FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of master's thesis

Title:	Probabilistic and Machine Learning Models for Anomaly Detection in Time Series Data
Student:	Bc. Anna Husieva
Supervisor:	Ing. Vojtěch Šalanský, Ph.D.
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2024/2025

Instructions

1) Review and compile relevant research articles and papers on both probabilistic and machine learning methods for anomaly detection in time series data such as [1, 2].
2) Implement at least two probabilistic models and two machine learning models tailored for time series data and evaluate their performance on several public available datasets such as [3, 4, 5].
3) Investigate the strengths and weaknesses of choosen models across diverse anomaly scenarios.
4) Propose and implement a method that combines probabilistic and machine learning models. Determine its effectiveness in enhancing anomaly detection performance across different types of anomalies. Compare the results of the proposed method with the selected models mentioned above.

[1] Schmidl, Sebastian, Phillip Wenig and Thorsten Papenbrock. "Anomaly Detection in Time Series: A Comprehensive Evaluation." Proc. VLDB Endow. 15 (2022): 1779-1797.
[2] Braei, Mohammad and Sebastian Wagner. "Anomaly Detection in Univariate Time-series: A Survey on the State-of-the-Art." ArXiv abs/2004.00433 (2020)
[3] Lavin, Alexander and Subutai Ahmad. "Evaluating Real-Time Anomaly Detection Algorithms -- The Numenta Anomaly Benchmark." 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA) (2015): 38-44.

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 22 December 2023 in Prague.

[4] Phillip Wenig, Sebastian Schmidl, and Thorsten Papenbrock. TimeEval: A Benchmarking Toolkit for Time Series Anomaly Detection Algorithms. PVLDB, 15(12): 3678 - 3681, 2022
[5] A Labeled Anomaly Detection Dataset (ydata-labeled-time-series-anomalies-v1_0), Yahoo! Webscope Program. 2020 http://labs.yahoo.com/

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Probabilistic and Machine Learning Models for Anomaly Detection in Time Series Data

Bc. Anna Husieva

Department of Applied Mathematics
Supervisor: Ing. Vojtěch Šalanský, Ph.D.

May 8, 2024

Acknowledgements

First and foremost, I would like to express my heartfelt appreciation to my thesis supervisor, Ing. Vojtěch Šalanský, Ph.D., whose invaluable guidance and insightful advice, as well as his patience and willingness to help, have significantly contributed to the success of my research.

I would also like to extend my deepest gratitude to my family, whose unwavering support has been instrumental in enabling me to have the opportunity to achieve this milestone. A special thanks to my father, whose resilience and determination always inspired me to never give up.

Declaration

In Prague on May 8, 2024 ....

Citation of this thesis

Husieva, Anna. *Probabilistic and Machine Learning Models for Anomaly Detection in Time Series Data.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2024.

Abstrakt

Tato práce zkoumá účinnost pravděpodobnostních technik a technik strojového učení při detekci anomálií v časových řadách. Přehled příslušné literatury vytváří základ pro implementaci a vyhodnocení dvou pravděpodobnostních modelů a tří modelů strojového učení. Prostřednictvím experimentování na reálných i syntetických souborech dat jsou identifikovány silné a slabé stránky těchto modelů v různých typech anomálií. Je navržen a implementován hybridní přístup, který integruje prvky z pravděpodobnostních metod a metod strojového učení. Jeho účinnost při zvyšování výkonnosti detekce anomálií je dál posuzována.

Klíčová slova Detekce anomálií, časové řady, pravděpodobnostní metody, strojové učení, hybridní modely

Abstract

This thesis examines the effectiveness of probabilistic and machine learning techniques in anomaly detection within time series data. A review of relevant literature lays the groundwork for implementing and evaluating two probabilistic and three machine learning models. Through rigorous experimentation on both real-world and synthetic datasets, the strengths and weaknesses of these models are identified across various anomaly types. A hybrid approach is proposed and implemented, integrating elements from both probabilistic and machine learning frameworks. Its efficacy in enhancing anomaly detection performance is assessed.

Keywords Anomaly detection, time series, probabilistic methods, machine learning, hybrid models

Contents

List of Figures

List of Tables

Introduction

Time series data, characterized by its sequential nature, presents unique challenges in the realm of anomaly detection. Outlier detection has become a field of interest for many researchers and practitioners and is now one of the main tasks of time series data mining. Outlier detection has been studied in a variety of application domains such as credit card fraud detection, intrusion detection, or fault diagnosis [2].

These anomalies can take various forms, including point anomalies, where individual data points deviate significantly from the norm, and contextual anomalies, where deviations occur in the context of specific conditions or events. Additionally, collective anomalies involve deviations that occur collectively in a group of data points, reflecting systemic abnormalities or patterns.

The number and variety of anomaly detection algorithms have grown significantly over time. Since many of these solutions have been developed independently and by different research communities, evaluating and comparing the different approaches remains a challenge. For this reason, choosing the best detection technique for a given anomaly detection task is a difficult challenge [3].

In recent years, researchers have turned to probabilistic and machine learning models to address the challenges associated with anomaly detection in time series data [4]. Probabilistic models leverage statistical principles to model the underlying distribution of normal behavior and identify deviations indicative of anomalies. Machine learning algorithms offer powerful tools for learning complex patterns and detecting anomalies in time series data [5].

The integration of probabilistic and machine learning frameworks presents a promising avenue for enhancing anomaly detection performance in time series data. By combining these methodologies, researchers aim to leverage their complementary strengths and address the diverse forms of anomalies present in time series datasets [6].

The main goal of this thesis is to contribute to the advancement of anomaly detection techniques in time series data by comprehensively reviewing existing literature, implementing and evaluating state-of-the-art models, investigating their strengths and weaknesses across three anomaly types, and proposing a novel hybrid approach. By addressing the challenges inherent in anomaly detection within time series data, this research aims to provide practical solutions and insights that can inform the development of more effective anomaly detection systems in various domains.

Aim of the Thesis

The primary aim of this thesis can be summarized as follows:

1. Review and compile relevant research articles and papers on both probabilistic and machine learning methods for anomaly detection in time series data.
2. Implement at least two probabilistic models and two machine learning models tailored for time series data and evaluate their performance on several publicly available datasets.
3. Investigate the strengths and weaknesses of chosen models across diverse anomaly types.
4. Propose and implement a method that combines probabilistic and machine learning models. Determine its effectiveness in enhancing anomaly detection performance across different types of anomalies. Compare the results of the proposed method with the selected models mentioned above.

Structure of the work

This work is structured into five chapters. Chapter 1 covers the fundamentals of time series data, anomaly detection, and the overview of previous work. Chapter 2 focuses on statistical anomaly detection approaches. Chapter 3 explores machine learning techniques for anomaly detection. Chapter 4 presents experimental investigations, detailing methodology, dataset selection, model implementation, evaluation metrics, results, and analysis. Finally, Chapter 5 concludes the study by summarizing key findings.

CHAPTER 1

Fundamentals

1.1 Time series

Time series data is a fundamental component of many fields, including finance, economics, signal processing, and environmental science. Understanding and analyzing time series data is crucial for making predictions, identifying patterns, and detecting anomalies. This chapter explores the concept of time series data, its attributes, and its significance across different fields.

Time series is a set of observations X_t. The index t of X_t is referred to as time, and X_t is considered as the state or output of a stochastic system at time t.

Definition 1.1.1. Let $(\Omega, \mathcal{F}, \mathcal{P})$ be a probability space, where Ω is a set of all possible outcomes, $\mathcal{F}$ is an event space, $\mathcal{P}$ is a probability function, and T be a set of indices interpreted as time. A time series is defined as the set $X_t, t \in T$, where X_t are random variables from $(\Omega, \mathcal{F}, \mathcal{P})$ [7].

1.1.1 Properties of time series

It is imperative to establish certain a priori structural assumptions regarding the time series. If the variables X_t were entirely arbitrary, the sequence $(X_1, \ldots, X_n)$ would represent a single observation from an entirely unknown distribution in $\mathbb{R}^n$. Drawing conclusions about this distribution, let alone predicting the distribution of future values $X_{n+1}, X_{n+2}, \ldots$, would be insurmountably challenging.

One fundamental form of structure is stationarity, which manifests in two primary types.

Definition 1.1.2. A time series is said to be strictly stationary if the joint distribution of any finite subset of $X_{t_1}, X_{t_2}, \ldots, X_{t_n}$ is invariant to shifts in time. Mathematically, it holds that for all k, h and all $t_1, t_2, \ldots, t_k \in T$, $(X_{t_1}, X_{t_2}, \ldots, X_{t_k})$ has the same distribution as $(X_{t_1+h}, X_{t_2+h}, \ldots, X_{t_k+h})$ [8].

Definition 1.1.3. A time series $\{X_t\}_{t=1}^{n}$ is said to be weakly stationary if it satisfies the following conditions:

- The expected value of the time series, denoted as μ, remains constant over time.

$$\mu = \mathrm{E}(X_t) = \mathrm{E}(X_{t+h}) \quad \text{for all} \quad t \in T, h \in \mathbb{N} \tag{1.1}$$

- The variance of the time series, denoted as σ^2, remains constant over time.

$$\sigma^2 = \mathrm{Var}(X_t) = \mathrm{Var}(X_{t+h}) \quad \text{for all} \quad t \in T, h \in \mathbb{N} \tag{1.2}$$

- The autocovariance function, denoted as $\gamma(h)$, depends only on the lag h and remains constant over time. Mathematically, it holds that

$$\mathrm{Cov}(X_t, X_{t+h}) = \gamma(h) = \gamma(-h) \quad \text{for all} \quad t \in T, h \in \mathbb{N} \tag{1.3}$$

  where $\mathrm{Cov}(X_t, X_{t+h})$ represents the covariance between the time series values at times t and $t + h$ [8].

The assumption of stationarity simplifies the modeling process by providing a stable foundation for analyzing past observations and making predictions. In the context of anomaly detection, stationary time series allow for the development of models that can effectively capture normal patterns and behaviors.

However, many real-world time series exhibit non-stationary characteristics due to various factors, including trends, seasonal patterns, and structural changes. These non-stationarities pose significant challenges for anomaly detection algorithms, as they may lead to false alarms or missed detections if not properly accounted for [8].

Definition 1.1.4. A trend in a time series $\{X_t\}_{t=1}^{n}$ refers to a systematic change or long-term pattern observed in the data over time. Mathematically, a trend can be represented by a function $f(t)$ that captures the underlying trend component of the time series [9].

Definition 1.1.5. A linear trend is characterized by a constant rate of change over time. It can be expressed as:

$$f(t) = \alpha t + \beta, \tag{1.4}$$

where:

- α represents the slope of the trend line,
- β represents the intercept of the trend line.

Definition 1.1.6. A non-linear trend exhibits a more complex pattern of change over time, which cannot be captured by a simple linear function. It may involve polynomial functions, exponential growth or decay, or other non-linear patterns.

$$f(t) = \sum_{i=1}^{k} \alpha_i g_i(t), \tag{1.5}$$

where $g_i(t)$ are basis functions representing different components of the non-linear trend [9].

Linear trends are characterized by a constant rate of change over time, represented by a straight line in the time series plot. On the other hand, non-linear trends exhibit more complex patterns of change, which may involve polynomial functions, exponential growth or decay, or other non-linear patterns.

Anomaly detection algorithms need to account for trends to identify deviations from the expected trend and flag potential anomalies. Machine learning models, such as regression algorithms or time series decomposition techniques, can be employed to capture and model trends in the data, enabling accurate detection of anomalies associated with abrupt changes or shifts in the underlying trend.

Definition 1.1.7. Seasonality in a time series refers to periodic fluctuations or patterns observed in the data at fixed intervals. These patterns typically repeat over regular time intervals, such as daily, weekly, or yearly cycles [8].

Seasonality introduces additional complexity to the modeling process as it reflects predictable patterns that need to be accounted for. Anomalies may coincide with normal seasonal variations, making it challenging to differentiate between expected and unexpected behavior. There are two main types of seasonality.

Definition 1.1.8. Additive seasonality implies that the seasonal variations remain constant regardless of the level of the time series [8]. Mathematically, it can be expressed as:

$$X_t = m_t + s_t + \epsilon_t, \tag{1.6}$$

where:

- m_t represents the trend component.
- s_t represents the seasonal component.
- ϵ_t represents the error term or random noise.

Definition 1.1.9. Multiplicative seasonality implies that the seasonal variations are proportional to the level of the time series. Mathematically, it can be expressed as:

$$X_t = m_t \times s_t \times \epsilon_t, \tag{1.7}$$

where:

- m_t represents the trend component.
- s_t represents the seasonal component.
- ϵ_t represents the error term or random noise [8].

In summary, trends and seasonality are critical components of time series data that significantly influence anomaly detection. Understanding and modeling trends and seasonality accurately is essential for distinguishing between normal variations and anomalous behavior in the time series data.

1.2 Anomalies in time series

In time series data, an anomaly or outlier can be termed as a data point which does not follow the common collective trend or seasonal or cyclic pattern of the entire data and is significantly distinct from the rest of the data. By significant, most data scientists mean statistical significance, which in other words, signifies that the statistical properties of the data point are not in alignment with the rest of the series [10].

A time series anomaly is a sequence of data points $(X_{t_1}, X_{t_2}, \ldots, X_{t_k})$ of length $j - i + 1 \geq 1$ that deviates concerning some characteristic embedding, model, or similarity measure from frequent patterns in the time series [3].

There are different classifications of anomalies in time series data. The most popular and used one was proposed in this article [11]. Visual representation of these anomaly types is depicted in Figure 1.1. Point anomalies represent individual data points that stand out from the rest of the dataset due to factors such as measurement errors, sensor failures, or isolated events. Contextual anomalies occur when data points deviate from the expected pattern within specific contexts or conditions. For instance, a sudden temperature spike during the winter season would be considered a contextual anomaly. Collective anomalies involve groups of data points exhibiting abnormal behavior collectively, indicating systemic issues or patterns within the dataset, such as a surge in network traffic during a cyberattack. The former classification features classes of anomalies that are not mutually exclusive, because a collective anomaly can also be a contextual anomaly.

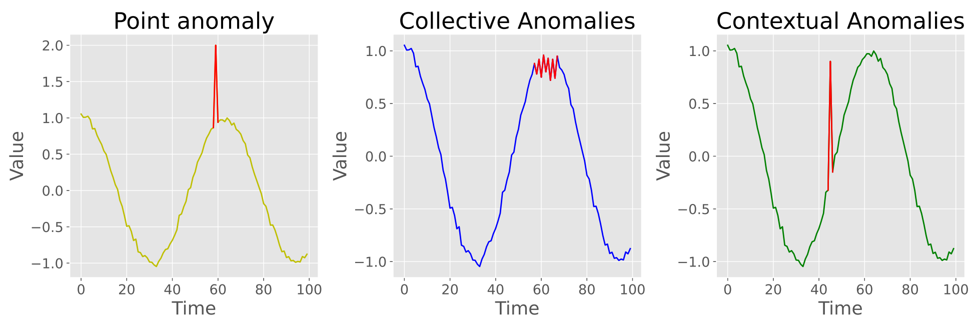

Figure 1.1: Types of anomalies.

Understanding the characteristics of anomalies in time series data is essential for effective detection and analysis. However, it is imperative to consider the quantification of anomaly severity. Anomaly scores serve as pivotal tools in this regard, offering a quantitative measure of the degree of anomaly exhibited by each data point. These scores are calculated based on various factors, including the deviation from the expected pattern, significance relative to the rest of the data, and contextual information.

1.3 Previous work

This chapter offers a thorough review of the existing literature, comprising seminal research articles and survey papers that contribute significantly to understanding anomaly detection techniques. Additionally, it emphasizes the importance of hybrid approaches in addressing the challenges associated with anomaly detection tasks.

Shaukat et al. (2021) [12] conducted a comprehensive review that categorizes anomaly detection techniques based on different types of anomalies. Their analysis provides valuable insights into the strengths and limitations of various methodologies, laying the groundwork for future research directions.

In the domain of time series forecasting, Zhang (2003) [13] introduced a pioneering hybrid model that combines Autoregressive Integrated Moving Average (ARIMA) with neural networks. While Zhang's study primarily emphasizes forecasting applications, the hybrid methodology holds significant relevance for anomaly detection tasks. Zhang's findings underscore the effectiveness of hybrid methodologies in capturing intricate temporal dynamics, thereby paving the way for similar approaches in anomaly detection. However, it is worth noting that Zhang's approach primarily utilizes feed-forward neural networks (FFNN) in conjunction with ARIMA. In contrast, this thesis will innovate by implementing a recurrent neural network (RNN) as part of the hybrid model.

Braei and Wagner (2020) [7] conducted a comprehensive evaluation of various methodologies, providing valuable insights into their performance characteristics. Specifically, Braei and Wagner observed that statistical methods not only exhibited superior anomaly detection accuracy but also boasted shorter training and prediction durations compared to machine learning and deep learning techniques. This advantage stems from the inherently simpler optimization process of statistical algorithms, which entail fewer hyperparameters to fine-tune.

Furthermore, Braei and Wagner's investigation revealed a critical insight regarding the performance of statistical methods in detecting different anomaly types. While statistical approaches surpassed in identifying point and collective anomalies, they encountered challenges with contextual anomalies. In contrast, deep learning approaches, particularly neural networks, showcased greater adaptability in optimizing models for diverse anomaly types, resulting in higher AUC values, especially in scenarios dominated by contextual anomalies.

The comparison conducted in Braei and Wagner's study between statistical, classical machine learning, and deep learning approaches provided a comprehensive benchmark for evaluating anomaly detection techniques in univariate time series data. This thesis at hand seeks to expand upon this work by incorporating multiple hybrid models. By comparing the performance of these hybrid approaches against traditional statistical and machine learning methods, the thesis aims to offer a more nuanced understanding of anomaly detection techniques across different anomaly types and datasets.

An empirical investigation conducted by Schmidl et al. (2022) [3] offers valuable insights into the performance characteristics of various anomaly detection techniques. In their comprehensive study, Schmidl et al. collected and re-implemented 71 anomaly detection algorithms, representing a diverse spectrum of techniques. Through rigorous evaluation of 976 time series datasets, they assessed the effectiveness, efficiency, and robustness of these algorithms.

Following related studies, Schmidl et al. found that despite the higher computational effort required for training, deep learning approaches are not yet competitive. The study also corroborates the notion that simple methods yield performance almost as good as more sophisticated methods [3]. However, the absence of a single algorithm that clearly outperforms others underscores the need for further research in several key areas.

The study emphasizes the necessity for research on holistic and hybrid anomaly detection systems. Such systems can leverage existing strengths to detect a wider range of anomalies in time series data with varying characteristics. Additionally, the study underscores the importance of enhancing the reliability of anomaly detection algorithms, particularly in terms of robustness and scalability. The sensitivity of most anomaly detection algorithms to parameter settings highlights the need for research on auto-configuring and self-tuning algorithms.

In summary, the incorporation of these seminal articles provides a concrete understanding of existing methodologies in anomaly detection within time series data. By synthesizing existing literature and offering concrete insights, this chapter lays the foundation for the proposed thesis's contributions, aiming to advance the field of anomaly detection and address real-world challenges.

Chapter 2

Statistical anomaly detection approaches

Statistical anomaly detection approaches leverage methods to identify anomalies in data by modeling the inherent stochastic nature of the underlying processes. These methods prove particularly effective when dealing with dynamic systems where normal behavior exhibits random variations. While acknowledging the existence of other models within this domain, such as Hidden Markov Models, it is important to note that their exploration falls beyond the scope of this thesis.

2.1 ARIMA

2.1.1 Autoregressive model

One of the most basic stochastical models for univariate time series is the Autoregressive model (AR).

Definition 2.1.1. The autoregressive model $AR(p)$ determines the value y_t of a process at an arbitrary time step t using a linear combination of the p-last values:

$$y_t = c + \phi_1 \cdot y_{t-1} + \phi_2 \cdot y_{t-2} + ... + \phi_p \cdot y_{t-p} + \varepsilon_t, \tag{2.1}$$

where p is the order of the AR model.The weights $\phi = [c, ..., \phi_p]^T$ of the linear combination are the model parameters. Furthermore, an AR model assumes that this process is superimposed by white noise; i.e. the ε_t are assumed to be uncorrelated with each other in time and identically distributed, with an expected value of zero and finite variance σ^2. This model is abbreviated as $AR(p)$ [14].

2.1.1.1 Parameter estimation using least squares

The autoregressive coefficients can be estimated using the least squares method, a mathematical optimization technique that minimizes the sum of squared differences between observed and predicted values.

Definition 2.1.2. Consider a time series comprising positive real values, and our objective involves characterizing it using an autoregressive model of order p. The aim is to estimate the regression coefficients denoted by ϕ [7]. Let X represent the design matrix, Y denote the measurement vector, and ϵ stand for the vector encompassing noise terms, specified as follows:

$$X = \begin{bmatrix} 1 & x_0 & \cdots & x_{p-1} \\ 1 & x_1 & \cdots & x_p \\ \vdots & \ddots & \vdots & \\ 1 & x_{N-p} & \cdots & x_{N-1} \end{bmatrix}, Y = \begin{bmatrix} x_p \\ x_{p+1} \\ \vdots \\ x_N \end{bmatrix}, \epsilon = \begin{bmatrix} \varepsilon_p \\ \varepsilon_{p+1} \\ \vdots \\ \varepsilon_N \end{bmatrix}, \phi = \begin{bmatrix} c \\ \phi_p \\ \vdots \\ \phi_1 \end{bmatrix}. \tag{2.2}$$

Then Y can be denoted as:

$$Y = X\phi + \epsilon. \tag{2.3}$$

The least squares estimate of ϕ is given by:

$$\hat{\phi} = (X^T X)^{-1} X^T Y. \tag{2.4}$$

2.1.1.2 Training and anomaly detection

AR models assume that the data is stationary. Thus, it is important to analyze the data and transform it if necessary.

Following the training process of an Autoregressive model, the prediction for each time point t, denoted as $\hat{X}_t$, can be obtained as the solution to the AR model equation:

$$\hat{X}_t = \hat{c} + \sum_{i=1}^{p} \hat{\phi}_i X_{t-i}. \tag{2.5}$$

Subsequently, the residual ε_t at time t is calculated as the difference between the observed value X_t and the predicted value $\hat{X}_t$:

$$\varepsilon_t = X_t - \hat{X}_t. \tag{2.6}$$

This residual, ε_t, serves as an indicator of the deviation between the model's prediction and the actual observation at time t [7]. The anomaly score AS_t for each time point is defined as the magnitude of the residual:

$$AS_t = |\varepsilon_t|. \tag{2.7}$$

The anomaly score quantifies the discrepancy between the predicted and observed values, providing a measure of how anomalous the current observation is according to the trained AR model. A higher anomaly score suggests a greater deviation and, consequently, a higher likelihood of the presence of an anomaly.

In summary, the anomaly score (AS_t) for each time point t in the time series is calculated as the absolute value of the residual ε_t, reflecting the degree of anomaly according to the Autoregressive model's predictions.

2.1.2 Moving average

While the AR model considers X_t as a linear transformation of the last p observations of a time series, the moving average Model (MA) considers the current observation X_t as a linear combination of the last q random shocks.

Moving average processes find widespread use across diverse domains, notably within econometrics. In this context, stochastic disruptions are ascribed to governmental determinations, scarcities of pivotal resources, and, more contemporarily, elements such as unconventional events. These stochastic perturbations are prone to diffusion into subsequent temporal instances, albeit not through direct means, as observed in the Autoregressive (AR) model.

Definition 2.1.3. The MA model with a preceding window of length q is also called the MA process of order q or $MA(q)$.

$$X_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q}. \tag{2.8}$$

In this model, μ is the mean of the time series, ε_t is the white noise error term, and $\theta_1, \theta_2, \ldots, \theta_q$ are parameters reflecting the weights of past error terms[7].

To apply Moving average models for anomaly detection, the parameters μ, $\theta_1, \theta_2, \ldots, \theta_q$ need to be estimated. This is typically done through statistical methods such as Maximum Likelihood Estimation (MLE) or other optimization techniques [7].

Once the model is trained, predictions $\hat{X}_t$ for future time points can be made using the estimated parameters. The residual at time t is then calculated the same as for the AR model.

Anomalies are often associated with large residuals, indicating deviations from the predicted values.

An anomaly score (AS_t) can be defined based on the standard deviation of residuals:

$$AS_t = \frac{\varepsilon_t}{\sigma_\varepsilon}, \tag{2.9}$$

where σ_ε is the standard deviation of residuals.

2.1.3 Autoregressive Integrated Moving Average Model

Autoregressive Moving Average (ARMA) model is a fundamental tool in time series analysis, often employed in various fields such as economics, finance, and signal processing. Combining the concepts of autoregressive (AR) and moving average (MA) models, ARMA seeks to capture the underlying patterns and dynamics present in sequential data. The AR component accounts for the linear dependency of a variable on its past values, reflecting the notion that current observations are influenced by preceding ones. Meanwhile, the MA component captures the dependency of the current observation on past white noise errors, indicating short-term fluctuations or shocks affecting the series, which is especially useful for modeling financial or economic data. By integrating both components, ARMA aims to provide a comprehensive framework for modeling and forecasting time series data, offering insights into the inherent structure and behavior of the observed phenomena. Its versatility and effectiveness make it a cornerstone in statistical modeling, facilitating rigorous analysis and prediction in diverse applications.

Definition 2.1.4. A time series $\{x_t; t = 0, \pm1, \pm2, \ldots\}$ is ARMA(p, q) if it is stationary and

$$x_t = \phi_1 x_{t-1} + \cdots + \phi_p x_{t-p} + w_t + \theta_1 w_{t-1} + \cdots + \theta_q w_{t-q}, \tag{2.10}$$

with $\phi_p \neq 0, \theta_q \neq 0$, and $\sigma_w^2 > 0$. The parameters p and q are called the autoregressive and the moving average orders, respectively. If x_t has a nonzero mean μ, $\alpha = \mu\,(1 - \phi_1 - \cdots - \phi_p)$ and the model is denoted as

$$x_t = \alpha + \phi_1 x_{t-1} + \cdots + \phi_p x_{t-p} + w_t + \theta_1 w_{t-1} + \cdots + \theta_q w_{t-q}, \tag{2.11}$$

where $w_t \sim wn\left(0, \sigma_w^2\right)$. As previously noted, when $q = 0$, the model is called an autoregressive model of order p, AR(p), and when $p = 0$, the model is called a moving average model of order q, MA(q) [15].

Box and Jenkins [16] developed a practical approach to building ARIMA models, which has a fundamental impact on the time series analysis and forecasting applications. The Box–Jenkins methodology includes three iterative steps of model identification, parameter estimation, and diagnostic checking. The basic idea of model identification is that if a time series is generated from an ARIMA process, it should have some theoretical autocorrelation properties. By matching the empirical autocorrelation patterns with the theoretical ones, it is often possible to identify one or several potential models for the given time series. Box and Jenkins [16] proposed to use the autocorrelation function and the partial autocorrelation function of the sample data as the basic tools to identify the order of the ARIMA model [13]. In the identification step, data transformation is often needed to make the time series stationary. When the observed time series presents trend and heteroscedasticity, differencing and power transformation are often applied to the data to remove the trend and stabilize the variance before an ARIMA model can be fitted [13].

2.1.3.1 Stationarity testing

This section introduces the concept of test statistics for assessing stationarity. A notable challenge when testing stationarity in observations of natural systems is the limited availability of realizations of the system under study, rather than direct access to the system itself. Stationarity, as a property, pertains to the underlying process. Consequently, each test implicitly assumes that the time series is representative of the system, offering only an upper limit on the extent of stationarity violation.

Ascertaining the stationarity of a time series constitutes a routine procedure in autoregressive models. Mastery of the Augmented Dickey-Fuller (ADF) test and the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test is imperative for conducting thorough time series analysis.

The ADF test is a frequently employed statistical method utilized to examine the existence of a unit root within a time series. A unit root denotes a feature of a time series process characterized by the perpetual increase in both mean and variance over time, resulting in the series failing to converge to a constant value, thus rendering it non-stationary [17].

$$\begin{aligned}\Delta y_t &= \Phi_0 y_{t-1} + \Phi_1 \Delta y_{t-1} + \cdots + \Phi_p \Delta y_{t-p} + \varepsilon_t = \\ &= \Phi_0 y_{t-1} + \sum_{i=1}^{p-1} \Phi_i \Delta y_{t-i} + \varepsilon_t,\end{aligned} \tag{2.12}$$

where, $\Phi_0 = (\Phi_1 + \Phi_2 + \cdots + \Phi_p) - 1$, $\Phi_i = (\Phi_{i+1} + \Phi_{i+2} + \cdots + \Phi_p)$ and $\sum_{i=1}^{p-1} \Phi_i y_{t-i} + \varepsilon_t$ is a lagged dependent variable.

$$\begin{aligned}&\mathrm{H}_0 : \Phi_0 = 0 \text{ meaning time series contains a unit root.}\\ &\mathrm{H}_1 : \Phi_0 < 0 \text{ meaning time series does not contain a unit root.}\end{aligned} \tag{2.13}$$

The examination adheres to an asymmetric t distribution, wherein the majority of values are expected to be negative. Subsequently, the calculated t value is juxtaposed with the simulated critical values provided by Fuller. The rejection of the null hypothesis is indicated by small values, as evident from the hypotheses, where the alternative proposes values less than zero.

The KPSS test serves as a contrasting approach to the ADF test for assessing stationarity and relies on linear regression. It decomposes the time series into components such as deterministic trend, random walk, and stationary error [17].

$$y_t = r_t + \beta_t + \varepsilon_t, \tag{2.14}$$

where, r_t is random walk, β_t is deterministic trend and, ε_t is stationary error with zero mean.

$$r_t = r_{t-1} + v_t, \tag{2.15}$$

where v_t is i.i.d. $(0, \sigma^2)$.

If the random walk component is present, indicating non-stationarity, the null hypothesis posits that the variance $\sigma^2 = 0$, rendering the time series, y_t, trend stationary. Subsequently, the KPSS test examines whether a unit root exists in r_t when $\beta \neq 0$. The corresponding hypotheses for this test are:

$$\begin{aligned} &H_0 : \text{The time series is trend stationary.} \\ &H_1 : \text{The time series contains a unit root.} \end{aligned} \tag{2.16}$$

In scenarios where $\beta = 0$, the null hypothesis shifts to the time series being level stationary.

When the data is stationary, it exhibits a constant intercept or remains stationary around a fixed level. The test employs OLS regression to derive the equation, with slight variations depending on whether one aims to test for level stationarity or trend stationarity [8].

Both tests are used to test stationarity, but in some cases, they both give contradictory results. There are 4 possible combinations of KPSS and ADF test results.

1. If KPSS and ADF agree that the series is stationary, consider it stationary.

2. ADF finds a unit root, but KPSS finds that the series is stationary around a deterministic trend. Then, the series is trend-stationary, and it needs to be detrended, meaning differentiation should be applied. Alternatively, a transformation may rid it of its trend.

3. ADF does not find a unit root, but KPSS claims that it is non-stationary, then the series is difference stationary, meaning differentiation should be applied.

4. If KPSS and ADF agree that the series is non-stationary, consider it non-stationary, meaning differentiation should be applied.

2.1.3.2 Determining the order of AR and MA

One central task of the ARIMA model building is to determine the appropriate model order (p, q) [13]. Autocorrelation analysis plays a crucial role in identifying patterns and assessing randomness within data. Its significance becomes particularly pronounced when considering the utilization of an autoregressive moving average (ARMA) model for forecasting, as it aids in parameter determination. This analysis entails examining plots of the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF).

Definition 2.1.5. The Autocorrelation Function (ACF) at lag k, denoted as $\rho(k)$, is a statistical measure that quantifies the linear relationship between observations in a time series separated by a lag of k time periods. It is defined as:

$$\rho(k) = \frac{\mathrm{Cov}(X_t, X_{t-k})}{\sqrt{\mathrm{Var}(X_t) \cdot \mathrm{Var}(X_{t-k})}}, \tag{2.17}$$

where X_t represents the observation at time t, and $\mathrm{Cov}(X_t, X_{t-k})$ and $\mathrm{Var}(X_t)$ denote the covariance and variance of the time series, respectively [13].

Definition 2.1.6. The Partial Autocorrelation Function (PACF) at lag k, denoted as $\phi(k)$, measures the correlation between X_t and X_{t-k} with the linear dependence on $\{X_{t-k+1}, \dots, X_{t-1}\}$ removed. It is defined recursively as:

$$\begin{aligned} \phi(1) &= \rho(1), \\ \phi(k) &= \frac{\mathrm{Cov}(X_t - \hat{X}_t, X_{t-k} - \hat{X}_{t-k})}{\sqrt{\mathrm{Var}(X_t - \hat{X}_t) \cdot \mathrm{Var}(X_{t-k} - \hat{X}_{t-k})}}, \end{aligned} \tag{2.18}$$

where $\hat{X}_t$ and $\hat{X}_{t-k}$ represent the predictions of X_t and X_{t-k}, respectively, based on a linear regression model involving the intervening observations [13].

It is firmly established that the examination of ACF and PACF offers valuable intuition regarding the selection of AR and MA orders in time series modeling. However, it is imperative to acknowledge that the determination of these orders from ACF and PACF is a nuanced process, often involving meticulous analysis and interpretation guided by statistical rigor and domain expertise.

Therefore, while ACF and PACF undeniably offer crucial insights into the selection of AR and MA orders, the methodology for extracting such information warrants careful consideration. However, this thesis will not delve into the specifics of how exactly to determine the order from ACF and PACF.

2.1.3.3 Seasonal ARIMA

In practice, the data often exhibits relationships between successive observations and those occurring at fixed seasonal intervals. Within the ARIMA framework, these relationships are accommodated by introducing lag terms into the model components. This approach enables the model to capture the temporal dependencies inherent in the data, both in terms of immediate sequential observations and those occurring at predefined seasonal intervals.

Definition 2.1.7. The seasonal ARIMA model incorporates non-seasonal and seasonal factors in a multiplicative model. The SARIMA model is noted as:

$$\mathrm{ARIMA}(p, d, q) \times (P, D, Q)S, \tag{2.19}$$

with p is a non-seasonal AR order, d is a non-seasonal differencing, q is a non-seasonal MA order, P is a seasonal AR order, D is a seasonal differencing, Q

is a seasonal MA order, and S is a time span of repeating seasonal pattern [17].

2.2 Exponential Smoothing

Exponential smoothing is another widely used technique in time series analysis, particularly in forecasting and anomaly detection tasks. This chapter delves into the application of exponential smoothing methods for detecting anomalies in time series data.

2.2.1 Single Exponential Smoothing

Single exponential smoothing (SES) is a fundamental technique that aims to capture the underlying trend of a time series by assigning exponentially decreasing weights to past observations.

Using the simplest method, all future forecasts are derived solely from the most recent observation in the series:

$$\hat{y}_{T+h|T} = y_T, \quad \text{for } h \in \mathbb{N}. \tag{2.20}$$

Here, the simplest method operates under the assumption that only the last observed value holds significance for future forecasts, with no consideration given to prior observations. This approach effectively assigns all weight to the most recent observation, resulting in a simplistic forecast [18].

Utilizing the averaging method, future forecasts are determined by computing the arithmetic mean of all observed data:

$$\hat{y}_{T+h|T} = \frac{1}{T}\sum_{t=1}^{T} y_t, \quad \text{for } h \in \mathbb{N}. \tag{2.21}$$

In contrast to the simplest method, the averaging method treats all observations equally, assigning identical weights to each when generating forecasts.

However, there exists a desire for a more nuanced approach, one that places greater emphasis on recent observations while still considering historical data. Simple exponential smoothing precisely addresses this requirement. Forecasts are computed using weighted averages, with exponentially decreasing weights assigned to observations based on their recency:

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \dots, \tag{2.22}$$

where $0 \leq \alpha \leq 1$ represents the smoothing parameter. This formulation calculates the one-step-ahead forecast for time $T+1$ as a weighted average of all preceding observations in the series $y_1, \dots, y_T$. The parameter α governs the rate of weight decay, allowing for adaptable forecasting based on the recency of observations [18].

For any α value ranging between 0 and 1, the weightings assigned to observations exhibit an exponential decrease as one traverses back through time, thus lending the technique its designation as exponential smoothing. A smaller α value (near 0) allocates greater weight to observations from the distant past, whereas a larger α value (close to 1) bestows more significance upon recent observations. In the extreme scenario where $\alpha = 1$, $\hat{y}_{T+1|T} = y_T$, resulting in forecasts identical to those generated via the simplest method.

2.2.1.1 Weighted average form

The forecast at time $T + 1$ is determined by a weighted average between the most recent observation y_T and the previous forecast $\hat{y}_{T|T-1}$:

$$\hat{y}_{T+1|T} = \alpha y_T + (1-\alpha)\hat{y}_{T|T-1}, \tag{2.23}$$

where $0 \leq \alpha \leq 1$ represents the smoothing parameter. Similarly, the fitted values can be expressed as:

$$\hat{y}_{t+1|t} = \alpha y_t + (1-\alpha)\hat{y}_{t|t-1}, \text{for } t = 1, \ldots, T. \tag{2.24}$$

The process initiates with the first fitted value at time 1, denoted by ℓ_0 (which requires estimation) [18]. Then,

$$\begin{aligned}
\hat{y}_{2|1} &= \alpha y_1 + (1-\alpha)\ell_0, \\
\hat{y}_{3|2} &= \alpha y_2 + (1-\alpha)\hat{y}_{2|1}, \\
\hat{y}_{4|3} &= \alpha y_3 + (1-\alpha)\hat{y}_{3|2}, \\
&\vdots \\
\hat{y}_{T|T-1} &= \alpha y_{T-1} + (1-\alpha)\hat{y}_{T-1|T-2}, \\
\hat{y}_{T+1|T} &= \alpha y_T + (1-\alpha)\hat{y}_{T|T-1}.
\end{aligned} \tag{2.25}$$

Substituting each equation into the subsequent equation yields:

$$\begin{aligned}
\hat{y}_{3|2} &= \alpha y_2 + (1-\alpha)\left[\alpha y_1 + (1-\alpha)\ell_0\right] \\
&= \alpha y_2 + \alpha(1-\alpha)y_1 + (1-\alpha)^2\ell_0, \\
\hat{y}_{4|3} &= \alpha y_3 + (1-\alpha)\left[\alpha y_2 + \alpha(1-\alpha)y_1 + (1-\alpha)^2\ell_0\right] \\
&= \alpha y_3 + \alpha(1-\alpha)y_2 + \alpha(1-\alpha)^2 y_1 + (1-\alpha)^3\ell_0, \\
&\vdots \\
\hat{y}_{T+1|T} &= \sum_{j=0}^{T-1} \alpha(1-\alpha)^j y_{T-j} + (1-\alpha)^T\ell_0.
\end{aligned} \tag{2.26}$$

2.2.1.2 Component form

An alternative perspective involves representing exponential smoothing methods in component form. For simple exponential smoothing, the only component considered is the level, denoted by ℓ_t. Other methods, explored later in this chapter, may incorporate additional components such as trend b_t and seasonal component s_t.

The component form entails a forecast equation and a smoothing equation for each component involved in the method. For simple exponential smoothing, the component form is expressed as:

$$\hat{y}_{t+h|t} = \ell_t, \tag{2.27}$$

$$\ell_t = \alpha y_t + (1-\alpha)\ell_{t-1}, \tag{2.28}$$

where Eq. 2.27 is a forecast equation and Eq. 2.28 is a smoothing equation. Here, ℓ_t represents the level (or smoothed value) of the series at time t. When $h = 1$, the fitted values are obtained, while $t = T$ yields true forecasts beyond the training data.

The forecast equation reveals that the forecast value at time $t+1$ is the estimated level at time t. The smoothing equation for the level, often referred to as the level equation, provides the estimated level of the series at each period t [17].

By substituting ℓ_t with $\hat{y}_{t+1|t}$ and ℓ_{t-1} with $\hat{y}_{t|t-1}$ in the smoothing equation, the weighted average form of simple exponential smoothing is regained.

While the component form of simple exponential smoothing may not be directly applicable, it serves as a foundational framework for incorporating additional components in more complex models [17].

2.2.1.3 Optimisation and anomaly detection

For the application of any exponential smoothing method, selecting the appropriate smoothing parameters and initial values is imperative. Specifically, for simple exponential smoothing, determining the values of α and ℓ_0 is essential. Once these values are known, forecasts can be computed from the data. However, for more complex methods, there may be multiple smoothing parameters and initial components to be chosen.

While subjective methods may involve the forecaster specifying the smoothing parameters based on prior experience, a more objective approach involves estimating these parameters from the observed data. Similar to the estimation of coefficients in a regression model, exponential smoothing parameters and initial values can be estimated by minimizing the sum of squared residuals (SSE).

The residuals $\varepsilon_t = y_t - \hat{y}_{t|t-1}$ for $t = 1, \dots, T$ are used to specify SSE as:

$$\text{SSE} = \sum_{t=1}^{T} \varepsilon_t^2 = \sum_{t=1}^{T} \left(y_t - \hat{y}_{t|t-1}\right)^2 . \tag{2.29}$$

Unlike the regression scenario where formulas exist to compute regression coefficients minimizing SSE, this poses a non-linear minimization problem, necessitating the use of optimization techniques to solve it effectively.

Similar to the ARIMA model, anomaly scores for the Holt-Winters ES model are determined based on the prediction errors. The anomaly score defined in Eq. 2.7, represents the deviation between the observed and predicted values, quantifying the abnormality of each data point in the time series.

2.2.2 Holt's linear trend method

Holt [19] introduced an extension to simple exponential smoothing to accommodate data exhibiting a trend. This method incorporates a forecast equation and two smoothing equations, one for the level and one for the trend:

$$\hat{y}_{t+h|t} = \ell_t + hb_t, \tag{2.30}$$

$$\ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}), \tag{2.31}$$

$$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}, \tag{2.32}$$

where Eq. 2.30 is a forecast equation, Eq. 2.31 is a level equation and Eq. 2.32 is a trend equation. ℓ_t represents the estimated level of the series at time t, while b_t denotes the estimated trend (slope) at time t. The smoothing parameter for the level α and the smoothing parameter for the trend β^* are constrained to the range $0 \leq \alpha \leq 1$ and $0 \leq \beta^* \leq 1$.

Similar to simple exponential smoothing, the level equation expresses ℓ_t as a weighted average of the observation y_t and the one-step-ahead training forecast for time t, given by $\ell_{t-1} + b_{t-1}$. The trend equation indicates that b_t is a weighted average of the estimated trend at time t based on $\ell_t - \ell_{t-1}$ and b_{t-1}, the previous estimate of the trend [19].

Unlike in simple exponential smoothing, the forecast function now exhibits a trend. The h-step-ahead forecast is determined by the last estimated level plus h times the last estimated trend value. Consequently, the forecasts follow a linear function of h [19].

2.2.2.1 Damped trend method

Holt introduced a parameter to address the tendency of linear forecasting methods to exhibit a constant trend indefinitely into the future. This parameter "dampens" the trend to a flat line at some point in the future. Models incorporating a damped trend have demonstrated notable success and are widely used when automated forecasts are required for numerous series [8].

In addition to the smoothing parameters α and β^* (both ranging from 0 to 1, akin to Holt's method), this approach introduces a damping parameter $0 < \phi < 1$:

$$\hat{y}_{t+h|t} = \ell_t + (\phi + \phi^2 + \cdots + \phi^h)b_t \tag{2.33}$$

$$\ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + \phi b_{t-1}) \tag{2.34}$$

$$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)\phi b_{t-1} \tag{2.35}$$

When $\phi = 1$, the method mirrors Holt's linear approach. However, for values between 0 and 1, ϕ moderates the trend towards a constant value in the future. As $h \to \infty$, the forecasts converge to $\ell_T + \frac{\phi b_T}{1-\phi}$ for any $0 < \phi < 1$. Consequently, short-term forecasts exhibit a trend while long-term forecasts stabilize. In practice, ϕ seldom falls below 0.8 due to its substantial impact at lower values. Values of ϕ close to 1 render a damped model indistinguishable from a non-damped one [17].

2.2.3 Holt-Winters' seasonal method

Holt and Winters expanded upon Holt's method to accommodate seasonal patterns. The Holt-Winters seasonal method involves a forecast equation and three smoothing equations — one each for the level ℓ_t, the trend b_t, and the seasonal component s_t, each with respective smoothing parameters α, β^*, and γ. Here, m denotes the frequency of seasonality [17].

This method offers two variations depending on the nature of the seasonal component. The additive method is suitable when seasonal variations remain relatively constant throughout the series, whereas the multiplicative method is preferable when seasonal variations change proportionally to the series level. In the additive method, the seasonal component is expressed in absolute terms, reflecting the scale of the observed series. Consequently, the series is seasonally adjusted by subtracting the seasonal component in the level equation. Within each year, the seasonal component approximately sums to zero. Conversely, in the multiplicative method, the seasonal component is expressed in relative terms and the series is seasonally adjusted by dividing through by the seasonal component. In this case, the seasonal component sums to approximately m within each year [3].

2.2.3.1 Holt-Winters' additive method

The component form for the additive method is as follows:

$$\hat{y}_{t+h|t} = \ell_t + hb_t + s_{t+(h-m(k+1))}, \tag{2.36}$$

$$\ell_t = \alpha\left(y_t - s_{t-m}\right) + (1-\alpha)\left(\ell_{t-1} + b_{t-1}\right), \tag{2.37}$$

$$b_t = \beta^*\left(\ell_t - \ell_{t-1}\right) + (1-\beta^*)b_{t-1}, \tag{2.38}$$

$$s_t = \gamma\left(y_t - \ell_{t-1} - b_{t-1}\right) + (1-\gamma)s_{t-m}, \tag{2.39}$$

where k represents the integer part of $(h-1)/m$, ensuring that the estimates of the seasonal indices used for forecasting come from the final year of the sample [18].

The level equation signifies a weighted average between the seasonally adjusted observation $(y_t - s_{t-m})$ and the non-seasonal forecast $(\ell_{t-1} + b_{t-1})$ for time t. The trend equation mirrors Holt's linear method. The seasonal equation represents a weighted average between the current seasonal index $(y_t - \ell_{t-1} - b_{t-1})$ and the seasonal index of the same season last year (i.e., m periods ago).

The equation for the seasonal component can be alternatively expressed as:

$$s_t = \gamma^*(y_t - \ell_t) + (1-\gamma^*)s_{t-m}. \tag{2.40}$$

Substituting ℓ_t from the smoothing equation for the level into the seasonal component's equation yields:

$$s_t = \gamma^*(1-\alpha)(y_t - \ell_{t-1} - b_{t-1}) + [1 - \gamma^*(1-\alpha)]s_{t-m}, \tag{2.41}$$

which aligns with the specified smoothing equation for the seasonal component, where $\gamma = \gamma^*(1-\alpha)$. The typical parameter constraint is $0 \leq \gamma^* \leq 1$, translating to $0 \leq \gamma \leq 1 - \alpha$ [18].

2.2.3.2 Holt-Winters' multiplicative method

The component form for the multiplicative method is outlined as follows:

$$\hat{y}_{t+h|t} = (\ell_t + hb_t)s_{t+(h-m(k+1))} \tag{2.42}$$

$$\ell_t = \alpha y_t s_{t-m} + (1-\alpha)(\ell_{t-1} + b_{t-1}) \tag{2.43}$$

$$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \tag{2.44}$$

$$s_t = \gamma y_t(\ell_{t-1} + b_{t-1}) + (1-\gamma)s_{t-m} \tag{2.45}$$

Similar to the additive method, k represents the integer part of $(h-1)/m$, ensuring that the estimates of the seasonal indices used for forecasting come from the final year of the sample [18].

The level equation indicates a weighted average between the seasonally adjusted observation $\alpha y_t s_{t-m}$ and the non-seasonal forecast $(\ell_{t-1} + b_{t-1})$ for time t. The trend equation remains consistent with Holt's linear method. The seasonal equation now signifies a weighted average between the current observation y_t and the seasonal index of the same season last year, with the additional multiplication by the level and trend components.

CHAPTER **3**

Machine Learning and Deep Learning approaches

3.1 Gaussian mixture model

The Gaussian mixture model (GMM) is a weighted linear combination of multiple Gaussian components, which is often used to solve the problem that data contains multiple different distributions. GMM can smoothly approximate the density distribution of arbitrary shapes and can be applied to complicated object modeling [20].

Definition 3.1.1. A Gaussian Mixture Model is a parametric probability density function represented as a weighted sum of Gaussian component densities. GMMs are commonly used as a parametric model of the probability distribution of continuous measurements or features in a biometric system, such as vocal tract related spectral features in a speaker recognition system [21].

A Gaussian mixture model is a weighted sum of M component Gaussian densities as given by the equation,

$$p(x \mid \lambda) = \sum_{i=1}^{M} w_i g\left(x \mid \mu_i, \Sigma_i\right), \tag{3.1}$$

where x is a D-dimensional continuous-valued data vector (i.e. measurement or features), $w_i = 1, \ldots, M$, are the mixture weights, and $g\left(x \mid \mu_i, \Sigma_i\right)$, $i = 1, \ldots, M$ are the component Gaussian densities. Each component density is a D-variate Gaussian function of the form,

$$g\left(x \mid \mu_i, \Sigma_i\right) = \frac{1}{(2\pi)^{D/2}\left|\Sigma_i\right|^{1/2}} \exp\left\{-\frac{1}{2}\left(x-\mu_i\right)' \Sigma_i^{-1}\left(x-\mu_i\right)\right\}, \tag{3.2}$$

with mean vector μ_i and covariance matrix Σ_i. The mixture weights satisfy the constraint that $\sum_{i=1}^{M} w_i = 1$.

The complete Gaussian mixture model is parameterized by the mean vectors, covariance matrices, and mixture weights from all component densities. These parameters are collectively represented by the notation,

$$\lambda = \{w_i, \mu_i, \Sigma_i\} \quad i = 1, \ldots, M \tag{3.3}$$

3.1.1 Parameter estimation

Given training vectors and a GMM configuration, the parameters, λ, are estimated which, in some sense, best match the distribution of the training feature vectors. There are several techniques available for estimating the parameters of a GMM. By far the most popular and well-established method is maximum likelihood estimation [21].

Maximum likelihood estimation aims to find the model parameters λ which maximize the likelihood of the GMM $p(X \mid \lambda)$ given the training data. For a sequence of T training vectors $X = \{x_1, \ldots, x_T\}$, the GMM likelihood, assuming independence between the vectors, can be written as,

$$p(X \mid \lambda) = \prod_{t=1}^{T} p\left(x_t \mid \lambda\right). \tag{3.4}$$

This expression is a non-linear function of the parameters, meaning the direct maximization is not possible. However, ML parameter estimates can be obtained iteratively using a special case of the expectation-maximization (EM) algorithm [22].

The basic idea of the EM algorithm is, beginning with an initial model λ, to estimate a new model $\bar{\lambda}$, such that $p(X \mid \bar{\lambda}) \geq p(X \mid \lambda)$. The new model then becomes the initial model for the next iteration and the process is repeated until some convergence threshold is reached. On each EM iteration, the following re-estimation formulas are used which guarantee a monotonic increase in the model's likelihood value,

$$\bar{w_i} = \frac{1}{T} \sum_{t=1}^{T} \Pr\left(i \mid x_t, \lambda\right), \tag{3.5}$$

$$\overline{\mu}_i = \frac{\sum_{t=1}^{T} \Pr\left(i \mid x_t, \lambda\right) x_t}{\sum_{t=1}^{T} \Pr\left(i \mid x_t, \lambda\right)}, \tag{3.6}$$

$$\bar{\sigma}_i^2 = \frac{\sum_{t=1}^{T} \Pr\left(i \mid x_t, \lambda\right) x_t^2}{\sum_{t=1}^{T} \Pr\left(i \mid x_t, \lambda\right)} - \bar{\mu}_i^2, \tag{3.7}$$

where σ_i^2, x_t, and μ_i refer to arbitrary elements of the vectors ${\sigma_i}^2, x_t$, and μ_i, respectively.

The a posteriori probability for component i is given by:

$$\Pr(i \mid x_t, \lambda) = \frac{w_i g(x_t \mid \mu_i, \Sigma_i)}{\sum_{k=1}^{M} w_k g(x_t \mid \mu_k, \Sigma_k)}. \tag{3.8}$$

3.1.2 Anomaly detection

The proposed adaptation of the GMM model [23] initially starts with splitting data into bins. The selection of bins on the data's intrinsic characteristics and the analyst's discretion. For each bin m in M, the data observations are represented as s_m, constituting a set of N observations. Mathematically, this can be expressed as:

$$s_m = \{x_1, x_2, ..., x_N\}, \tag{3.9}$$

where x_i represents the i-th observation in the m-th bin.

Gaussian Mixture Models are constructed for each bin m, denoted as G_m, forming the set $G = \{G_1, G_2, ..., G_M\}$. The GMMs capture the underlying distribution of data within each bin and are crucial for modeling seasonal patterns.

It is important to note that the complexity of the GMMs can vary across bins. Some bins may require only one Gaussian component to adequately model the data distribution, while others may necessitate multiple Gaussians to capture higher variability. This variation in complexity is essential for accurately representing the diverse patterns present in the network traffic data. This systematic approach, rooted in mathematical principles, provides a robust framework for effectively modeling time series data.

To enhance the discriminative power between outliers and low-probability data points, a strategy is employed to amplify this discrepancy. Specifically, by applying the natural logarithm function to the probabilities associated with the data points, the relative differences in probability values are magnified. This logarithmic transformation serves to augment the contrast between outliers and data points with comparatively lower probabilities, thereby facilitating more effective discrimination between anomalous and regular observations. The anomaly score is then defined as follows:

$$AS_x = (\log(p(x)))^{2f}, \tag{3.10}$$

where AS_x is the anomaly score, $p(x)$ is the probability density function of the data point in x, and f is some factor used to scale the log of probabilities. Increasing f increases the difference between the scores for the normal and anomalous data points [23].

Gaussian Mixture Models operate as unsupervised clustering algorithms, eliminating the necessity for human intervention in determining the optimal number of mixture components. An additional advantage lies in their inherent capability to incorporate the seasonal characteristics inherent in the dataset. This intrinsic feature enables GMMs to capture and model the seasonal patterns present in the data effectively, contributing to their robustness in anomaly detection tasks.

3.2 Isolation Forest

Isolation Forest emerges as an unsupervised anomaly detection approach introduced by Prof. Zhi-hua Zhou et al. in 2008 [5]. Later, Ding et al. implemented an algorithm for anomaly detection of streaming data based on the isolated forest algorithm and the sliding window technique [24]. The design of Isolation Forest is attributed to two special characteristics of abnormal data: (i) anomaly is defined as the outlier that is easily isolated, and (ii) the amount of abnormal data is very small. In the isolated forest, a data set is recursively randomly partitioned until each point is isolated. The abnormal point is closer to the root node of the isolation tree (iTree), while the normal point is far from the root node of iTree [25].

Definition 3.2.1. Isolation Tree (iTree). Assuming T is a node of an isolation tree. T is either a leaf node without sub-nodes or an internal node with only two sub-nodes (T_l, T_r). Each step of the split consists of a feature q and a split value p. The data with $q < p$ will be divided into T_l while the data with $p < q$ will be divided into T_r.

At first, the time series training data set X is divided into the data blocks having the same time interval and the same number of instances in a data block, which are named window data block and can be described in detail as follows. $X = \{X_1, X_2, \ldots . X_i, \ldots\}$ with the follow forms:

$$\begin{aligned} Z_1 &= z_1, z_2, \ldots, z_M \\ Z_2 &= z_{M+1}, z_{M+2}, \ldots, z_{2M} \\ &\vdots \\ Z_i &= z_{(i-1)M+1}, z_{(i-1)M+2}, \ldots, z_{iM}, \end{aligned} \tag{3.11}$$

where M denotes the size of the sliding window, Z_i the i-th time series data block. The main task of anomaly detection for streaming data is to examine whether anomalous instances exist based on the anomaly detector.

When processing time series data, the initial step involves feeding it into a sliding window with a predetermined size. Determining the size of the sliding window poses the first challenge during the training stage of the detector. Unfortunately, there is no specific theoretical guideline for this parameter, and it is typically selected based on prior knowledge and experimentation.

Once several windows of data are collected, an initial detector is trained using the iForest Algorithm 1. In the subsequent test stages, each instance within the sliding window undergoes examination by the anomaly detector to ascertain whether it qualifies as an anomaly based on its anomaly score.

Upon completing the instances within a sliding window, a statistical analysis ensues. If the anomaly rate within a sliding window's data falls below the predefined threshold, it indicates that no concept drift has occurred. Consequently, the trained anomaly detector remains unchanged.

However, if a concept drift occurs, indicating a discrepancy between the previous training and the current data concepts, immediate action is necessary. Failure to promptly update the trained detector may result in degraded detection performance. Therefore, to maintain the efficacy of anomaly detection, it becomes imperative to either retrain the model or update it based on the incoming dataset [25].

For the time series dataset X_i, the initial anomaly detection model is built, at first, the iTrees in terms of the bootstrap sampling from the Z_i is built. The ensemble detection model E is composed of L iTrees, namely, $E = \{E_1, E_2, \dots E_L\}$, which is built from the data in i-th sliding window.

Algorithm 1: `iForest`(X, L, N)

```
Input  : $X$ - input dataset, $L$ - number of trees, $N$ -
         subsampling size in the sliding window.
Output : Set of $L$ iTrees.

  // Initialize Forest
1 E ← {}
  // Set iTree height
2 $h \leftarrow \lceil \log_2 N \rceil$
3 for $i \leftarrow 1$ to $L$ do
4     $X' \leftarrow$ sample$(X, N)$;
5     E ← E ∪ iTree$(X', 0, h)$;
6 end
7 return $E$
```

An Isolation Forest is a collection of isolation trees, which are constructed by randomly selecting attributes and their corresponding values. At each node within these isolation trees, the dataset is partitioned based on randomly chosen attributes and their associated values. This random selection process includes the choice of attributes and the split value, which is determined randomly between the minimum and maximum values of the selected attribute.

Anomalies in the dataset are typically identified as instances with attribute values significantly divergent from those of normal instances. These anomalous instances tend to be more easily separated during the isolation process and are thus located closer to the root of the trees, facilitating their division.

To address the inherent randomness introduced during the construction of the Isolation Forest, the average depth of instances across multiple isolation trees is computed. This average depth serves as the anomaly score for each instance, with lower scores indicating a higher likelihood of being an anomaly. The creation of the isolation trees is a critical aspect of the iForest algorithm, and detailed procedures for generating iTree are outlined in this work [5].

In the realm of time series data analysis, the phenomenon known as concept drift frequently arises. Detecting concept drift promptly is essential for effective data processing. To achieve this, the anomaly rate is computed once all instances within a sliding window have been identified. Initially, a ranking of all instances in the sliding window is established. Each instance is assigned a score based on its probability of being an anomaly, determined by its average depth within the isolation trees of the forest. The anomaly score, denoted as $AS(x, N)$, is then calculated using the equation:

$$AS(x, N) = 2^{\frac{E(h(x))}{c(N)}}, \tag{3.12}$$

where

$$E(h(x)) = \frac{1}{L}\sum_{i=1}^{L} h_i(x), \tag{3.13}$$

and N denotes the subsampling size. $h_i(x)$ denotes the length of the i-th $iTree$, $E(h(x))$ is the average of $h(x)$ from a collection of iTrees, and c(N) is the average of h(x) given N.

Because iTree has the same structure as the Binary Search Tree (BST), the estimation of average $h(x)$ for external nodes is the same as that of the unsuccessful searches in BST. The method to calculate the average path length of unsuccessful searches in BST is given as:

$$c(n) = \begin{cases} 2H(n-1) - 2(n-1)/n & \text{for } n > 2 \\ 1 & \text{for } n = 2 \\ 0 & \text{otherwise} \end{cases}, \tag{3.14}$$

where $H(i)$ is the harmonic number that can be estimated by $\ln(i) + \gamma(\gamma$ represents Euler's constant), $c(n)$ denotes the average of $h(x)$ given n, which is used to normalize $h(x)$ [25].

The statistical outcome provides insights into the anomaly rate within the sliding window. If the anomaly rate within the current window exceeds the predefined threshold, indicating a deviation from the expected anomaly rate based on prior knowledge, a concept drift occurs in the streaming data. Consequently, the existing anomaly detector becomes ineffective, prompting its immediate replacement with a newly trained detector.

Considering the concept of drift, employing a fixed sliding window size proves inadequate. If the window size is excessively large, the model trained on it fails to accurately capture the concept drift. Conversely, if the window size is too small, insufficient data is available to construct a precise model. Unfortunately, no standardized method exists for setting this parameter, typically, it is determined through trial and error specific to the application under study.

One of the key advantages of Isolation Forests is their ability to handle anomalies in various domains without requiring extensive parameter tuning. Moreover, they surpass in detecting anomalies in large datasets, where other methods might struggle due to computational complexity.

However, Isolation Forests also have limitations. They may not perform optimally when anomalies are densely populated or when the data distribution is highly imbalanced.

3.3 Long-short term memory network

The application of artificial neural networks (ANNs) has emerged as a powerful tool for tackling a wide array of complex problems, owing to their capability to comprehend non-linear systems and discern the intrinsic behaviors and dependencies inherent in the data they receive.

In the realm of time series analysis, the integration of deep learning techniques, particularly through such architectures as Long Short-Term Memory (LSTM) networks, represents a relatively recent yet promising avenue. The flexibility of deep learning renders it advantageous for temporal data analysis, notably offering the potential to model intricate and non-linear temporal patterns without the need to make assumptions about functional forms. This paradigm shift holds the promise of revolutionizing nonstatistical forecasting methods.

In this chapter, the principles and applications of LSTM networks for anomaly detection in time series data are explored, delving into their capabilities in deciphering complex temporal patterns and identifying anomalous behavior amidst sequential data. Through a comprehensive examination of LSTM-based anomaly detection techniques, the potential of these neural network architectures in addressing the challenges posed by anomalies in time series datasets is illuminated.

3.3.1 Artificial neural network

Artificial neural network architectures have been devised based on established models of biological nervous systems, drawing inspiration from the intricate workings of the human brain. These networks comprise computational units known as artificial neurons, which is visualized in Figure 3.1, which serve as simplified counterparts to their biological counterparts. These artificial neurons typically exhibit non-linearity and offer continuous outputs, executing basic functions such as signal aggregation from input sources, integration based on predefined operational functions, and generation of responses guided by their inherent activation functions.

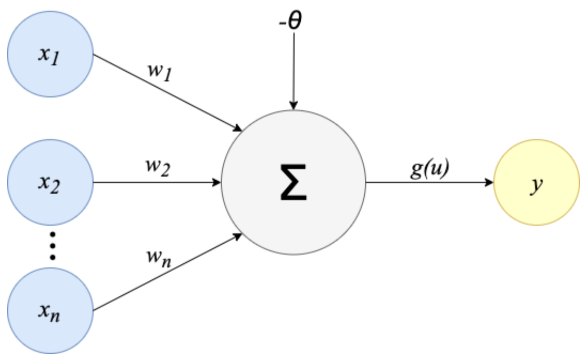

Figure 3.1: The artificial neuron structure.

An artificial neuron is comprised of seven fundamental components.

Definition 3.3.1. Input signals $(x_1, x_2, ..., x_n)$ represent the samples or signals originating from the external environment, embodying the values assumed by the variables pertinent to a specific application. These input signals are typically normalized to bolster the computational efficiency of learning algorithms [26].

Definition 3.3.2. Synaptic weights $(w_1, w_2, ..., w_n)$ are the values assigned to each input variable, facilitating the quantification of their importance concerning the neuron's functionality [26].

Definition 3.3.3. The linear aggregator $(\sum)$ accumulates all input signals weighted by the synaptic weights to generate an activation voltage [26].

Definition 3.3.4. The activation threshold or bias (θ) is a parameter used to specify the requisite threshold for the output produced by the linear aggregator to trigger a response from the neuron [26].

Definition 3.3.5. The activation potential (u) is determined by the disparity between the linear aggregator and the activation threshold. A positive value of u $(u \geq \theta)$ results in excitatory potential, whereas a negative value indicates inhibitory potential [27].

Definition 3.3.6. The activation function (g) constrains the neuron output within a reasonable range of values, defined by its functional image. Common activation functions include sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU), each serving to introduce different properties into the network's decision-making process [27].

Definition 3.3.7. The output signal (y) denotes the final value generated by the neuron given a specific set of input signals, potentially serving as input for other sequentially interconnected neurons [26].

The synaptic weighting in the neural network is manifested within the artificial neuron through a collection of synaptic weights ($w_1, w_2, ..., w_n$). Likewise, the significance of each input signal (x_i) to the neuron is gauged by multiplying it with its corresponding synaptic weight (w_i), thereby weighting all incoming external information. Consequently, the output (u) of the artificial neuron is discerned as the weighted summation of its inputs.

As information flows through a network, neurons are organized into layers, forming a hierarchical structure that enables the network to learn complex patterns [7].

Layers in a neural network are akin to functional units, each serving a specific purpose in the information processing pipeline. The input layer acts as the entry point for data, receiving input signals from external sources and transmitting them to subsequent layers for processing. Subsequent layers, known as hidden layers, perform intricate computations on the input data, extracting features and representations relevant to the task at hand. The final layer of the network, known as the output layer, generates predictions or classifications based on the processed information.

As the network learns from data through a process called training, the synaptic connections between neurons, represented by synaptic weights, are adjusted to minimize the disparity between predicted and actual outputs. This optimization process, driven by mathematical techniques such as gradient descent and backpropagation, fine-tunes the network's parameters to improve its performance on the given task [7].

Through this iterative process of learning and optimization, artificial neural networks can model complex relationships in data, paving the way for advancements in various domains such as image recognition, natural language processing, and financial forecasting. In the following sections, the difference between feed-forward neural networks and recurrent neural networks will be provided, explaining the advantage of the second for time series anomaly detection use case.

3.3.2 Feed-forward and recurrent network architecture

Two primary categories of neural networks emerge feed-forward neural networks (FFN) and recurrent neural networks (RNN). FFNs, prevalent in neural network architectures, consist of layered nodes where data progresses unidirectionally, from the input layer to the output layer. Conversely, RNNs introduce feedback loops, enabling a bidirectional flow of information between network layers. This bidirectional communication fosters enhanced capacity for temporal analysis and dynamic sequence processing within RNNs. RNNs are a type of artificial neural network that were specifically designed to process sequential data or time series data [7].

3.3.2.1 Feed-forward neural networks

Feed-forward neural networks represent the prevailing archetype among neural networks. Structured as layers of nodes, these networks facilitate unidirectional data flow, progressing seamlessly from the input layer through intermediary nodes to the output layer. Each node undertakes a straightforward computation, transmitting its output to subsequent nodes within the layer. At the network's culmination lies the output layer, generating the network's predictive outcome.

The prevailing method for training neural networks is the error backpropagation algorithm, which employs gradient descent to adjust the weights within multilayer networks. This process unfolds iteratively, commencing from the output layer and proceeding backward toward the input layer. An essential condition is that the activation function of each neuron remains differentiable. Typically, the weights of a feed-forward neural network are initialized with small, normalized random values alongside bias values. Subsequently, error backpropagation processes all training samples through the neural network, calculating the input and output of each unit across all hidden and output layers [26].

Feed-forward neural networks are characterized by their absence of loops and full connectivity, which is shown in Figure 3.2. In such networks, every neuron within a layer provides input to each neuron in the subsequent layer, while none of the weights convey input to neurons in the preceding layers.

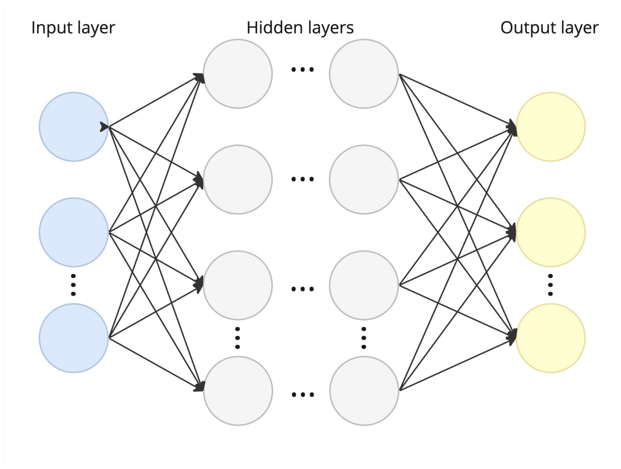

Figure 3.2: Feed-forward Architecture.

The primary advantage of feed-forward neural networks lies in their simplicity and efficiency, rendering them adept for various tasks. Their straightforward architecture facilitates ease of training, making them an appealing option for numerous applications. However, feed-forward neural networks exhibit limitations in handling sequential data compared to recurrent neural networks, which surpass tasks requiring temporal analysis and dynamic sequence processing [7].

3.3.2.2 Recurrent neural networks

Recurrent neural networks (RNNs) constitute a neural network variant characterized by feedback loops, enabling bidirectional information flow across network layers depicted in Figure 3.3. This unique architecture empowers RNNs to effectively process sequential data, speech recognition, and machine translation.

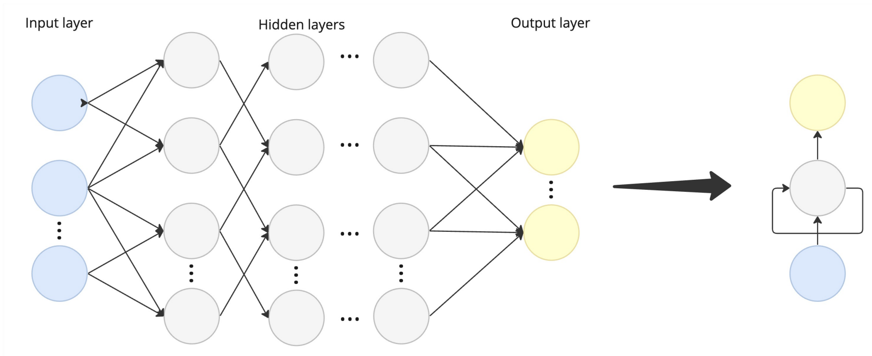

Figure 3.3: Recurrent Architecture.

Typically structured akin to feed-forward neural networks, RNNs comprise stacked layers interconnected in a loop configuration, facilitating feedback of each layer's output to its own input. This recurrent connectivity enables the network to discern temporal relationships inherent in the data [26].

Recurrent neural networks (RNNs) exhibit dynamism as they maintain an internal state at every time step during classification. This is attributed to the presence of circular connections between neurons across different layers and the availability of self-feedback connections. Such feedback mechanisms empower RNNs to convey information from prior events to ongoing processing stages, thereby encoding a memory of temporal sequences within time series data.

Nonetheless, training RNNs can pose challenges, and their computational demands may necessitate substantial resources for implementation. Another limitation of simple RNNs is their short-term memory, which restricts their ability to retain information over long sequences. To overcome this, more advanced RNN variants have been developed, including Long Short-Term Memory [28].

3.3.3 LSTM architecture

Standard RNNs encounter limitations in bridging more than 5–10 time steps due to the tendency of back-propagated error signals to either escalate or diminish with each successive time step. Across numerous time steps, this phenomenon results in the error signals either escalating excessively, leading to oscillating weights, or diminishing to such an extent that learning becomes impractically slow or fails altogether. The LSTM model is a powerful recurrent neural system specially designed to overcome the exploding/vanishing gradient problems that typically arise when learning long-term dependencies, even when the minimal time lags are very long [28].

The LSTM architecture consists of a set of recurrently connected subnetworks, known as memory blocks. The idea behind the memory block is to maintain its state over time and regulate the information flow through non-linear gating units. Figure 3.4 displays the architecture of the LSTM block, which involves the gates, the input signal $x^{(t)}$, the output $y^{(t)}$, the activation functions, and peephole connections. The output of the block is recurrently connected back to the block input and all of the gates [1].

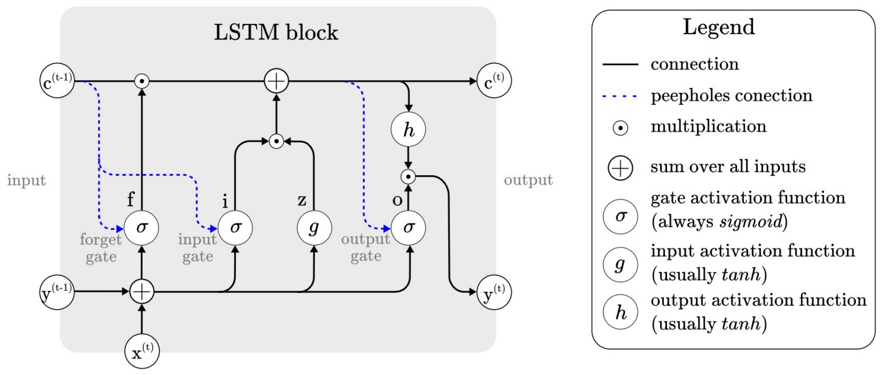

Figure 3.4: Architecture of a typical LSTM block [1].

To illustrate the functionality of the LSTM model, consider a network consisting of N processing blocks and M inputs. The forward pass in this recurrent neural system unfolds as follows:

Block input This step is devoted to updating the block input component, which combines the current input $x^{(t)}$ and the output of that LSTM unit $y^{(t-1)}$ in the last iteration. This can be done as depicted below:

$$z^{(t)} = g\left(W_z x^{(t)} + R_z y^{(t-1)} + b_z\right), \tag{3.15}$$

where W_z and R_z are the weights associated with $x^{(t)}$ and $y^{(t-1)}$, respectively, while b_z stands for the bias weight vector [1].

Input gate During the input gate step, the update process involves combining the current input $x^{(t)}$, the output of that LSTM unit $y^{(t-1)}$ and the cell value $c^{(t-1)}$ from the previous iteration. This operation is formulated as:

$$i^{(t)} = \sigma\left(W_i x^{(t)} + R_i y^{(t-1)} + p_i \odot c^{(t-1)} + b_i\right), \tag{3.16}$$

where σ represents the sigmoid function applied element-wise to a vector, $\odot$ denotes point-wise multiplication of two vectors, W_i, R_i and p_i are the weights associated with $x^{(t)}, y^{(t-1)}$ and $c^{(t-1)}$, respectively, while b_i represents for the bias vector associated with this component [7].

In the preceding steps, the LSTM layer determines the information to retain within the network's cell states $c^{(t)}$. This included the selection of the candidate values $z^{(t)}$ which may potentially be incorporated into the cell states, and the activation values $i^{(t)}$ of the input gates.

Forget gate In this step, the LSTM unit determines which information to discard from its previous cell states $c^{(t-1)}$. Consequently, the activation values $f^{(t)}$ of the forget gates at time step t are computed based on the current input $x^{(t)}$, the outputs $y^{(t-1)}$ and the state $c^{(t-1)}$ of the memory cells at the previous time step $(t-1)$, the peephole connections, and the bias terms b_f of the forget gates [1]. This calculation is expressed as:

$$f^{(t)} = \sigma\left(W_f x^{(t)} + R_f y^{(t-1)} + p_f \odot c^{(t-1)} + b_f\right), \tag{3.17}$$

where W_f, R_f and p_f are the weights associated with $x^{(t)}, y^{(t-1)}$ and $c^{(t-1)}$, respectively, while b_f denotes for the bias weight vector.

Cell This step computes the cell value, combining the block input $z^{(t)}$, the input gate $i^{(t)}$ and the forget gate $f^{(t)}$ values, with the previous cell value [1]. This computation is represented as:

$$c^{(t)} = z^{(t)} \odot i^{(t)} + c^{(t-1)} \odot f^{(t)}. \tag{3.18}$$

Output gate This step calculates the output gate, which combines the current input $x^{(t)}$, the output of that LSTM unit $y^{(t-1)}$ and the cell value $c^{(t-1)}$ in the last iteration. This can be done as depicted below:

$$o^{(t)} = \sigma\left(W_o x^{(t)} + R_o y^{(t-1)} + p_o \odot c^{(t)} + b_o\right), \tag{3.19}$$

where W_o, R_o and p_o are the weights associated with $x^{(t)}, y^{(t-1)}$ and $c^{(t-1)}$, respectively, while b_o denotes for the bias weight vector [1].

Block output Finally, the block output is calculated, which combines the current cell value $c^{(t)}$ with the current output gate value as follows:

$$y^{(t)} = g\left(c^{(t)}\right) \odot o^{(t)}. \tag{3.20}$$

In the above steps, σ, g, and h denote point-wise non-linear activation functions [1]. The logistic sigmoid $\sigma(x) = \frac{1}{1+e^{1-x}}$ is used as a gate activation function, while the hyperbolic tangent $g(x) = h(x) = \tanh(x)$ is often used as the block input and output activation function.

3.3.4 Model training

The LSTM model employs full gradient training to adjust the learnable parameters (weights) within the network. Specifically, Backpropagation Through Time is utilized to compute the weights connecting the various components in the network [29]. Consequently, during the backward pass, pass, the cell state $c^{(t)}$ receives gradients from $y^{(t)}$ as well as the next cell state $c^{(t+1)}$. Those gradients are accumulated before being backpropagated to the current layer.

In the final iteration T, the change $\delta_y^{(T)}$ corresponds to the network error $\partial E / y^{(T)}$, where E represents the loss function. Otherwise, $\delta_y^{(t)}$ represents the vector of delta values passed down $\Delta^{(t)}$ from the layer above, including the recurrent dependencies. This can be mathematically expressed as follows:

$$\delta_y^{(t)} = \Delta^{(t)} + R_z^T \delta_z^{(t+1)} + R_i^T \delta_i^{(t+1)} + R_f^T \delta_f^{(t+1)} + R_o^T \delta_o^{(t+1)}. \tag{3.21}$$

In the second step, the changes in the parameters associated with the gates and the memory cell are calculated as:

$$\delta_o^{(t)} = \delta_y^{(t)} \odot h\left(c^{(t)}\right) \odot \sigma'\left(\hat{o}^{(t)}\right) \tag{3.22}$$

$$\begin{aligned} \delta_c^{(t)} = \delta_y^{(t)} \odot o^{(t)} \odot h'\left(c^{(t)}\right) + p_o \odot \delta_o^{(t)} + \\ + p_i \odot \delta_i^{(t+1)} + p_f \odot \delta_f^{(t+1)} + \delta_c^{(t+1)} \odot f^{(t+1)} \end{aligned} \tag{3.23}$$

$$\delta_{f^{(t)}} = \delta_c^{(t)} \odot c^{(t-1)} \odot \sigma'\left(\hat{f}^{(t)}\right) \tag{3.24}$$

$$\delta_{i^{(t)}} = \delta_c^{(t)} \odot z^{(t)} \odot \sigma'\left(\hat{i}^{(t)}\right) \tag{3.25}$$

$$\delta_{z^{(t)}} = \delta_c^{(t)} \odot i^{(t)} \odot g'\left(\hat{z}^{(t)}\right), \tag{3.26}$$

where $\hat{o}^{(t)}, \hat{i}^{(t)}, \hat{z}^{(t)}$ and $\hat{f}^{(t)}$ represent the raw values associated with the output gate, input gate, block input, and forget gate, respectively, before transforming by the corresponding transfer function.

The delta values for the inputs are only required if there is a layer below that needs to be trained, thus:

$$\delta_x^{(t)} = W_z^T \delta_z^{(t)} + W_i^T \delta_i^{(t)} + W_f^T \delta_f^{(t)} + W_o^T \delta_o^{(t)}. \tag{3.27}$$

Finally, the gradients for the weights are calculated as follows:

$$\begin{aligned}
\delta_{W_*} &= \sum_{t=0}^{T} \delta_*^{(t)} \otimes x^{(t)} & \delta_{p_i} &= \sum_{t=0}^{T-1} c^{(t)} \odot \delta_i^{(t+1)} \\
\delta_{R_*} &= \sum_{t=0}^{T-1} \delta_*^{(t+1)} \otimes y^{(t)} & \delta_{p_f} &= \sum_{t=0}^{T-1} c^{(t)} \odot \delta_f^{(t+1)} \\
\delta_{b_*} &= \sum_{t=0}^{T} \delta_*^{(t)} & \delta_{p_o} &= \sum_{t=0}^{T} c^{(t)} \odot \delta_o^{(t)},
\end{aligned} \tag{3.28}$$

such that $\otimes$ represents the outer product of two vectors, whereas $*$ can be any component associated with the weights: the block input $\hat{z}$, the input gate $\hat{i}$, the forget gate $\hat{f}$ or the output gate $\hat{o}$ [1].

3.3.5 LSTM-based anomaly detection

In the context of anomaly detection, LSTM networks are utilized to learn the underlying patterns in time series data and identify deviations from the expected behavior.

The time series data is represented as input sequences $(x^{(1)}, x^{(2)}, ..., x^{(T)})$, where T denotes the total number of time steps. Each $x^{(t)}$ represents the feature vector at time step t. The LSTM model is trained on instances of the time series data to learn the temporal patterns and dependencies. The training process involves adjusting the weights and biases of the LSTM network using backpropagation through time (BPTT) to minimize the prediction error. Once the LSTM model is trained, it is used to predict the next data point in the sequence based on the preceding data points. The predicted values $\hat{y}^{(t)}$ are compared with the actual values $y^{(t)}$.

The residual error, or the difference between the predicted and actual values, is computed at each time step and then is used as an anomaly score:

$$\text{Residual}(t) = AS(t) = |y^{(t)} - \hat{y}^{(t)}| \tag{3.29}$$

A threshold ϵ is set to distinguish between normal and anomalous behavior. Data points with residual errors exceeding the threshold are classified as anomalies:

$$\text{Anomaly}(t) = \begin{cases} 1, & \text{if AS}(t) > \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{3.30}$$

The performance of the LSTM-based anomaly detection model is evaluated using appropriate metrics such as precision, recall, F1-score, and area under the receiver operating characteristic (ROC) curve. These metrics assess the model's ability to accurately detect anomalies while minimizing false positives and false negatives.

One disadvantage of using LSTM for anomaly detection is its computational complexity, especially when dealing with large-scale datasets. Training LSTM models requires significant computational resources and time, particularly when optimizing hyperparameters and fine-tuning the network architecture. Additionally, the training process may suffer from overfitting, where the model learns to memorize the training data rather than generalize to unseen examples, leading to poor performance on new data [1].

Another drawback is the interpretability of LSTM models. While LSTM networks surpass at capturing complex temporal patterns, understanding the underlying reasons for anomalous behavior detected by the model can be challenging. The black-box nature of deep learning models like LSTM makes it difficult to interpret the features or variables driving the anomaly detection decisions, limiting the model's explainability and interpretability [26].

Furthermore, LSTM-based anomaly detection may struggle with handling long-range dependencies and capturing subtle changes in the data. Standard LSTM architectures have been reported to have difficulty in remembering information over long time intervals, which can result in the model overlooking important contextual information leading to false positives or false negatives in anomaly detection.

Despite these challenges, ongoing research efforts focus on addressing these limitations and enhancing the effectiveness and efficiency of LSTM-based anomaly detection approaches. By exploring techniques such as model regularization, attention mechanisms, and architecture modifications, researchers aim to overcome the shortcomings of LSTM models and improve their performance in anomaly detection tasks.

Chapter **4**

Experiments

This chapter focuses on practically applying and comparing the methods researched. The techniques outlined in Chapters 2 and 3 are implemented, and ensemble models comprising both statistical and machine learning approaches are constructed and assessed. Specifically, statistical models including ARIMA and ES are employed, alongside ML models such as LSTM, iForest, and GMM.

Experiments are conducted on three distinct datasets. The synthetic dataset is partitioned into three segments, each represents a different anomaly type, to facilitate a more comprehensive evaluation of model performance.

The computations were performed on an Intel (R) Xeon(R) CPU running at 2.20 GHz with x86_64 ISO architecture. The processor supports both 32-bit and 64-bit operating modes and operates as a single-core processor with two threads. No additional hardware accelerators such as GPU were utilized. The system has 13 GB of RAM.

4.1 Used tools

Python was chosen as the primary programming language for implementation primarily due to its widespread adoption in the field of machine learning. This decision was made considering Python's extensive array of dedicated frameworks and libraries, which were utilized in the implementation. Such libraries and frameworks were used:

- **Pandas:** data manipulation control.
- **NumPy:** data manipulation and high-level math functions.
- **Matplotlib:** data visualization.
- **Seaborn:** data visualization.

- **Scikit-learn:** machine learning.
- **TensorFlow:** machine learning and artificial intelligence. Used for LSTM model implementation.
- **Statsmodels:** statistical models. Used for ARIMA and exponential smoothing implementation.
- **PyOD:** library for detecting anomalous objects and patterns.
- **pmdarima:** automatically discover the optimal order for an ARIMA model.
- **GutenTAG:** Synthetic data generator.

4.2 Datasets Overview

Each anomaly detection method and its parameters underwent testing on three distinct fully labeled datasets. The first dataset originates from Yahoo! Research [30], while the second dataset is sourced from the Numenta Anomaly Benchmark (NAB) corpus [31]. The final dataset is a synthetic one generated using the GutenTAG framework [32]. These datasets encompass both real-world and synthetic data, offering a diverse and comprehensive evaluation platform for assessing the efficacy of anomaly detection algorithms. Table 4.1 shows a number of data points and anomalies in each dataset.

Dataset	Number of Datapoints	Number of Anomalies
Yahoo S5	142100	466
NAB	22695	588
GutenTAG	100000	1667

Table 4.1: Basic statistics about used datasets.

4.2.1 Yahoo Finance Dataset

The Yahoo S5 Benchmark dataset serves as a valuable resource for evaluating anomaly detection algorithms, providing real and synthetic time series data with tagged anomaly points. This dataset facilitates the testing of detection accuracy across various anomaly types, including outliers and change points. The synthetic data includes time series with diverse characteristics such as trend, noise, and seasonality, while the real data comprises time series metrics from different Yahoo services [30].

Figure 4.1 illustrates the diverse anomalies present in the dataset. It is worth noting the dataset encompasses various anomaly types, particularly highlighting contextual anomalies. However, there is a significant prevalence of point anomalies characterized by notable deviations from established patterns. These anomalies lack explicit labeling according to their types but are rather marked as anomalous occurrences. Primarily, the observed anomalies manifest as global outliers, displaying substantial deviations from established norms. Moreover, a discernible seasonality pattern is evident, underscoring the necessity for preprocessing before applying analytical techniques.

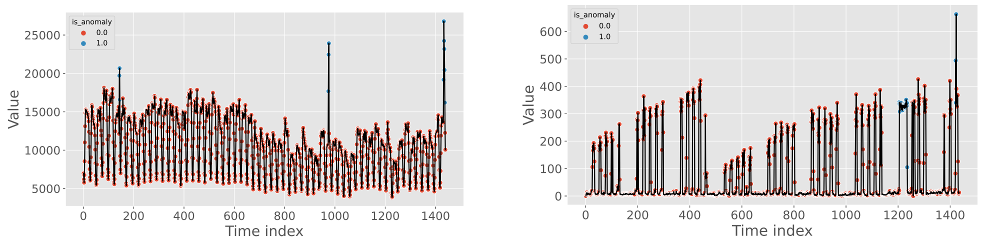

Figure 4.1: Example of anomalies in Yahoo Finance dataset.

4.2.2 Numenta Anomaly Benchmark Dataset

NAB, standing for Numenta Anomaly Benchmark, represents a cutting-edge benchmark designed specifically for assessing algorithms tailored for anomaly detection in streaming, real-time applications. This benchmark comprises over 50 labeled time series data files, encompassing both real-world and artificial datasets. The dataset predominantly consists of real-world data sourced from various domains, including AWS server metrics, Twitter volume, advertisement clicking metrics, and traffic data, among others [31].

For the evaluation of models, the machine temperature system failure dataset was utilized. This dataset contains temperature sensor data from an internal component of a large industrial machine. This dataset provides a practical scenario for evaluating the effectiveness of anomaly detection algorithms in identifying critical events and potential system failures in real-world applications.

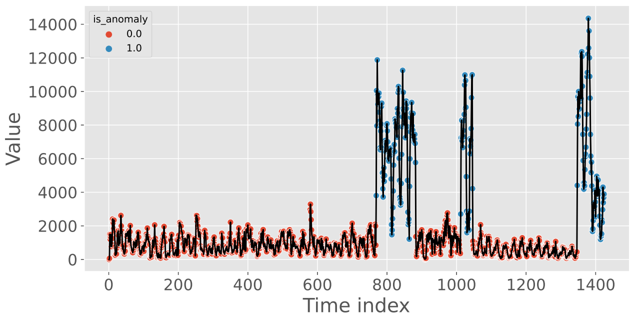

Figure 4.2: Example of anomalies in NAB dataset.

Furthermore, upon closer inspection of the dataset depicted in Figure 4.2, it becomes apparent that the anomalies within it primarily manifest as cohesive clusters of collective anomalies, rather than discrete individual anomalies. This observation underscores the significance of algorithms capable of detecting anomalies within such collective patterns.

4.2.3 Synthetic GutenTAG Dataset

GutenTAG is an extensible tool designed to generate time series datasets with and without anomalies. A GutenTAG time series consists of either a single (univariate) or multiple (multivariate) channels, each containing a base oscillation with anomalies at different positions and of varying kinds [32].

The tool was employed in this study to create a labeled dataset featuring various anomaly types. This approach aimed to improve the evaluation of selected algorithms by providing them with a dataset covering a wide range of anomaly scenarios. By incorporating anomalies of different types at different positions within the time series, the generated dataset facilitated a more comprehensive assessment of the algorithms' performance across diverse anomaly types.

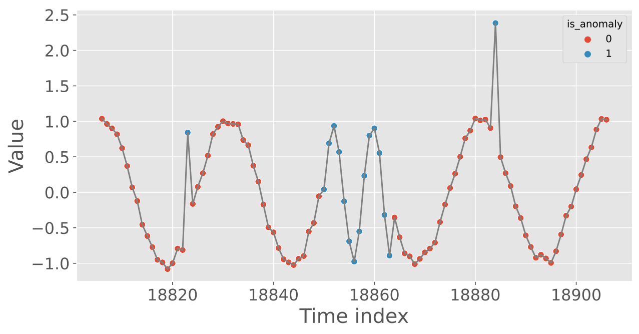

Figure 4.3: Example of anomalies in synthetic dataset.

The dataset in question contains all types of anomalies, each labeled according to its respective type. However, it is worth noting that the deviations from the established patterns are comparatively lower, making anomalies more challenging to detect compared to datasets like NAB or Yahoo Finance. This deliberate generation of anomalies with reduced deviation from the pattern serves a crucial purpose in this study. By creating anomalies that are harder to detect, the evaluation of selected algorithms becomes more rigorous and realistic. Figure 4.3 provides a visual representation of the generated anomalies.

4.3 Experiment Design

This chapter outlines the experiment design and methodology employed to evaluate the performance of various anomaly detection models. The experiment involves training and testing both statistical and machine learning models, including ARIMA, exponential smoothing, LSTM, Isolation Forest, and Gaussian Mixture Model.

In addition to evaluating individual models, six ensemble models were created by combining different combinations of statistical and machine learning approaches. These ensemble combinations include:

1. Gaussian Mixture Model with ARIMA (GMM_ARIMA).
2. Gaussian Mixture Model with Exponential Smoothing (GMM_ES).
3. Isolation Forest with ARIMA (iForest_ARIMA).
4. Isolation Forest with Exponential Smoothing (iForest_ES).
5. LSTM with ARIMA (LSTM_ARIMA).
6. LSTM with Exponential Smoothing (LSTM_ES).

By combining the anomaly scores from machine learning models with prediction errors from statistical models, these ensemble models aim to leverage the complementary strengths of both approaches to enhance anomaly detection performance.

4.3.1 Data preprocessing

In the preprocessing phase, each dataset underwent a series of transformations to ensure its suitability for modeling. Stationarity tests, as discussed in section 2.1.3.1, were conducted for each dataset to assess their temporal characteristics. These tests included the Augmented Dickey-Fuller (ADF) test and Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test. These tests provided insights into the presence of trends and seasonality within the data.

For datasets exhibiting non-stationarity, particularly the Yahoo Finance dataset, differencing was applied based on the results of the stationarity tests. Differencing involves computing the difference between consecutive observations to stabilize the mean and variance of the time series data. The differenced series is often more amenable to modeling, especially for methods like ARIMA which require stationarity.

Following differencing, the preprocessed data were standardized and normalized to ensure consistency and comparability across different datasets and models. Standardization involves transforming the data such that it has a mean of zero and a standard deviation of one, typically achieved using the formula:

$$z = \frac{x - \mu}{\sigma}, \tag{4.1}$$

where x represents the original data, μ is the mean, σ is the standard deviation, and z is the standardized value.

Normalization, on the other hand, scales the data to a fixed range, usually between 0 and 1, using the formula:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}, \tag{4.2}$$

where x' represents the normalized value, $\min(x)$ is the minimum value in the dataset, and $\max(x)$ is the maximum value.

These preprocessing techniques ensure that the data are appropriately transformed and standardized, making them suitable for consumption by a wide range of models, including machine learning algorithms like LSTM and Gaussian Mixture Models.

4.3.2 Configuration of evaluated detection methods

This section delves into the meticulous process of configuring and preparing the various anomaly detection methods employed in the study. Each method, whether statistical models like ARIMA and exponential smoothing or machine learning algorithms such as LSTM, Isolation Forest, and Gaussian Mixture Models, requires careful configuration to ensure optimal performance.

4.3.2.1 ARIMA

ARIMA configuration was selected with *pmdarima* Python library. It is utilized for the automated selection of optimal model parameters, minimizing criteria such as the Akaike Information Criterion.

Definition 4.3.1. The Akaike Information Criterion (AIC) quantifies the relative quality of statistical models, balancing goodness of fit with model complexity. It is calculated as:

$$AIC = -2 \times \log(L) + 2k, \tag{4.3}$$

where L is the likelihood of the model given the data, and k is the number of parameters [33].

Once the optimal parameters are determined, the ARIMA or seasonal ARIMA models are fitted to the training data, utilizing maximum likelihood estimation. Once trained, the models are applied to the test dataset for prediction. During the prediction phase, the models perform one-step-ahead forecasting without updating the parameters based on the observed outcomes. This approach ensures that the models maintain consistency and do not adapt to the test data, preserving their generalization capabilities and avoiding overfitting [15]. The error of the prediction and anomaly score is computed for each data point as described in section 2.1.1.2.

4.3.2.2 Exponential smoothing

Parameter selection for the Holt-Winters ES model involves determining the smoothing parameters. Additionally, the seasonal period m must be specified to capture the recurring patterns in the data.

The Holt-Winters ES model is trained using historical time series data to estimate the level, trend, and seasonal components. The ‘statsmodels‘ library provides robust functionality for implementing the Holt-Winters ES model.

Once trained on the training data, the Holt-Winters ES model performs one-step ahead prediction on the test data. During prediction, the model utilizes the estimated level, trend, and seasonal components to forecast future values. The error of the prediction and anomaly score are computed for each data point as described in section 2.2.1.3.

4.3.2.3 Gaussian Mixture Model

For the Gaussian mixture model such parameters as a number of components and length of the bin were tuned. The number of components parameter determines the complexity of the Gaussian mixture model by specifying the number of Gaussian distributions used to model the data. This parameter was selected from this set of values $n\_components \in \{1, 2, 3, 4, 5\}$. The length of bins parameter denoted in Eq. 3.9, which determines the size of the bins used for discretizing the time series data was selected from these values $N \in 5, 10, 12, 24, 48$.

4.3.2.4 Isolation Forest

To optimize the performance of the iForest model, several key parameters are tuned, including the length of sliding window Eq. 3.11, maximum tree depth, and the number of trees in the forest.

By varying the sliding window length, different subsets of the time series are considered, allowing the model to capture temporal patterns at varying granularities. The sliding window length parameter was tuned across such values $M \in \{5, 10, 15, 20, 25, 50, 100\}$. The maximum tree depth parameter controls the depth of the isolation trees constructed within the iForest algorithm. Deeper trees can capture more intricate patterns in the data but may also lead to overfitting, while shallower trees may fail to capture complex anomalies. Various values for the maximum tree depth were considered during the parameter tuning process $max\_depth \in \{2, 5, 10, 25, 50, 100, 200\}$. The number of trees parameter defines the size of the ensemble forest in the iForest algorithm. Increasing the number of trees in the forest can enhance the robustness of anomaly detection by aggregating predictions from multiple trees. However, a larger number of trees may also lead to higher computational costs. The parameter tuning process involved experimenting with such values $n\_trees \in \{5, 10, 25, 50, 100, 250\}$ This parameter tuning process was conducted individually for each dataset to accommodate the specific characteristics and temporal dynamics of the data.

4.3.2.5 LSTM

The Long Short-Term Memory model was configured with the following parameters:

- **Number of Layers:** The LSTM architecture experimented with different numbers of layers, ranging from 3 to 9. The selected number of layers was 5.
- **Number of Neurons per Layer:** Each LSTM layer had a varying number of neurons, with configurations ranging from 32 to 128 neurons per layer.
- **Sliding Window Length:** The sliding window length, denoted by w, determined the number of previous time steps used as input to predict the next time step. Different window lengths of $w \in \{5, 10, 15, 20\}$ time steps were considered.
- **Activation Functions:** *ReLU*, *Sigmoid*, and $Tanh$, were tested to introduce non-linearity into the LSTM layers.
- **Batch Size:** Different batch sizes, such as 32, 64, 128, and 256, were experimented with.
- **Loss Function:** The loss function used during training was the Mean Absolute Error (MAE) to measure the difference between the predicted and actual values.
- **Optimizer:** The *AdamW* optimizer, an extension of the Adam optimizer with weight decay, was used to update the model parameters during training.

The LSTM model was trained on the training dataset using these configurations, and one-step-ahead predictions were generated for the test dataset. Anomaly scores were computed based on the LSTM model's predictions using the same method as for the ARIMA and ES models.

4.3.3 Ensemble models output computation

To devise ensemble models, the combination of outputs from both machine learning and statistical models was orchestrated to enhance anomaly detection. This approach aimed to capitalize on the distinct capabilities of each model type, thereby refining the overall detection performance. Specifically, the ensemble strategy involved combining the predictive errors from statistical models such as ARIMA and exponential smoothing with the anomaly scores generated by ML algorithms like Isolation Forest and Gaussian mixture models.

The ensemble architecture comprised six distinct models, each comprising a coupling of one statistical and one ML model. Four ensembles were fashioned by pairing ARIMA or ES with iForest or GMM. Within these ensembles, the amalgamation process involved computing the weighted average of the prediction error from the statistical model and the anomaly score from the ML model at each data point.

The ensemble incorporating LSTM and statistical models adopted a different methodology. For this configuration, an additional feature was introduced, derived from the statistical model's outputs. Specifically, this feature represented the aggregated output from the statistical model, computed as the average prediction error over a window of the previous n data points. The window size n corresponded to the training window size employed for training the LSTM model.

4.3.4 Evaluation metrics and performance validation

Anomaly detection often deals with datasets where anomalies are rare compared to normal data points, leading to class imbalance. Anomaly detection can be seen as a kind of binary classification problem on highly imbalanced data.

Definition 4.3.2. Confusion matrix for binary anomaly classification is depicted in Table 4.2:

		Predicted Class	
Actual Class		Negative (Normal)	Positive (Anomalous)
	Negative (Normal)	TN	FP
	Positive (Anomalous)	FN	TP

Table 4.2: Confusion matrix.

Values TP, FP, TN, and FN correspond to:

True Negative (TN): Actual event is normal and predicted as normal

False Positive (FP): Actual event is normal, but predicted as anomalous

False Negative (FN): Actual event is anomalous, but predicted as normal

True Positive (TP): Actual event is anomalous and predicted as anomalous

Definition 4.3.3. The precision is described with the next equation:

$$\frac{TP}{TP + FP}. \tag{4.4}$$

Definition 4.3.4. The recall is described with the following equation:

$$\frac{TP}{TP + FN}. \tag{4.5}$$

The F1 score is a way to measure how well a model balances precision and recall. Precision looks at the proportion of true positive predictions out of all positive predictions made by the model, while recall looks at the proportion of true positive predictions out of all actual positive instances. The F1 score combines these two metrics to give an overall assessment of a model's performance, especially useful when there is an imbalance between the number of normal and anomalous data points.

Definition 4.3.5. The F1 score is defined as the harmonic mean of precision and recall [34]. F1 score can be described with next equation:

$$\frac{TP}{TP + \frac{FN+FP}{2}}. \tag{4.6}$$

The precision-recall curve shows the trade-off between precision and recall at different decision thresholds used in classification. Precision is plotted on the y-axis, while recall is plotted on the x-axis. This curve helps visualize how precision and recall change as the decision threshold shifts. A higher area under the precision-recall curve indicates better overall performance of the model in identifying anomalies across different thresholds.

Definition 4.3.6. A precision-recall curve depicts the relationship between precision and recall across various classification thresholds.

Definition 4.3.7. PR-AUC is an area under the interpolated precision-recall curve, obtained by plotting (recall, precision) points for different values of the classification threshold.

In the context of anomaly detection, where anomalies are typically rare compared to normal instances, the PR-AUC provides a more informative assessment of model performance. It quantifies the ability of a model to balance precision (the fraction of relevant instances among the retrieved instances) and recall (the fraction of relevant instances that have been retrieved over the total amount of relevant instances) across various decision thresholds [35].

The F1 score and PR-AUC metrics are widely recognized and easily interpretable measures, making them ideal choices for assessing anomaly detection performance. Consequently, they have been selected for evaluating all conducted experiments. The F1 score is utilized as the primary criterion for ranking the detection algorithms.

Each dataset underwent time series cross-validation with a parameter of *n_split* = 3. This cross-validation technique was implemented using the *sklearn.model_selection.TimeSeriesSplit* module.

Time series cross-validation is particularly suited for sequential data like time series because it preserves the temporal order of the data during validation. The *TimeSeriesSplit* function divides the dataset into sequential folds, ensuring that data in each fold comes after the data in preceding folds. This prevents data leakage and ensures that the model is evaluated on unseen data. During each fold of cross-validation, the model is trained on a portion of the dataset and evaluated on the subsequent portion.

4.4 Experiment results

This section encompasses the outcomes derived from the conducted experiments, with a particular emphasis on the combination of statistical models with machine learning approaches, aimed at assessing the efficacy of ensemble models. Structurally, it is divided into two key segments. Initially, the spotlight is on showcasing the most effective models identified for each method across the various datasets. Subsequently, a comprehensive overview of the results across all methods and datasets is provided, allowing for a comparative analysis, including an assessment of the ensemble models' performance.

The primary assessment metric employed throughout the analysis remains the F1 score. Within each method-dataset pairing, the model configuration yielding the highest F1 score is highlighted, facilitating a comprehensive cross-method comparison. Additionally, as supplementary metrics, PR-AUC and precision-recall curve for each method and dataset are documented. It is important to note that the focus extends beyond individual models to encompass the effectiveness of ensemble models, which combine statistical and machine learning techniques. This holistic approach ensures a thorough evaluation of the model's effectiveness, shedding light on the potential advantages offered by ensemble methodologies in the realm of anomaly detection.

4.4.1 Results on Yahoo Finance dataset

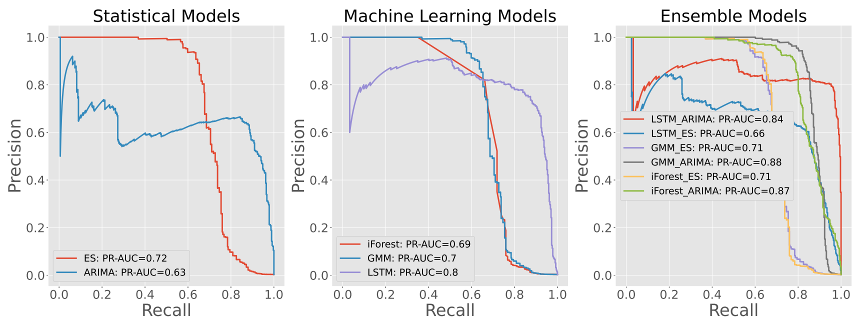

Figure 4.4: Comparison of Precision-Recall Curve Scores for Statistical, Machine Learning, and Hybrid Models on the Training Set of Yahoo Finance Dataset.

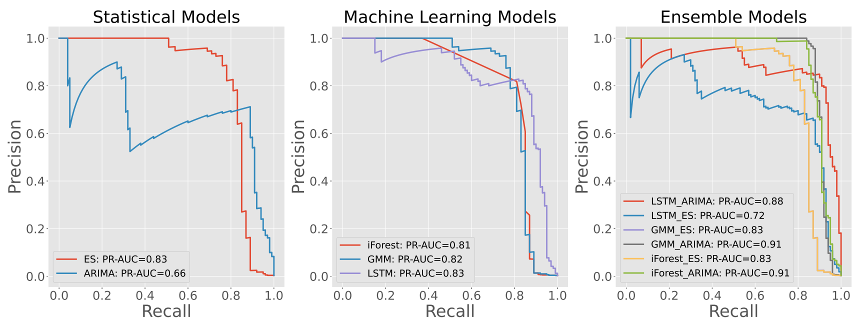

Figure 4.5: Comparison of Precision-Recall Curve Scores for Statistical, Machine Learning, and Hybrid Models on the Testing Set of Yahoo Finance Dataset.

Table 4.3: Comparison of F1 Scores for Statistical, Machine Learning, and Hybrid Models on the Training Set of Yahoo Finance Dataset.

		ES	ARIMA
		0.7577	0.7437
GMM	0.7456	0.7577	**0.8772**
IForest	0.7322	0.7577	0.8398
LSTM	0.8081	0.7078	0.8725

Table 4.4: Comparison of F1 Scores for Statistical, Machine Learning, and Hybrid Models on the Testing Set of Yahoo Finance Dataset.

		ES	ARIMA
		0.8306	0.7807
GMM	0.8172	0.8306	**0.9101**
IForest	0.5401	0.8306	0.8984
LSTM	0.8208	0.7532	0.8696

The comparison of precision-recall curve scores for statistical, machine learning, and hybrid models on the training set can be found in Figure 4.4, while the comparison on the testing set is presented in Figure 4.5.

Table 4.3 and Table 4.4 displays the F1 score performance of various models on the training and testing set. The first column corresponds to the F1 scores of machine learning (ML) models, while the first row represents the F1 scores of statistical models. The intersection of rows and columns denotes the F1 score performance of hybrid models

ES demonstrates commendable performance with an F1 score of 0.8306, indicating its efficacy in capturing anomalies. ARIMA, with a slightly lower F1 score of 0.7807, nonetheless exhibits respectable performance, albeit trailing behind ES. ES and ARIMA, as conventional statistical models, prevail in modeling linear trends and seasonal patterns. Their performance suggests that anomalies within the Yahoo Finance dataset may exhibit some degree of predictability and adherence to established patterns. GMM yields a competitive F1 score of 0.8172, showcasing its ability to capture complex data distributions. GMM's performance indicates proficiency in identifying anomalies characterized by distinct clusters or seasonal patterns, suggesting potential separability within the feature space. iForest exhibits a comparatively lower F1 score of 0.5401, signaling overfitting or challenges in isolating anomalies effectively. iForest's performance suggests anomalies within the dataset may not be as easily separable or isolated in feature space, posing difficulties in detection. LSTM achieves an F1 score of 0.8208, underscoring its capability to capture temporal dependencies and patterns. LSTM's effectiveness in modeling intricate temporal patterns aligns with the dataset's characteristics, which likely exhibit complex temporal dynamics.

The ensemble models exhibit notable enhancements in detection accuracy compared to individual models. This improvement underscores the efficacy of combining multiple algorithms to capitalize on their complementary strengths. Each ensemble variant, whether integrating the Gaussian Mixture Model, Isolation Forest, or Long Short-Term Memory, showcases distinct performance characteristics. Notably, the ensemble models that incorporate ARIMA alongside GMM or iForest demonstrate particularly robust performance, with F1 scores exceeding those of standalone models. This suggests that combining the predictive capabilities of ARIMA with the clustering or isolation strengths of GMM or iForest contributes significantly to anomaly detection accuracy. However, it is important to note that not all ensemble combinations yield substantial improvements. For instance, ensemble models incorporating Exponential Smoothing (ES) alongside GMM or iForest exhibit mixed results, with some variants failing to surpass the performance of their individual counterparts.

4.4.2 Results on NAB dataset

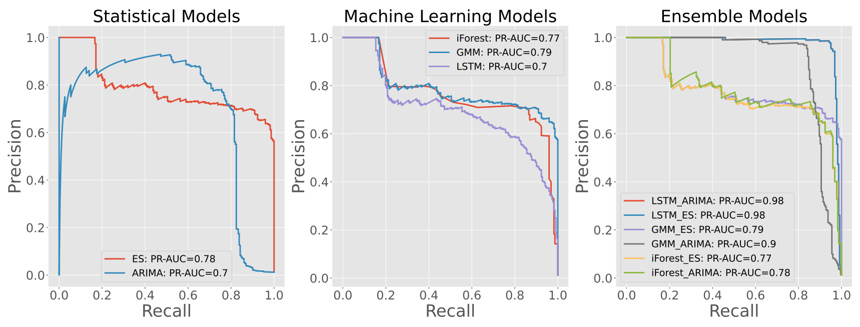

Figure 4.6: Comparison of Precision-Recall Curve Scores for Statistical, Machine Learning, and Hybrid Models on the Training Set of NAB Dataset.

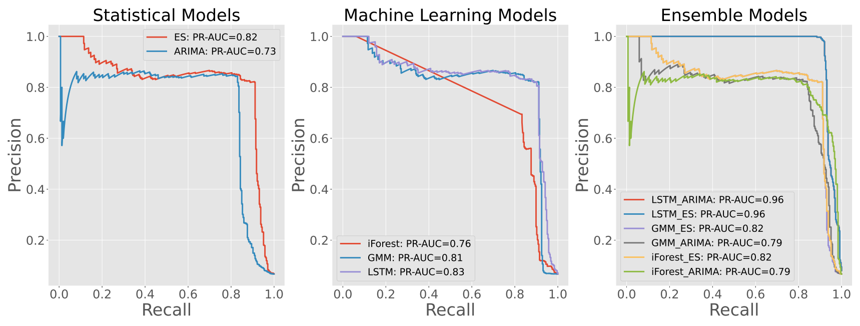

Figure 4.7: Comparison of Precision-Recall Curve Scores for Statistical, Machine Learning, and Hybrid Models on the Testing Set of NAB Dataset.

Table 4.5: Comparison of F1 Scores for Statistical, Machine Learning, and Hybrid Models on the Training Set of NAB Dataset.

		ES	ARIMA
		0.7875	0.7689
GMM	0.7937	0.7937	0.8986
IForest	0.6385	0.7891	0.8833
LSTM	0.6829	0.9555	**0.9635**

Table 4.6: Comparison of F1 Scores for Statistical, Machine Learning, and Hybrid Models on the Testing Set of NAB Dataset.

		ES	ARIMA
		0.8516	0.8545
GMM	0.8019	0.8395	0.8686
IForest	0.7009	0.8737	0.8623
LSTM	0.7991	0.9370	**0.9490**

The comparison of precision-recall curve scores for statistical, machine learning, and hybrid models on the training set can be found in Figure 4.6, while the comparison on the testing set is presented in Figure 4.7. Table 4.5 and Table 4.6 displays the F1 score performance.

Both ES and ARIMA exhibit competitive performance, indicating their suitability for detecting anomalies in this dataset. Despite exhibiting slightly lower performance compared to ES and ARIMA, GMM demonstrates proficiency in identifying anomalies. The performance of iForest suggests challenges in effectively isolating anomalies within cohesive clusters, potentially due to the collective nature of anomalies in the dataset. LSTM's effectiveness in modeling non-linear patterns aligns with the dataset's characteristics, which likely exhibit complex dynamics.

Notably, ensemble models that combine LSTM with ES or ARIMA showcase substantial improvements, with F1 scores nearing or exceeding 0.9. This highlights the synergistic effects of combining LSTM's ability to capture dependencies with the predictive power of ES or ARIMA in detecting anomalies within cohesive clusters. However, not all ensemble combinations yield significant improvements. Variants, incorporating Gaussian Mixture Model or Isolation Forest alongside ES or ARIMA, exhibit more modest improvements in detection accuracy.

4.4.3 Results on synthetic dataset

4.4.3.1 Point anomalies

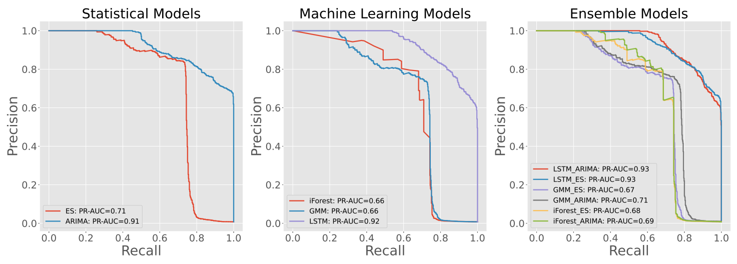

Figure 4.8: Comparison of Precision-Recall Curve for Statistical, Machine Learning, and Hybrid Models on the Training Set of Synthetic Dataset with Point Anomalies.

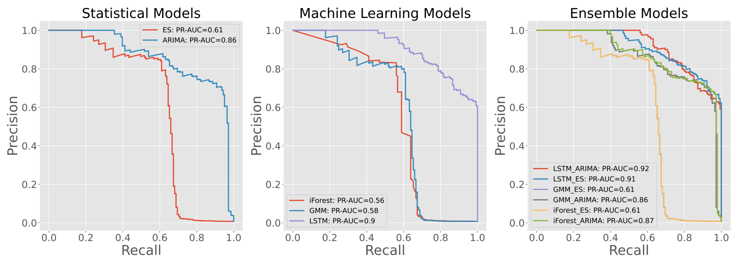

Figure 4.9: Comparison of Precision-Recall Curve for Statistical, Machine Learning, and Hybrid Models on the Testing Set of Synthetic Dataset with Point Anomalies.

Table 4.7: Comparison of F1 Scores for Statistical, Machine Learning, and Hybrid Models on the Training Set of Synthetic Dataset with Point Anomalies.

		ES	ARIMA
		0.7821	0.8180
GMM	0.7303	0.7814	0.8150
IForest	0.7318	0.7814	0.8124
LSTM	0.8220	**0.8335**	0.8313

Table 4.8: Comparison of F1 Scores for Statistical, Machine Learning, and Hybrid Models on the Testing Set of Synthetic Dataset with Point Anomalies.

		ES	ARIMA
		0.6908	0.7703
GMM	0.6855	0.6908	0.7716
IForest	0.6222	0.6908	0.7546
LSTM	0.7958	0.8057	**0.8195**

The comparison of precision-recall curve scores for statistical, machine learning, and hybrid models on the training set can be found in Figure 4.8, while the comparison on the testing set is presented in Figure 4.9. Table 4.7 and Table 4.8 displays the F1 score performance.

Point anomalies, characterized by individual data points deviating significantly from the rest of the dataset, may be effectively captured by ES and ARIMA. These models dominate in modeling trends and seasonality, enabling them to detect deviations from expected patterns. However, the performance improvement of ARIMA over ES suggests that ARIMA's ability to capture temporal dependencies may provide an advantage in identifying isolated events. GMM's performance lags behind ES and ARIMA, indicating potential challenges in capturing anomalies characterized by subtle deviations from the established pattern.

iForest's performance in detecting point anomalies is comparatively lower, suggesting limitations in isolating anomalies within the feature space. Point anomalies may not always exhibit clear isolation in the feature space, potentially hindering iForest's effectiveness. It was not expected that iForest's performance in detecting point anomalies would be comparatively lower. However, this outcome could be attributed to the lower deviation of point anomalies in this dataset. This lack of distinctiveness can potentially hinder iForest's effectiveness in isolating anomalies within the feature space. Similar to previous results LSTM demonstrates strong performance.

Ensemble models incorporating LSTM variants showcase superior performance, with F1 scores nearing or exceeding 0.8. This highlights the effectiveness of leveraging LSTM's ability to capture temporal dependencies and sequential patterns in detecting point anomalies within the GutenTAG dataset. Some variants, particularly those incorporating Gaussian Mixture Model or Isolation Forest alongside ES or ARIMA, exhibit more modest improvements in detection accuracy. Overall, while ensemble models offer promising avenues for enhancing anomaly detection accuracy, the performance gains may vary depending on the combination of constituent models.

4.4.3.2 Collective anomalies

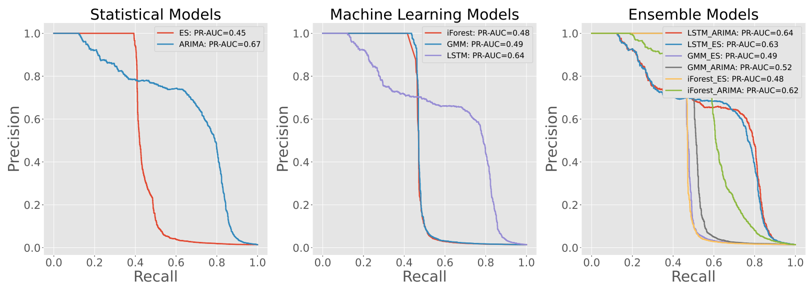

Figure 4.10: Comparison of Precision-Recall Curve for Statistical, Machine Learning, and Hybrid Models on the Training Set of Synthetic Dataset with Collective Anomalies.

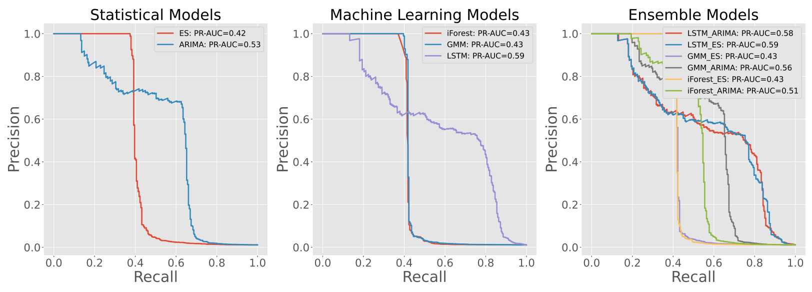

Figure 4.11: Comparison of Precision-Recall Curve for Statistical, Machine Learning, and Hybrid Models on the Testing Set of Synthetic Dataset with Collective Anomalies.

Table 4.9: Comparison of F1 Scores for Statistical, Machine Learning, and Hybrid Models on the Training Set of Synthetic Dataset with Collective Anomalies.

		ES	ARIMA
		0.5643	0.6865
GMM	0.6165	0.6132	0.6927
IForest	0.6001	0.6173	0.6868
LSTM	0.6726	**0.6990**	0.6976

Table 4.10: Comparison of F1 Scores for Statistical, Machine Learning, and Hybrid Models on the Testing Set of Synthetic Dataset with Collective Anomalies.

		ES	ARIMA
		0.5409	0.5401
GMM	0.5680	0.5636	**0.6402**
IForest	0.5528	0.5688	0.6287
LSTM	0.5977	0.6112	0.6061

The comparison of precision-recall curve scores for statistical, machine learning, and hybrid models on the training set can be found in Figure 4.10, while the comparison on the testing set is presented in Figure 4.11. Table 4.9 and Table 4.10 displays the F1 score performance.

Collective anomalies pose challenges for traditional statistical models like ES and ARIMA. These models may struggle to detect anomalies that manifest as cohesive clusters within the dataset, particularly when the deviations from the established pattern are subtle. GMM's performance in detecting collective anomalies is marginally better than ES and ARIMA. Its ability to model complex data distributions and seasonal patterns may aid in capturing collective anomalies. Isolation Forest's performance in detecting collective anomalies is comparable to GMM, suggesting that its inherent ability to isolate anomalies within the feature space may provide some advantages. However, the performance improvement is modest, indicating potential challenges. LSTM's ability to capture temporal dependencies and sequential patterns makes it better suited for detecting collective anomalies compared to traditional statistical models. Its performance improvement over ES and ARIMA highlights the importance of considering temporal dynamics in anomaly detection.

Ensemble models incorporating LSTM variants showcase relatively better performance, with F1 scores nearing or exceeding 0.6. This highlights the importance of leveraging LSTM's ability to capture temporal dependencies and non-linear trends. Notably, ensemble models that combine GMM or Isolation Forest with LSTM variants exhibit promising performance improvements. This suggests that integrating the clustering or isolation capabilities of GMM or iForest with LSTM's ability to capture temporal dependencies can enhance detection accuracy for collective anomalies. However, it is important to note that ensemble performance varies depending on the combination of constituent models. Some combinations may yield only marginal improvements in detection accuracy, underscoring the importance of careful model selection and configuration.

4.4.3.3 Contextual anomalies

The comparison of precision-recall curve scores for statistical, machine learning, and hybrid models on the training set can be found in Figure 4.12, while the comparison on the testing set is presented in Figure 4.13. Table 4.11 and Table 4.12 displays the F1 score performance.

ARIMA's proficiency in modeling temporal dependencies and patterns enables it to effectively capture contextual anomalies. Its ability to adapt to changes in data patterns over time may contribute to its superior performance in detecting deviations from expected patterns within specific contexts. GMM's performance in detecting contextual anomalies is comparable to ES, indicating potential challenges in capturing nuanced contextual dependencies.

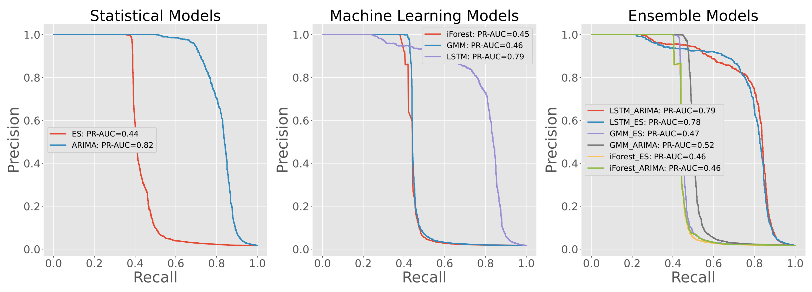

Figure 4.12: Comparison of Precision-Recall Curve for Statistical, Machine Learning, and Hybrid Models on the Training Set of Synthetic Dataset with Contextual Anomalies.

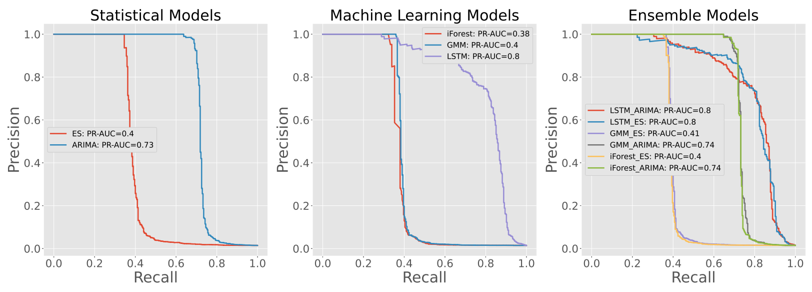

Figure 4.13: Comparison of Precision-Recall Curve for Statistical, Machine Learning, and Hybrid Models on the Testing Set of Synthetic Dataset with Contextual Anomalies.

Table 4.11: Comparison of F1 Scores for Statistical, Machine Learning, and Hybrid Models on the Training Set of Synthetic Dataset with Contextual Anomalies.

		ES	ARIMA
		0.5522	0.8045
GMM	0.5938	0.6004	**0.8089**
IForest	0.5647	0.5953	0.8055
LSTM	0.7804	0.7830	0.7854

Table 4.12: Comparison of F1 Scores for Statistical, Machine Learning, and Hybrid Models on the Testing Set of Synthetic Dataset with Contextual Anomalies.

		ES	ARIMA
		0.5062	0.7977
GMM	0.5274	0.5333	0.8047
IForest	0.4843	0.5230	**0.8061**
LSTM	0.7574	0.7807	0.7539

The limitations observed in the GMM's ability to detect contextual anomalies may stem from its fundamental approach. The segmentation of data into seasonal bins by the GMM model implies a partitioning that may overlook subtle contextual anomalies existing across these divisions. Contextual anomalies may not always exhibit clear isolation in the feature space, posing challenges for iForest in detecting such anomalies effectively. LSTM demonstrates the second highest F1 score after ARIMA.

Ensemble models that combine ARIMA with other models, such as GMM and ARIMA or iForest and ARIMA, showcase relatively better performance, with F1 scores exceeding 0.8. Ensemble models incorporating LSTM variants also demonstrate competitive performance. This suggests that LSTM's ability to contribute to improved detection accuracy for contextual anomalies.

4.4.4 Discussion

Based on the experimental results, several notable observations and conclusions can be drawn regarding the performance of anomaly detection models in time series data.

Firstly, statistical models such as Exponential Smoothing and Autoregressive Integrated Moving Average consistently demonstrated robust performance across various datasets and anomaly types. These models, leveraging statistical principles, showcased effectiveness in capturing temporal patterns and identifying anomalies, particularly point and collective anomalies.

Machine learning models, including the Gaussian Mixture Model, Isolation Forest, and Long Short-Term Memory networks, exhibited varying degrees of success in anomaly detection. GMM showed competitive performance, particularly when combined with ES or ARIMA in ensemble models. iForest demonstrated effectiveness in detecting anomalies, albeit with some limitations in handling contextual anomalies. LSTM networks, while powerful in capturing complex temporal dependencies, exhibited mixed performance across datasets and anomaly types.

Ensemble models combining multiple techniques showcased promising results, often outperforming individual models. The fusion of statistical and machine learning frameworks in ensemble models, such as GMM combined with ES or ARIMA, offered enhanced anomaly detection capabilities, leveraging the complementary strengths of each approach.

In conclusion, while no single model emerged as a clear winner in anomaly detection, the experimental findings highlight the strengths and limitations of different techniques and emphasize the efficacy of hybrid and ensemble approaches in enhancing anomaly detection performance in time series data.

Chapter 5

Conclusion

The advent of time series data has brought forth a myriad of challenges in anomaly detection, necessitating the development of robust and adaptive techniques. Anomaly types, ranging from point anomalies to contextual and collective anomalies, present unique obstacles, requiring specialized methodologies for effective detection.

Traditional statistical models, such as Exponential Smoothing (ES) and ARIMA, offer foundational frameworks for anomaly detection but may exhibit limitations in capturing complex patterns inherent in time series data. Conversely, machine learning algorithms, exemplified by Long Short-Term Memory, demonstrate prowess in capturing temporal dependencies and sequential patterns, offering promising avenues for anomaly detection.

However, the disparate nature of anomaly types and the limitations of individual models underscore the need for a novel approach. In response to this challenge, a hybrid methodology is proposed in this thesis, combining elements of both machine learning and statistical models. This hybrid approach aims to leverage the complementary strengths of each paradigm, offering a more robust and adaptive framework for anomaly detection in time series data.

By integrating machine learning's capacity for pattern the proposed hybrid approach seeks to overcome the limitations of traditional methodologies. Through empirical evaluation and comparative analysis, the efficacy of this hybrid approach is demonstrated, highlighting its potential in enhancing anomaly detection performance across different anomaly types.

In summary, this thesis contributes to the advancement of anomaly detection techniques in time series data by addressing the challenges inherent in anomaly detection and proposing a novel hybrid approach. By bridging the gap between machine learning and statistical models, this research lays the groundwork for the development of more effective and adaptive anomaly detection systems in various domains.

Bibliography

[1] Van Houdt, G.; Mosquera, C.; et al. A review on the long short-term memory model. *Artificial Intelligence Review*, volume 53, no. 8, Dec. 2020: pp. 5929–5955, ISSN 0269-2821, doi:10.1007/s10462-020-09838-1.

[2] Teng, M. Anomaly detection on time series. In *2010 IEEE International Conference on Progress in Informatics and Computing*, Jan 2010, pp. 603–608, doi:10.1109/PIC.2010.5687485.

[3] Schmidl, S.; Wenig, P.; et al. Anomaly detection in time series: a comprehensive evaluation. *Proc. VLDB Endow.*, volume 15, no. 9, may 2022: p. 1779–1797, ISSN 2150-8097, doi:10.14778/3538598.3538602. Available from: `https://doi.org/10.14778/3538598.3538602`

[4] Nassif, A.; Abu Talib, M.; et al. Machine Learning for Anomaly Detection: A Systematic Review. *IEEE Access*, volume PP, 05 2021: pp. 1–1, doi:10.1109/ACCESS.2021.3083060.

[5] Liu, F. T.; Ting, K. M.; et al. Isolation-Based Anomaly Detection. *ACM Trans. Knowl. Discov. Data*, volume 6, no. 1, Mar 2012, ISSN 1556-4681, doi:10.1145/2133360.2133363. Available from: `https://doi.org/10.1145/2133360.2133363`

[6] Gunning, D.; Stefik, M.; et al. XAI—Explainable artificial intelligence. *Science Robotics*, volume 4, Dec 2019: p. eaay7120, doi:10.1126/scirobotics.aay7120.

[7] Braei, M.; Wagner, S. Anomaly Detection in Univariate Time-series: A Survey on the State-of-the-Art. *CoRR*, volume abs/2004.00433, 2020, 2004.00433. Available from: `https://arxiv.org/abs/2004.00433`

[8] Shumway, R.; Stoffer, D. *Time Series Analysis and Its Applications: With R Examples*. Springer Texts in Statistics, Springer New York, 2010, ISBN

9781441921253. Available from: https://books.google.cz/books?id=ahdvcgAACAAJ

[9] Hamilton, J. D. *Time Series Analysis*. Princeton University Press, 1994, ISBN 9780691042893. Available from: https://www.worldcat.org/title/time-series-analysis/oclc/1194970663&referer=brief_results

[10] Bhattacharya, A. Effective Approaches for Time Series Anomaly Detection. *Towards Data Science [online]*, 2020, [Cited 2021-12-12]. Available from: https://towardsdatascience.com/effective-approaches-for-time-series-anomaly-detection-9485b40077f1

[11] Chandola, V.; Banerjee, A.; et al. Anomaly Detection: A Survey. *ACM Comput. Surv.*, volume 41, 07 2009, doi:10.1145/1541880.1541882.

[12] Shaukat Dar, K.; Mahboob Alam, T.; et al. *A Review of Time-Series Anomaly Detection Techniques: A Step to Future Perspectives*. 04 2021, ISBN 978-3-030-73099-4, doi:10.1007/978-3-030-73100-7_60.

[13] Zhang, P. Time Series Forecasting Using a Hybrid ARIMA and Neural Network Model. Neurocomputing 50, 159-175. *Neurocomputing*, volume 50, 01 2003: pp. 159–175, doi:10.1016/S0925-2312(01)00702-0.

[14] Ullrich, T. On the Autoregressive Time Series Model Using Real and Complex Analysis. *Forecasting*, volume 3, no. 4, 2021: pp. 716–728, ISSN 2571-9394, doi:10.3390/forecast3040044. Available from: https://www.mdpi.com/2571-9394/3/4/44

[15] Shumway, R. H.; Stoffer, D. S. *ARIMA Models*. Cham: Springer International Publishing, 2017, ISBN 978-3-319-52452-8, pp. 75–163, doi:10.1007/978-3-319-52452-8_3. Available from: https://doi.org/10.1007/978-3-319-52452-8_3

[16] Box, G.; Jenkins, G. M. *Time Series Analysis: Forecasting and Control*. Holden-Day series in time series analysis and digital processing, Holden-Day, 1976, ISBN 9780816211043. Available from: https://books.google.cz/books?id=1WVHAAAAMAAJ

[17] Montgomery, D.; Jennings, C.; et al. *Introduction to Time Series Analysis and Forecasting*. Wiley Series in Probability and Statistics, Wiley, 2008, ISBN 9780471653974. Available from: https://books.google.cz/books?id=IY6OQgAACAAJ

[18] Fildes, R.; Harvey, A.; et al. Forecasting, Structural Time Series Models and the Kalman Filter. *The Journal of the Operational Research Society*, volume 42, 11 1991: p. 1031, doi:10.2307/2583225.

[19] Holt, C. *Forecasting seasonals and trends by exponentially weighted moving averages*, volume 20. International Institute of Forecasters, 2004, 5-10 pp., doi:https://doi.org/10.1016/j.ijforecast.2003.09.015. Available from: `https://www.sciencedirect.com/science/article/pii/S0169207003001134`

[20] Liu, J.; Zhu, H.; et al. Anomaly detection for time series using temporal convolutional networks and Gaussian mixture model. *Journal of Physics*, volume 1187, no. 4, Apr. 2019: p. 042111, doi:10.1088/1742-6596/1187/4/042111. Available from: `https://doi.org/10.1088/1742-6596/1187/4/042111`

[21] Douglas, R. *Gaussian Mixture Models*. Boston, MA: Springer US, 2009, ISBN 978-0-387-73003-5, pp. 659–663, doi:10.1007/978-0-387-73003-5_196. Available from: `https://doi.org/10.1007/978-0-387-73003-5_196`

[22] Dempster, A. P.; Laird, N. M.; et al. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, volume 39, no. 1, 1977: pp. 1–38, ISSN 00359246. Available from: `http://www.jstor.org/stable/2984875`

[23] Reddy, A.; Ordway-West, M.; et al. Using Gaussian Mixture Models to Detect Outliers in Seasonal Univariate Network Traffic. In *2017 IEEE Security and Privacy Workshops (SPW)*, 2017, pp. 229–234, doi:10.1109/SPW.2017.9.

[24] Sun, L.; Versteeg, S.; et al. Detecting Anomalous User Behavior Using an Extended Isolation Forest Algorithm: An Enterprise Case Study. 2016, `1609.06676`.

[25] Zhong, S.; Fu, S.; et al. *A novel unsupervised anomaly detection for gas turbine using Isolation Forest*. 2019, 1-6 pp., doi:10.1109/ICPHM.2019.8819409.

[26] Alpaydin, E. *Introduction to Machine Learning*. Adaptive Computation and Machine Learning, Cambridge, MA: MIT Press, third edition, 2014, ISBN 978-0-262-02818-9.

[27] Bishop, C. M. *Neural Networks for Pattern Recognition*. USA: Oxford University Press, Inc., 1995, ISBN 0198538642.

[28] Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Computation*, volume 9, no. 8, 11 1997: pp. 1735–1780, ISSN 0899-7667, doi:10.1162/neco.1997.9.8.1735, `https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf`.

[29] Werbos, P. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, volume 78, no. 10, 1990: pp. 1550–1560, doi:10.1109/5.58337.

[30] Laptev, N.; Amizadeh, S. Yahoo anomaly detection dataset s5. 2015.

[31] Lavin, A.; Ahmad, S. Evaluating real-time anomaly detection algorithms - the Numenta Anomaly Benchmark. Nov 2015. Available from: `https://arxiv.org/abs/1510.03336`

[32] Wenig, P.; Schmidl, S.; et al. TimeEval: A Benchmarking Toolkit for Time Series Anomaly Detection Algorithms. *Proceedings of the VLDB Endowment (PVLDB)*, volume 15, no. 12, 2022: pp. 3678 – 3681, doi: 10.14778/3554821.3554873.

[33] Akaike, H. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, volume 19, no. 6, 1974: pp. 716–723, doi: 10.1109/TAC.1974.1100705.

[34] Korstanje, J. The F1 score. [online], Aug 2021, [cit. 2022-02-02]. Available from: `https://towardsdatascience.com/the-f1-score-bec2bbc38aa6`

[35] Davis, J.; Goadrich, M. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, New York, NY, USA: Association for Computing Machinery, 2006, ISBN 1595933832, p. 233–240, doi: 10.1145/1143844.1143874. Available from: `https://doi.org/10.1145/1143844.1143874`

Appendix A

Acronyms

AR Autoregressive

MA Moving average

ADF test Augmented Dickey–Fuller test

KPSS test Kwiatkowski–Phillips–Schmidt–Shin test

ARIMA Autoregressive integrated moving average

ES Exponential smoothing

AI Artificial intelligence

ML Machine learning

iForest Isolation Forest

GMM Gaussian mixture model

FFNN Feed-forward neural network

RNN Recurrent neural network

LSTM Long short-term memory

NAB Numenta Anomaly Benchmark

PR Precision-Recall

AUC Area under the curve

Appendix **B**

Contents of enclosed CD

```
readme.txt ......................... the file with CD contents description
implementation ............ the directory of code of the implementation
thesis ........................................ the thesis text directory
    thesis.pdf ........................... the thesis text in PDF format
    src .................. the directory of LaTeX source codes of the thesis
```