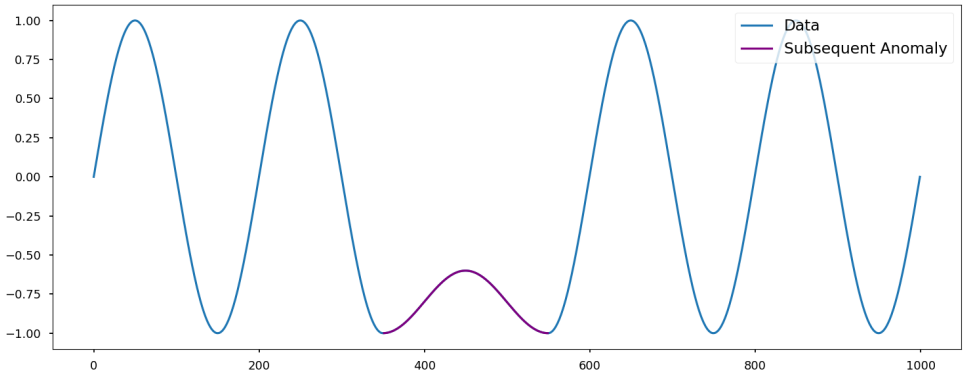

Figure 3.3: Subsequent Anomaly in Sine Wave

Point anomalies are individual data points that significantly deviate from most data in a time series, potentially indicating unusual or noteworthy events [8]. For example, in the series shown in Fig. 3.1, such an anomaly is recognizable as an upward outlier, further away from the rest of the data. Contextual anomalies "[...] are observations or sequences which deviate from the expected patterns within the time series, however, if taken in isolation they may be within the range of values expected for that signal" [13]. Fig. 3.2 shows an example of this type of outlier in a sine wave. While this instance could be part of the normal data, it behaves anomalously in the context of time. At last, collective anomalies, or subsequent anomalies, are groups of consecutive anomalous data instances. Single points of these groups may not be anomalous on their own, but seen as a subsequence, they are [13], see Fig. 3.3.

While the detection of point outliers is the most common task in anomaly detection, and most models specialize in this type of outlier, there are also ways to detect the other types of anomalies. However, they are often more complex [8].

3.1.3 *Methods for Detecting Anomalies*

Anomaly detection algorithms can be broadly categorized into three distinct settings: supervised, unsupervised, and semisupervised. Supervised algorithms work with labeled data. The dataset holds a binary label for each example to determine whether it is an anomaly. Traditional supervised methods for anomaly detection are classifiers that learn to distinguish regular instances from anomalous ones based on some given features. Obtaining labeled data is often infeasible since the labeling process has to be done manually in many cases, making it expensive and time-consuming [11]. Therefore, unsupervised anomaly detection techniques are more common. Unsupervised algorithms do not use labels for training. Instead, they rely on various assumptions regarding, for example, the data's structure, behavior, or distribution [37]. Semisupervised methods work with partially labeled data, but they are beyond the scope of this thesis. Another characteristic that distinguishes anomaly detection algorithms is their output, either binary labels or anomaly scores [1, p. 2]. While binary labels enable direct evaluation of the model's accuracy if the actual labels are known, using outlier scores is more common. The transition from scalar scores to binary labels can be done using thresholds, but that is a separate problem [37].

There are different concrete approaches to unsupervised anomaly detection. More straightforward methods use statistical metrics to detect anomalies, and more complex systems rely on ML or DL. Most anomaly detection algorithms work based on an assumption about the normal behavior of the data, and the difficulty behind detection lies in defining such a normal behavior [11]. Aggarwal wrote that "[...] all outlier detection algorithms create a model

of the normal patterns in the data, and then compute an outlier score of a given data point on the basis of the deviations from these patterns" [1, p. 5].

The most basic forms of outlier detection on sequential data assume that statistical properties like mean and variance are constant over time. This premise makes it easy to manually set high and low thresholds and declare data instances outside these thresholds anomalous [13]. Suppose the data is assumed to be normally distributed. In that case, it is viable to look at the distribution's tails and flag everything beyond three standard deviations (σ) away from the mean as anomalous, also known as the "3σ rule" [11, p. 30]. Aggarwal describes using the *Z-value*, the number of standard deviations a data point is distant from the mean, as an anomaly score and uses the threshold of three to declare a point anomalous [1, p. 6]. This approach only works for point anomalies [1, p. 11], and if the approximation to the normal distribution is poor, the method generates insufficient results. Distance-based techniques assume that normal data is close to each other and anomalous instances are further away. Therefore, they use *Nearest Neighbor* algorithms to determine anomaly scores [11]. Clustering-based algorithms work similarly, but they group data into clusters and determine if newly seen data is closer to a normal or anomalous cluster [27]. Other approaches include reconstruction-based ones [37] using Autoencoder Networks [34] or tree-based algorithms like *Half Space Trees* [40].

The definition of outliers as values that deviate from expected behavior leads to the idea of prediction-based anomaly detection [8]. A well-chosen ML model can learn the normal behavior of a system [1, p. 65]. This model can then predict future behavior, which it considers normal. Comparing the prediction to the actual data point, known as the ground truth, the model can then calculate the anomaly score based on the difference between these two, called the error. Instances that deviate significantly are considered outliers [8]. This approach allows for the detection of all anomaly types. Simple examples in the literature include training linear regression models for predicting scalar values and using the errors of those predictions as an anomaly score [1, 13]. The precision of the underlying model directly correlates with the accuracy of such a detection algorithm [26]. Since different models make distinct assumptions about the data, choosing a suitable model is particularly important. If a model cannot represent the normal behavior, this leads to insufficient performance [1, p. 5].

3.2 TIME SERIES FORECASTING

3.2.1 *Definition*

Time series are sequences of data points recorded or measured at successive points in time, typically at regular intervals [12, p. 10]. The single observations that comprise a series are generally discrete values [9, p. 2]. There are two types of time series data: univariate and multivariate. In a univariate time series, there is a single observation for each point in time. On the other hand, in a multivariate time series, there are multiple observations of multiple variables at each timestamp [13].

Although these individual observations of a series might appear incoherent to the bare eye, machine learning algorithms can deduct information from them to predict future values [45]. A model is fitted to the data set and is then used to predict data points past the time range of the given training data. Predicting future values of a time series based on historical data and using statistical models to identify patterns and trends that inform these predictions is called forecasting [9, p. 2]. It is one of the most essential tasks in time series analysis. Most often, the utilized algorithm uses a combination of past values from the time series, and possibly other explanatory variables, to predict the series's upcoming h values, where h is called the lead time or forecasting horizon [12, p. 12]. Hence, Chatfield stated that "[...] forecasts generally depend on the future being like the past" [12, p. 16].

3.2.2 *Components of Time Series*

Time series data comprises several components. One of these components is a trend, which indicates overall movement, meaning an increase or a decrease over a prolonged interval. Another component is seasonality, reflecting regular, predictable patterns or cycles. Similar to seasonality are cyclic changes, which are fluctuations without a fixed period. In many forecasting scenarios, algorithms often disregard them. At last, there is irregular or random noise, representing unpredictable, sporadic deviations in the data [12, p. 22], [46]. Understanding these structural components of a time series is necessary to use statistical models for forecasting based on past values.

3.2.3 *ARIMA Models*

One of the most frequently used models for time series forecasting is the ARIMA model or variations of it [45]. Noted for its flexibility and decent performance, ARIMA is extensively used in diverse real-world scenarios, predicting future values as linear functions of past observations [46].

Initially, ARMA models, consisting of an Autoregressive (AR) and a Moving Average (MA) component, were proposed to forecast stationary time series

[9, p. 52]. A series is stationary if the mean, variance, and autocorrelation, the measure of the correlation of a time series with its past values, do not differ with time [13]. The AR part of the model captures the relationship between an observation and a specified number of lagged observations. Essentially, it assumes that the current value of the series is a function of its previous values:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \ldots + \phi_p X_{t-p} + \varepsilon_t. \tag{3.1}$$

X_t being the value of the time series at time t, the symbols $\phi_1, \phi_2, ..., \phi_p$, being the weight parameters that are optimized in the learning process, and ε_t being the error term at time t. The variable p is called the order of the AR process [9, p. 52]. The MA part, on the other hand, allows the modeling of the error term, or residual, as a linear combination of error terms occurring at various times in the past:

$$X_t = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q}. \tag{3.2}$$

The symbols $\theta_1, \theta_2, ..., \theta_q$, are the respective learnable parameters and the variable q is called the order of the MA process [9, p. 53]. The combined model is defined as $ARMA(p, q)$, where p and q denote the order of both components. In the real world, time series are not always stationary though [13]. For this purpose, an ARMA model extends an additional "Integrated" component to become an ARIMA model. The new component involves differencing the time series data to make it stationary [9, p. 92]. A complete ARIMA model is now defined as $ARIMA(p, d, q)$ with d describing the order of differencing [9, p. 94]. At its core, it is a simple linear regression model that includes the AR and MA components to predict future points in the series. The model learns the respective coefficients, which are weights, using an optimization algorithm [45]. Furthermore, seasonal ARIMA models, called SARIMA, extend ARIMA by supporting seasonal components [9, p. 305]. Box and Jenkins developed a practical ARIMA model-building approach focusing on the autocorrelation structures within time series because finding the proper order of the different components is challenging. Their methodology unfolds in a three-step process for optimal parameter selection [9, pp. 177–178], [45]. The process begins with model identification, leveraging autocorrelation and partial autocorrelation to determine the order of the ARIMA model. Partial autocorrelation refers to the correlation between a variable and its lag, excluding the influence of correlations at shorter lags [9, p. 64]. The next step is parameter estimation, which involves ensuring stationarity through differencing and choosing parameters ϕ and θ that minimize error measures. The final stage is diagnostic checking, verifying the model's accuracy and reliability, leading to repeating steps one and two if the result is unsatisfactory.

3.3 CONCEPT DRIFT

3.3.1 *Definition*

Concept drift refers to a phenomenon where the statistical properties of a target variable, which an ML model aims to predict, undergo unexpected changes over time. More precisely, this means there is a change of joint probability of input X and output y at time t, denoted by $P_t(X, y)$ [29]. Distributions can evolve, especially in dynamic data-producing environments that change over time for various reasons, such as hidden changes to the underlying configuration [16]. Time series data treated as a stream tends to exhibit this changing behavior [19]. Concept drift, sometimes also referred to as *structural breaks* [12, p. 244], is one of the most important problems in modern ML [7]. The knowledge the model learned from previous data no longer applies to new data, resulting in suboptimal predictions [29, 43]. Addressing concept drift requires strategic approaches, including detecting change occurrences, data retention or exclusion decisions, and revising current models to adapt to new data patterns [7].

3.3.2 *Types of Concept Drift*

The joint probability $P_t(X, y)$ is the product of the probability of X, denoted as $P_t(X)$, and the probability of the output y given the input X, represented as $P_t(y|X)$. Hence, there is a distinction between two different kinds of concept drift: *real* and *virtual*. "The real concept drift refers to changes in the conditional distribution of the output (i.e., the target variable) given the input (input features) while the distribution of the input may stay unchanged" [16]. Virtual drift, or data drift, on the other hand, refers to a change of $P_t(X)$ only, which does not affect a model's predictions [29]. Real concept drift can occur in different ways. *Sudden* drift occurs abruptly, creating a more apparent boundary between the data distributions before and after the change point. In contrast, *incremental* drift unfolds over a more extended period. It involves a slow transition from one data distribution to another. *Gradual* drift slowly replaces old concepts step by step. Another common occurrence in scenarios with seasonal patterns is *reoccurring* concept drift, characterized by the cyclical appearance and disappearance of specific data trends [16, 29].

To illustrate the idea of concept drift and its effect on batch-trained ML models, Fig. 3.4 shows a synthetic time series that contains a sudden structural break. Before the change point, marked in yellow, the data approximates a sine wave with some random noise. After the change point, it is just random noise with a mean and variance different than before. When fitting a SARIMA model to a part of the data before the change and looking at its forecasts on the whole series, it is visible how the model keeps predicting the pattern it learned before the drift, resulting in higher errors afterward. Therefore, its performance suffers.

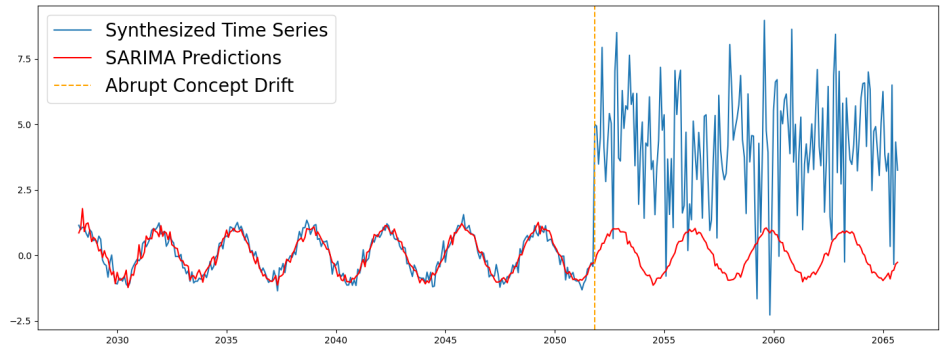

Figure 3.4: Forecasting on Time Series with Abrupt Concept Drift

3.3.3 *Solution Approaches*

Since the effects on performance are not tolerable for most use cases, data scientists developed ways to adapt to this behavior. A straightforward way to do this is to retrain the model on new data regularly [43]. However, this approach raises the question of when to retrain a model. While doing so on a fixed schedule can work for some use cases, another approach is to retrain a model dynamically whenever concept drift is detected. Techniques for this task include error rate-based detection and distribution-based detection [16, 29]. A widespread implementation that monitors distributions between two dynamically sized windows, or sub-series, is ADWIN [7]. It also allows for the preservation of more recent data for model retraining.

In addition to the conventional method of retraining, techniques such as online learning enable ML models to learn from data one example at a time and adapt to changes in underlying distribution [16, 29]. These approaches, discussed extensively over the years, are pivotal in maintaining the relevance and effectiveness of ML models in dynamic settings [43]. In the following section, the paper examines the basics of this incremental learning process.

3.4 ONLINE MACHINE LEARNING

3.4.1 *Definition*

To address the problem of concept drift, ML models need to keep learning from new data and adjust their parameters and weights accordingly. They update themselves based on the new distribution of the data. [29]. A continual learning process like this is called online learning or incremental learning. The models update by processing individual instances from a data stream sequentially, one element at a time. A stream constitutes a series of individual elements, each representing a collection of features known as a sample. This technique directly contrasts traditional batch-trained ML models, which learn from large datasets that must be available at the beginning of training. On the other hand, online learners can operate without having all the data available right away [16]. However, single-instance processing has the downside of bad scaling to big data since optimization algorithms cannot use the advantages of vectorization [20]. Most online learners assume that the most recent data holds the most significant relevance for current predictions and that a data instance's importance diminishes with age. Therefore, *single example models* store only one example at a time in memory and learn from that example in an error-driven way. They cannot use old examples later in the learning process [16]. While online learning algorithms usually do not have an explicit forgetting mechanism, like *abrupt forgetting* or *gradual forgetting*, they can still forget old information because the model's parameters update in a way that overwrites or dilutes the knowledge it previously acquired. This dilution can still lead to a significant challenge in online learning called catastrophic forgetting. In this phenomenon, a shift in tasks or concepts occurs unpredictably, and learning new information leads to a sudden loss of knowledge about previously learned patterns. Especially in environments prone to drift, catastrophic forgetting is a problem for models that continually learn from data streams. Methods to deal with this include constraining the updates of the model during new task learning using regularization techniques or dynamically selected learning rates [24].

3.4.2 *Optimization for Online Learners*

To understand how online models learn, it is necessary to understand how traditional optimization algorithms work. A straightforward example showcasing the inner workings of such an algorithm is a simple linear regression model using *Gradient Descent* as an optimizer.

The goal of a regression problem is to predict a continuous output value based on a set of input features. The most simple algorithm for regression problems is linear regression, a method for modeling the relationship between a dependent variable and one (univariate) or more (multivariate) independent variables [42]. It is called "linear" because the model assumes that

the relationship between the variables is linear, meaning that the change in the dependent variable is directly proportional to the changes in the independent variables. The mathematical equation for such a model with only one independent variable is denoted as follows:

$$y_i = \beta_0 + \beta_1 x_i. \tag{3.3}$$

The variables β_0 and β_1 are the learnable parameters of the model, where β_0 ist the intercept of the line and the y-axis and β_1 is the slope of the line [42]. Training happens by fitting the model to the available data, using a method such as least squares optimization, or by solving a system of linear equations. At its core, training means fitting this straight line to data. A resulting model could look like the following Fig. 3.5.

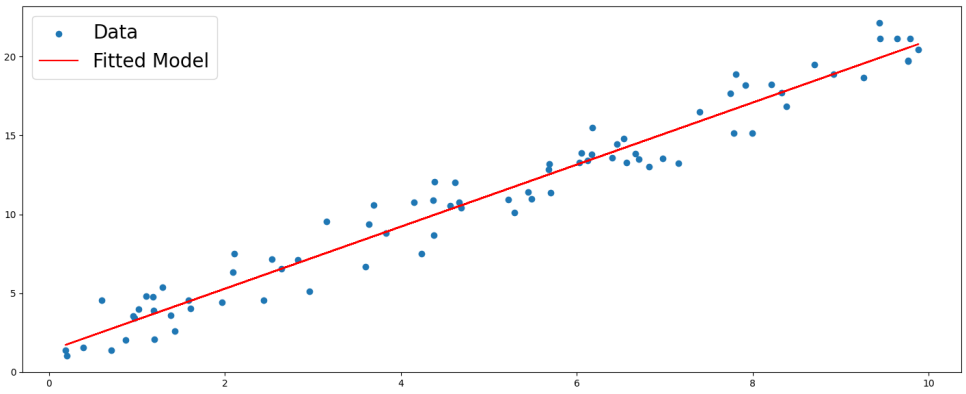

Figure 3.5: Linear Model Fitted to Data

The main difference between linear and non-linear regression is the form of the relationship between the dependent and independent variables. The relationship is more complex in non-linear regression and may not be a straight line. The equation is a polynomial of degree n:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_n x_i^n. \tag{3.4}$$

Overall, the choice between linear and non-linear regression depends on the nature of the data. Linear regression is a good starting point for modeling superficial relationships, but non-linear regression may be necessary for more complex data. Fig.3.6 shows an example of a non-linear model.

Although training can happen using least squares optimization, a more popular algorithm called Gradient Descent is commonly used in training machine learning models. It is an iterative algorithm that seeks to find "[...] the parameters which minimize a mathematical function" [23], a classical optimization problem. The function to optimize is the loss function of the model. Since the loss function is a complex, high-dimensional function, in most cases, it is not feasible to calculate the global minimum of it directly. The loss function determines how good or bad the model is. It determines the difference or error between a model's predicted output and the given

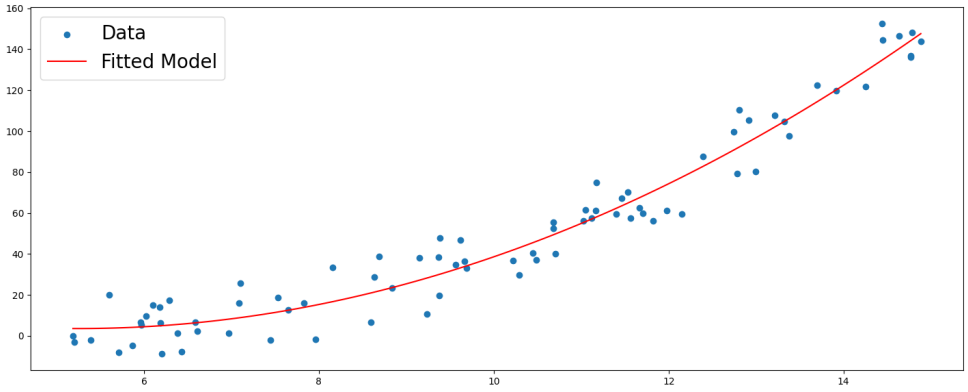

Figure 3.6: Non-Linear Model Fitted to Data

data's actual value. Choosing the proper loss function is essential to reflect the model's goals. Given the loss function $L(x)$, the input variables x need to be tweaked in a way that $L(x)$ is minimized [23]. The algorithm achieves this by making minor adjustments to the function parameters in the direction of the negative gradient. The gradient of a function is "[...] a vector of partial derivatives" [10, p. 8], a derivative $f'(x)$ being a function that describes how fast the function $f(x)$ grows or decreases at a given point x. A partial derivative is simply the derivative of a function focusing only on one of the input variables and considering the other inputs as constants. The partial derivative of a function f with respect to an input variable x would be denoted as:

$$\frac{\partial f}{\partial x}. \tag{3.5}$$

It describes the rate of change for the function with respect to the input x. So, the gradient is a vector that points toward the most significant rate of increase of the function and is used to determine how the function's inputs have to be tweaked to minimize or maximize $f(x)$. It is represented as:

$$\nabla f(x). \tag{3.6}$$

To find the minimum of the function, the algorithm takes steps in the opposite direction of the gradient, as this will move it toward the minimum. Calculating such a step downhill involves subtracting the gradient of the loss function $\nabla_x L(x_{i-1})$ from the input variables' vector x_{i-1}. The subscript x indicates that the gradient is taken with respect to the input variables. The resulting formula for calculating this update can be described as:

$$x_i = x_{i-1} - \alpha \nabla_x L(x_{i-1}). \tag{3.7}$$

The parameter α is called the learning rate. It describes the step size taken in the opposite direction of the gradient. This whole concept is more accessible when looking at it graphically (Fig. 3.7). The loss function for this example, denoted as $f(x)$, is straightforward. There is only one input, mak-

ing it a one-dimensional problem. Real-world problems can have thousands of inputs, but these high-dimensional problems are hard to visualize. The learnable parameter is initialized randomly at the beginning of the Gradient Descent algorithm. After that, the algorithm finds the corresponding gradient to take a step in the opposite direction, depicted in Fig. 3.7 by the red arrow. This process repeats until the loss function reaches a minimum and the gradient is near zero. For the example shown in the figure, the algorithm will most likely converge to the local minimum at $x \approx 1.3$.

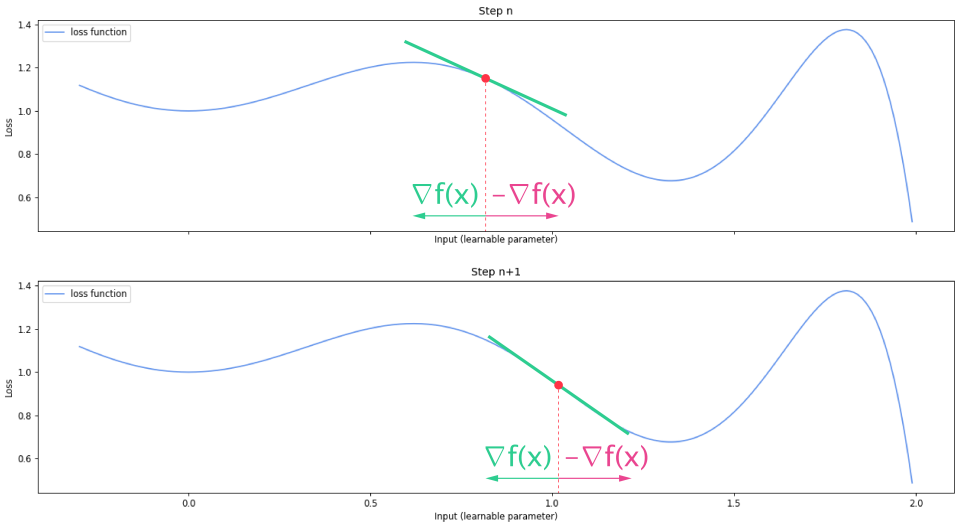

Figure 3.7: Gradient Descent Steps Visualized

Several problems may come with Gradient Descent, like overfitting the given data. Overfitting means that a model "[...] performs well on the training data, but it does not generalize well" [17]. In this case, the model memorizes the examples it is shown instead of learning a pattern behind them. Another one of those problems is a local minimum, a "[...] suboptimal solution [...]" [23, p. 116] for the loss function in which the iterative algorithm may get stuck. Since the gradient of such a point is zero, the algorithm may get stuck here. The most crucial parameter to tweak concerning these problems is the learning rate. If it is too small, the algorithm will take a long time to converge, but if it is too large, it may overstep the minimum and diverge. Finding a reasonable learning rate is often a matter of trial and error and can require careful tuning. Today, there are many variations of Gradient Descent, all of which try to bypass the abovementioned problems. Some more popular variations are *Momentum*, *Adagrad*, *RMSProp*, and *Adam* [23]. Examining these variations further would go beyond the scope of this thesis. Traditional Gradient Descent uses the entire dataset to compute the gradient at each step. This approach is expensive, but in exchange, the approach guarantees the most accurate update direction [23].

Unlike traditional Gradient Descent, which uses the entire dataset to compute the gradient at each step, Stochastic Gradient Descent uses a random sample of the data, or a mini-batch, to compute the gradient, which has several benefits. First, it allows the algorithm to scale to large datasets that may not fit in memory. Second, it introduces randomness into the process, which can help the algorithm escape from local minima and converge to the global minimum [23].

Finally, using Gradient Descent for online learning requires another variation, called Online Gradient Descent [22]. It involves using only a single example at a time. The model will be trained incrementally with every new example it sees. This method is a good choice if the model should learn from a data stream instead of a batch of data. The algorithm is designed for efficiency in its learning steps, enabling it to quickly and cost-effectively adapt to new data as it becomes available. This feature facilitates on-the-fly learning, accommodating incoming data in real-time [17, p. 16]. It is relatively cheap compared to training on the whole batch, but the update direction will be less precise, which leads to slower or no convergence. However, this circumstance can be good since the model may not get caught in a local minimum as quickly or overfit the training data [23]. As seen in the first sections of this chapter, online learning is a special type of ML in which models learn from streams of data instead of batches, which means that the model can learn and adapt to new data in real-time without retraining the entire model from scratch on the entire dataset. An online model does so by deducting information from the data iteratively, where it is provided the input data, along with the ground truth in each iteration [39]. It makes a prediction and compares it to the ground truth, measuring the error or loss. Using an optimization algorithm, such as Online Gradient Descent, the learnable parameters of the model are tweaked to minimize the loss. Consequently, online learning can handle datasets that tend to concept drift [22, 39].

4 CONCEPT AND IMPLEMENTATION

As stated in the introduction, one of the contributions of this thesis is to develop a solution for anomaly detection on time series data under concept drift and later compare it with state-of-the-art methods in different benchmarks. The concrete approach proposed in this chapter is prediction-based. While traditionally batch ML has often been used for this kind of application [26, 28, 31, 33], some implementations leverage online ML for training models and making predictions as well [2, 19, 36]. Even though these studies lay the groundwork for the new ideas explored in this chapter, a gap exists in online methods for anomaly detection in time series, especially in applying ARIMA models for forecasting. The following sections introduce the proposed solution's core concept, explain design decisions and implementation details, and discuss encountered challenges and their resolutions.

4.1 CONCEPTUAL FRAMEWORK

Based on the telemetry data from the IP Backbone monitoring systems, DT needs an accurate and robust way of detecting anomalies in those systems. Since there is no labeled data, the detection algorithm has to be unsupervised. The proposed solution adopts a prediction-based approach to anomaly detection in time series capable of generating anomaly scores. This type of anomaly detection operates on the principle that outliers manifest as high prediction errors in a model trained on the data's normal behavior [8]. Choosing the appropriate model to learn the normal behavior of the data is crucial, as an unsuitable choice results in insufficient predictions and, therefore, low detection accuracy. What is the best fitting model depends on the underlying data and associated assumptions [1, pp. 5–7]. Certain assumptions become relevant for telemetry data from the IP Backbone. As this is time series data, the order and autoregression of it are significant, leading to the conclusion of using ARIMA for building a normal model and forecasting expected future behavior [1, pp. 276–279].

A critical problem with this approach so far is handling concept drift. Real-world time series data, especially in dynamic environments like DT's IP Backbone, often exhibit nonstationarity and continuous evolution, resulting in changes in the underlying distribution of the data over time [13]. DT is currently working on several configuration automation projects, leading to environments changing even more rapidly, necessitating continuous adaptation of the ML models used in monitoring. Instead of manually retraining models on a regular schedule or dynamically whenever concept drift is detected, the proposed solution to this problem is utilizing online learning

instead of traditional batch ML. Some time series, like monitoring data, are continuous, meaning that online learning algorithms can address them as a data stream [18]. Online learning enables a model to adapt incrementally to the evolution of this stream, automatically adapting to changes in underlying statistics [2]. An online variant of ARIMA proposed by Anava et al. [6] can handle streamed time series data. It employs a particular form of Gradient Descent designed for continuous model training, which works like classic Gradient Descent but only processes one example simultaneously. The algorithm randomly initializes the coefficients ϕ and θ for the AR and MA components, so the model can make its first prediction $\hat{y}_t$ at iteration t. The model suffers a loss when it receives the ground truth value y_t. *ARMA Online Gradient Descent* aims to minimize the sum of these losses by tweaking the weights in the right direction, as shown in Chapter 3.4.

4.2 DESIGN AND IMPLEMENTATION

The open-source Python library *River* holds a diverse suite of existing tools and models for online learning. Examples include regression models, classification models, clustering algorithms, and forecasting models such as ARIMA's online variant mentioned above. Besides different ML models, the library also offers utilities such as pipelines, tools for hyperparameter tuning, evaluation, and feature engineering, to name a few, specifically designed for online learning [20]. Therefore, this thesis presents the proposed solution as an additional module for River, called *PredictiveAnomalyDetection (PAD)*[1], actively enhancing its already available range of features[2]. The module's design as a flexible framework aims to make prediction-based anomaly detection universally applicable across various applications. The underlying model for learning normal behavior is not set in the module but can be defined when initializing a new detector instance. This design adds versatility, allowing users to choose from various models available under the *estimator* class, which is the base class of different ML models in River and the placeholder for the model of normal behavior inside the presented module. For the problem stated in this thesis, the online ARIMA variant plugs into this framework to detect anomalies in time series data under concept drift. However, users can choose their base estimator depending on their use case and the characteristics of their data. Notably, while anomaly detection is an unsupervised task that does not require anomaly labels, the base estimator still needs labeled data for incremental training, e.g., timestamps and measurement values when working with time series.

The chosen design conceptually separates the modeling of expected behavior from the scoring process, similar to the approach used in [26]. The base estimator predicts the expected behavior of the data and compares it to the ac-

1 The code is in the official repository: https://github.com/online-ml/river/pull/1458.
2 The complete documentation for River can be found on their official website: https://riverml.xyz/latest/.

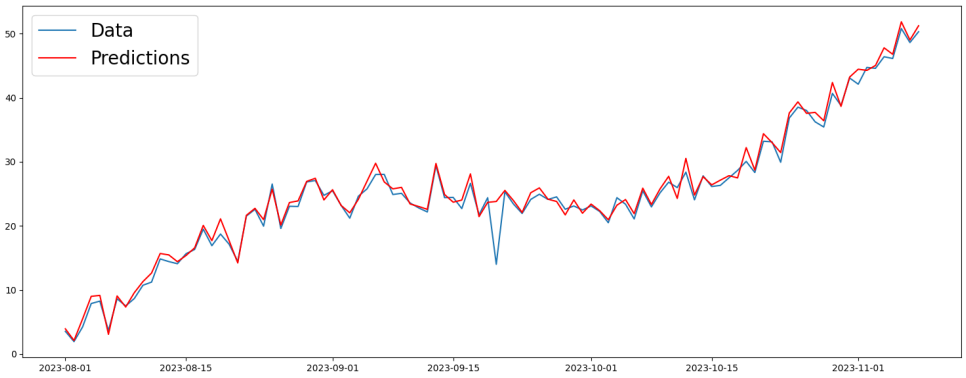

Figure 4.1: Predictions of Online ARIMA Model

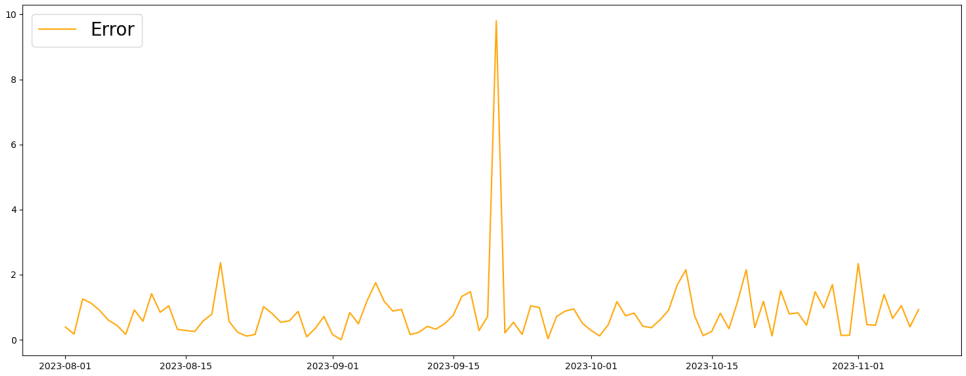

Figure 4.2: Error of Online ARIMA Model

tual value to calculate the error. The detection algorithm uses this error value, independently of the base estimator it came from, to calculate the anomaly score. Fig. 4.1 shows a cutout of forecasts made by an online ARIMA model on time series data. The model handled the series incrementally, so it made a forecast with a horizon of one and then got the ground truth value to update its weights accordingly. The data has a point outlier in it. Fig. 4.2 depicts the error the online model made on each forecast. It is visible that the error is most significant at the location of the point anomaly.

The scoring mechanism involves comparing the prediction with the ground truth, where deviation signifies error and, consequently, the score. The most straightforward idea is to use the error as the anomaly score directly, but usually, an anomaly score is a floating point value between 0 and 1.0. An assumption made in literature is that the error distribution, also called *error reference distribution*, is Gaussian [19, 26]. Based on this assumption, Aggarwal proposed to use the *Z-value*, the number of standard deviations a data point is distant from the mean, as an anomaly score [1, p. 73]. Others suggested laying a threshold on the error by declaring any value more than three standard deviations away from the mean (3σ rule) as anomalous [13]. These approaches also do not generate an anomaly score between 0 and 1.0,

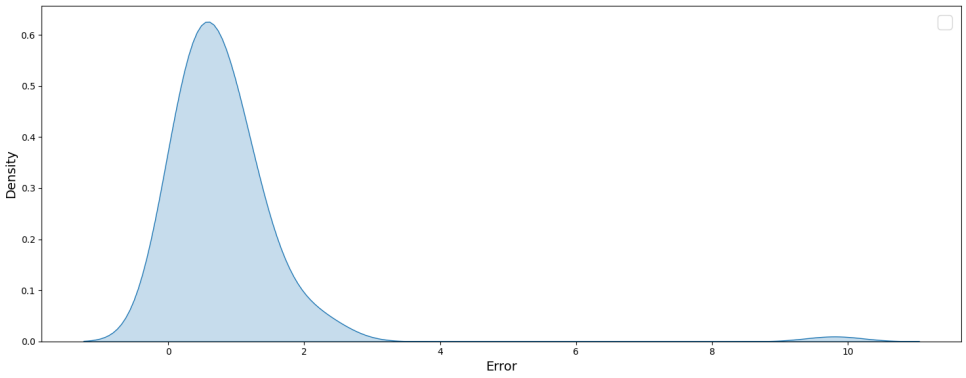

Figure 4.3: Error Distribution of Online ARIMA Model

but they inspired the solution used in the PAD module. This process entails keeping track of error values, including their mean μ and standard deviation σ, and dynamically updating these statistics as new data comes in [15]. After the base estimator makes a new prediction and calculates the new error ε, the scoring mechanism checks if the following equation is true:

$$\varepsilon \geq \mu + 3\sigma. \tag{4.1}$$

If the equation is valid, the mechanism sets the anomaly score to its maximum of 1.0. If not, the score gets linearly distributed between 0 and $0.\overline{99}$ accordingly. The user can set the coefficient of σ as a hyperparameter of the PAD module to adjust the sensitivity to anomalies. Fig. 4.3 presents the error distribution from the earlier example using a Kernel Density Estimation (KDE) plot. The KDE plot suggests that the errors approximate a normal distribution. Additionally, the influence of the point outlier is observable as a distinct feature in the right tail of the plot. The more accurate the model, the better the approximation of a normal distribution, resulting in more accurate anomaly detection.

4.3 CHALLENGES

During the development and testing phases, two main problems surfaced, highlighting the complexities inherent in implementing online anomaly detection. These difficulties are best illustrated using an example. For this purpose, Fig. 4.4 shows another time series and an online ARIMA model's forecasts. This time, the data depicts a sine wave with a sudden shift, simulating concept drift. In Fig. 4.5, the corresponding errors made by the model during forecasting are shown. One of the primary challenges visible in this Figure is the model's initial inaccuracy. At the outset, the model lacked sufficient data to make accurate predictions, resulting in high error values and erroneously high anomaly scores. A fix for this problem is incorporating a *warmup* period in which the model can acquire the necessary knowledge to identify anomalies effectively without outputting scores. The user can set the length

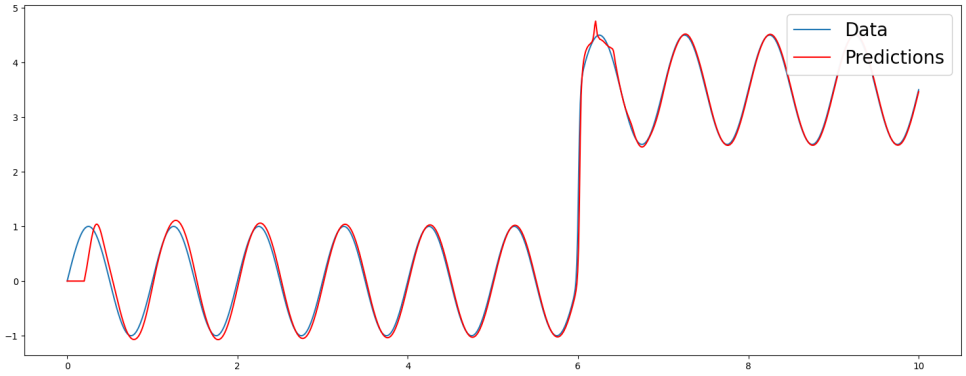

Figure 4.4: Forecasting on Time Series with Concept Drift

of this period as needed. Furthermore, it is hard for an online learning algorithm to distinct between an outlier and concept drift [1, p. 273]. An outlier and a structural break will both lead to the detector identifying an anomalous point since the boundary between the two is not strictly specified [19]. While an online learner's purpose is to be able to adapt to drift, learning incrementally also leads to the adaptation to anomalous data instances. Outlier observations can cause the model to deviate from standard patterns. The optimizer uses the residual from such an observation to perform a single online Gradient Descent step as it would with every error, tweaking the model weights to fit the anomalous data instances better. This model update distorts the error values following the anomalous point. This wave-like effect that any anomalous point can have on the error value is also visible in Fig. 4.5 at the location of the drift in the data. Subsequent regular observations will correct this behavior of the model, but it takes some iterations to recover fully [19]. Simply not learning outliers is the most straightforward solution discussed in some reviewed literature. Whenever an anomaly is detected, the model does not learn the instance. However, this approach also hinders its ability to adapt to concept drift. The problem is that an anomaly detection algorithm initially recognizes abrupt drift as an outlier. Usually, the model learns the new behavior and quickly adapts without problems. Guo et al. [19] explain that if a change point is misclassified and withheld from the model, subsequent data points will likely be incorrectly identified as outliers, possibly delaying or halting the learning of new distributions.

Such a termination to incremental learning is not expedient. Consequently, there is another approach to solving this issue. Methods like *Adaptive Gradient Learning*, proposed by Guo et al., dynamically adjust the learning rate depending on what type of outlier to expect. The learning rate is reduced when a new observation is a potential point anomaly since the model should not adapt to an anomalous example. Conversely, if it is likely a change point regarding concept drift, the learning rate is adjusted to quickly adapt to the new distribution [19, 36]. This solution minimizes the impact of point anomalies while maintaining the ability to handle concept drift gracefully

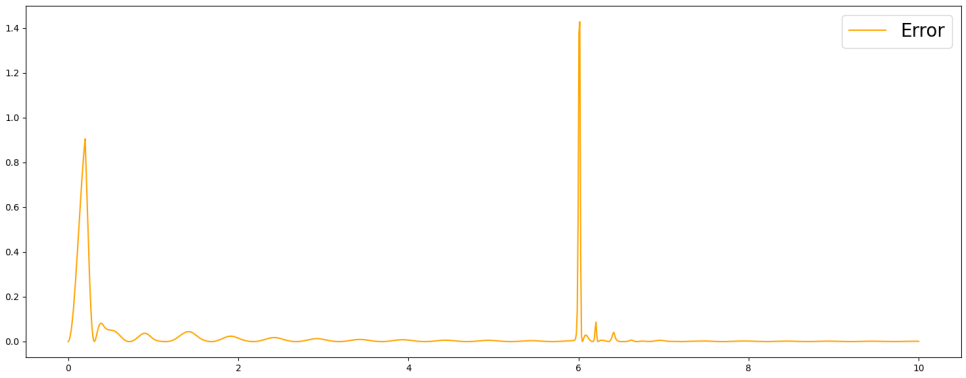

Figure 4.5: Errors of Forecasting under Concept Drift

and robustly. The researchers implement this by keeping a sliding window over the last n anomaly scores and calculating the so-called *suspicion ratio* [19], the ratio of suspicious points in the sliding window. If the suspicion ratio is high and there are a lot of large values in the last n anomaly scores, the probability that concept drift is present is significant. On the other hand, if the ratio is low, the current instance has a higher probability of being an outlier. The learning rate is then adjusted according to the suspicion ratio, positively correlated with the learning rate. River enables the dynamic adaptation of hyperparameters, such as the learning rate, by providing a function to mutate estimators.

5

EMPIRICAL FINDINGS

This chapter presents the empirical findings of the experiments conducted to answer the research questions outlined earlier in this thesis. There are a series of benchmarks to evaluate the proposed online learning approach for prediction-based anomaly detection. The first set of benchmarks examines the proposed method's detection accuracy and time series forecasting performance. It compares these against state-of-the-art techniques used inside DT. The subsequent benchmarks evaluate the computational efficiency regarding time expenditure and resource utilization[1].

5.1 EXPLORATORY DATA ANALYSIS

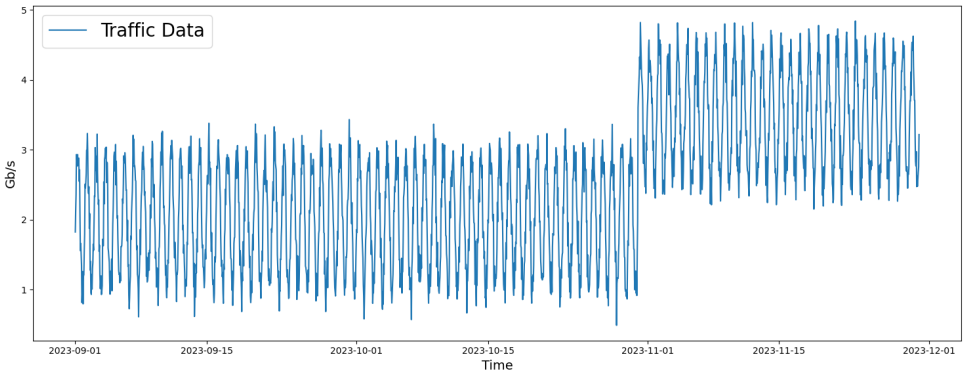

Figure 5.1: Traffic Data of Router in IP Backbone

Since the experimental analysis aims to provide realistic insights into the operational efficiency of the presented online learning solution and a realistic comparison to state-of-the-art batch learning techniques, the benchmarks resemble an actual use case based on historical data from the IP Backbone environment. Fig. 5.1 depicts the real-life traffic data utilized for the experiments. The monitoring system currently active at DT collects this data from a router inside the Backbone and makes it available with the help of *Elasticsearch*[2], a distributed search and analytics engine. The final dataset contains three months of data, with instances recorded hourly, resulting in 2160 individual data points. Each point resembles the hourly average incoming traffic of the router measured in gigabits per second. This data, predominantly clean and not excessively noisy, provides a realistic basis for the analysis.

1 The code for preparation, visualization, and benchmarking can be found in the repository: https://github.com/sebiwtt/OnlineVBatchExperiments/tree/669e.
2 See https://www.elastic.co/de/elasticsearch.

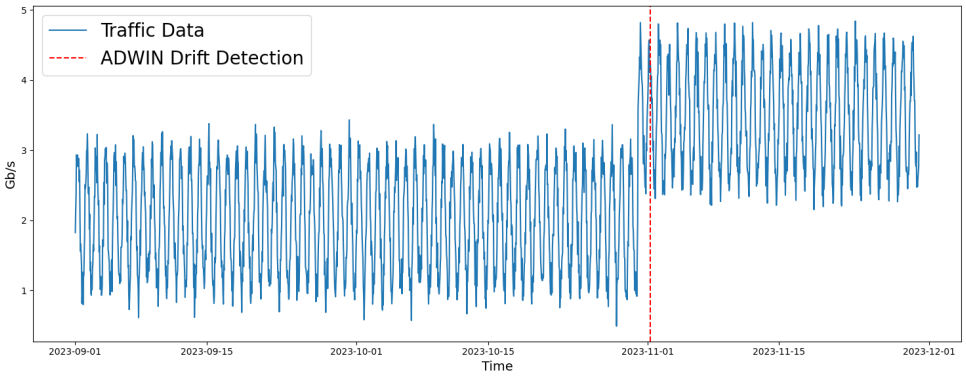

Figure 5.2: Concept Drift Detection using ADWIN

Yet, there is one apparent shift in the data approximately after two months. Drift detection algorithms like ADWIN identify this abrupt change in the data as concept drift, and Fig. 5.2 shows the index at which the algorithm detects the drift. It is hard to determine the exact source of this drift, but a configuration change likely resulted in more incoming traffic to the observed router. After the change, the mean traffic is higher than before, but the overall seasonal pattern stays the same.

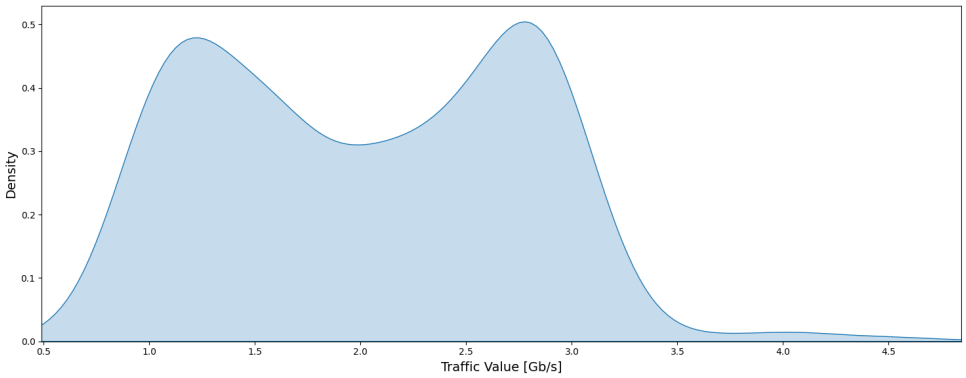

Figure 5.3: Distribution of Traffic Data before Drift

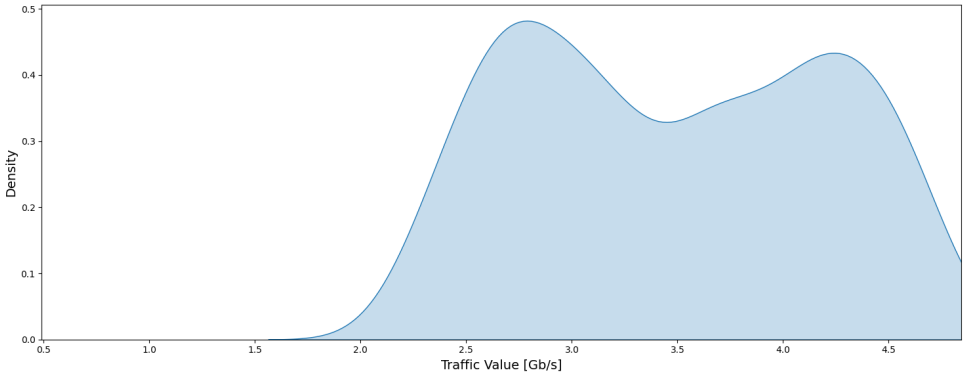

Figure 5.4: Distribution of Traffic Data after Drift

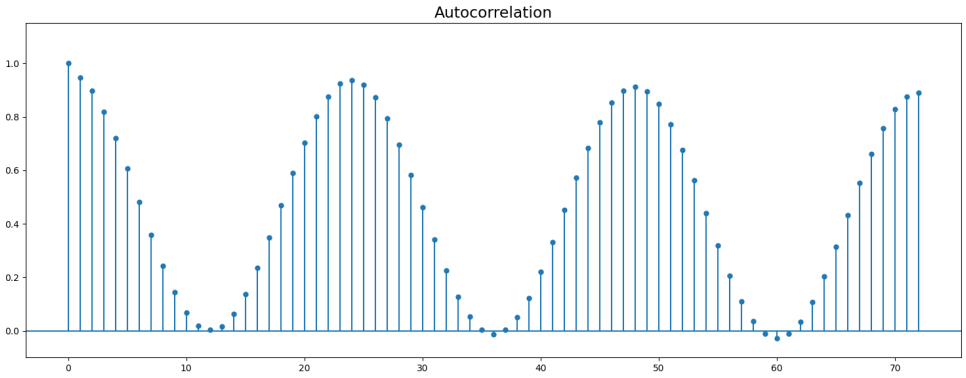

Figure 5.5: Autocorrelation Function of Traffic Data

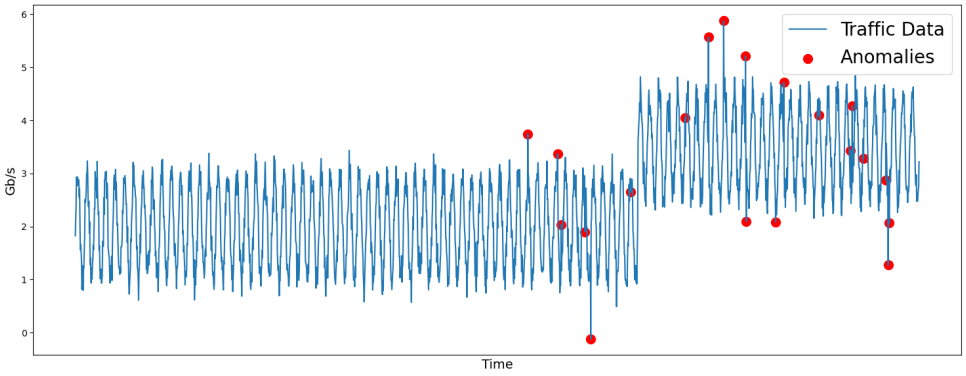

Figure 5.6: Traffic Data with Synthesized Anomalies

The KDE Plots in Fig. 5.3 and Fig. 5.4 illustrate the distribution before and after the concept drift making the shift in average traffic visible. Furthermore, it is necessary to analyze the seasonal behavior of the data for time series forecasting. A repeating pattern is visible when looking at the data. Such a pattern likely arises because the underlying system, the router, behaves similarly over a fixed period. In this case, the pattern repeats daily, meaning that the router goes through the same cycle once a day. The autocorrelation function plot in Fig. 5.5 confirms this assumption. It cycles with a peak every 24 lags. Since the data is recorded hourly, 24 instances correspond to one day.

5.2 EXPERIMENTAL SETTING

5.2.1 *Methodology*

To ensure unadulterated and comparable results, all of the benchmarks ran on the same dedicated virtual machine inside a docker container, with no other active processes to ensure encapsulation of the system running the simulation as much as possible.

Two benchmarks measure different performance metrics to answer the first research question of how accurate the proposed approach is compared to the baselines. The first experiment involves the time series forecasting performance of the different models. Forecasts of future behavior are the basis of prediction-based approaches, and more accurate prediction results in more accurate detection of anomalies [26]. The better the model can represent the normal behavior of the data, the greater the relative deviation between forecast and ground truth is when an anomaly occurs. Thus, the accuracy of the underlying time series model is directly proportional to the anomaly detection effectiveness. The first benchmark measures the Mean Absolute Error (MAE) and Mean Squared Error (MSE) to evaluate the forecasting performance [17, p. 44]. The MAE is the average of the absolute differences between predicted and actual values. The MSE, on the other hand, is the average of the squares of the differences between the predicted and actual values, giving more weight to larger errors.

The second test addresses anomaly detection accuracy, focusing on identifying point anomalies and contextual anomalies. Since the data is clean and no authentic anomaly labels are available, the benchmark relies on synthesized anomalies to measure the quality of the different algorithms. Fig. 5.6 displays a manipulated dataset version with synthesized anomalies in the latter third of the set. This way, the majority of the anomalies occur after the concept drift. The benchmark randomly selects 2% of data points in this interval for each run and turns them into anomalous instances. The data set now also contains binary labels that show whether a data point is normal or anomalous. The evaluation metrics include *precision*, *recall*, and the *F1 score* [17, pp. 93–95]. The precision is the ratio of correctly predicted positive observations, True Positives (TP), to the total predicted positives, TP and False Positives (FP):

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{5.1}$$

On the other hand, the recall is the ratio of correctly predicted positive observations (TP) to all the instances that are actually positive, TP, and False Negatives (FN). It is also called the true positive rate:

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{5.2}$$

Following, the F1 score is the harmonic mean of precision and recall, providing a single metric that balances both. It is often used when there is a class imbalance in the data, which is the case for this use case. The higher this value, the better the algorithm performed:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{5.3}$$

Using the plain accuracy measured in percent is infeasible in case of imbalanced data. Since only 2% of the data is anomalous, an algorithm always outputting an anomaly score of 0 would achieve an accuracy of 98% while missing all actual anomalies. Further, the algorithms need binary labels instead of anomaly scores to calculate these metrics. Accordingly, they apply a threshold to the scores to convert them to labels. However, thresholding is a problem of its own and out of the scope of this thesis [37]. Consequently, the thresholds for converting scores to labels are automatically determined to optimize each model's F1 score, assuring a fair comparison.

Next, the second research question deals with the proposed module's computational efficiency and resource utilization compared to the baselines. Subsequently, the third benchmark addresses resource consumption by running the models for a fixed period and keeping track of CPU and RAM utilization. Finally, the fourth and last experiment will examine the models' time consumption differences by measuring each model's time for training, making predictions, and calculating anomaly scores.

Each of the benchmarks discussed in this section performs its measurements 100 times. They then average their samples to provide precise outcomes.

5.2.2 *Software and Models*

This research incorporates two baseline models for comparative analysis, selected due to their current application for anomaly detection at DT. Hence, these models provide a realistic comparison for the proposed PAD module. The models used for the benchmarks are a batch SARIMA model from the *statsmodels* library[3] for Python and the *Prophet* model by Meta, presented earlier in this thesis [41]. These time series forecasting models can also serve as a foundation for prediction-based anomaly detection. Hyperparameter tuning for all three models occurred manually before the measurements to ensure the optimal operation of each model. The benchmarks measuring time and resource consumption do not consider the hyperparameter tuning to provide comparative results.

The first baseline is the traditional SARIMA model. It needs to train on a batch of data to do time series forecasting. Therefore, the dataset is divided into a training set containing the first 800 hours of data and a test set containing the rest. After the model fits the training data, it can predict future data. SARIMA outputs the predictions as a batch of data, which makes it unsuitable for online detection. The model stays static during the benchmark and does not undergo training with new data. Anomaly detection works similarly to the techniques proposed in the PAD module. A SARIMA model also outputs a confidence interval for its forecasts, quantifying the uncertainty in the predictions. This interval is the range in which future observations

3 See https://www.statsmodels.org/stable/index.html.

fall within a certain probability. Applications inside of DT use the forecasts and confidence intervals to calculate an anomaly score by measuring the distance of the ground truth from the upper or lower bound of the interval. If the ground truth lies outside this interval, the anomaly score is set to 1.0, and if it is inside, the score is linearly distributed between 0 and $0.\overline{99}$. Another challenge is finding optimal parameters for a SARIMA model, but since this topic is outside this thesis's scope, this section does not explore it. The model applied in the following benchmarks was found using *AutoArima* provided by the *pmdarima* library[4].

Prophet is the additional baseline. It is an alternative forecasting method to SARIMA and is vastly used at DT due to its speed and simplicity [41]. Like the first baseline, it follows a batch training approach, utilizing the initial 800 hours of data for training and the remaining dataset for evaluation. The calculation of anomaly scores works in the same way as it does for the batch SARIMA model.

Chapter 4 already introduced the new online learning-based approach for prediction-based anomaly detection called PAD. It stands out for its online prediction capability, continually updating parameters with incoming data. Therefore, training covers the entire dataset, yet performance metrics calculation begins from hour 801, ensuring fairness compared to the other models that only evaluate the test set. To handle the time series data of this use case, the underlying model of normal behavior for the PAD module is the online SARIMA variant discussed earlier [5]. Both the online and batch versions of the SARIMA model work with the same parameters to ensure a consistent basis for comparison.

5.3 RESULTS

Table 5.1 holds the result of the time series forecasting benchmark with the MAE and MSE values rounded to the fourth decimal place.

Model	**MAE**	**MSE**
PAD	0.1912	0.0366
SARIMA	0.8727	0.7616
Prophet	0.9140	0.8354

Table 5.1: Result of Forecasting Performance Benchmark

PAD performed better than SARIMA and Prophet, scoring a lower MAE and MSE value. The absolute error for the online model is 0.1912 on average, which, in the dataset context, means that the model's predictions of incoming traffic are off by 0.19 gigabit/s on average.

4 See https://pypi.org/project/pmdarima/.

Following, Table 5.2 displays the values of the precision, recall, and F1 score values also rounded to the fourth decimal place.

Model	F1 Score	Precision	Recall
PAD	0.7589	0.7762	0.7616
SARIMA	0.5970	0.9259	0.4444
Prophet	0.1772	0.5931	0.1684

Table 5.2: Result of Detection Accuracy Benchmark

Similar to the results from the first benchmark, PAD outperforms SARIMA and Prophet with an F1 score of 0.7589. Prophet performs considerably worse than SARIMA, and the SARIMA model scores the highest precision. The observation that precision and recall metrics for the PAD model are approximately equivalent is noteworthy. In contrast, the SARIMA and Prophet models exhibit a higher precision than their respective recall values.

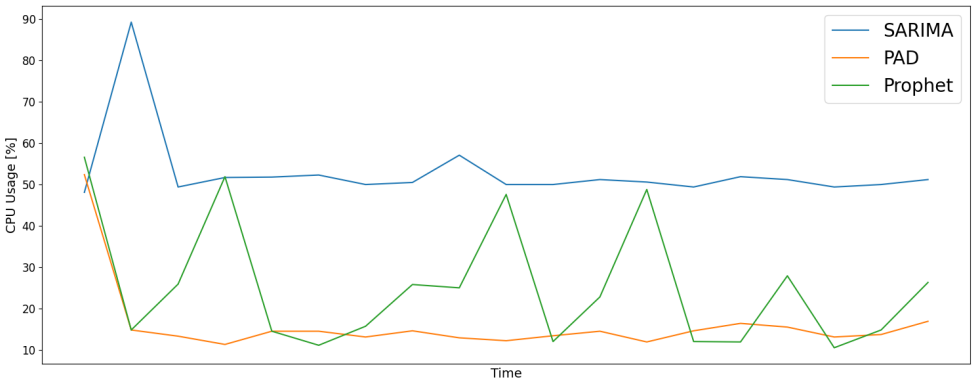

Figure 5.7: Result of CPU-Usage Benchmark

Fig. 5.7 shows the CPU usage of the models in comparison. It is visible that SARIMA and Prophet utilize the CPU more than PAD. On average, SARIMA occupied 53.42% of the CPU's power, whereas PAD used 15.79%, and Prophed averaged 25.04% but with a more uneven progression.

The plot in Fig. 5.8 visualizes the results of the memory usage benchmark. All three models have an even memory utilization. PAD occupies less of the system's memory than the SARIMA and Prophet models, which train on the batch of training data. The difference is relatively small. While PAD used 53.78% of the system's memory, SARIMA used 53.20% on average. Prophet uses the most memory with an average of 57.00%. The total numbers may vary depending on the underlying system, but the relative difference is noteworthy.

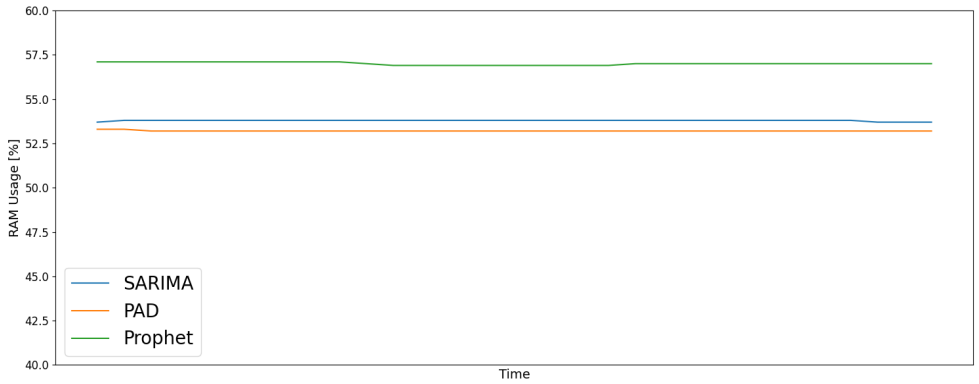

Figure 5.8: Result of RAM-Usage Benchmark

The last experiment dealt with the time spent training the models, forecasting future values, and calculating anomaly scores. Table 5.3 shows the mean values and variance of the recorded samples.

Model	**Mean Time**	**Variance**
PAD	200 ms	533 µs
SARIMA	1min 14s	559 ms
Prophet	466 ms	4.54 ms

Table 5.3: Result of Timing Benchmark

PAD showcased the fastest mean execution time at 200 milliseconds, with a low variance of 533 microseconds, indicating quick processing and consistency in its performance. Prophet demonstrated a moderate mean execution time of 466 milliseconds. On the other hand, using SARIMA resulted in a significantly longer mean execution time of 1 minute and 14 seconds, suggesting that while SARIMA provides thorough analysis, it does so at the cost of time efficiency[5]. On larger datasets batch-trained models may be able to perform better on this benchmark because they can make better use of vectorization.

5 While this disproportionately long training time seems off, numerous repetitions have confirmed this measured value. An issue was opened in the repository of the *statsmodels* library to investigate the reason for this behavior: https://github.com/statsmodels/statsmodels/issues/9153.

6 DISCUSSION

This thesis revolves around a series of research questions and objectives to explore the performance and resource consumption of online ML in prediction-based anomaly detection under conditions of concept drift. The RQs addressed were:

RQ1: How accurate is the proposed online learning approach to prediction-based anomaly detection compared to state-of-the-art techniques on time series data under concept drift?

RQ2: How does the proposed approach perform compared to state-of-the-art techniques regarding computational efficiency and resource utilization?

RQ3: What are the specific challenges and limitations when using online ML for prediction-based anomaly detection on time series data under concept drift?

The following sections will discuss the results obtained from the benchmarks, highlighting the strengths of the proposed approach in certain aspects while also acknowledging potential weaknesses of the model and the study itself. In addition, this chapter aims to answer the research questions posed at the outset comprehensively and explore the broader implications of the findings.

6.1 KEY FINDINGS

The first benchmark evaluated the forecasting performance of the different models by measuring their MAE and the MSE. These measures are essential in understanding the accuracy and reliability of the compared approaches

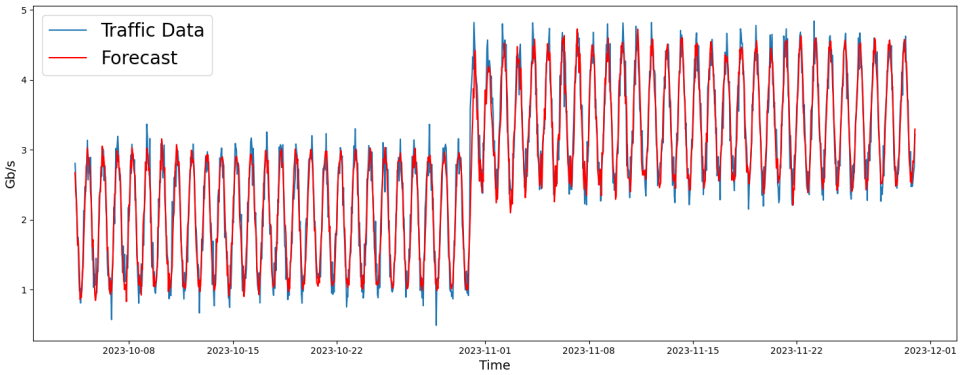

Figure 6.1: PAD Forecast on Traffic Data

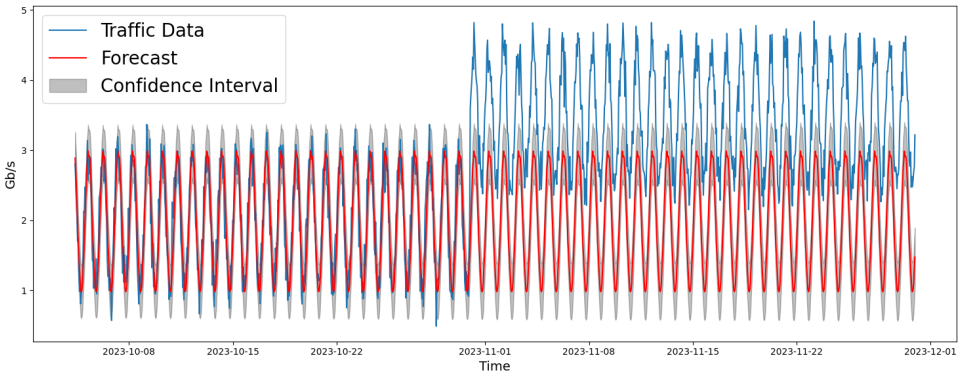

Figure 6.2: SARIMA Forecast on Traffic Data

in predicting anomalies under conditions of concept drift since the better a model can represent the regular patterns in data, the easier it can identify outliers. Section 3.4.2 argued that the online gradient descent steps are computed over a single example instead of a batch of data, making them cheaper to compute but less accurate. This knowledge leads to the assumption that a batch model will outperform an online model in this benchmark. However, both models operating on batches have a higher MAE and MSE value, indicating worse performance than the online learning competition. This circumstance is likely due to the online model's adaptation to the concept drift. In contrast, the batch models do not adjust and suffer significant losses because of the sudden break in the data. Another possibility may be that online gradient descent enabled the model to escape a local minimum that the batch model could not escape. So, in this case, the online model was superior because the concept drift in the data did not influence it as much as the batch model.

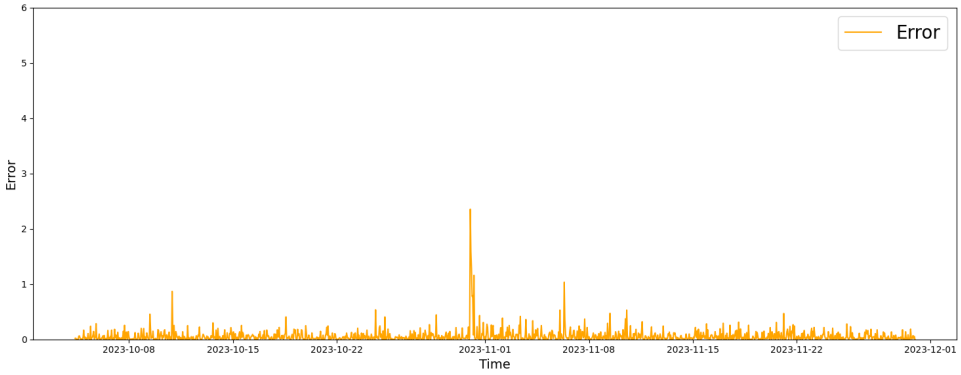

Figure 6.3: Error of PAD Forecast on Traffic Data

This assumption is confirmed when looking at the visualization of the forecasts from PAD in Fig. 6.1 and the SARIMA model in Fig 6.2. Since SARIMA is trained once and does not learn continuously, it cannot cope with the abrupt drift. On the other hand, the forecasting model of PAD adapts to

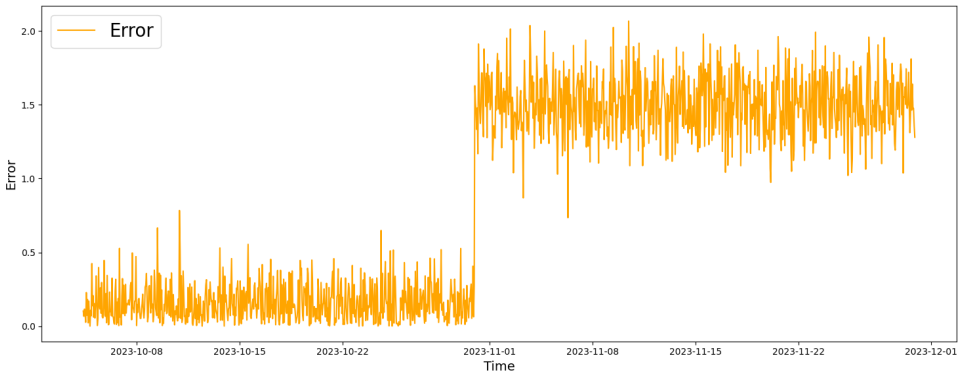

Figure 6.4: Error of SARIMA Forecast on Traffic Data

changes in the data fast. Therefore, the error values it produces remain stable with a single peak at the position of the abrupt change (Fig. 6.3), while the forecasts of both SARIMA (Fig. 6.4) and Prophet result in very high errors after the concept drift.

PAD also performs best regarding the F1 score. Its recall and precision values are close to each other. While the SARIMA model has the highest precision overall, both SARIMA and Prophet have a lower F1 score, as their precision is higher than their corresponding recall, indicating that the instances the models identify as positive are indeed positive most of the time. However, the models still miss many actual anomalies, leading to a higher number of false negatives. This circumstance is assumably due to the batch models' high errors after the concept drift and the automatic threshold set to optimize the F1 score. Table 6.1 shows the F1 scores of the models when calculating them before and after the concept drift independently, each with its own optimized threshold. The performance of PAD is relatively consistent before and after the drift happens. The batch models, on the other hand, show considerable differences between the scores before and after the drift. Before the abrupt change, SARIMA performed best, and Prophet performed better than in the previous benchmark. This behavior strengthens the suspicion of the problems batch models have with adapting to concept drift.

Model	**F1 Score before Drift**	**F1 Score after Drift**
PAD	0.8889	0.7826
SARIMA	0.9091	0.5000
Prophet	0.6000	0.3333

Table 6.1: F1 Scores before and after Drift

In light of these findings, it is evident that while batch learning methods like SARIMA excel in scenarios devoid of concept drift, their performance is inferior in the presence of such drift compared to online learning approaches like PAD. This discrepancy underscores the inherent limitations of batch learn-

ing methods in dynamically changing environments. Regarding the first research question, this leads to the conclusion that the proposed online learning approach demonstrates superior accuracy in prediction-based anomaly detection on time series data under concept drift compared to state-of-the-art techniques, affirming its effectiveness and robustness despite evolving conditions.

Though retraining batch models could theoretically address concept drift, it is unclear if an online learning approach is more efficient regarding resources and time. The second research question addresses this problem. The results of the CPU usage benchmark show that PAD requires the least computing power on average. In addition, the results of the memory usage benchmark also show that PAD allocates less RAM than the SARIMA and Prophet models. The difference is marginal, and all three models show an even memory consumption. The online gradient descent algorithm behind the forecasting model of PAD uses a single example of data to calculate the corresponding gradient and take a step in the direction that minimizes the loss function. Each example can be loaded into memory one at a time. Hence, the whole dataset does not occupy the memory simultaneously, as does the batch of training data for SARIMA and Prophet. Further supporting the efficiency of the proposed online learning approach, the timing benchmarks reveal that PAD consistently outperforms the batch models in terms of speed. The low computational cost of PAD's calculations ensures faster processing times. In stark contrast, the SARIMA model is significantly slower. It is necessary to remember that an online model must always be available to receive incoming data. Although it will run idle a lot of this time, it will still occupy some resources.

The faster processing speed, combined with the previously discussed advantages in CPU and RAM usage, underscores the superior overall efficiency of the PAD approach in managing resources. These results match the points discussed in section 3.4.2, answering the second research question. The individual optimization steps the online gradient descent algorithm performs are cheaper to compute, resulting in lower resource usage and faster execution time at the cost of slightly less accurate updates to learnable parameters [22, 23]. Consequently, this research highlights that online learning approaches offer a more robust and cost-effective solution in environments suffering from concept drift despite being marginally less accurate under stable conditions.

6.2 LIMITATIONS

The methodology employed in this study involves measurements and benchmarks performed using actual data collected from a router in the IP Backbone of DT, which reflects only a single specific use case. This data includes one type of concept drift and particular anomaly types, namely point and

contextual anomalies. Future research could address this limitation by exploring consecutive anomalies and using a predictive model-based approach along with longer forecasting horizons [8], which this thesis notably did not explore. Besides, the accuracy of prediction-based anomaly detection depends on the suitability of the underlying model to the use case and the data, emphasizing the need for precise tailoring. In this thesis, the focus on just one dataset and use case presents a challenge in terms of generalizability. As demonstrated by Schmidl et al. [37], who compared various algorithms across real-world and synthesized benchmarks, no single anomaly detection algorithm is universally superior. However, the specific characteristics of the use case to which an algorithm is applied determine its effectiveness. One assumption often made in the literature is that the distribution of forecasting errors approximates a Gaussian distribution, which would simplify detecting outliers [1, p. 73]. This assumption is not always valid and depends on the adequacy of the model for the given context. For example, the KDE plots in Fig. 6.5 and Fig. 6.6 show the corresponding error distributions of PAD and the SARIMA model. Using the error distribution of the PAD model, it is easier to distinguish outliers based on the prediction error.

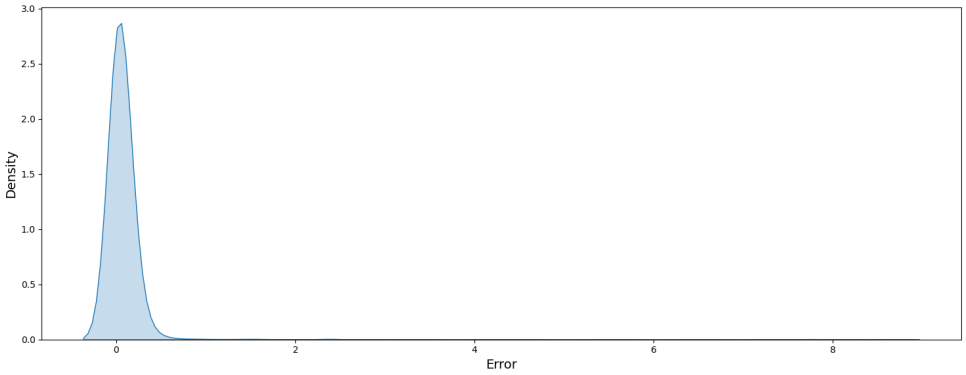

Figure 6.5: KDE Plot of PAD's Forecast Error

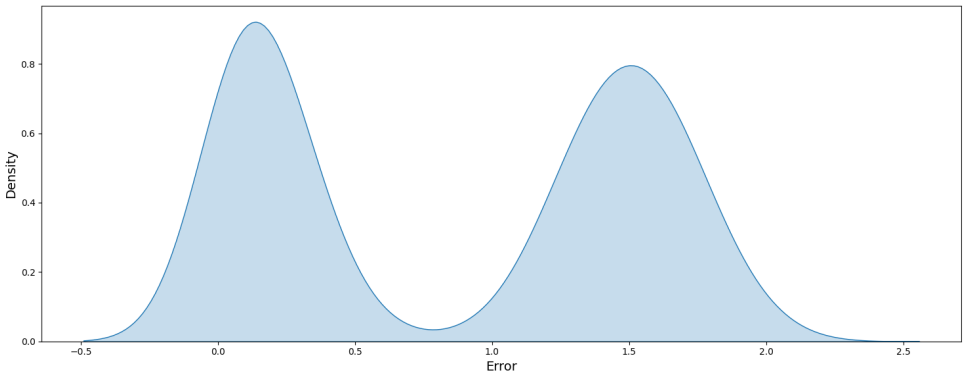

Figure 6.6: KDE Plot of SARIMA's Forecast Error

Hence, it is a better fit for this specific use case. Other datasets from other use cases might come with other explicit needs for a normal behavior model.

A further challenge regarding online learning models identified in this study is the issue of distinguishing between concept drift and outliers, which is particularly critical in anomaly detection, making them related problems [1, p. 23]. Abrupt changes may appear as anomalies, and gradual drift might not be as easily identifiable, making the line between drift and an anomaly blurry. While innovative, the Adaptive Gradient Learning approach by Guo et al. [19] presented in Chapter 4 is not infallible and requires thorough testing for its effectiveness in different scenarios. Guo et al. found that the concept of Adaptive Gradient Learning tends to be more effective when predicting multiple steps, but this necessitates multiple ground truth values, resulting in a delay in detection [19, 36]. In summary, the distinction between concept drift and outliers presents a complex challenge in anomaly detection, requiring careful consideration of the model's response to different types of drift and the potential introduction of specific tests and strategies to enhance the model's adaptability and accuracy.

Another crucial aspect not fully addressed in this study is the complication of hyperparameter tuning. The benchmarks did not consider this topic to provide a fair comparison of the time and resource consumption during training and inference only. In real-world applications, however, this is a crucial aspect of the ML lifecycle and is often covered by ML Operations (MLOps) practices [25, 30, 38]. While online learning provides a solution to concept drift, reducing the need for regular retraining often associated with MLOps, it introduces specific challenges that MLOps must address. Modern "parameter-laden" algorithms require delicate tuning because they can be susceptible to parameter settings, such as internal thresholds or the learning rate [27]. In online learning environments, this complexity intensifies because naive methods such as grid or random search, which leverage brute force, are not readily applicable [18]. Depending on the behavior of the underlying system, it may even be necessary to adapt the fundamental structure of a model. For example, changing any parameter p, d, q might be necessary in the case of an ARIMA model. MLOps is pivotal in addressing these challenges. It involves hyperparameter tuning, continuous model performance monitoring, the possibility of rollback to stable versions, and efficient deployment strategies. However, most MLOps frameworks neglect the unique challenges online learning presents for these practices and focus on classical batch learning setups.

These limitations also answer the third research question. The difficulties of using online learning models as a basis for prediction-based anomaly detection lie in selecting the appropriate model, thorough hyperparameter tuning, either manual or with the help of automatic processes, and the inherent challenges in differentiating between anomalies and concept drift.

6.3 IMPLICATIONS

The findings from this research have practical implications, particularly for industries that depend on real-time data analysis. The superiority of PAD in individual and specific settings is evident, offering an efficient solution for online anomaly detection if concept drift is present in the data. Using an online ML model as the base for prediction-based anomaly detection can be advantageous compared to a traditional batch-learning one. Unlike batch models, PAD's online learning capability makes it feasible for real-time applications and online processing of continuous data streams. Furthermore, PAD's proficiency in handling concept drift becomes a significant benefit in industries where data non-stationarity is a norm. Distributions and trends can change unpredictably in such environments, making traditional batch-learning methods less effective. The ability of PAD to adapt continuously to new data patterns without retraining makes it an invaluable tool in these dynamic settings. This distinction is crucial in industries where timely and accurate anomaly detection is essential, and it is not practicable to retrain larger batch-trained models regularly [16].

One notable application of PAD is in the field of IoT, where it is common for many devices to generate data continuously. Processing data online rather than in batches is essential for real-time applications. With its low resource consumption and rapid processing capability, PAD is well-suited to environments constrained by lower-end devices with limited computing power and storage capacity. As Cook et al. suggest, in complex applications like the IoT, incremental learning algorithms are a cost-effective and reliable solution for real-time data processing [13].

7

SUMMARY AND FUTURE WORK

The objective of the presented work was to investigate the potential of online ML for prediction-based anomaly detection for time series data, focusing on addressing the challenges posed by concept drift. Dealing with drift is relevant within dynamic and evolving landscapes, necessitating a robust, dynamic, and resource-efficient solution. Traditional batch-trained ML models often falter in such scenarios due to their inherent limitations in adapting to new patterns in the data. Online ML is a promising alternative, offering real-time, instance-based processing capabilities and the ability to adapt to concept drift. This study aimed to implement a PoC for prediction-based anomaly detection using online ML and compare it to established methods based on a use case provided by DT. Therefore, the thesis first conducted a literature review, which informed the development of the PoC in the form of the River module PAD. Previous research and implementations in the field influenced the development of the presented module [26, 28, 31, 33]. The subsequent steps included benchmarking and comparing the proposed solution to batch-trained SARIMA and Prophet, two techniques used at DT. The discussion then centers on the empirical findings from these experiments, outlining the key outcomes, limitations, and broader implications.

Online and batch learning are two different approaches to training ML models. First, batch learning uses the entire dataset or large batches of data to train a model. Once trained, the model becomes static and cannot adapt to structural breaks in the data. In the case of concept drift, retraining the model using more recent data is necessary. Conversely, online learning enables a model to learn incrementally using one example of data at a time, which makes the model flexible. This learning strategy can also help prediction-based anomaly detection algorithms, which use an ML model to learn the normal data pattern and predict future behavior. These algorithms determine if a new data instance is an anomaly by comparing the forecast to the actual data. The anomaly detection accuracy decreases if the base model is flawed due to concept drift. Hence, online learning models can fix this issue by offering adequate prediction performance, which is more robust to concept drift.

The proposed solution, PAD, allows for selecting different online learning models from the River library as the base model for learning normal behavior. For the benchmarks, an instance of PAD integrates online SARIMA as its core model for learning behavioral patterns in time series data. This approach is benchmarked against traditional batch-trained SARIMA and Prophet models, focusing on timing, CPU and RAM usage, time series fore-

casting performance, and anomaly detection accuracy. The evaluation is based on real-world data from DT's IP Backbone, ensuring a fair and practical comparison. Although the benchmarks show that PAD has better forecasting performance and anomaly detection accuracy overall, they also indicate that batch-trained SARIMA is the superior model if there is no concept drift in the data. The online learning-based method outperforms SARIMA and Prophet because of its quick adaption to the abrupt drift. It can adapt quickly and maintain a relatively small error. In contrast, the batch learning-based algorithms exhibit a significant error after the concept drift. Moreover, the benchmarks showed that batch learning is slower than online learning and requires more computing power and memory. In summary, the practical applications of online learning for prediction-based anomaly detection extend across industries requiring real-time data analysis and operation in environments with non-stationary data and limited computing resources, making it a good fit for the monitoring use case of DT. In general, the choice between online and batch learning methods heavily depends on the specific requirements of the problem. This research underlines the growing need for efficient, real-time, and adaptive machine learning solutions like PAD in the modern data-driven landscape.

Effectively, while the study offers insights into the specific scenario examined, it underscores the need for further research addressing the core limitations and implications discussed in the last chapter. First, there is the need to investigate different types of online learning algorithms on more diverse datasets. This exploitation helps validate this thesis's findings across various use cases and assess the generalizability and effectiveness of the proposed approach across different contexts. Future studies might also examine the application of DL models in an online setting for prediction-based anomaly detection. DL could perform better than traditional models like ARIMA, especially in handling complex and non-linear data patterns. However, this approach would require carefully considering the increased computational costs and training time [46]. Besides, future research should give special priority to the topic of MLOps. In the online learning context, there are a lot of open questions regarding monitoring model performance, hyperparameter tuning, model versioning and deployment, and many others. Researchers have already started to explore this topic, and developers have begun to work on frameworks like *Beaver*[1] or *ChaCha* [44]. Additionally, the distinction between anomalies and concept drift remains a prominent challenge. Other than the idea of Adaptive Gradient Learning, it could also be feasible to perform dedicated tests to determine whether an outlier is a structural break, enabling models to differentiate between outliers and concept drift. Further research in this area is crucial to develop more reliable models capable of adapting to structural changes in data. Future research can significantly contribute to advancing prediction-based anomaly detection in online ML by addressing these areas.

1 See https://github.com/online-ml/beaver

APPENDIX

BIBLIOGRAPHY

[1] Charu C Aggarwal. *An introduction to outlier analysis*. Springer, 2017.

[2] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. "Unsupervised real-time anomaly detection for streaming data." In: *Neurocomputing* 262 (2017), pp. 134–147.

[3] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. "A survey of network anomaly detection techniques." In: *Journal of Network and Computer Applications* 60 (2016), pp. 19–31.

[4] Nesreen K Ahmed, Amir F Atiya, Neamat El Gayar, and Hisham El-Shishiny. "An empirical comparison of machine learning models for time series forecasting." In: *Econometric reviews* 29.5-6 (2010), pp. 594–621.

[5] Oren Anava, Elad Hazan, Shie Mannor, and Ohad Shamir. "Online learning for time series prediction." In: *Conference on learning theory*. PMLR. 2013, pp. 172–184.

[6] Oren Anava, Elad Hazan, Shie Mannor, and Ohad Shamir. "Online learning for time series prediction." In: *Conference on learning theory*. PMLR. 2013, pp. 172–184.

[7] Albert Bifet and Ricard Gavalda. "Learning from time-changing data with adaptive windowing." In: *Proceedings of the 2007 SIAM international conference on data mining*. SIAM. 2007, pp. 443–448.

[8] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A Lozano. "A review on outlier/anomaly detection in time series data." In: *ACM Computing Surveys (CSUR)* 54.3 (2021), pp. 1–33.

[9] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[10] Andriy Burkov. *The Hundred-Page Machine Learning Book*. 2020.

[11] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey." In: *ACM computing surveys (CSUR)* 41.3 (2009), pp. 1–58.

[12] Chris Chatfield. *Time-series forecasting*. Chapman and Hall/CRC, 2000.

[13] Andrew A Cook, Göksel Mısırlı, and Zhong Fan. "Anomaly detection for IoT time-series data: A survey." In: *IEEE Internet of Things Journal* 7.7 (2019), pp. 6481–6494.

[14] Jan G De Gooijer and Rob J Hyndman. "25 years of time series forecasting." In: *International journal of forecasting* 22.3 (2006), pp. 443–473.

[15] Tony Finch. "Incremental calculation of weighted mean and variance." In: *University of Cambridge* 4.11-5 (2009), pp. 41–42.

[16] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. "A survey on concept drift adaptation." In: *ACM computing surveys (CSUR)* 46.4 (2014), pp. 1–37.

[17] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.

[18] Heitor Murilo Gomes, Jesse Read, Albert Bifet, Jean Paul Barddal, and João Gama. "Machine learning for streaming data: state of the art, challenges, and opportunities." In: *ACM SIGKDD Explorations Newsletter* 21.2 (2019), pp. 6–22.

[19] Tian Guo, Zhao Xu, Xin Yao, Haifeng Chen, Karl Aberer, and Koichi Funaya. "Robust online time series prediction with recurrent neural networks." In: *2016 IEEE international conference on data science and advanced analytics (DSAA)*. IEEE. 2016, pp. 816–825.

[20] Max Halford, Jacob Montiel, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, et al. "River: machine learning for streaming data in Python." In: (2021).

[21] Douglas M Hawkins. *Identification of outliers*. Vol. 11. Springer, 1980.

[22] Steven CH Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. "Online learning: A comprehensive survey." In: *Neurocomputing* 459 (2021), pp. 249–289.

[23] Nikhil Ketkar. "Stochastic gradient descent." In: *Deep learning with Python*. Springer, 2017, pp. 113–132.

[24] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. "Overcoming catastrophic forgetting in neural networks." In: *Proceedings of the national academy of sciences* 114.13 (2017), pp. 3521–3526.

[25] Dominik Kreuzberger, Niklas Kühl, and Sebastian Hirschl. "Machine learning operations (mlops): Overview, definition, and architecture." In: *IEEE Access* (2023).

[26] Nikolay Laptev, Saeed Amizadeh, and Ian Flint. "Generic and scalable framework for automated time-series anomaly detection." In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 2015, pp. 1939–1947.

[27] Rikard Laxhammar and Göran Falkman. "Online learning and sequential anomaly detection in trajectories." In: *IEEE transactions on pattern analysis and machine intelligence* 36.6 (2013), pp. 1158–1173.

[28] Wen Liu, Weixin Luo, Dongze Lian, and Shenghua Gao. "Future frame prediction for anomaly detection–a new baseline." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 6536–6545.

[29] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. "Learning under concept drift: A review." In: *IEEE Transactions on Knowledge and Data Engineering* 31.12 (2018), pp. 2346–2363.

[30] Sasu Mäkinen, Henrik Skogström, Eero Laaksonen, and Tommi Mikkonen. "Who needs MLOps: What data scientists seek to accomplish and how can MLOps help?" In: *2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)*. IEEE. 2021, pp. 109–112.

[31] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, Puneet Agarwal, et al. "Long Short Term Memory Networks for Anomaly Detection in Time Series." In: *Esann*. Vol. 2015. 2015, pp. 89–94.

[32] H Zare Moayedi and MA Masnadi-Shirazi. "Arima model for network traffic prediction and anomaly detection." In: *2008 international symposium on information technology*. Vol. 4. IEEE. 2008, pp. 1–6.

[33] Mohsin Munir, Shoaib Ahmed Siddiqui, Andreas Dengel, and Sheraz Ahmed. "DeepAnT: A deep learning approach for unsupervised anomaly detection in time series." In: *IEEE Access* 7 (2018), pp. 1991–2005.

[34] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. "Deep learning for anomaly detection: A review." In: *ACM computing surveys (CSUR)* 54.2 (2021), pp. 1–38.

[35] Mahsa Salehi and Lida Rashidi. "A Survey on Anomaly detection in Evolving Data: [with Application to Forest Fire Risk Prediction]." In: *ACM SIGKDD Explorations Newsletter* 20.1 (2018), pp. 13–23.

[36] Sakti Saurav, Pankaj Malhotra, Vishnu TV, Narendhar Gugulothu, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. "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*. 2018, pp. 78–87.

[37] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. "Anomaly detection in time series: a comprehensive evaluation." In: *Proceedings of the VLDB Endowment* 15.9 (2022), pp. 1779–1797.

[38] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. "Hidden technical debt in machine learning systems." In: *Advances in neural information processing systems* 28 (2015).

[39] Shai Shalev-Shwartz et al. "Online learning and online convex optimization." In: *Foundations and Trends® in Machine Learning* 4.2 (2012), pp. 107–194.

[40] Swee Chuan Tan, Kai Ming Ting, and Tony Fei Liu. "Fast anomaly detection for streaming data." In: *Twenty-second international joint conference on artificial intelligence*. Citeseer. 2011, pp. 1511–1516.

[41] Sean J Taylor and Benjamin Letham. "Forecasting at scale." In: *The American Statistician* 72.1 (2018), pp. 37–45.

[42] Mark Tranmer and Mark Elliot. "Multiple linear regression." In: *The Cathie Marsh Centre for Census and Survey Research (CCSR)* 5.5 (2008), pp. 1–5.

[43] Daniel Vela, Andrew Sharp, Richard Zhang, Trang Nguyen, An Hoang, and Oleg S Pianykh. "Temporal quality degradation in AI models." In: *Scientific reports* 12.1 (2022), p. 11654.

[44] Qingyun Wu, Chi Wang, John Langford, Paul Mineiro, and Marco Rossi. "Chacha for online automl." In: *International Conference on Machine Learning*. PMLR. 2021, pp. 11263–11273.

[45] G Peter Zhang. "Time series forecasting using a hybrid ARIMA and neural network model." In: *Neurocomputing* 50 (2003), pp. 159–175.

[46] G Peter Zhang and Min Qi. "Neural network forecasting for seasonal and trend time series." In: *European journal of operational research* 160.2 (2005), pp. 501–514.