BSc Thesis Task Description

Bálint Bicski
candidate for BSc degree in Computer Engineering

Anomaly Detection in Industry 4.0-based Manufacturing

Industry 4.0 refers to a new phase in the Industrial Revolution that focuses heavily on interconnectivity, automation, machine learning (ML), and real-time data. Industry 4.0, also referred to as Industrial IoT or smart manufacturing, couples physical production and operations with innovative technologies and approaches to create a more holistic and better-connected ecosystem for companies that focus on manufacturing and supply chain management. However, with the development of Industry 4.0, the need for real-time ecosystem monitoring, including anomaly detection, has also emerged. Real-time identification of abnormal behavior in industrial systems is crucial as it can help detect signs of critical equipment failure or deliberate cyber-attacks. ML has been proven effective in making an operational sense of the enormous volume of time-series data composed of multiple variables. However, ML is not always necessary to achieve appropriate performance. The thesis assignment builds on this supposition. Its main goal is to examine how and in what conditions can simple probabilistic approaches outperform state-of-the-art ML anomaly detection techniques. To this end, various techniques should be considered to extend the capability of selected simple probabilistic methods to work in multivariate scenarios and examine their efficiency when detecting non-cross-variable anomalies in real time compared to ML approaches.

Tasks to be performed by the student include:

- Study Industry 4.0 and its evolution in the past decade.
- Analyze anomaly detection techniques both ML- and simple probabilistic-based.
- Perform comparison on univariate datasets using a set of selected techniques.
- Extend the statistical approaches to work in multivariate scenarios.
- Evaluate the performance of selected statistical approaches on multivariate datasets.

Supervisors at the department: Dr. Adrián Pekár, assistant professor
Dr. Károly Farkas, associate professor

Budapest, 8 October 2021

/ Dr. Sándor Imre /
professor
head of department

Supervisors' proposals:
Supervisor at the department: ☐Acceptable, ☐Not acceptable, date: signature:
External supervisor: signature:

Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Networked Systems and Services

H-1117 Budapest, Magyar tudósok körútja 2.
Building I, B 121 Phone: +36 1 463 3261
Fax: +36 1 463 3263, www.hit.bme.hu

Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Networked Systems and Services

Anomaly Detection in Industry 4.0 using Probabilistic Approaches

Bachelor's Thesis

Author
Bálint Bicski

Advisors
Dr. Adrián Pekár
Dr. Károly Farkas

December, 2021

Contents

HALLGATÓI NYILATKOZAT

Alulírott *Bicski Bálint*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2021. december 10.

Bicski Bálint
hallgató

Kivonat

Ipari rendszerekben az abnormális folyamatok valós idejű azonosítása meghatározó folyamat, mivel így azonnal detektálható egy kritikus komponens meghibásodása, vagy akár egy szándékosan indított kibertámadás a rendszer ellen. A gépi tanulás hatékonysága beigazolódott hatalmas mennyiségű adatokból történő információkinyerésre, mégis sok esetben nem szükséges használata a megfelelő teljesítmény eléréséhez. Ez a dolgozat erre a feltevésre épít, és azt vizsgálja meg, hogyan, milyen körülmények között tudják a valószínűségi modelleket használó megközelítések felülmúlni a korszerű gépi tanulást alkalmazó módszerek anomáliadetektáló teljesítményét. Dolgozatomban körüljárom az anomáliadetekció elméletét, megismertetem az Ipar 4.0 fogalmát, az ipari adatanalízis potenciális előnyeit és kihívásait. Továbbá kiterjesztem néhány kiválasztott, egyszerű valószínűségi módszer működőképességét többdimenziós környezetekre, valamint megvizsgálom hatásfokukat a nem dimenzióközti anomáliák valós idejű felismerését tekintve, gépi tanulásra alapozó megközelítésekkel szemben. A kiértékelés eredményei alapján a feltevést sikeresen be tudtam igazolni, az egyszerűbb valószínűségi megközelítések fel tudják venni a versenyt a mélytanuló algoritmusokkal többdimenziós ipari idősorok feldolgozása során is. Anomália metrikák közlése mellett megvizsgálom a különböző dimenzió-aggregációs, valamint anomália tűréshatárt meghatározó megoldások hatását is a választott algoritmusokra és adatsorokra.

Abstract

Real-time identification of abnormal behavior in industrial systems is crucial since it can help detect signs of critical equipment failure or deliberate cyber-attacks. ML has been proven effective in making an operational sense of the enormous volume of time-series data composed of multiple variables. However, ML is not always necessary to achieve appropriate performance. This work builds on this supposition and examines how and in what conditions can simple probabilistic approaches outperform state-of-the-art ML anomaly detection techniques. In my thesis I explore the theory of anomaly detection, present the concept of Industry 4.0 and the potential benefits and challenges of industrial data analysis. Furthermore, I extend the capability of selected simple probabilistic methods to work in multivariate scenarios and examine their efficiency when detecting non-cross-variable anomalies in real-time compared to ML approaches. Based on the evaluation results I managed to prove the supposition, simpler probabilistic approaches can compete with Deep Learning algorithms even on multivariate industrial time series. In addition to reporting the anomaly detection metrics I examine the effect of various channel-aggregation and thresholding functions on the selected algorithms and datasets.

Chapter 1

Introduction

In recent decades the need to monitor systems of any domain in an automated fashion has materialized. Aiming to fulfill this need system state monitoring has developed rapidly. This process consists of collecting status and behavioral data of multiple levels of a system — from physical equipment to software components.

The same process has taken place in industrial systems. In the 21st century a new revolution in manufacturing has emerged, called the Fourth Industrial Revolution or Industry 4.0. This innovative approach builds on the achievements of the automation of the manufacturing process that became widespread during the last century and advances them by incorporating state-of-the-art data collection and processing. When monitoring an industrial manufacturing process, measurements are usually taken at constant intervals of multiple variables, therefore, the monitored complex data arrives in a streaming fashion. These kind of data are called time-series data.

One important goal of system state monitoring is to identify abnormal behavior in the processes of the system, or try to prevent these beforehand. The former is called anomaly detection and the latter anomaly prevention. In this thesis, I focus on anomaly detection. These two processes are tremendously important, as abnormal behavior may be a sign of critical equipment failure or a deliberate cyber-attack against the system. Even if the cause is not so grave, the consequence may be system unavailability or financial loss.

As the monitoring of industrial processes generates an enormous amount of data, and multiple variables are monitored at the same time, big data techniques are utilized for data processing. A popular area of approaches is machine learning, many algorithms exist specifically tailored for anomaly detection in this field. These approaches can extract intricate details in the data and make their detection based on this. However, as the time series data arrives in a streaming fashion, it is desirable to process it in real-time, in order to have an up-to-date view of the industrial system at all times. This requires a quick algorithm which can analyze the data in an online fashion, i.e. as the data pieces arrive as opposed to when the whole dataset becomes available.

However, machine learning algorithms often do not comply with this requirement as they need a substantial amount of data to train the model, making them offline or quasi-real-time only. Therefore, simpler anomaly detectors may be advisable to use provided that they offer good detection performance. Gers et al. [12] and Polge et al. [34] claim that many time series problems do not require complex machine learning models at all because all relevant information about the next event is conveyed by a few recent events contained within a small time window, necessitating only simpler, more lightweight approaches. In

my work, I set out to test this hypothesis and compare a state-of-the-art machine learning anomaly detection approach with simpler probabilistic approaches.

In any system it is assumed that the normal behavior is dominant, and the unexpected unwanted abnormal behavior, i.e. anomalies are rare. Siffer et al. base their algorithm, SPOT [38] on this assumption using Extreme Value Theory to automatically set a threshold for extreme values (anomalies) and detect them. However, a shortcoming of the approach is that it only detects outliers which jump compared to normal data. Any unusual pattern within the normal data distribution is ignored. Li et al. realize this and propose a two-step smoothing process to clean the data of noise and a potential periodic pattern, and leave only the abnormal behaviors. On this pre-processed data SPOT can be used effectively as any kind of abnormal behavior is transformed into an outlier using the pre-processing steps. Other probabilistic approaches, like a simple auto-regressive model (AR) also showed promising results in [7], however, the whole dataset needs to be pre-processed before the algorithm can run, therefore, this approach requires modifications to be able to run in a streaming fashion.

A time series is mulivariate if multiple variables are monitored synchronously. Performing anomaly detection in this case is more difficult, as the different monitored values need to be processed at the same time. I focus on only those kind of anomalies which happen in one monitored subsytem, i.e. one variable individually. While this assumption may not cover all occurring anomalies, Garg et al. [11] have shown that it is true in the majority of cases and performed their study under the same assumption.

In this thesis I perform anomaly detection performance comparison of the aforementioned algorithms both in univariate (one variable) and multivariate (multiple variables) time series. To my understanding there have not been any studies that take into consideration the streaming data processing requirement and test statistical approaches on multivariate datasets at the same time.

The main contributions of this thesis are the following:

- Focusing on the streaming data processing requirement I introduce changes to some approaches and correct observed coding errors.
- I compare anomaly detection performance on univariate datasets using the aforementioned lightweight approaches, extending the work of Braei and Wagner [7].
- I propose a simple transformation of the output of probabilistic approaches to work in multivariate scenarios and study its effect with different aggregation methods.
- I test the performance of statistical approaches on multivariate datasets, comparing them to deep learning techniques, extending the work of Garg et al. [11]. I investigate the results with multiple aggregation and thresholding methods.
- I study the detected anomaly groups by each algorithm and propose potential ensemble pairs for future work.

The rest of this thesis is organized as follows.

- In Chapter 2 I present related work in the field of univariate and multivariate anomaly detection, contemporary studies and approaches for distinct types of anomaly detection, and introduce works and sources related to the the theory and study of Industry 4.0.

- Chapter 3 details the background of my work. Section 3.1 introduces the theory of time series, anomalies, anomaly types and concepts related to them. It also categorizes anomaly detection approaches along various characteristics. Section 3.2 presents the theory of Industry 4.0, the main alternations in a manufacturing business it proposes, the potential improvements it can bring about along with the challenges it faces in real world implementations.
- Chapter 4 details all the selected algorithms, SPOT in Section 4.1.1, FluxEV in Section 4.1.2, the Autoregressive model in Section 4.1.3 and the Autoencoder in Section 4.1.4 and describes the modifications I realized on them. Section 4.2 presents the multivariate framework I use for evaluation, along with the potential aggregation and thersholding approaches. Section 4.2.3 proposes a way to integrate SPOT and FluxEV into the framework, and make them compatible with multivariate anomaly detection.
- In Chapter 5 I set up and perform the evaluation of the algorithms. Section 5.1 presents the anomaly metrics I use in the experiments, Section 5.2 details the univariate-, Section 5.3 the multivariate evaluation.
- Chapter 6 concludes my work by summarizing the results and draws up future work.

Chapter 2

Related Work

Diez-Olivan et al. [9] study system integration concepts and state-of-the-art machine learning data processing approaches in their Industry 4.0 study. They divide the approaches into three main categories based on functionality and temporality of the method. One group, descriptive prognosis is for analyzing system health, marking unusual behavior and investigating the cause. Predictive prognosis approaches are tailored for making assumptions regarding future data points and potential failures using past data. The third group is prescriptive prognosis. These approaches recommend steps to optimize the functioning of the industrial systems. The authors outline several trends and challenges each of these categories face and describe the limitations of the real world human and hardware environment they need to operate in.

4.0 Solutions [40] is an industry leading for-profit educational organization for Industry 4.0 concepts. In their video material they explain the most important requirements to achieve a so called digital transformation in an industrial system and outline a Common Enterprise Namespace as a system architecture at the center of the concept. Professor Hardt, a professor at MIT describes the history and main points of industrial manufacturing and sheds light on the main reasons we are starting to see the manufacturing changes nowadays [35]. He stresses the importance of real time data processing.

Aggarwal [1] does an in-detail introduction into outlier analysis. The author defines the main concepts of the topic, investigates probabilistic, linear and proximity based solutions. Their explanation includes a description of outlier detection in time series including a multidimensional one in a streaming fashion. The author also analyzes the possible uses of anomaly detection ensembles – where multiple detectors work together.

Braei and Wagner [7] survey the state of anomaly detection in univariate time series. They describe the concept of time series, anomalies, and group approaches into statistical, classic machine learning and deep learning types. The authors perform extensive anomaly detection comparison on real world network traffic, taxi cab traffic and synthetic data. While they conclude that statistical approaches can outperform traditional machine learning and deep learning approaches, their comparison requires a step which makes the detection unable to work in real time and the test sets do not originate from an industrial domain.

Garg et al. [11] focus on several datasets of industrial data which span through multiple channels, contain multivariate time series. They train deep learning anomaly detectors on each channel individually and use novel aggregation functions to merge the results into one dimensional detections. While this may omit the detection of cross-channel anomalies, they argue that their method achieves satisfactory results. Another goal of their study

besides anomaly detection is the diagnosis of the location of the anomaly, essentially finding the culprit channel that caused an anomaly.

Siffer et al. [38] and Li et al. [28] both propose probabilistic approaches (SPOT and FluxEV respectively) that leverage Extreme Value Theory [4] for anomaly detection. This builds on the observation that anomalies are relatively rare compared to normal data. According to Li et al., FluxEV is capable of detecting a larger variety of anomalies compared to SPOT. Both of these approaches adapt to data drift, adaptively learn and work in streaming time series scenarios for point anomaly detection, however only support univariate data.

Wang et al. propose another solution called Self-Learning Anomaly Detection for Time Series or SLADE-TS for short. The approach is clustering-based and is tailored for periodic time series where in each period the same behavior is expected. Using adaptive techniques the algorithm keeps track of multiple normal and anomaly classes and has an adaptive window length to identify reoccurring subsequences inside of periods. With these it can account for data drift and pinpoint the exact location of an anomalous subsequence, not just the abnormal period. SLADE-TS works in an online fashion on streaming data fine-tuning the algorithm with new data. The authors also propose a variant for multivariate data, SLADE-MTS [47] which loses the real-time capability and – building on the univariate algorithm – leverages DBScan [10] for clustering. According to the authors, both approaches show promising anomaly detection performance, with the multivariate one being able to detect correlation anomalies as well. Unfortunately, they do not publish the source code.

In a non-periodic time series, NormA [6] by Boniol et al. can be used for subsequence anomaly detection. Their algorithm takes a sum of weighted distances to calculate an anomaly score. The components of the sum are based on the distance from all the normal centroids, which means that multiple normal classes may be supported. Based on common distance behavior the anomalies can also be grouped into distinct clusters. The calibration phase of the algorithm may contain anomalous values, owing to the fact that these will be relatively rare compared to normal sequences. The only downside of the approach is the necessity of a pre-defined anomaly length. However, according to the authors this should be readily available in every field of application by an expert.

In terms of deep learning anomaly detection approaches, Vajda et al. [44] propose an LSTM-based solution for anomaly detection in streaming data. The algorithm is adaptive and is claimed to be able to detect not only anomalous data points but pattern changes as well, with far superior performance to the LSTM approach it builds upon. Only univariate operation is supported by the approach, however, it needs no prior set parameters and offers a quick execution performance. Garg et al. [11] identify a univariate autoencoder to be the best option for multivariate time series. Their approach deploys an autoencoder for each channel, and uses a Gaussian cumulative distribution function to add the channel-wise anomaly scores together. The authors achieved good detection results on real industrial test data with the technique compared to other deep learning alogrithms like LSTM, GAN, MSCRED, OmniAnomaly and a fully connected autoencoder that has all channel data as its input.

Lavin and Ahmad [27, 2] propose Numenta Anomaly Benchmark (NAB), an open-source anomaly detection framework with univariate datasets of various domains. They provide detection results with multiple algorithms on [32]. Additionally to the framework, they also put forward a novel scoring method, called the NAB-score. However this has received criticism in [39] and has not become widespread in contemporary works.

Chapter 3

Background

3.1 Anomaly Detection in Time Series

3.1.1 Time Series

Definition 1 (Time Series). A time series T is a set of observations ($T = o^{(1)}, o^{(2)}, ...o^{(n)}$) of the current state of a process registered consecutively in time by taking measurements at equidistant time intervals. $n \in \mathbb{N}$ is the length of the series, which may be unknown. ▪

Since the observations are of the same process, neighboring records are correlated. Time series type data appear anywhere where measurements are taken successively given some type of time-related dimension. Therefore, time series take shape as data streams of any domain: from stock market prices to weather related data and network behavior monitoring. There may be no pre-defined end-point to the series, in this case values are assumed to arrive in a continuous fashion.

Observations can be either scalar values or a vector of different measurements taken at the same time. These time series are separately defined by the following definitions.

Definition 2 (Univariate Time Series). A univariate time series $T_{univariate}$ is a time series $T = o^{(1)}, o^{(2)}, ...o^{(n)}$, where $o^{(i)} \in \mathbb{R}$. ▪

Definition 3 (Multivariate/Multidimensional Time Series). A multivariate time series $T_{multivariate}$ is a time series $T = o^{(1)}, o^{(2)}, ...o^{(n)}$, where ($\boldsymbol{o^{(i)}} = [o_1^{(i)}, o_2^{(i)}, ...o_m^{(i)}]$, where $i > 0$: observation index, $m > 0$: dimensionality count.

In another representation the time series is an observation matrix $\boldsymbol{O} \in \mathbb{R}^{n \cdot m}$. The columns of the observation matrix are called dimensions or channels. ▪

While multivariate time series can be regarded as m univariate time synchronous time series, it is defined separately, because additional correlation may be present among the dimensions, the behavior of one channel may affect another.

Time series may have the following characteristics or patterns which play an important role in the functionality of data processing techniques.

Definition 4 (Periodicity/Seasonality). If the observations repeat with some period $p \in \mathbb{Z}^+$, i.e. the same values are measured at timestep t and at timestep $t + p$ for

every timestep in the time-series ($o^{(t)} = o^{(t+p)}$), the time series exhibits periodicity or seasonality. ▪

Most time series have seasonality as the same behavior repeats. For instance, temperature measurements typically have yearly periodicity, and industrial data might have repetition on an hourly basis as the next product reaches the same point in the production line.

Definition 5 (Cycle). A cycle is a period that is not fixed in time, i.e. the series does not have a strict period. Cycles typically play out in wider time intervals, longer than years, therefore their effect is weaker than the effect of seasonality. ▪

Definition 6 (Level). The mean of the time series is commonly referred to as the level of the time series. ▪

Definition 7 (Trend). A time series observes a trend if the level is not constant but increases or decreases over time. ▪

Definition 8 (Stationarity). A time series is stationary if its distribution does not depend on the time at which the series is observed. ▪

The stationarity of a time series implies that

- the level of the series is constant (there is no trend);
- the variance of the series is constant;
- there is constant autocorrelation over time, i.e. the previous values describe well the upcoming value;
- the series has no seasonality;
- the series may have cycles because the peaks and valleys of cycles are not predefined.

Definition 9 (White Noise). A white noise is a special kind of time series with a constant level $\mu = 0$ and a finite and constant variance, making white noise stationary. White noise is uncorrelated, i.e there is no dependence between two timestamps. ▪

In recent decades, the processing of time series data in a streaming fashion has emerged as a popular area of research. These processing techniques include data behavior classification, data forecasting and anomaly detection. This study focuses on the latter.

3.1.2 Time Series Pre-processing

In time series data analysis the data is usually not processed in its raw format but there may be several pre-processing steps performed on it, in order to fulfill the requirements of the algorithm or make disappear certain characteristics. Data pre-processing has two main types.

Definition 10 (Online Pre-processing). An online pre-processing of a time series T at timestep t is a data transformation so that only information from $o^{(i)} \quad i \leq t$ datapoints may be used, and changes can only be made directly or indirectly using these datapoints. Illustratively an online pre-processing only has the time series available up to timestep t.▪

Definition 11 (Offline Pre-processing). An offline pre-processing of a finite time series T is any pre-processing that is performed on T using information from the whole series. Offline pre-processing can only be performed if every single value of T is available.▪

The following pre-processing steps are frequently utilized in time series analysis. The processes are defined for univariate time series. In case of multivariate time series the single time series is to be replaced with every channel of the time series.

Definition 12 (Data Normalization). The aim of normalization is to transform every data point proportionally so that $o^{(t)} \in [0, 1], \forall t \in |T|$. ▪

To do this proportionally the extrema of the series needs to be known, therefore, normalization is an offline pre-processing technique.

Definition 13 (Standardization). Standardization is the transformation of time series T, so that $\hat{o}^{(t)} = \frac{o^{(t)} - \mu}{\sigma}$ where μ is the mean of the time series and σ is the standard deviation of the time series. ▪

Since any datapoint can change both the mean and the standard deviation, all of the values of the time series are necessary to perform standardization, therefore this is an offline pre-processing method.

Definition 14 (Data Clipping). Clipping is the simple process of of clipping a value at a timestep if it is greater than or lower than the clipping maximum or minimum.
if $o^{(t)} > clip_{max}, \quad o^{(t)} := clip_{max},$
if $o^{(t)} < clip_{min}, \quad o^{(t)} := clip_{min}.$ ▪

Clipping can be done as new data points arrive, therefore it is an online pre-processing technique.

Definition 15 (Data Differencing). Data differencing is the computation of differences between consecutive observations.
Let $T' = o'^{(1)}, o'^{(2)}, ...o'^{(n-1)}$ be the first differenced time series of $T = o^{(1)}, o^{(2)}, ...o^{(n)}$. In this case $o'^{(t-1)} = o^{(t)} - o^{(t-1)} \quad \forall t > 1 \in |T'|$. Seasonal differencing is a special case of differencing, when instead of the next previous value, the value in the previous period is subtracted, i.e. $o'^{(t')} = o^{(t)} - o^{(t-p)} \quad \forall t > p \in |T|$. Logically, the length of the series is shortened by p values, as the first period has no previous period ▪

If we allow a lag of 1, at the first timestep, differencing can be done in an online fashion. Differencing can be performed multiple times. Every differencing process shortens the time series by one value.

Differencing helps stabilize the mean of the time series by removing changes in the level, thus eliminating trend and seasonality. One or two levels of differencing can make the data stationary which helps some analysis algorithms and is a requirement for others.

The Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test [26] can be used to check whether a time series is stationary. In this test the null hypothesis is that the series is stationary. We look for evidence for the contrary, if one is found, differencing is necessary. On the differenced series the test can be performed again, to determine, wheter further differencing is required. The test can only be performed as an offline pre-processing step, however, in an online manner a one or two step differencing is usually enough as a good heuristic [20].

3.1.3 Time Series Anomalies

There exist multiple definitions of an *anomaly* [1, 8, 14, 33, 16]. However, all the definitions grasp the same concept in different contexts. Using these, Braei and Wagner [7] define anomalies the following way.

Definition 16 (Anomaly). An anomaly is an observation or a sequence of observations that deviates remarkably from the general distribution of the data. The set of anomalies form a very small part of the dataset. ▪

In time series analysis this can have two meanings:

- **Unwanted data** are observations that – from the point of view of the analyst – do not represent well the data. These can be related to noise or erroneous data as well, and negatively impact data analysis or data forecasting. Unwanted data should either be omitted or corrected to improve data quality. This process is called *Data Cleaning*.
- **True anomalies** or **outliers** are data points or data patterns that do not conform to the normal data behavior. In the rest of this paper, we will use the term "anomaly" in this meaning. While these are almost certainly unwanted for data analysis, in recent years research in the area of finding these events of interest has gained traction. This process is called *anomaly detection*.

Outliers can be grouped into two main types The terminology used in the sub-types of outliers partially follows that of the IBM SPSS Modeler [21].

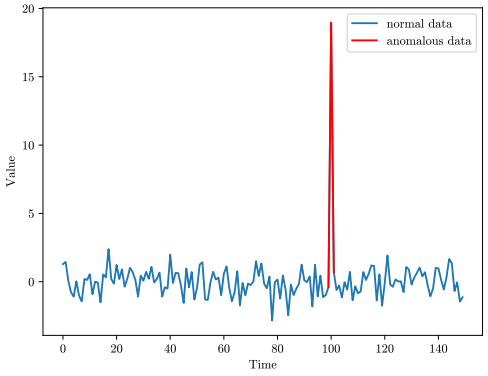
Point Outliers

Definition 17 (Point Outlier). A point outlier is a datum that exhibits unusual behavior at a specific time instance. If this discrepancy is true in comparison to all points of the time series, the outlier is a **global outlier**. In contrast, if it only differs greatly from the neighboring points, it is a **local outlier**, also called a **contextual point anomaly.**.

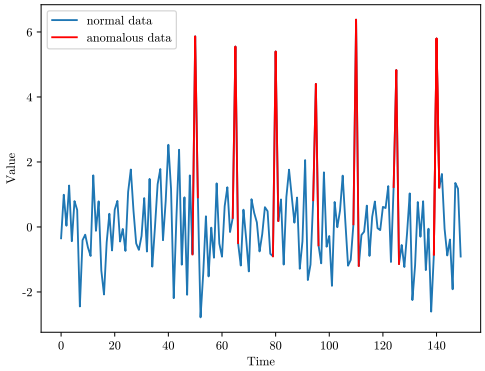
The unusual behavior may surface in the following ways:

- An **additive outlier** is a substantial shift in the level of the time series in comparison to other values. When plotting the time series, additive outlier anomalies manifest themselves in sudden, unexpected spikes in the positive or negative direction relative to normal data.
- **Seasonal additive outliers** are additive outliers that occur repeatedly at regular intervals. These do not necessarily appear only in periodic time series, but can develop in non-seasonal series as well. The seasonality refers to the anomaly itself.
- In periodic time series, point anomalies need not be extreme values, but a value out of period or an unexpected lack of one can also be considered a point outlier. This is called an **out-of-period outlier**.

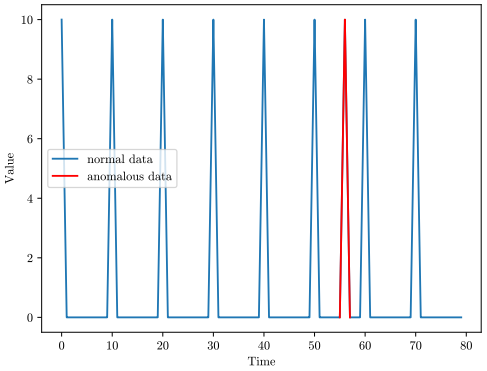
Point outliers can appear in univariate, as well as multivariate time series. In the latter case, the deviation can be seen in multiple recorded dimensions at the same timestep.



(a) An additive outlier in noisy data



(b) A seasonal additive outlier in noisy data



(c) An out-of-period outlier

Figure 3.1: Point outlier types

Figure 3.1 shows the types of point outliers on a sample time series. Figures 3.1a and 3.1b have white noise as their normal data. Figure 3.1c has periodic spikes at constant intervals as expected normal behavior and the anomaly manifests itself as a normal pattern at an unexpected time. The non-anomalous data is marked with the blue curve, while red shows anomalies.

Subsequence Outliers

Definition 18 (Subsequence). A subsequence of length $m < n$ of a time series is a contiguous sampling of $t^{(i)}$, ..., $t^{(i+m-1)}$ of points where

n = the length of the series,
m = the length of the subsequence,
i = an arbitrary position in the series such that $1 \leq i \leq n - m + 1$ and
t_i = the value (or set of values) measured at timestep i.

Definition 19 (Subsequence Outlier). A subsequence outlier is a subsequence whose behavior is unusual, whilst looking at its points individually, those might not be point anomalies. Subsequence anomalies can also be local or global in the same way as point anomalies. ▪

There are multiple types of these anomalies.

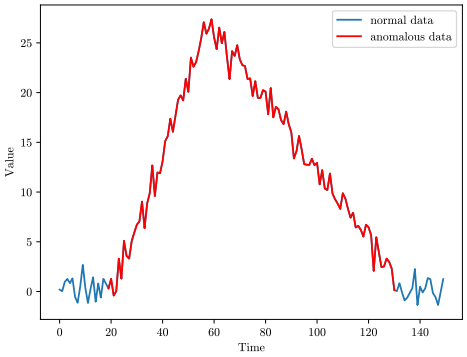
- An **innovational outlier** is a point anomaly, the effect of which remains present over subsequent observations. Its influnce might increase over time.
- **Transient change outliers** are similar, but their effects diminish exponentially over subsequent observations eventually returning to normal.
- A **level shift outlier** is a type of anomaly when – after an additive outlier – all observations move to a new level, i.e. it has a long-term or permanent effect. In other words, the behavior or pattern of the data changes with a level shift.
- **Local trend outliers** are similar but occur when a local trend, i.e. a gradual upward or downward shift in the level of the series, arises. Since the trend is local, normal behavior resumes after the anomaly.
- In multivariate time series an anomaly can happen even if in every dimension the data pattern shows normal behavior. This can develop when – examining all the dimensions for the same subsequence – the common pattern is unusual, i.e. the present normal behaviors normally do not occur together. This type of outliers is called a **correlation/cross-channel outlier** since the anomaly only emerges due to the correlation of the different dimensions. As an example, a correlation anomaly is when on a production line the units working at the same time are out-of-sync, i.e. even though they perform operations within normal bounds, the operations are not applied in the correct order or at the correct place, resulting in a faulty end-product.

In Figure 3.2 examples of each subsequence outlier type are shown in noisy data in the case of non-periodic data (Figures 3.2a to 3.2d) with white noise as normal data. Figure 3.2e shows an anomalous subsequence in otherwise periodic data. The blue color marks normal behavior while the red shows the anomalous subsequences with the distinct characteristics described above. Figure 3.2f gives an example of a correlation outlier in a two-channel multivariate time series. Here the uppermost and the middle graphs show two distinct normal behavior pairs. The appearing pattern in one channel implies the corresponding pattern in the other channel in both directions. The lowermost graph shows the anomalous pair. In this case, the red curve shows a normal behavior of $channel_1$ in the first case and the magenta curve shows normal behavior of $channel_2$ in the second case. None of the subsequences are considered an anomaly when looked at individually. However, the two never appear together as per the above implication rule. This discrepancy is the cross-channel or correlation anomaly.

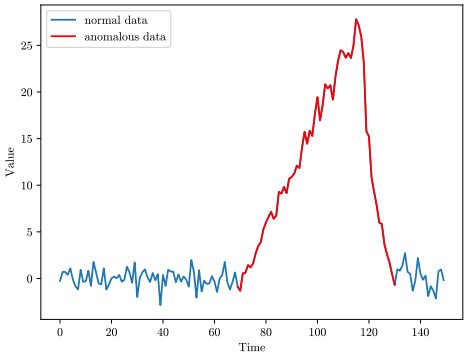
Changepoint Detection

In anomaly detection, outliers are mostly considered to be signs of abnormality. A higher number of traffic due to road closures, a failure in an assembly line, suspicious network traffic caused by a cyberattack serve as examples of this. However, anomalies could well be indication of changes in the system behavior to another normal pattern. Searching for the root-cause of an outlier carries great importance in order to differentiate abnormal behavior from changing normal behavior.

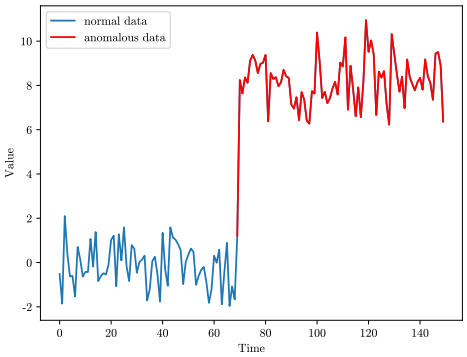
Level shifts and local or global trends are the most likely manifestations of this change, however, changing seasonality may also be a sign of this. The process of detecting these events which indicate changing behavior is called **changepoint detection** or **pattern change detection**.



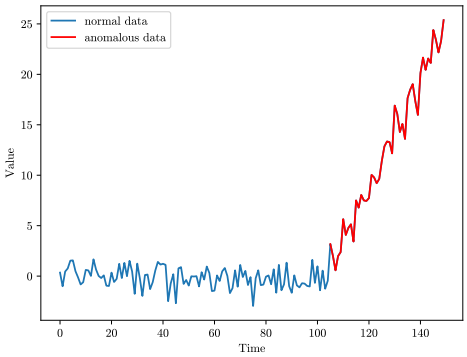
(a) An innovational outlier in noisy data



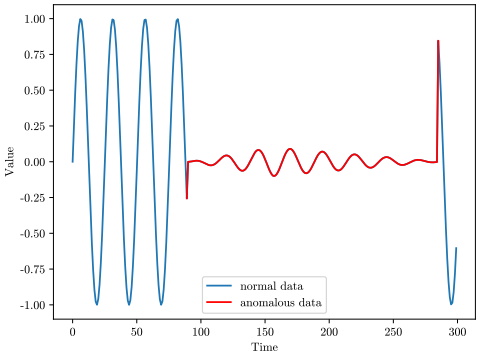
(b) A transient change outlier in noisy data



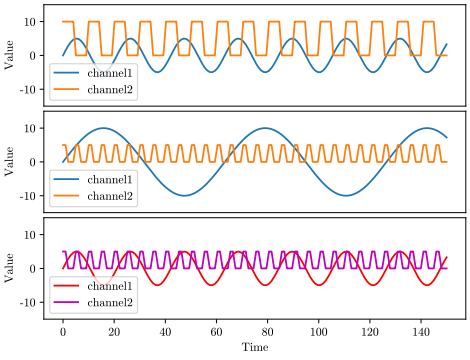
(c) A level shift outlier in noisy data



(d) A local trend outlier in noisy data



(e) An out-of-period subsequence outlier



(f) Correlation outlier across two channels

Figure 3.2: Subsequence outlier types

3.1.4 Time Series Anomaly Detection

General anomaly detection uses a subset of available data to calibrate or train the detection component so that it may offer the best performance on the observed data characteristics.

Definition 20 (General Definition of Anomaly Detection). Formally, given a multivariate time-series $\boldsymbol{X} \in \mathbb{R}^{n \cdot m}$ and calibration set $\boldsymbol{X}^{\boldsymbol{calibration}} \in \mathbb{R}^{n_1 \cdot m}$ with n_1 regularly sampled observations across m channels, anomaly detection can be written as the following assignment:

$$\phi : \mathbb{R}^m \mapsto \mathbb{R} \tag{3.1}$$

$$\phi(\boldsymbol{x}^{\boldsymbol{i}}) = \gamma \tag{3.2}$$

▪

where

$\boldsymbol{x}^{\boldsymbol{i}}$ = an observation of the test observation matrix ($\boldsymbol{x}^{\boldsymbol{i}} \in \boldsymbol{X}^{\boldsymbol{test}} \in \mathbb{R}^{n_2 \cdot m}$),
n_1 = number of observations in the calibration set,
n_2 = number of observations in the test set ($n_1 < i <= n_2$),
$\boldsymbol{X}^{\boldsymbol{test}}$ = observation matrix of the test set,
γ = anomaly score.

To identify the anomalous data points the anomaly score needs to be examined. In case of binary anomaly labeling, i.e. anomaly vs normal timesteps:

$$\phi_{binary} : \mathbb{R}^m \mapsto \{anomaly | normal\ data\} \tag{3.3}$$

$$\phi_{binary}(\boldsymbol{x}^{\boldsymbol{i}}) = \begin{cases} anomaly & if\ \phi(\boldsymbol{x}^{\boldsymbol{i}}) > \delta \\ normal & otherwise \end{cases} \tag{3.4}$$

where

$\boldsymbol{x}^{\boldsymbol{i}}$ = an observation of the test observation matrix ($\boldsymbol{x}^{\boldsymbol{i}} \in \boldsymbol{X}^{\boldsymbol{test}} \in \mathbb{R}^{n_2 \cdot m}$),
δ = anomaly threshold ($\delta \in \mathbb{R}$).

Thus, anomaly detection requires an anomaly threshold which acts as a separation point between anomalies and normal data points. The threshold can be constant for the duration of the detection and be only based on the calibration data, or alternatively it can dynamically change as the data characteristics change. The threshold calculation depends on the anomaly detection used.

Equation (3.1) describes a general definition of anomaly detection, however – depending on the algorithm – the process can be more complex. These include:

- **Data Pre-processing** Anomaly detection is normally not performed on raw data points but several steps of data pre-processing may be included. Such processes are normalization, standardization, differencing, distribution analysis, clipping.
- **Window-based Anomaly Detection** Some algorithms do not perform anomaly detection on individual data points but a set of contiguous data points, i.e. windows.

The windows can be overlapping or distinct. Windows-based anomaly detectors usually give the whole window an anomaly score, therefore the individual anomalous point is not identified.

- **Non-binary Anomaly Detection** There may exist multiple anomaly classes or changepoints are to be detected. Some detectors have the ability to distinguish these classes. To perform this, multiple thresholds and multiple anomaly score values based on different metrics may be calculated and the detection is made based on these.
- **Separate Channel Multivariate Anomaly Detection** In case of multivariate datasets anomaly detection can be performed by running the detector on all channels individually and either aggregating the anomaly scores into a common anomaly score, or making a decision based on the individual detection results for each timestep (or window). The downside of such an approach is that correlation outliers are likely to be missed.

3.1.5 Time Series Anomaly Detection Types

In contemporary anomaly detection research, there are several approaches that can be used for anomaly detection in time series. These can be grouped into categories based on various characteristics.

Supervision

- **Supervised Learning** If a time series is labeled, for every timestep (or window) it is known whether it is anomalous or not (or to which class it belongs to if present), i.e. every timestep has a label. Time series detectors can use this information during the calibration phase to determine the optimal threshold(s) to best classify the timesteps into the correct category. Such datasets require a substantial amount of effort to create, since an expert has to go through a time series by hand and label the anomalies.
- **Semi-supervised Learning** In semi-supervised learning only a subset of timesteps need to have a label. This usually means that only normal timesteps are used for the model training. Semi-supervised algorithms identify normal behaviors so that any future point that deviates from these can be marked as an anomaly. Alternatively if the new behavior becomes widely representative of the series, it may be added to the normal model as a new normal behavior and a changepoint detected instead.
- **Unsupervised Learning** Unsupervised learning needs no anomaly labels of any kind to calibrate the model and set the threshold. Such detectors usually use the data distribution or predefined percentage of anomalous data points in the data to do so. A commonly used method for threshold setting is the so called *Three Sigma Rule* $\delta = 3\sigma$, where σ is the standard deviation of the points of the time series (or of a subsequence of the time series) [17].

Model category Braei and Wagner [7] categorize anomaly detectors into three main categories based on the characteristics of their underlying model.

- **Statistical Methods** Statistical anomaly detectors assume that the data is generated by a specific statistical model [8]. The autoregressive model or ARIMA are examples of this category.

- **Classical Machine Learning (ML) Models** Classical machine learning models are similar to the statistical methods but they regard the data generation process as a black box. Without taking assumptions they try to fit the best possible model learning from the data only. K-Means Clutering DBSCAN, Isolation Forest, Onc-Class Support Vector Machines (OC-SVM) serve as examples of this category.

- **Neural Network Models/Deep Learning** A neural network is a network of connected nodes organized into multiple layers (hence the name deep learning). Each connection has a function and a coefficient and the input is transferred to the end-node of the connection using these parameters. The user only interfaces with the input and output layers of the model, the parameters of the hidden layers are fine-tuned so that the output is closest to the desired value. This is the most popular area for anomaly detection. Models like Convolutional Neural Network (CNN), Long-Short Term Memory (LSTM), Gated Recurrent Unit (GRU) and autoencoder belong to this category.

Functioning of the Model Another aspect of categorization is how the anomaly detector functions.

- **Forecasting-based Models** Data forecasting in time series means the prediction of the upcoming value or subsequence. Many anomaly detectors use this type of model in their core. The aim of this technique is to unify a time series predictor and an anomaly detector. Forecasting assumes that using the past values in the time series (or a subset of them) with the addition of a random value (usually white noise), the value (or subsequence) at the next timestep can be predicted with great confidence. Then, comparing the difference in magnitude with the actual value, if the difference is great, i.e. it exceeds a previously tuned threshold, the likely cause is an anomaly. For forecasting any kind of model may be used, be it the statistical, machine-learning- or neural-network-based. The role of the anomaly detector is to set the threshold. Deep learning predictors are usually accurate because these can effectively learn even peculiar data behaviors. However, due to the prediction component, the performance of approaches of this kind will suffer when only limited data is available for training. The quality of prediction will be lower in this case, and with it, the quality of anomaly detection.

- **Clustering-based Models** The aim of clustering methods is to group the data points together based on underlying common patterns. These are semi-supervised approaches which create a number of normal clusters. The assumption is that anomalous values will fall outside of these clusters, i.e. the difference to all of the clusters will exceed a threshold or a set of thresholds. These methods have the potential to manage multiple distinct normal or anomaly clusters and adapt to changing data behavior as the population of a previously outlier cluster grows and exceeds a threshold. However, processing complex multivariate time series with a large amount of features might have a significant hit on detection time.

- **Data Reconstruction** Autoencoder anomaly detector is the main semi-supervised reconstruction-based anomaly detection method. A unique property of autoencoders is that their output is expected to equal its input. The reason is that the main purpose of autoencoders is dimensionality reduction. This is achieved by consistent reduction of nodes in the the hidden layers. These layers together are called encoder. In the middle there is the layer with the fewest nodes. This is the bottleneck where

the data is represented in a latent space at a lower dimension. The same pattern in reverse is repeated towards the output in the decoder part. The structure of the autoencoder can be seen in Figure 3.3. An autoencoder can be trained by tuning a weights so that the output gets closest to the input. In this case, the lower dimensional bottleneck represents well the latent features, main patterns of the data, and can be used for complex processes. Autoencoders are not limited by linear algebra and can model more complex reduction functions. Autoencoders

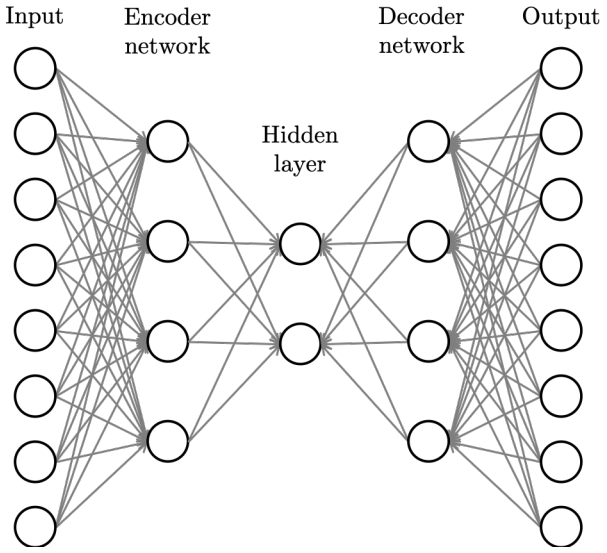


Figure 3.3: Model structure of an autoencoder

are useful for dimensionality reduction by taking the latent space representation, however, can also be utilized as an anomaly detector. Here the assumption is that – after training – on new normal data the difference between input and output values will be minimal. However, on anomalous datapoints, this difference gets substantially larger as the model is only trained to reproduce normal data behavior. While these models can achieve great results in anomaly detection, choosing a threshold or latent pattern analysis is necessary for behavior classification. Training the model is also a time consuming task.

- **Adaptive Threshold Methods** Adaptive threshold anomaly detectors use the calibration data to set up a threshold that adapts to the data distribution and use the relationship of the raw or pre-processed data to this threshold to decide which points are anomalous. They are similar to clustering-based methods in that they define a normal cluster that is below the threshold and an anomaly cluster that is above. The threshold can be dynamically adjusted as the data distribution changes. Streaming Peaks-over Threshold (SPOT) [38] is an example of these approaches.

3.2 Data Analysis in Industry 4.0

Industry 4.0 – sometimes also called the 4^{th} *industrial revolution* – is the fourth major shift in manufacturing. After the proliferation of specialized manufacturing, mass-scale production lines and the application of robotics to the manufacturing process, the main concept of Industry 4.0 is the digitalization of the existing systems. The goal is to use all resources available to monitor the manufacturing process and make automatic self-

diagnosis possible. A key factor that makes this possible is the newfound connectedness of hardware which in an industrial context is referred to as Industrial Internet of Things or IIoT for short.

3.2.1 Industry 4.0 in Theory

Professor Hardt from MIT explains in a talk [35] that the cornerstone of Industry 4.0 is the capture of data in every part of the organization (not just the production line) and the processing of this data. According to him the steppingstones to achieve this is first the transformation from paper-based records to electronic records (Industry 4.0.0) and second the visualization of the collected data in a central dashboard which fuses multiple data sources (Industry 4.0.1). He explains that the reason we can suddenly implement these techniques is not because any of them were previously absent and have just recently been discovered. Quite the contrary, all the technologies the new system builds on (the Internet, powerful PCs, industrial robots, wireless communication, AI, 3D printers, IoT) are relatively old, having been pioneered before the beginning of the century. The change is due to financial reasons. The price has only recently dropped to an economical level while performance has greatly increased. According to Professor Hardt, the goal of industry has not changed with Industry 4.0. It is the continual flow of new material (in factory and in the supply chain) and the production of products with minimal variation. Variation is influenced by the procedure, the equipment and material used but also by a stochastic variable. There are four success factors which a company should optimize: quality, production rate, cost and flexibility.

The idea of Industry 4.0 is therefore to know exactly the state of every process and equipment in *real time*, the ability to analyze the collected data in the cloud with machine learning methods also in real time and the automation of the manufacturing business itself by taking over or making suggestions for decision making and execution of such decisions.

3.2.2 Digital Transformation

4.0 Solutions [40] is a for-profit educational and industrial mentor organization that is dedicated to helping system integrators at companies transform their manufacturing and business to an Industry 4.0 compatible level, a process they aptly call *digital transformation*. Their goal is to achieve this on a level that

- everything and everyone is connected to the network;
- the layers of the business are integrated and operate based on data and information from all other layers in real time;
- stakeholders know the current and future state of the business;
- Artificial Intelligence (AI) and Machine Learning (ML) are leveraged to analyze data;
- Machine Learning predicts future outcomes based on past patterns and current state;
- Artificial Intelligence recommends operational adjustments for stakeholders to improve future outcomes.

This way a fully integrated business made up of so-called *digital factories* can be created. One point they emphasize is the importance of a *Common Enterprise Namespace* for the

data collection, analysis, monitoring and control. All systems publish data in this or subscribe to data from this namespace. The architecture of such a business as well as that of a legacy business is visualized in Figure 3.4.

Legacy Manufacturing Business Architecture

The legacy architecture, called the *Automation Stack* [40] is composed of layers. The data can only be exchanged between adjacent layers.

The lowest level in the legacy architecture – also referred to as Edge – is the hardware and software on the production plant floor, usually integrated into a production line. It includes sensors, switches, routers, actuators, instruments and Programmable Logic Controllers (PLCs) that are hardware-software devices which provide and interface to run automatized programs on the production hardware. The Human-Machine Interface (HMI) devices are also found on the plant premises. The operator uses these to interact with PLCs and – indirectly – with production devices to perform and monitor the job assigned to them.

The Supervisory Control And Data Acquisition (SCADA) collects data, performs real-time process analysis and provides supervisory control on a plant-level basis. The system is also used for handling dispatching, notifications and alarms. The supervisors monitor the state of the machines through it, and the control room can direct distributed personnel to production lines that have raised an alarm.

The Manufacturing Execution Systems (MES) is the software that manages the work order that is set out by the ERP for that particular production site. It invokes the production run, collects data during the run, and reports back the results to the ERP once the run has finished. This level in the stack serves as the connection between the business decision makers and the plant floor.

The Enterprise Resource Planning (ERP) is a software system on the enterprise-level that contains all the intellectual property of the business including product details, employee and business information, supply chain information and company architecture. The production goals are set in this system, and budget is estimated by it. Other than business decision makers, the accounting and IT departments all interact with this level of the business.

On top of these levels machine learning and artificial intelligence can be leveraged for business optimization.

IIoT Architecture

In the legacy architecture accessing data from deeper layers in the Automation Stack is not straightforward, as the data needs to pass through all intermediate layers. Furthermore, interfacing between adjacent layers requires a substantial amount of engineering by system integrators as different systems are often developed by distinct companies.

The IIoT architecture aims to seamlessly integrate all these layers, so that data from all levels of the business can be accessed by any other layer through the Common Enterprise Namespace. This helps cross-compatibility among products of distinct vendors, and opens the door for data data analysis by making information be easily accessible.

This way not only can a ML algorithm consider information from different sites and various levels of the business better exposing connections in the data, but a plant worker can also

monitor and visualize data for themselves, compare information from multiple identical machines (even from other production sites) and give suggestions and feedback on how to improve the system based on past experience.

IIoT devices may already have a functionality for this, e.g. by incorporating standard TCP/IP connectivity, while legacy devices may also be connected through a Bridge to the namespace.

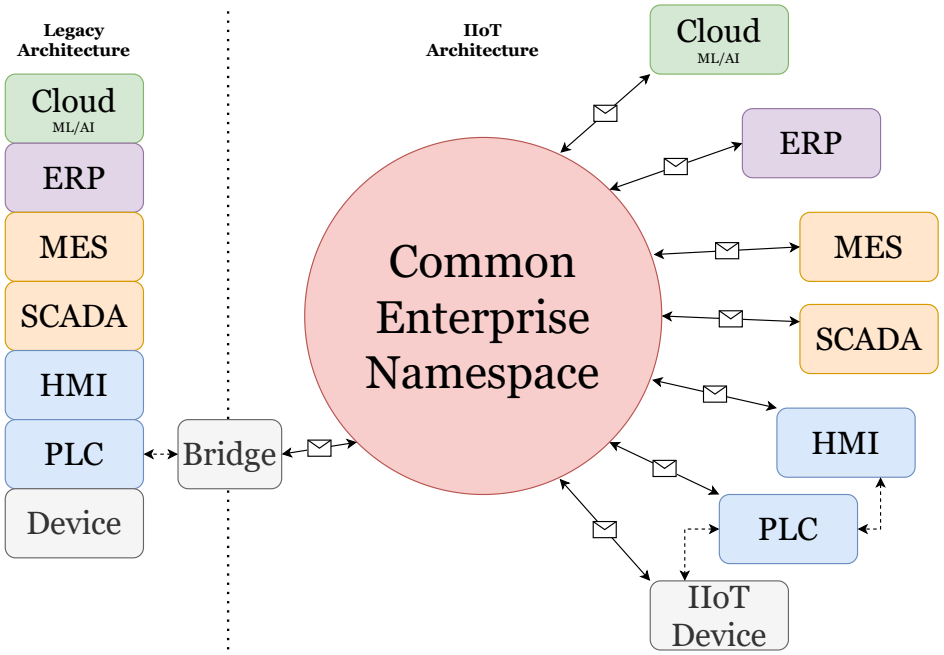


Figure 3.4: Architecture of Industry 4.0 integrated business showing both the Legacy Architecture and the novel IIoT Architecture utilizing Common Enterprise Namespace

3.2.3 Data Processing Techniques in an Industrial Context

A Case-Study of Approaches

Diez-Olivan et al. [9] in a survey article try to categorize data processing techniques used in an industrial context. They also collect and describe the utilized industrial ML data processing methods as of 2019. Their terminology refers to data-based workflows as *prognoses*. There are 3 main types of prognoses in use.

Descriptive Prognosis Descriptive prognosis uses the captured data only to describe the case under study. This means that these kinds of approaches ignore all a priori assumptions regarding the data and do not take any assumptions on the possible root cause of a problem either. This way no bias will be present in the obtained insights from data, which brings the added value of such an approach. In practice this type of prognosis usually utilizes unsupervised learning with clustering techniques and outlier detection for fault detection which can effectively learn the presence of patterns of interest in the data without prior knowledge. According to the research of the authors, this is the most common prognosis type discussed in contemporary works. Articles in this area can be classified to *pattern recognition or classification* or *health management* approaches. Anomaly detection falls into this category.

Predictive Prognosis The goal of Predictive prognosis is to predict the time a fault in a monitored system will occur and give estimates on such a fault's severity and coverage over the production chain. These approaches leverage information on fault events from a dataset of past events. Learning algorithms of this kind can learn the pattern beforehand and correlate the captured data to the target variable based on this knowledge. Such applications are typically supervised learning approaches. The articles in this field usually discuss *condition-based or predictive maintenance.*

Prescriptive prognosis The third type of prognoses is called prescriptive prognosis which aims to prescribe optimal actions as a result of a fault alarm. There can be two types of scenarios this can occur.

Should a predictive model raise the alarm the aim is to reduce the chances that the fault will occur. This can be done by modifying working parameters and variables affected by the fault. The other scenario is when a descriptive model raises the alarm which in turn means that the occurrence of a fault has been confirmed. In this case the goal is to minimize the impact of the problem over the production of the industry. Methods such as rerouting assets or allocating human resources for unexpected maintenance can be effective. Prescriptive prognosis approaches try to solve optimization problems therefore *optimization solvers* are used. *Production scheduling*, *life cycle optimization* and *supply chain management and logistics* are the main approaches.

The different prognosis techniques can cooperate as we saw in the case of prescriptive prognosis which directly builds on the performance of the two other types. Hybrid approaches also exist which cannot be firmly categorized into only one prognosis type.

A system which implements all the prognosis techniques in every part of said system can fulfill the requirements regarding data processing set out in Section 3.2 for an Industry 4.0 integrated business. However, this is only true if real-time operation can be achieved by the implemented techniques.

Challenges for the Future

The article also outlines the future trends. The authors assign the task of developing visual analytics for an enhanced understandability to descriptive prognosis. This is to help make IIoT technologies widespread by providing feedback to workers and management of the functioning system making them able to crosscheck the performance of the system with the data and prior knowledge. This is in line with the *Common Namespace* approach of 4.0 Solutions described in Section 3.2.2.

For predictive prognosis, the biggest challenge is overcoming class imbalance which is the unequal representation of fault data compared to the normal behavior. If this relatively low occurrence is unaccounted for, there is a potential bias towards the normal class. To avoid this problem a few learning techniques can be applied. *Online learning* incrementally updates the model with the arrival of new data. It can control for changes in equipment which can change due to different working regimes, drift, re-calibration, or the lack of calibration. If a model is not capable of this, the number of false alarms could increase. Another learning method is *Transfer Learning and Domain Adaptation.* It works by transferring prior knowledge from a different model and using this knowledge as the base of the new model. This can account for the scarcity of data by transferring knowledge between identical equipment and for different working conditions (maintenance policies, personnel

skills and quality of raw materials can be different in distinct locations). This property can be important for a large multinational corporation.

The challenge of prescriptive prognosis is to operate inside complex constraints and with realistic objectives. In practice it must plan for the real environment: connection between machines and humans, time and financial constraints, balance between productivity and reliability; and be a cost-effective solution for production.

Integration of expert knowledge and physics models can improve model quality as well, however, usually these kinds of information are difficult to gather.

Constraints of Machine Learning Approaches in an Industrial Context

When implementing ML techniques in an industrial context many challenges arise. The first obstacle is the sparsity of available data. For research purposes there is a small amount of publicly available data, and these often contain holes, missing variables which either need to be dropped or carefully imputed. Wide-scale availability of data with high measuring frequency from every level of a manufacturing business is virtually nonexistent. When enough data is available, it can often be overwhelming since it is of a complex system with several variables. Feature selection can be challenging.

A unique property of the industrial systems is that recreating an anomaly can be difficult if not impossible making testing hard. This is because the systems are complex and artificially changing their behavior can directly affect processes down the line and have a financial impact. Furthermore, since the abnormality depends on the qualities of a system, many solutions cannot be effectively used for different scenarios, therefore model creation requires special data processing methods for a system.

Other challenges are that false positives and negatives can not only cause sub-optimal manufacturing performance, but can also harm the system itself; the behavior can change with time and models need to account for this. The cost of a false detection may also differ from the cost of the lack of detection of an anomaly. Anomalies are hard to detect because they can be obscured by noise as well.

Industrial data if often multivariate with measurements from multiple sensors of multiple variables. This further obscures the real anomalies and presents a couple of additional challenges:

- The irrelevant variables need to be filtered out, this can be done manually or preferably in an automatized way.
- Users require the solution to help identify where, i.e. in which dimension and at around which time point an anomaly occurred. Simply stating that there was an anomaly somewhere is not enough since this impedes information about the issue crucial to prevent it in the future.
- Correlation/cross-dimension anomalies may be present.

3.2.4 Anomaly Detection in Industrial Time Series Data

This study focuses on descriptive prognosis in industrial data, namely anomaly detection. There are multiple constraints that need to be addressed and overcome to perform it with the requirements set out in Section 3.2.2. The main restriction is that the detection has

to be performed in real-time. This implicitly brings about the following requirements of anomaly detection methods:

- For the calibration process there may be only a relatively small amount of datapoints available. This is to decrease the potential delay in the start of detections. An ideal detector is able to start working at the very first datapoint, fine-tuning its model as new data arrives, instead of waiting for a large number of datapoints for time consuming prior calibration. This requirement can be eluded if the model is previously calibrated from historical data, instead of at the beginning of the detection.
- If calibration is performed using the beginning of the same time series as the one used for detection, the calibration time is to be relatively short. This is also to lower delay till the first actual detection made.
- The inference – in this case actual anomaly detection – time needs to be relatively short as well. Anomalies are to be detected with minimal delay, therefore a prompt detection is desireable.
- Offline data pre-processing can only be applied to the calibration part of the time series or only online pre-processing can be perfomed.

Chapter 4

Algorithms

4.1 Selected Time Series Anomaly Detection Algorithms

4.1.1 SPOT

Streaming Peaks over Threshold (SPOT) by Siffer et al. [38] is an unsupervised, probabilistic, adaptive threshold anomaly detection tool for univariate time series data with high throughput, i.e. with quick data processing. The authors set out to learn the normal behavior of a time series and setting an anomaly threshold based on this, using a statistical technique called Extreme Value Theory (EVT) [4].

Extreme Value Theory

Extreme Value Theory was developed by the study of the law of extreme values in the distribution function following dramatic events. It describes the distribution of these extreme values and states that the distribution of the extreme values is almost independent of the distribution of the underlying data in which the extreme values appear, however, the extreme values follow a common distribution. Illustratively, the occurrence of rare events is similar regardless of the domain the data is from.

The distributions of extreme events are called Extreme Value Distributions (EVDs) and follow the following form:

$$G_\gamma : x \mapsto exp(-(1+\gamma x)^{-\frac{1}{\gamma}}), \qquad \gamma \in \mathbb{R}, \quad 1+\gamma x > 0 \tag{4.1}$$

The *extreme value index* (γ) depends on the initial distribution, e.g. in the case of Gaussian distribution, $\gamma = 0$. Function $\bar{F} = \mathbb{P}(X > x)$ represents the tail of the distribution of random variable X, this is where the extreme values lie. G_γ tries to fit γ so that the resulting function fits $\bar{F}$ best.

The power of EVT is that it infers the distribution of the tail without knowing the initial distribution. EVT states that the maximum of n independent and identically distributed random variables converges in distribution to the normal distribution, similarly to the *central limit theorem* which states the same for the mean of the variables.

If a distribution is inferred for the tail, from a given probability q it is possible to calculate z_q so that $\mathbb{P}(X > z_q) < q$. Here z_q gives us a threshold. Values greater than the threshold

are so extreme that their appearance is less probable than q. Of this an anomaly detector can be built, where the only parameter necessary is parameter q, essentially, how rare are anomalies in the data.

Peaks-Over-Threshold (POT)

To get the value of γ, estimations can be used which may only work well for certain tail behaviors. A more effective solution, *Peaks-Over-Threshold* is used by the authors. The Pickands-Balkema-De Haan Theorem [22] states that data values that exceed a threshold t are likely to follow a Generalized Pareto Distribution (GPD) with parameters γ, σ and $\mu = 0$. Using this theorem, POT tries to fit a GPD to the excesses themselves, rather than fitting an EVD to the extreme values of the distribution of the data. With this approach the anomaly detection quantile can be calculated in the following way:

$$z_q \simeq t + \frac{\hat{\sigma}}{\hat{\gamma}} \left(\left(\frac{qn}{N_t} \right)^{\hat{\gamma}} - 1 \right) \tag{4.2}$$

where

$\hat{\sigma}$ = estimate of σ,
$\hat{\gamma}$ = estimate of γ,
q = the desired probability,
n = number of total observations,
N_t = number of peaks where the values are over threshold t.

Maximum Likelihood Estimation

To estimate the values of σ and γ, maximum likelihood estimation is used using the first n observations. If $X_1, ...X_n$ are n independent realizations of the random variable X with density function f_θ the joint density of these n observations, also called likelihood function is:

$$\mathcal{L}(X_1, ...X_n; \quad \theta) = \prod_{i=1}^{n} f_\theta(X_i) \tag{4.3}$$

Given that in this case all the n observations are known, i.e. $X_1, ...X_n$ are fixed, the problem is setting the parameter θ so that the likelihood is maximized, meaning the observations are the most probable. In the case of the GPD fit the following log-likelihood needs to be maximized:

$$log\mathcal{L}(\gamma, \sigma) = -N_t log\sigma - \left(1 + \frac{1}{\gamma}\right) \sum_{i=1}^{N_t} log\left(1 + \frac{\gamma}{\sigma} Y_i\right) \tag{4.4}$$

where

$Y_i > 0$ = the excesses of X_i over threshold t.

Therefore the equation the roots of which we are looking for is

$$\nabla log\mathcal{L}(\gamma, \sigma). \tag{4.5}$$

Grimshaw's Trick

This optimizations is a lengthy process numerically. To speed it up, the authors use the procedure of Grimshaw [15]. The essence of this is to reduce the two parameter optimization to a one parameter problem. Grimshaw has shown in [15] that if the tuple of (γ^*, σ^*) is a solution to Equation (4.5), then variable $x^* = \gamma^*/\sigma^*$ is a solution to

$$u(x)v(x) = 1 \tag{4.6}$$

where

$$u(x) = \frac{1}{N_t}\sum_{i=1}^{N_t}\frac{1}{1+xY_i},$$
$$v(x) = 1 + \frac{1}{N_t}\sum_{i=1}^{N_t} log(1+xY_i).$$

By finding the solution x^* to this problem the (γ^*, σ^*) candidate pairs can be calculated. In the end only the best pair $(\hat{\gamma}, \hat{\sigma})$ which yields to the maximum likelihood is kept.

After optimization, the intervals where the authors look for roots (x^*) for Equation (4.6) are:

$$\left[-\frac{1}{Y_M} + \epsilon, -\epsilon\right] \qquad \text{and} \qquad \left[2\frac{\bar{Y} - Y_m}{\bar{Y}Y_m}, 2\frac{\bar{Y} - Y_m}{{Y_m}^2}\right]$$

where

Y_M = maximum of Y_i values
Y_M = minimum of Y_i values
$\bar{Y}$ = mean of Y_i values.

ϵ is a small real number to avoid $x = \frac{1}{Y_M}$ where Equation (4.6) is undefined and $x = 0$ which is always a solution. The authors used $\epsilon = 10^{-8}$. Instead of the root search a function minimization is carried out numerically to further aid computation time and since no general solution for scalar function root search exists. In each interval, the sum of the square of values from a sliding window is minimized to get a list of root-candidates. The log-likelihood is checked for all of them to maximize that and get the detection threshold.

The Functioning of the Algorithm

The user parameters necessary are the number of values for the initialization step n and the probability of anomalies q. The initial threshold t needs to be lower than the actual threshold z_q, the authors set it to the $98th$ percentile of the sorted initial data. Then the peak values Y are filled. $\hat{\gamma}$ and $\hat{\sigma}$ are calculated with the Grimshaw procedure, and the z_q threshold by Peaks-Over-Threshold.

After the initialization step the algorithm is ready to decide for each data point arriving in a streaming fashion whether it is normal or anomalous. If a value exceeds z_q it is marked as anomalous. If the value does not exceed threshold z_q but it is higher than the initial threshold t, the value is added to the peaks (Y) and z_q is calculated again with the new peaks using the procedure above.

Threshold setting and detections are possible in both positive and negative directions, that is additive anomalies can be found whether they are abnormally high or abnormally low values. The variant that uses both thresholds is called BiSPOT.

DSPOT

The authors also put forward a variant of the solution *Drift SPOT* which can adaptively move the threshold as drift is introduced in the time series.

In this case instead of SPOT running on the absolute values (X_i), it runs on the relative $X'_i = X_i - M_i$ values. Here M_i represents the local behavior which is calculated by the moving average of values form a d sized window the following way:

$$M_i = \frac{\sum_{k=1}^{d} X^*_{i-k}}{d} \tag{4.7}$$

where

$X^*_{i-1}, ...X^*_{i-d}$ = the last d normal observations
d = sliding window size

The window does not contain any abnormal values and therefore may be non contiguous.

Issues Found with the Algorithm and the Changes Made

The source-code for the algorithm is available on GitHub [3]. For the evaluation part of our work a few changes needed to be made to the source-code to correct semantic errors resulting from edge-case scenarios.

- When populating the peaks (Y), only those values are taken into consideration which are greater than the initial threshold t. This works well for diverse data where there are peak values greater than this threshold. However, in the case that there are no values greater than t, only equal to it (which may occur in TS data with discrete values and data where an upper limit is capping the values) there were no selected peak values. To solve this problem, I modified the peak selection, so that it selects values equal to the threshold as well.
- When looking for possible roots of x from intervals in Section 4.1.1, it may happen that in a limit to one of the intervals (or both) the denominator is 0. This can happen if Y_M, Y_m or $\bar{Y}$ is 0, i.e. the maximum or minimum or mean of the peaks is 0. This would result in an undefined fraction with the limit in positive or negative ∞. Since looking through infinite values with the numerical function optimization approach utilized instead of root search is impossible, I have made the decision that only the ranges with defined finite end-points are to be searched.
- When calculating the log-likelihood, if the value of σ is 0 or if the value of σ is not 0 but the value of γ is, a mathematical error will be raised as either the log function has it as input or a division is made by it. This can happen at the first calculation step when σ is chosen to be 0 as it is always a root and γ is $\bar{Y}$. If the latter is 0, the logarithm will be undefined. To avoid limits and (negative) infinite likelihoods I chose the likelihood of this to be 0.

- The authors accidentally added ϵ to the limits of the intervals in Section 4.1.1 twice resulting in possible errors during execution. I removed on of them to be in line with the documented algorithm.

4.1.2 FluxEV

FluxEV by Li et al. [28] is a similar unsupervised solution to SPOT, based on the same principles, using a forecasting component and data differencing. However, the authors pinpoint a few shortcomings of the approach. They realize that SPOT is only applicable for extreme outlier detection (additive outliers) and cannot detect anomalies that are inside the bounds of the normal data distribution but are represented by abnormal patterns (seasonal additive outliers or subsequence outliers). While this type of anomalies is common in streaming data, SPOT fails to detect them.

To solve this problem, the authors propose a data pre-processing solution that transforms the data into a format that SPOT behaves effectively on. This is achieved by discovering the features of normality in the data and minimizing them by a two-step smoothing process. At the end of this process only the abnormal behavior remains with an emphasized amplitude which will be the extremes of the smoothed dataset. The data pre-processing procedure is detailed in the following subsections.

The authors also made a modification in SPOT by changing the parameter estimation function from Grismshaw's trick to Method of Moments. The latter is a substantially less complex computation which potentially results in faster execution times.

Fluctuation Extraction

The first step in the process is using a simple predictor to predict the next expected value. This is done by the simple algorithmic Exponentially Weighted Moving Average (EWMA) [19] approach which – as its name suggests – calculates the average of data points inside a moving window with exponentially decreasing weights assigned to values inside the window the older the observations are. The predicted value is calculated using the following formula:

$$EWMA(X_{i-s,i-1}) = \frac{\sum_{k=1}^{s}(1-\alpha)^{k-1}X_{i-k}}{\sum_{k=1}^{s}(1-\alpha)^{k-1}} \tag{4.8}$$

where

i = timestep to predict,
X_i = value at timestep i,
s = moving window size,
α = smooting factor $\alpha \in [0, 1]$.

EWMA is used as a simple predictor instead of a more complex solution like LSTM to allow a faster processing time. Following the work of Li et al., alpha is set to 0.4. Having the expected value and error is calculated between the real value and the predicted one:

$$E_i = X_i - EWMA(X_{i-s,i-1}) \tag{4.9}$$

where

X_i = value at timestep i,
$EWMA(X_{i-s,i-1})$ = predicted value at timestep i,
s = moving window size,
E_i = prediction error.

First-step Smoothing

The authors define two key ways anomalies are usually present in data:

Local fluctuations are any fluctuation, i.e. change in data pattern, that is an indication of abnormal behavior. Additive outliers belong to this group, however smaller fluctuations may also be present, e.g. in a dataset with relatively stable time slots a small, non-extreme fluctuation may be an anomaly, whereas with unstable datasets, even greater fluctuations may be normal patterns.

Periodic pattern fluctuations are abnormal forms of behavior in periodic datasets where the anomalies are a set of datapoints that do not conform to the data period.

To eliminate the normal behavior due to noise in unstable data and the normal periodic pattern, a two-step smoothing approach is proposed.

The **first-step smoothing** is used to eliminate local noise by processing the error values (E_i) sequentially (i.e. with a sliding window of size s) with Equations (4.10) and (4.11).

$$\Delta\sigma = \sigma(E_{i-s,i}) - \sigma(E_{i-s,i-1}) \tag{4.10}$$

$$F_i = max(\Delta\sigma, 0), \tag{4.11}$$

where

$E_{x,y}$ = the list of errors/fluctuations from x to y,
$\sigma(X)$ = the standard deviation of list X,
$\Delta\sigma$ = the change of standard deviation, in this case when E_i is added to the window,
F_i = the fluctuation value at timestep i.

The idea behind this is if the value added to the local window causes a great fluctuation, this is because it causes an increase in standard deviation (which is caused by an unusual error value which is a sign of a poor prediction, i.e. a potential outlier). Whereas, if a normal noisy value is added as the new error, the difference will approach zero. The max operation means that if the new value decreases the standard deviation in the local window, i.e. it is even less fluctuating, the new point is considered normal and a fluctuation value of 0 is assigned.

Second-step Smoothing

The **second-step smoothing** is utilized to eliminate the normal patterns resulting form periodic behavior. This is desirable, since periodic extreme values are usually the sign of expected behavior, yet approaches which only look at the raw value like SPOT would likely categorize every spike as an anomaly.

One challenge with achieving this is that drift may be present in the data, owing to this data values do not ususally line up perfectly from period to period. To overcome this, the

authors use a sliding safety window around the datapoint under consideration instead of just one point. If the current point is X_t and the period length is l to calculate the value at p^{th} period before the current period instead of X_{t-pl}, $max(X_{t-pl-d'}, ...X_{t-pl}, ...X_{t-pl+d'})$ is used.

The second-step period smoothing is performed using Equations (4.12) to (4.14).

$$M_{i-d'} = max(F_{i-2d',i}) \tag{4.12}$$

$$\Delta F_i = F_i - max\left(M_{i-l(p-1)}, ...M_{i-2l}, M_{i-l}\right) \tag{4.13}$$

$$S_i = max(\Delta F_i, 0) \tag{4.14}$$

where

d' = half the window size to handle potential data drift, $d' = \frac{d}{2}$,
$F_{x,y}$ = list of fluctuation values between x and y,
M = array to store the local maximums of F,
S_i = fluctuation value after the second step smoothing at index i.

The rational behind this is that in M we store the local maxima of the fluctuation values of the previous p periods. This serves as an upper limit of the normal fluctuations at the current timestamp. Taking the maximum of these we select the largest possible fluctuation. If the fluctuations are present due to periodic patterns, then the fluctuations will be similar in each period, therefore by subtracting the maximum possible fluctuation of the current fluctuation F_i the result will approach zero. However, should the local fluctuation be greater (as previously explained caused by an unusual prediction error), the periodic fluctuations will not be enough to eliminate it and therefore, an extreme value will remain. Similarly to the first-step smoothing, if the current fluctuation is smaller than the previous ones, the current timestep is considered normal by assigning a zero fluctuation value.

Functioning of the Algorithm

To run all layers of FluxEV one needs to provide periodic streaming datasets for the algorithm. Alternatively, the algorithm can be modified - as we have done in our tests - to run on non-periodic data by just using the first-step smoothing process. For periodic datasets the period length l, the number of periods to look back for calibration p, the sliding window size used for fluctuation extraction and first-step smoothing s, and the drift compensation window size used for the second-step smoothing d needs to be provided. Additionally, k is the number of data points to initialize SPOT and q is the risk parameter used in the same way as in SPOT. Altogether the algorithm needs $2s + d + l(p-1) + k$ number of datapoints to fully initialize. After this, a real time operation is possible by calculating the predicted value, the error, the first-step smoothing fluctuation value, the second-step fluctuation value on the fly.

Detection and re-calibration works in the same way as in SPOT, if the value is above the dynamic threshold and anomaly is signaled. If it is below the threshold but above the initial threshold, the value is added to peaks and the threshold is readjusted with the new peak value taken into consideration.

To speed up processing instead of the Grimshaw procedure, Method of Moments (MOM) [5] is used to estimate the best values of $(\hat{\sigma}, \hat{\gamma})$. This is done the following way:

$$\mu = \sum_{i=1}^{N_t} \frac{Y_i}{N_t} \qquad S^2 = \sum_{i=1}^{N_t} \frac{(Y_i - \mu)^2}{N_t - 1} \tag{4.15}$$

$$\hat{\sigma} = \frac{\mu}{2}\left(1 + \frac{\mu^2}{S^2}\right) \tag{4.16}$$

$$\hat{\gamma} = \frac{1}{2}\left(1 - \frac{\mu^2}{S^2}\right) \tag{4.17}$$

where

Y = the set of peaks
N_t = the number of peaks.

MOM uses these simple calculations in Equation (4.15) to substitute the expected value ($\mathbb{E}(Y) = \frac{\sigma}{1-\gamma}$) with μ and the variance ($var(Y) = \frac{\sigma^2}{(1-\gamma)^2(1-2\gamma)}$) with S^2 of GPD.

Issues Found with the Algorithm and the Changes Made

The same problems were present in the source-code of the algorithm [45] reproduced by Wang as detailed in Section 4.1.1. I applied the same solutions to fix the issues.

Additionally, the use of Method of Moments to estimate the values of $(\hat{\sigma}, \hat{\gamma})$ may introduce another runtime error. Should the variance of the time series (in the case of MOM S^2) be 0 – i.e. the standard deviation is 0 so the series is constant, since there is no deviation from the mean – the estimate for $\hat{\sigma}$ and $\hat{\gamma}$ will be undefined because S^2 appears as a denominator in Equations (4.16) and (4.17). During the execution of the algorithm the threshold only needs to be recalculated when the value at the current timestamp exceeds the initial threshold value t, but remains below the anomaly threshold z_q. If this is the case, the series is no longer constant and the calculations in Equations (4.15) to (4.17) will not fail. Therefore, the problem only arises when calculating the very first threshold value. As a solution I only apply the threshold calculation when $\hat{\sigma}$ and $\hat{\gamma}$, otherwise the threshold will be set to $t + \epsilon$, where t is the initial threshold and ϵ is there to push the threshold slightly above the initial threshold. I set this to 10^{-5}.

4.1.3 Autoregressive Anomaly Detector

The autoregressive model is a simple, statistical, forecasting-based anomaly detection method. Autoregression is linear model to forecast the next value in a time series. It is based on the assumption that the consecutive values of a time series are strongly correlated, therefore the next value can be written as a linear combination of a certain amount of previous values, also called lags. Mathematically the model is written in the following way:

$$o^{(t)} = \sum_{i=1}^{l} \alpha_i \cdot o^{(t-i)} + d + \epsilon_t \tag{4.18}$$

where

$o^{(t)}$ = the next value of the time series,
l = the number of lagged values used for forecasting,
α_i = the coefficients for each lagged value,
d = a constant, the necessary delta to get the next value from the linear regression,
ϵ_t = white noise, represents the anomaly score.

The model is calibrated on the calibration portion of the time series to determine the best values for the α_i and d coefficients. The optimization is done by solving the corresponding linear equations with the least-squared regression. Least-squared regression tries to fit a regression line to the data points so that the variance, i.e. the sum of the squares of the distances/errors is minimal.

The number of lags l is a user set parameter, however a correlogram (a diagram that shows the correlation for at each lagged value) can be used to determine the optimal number of lags.

The autoregressive model requires a strongly correlated time series. Therefore it works the best on stationary time series.

Implementation

I implemented the autoregression model using the AutoReg model form the *statsmodels* Python library [37, 41]. I examined three implementation methods: two static and one dynamic.

- The AutoReg model itself offers support for data with trend and periodicity by using seasonal dummies. Separate coefficients are calibrated for the lagged periods. The coefficients are only added to the linear combination if the lag is multiple of the period. Using this I calibrate a static model on the raw data for forecasting.
- To make the input data stationary I perform the common and seasonal differencing on the data introduced in Definition 15. The AutoReg model is fitted to this without its built in trend or seasonality components.
- The model is quite lightweight, therefore re-calibration does not introduce a substantial detection speed decrease. I examine a dynamic variant which uses the l lagged values to predict the next l values, then re-calibrates the model. For the re-calibration the training subsequence of the time series is a sliding window, from which the first l values slide out, and the last l timepoints slide into.

The approach can be used for anomaly detection by using the coefficients calibrated during the training step and Equation (4.18) to get the next value. $\sum_{i=1}^{l} \alpha_i \cdot o^{(t-i)} + d$ represent the prediction made for the next value, while the error component compared to the observed value, ϵ_t represents the *anomaly score*. The *anomaly score* is compared to a threshold to determine whether the observation is anomalous.

Thresholding

In order to use the forecasting autoregressive model as an anomaly detector a detection threshold needs to be set. There are two ways to determine this:

- An unsupervised solution is to examine the distribution of the *anomaly score* data. The *Three Sigma Rule* is commonly used to determine the threshold $\delta = \mu + 3 * \sigma$ [17], where μ is the mean of the series and σ is the standard deviation of the series. To make the algorithm work on streaming time series in real-time, only the previous anomaly score values are in the calculation. While this leaves the threshold imprecise for the first handful of observations and fine-tunes it later, this is not expected to cause miss of anomalies due to the rarity of outliers.

- If anomaly labels are available for the calibration dataset, the *Best-F-score Method* can be also utilized for threshold setting. The aim of this approach is to find the threshold that corresponds to the best F-score in the calibration data.

  The F-score is a metric which describes the quality of the anomaly detection. It is the harmonic mean of precision (the ratio of datapoints correctly labeled as anomalies by the alogrithm to the number of all the datapoints marked as anomalies in the data by the algorithm) and the recall (the ratio of datapoints correctly labeled as anomalies by the alogrithm to the actual number of anomalies in the data). Further details on these metrics can be found in Section 5.1.2 with the difference being that if a metric is undefined, the metric is set to 1. The reasoning behind this is that if there are no anomalies in the calibration dataset and the algorithm correctly identifies this, the F-score should reflect that this is the desired behavior and therefore the corresponding threshold is to be chosen.

  The potential thresholds are all the anomaly scores for the calibration datapoints (minus a small delta to be able to detect the lowest error if necessary). If there are no anomalous datapoints in the calibration data, the threshold has to be bigger than any of the anomaly scores observed, therefore it is placed just above the maximum error.

4.1.4 Autoencoder

As described in Section 3.1.5, autoencoder can be used as a reconstruction-based versatile anomaly detector. The model is trained on normal (or mostly normal) data to learn the characteristics of the normal behavior(s). New data is used as a raw input to this model, which is subsequently reconstructed. If the reconstruction is close to the actual data, the data is likely normal, otherwise its anomalous.

Autoencoders are window-based anomaly detectors as the input layer of an autoencoder consists of multiple nodes to which data needs to be fed. Therefore a single datapoint is not enough as an input, a longer subsequence or window is used instead. The newly arriving data thus can only be fed to the model in windows as well. This introduces additional delays, however this is not substantial. The anomaly detection decision is applied to the whole window (all of its timesteps), further details about the exact anomalous timestep(s) are not given.

Implementation

For univariate experiments I used implemented the autoencoder model using the Keras Python Library [25] with the Adam optimizer and the mean-squared error (MSE) loss function. The reconstruction error is calculated in Equation (4.19). The error is compared to the threshold to determine, whether the timestamp – and the corresponding window of previous WS values – is anomalous or not.

$$\epsilon_t = \sqrt{\sum_{i=1}^{WS}(reconstructed_i - observed_i)^2} \tag{4.19}$$

where

ϵ_t = anomaly score for timestamp t,
WS = window size,
$observed$ = observed values in the $[t - WS, t]$ window,
$reconstructed$ = reconstructed (predicted) values in the $[t - WS, t]$ window.

In the multivariate case I used a similar implementation of Garg et al. [11] who had it available as part of their anomaly detection framework.

Thresholding

The same methods can be used for setting a threshold for an autoencoder-based anomaly detector, i.e. the *Three Sigma Rule* or the *Best-F-score Method*. However, in the latter case – since instead of distinct timesteps there are windows for which there is a uniform anomaly detection decision – an adjustment needs to be made in the metric calculation. This is due to the second part of Definition 16 which states that the set of anomalies form a very small part of the dataset. Therefore it is assumed that inside a window, even if there are anomalous datapoints, normal datapoints are the majority. To reflect this I use the F_β-score [29] instead of the F-score The F_β-score is described in Equation (4.20).

$$F_\beta = (1 + \beta^2) \cdot \frac{P \cdot R}{\beta^2 \cdot P + R} \tag{4.20}$$

where

P = precision,
R = recall,
β = adjustment coefficient $0 < \beta <= 1$.

In my implementation I set β to 0.05 in accordance with Malhotra et al. [29].

4.2 Multivariate Application

For the multivariate experiments conducted on multivariate data I use the anomaly detection framework of Garg et al. [11]. The structural diagram of the framework can be seen in Figure 4.1. The authors also define an anomaly diagnosis part of their framework, however, this is not part of this thesis, thus that part is omitted. The triple-layer rectangles represent a node which consists of multiple units of the same characteristics, e.g. multiple reconstruction models working on each channel simultaneously. The thick arrows represent multivariate data flow, while the thin arrows mark one-dimensional data.

Firstly the time series is pre-processed: the training part of the series is normalized while the testing part is clipped into the $[-4, 5]$ interval. The pre-processed channel data is fed

into the prediction or reconstruction model for each channel. The output of such a model is a reconstruction error for each timestep. Before aggregating the channel-wise errors into one anomaly score, scoring transformation may be performed. This is to provide a better, more representative anomaly score than the simple reconstruction error. The channel-wise anomaly scores are aggregated using an aggregation function to achieve a single anomaly score for the whole multivarite time series at every timestep. In this univariate series thresholding can be performed to determine which points are anomalous.

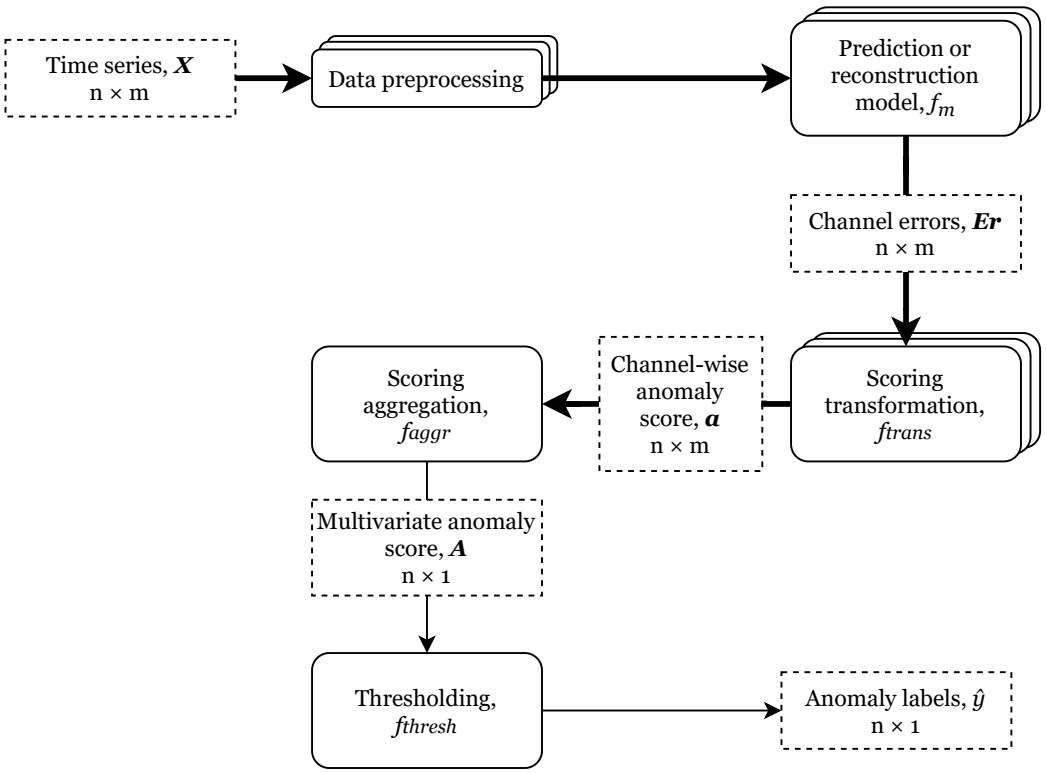


Figure 4.1: Multivariate anomaly detection framework

4.2.1 Aggregating Independent Channel Results

When running univariate anomaly detectors on each channel of a multivariate dataset, the channel-wise anomaly scores need to be aggregated into a common multivariate anomaly score. There exist multiple ways of achieving this.

Voting

A simple method of aggregating results is performing the whole anomaly detection process on each channel individually with the thresholding included. At the end of this process each timestamp – or depending on the algorithm subsequence – will be labeled anomalous or not in the case of a binary detector. Aggregating these labels can be done using a voting process, where the *voting threshold* determines how many channels need to flag a timestamp at one in order for it to be marked an anomaly regarding the multivariate time series. The value of the voting threshold may be fine-tuned with empirical testing or using expert knowledge, however three logical threshold values can be defined:

- **Single voting** In single voting, 1 anomalous channel is enough to trigger an anomaly alarm in the original multivariate time series. This can be effective if it is known that the system is not robust, a deviation in any of the observed channels may lead to failure.
- **Majority voting** In majority voting, an anomaly alarm is only triggered if the models mark an anomaly in more than half of the channels at that timestamp.
- **Unanimous voting** Unanimous voting means all of the models from every channel agree that there is an anomaly. This technique may be used to reduce the number of false alarms, however misses more subtle anomalies.

Scoring Functions (f_{trans} and f_{aggr})

The authors propose multiple aggregation functions which take the channel-wise anomaly scores (real numbers) as input, and aggregate them into a single multivariate anomaly score. Thresholding is done on this aggregate score afterwards.

- **Normalized Errors** The reconstruction error output of the channel-wise algorithms can be used directly as a multivariate score. To compensate for training error differences across channels, a mean training reconstruction error is subtracted from the anomaly score of each channel. To aggregate the scores into one single multivariate score the root-mean-square of the channel-wise scores is taken.

$$\mathbf{A}^{(t)} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\mathbf{Er}_i^{(t)})^2} \tag{4.21}$$

where

t = timestep,
i = channel index,
$\mathbf{A}^{(t)}$ = aggregated multivariate anomaly score at timestep t,
$\mathbf{Er}_i^{(t)}$ = channel-wise reconstruction error at timestep t in channel i,
m = number of channels in the dataset.

- **Gauss-Static (Gauss-S) Scoring** The distribution of anomaly scores approximates a normal distribution [42] with the small reconstruction errors being the most likely, while perfect reconstruction and anomalies have low probability. If we assume that the channels are independent as proposed by Ahmad et al. [2], the likelihood of a multivariate random variable can be calculated as the the product of the cumulative distribution function in each dimension, and the log-likelihood as the sum of the individual log likelihoods. After fitting the distributions, an anomaly score can be designed based on this.

  There are two functions that can be used to describe the distribution of a continuous random variable X. The *probability density function (PDF)* for each point provides the relative likelihood that that the value of the random variable will be close to the chosen sample, i.e. $\phi_X(x)dx = \mathbb{P}(x \in [x, x+dx])$ where ϕ_X is the PDF of X and $dx \to 0$.

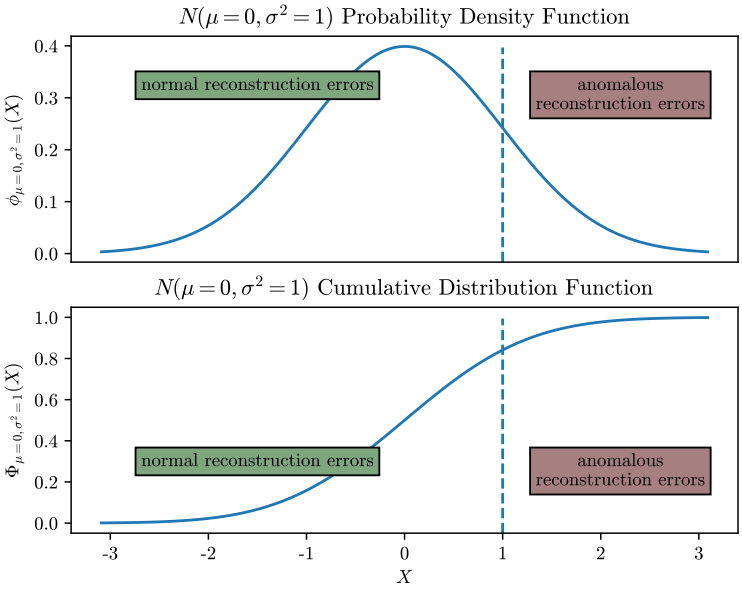


Figure 4.2: Normal distribution Probability Density Function and Cumulative Distribution Function

The *cumulative distribution function (CDF)* is a monotonically increasing function that gives the area under the PDF and describes the probability of $\mathbb{P}(X < x)$ It can be calculated from PDF in the following way: $\Phi_X(t) = \int_{t=-\infty}^{x} \phi_X(t)\, dt$ where X is a continuous random variable, ϕ_X is the PDF of X and Φ_X is the CDF of X.

In case of anomaly detection, the reconstruction errors that lie left to the center of the bell curve on the PDF are the small reconstruction errors with perfect reconstruction being the rarest, and those to the left are the big reconstruction errors representing potential anomalies. PDF is not a good candidate to represent anomaly scores since the rare normal and rare anomalous values would receive the same anomaly score. The CDF assigns a higher value to high reconstruction errors than to low reconstruction errors. This is visualized in Figure 4.2. The vertical line in the figure separates the normal and anomalous reconstruction errors, however, it as for illustrative purposes only as the line is fuzzy and is not set at this step, but is step during thresholding. The line is put higher than $X = 0$ because the reconstruction errors cannot take negative values. To take advantage of the negative values it can beneficial to perform a transformation which transforms the errors into the complete set of real numbers. Garg et al. define the Gauss-S scoring in the following way (Equation (4.22)):

$$\mathbf{a}_i^{(t)} = -log(1 - \Phi(\frac{\mathbf{Er}_i^{(t)} - \hat{\mu}_i}{\hat{\sigma}_i}) \quad ; \quad \mathbf{A}^{(t)} = \sum_{i=1}^{m} \mathbf{a}_i^{(t)} \tag{4.22}$$

where

t = timestep,
i = channel index,
$\mathbf{Er}_i^{(t)}$ = reconstruction error at timestep t in channel i,
$\mathbf{a}_i^{(t)}$ = channel-wise anomaly score at timestep t in channel i,
Φ = CDF of $N(0,1)$ normal distribution,
$\hat{\mu}_i$ = empirical mean of reconstruction errors in channel i,
$\hat{\sigma}_i$ = empirical standard deviation of reconstruction errors in channel i,
m = number of channels in the dataset,
$\mathbf{A}^{(t)}$ = aggregated multivariate anomaly score at timestep t

The authors apply an additional transformation (f_{trans}) to the reconstruction error before putting it into the CDF. $\frac{\mathbf{Er}_i^{(t)} - \hat{\mu}_i}{\hat{\sigma}_i}$ represents the magnitude of the error when compared to the standard deviation of the errors. This may be negative when the reconstruction error remains below the average of reconstruction error – indicating a strongly non-anomalous observation. The negation of the log-likelihood is necessary to assign higher scores to large errors and lower scores to small and negative errors.

- **Gauss-Dynamic (Gauss-D) Scoring** Gauss-S uses a static main and standard deviation, this does not reflect data drift. The Gauss Scoring function can be made data characteristics adaptive by replacing the mean and standard deviation with a dynamic pair, $(\hat{\mu}_i^{(t)}, \hat{\sigma}_i^{(t)})$. The pair can be calculated in the following way (Equation (4.23)):

$$\hat{\mu}_i^{(t)} = \frac{1}{W} \sum_{j=0}^{W-1} \mathbf{Er}_i^{(t-j)} \quad ; \quad \hat{\sigma}_i^{(t)} = \sqrt{\frac{1}{W-1} \sum_{j=0}^{W-1} (\mathbf{Er}_i^{(t-j)} - \hat{\mu}_i^{(t)})^2} \tag{4.23}$$

where

t = timestep,
i = channel index,
W = the window size,
$\mathbf{Er}_i^{(t)}$ = reconstruction error at timestep t in channel i,
$\hat{\mu}_i^{(t)}$ = the mean of reconstruction errors inside the window,
$\hat{\sigma}_i^{(t)}$ = the standard deviation of reconstruction errors inside the window.

The channel-wise and multivariate anomaly scores are calculated as in Equation (4.22)

The windows for every timestamp are the value at the timestamp and the previous $W-1$ timestamps. For the first W timestamps in the testing time series the previous timestamps are from the end of the training time series.

- **Smoothing with Convolution (Gauss-D-K)** Garg et al. and [2] both propose a convolution based scoring function which smooths out the scores before aggregation which potentially amplifies the multivariate anomaly score when different channels respond at slightly different times to the same anomaly. The channel-wise scores are calculated in the following way:

$$G(x, \sigma_k) = e^{-\frac{x^2}{2 \cdot (\sigma_k)^2}} \quad ; \quad \mathbf{a}_i^{(t)} = G * \mathbf{a}_{i,Gauss-D}^{(t)} \tag{4.24}$$

where

$G(u, \sigma_k)$ = the Gaussian-filter with kernel sigma σ_k,
$*$ = convolution operator,
t = timestep,
i = channel index,
$\mathbf{a}_{i,Gauss-D}^{(t)}$ = the channel-wise anomaly score from Gauss-D,
$\mathbf{a}_i^{(t)}$ = channel-wise anomaly score for timestamp t in channel i.

To calculate the multivariate anomaly score $\mathbf{A}_i^{(t)}$ the same sum as in Equation (4.22) can be used. The challenge with this method is choosing the value for the kernel sigma (σ_k) value which is not straightforward. The authors did this by empirically choosing the value yielding to the best results. However this may be impossible or may require prior knowledge on datasets with similar characteristics.

4.2.2 Thresholding (f_{thresh})

Once a one-dimensional aggregated anomaly score is available it needs to be thresholded to determine, whether the algorithm flags it as an anomaly or it is marked as a normal value. There are multiple ways of doing this.

- **Top-k** The aim of the top-k thresholding is to choose exactly k timepoints that are to be labeled as anomalous. These are the k most – hence the name – anomalous timesteps. A drawback of the approach is that one needs to know the value of k that is usually set to the actual number of anomalies in the dataset which only makes it suitable for benchmarking. Furthermore, this metric can only be used in offline detection since only then is it known which are the timepoints with the most extreme anomaly score.
- **Best-F-score** In the same way as in the univariate case, if the training portion of the dataset is labeled, a good threshold can be chosen by setting it so that the best F-score is achieved during training.
- **Tail-probability** When the anomaly scores correspond to probabilities (like with Gauss-S, Gauss-D and Gauss-D-K) the tail-p thresholding can be used, which labels the timepoints as anomalous if their probability is below a pre-defined tail probability. Since the scoring functions give the anomaly score as a sum of negative log probabilities the threshold is to be defined in a similar way: $th_{tail-p} = -m \cdot log_{10}(\epsilon)$ where m stands for the number of aggregated channels (and for 1 for datasets where only one channel is used for thresholding). The anomaly score is above the threshold when $x_1, ...x_m$ stand for the channel errors after f_{trans} and

$$\begin{aligned} -log_{10}(x_1) - log_{10}(x_2) - ... - log_{10}(x_m) &> -m \cdot log_{10}(\epsilon) \\ log_{10}(x_1) + log_{10}(x_2) + ... + log_{10}(x_m) &< m \cdot log_{10}(\epsilon) \\ 10^{log_{10}(x_1)+log_{10}(x_2)+...+log_{10}(x_m)} &< 10^{m \cdot log_{10}(\epsilon)} \\ 10^{log_{10}(x_1)} \cdot 10^{log_{10}(x_2)} \cdot .. \cdot 10^{log_{10}(x_m)} &< (10^{log_{10}(\epsilon)})^m \\ x_1 \cdot x_2 \cdot .. \cdot x_m &< \epsilon^m. \end{aligned}$$

The authors compare different values for the threshold ($-log_{10}(\epsilon) \in \{1, 2, 3, 4, 5\}$) to determine which one yields the best F-score.

The tail-probability thresholding can be automatized by using an algorithm based on Extreme Value Theory, like SPOT, if a dynamic threshold is desired, DSPOT, and running it on the aggregated anomaly scores.

4.2.3 Multivariate Anomaly Score for POT-based Approaches

The framework uses forecasting-based and reconstruction-based anomaly detectors, where this channel-wise anomaly score can be easily calculated by the difference in magnitude of the observed value and predicted or reconstructed value. However, my goal is to test the performance of SPOT and FluxEV, both of which rely on the adaptive threshold anomaly detector, Peaks-over Threshold (POT). Therefore, these algorithms need to be adapted to function in this framework.

In contrast to forecasting or reconstruction detectors which bring the added value of anomaly detection by an estimate of the next value, POT does not change the input values in any way, the added value comes from the threshold itself. Thus the relationship between the raw value and the threshold needs to be kept.

The idea is taking the difference of the raw value at timestep t and the threshold at timestep t, δ_t, i.e. the reconstruction error. The goal is to achieve a score that has the same characteristics as the scoring of other anomaly detection methods.

The characteristics of the anomaly score of forecasting- and reconstruction-based models are:

- A bigger score reflects a bigger deviation from the actual value, yielding to a possible anomaly.
- Similarly a lower score reflects a small deviation from the actual value, which means the forecasting/reconstruction was successful, therefore the value is probably normal.
- The used magnitude usually can only be positive or 0, which is achieved by taking the absolute value of the distance or using a magnitude function with such characteristics (e.g.: squared distances). This is due to that great deviation in either direction is considered a potential anomaly, therefore negative scores are just as anomalous as positive ones, 0 being the perfect reconstruction.

If we examine the above suggested conversion scheme for adaptive threshold-based methods along the same characteristics, we find that:

- A bigger score reflects a bigger deviation from the threshold in the positive direction. The further a value is from the threshold the more likely it is that it is an actual anomaly and not a false detection, therefore this is a desired outcome.
- A lower score reflects that the deviation from the threshold is small or a value is below the threshold. The algorithm would mark these as negative or only with a small margin positive, so this reflects the functioning of the original algorithm as well.
- Scores may be negative. This happens when a value is below the threshold. Since for these algorithms deviation is only considered anomalous in one direction, and the goal is to assign low scores to data points that are considered normal, this is also desirable. Here the score 0 is not the certainly normal data but the data that is on the edge of the normal and anomalous behavior. Negative values help offset this imbalance.

Note that in the arguments above I assumed an adaptive threshold-based detector with only an upper threshold. The detector may have a lower threshold, like a version of SPOT (BiSPOT) does. In this case, the lower threshold has to be taken into consideration as well, and the threshold that lies closer in absolute distance needs to be chosen for differencing. This is calculated as in Equation (4.25) where the max function ensures that the larger number is taken, i.e. the negative number closer to zero – corresponding to the smaller distance from the two threshold –, or the positive one in which case the the distance from the other threshold would be negative indicating a normal value instead of an anomalous one. The error calculation process is visualized in Figure 4.3. The upper and lower thresholds are marked with dashed lines, timestamps with values outside these are considered anomalous (marked in red in the figure). For the error calculation the closer threshold is chosen. Positive errors correspond to anomalous values, negative errors correspond to normal values.

$$\mathbf{Er}^{(t)}_{BiSPOT} = max(o^{(t)} - \delta^{(t)}_{up}, \quad \delta^{(t)}_{down} - o^{(t)}) \tag{4.25}$$

where

$\mathbf{Er}^{(t)}_{BiSPOT}$ = the error value of BiSPOT for multivariate use-case at timestamp t,
$o^{(t)}$ = raw data value of the time series at timestamp t,
$\delta^{(t)}_{up}$ = upper threshold of BiSPOT at timestamp t,
$\delta^{(t)}_{down}$ = lower threshold of BiSPOT at timestamp t.

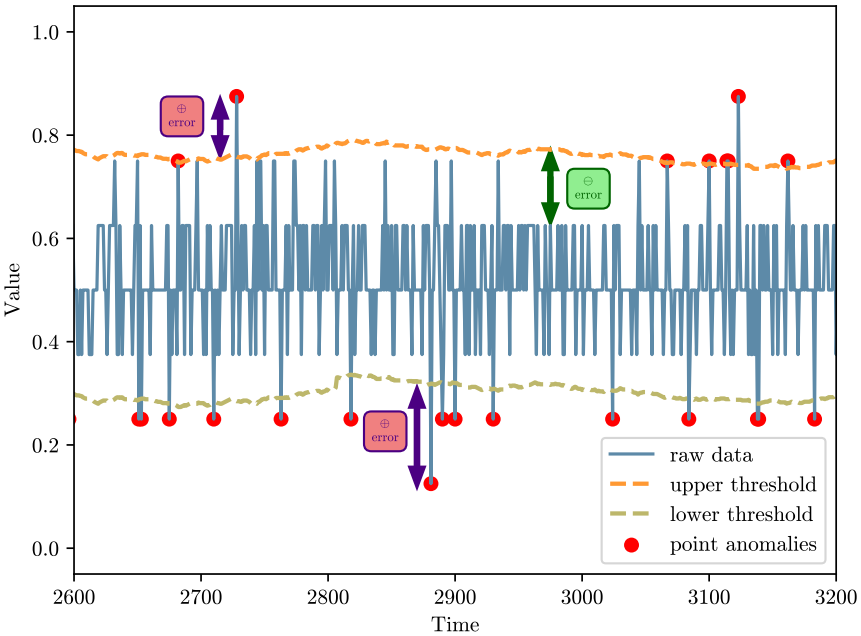


Figure 4.3: BiDSPOT multivariate error calculation on a portion of one channel of a multivariate time series.

Scoring Aggregation

As channel-wise anomaly score the threshold-based differencing has merit. However, the distinct channel-wise scores need to be aggregated, and – depending on the aggregation function – this method may not be a suitable solution.

The *Normalized errors* aggregation is not suitable for aggregating potentially negative channel-wise errors. This is due to the fact that the root-mean-square calculation method takes the square of each channel-wise error which hides whether the error is negative or positive: a strongly normal value with a negative score will increase the aggregate anomaly score instead of decreasing it.

Using the *Gauss-scoring* aggregation functions however, no such obstacle arises. Using the raw error values as input, the CDF of the Normal distribution (Figure 4.2) will assign lower values to negative errors resulting in a lower overall multivariate anomaly score. The CDF may be treated as a smoothing of the threshold calculated by POT-based methods, since timestamps that have been deemed anomalous by only a small margin may be reconsidered after aggregation. By adding the transformation described in Equations (4.22) and (4.23) the same behavior can be achieved as with forecasting- or reconstruction-based detectors. Therefore, in the evaluation I will only report results achieved by utilizing the *Gauss-D* and *Gauss-D-K* scoring methods.

Chapter 5

Experiments

5.1 Anomaly Detection Evaluation Methods and Metrics

To evaluate the performance of anomaly detection algorithms, various metrics exist which describe the accuracy of the detection. Popular metrics in contemporary works are *Precision* and *Recall* or the combination of the two, the F-score or F_1-score metric. To compare algorithms in a way that their threshold is flexible, the Receiver Operating Characteristics (ROC) curve can be used.

5.1.1 Confusion Matrix

Table 5.1: Confusion matrix.

		Predicted Class	
		PP	*PN*
Actual Class	*P*	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

All of the aforementioned metrics rely on four numerical metrics which are often organized into a confusion matrix which can be seen in Table 5.1. The matrix describes the relationship between the prediction labels made by the algorithm (*Predicted Positive (PP)* and *Predicted Negative (PN)*) and the actual labels that each timestep belongs to given by the Ground Truth in anomaly benchmark datasets (*Positive (P)* and *Negative (N)*). The sum of $P + N = PP + PN = TP + FP + FN + TN$ gives the whole population, i.e. every timepoint in the surveyed dataset.

Point-wise Evaluation

To fill the confusion matrix the naive approach is the point-wise evaluation in which for every single timestep a decision is made:

- If the predicted label is positive and the actual label is also positive (in case of anomaly detection an anomaly flag has been raised where there is an actual anomaly), the number of True Positives is increased by one.
- If the predicted label is positive but the actual label is negative (in case of anomaly detection an anomaly flag has been raised where there is no anomaly label in the ground truth), the number of False Positives is increased by one.
- If the predicted label is negative but the actual label is positive (in case of anomaly detection an anomaly has been missed by the algorithm), the number of False Negatives is increased by one.
- If the predicted label is negative and the actual label is also negative (in case of anomaly detection the algorithm correctly identifies normal behavior), the number of True Negatives is increased by one.

5.1.2 Anomaly Detection Metrics

While the four-tuple of (TP, FP, FN, TN) has all the information about the anomaly detection, it is cumbersome to compare different algorithms or different parameter settings of the same algorithm along four distinct values. Therefore, metrics combining these values are used. Commonly used metrics are defined in Equations (5.1) to (5.4).

$$Precision = \frac{TP}{PP} = \frac{TP}{TP + FP}; \tag{5.1}$$

$$Recall = True\ Positive\ Rate\ (TPR) = \frac{TP}{P} = \frac{TP}{TP + FN}; \tag{5.2}$$

$$Specificity = True\ Negative\ Rate\ (TNR) = \frac{TN}{N} = \frac{TN}{TN + FP}; \tag{5.3}$$

$$F\text{-}score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \tag{5.4}$$

The *Precision* reports what ratio of the predicted positives by the algorithm was actually positive. *Recall* also referred to as *True Positive Rate* or *Sensitivity* describes how well the algorithm managed to recall the positive timestamps, i.e. what ratio of them did it detect correctly. The *Specificity*, *Selectivity* or *True Negative Rate* measures the same thing but for the negative timestamps. The *F-score* is the harmonic-mean of the *Precision* and *Recall* metrics.

Should any of the metrics be undefined – due the the denominator being 0 – the whole metric shall be set to 0 in the evaluation.

5.1.3 Windowed Evaluation

The problem with the naive evaluation approach is that only in an ideal case can an anomaly detection raise an anomaly flag at the exact timestamp for a point outlier as in the Ground Truth. In reality, some delay is to be expected and is acceptable for taking counteractions as well. Therefore, in the evaluation I take a windowed evaluation approach.

The main idea of this is creating anomaly windows around the anomalies in the Ground Truth so that the edges of the window extend to $\frac{AWS-1}{2}$ timesteps to both sides of the

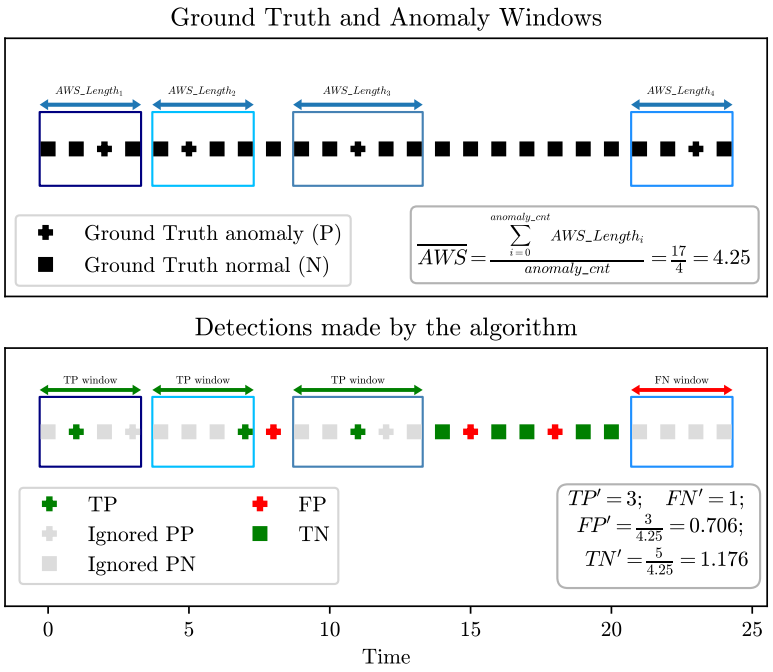


Figure 5.1: An example of the Windows Evaluation on a short sample time series

anomaly, where AWS is the Anomaly Window Size. If there is a positive detection anywhere inside the window, the detection is assigned to that anomaly, and the number of True Positives increases by only one for such a window, further timestamps are disregarded. Should there be no detection made on any timestamp of the window, the False Negatives metric is increased. If two anomalies are so close to each other that their windows would overlap or the window would stretch outside of the time series, the timestamps are split equally between the two windows and the invalid indexes in the window are dropped.

Outside the windows the evaluation is made on a per-timestamp basis (also called point-wise evaluation) like in the naive approach. However, this way the True Negatives and False Positives will be over-represented, since for True Positives and False Negatives instead of AWS increases for one window, only 1 increase is made.

To combat this and correct the proportions, a normalization of True Negatives (TN') and False Positives (FP') with the average window length is performed (Equation (5.5)). The idea behind this is that the True Negative and False Positive detections would on average fill TN' and FP' anomaly windows respectively. TP' and FN' is calculated in the above described way.

An illustrative example of the process is shown in Figure 5.1 on a time series of length 20 and AWS set to 5. The upper plot shows the labels in the Ground Truth (P and N) with the anomaly windows around the anomalous timestamps. The first and second window needs to be adjusted since they would overlap, and the last one because it would stretch beyond the end of the time series. The lower plot shows the detections made by the anomaly detection algorithm PP and PN. Green color shows correct classification, while red describes incorrectly classified timestamps. Inside the windows only the first PP is taken into consideration, the other values are disregarded shown in gray. The last window only has PN detections, therefore it is marked as a False Negative detection.

The calculation method and calculated values for this exact scenario is also shown for the $\overline{AWS}$ and the 4-tuple (TP', FP', FN', TN') metrics.

In following equations and evaluation reports the metrics from Equations (5.1) to (5.4) and (5.7) are calculated using the normalized 4-tuple. The approach is equivalent with marking all timestamps inside True Positive windows as TP and the False Negative windows as FN and performing the point-wise evaluation on the remaining values.

$$TN' = \frac{TN}{\overline{AWS}}; \quad FP' = \frac{FP'}{\overline{AWS}}. \tag{5.5}$$

where

TN' = normalized True Negatives,
FP' = normalized False Positives,
$\overline{AWS}$ = average anomaly window size.

5.1.4 Composite F-score (F_c-*score*)

Garg et al. propose in [11] a novel anomaly metric called the Composite F-score or F_c-score calculated as in Equation (5.6). They argue that in comparison with the point-wise F-score – which requires correct detection for each timestamp for a high score – and the event-wise F-score proposed by Hundman et al. [18] – that increases true positives with only one positive detection inside an anomaly window, but biases longer sequence length for a better score – the Composite F-score serves as a balance between the two metrics. However, the Point-wise Precision still needs to have exact detection for a high value, no delay is allowed. Furthermore, all but one value that for the event-wise recall belong to the anomaly window need to be detected as negatives to positively influence the Precision in the case of point outliers. I argue that the values inside the anomaly windows should not be considered when counting False Positives and True Negatives like the Windowed Evaluation approach does so.

In order to provide values to compare with the study of Garg et al. I will report the Point-wise Precision and Composite F-score metrics for the multivariate scenarios in Section 5.3.3. Charts will use the metrics of the Windowed Evaluation method unless stated otherwise.

$$F_c = 2 \cdot \frac{Pr_t \cdot Rec_e}{Pr_t + Rec_e} \tag{5.6}$$

where

F_c = Composite F-score,
Pr_t = Point-wise Precision calculated as in Equation (5.1),
Rec_e = Event-wise Recall calculated as in Section 5.1.3.

5.1.5 Receiver Operating Characteristics Metric

Another popular metric to evaluate and compare different parameter settings of the same algorithm or distinct algorithms is the Receiver Operating Characteristics curve [7, 44].

The curve describes the relationship between the TPR (Equation (5.2)) and the FPR (Equation (5.7)) of an algorithm. The FPR is the ratio of the incorrectly classified negative timestamps to all the negatives. Each point on the curve corresponds to a different decision threshold.

$$\textit{False Positive Rate (FPR)} = 1 - TNR = \frac{FP}{N} = \frac{FP}{FP + TN} \tag{5.7}$$

For binary classification approaches that use the probability distribution of a continuous random variable X for scoring the a continuous curve can be drawn by plotting the CDF of the false-alarm probability along the x-axis, and the CDF of the detection probability along the y-axis.

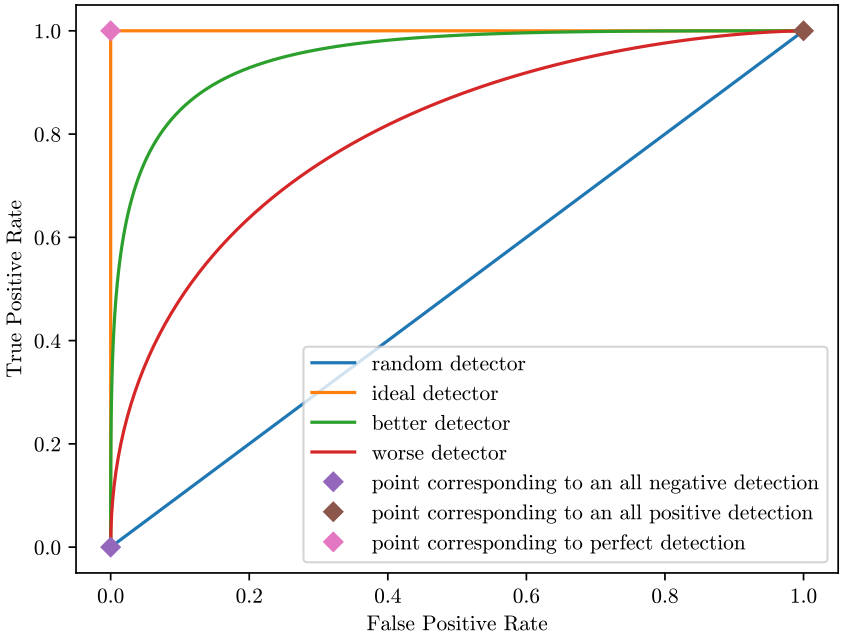


Figure 5.2: Characteristics of the Receiver Operation Characteristics curves

Another approach that can be applied for algorithms with no continuous random variable scoring is a sampling approach by evaluating the performance of the algorithm at distinct thresholds. One specific run only results in one point on the curve (just one (TPR, FPR) pair). To populate the curve with additional points, the detection threshold needs to be moved higher and lower. The goal is to move it so low, that there are only predicted positive detections – in this case $TP = P$ therefore $TPR = 1$ and $FP = N$ so $FPR = 1$, the point on the curve that belongs to this threshold is $(1, 1)$. The other extreme is to move the threshold so high that all the detection are predicted negative – both TP and FP will be 0, therefore TPR and FPR will both take on 0 and the point on the curve belonging to this threshold is $(0, 0)$. Coordinates for points between the the two extremes can be calculated by moving the threshold. This can be done with linear sampling – that is at equidistant increases – however, using empirical testing, it may be beneficial to increase the sampling density around the point closest to the ideal detector. The performance

between the points on a curve generated this way can be interpolated simply be a straight line between the discrete points to form a broken line. Alternatively smoothing techniques may be used by fitting a curve to the points, however, in this case the points of the fitted curve may differ slightly from the original points [30].

The ideal detector has perfect recall $TPR = 1$ and perfect specificity $TNR = 1$, therefore the point belonging to this on the ROC curve is $(0, 1)$. The more accurate detectors will approach this point, i.e. their corresponding ROC curve will bulge further onward to the upper-left [30]. A detector the accuracy of which is down to chance will approach the 45° line where $TPR = FPR$. Any curve that is lower than this line statistically perform worse than the random detector. However, the output of such a detector can simply be flipped to get a better than random detector. A visual representation of the characteristics of ROC curves can be found in Figure 5.2

When comparing multiple algorithms, in order to draw conclusions about the dominance of one algorithm over the other, it is important that the curve of the dominant algorithm be always above the curve of the approach with poorer performance. Otherwise no algorithm dominates over the other. When using the ROC curve to examine the performance of a single detection approach non-convex behavior, i.e. non-monotonously increasing curves may be an indicator that the algorithm needs further tuning [30].

While the ROC curve incorporates a large amount of information about detection performance in a visual form, it is usually quantified into a single real number that is derived from the area under the ROC curve. This is commonly referred to as the Area under ROC (AUROC) or simply Area under Curve (AUC). A higher AUC value generally indicates higher detection performance. However, a higher value does not necessarily indicate that one algorithm clearly dominates over another.

5.2 Evaluation on Univariate Datasets

5.2.1 Datasets used for Comparison

To evaluate the approaches in a univariate scenario I opted for the Yahoo! Webscope datasets, available at [48] tailored for anomaly detection benchmarking. The datasets are labeled. The benchmark consists of four distinct categories.

- **A1 Benchmark** contains 67 different time series of 1420 timestamps each on average. The data is of real Yahoo! services with personal identifiers stripped from the data. The timestamps are observed on an hourly basis. Anomalies are marked by humans which may lead to inconsistency. The anomaly rate of the time series is 1.9%.
- **A2 Benchmark** consists of 100 synthetic time series with random seasonality, trend and noise. The average length of the series is 1421 timestamps. Point anomalies have been inserted randomly with an average anomaly rate of 0.3%.
- **A3 Benchmark** has a 100 synthetic time series of 1680 timestamps. The datasets have three pre-specified seasonalities and varying pre-defined noise and trends. Point anomalies are inserted randomly, yielding to an anomaly rate of about 0.3%.
- **A4 Benchmark** has the same characteristics as the A3 benchmark, however, also has changepoints in addition to point outliers. In this evaluation I only focus on the anomalies and disregard pattern changes.

The anomaly detection approaches are run separately on each dataset in each benchmark as the distinct datasets have differing data distributions. However, the anomaly metrics are reported on a per-benchmark basis.

The Yahoo! Webscope datasets only contain network service utilization or synthetic data which is good for benchmarking, however does not necessarily reflect real data behavior. To test on real-world data, I choose the *Real Known Cause* category of the Numenta Anomaly Benchmark (NAB) [27] datasets. The cause of the anomalies in these datasets are documented, therefore, the exact timestamps are known [31]. This category of the benchmark consists of 7 datasets including industrial data:

- the ambient temperature in an office setting (**benchmark name**: *ambient_temperature_system_failure*),
- Amazon Web Services (AWS) average CPU usage across a given cluster (**benchmark name**: *cpu_utilization_asg_misconfiguration*),
- CPU usage from a server in Amazon's East Coast datacenter ending in a complete system failure of the AWS API servers (**benchmark name**: *ec2_request_latency_system_failure*),
- temperature sensor data of an internal component of a large industrial machine with three anomalies: planned shutdown, catastrophic failure and precursor of catastrophic failure (**benchmark name**: *machine_temperature_system_failure*),
- number of New York City passengers aggregating the total number of passengers into 30 minute buckets (**benchmark name**: *nyc_taxi*),
- timing of key holds of several users of a computer with user changes as changepoints (**benchmark name**: *rogue_agent_key_hold*) and
- timing of key strokes of several users of a computer with user changes as changepoints (**benchmark name**: *rogue_agent_key_updown*).

The anomaly detection metric results are aggregated for all datasets, however individual detection results are available in Table A.1.

5.2.2 Evaluation Preliminaries

When setting up the algorithms for anomaly detection it is important to keep in mind the real time restrictions and requirements. In streaming data, the length of the series is not known upfront and fast startup is desirable. Adding to this the fact that in case of the Yahoo! Webscope datasets the length of individual time series is around 1500, a calibration size of only 100 values was chosen.

SPOT

For SPOT, Drift SPOT was chosen with a sliding window size (d) of 32. The data is thresholded both in the up and down directions (BiDSPOT). Three separate values were tested for the value of risk parameter (q): 10^{-1}, 10^{-2}, 10^{-3}.

FluxEV

FluxEV has a sliding window size used for the EWMA calculation (s), I set this to 10. For the *Second-step (periodic) smoothing* it uses half of this value to account for potential data drift. The smoothing step is allowed to use $p = 3$ periods in the calibration step and the Peaks-over-Threshold component calibrates on $k = 80$ timestamps. For the periodic smoothing the period length is set as the first pre-specified period in the Yahoo! Webscope data. All other datasets are processed with First-step smoothing only. I examined the same risk parameter values ($q \in \{10^{-1}, 10^{-2}, 10^{-3}\}$) as in the case of SPOT to determine the sensitivity of algorithm performance to this parameter.

Autoregressive Models

For the Autoregressive model, I performed tests using the three possible implementations introduced in Section 4.1.3, namely:

- A variant which runs on raw data and uses the AutoReg library from *statsmodels* [41] to perform the differencing. Three separate models are built when the three seasonalities are provided by Yahoo!, otherwise just one. When calculating the threshold for the Best-F-score thresholding method, the model with the maximum anomaly score is chosen to have a wider variety of possible thresholds. Conversely, when performing the anomaly detection the model with the minimum anomaly score is chosen to minimize False Positive detections.
- I perform common (one-distance) differencing (Definition 15) and seasonal differencing using the first seasonality in Yahoo! datasets if available. I consequently run the algorithm with seasonal and trend components disabled.
- The third version is a dynamic variant where the model is only to predict a window size (set to 32) number of data. After each window, the model is re-calibrated with the new observations taken into consideration. The model uses the first seasonal component in Yahoo! datasets.

The lags value, i.e. the number of previous observations taken into consideration for the prediction was set to 32 for all three variants.

For thresholding, both the Best-F-score and Three Sigma Rule methods were evaluated.

Autoencoder

The autoencoder model needs a portion of the data to serve as validation data. The model uses these values during the training to fine-tune its hyperparameters and yield to the best possible model. Potentially a better model can be built if there is a substantial amount of data is available for training and validation. However, given the real-time streaming scenario and the limited size of benchmark datasets, the validation ratio was set to 0.4 of the initial data. Therefore the length of the training data for the autoencoder was reduced to 60 timestamps and 40 timestamps were used for validation. The window size and consequently the input and output and batch size of the model is 32. The algorithm has 2 hidden layers with half the size of the input each and trains for 50 epochs following the settings of Braei and Wagner [7]. All other parameters are the default values of the Keras library [25].

Like in the case of the autoregressive models, both the Best-F-score-based and Three Sigma Rule-based thresholding approaches were evaluated.

5.2.3 Experiment Results

Evaluation of Anomaly Metrics

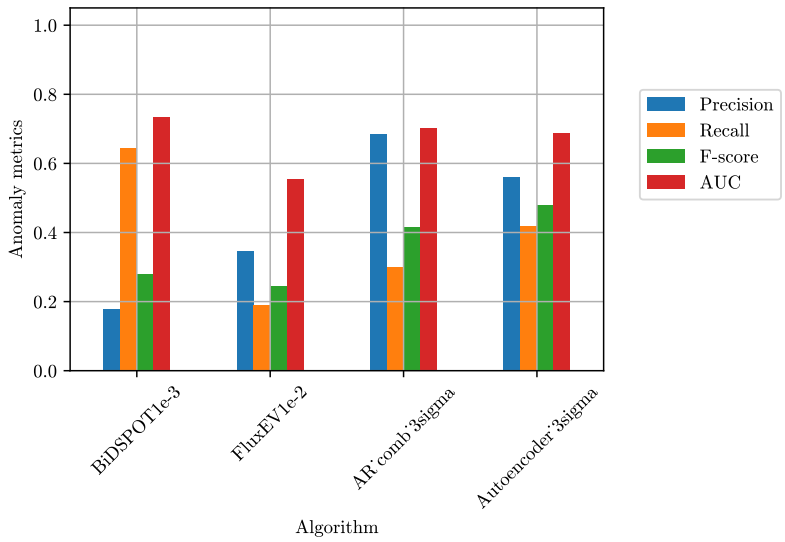


Figure 5.3: Precision, Recall, F-score and AUC metrics for the best performing algorithm from each category on A1 Benchmark

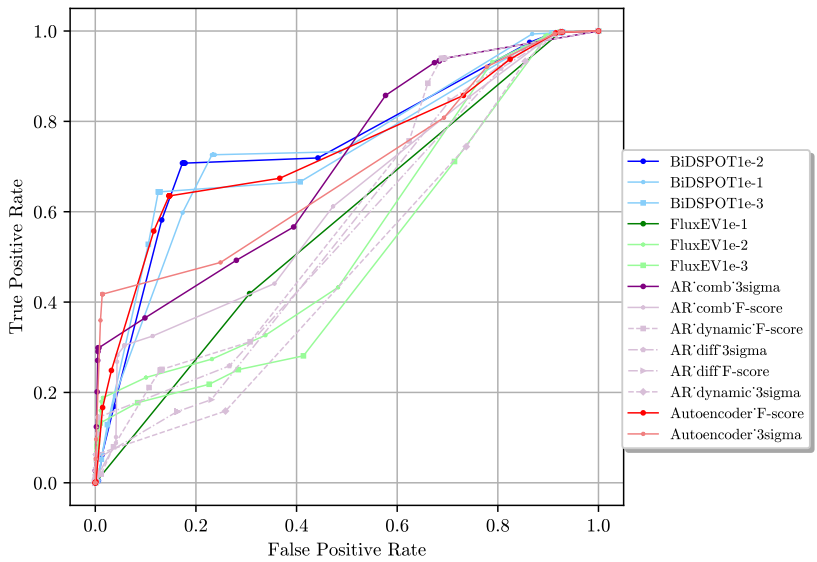


Figure 5.4: ROC curves for all tested algorithms on A1 Benchmark

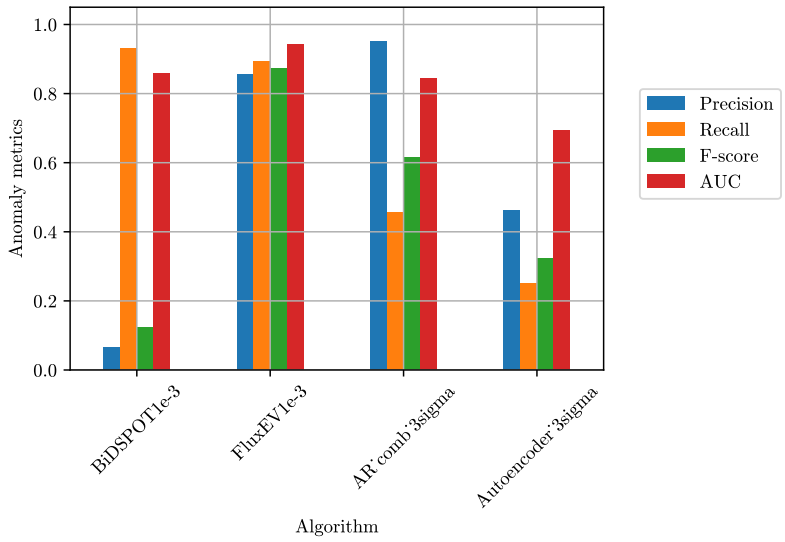


Figure 5.5: Precision, Recall, F-score and AUC metrics for the best performing algorithm from each category on A2 Benchmark

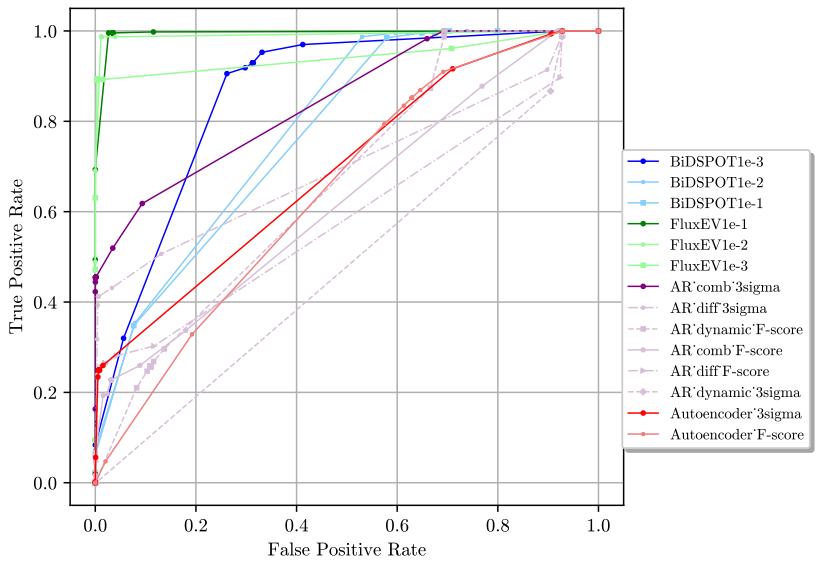


Figure 5.6: ROC curves for all tested algorithms on A2 Benchmark

For all the benchmarks I report the evaluation metrics of Precision, Recall, F-score and AUC for each algorithm set up with the aforementioned parameters. These metrics are detailed in Tables 5.2 to 5.6 where the bold values from each column mark the best detection performance in terms of that metric for the corresponding algorithm. I report these metrics for the best performing algorithm – the parameter setting and thresholding method that corresponds to the highest F-score – on bar charts for easier visualization in Figures 5.3, 5.5, 5.7, 5.9 and 5.11.

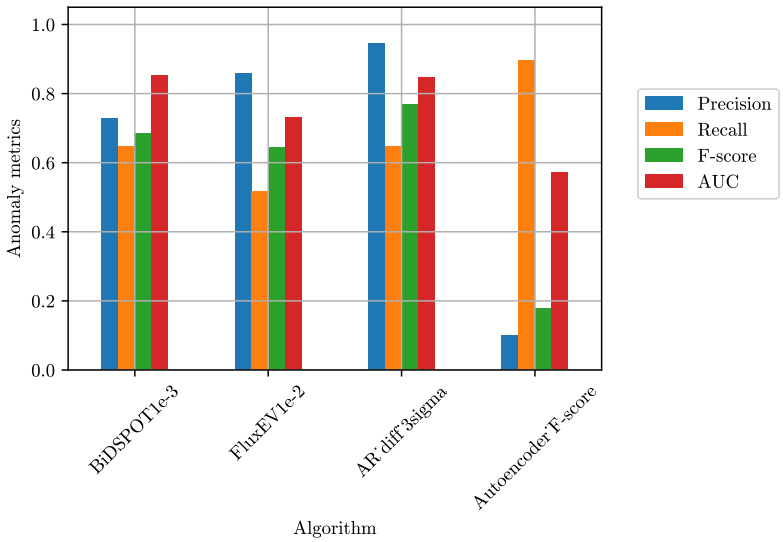


Figure 5.7: Precision, Recall, F-score and AUC metrics for the best performing algorithm from each category on A3 Benchmark

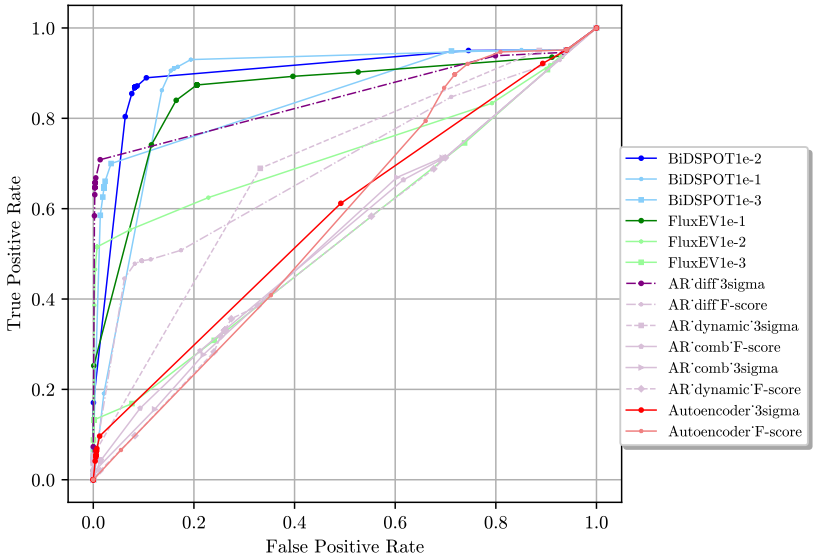


Figure 5.8: ROC curves for all tested algorithms on A3 Benchmark

Additionally, I plot the ROC curve for each tested algorithm (Figures 5.4, 5.6, 5.8, 5.10 and 5.12). Each category is marked with a base color. The ROC curve with the highest AUC value from each category is drawn with a darker color, while the poorer performing algorithms from the same category are shown in a lighter shade. The different parameter settings are differentiated using distinct markers on the curves, and in the case of Autoregressive models, with different line styles.

In the case of the A1 Benchmark, BiDSPOT with $q = 10^{-2}$ parameter setting achieved the best results in terms of AUC, and the Autoencoder with Three Sigma thresholding

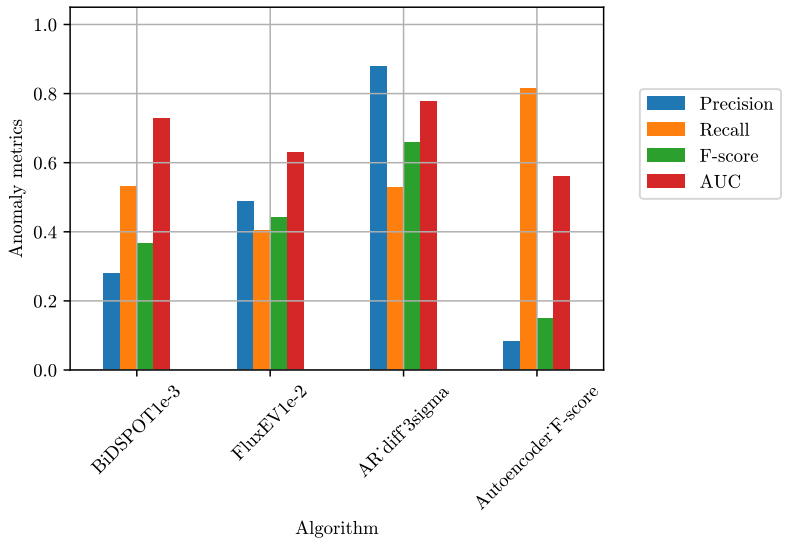


Figure 5.9: Precision, Recall, F-score and AUC metrics for the best performing algorithm from each category on A4 Benchmark

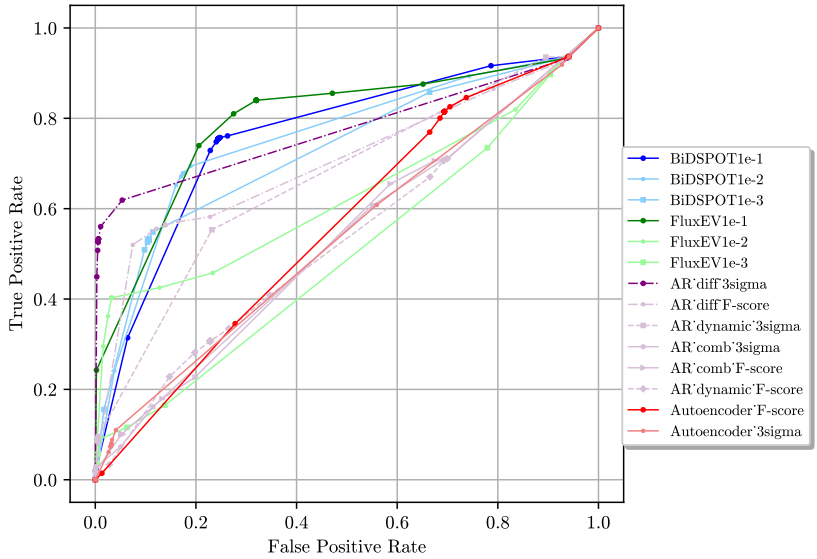


Figure 5.10: ROC curves for all tested algorithms on A4 Benchmark

concerning the F-score metric. The other parameter setting of BiDSPOT do not lie far behind with only a 1.5% performance drop on average.

On the A2 Benchmark FluxEV with all settings, and on the A3 Benchmark BiDSPOT both with $q = 10^{-1}$ and $q = 10^{-2}$ approached the perfect detector closely. FluxEV achieves an exceptionally high F-score as well on A2. On A3 the Autoregressive algorithm with differenced input data and Three Sigma Rule thresholding triumphs in terms of F-score.

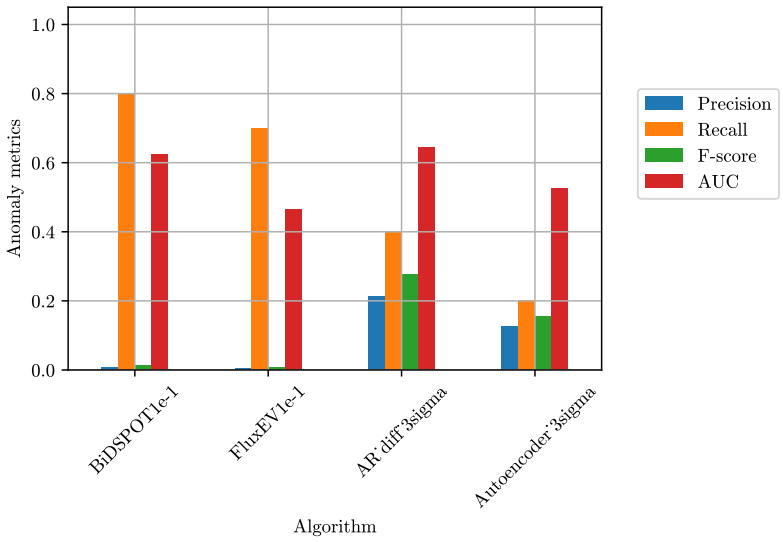


Figure 5.11: Precision, Recall, F-score and AUC metrics for the best performing algorithm from each category on NAB datasets

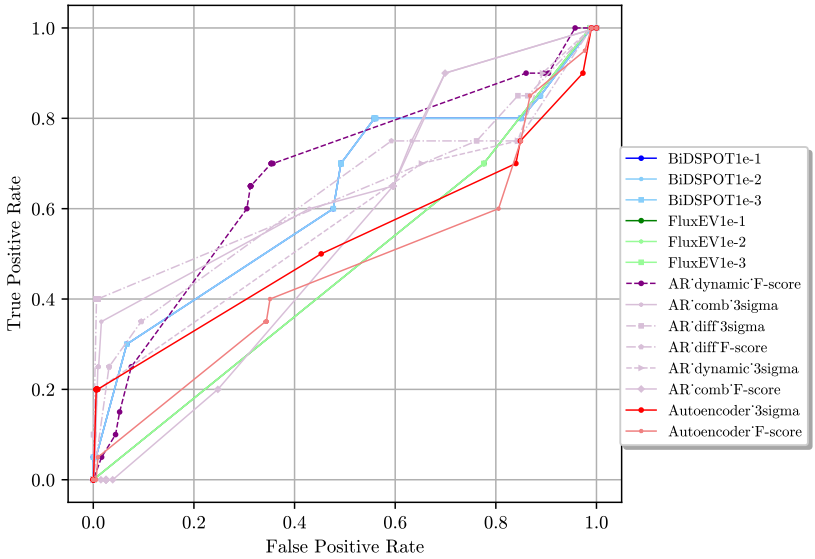


Figure 5.12: ROC curves for all tested algorithms on NAB datasets

Looking at the A4 Benchmark the FluxEV takes the lead again in AUC with $q = 10^{-1}$, however only slightly, as the differenced Autoregressive algorithm with Three Sigma thresholding only gives a 2.2% lower value however achives a higher TPR while keeping the FPR close to 0.

On real word data tests using the NAB datasets all algorithms present a lower performance regarding all anomaly metrics. Recall is high for both FluxEV and BiDSPOT but Precision is remarkably low. The most likely reason for this is that anomalies are hidden inside the

Table 5.2: Evaluation metrics on A1 Benchmark

Algorithm	Precision	Recall	F-score	AUC
AR_comb_3sigma	**0.683**	0.299	0.416	0.702
AR_comb_F-score	0.182	0.304	0.228	0.62
AR_diff_3sigma	0.524	0.144	0.226	0.559
AR_diff_F-score	0.039	0.158	0.063	0.552
AR_dynamic_3sigma	0.485	0.062	0.11	0.486
AR_dynamic_F-score	0.075	0.25	0.116	0.593
Autoencoder_3sigma	0.56	0.418	**0.479**	0.687
Autoencoder_F-score	0.155	0.635	0.249	0.736
FluxEV1e-1	0.055	0.419	0.097	0.577
FluxEV1e-2	0.345	0.188	0.243	0.555
FluxEV1e-3	0.362	0.132	0.194	0.496
SPOT1e-1	0.116	**0.726**	0.201	0.742
SPOT1e-2	0.146	0.708	0.242	**0.749**
SPOT1e-3	0.178	0.644	0.278	0.733

Table 5.3: Evaluation metrics on A2 Benchmark

Algorithm	Precision	Recall	F-score	AUC
AR_comb_3sigma	**0.951**	0.455	0.616	0.844
AR_comb_F-score	0.145	0.227	0.177	0.626
AR_diff_3sigma	0.633	0.412	0.499	0.704
AR_diff_F-score	0.242	0.264	0.252	0.592
AR_dynamic_3sigma	0.0	0.0	0.0	0.485
AR_dynamic_F-score	0.052	0.258	0.087	0.664
Autoencoder_3sigma	0.461	0.249	0.323	0.693
Autoencoder_F-score	0.031	0.852	0.06	0.645
FluxEV1e-1	0.403	**0.996**	0.574	**0.995**
FluxEV1e-2	0.645	0.987	0.78	0.992
FluxEV1e-3	0.855	0.893	**0.873**	0.943
SPOT1e-1	0.034	**0.996**	0.065	0.771
SPOT1e-2	0.036	**0.996**	0.07	0.788
SPOT1e-3	0.065	0.929	0.122	0.859

Table 5.4: Evaluation metrics on A3 Benchmark

Algorithm	Precision	Recall	F-score	AUC
AR_comb_3sigma	**1.0**	0.007	0.015	0.537
AR_comb_F-score	0.21	0.043	0.072	0.541
AR_diff_3sigma	0.946	0.647	**0.768**	0.846
AR_diff_F-score	0.308	0.485	0.377	0.695
AR_dynamic_3sigma	0.998	0.043	0.083	0.687
AR_dynamic_F-score	0.101	0.331	0.154	0.526
Autoencoder_3sigma	0.464	0.054	0.097	0.58
Autoencoder_F-score	0.099	0.897	0.179	0.573
FluxEV1e-1	0.273	0.874	0.416	0.855
FluxEV1e-2	0.859	0.515	0.644	0.73
FluxEV1e-3	0.904	0.133	0.231	0.542
SPOT1e-1	0.334	**0.911**	0.489	0.876
SPOT1e-2	0.482	0.869	0.62	**0.899**
SPOT1e-3	0.729	0.647	0.685	0.852

Table 5.5: Evaluation metrics on A4 Benchmark

Algorithm	Precision	Recall	F-score	AUC
AR_comb_3sigma	0.774	0.018	0.035	0.533
AR_comb_F-score	0.129	0.1	0.113	0.533
AR_diff_3sigma	**0.878**	0.527	**0.658**	0.776
AR_diff_F-score	0.238	0.564	0.334	0.705
AR_dynamic_3sigma	0.565	0.09	0.155	0.667
AR_dynamic_F-score	0.094	0.307	0.144	0.533
Autoencoder_3sigma	0.157	0.075	0.102	0.539
Autoencoder_F-score	0.083	0.815	0.15	0.561
FluxEV1e-1	0.168	**0.84**	0.28	**0.794**
FluxEV1e-2	0.488	0.403	0.441	0.629
FluxEV1e-3	0.464	0.09	0.15	0.498
SPOT1e-1	0.192	0.755	0.306	0.761
SPOT1e-2	0.23	0.679	0.344	0.757
SPOT1e-3	0.279	0.53	0.365	0.729

Table 5.6: Evaluation metrics on NAB datasets

Algorithm	Precision	Recall	F-score	AUC
AR_comb_3sigma	0.104	0.25	0.147	0.67
AR_comb_F-score	0.0	0.0	0.0	0.535
AR_diff_3sigma	**0.212**	0.4	**0.277**	0.645
AR_diff_F-score	0.033	0.25	0.059	0.623
AR_dynamic_3sigma	0.101	0.2	0.134	0.572
AR_dynamic_F-score	0.009	0.65	0.018	**0.676**
Autoencoder_3sigma	0.125	0.2	0.154	0.524
Autoencoder_F-score	0.004	0.35	0.009	0.462
FluxEV1e-1	0.004	0.7	0.008	0.464
FluxEV1e-2	0.004	0.7	0.008	0.464
FluxEV1e-3	0.004	0.7	0.008	0.464
SPOT1e-1	0.006	**0.8**	0.012	0.624
SPOT1e-2	0.006	**0.8**	0.012	0.624
SPOT1e-3	0.006	**0.8**	0.012	0.624

distribution of normal data point and are not pronounced additive outliers, but contextual anomalies. Therefore there will be many predicted positive timestamps including most of the anomalies – hence the high Recall –, but also resulting in a high number of False Positives – hence the low Precision. While this result is expected from BiDSPOT, FluxEV was tailored for such scenarios. The differenced Autoregressive model more effectively

learns the characteristics of the data and achieves the best F-score one again by striking a balance between lower Recall and higher Precision. The overall poor performance points to the possibility that even though the datasets are of real world scenarios, some of the anomalies are not very distinct from normal datapoints, Ahmad et al. [2] show an example of this on the *machine_temperature_system_failure* dataset.

Time Measurements

In addition to the the evaluation metrics I also took measurements regarding the computation time of the algorithms on the Yahoo! datasets, since the real-time operation is constraint of industrial data analysis as described in Section 3.2.4. NAB datasets were omitted from this metric as I have shown that the Yahoo! datasets provide a much more consistent benchmarks than the NAB datasets.

There are a total of 367 datasets across all Yahoo! benchmarks. The average calibration and average inference time in seconds for each algorithm is reported in Table 5.7. The measurements were taken on a computer with Intel(R) Core(TM) i5-4440@3.10GHz CPU, 7 GBs of average available RAM and AMD Radeon R7 200 Series GPU.

Similarly to Braei and Wagner, I plot the the mean AUC from all Yahoo! benchmarks and the average of the calibration and inference times per algorithm group (with the Autoregressive models appearing separately) on a scatterplot, shown in Figure 5.13. The ideal algorithm would be situated at coordinate $(1, 0)$, at the lower right corner of the plot as this corresponds to the ideal detector at 0 execution time. The better suited for industrial anomaly detection is, the closer it will be to this point on the chart. On the figure SPOT approached this the most with an average execution time of 0.068 seconds and average AUC of 0.793. The Autoencoder model takes only the 4^{th} place in terms of average AUC, however, it does this at the expense of 38.56 seconds of mean execution time. In contrast, all the probabilistic anomaly detection solutions approach 0 on their computation time needs while performing on par or better than the Autoencoder.

Table 5.7: Average of computation times per algorithm on Yahoo! benchmarks

Algorithm	Average calibration time/dataset (s)	Average inference time/dataset (s)
AR_comb	0.182844	0.049338
AR_comb_3sima	0.029204	0.249066
AR_diff_3sigma	0.004136	0.29614
AR_diff_F-score	0.010541	0.040176
AR_dynamic_3sigma	0.004712	1.459437
AR_dynamic_F-score	0.008429	1.095265
Autoencoder_3sigma	3.091055	72.883122
Autoencoder_F-score	6.974373	71.27523
FluxEV1e-1	**0.0**	**0.0**
FluxEV1e-2	**0.0**	**0.0**
FluxEV1e-3	**0.0**	**0.0**
BiDSPOT1e-1	0.015716	0.132997
BiDSPOT1e-2	0.01528	0.113712
BiDSPOT1e-3	0.015284	0.113429

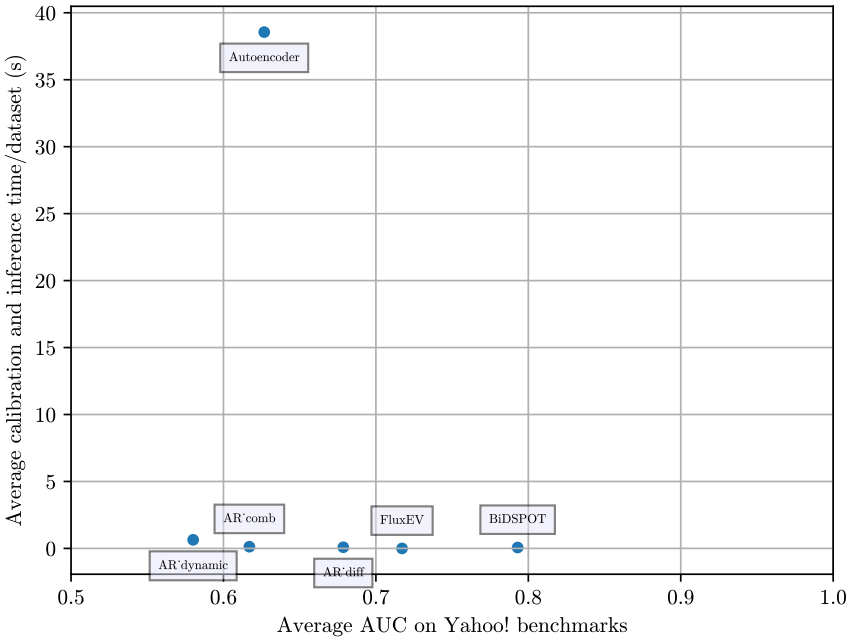


Figure 5.13: Average AUC vs average computation time measured on Yahoo! benchmarks

5.3 Evaluation on Multivariate Datasets

5.3.1 Datasets Used for Comparison

In the multivariate evaluation scenario I used the following multivariate time series:

- **Secure Water Treatment (SWAT) dataset [13]** The dataset is benchmark dataset for cyber-security research at a water treatment testbed site hosted by iTrust, Centre for Research in Cyber Security, Singapore University of Technology and Design, available through a request form on the iTrust Website [23].

  The testbed is a scaled-down replica of a commercial water treatment plant consisting of six stages:

  - **Stage 1**: Raw water intake
  - **Stage 2**: Chemical disinfection
  - **Stage 3**: Ultrafiltration
  - **Stage 4**: Dechlorination using ultraviolet lamps
  - **Stage 5**: Purification by reverse osmosis
  - **Stage 6**: Ultrafiltration membrane backwash and cleaning

  The physical system is composed of various tanks with liquid-level and chemical-property sensors inside them. There are several valves, pumps and differential pressure meters (flow sensors) installed that control when to start certain cycles. In Stage 4 and oxidation reduction potential (ORP) sensor monitors chlorine levels,

while Stage 5 has the most complex physical design and programming due to the sensitivity of the nanometer membranes in the reverse osmosis unit.

The components are connected by a multi-level network. L1 houses the SCADA and Historian servers and HMIs. This layer is connected to each PLC with a star topology Ethernet network. The Historian server periodically queries all the PLCs of the current sensor data. On L0 the Ethernet capable PLCs are found in a dual configuration for each Stage to serve as a backup. The sensors and actuators are not directly connected to the PLCs, but through Remote IO (RIO) devices that perform the digital-analog conversions of signals and commands. The RIOs and PLCs are connected via a ring topology Ethernet network which also serves as a redundant replacement for L0. The PLCs can be managed through TCP/IP.

Data is collected throughout all the stages from all 51 sensors and actuators, as well as network traffic data from the SCADA system and operator workstations. In this evaluation I only consider the physical data. (There are an additional 14 channels of pump data that are constant during the calibration portion of the data, but in the test data changes are not considered anomalous. Due to this behavior I forgo these channels in the evaluation.)

The data was collected through 11 days of operation, 7 of them under normal circumstances and 4 with attack scenarios, all datapoints have a binary label accordingly.

Using the framework of Garg et al. [11], the longest anomaly of 598 minutes is cut back to the average event length of 550 seconds, as the scores have largely reflected only whether or not the long anomaly was detected without this change. Furthermore, as two attack scenarios were launched one after the other, these two are merged into one single anomaly event. In total there are 35 anomalous events recorded in the dataset.

- **Skoltech Anomaly Benchmark (SKAB) [24]** The Skoltech Anomaly Benchmark is a multivariate dataset designed for evaluating anomaly detection algorithms. The dataset contains both point outliers and changepoints.

  The data is collected from a water circulation system testbed that simulates a real industrial scenario with its control system. Anomalies are induced in the system by partially closing valves, temperature variations, reduction of motor power, drastic water level changes and scenarios leading to cavitation (the formation of small vapor-filled cavities in the liquid).

  The authors provide separate train and test datasets.

- **SMAP and MSL datasets** These datasets are telemetry data collected from two space missions, Soil Moisture Active Passive (SMAP) satellite and Mars Science Laboratory (MSL) Curiosity Rover. The data is labeled by experts and is used by mission personnel to process unexpected events that impact a spacecraft or mean a potential risk for it during mission operations [18].

  In these datasets there is only one channel with sensory data, the values in the other channels are one-hot-encoded command data. As Hundman et al., I use all the channels as inputs for the detection algorithms, however only the sensor channel error is used for anomaly detection.

  The authors provide train-test splits with the training data going from $t-5$ days to $t-3$ days, and the test portion from $t-3$ to $t+2$ days, where t stands for the timestamp of the first anomaly in the test set. With a minute-based sampling, the training sets are shorter than for the other datasets.

- **Server Machine Dataset (SMD) [42, 43]** SMD is a minute-based sampled dataset collected over 5 weeks at a large Internet company by Su et al.. The anomalies have been labeled by domain experts based on incident reports. There are 38 channels in the dataset in total. As the authors suggest, the first half of the dataset is used for model calibration, and the second half for testing.

The SMAP, MSL and SMD datasets are formed of multiple entities, essentially multiple physical units of the same type, or the same channels observed at a different time. This results in multiple multivariate time series for these datasets. As in [11, 18, 42], separate models are trained and evaluated for each entity with the average of the individual metric results reported.

Table 5.8: Summary of multivariate datasets used for evaluation

Name	Entities	Channels, m	Average train length	Average test length	Anomaly percentage (%)	Average number of anomaly events	Event time (min)	Sampling rate (s)
SWAT	1	51	473400	414569	4.65	35.00	1.7-28.1	1
SKAB	1	8	9401	35600	36.70	34.00	2.4-9.8	1
MSL	27	55	2160	2731	12.02	1.33	11-1141	60
SMAP	55	25	2556	8071	12.40	1.26	31-4218	60
SMD	28	38	25300	25301	4.21	11.68	2-3160	60

5.3.2 Evaluation Preliminaries

Data Pre-processing

As mentioned in Section 4.2, the calibration/training portion of the datasets are normalized, while the test datasets are clipped into the $[-4, 5]$ interval to prevent excessively large values from a particular channel skewing the overall scores [11].

Parameter Settings

The algorithms are set up similarly to the univariate evaluation settings from Section 5.2.2.

BiDSPOT and FluxEV are run on each channel and evaluated with ($q \in \{10^{-1}, 10^{-3}, 10^{-5}, 10^{-10}\}$). The depth parameter for SPOT and EWMA window size for FluxEV is set to 100 for each dataset as recommended by [42, 11], the depth of FluxEV is set to 50. FluxEV is run without the *Second-step (periodic) smoothing* component. The channel-wise thresholds of the algorithms are transformed into channel-wise errors as described in Section 4.2.3.

I evaluate the static, dynamically recalibrating and the differenced version of the Autoregressive model form Section 4.1.3. The lag parameter is set to the recommended window size of 100. The static version of the algorithm does not combine multiple models unlike the univarite version. However, in order to stay consistent with the naming scheme in the charts, I kept the name *AR_comb* for this version.

For the Autoencoder, I used the Univariate Autoencoder implementation provided by Garg et al. The algorithm trains for 100 epochs with 5 hidden layers, a learning rate of 0.001 and a batch size of 256. The authors found the best values for the hyperparameters empirically. The Autoencoder uses the the recommended window size of 100 for each dataset as the subsequence length and input size of the alogrithm. For the SWAT dataset the subsequence window does not move by only 1 timestep, but by 10 to compensate for the length of the dataset.

Table 5.9: Parameters for Gauss-D and Gauss-D-K per multivariate datasets

Dataset	W	σ_k
SWAT	100000	120
SKAB	100	1
MSL	2000	10
SMAP	2000	10
SMD	25000	1

The algorithms are also evaluated for Voting aggregation (for further details refer to Section 4.2.1). The threshold used for channel-wise anomaly scoring is the automatic threshold(s) set up by SPOT and FluxEV, and the ones obtained by the Three Sigma Rule for the Autoregressive and Autoencoder models.

Aggregation Parameters

For the scoring functions only results with Gauss-D and Gauss-D-K aggregation are reported (for further details see Section 4.2.1), as only these are apt for streaming scenarios and are compatible with all evaluated approaches. The window sizes and convolution kernel sigmas were tuned by Garg et al. to values comparable to the training size and can be found in Table 5.9.

Thresholding

I report evaluation results with the Best-F-score and the Tail-probability methods. Top-k was omitted for reasons detailed in Section 4.2.2. Having tested DSPOT on a subset of the datasets for thresholding, I saw no improvement over the ones provided by the Tail-probability thresholding, however evaluation time has substantially increased. Therefore I do not report results with this thresholding method.

5.3.3 Experiment Results

In this section I report the anomaly detection performance achieved by the tested algorithms on the multivariate datasets. The comparison is done following several aspects:

- reporting anomaly detection metrics across the datasets,
- comparing Voting, Gauss-D and Gauss-D-K aggregation techniques,
- comparing Windowed Evalutation F-score and Composite F-score,
- investigating the effectiveness of Tail-probability thresholding,
- investigating the possibilities of ensemble learning to improve detection performance.

Several of these comparisons use box-plots to compare metrics. The box-plot is a statistical tool that visualizes data distribution. An explanation of the features can be seen in Figure 5.14. The data distribution is observed and a box is drawn from the first quartile (25^{th} percentile) to the third quartile (75^{th} percentile) of the data. These quartiles are the median values of the lower- and upper-half of the dataset respectively. The second

quartile is the median which is marked with a horizontal line on the plot. The min and max extends beyond the first and third quartiles by $1.5 \cdot IQR$, where IQR stands for the distance between the first and third quartiles. Should there be any values that are outside this range, those are marked individually and are considered extreme values or outliers regarding the data distribution. The mean value of the data may also be marked on the box-plot, in this thesis the mean is marked by a green triangle.

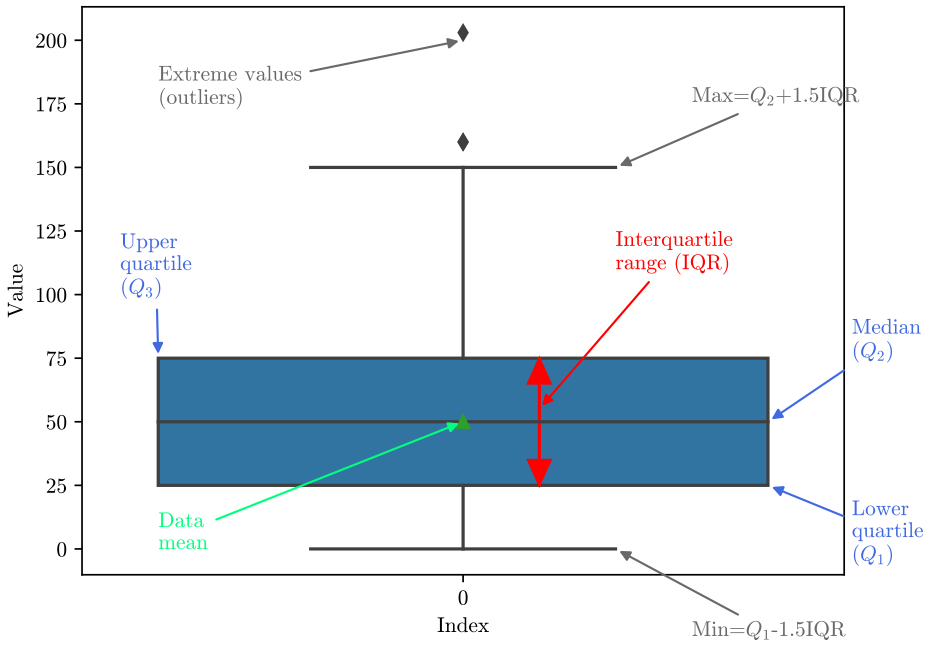


Figure 5.14: Characteristics of the box-plot

Full results for every anomaly detection algorithm with both Gauss-D and Gauss-D-K aggregation, as well as all 6 thresholding methods (Best-F-score + 5 distinct Tail-probability thresholding methods) are reported in Tables A.2 to A.6.

Results with Voting Aggregation

Voting aggregation is a simple aggregation method for multivariate time series. I set up all algorithms with a Single-, a Majority- and a Unanimous Voting functionality and ran them on a subset of available datasets. The results with the average F-scores from two datasets can be seen in Figure 5.15, similar behavior was shown by the algorithms on other datasets. It can be clearly seen that compared to the more sophisticated Gauss-D aggregation Voting stays behind in terms of detection performance. The Single Voting only approached the performance of the differenced Autoregressive and the Autoencoder models, while Majority Voting attained less than half of the F-scores of Gauss-D scoring for all algorithms. Unanimous voting achieved an F-score of 0 in the case of every algorithm. This shows that – at least in the case of these datasets – anomalies are not present across all channels at the same time.

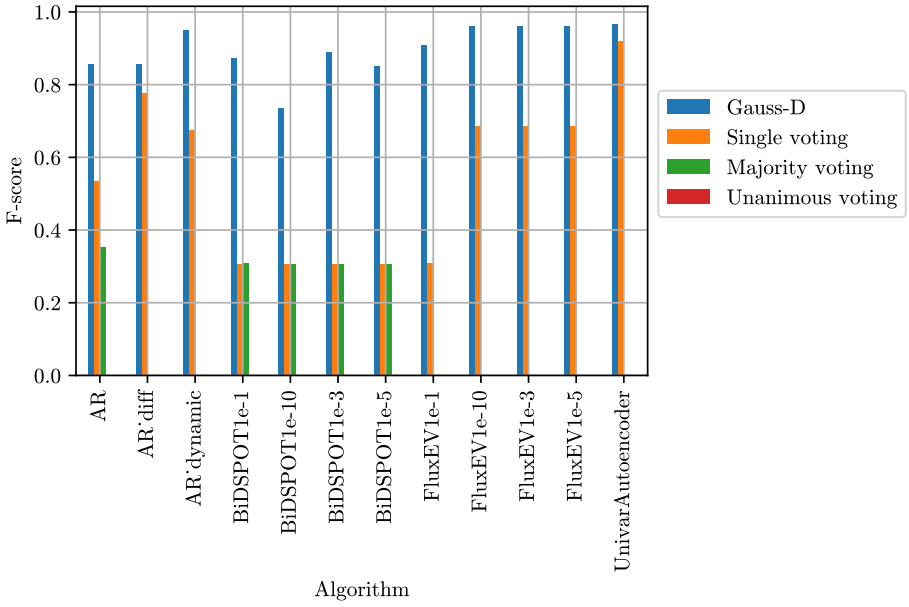


Figure 5.15: Comparison of F-scores achieved by the Voting aggregation and the Gauss-D aggregation functions on SKAB and SMD datasets

Evaluation of Anomaly Metrics with Gauss-D Aggregation and Best-F-score Thresholding

I report the evaluation metrics of Precision, Recall, F-score and AUC of all the evaluated algorithms on all multivariate datasets in this section. Results are visualized in bar-charts in Figures 5.16 to 5.20 and are shown numerically in Tables 5.10 to 5.14. In the tables Precision is written as Normalized Precision to contrast it to Point-wise Precision that is used for the F_c-score also shown in the tables. The AUC metric was calculated from the point-wise anomaly detection metrics. The best metric value in each column of the tables is marked in bold.

Unfortunately, due to limited computational resources I was not able to run FluxEV with $q = 10^{-10}$ on the SWAT dataset, and the dynamic Autoregressive model on the SWAT and SMD datasets, hence these algorithms are omitted from corresponding tables and charts.

On the SWAT dataset the overall detection performance of every algorithm was good with high Precision across all of them, however, with mediocre Recall. FluxEV took the lead with $q = 10^{-5}$ in terms of F-score, performing on par with the Autoencoder. When looking at the F_c-score however, the Autoencoder is the clear winner, with the other algorithms achieving substantially lower Precision than in the normalized case. The exception to this is BiDSPOT which achieves the highest Point-wise Precision with $q = 10^{-1}$ although, despite being the second best in terms of F_c-score, it still performs 16.36% worse than the Autoencoder.

On SKAB, all algorithms successfully identify all of the anomalies with Recall being 1 across the board. In spite of this only the Autoencoder and BiDSPOT manage to achieve high Precision as well. The Autoencoder achieves the best results in all reported metrics,

however BiDSPOT with $q = 10^{-5}$ manages to produce comparable results to it that only differ by 2% on average across all metrics. When observing Point-wise Precision and F_c-score, all algorithms attain approximately the same scores.

On MSL and SMAP the probabilistic algorithms perform poorly relative to the Univariate Autoencoder. In the case of MSL the differenced Autoregressive model wins second place by a difference of 13.6% in F-score, while on SMAP the second best algorithm, FluxEV with $q = 10^{-10}$ lags behind by 26.7%. The poor performance on these datasets can be potentially explained by the shorter length of the training data and the large number of one-hot-encoded channels with discrete values where outliers are not necessarily the timestamps with the most extreme values. This shows in the subpar performance of BiDSPOT, while FluxEV achieves a better score since it is able to handle the non-extreme outliers with unusual patterns within the normal distribution threshold. The relatively good run of the Autoencoder model shows that it is more capable of finding obscure patterns in the data by tuning the model parameters to it.

Algorithms show a more favorable performance on SMD. While the Autoencoder once again triumps in almost all metrics but both BiDSPOT ($q = 10^{-1}$ and $q = 10^{-3}$) and FluxEV with all parameter settings manage to exceed an F-score of 0.8 and come close to the Autoencoder model. FluxEV ($q = 10^{-5}$) achieves the second highest F-score that is only 4.7% worse than that of the Autoencoder. The Point-wise Precision and F_c-score tells a similar story with BiDSPOT ($q = 10^{-3}$) lagging by 9.2% behind the Autoencoder.

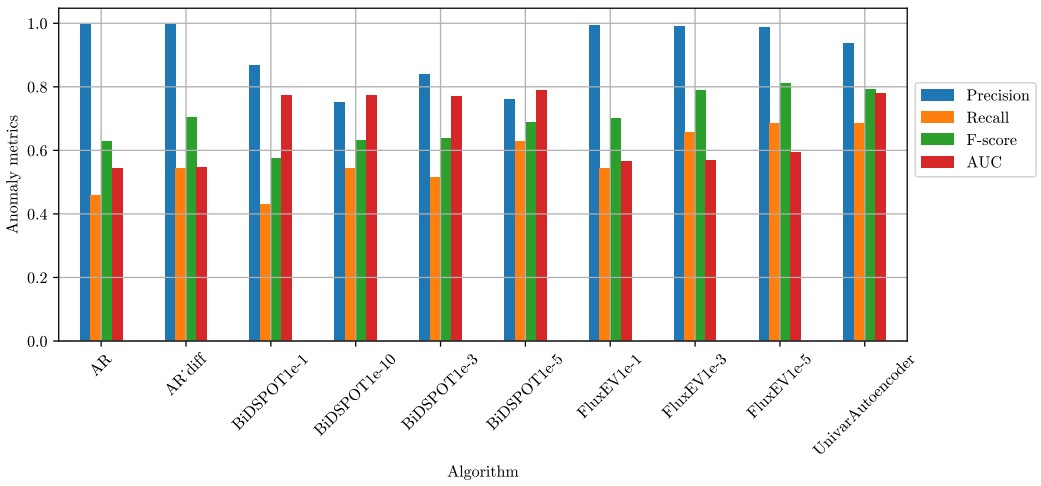


Figure 5.16: Precision, Recall, F-score and AUC metrics on SWAT

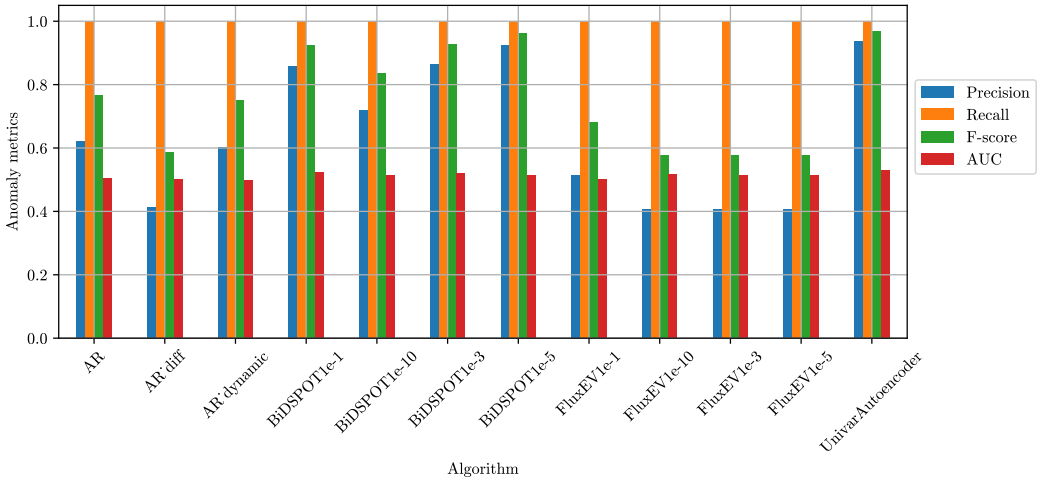


Figure 5.17: Precision, Recall, F-score and AUC metrics on SKAB

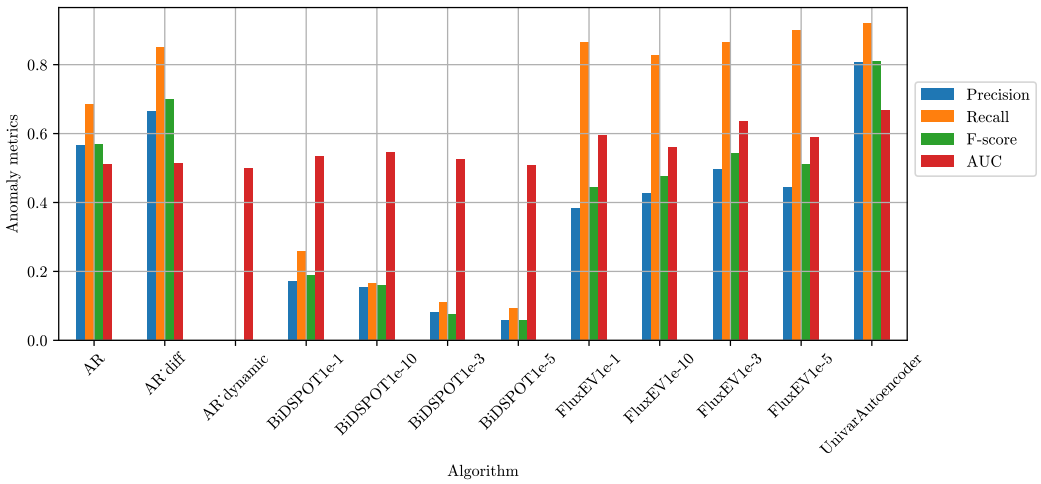


Figure 5.18: Precision, Recall, F-score and AUC metrics on MSL

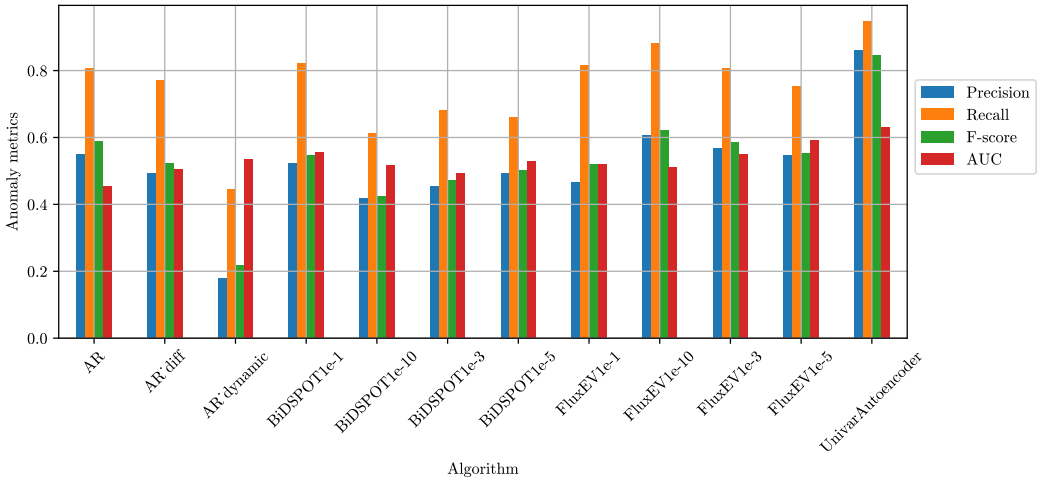


Figure 5.19: Precision, Recall, F-score and AUC metrics on SMAP

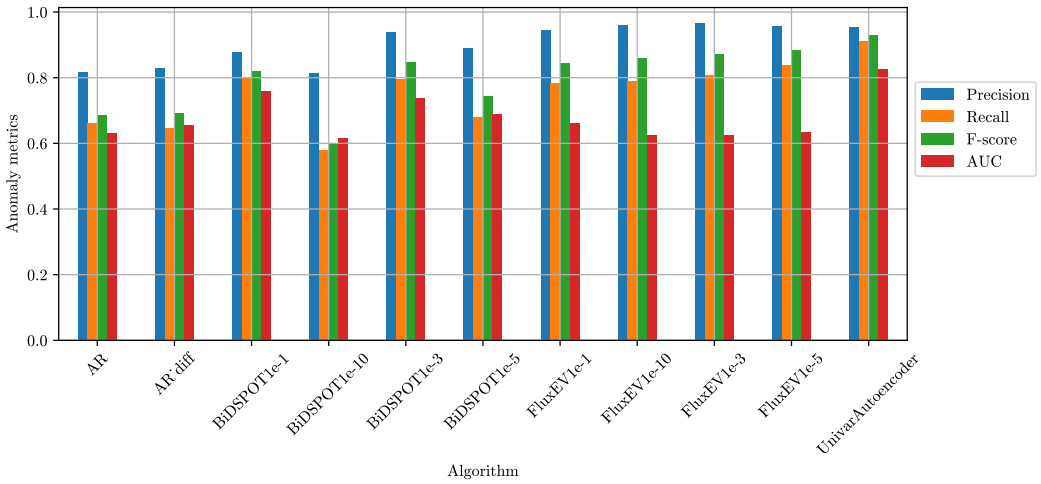


Figure 5.20: Precision, Recall, F-score and AUC metrics on SMD

Table 5.10: Evaluation metrics on SWAT dataset with Gauss-D aggregation and Best-F-score thresholding

Algorithm	Point-wise Precision	Normalized Precision	Recall	F-score	F_c-score	AUC
AR	0.651	**0.998**	0.457	0.627	0.537	0.542
AR_diff	0.562	0.997	0.543	0.703	0.553	0.546
BiDSPOT1e-1	**0.724**	0.869	0.429	0.574	0.538	0.773
BiDSPOT1e-10	0.573	0.752	0.543	0.631	0.557	0.771
BiDSPOT1e-3	0.66	0.838	0.514	0.637	0.578	0.768
BiDSPOT1e-5	0.549	0.761	0.629	0.688	0.586	**0.79**
FluxEV1e-1	0.418	0.992	0.543	0.702	0.473	0.564
FluxEV1e-3	0.364	0.989	0.657	0.79	0.469	0.567
FluxEV1e-5	0.357	0.988	**0.686**	**0.81**	0.47	0.593
UnivarAutoencoder	0.716	0.936	**0.686**	0.792	**0.701**	0.78

Table 5.11: Evaluation metrics on SKAB dataset with Gauss-D aggregation and Best-F-score thresholding

Algorithm	Point-wise Precision	Normalized Precision	Recall	F-score	F_c-score	AUC
AR	0.372	0.62	**1.0**	0.765	0.542	0.504
AR_diff	0.371	0.414	**1.0**	0.586	0.541	0.501
AR_dynamic	0.369	0.602	**1.0**	0.751	0.539	0.499
BiDSPOT1e-1	0.398	0.859	**1.0**	0.924	0.57	0.523
BiDSPOT1e-10	0.38	0.718	**1.0**	0.836	0.55	0.513
BiDSPOT1e-3	0.385	0.865	**1.0**	0.928	0.556	0.52
BiDSPOT1e-5	0.382	0.924	**1.0**	0.961	0.553	0.513
FluxEV1e-1	0.384	0.515	**1.0**	0.68	0.555	0.501
FluxEV1e-10	0.382	0.407	**1.0**	0.578	0.553	0.516
FluxEV1e-3	0.382	0.407	**1.0**	0.578	0.553	0.515
FluxEV1e-5	0.382	0.406	**1.0**	0.578	0.553	0.515
UnivarAutoencoder	**0.401**	**0.936**	1.0	**0.967**	**0.572**	**0.53**

Table 5.12: Evaluation metrics on MSL dataset with Gauss-D aggregation and Best-F-score thresholding

Algorithm	Point-wise Precision	Normalized Precision	Recall	F-score	F_c-score	AUC
AR	0.431	0.567	0.685	0.568	0.466	0.511
AR_diff	0.466	0.666	0.852	0.701	0.528	0.515
AR_dynamic	0.0	0.0	0.0	0.0	0.0	0.5
BiDSPOT1e-1	0.16	0.17	0.259	0.19	0.179	0.534
BiDSPOT1e-10	0.147	0.154	0.167	0.159	0.154	0.547
BiDSPOT1e-3	0.075	0.082	0.111	0.077	0.071	0.527
BiDSPOT1e-5	0.056	0.058	0.093	0.06	0.059	0.509
FluxEV1e-1	0.335	0.383	0.864	0.445	0.407	0.596
FluxEV1e-10	0.393	0.428	0.827	0.477	0.446	0.56
FluxEV1e-3	0.441	0.497	0.864	0.543	0.504	0.635
FluxEV1e-5	0.414	0.445	0.901	0.51	0.483	0.588
UnivarAutoencoder	**0.71**	**0.808**	**0.92**	**0.811**	**0.727**	**0.667**

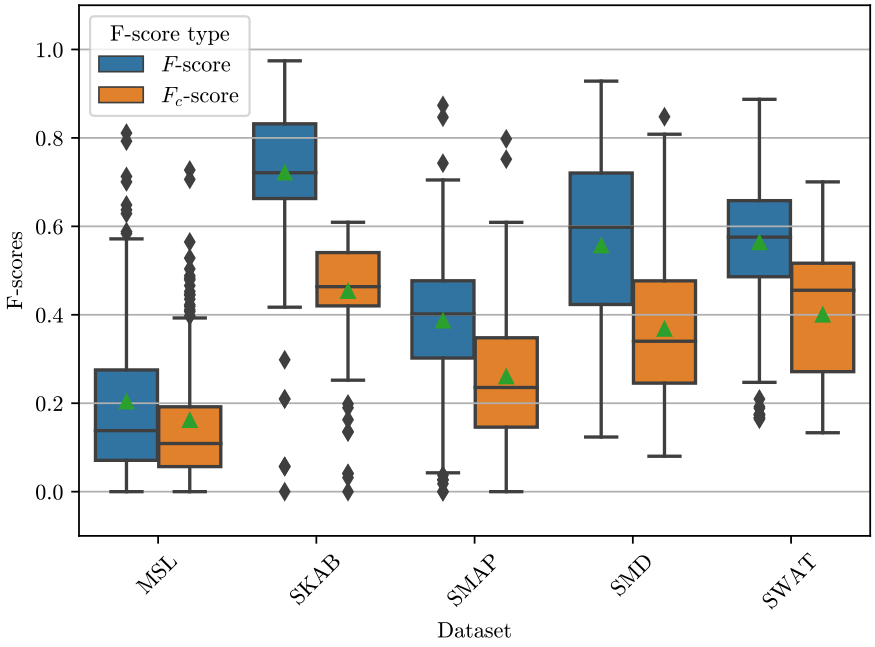
Table 5.13: Evaluation metrics on SMAP dataset with Gauss-D aggregation and Best-F-score thresholding

Algorithm	Point-wise Precision	Normalized Precision	Recall	F-score	F_c-score	AUC
AR	0.371	0.549	0.806	0.587	0.429	0.454
AR_diff	0.312	0.494	0.772	0.522	0.374	0.505
AR_dynamic	0.152	0.179	0.444	0.216	0.189	0.536
BiDSPOT1e-1	0.44	0.523	0.821	0.548	0.478	0.557
BiDSPOT1e-10	0.339	0.417	0.611	0.424	0.353	0.518
BiDSPOT1e-3	0.359	0.454	0.682	0.47	0.39	0.493
BiDSPOT1e-5	0.448	0.493	0.66	0.503	0.466	0.528
FluxEV1e-1	0.302	0.465	0.815	0.521	0.353	0.521
FluxEV1e-10	0.445	0.606	0.883	0.62	0.503	0.512
FluxEV1e-3	0.38	0.567	0.806	0.586	0.436	0.549
FluxEV1e-5	0.379	0.546	0.753	0.553	0.426	0.591
UnivarAutoencoder	**0.719**	**0.86**	**0.948**	**0.847**	**0.752**	**0.63**

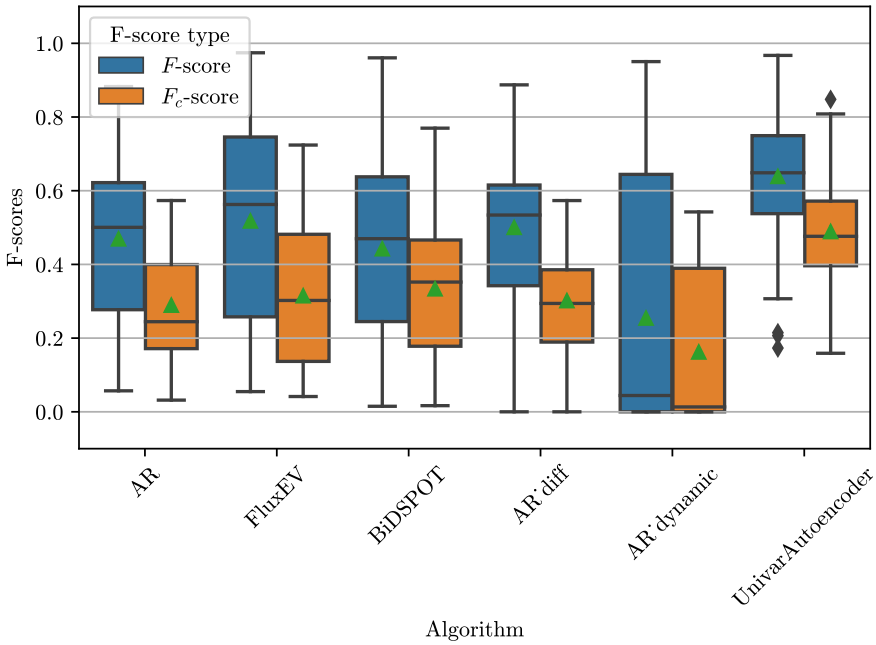
Table 5.14: Evaluation metrics on SMD dataset with Gauss-D aggregation and Best-F-score thresholding

Algorithm	Point-wise Precision	Normalized Precision	Recall	F-score	F_c-score	AUC
AR	0.398	0.815	0.662	0.685	0.449	0.632
AR_diff	0.376	0.827	0.647	0.693	0.441	0.657
BiDSPOT1e-1	0.609	0.879	0.799	0.821	0.649	0.759
BiDSPOT1e-10	0.7	0.815	0.58	0.601	0.568	0.614
BiDSPOT1e-3	0.777	0.938	0.795	0.846	0.77	0.739
BiDSPOT1e-5	0.754	0.89	0.68	0.742	0.683	0.688
FluxEV1e-1	0.6	0.945	0.782	0.844	0.65	0.662
FluxEV1e-10	0.665	0.959	0.79	0.859	0.709	0.624
FluxEV1e-3	0.681	**0.965**	0.807	0.871	0.724	0.626
FluxEV1e-5	0.651	0.957	0.838	0.884	0.716	0.634
UnivarAutoencoder	**0.808**	0.955	**0.912**	**0.928**	**0.848**	**0.825**

Comparison of F-score and F_c-score



(a) Comparison on a per-dataset basis



(b) Comparison on a per-algorithm basis

Figure 5.21: Comparison of F-score and F_c-score metrics

In the previous subsection I have evaluated the performance of the tested algorithms on multivariate datasets based on both the F-score calculated with the Windowed Evaluation (Section 5.1.3) and by F_c-score (Section 5.1.4). The latter is used by the authors of the multivariate evaluation framework, Garg et al.. In this section I compare these two metrics.

The comparison can be seen in Figure 5.21 represented with boxplots. Figure 5.21a shows a per-dataset comparison. For each dataset all of the algorithms with both Gauss-D and Gauss-D-K aggregation and with every thresholding method was grouped together and plotted on the corresponding boxplot. For all datasets the Composite F-score achieves substantially lower scores on average with varying degree. On SKAB all of the F_c-scores lie below the 25^{th}-percentile of the F-scores while on MSL the difference is much smaller, with a median lower by 0.03 and the mean by 0.04.

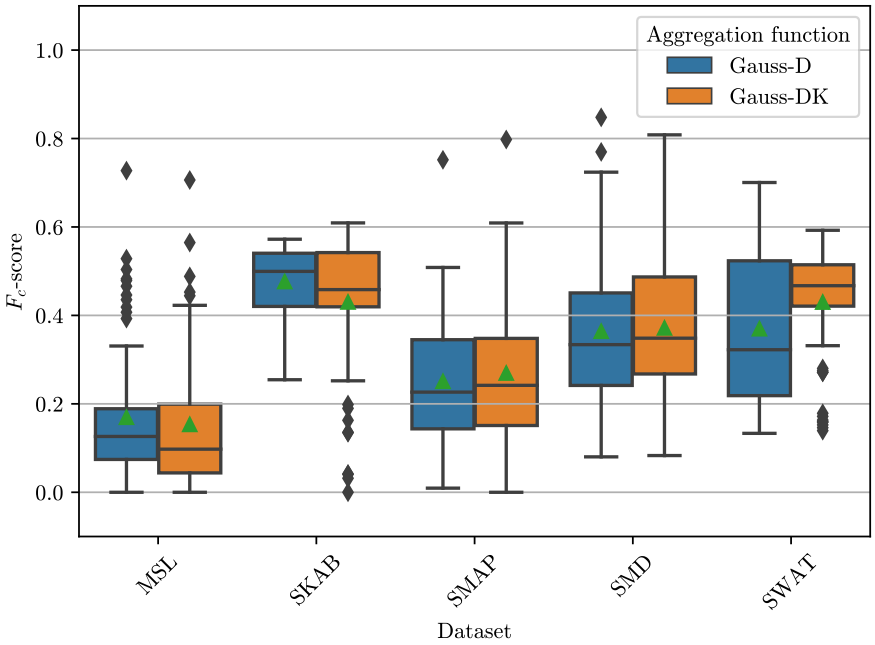
I also evaluated the same metrics with aggregation by algorithm in Figure 5.21b. The results are similar to the per-dataset results with clearly lower scores across all algorithms.

On both charts the F-score and F_c-score boxplots show a similar pattern shifted slightly down, with all of the results falling between $20-35\%$ for per-dataset comparison, and $25-40\%$ for per-algorithm comparison. This means that on average the evaluation results with both metrics will have a similar distribution. While this is true for SKAB and SMD, we saw that there were drastic differences in the case of SWAT.

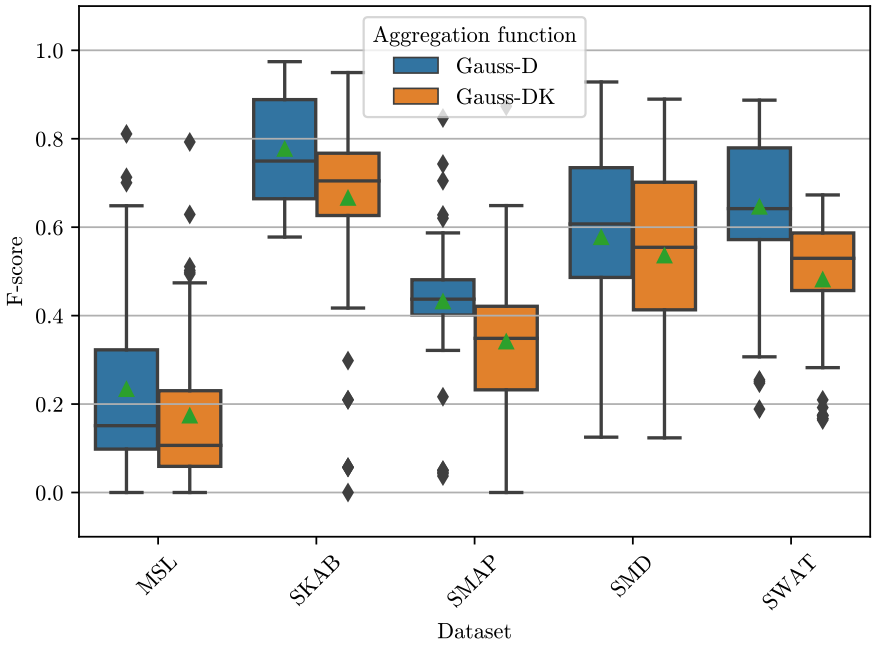
Comparison of Gauss-D and Gauss D-K Aggregation Functions

There are two possible ways for aggregation that I evaluate, Gauss-D and Gauss-D-K. The latter requires a kernel sigma (σ_k) parameter and provides a smoothing for the anomaly score (see Section 4.2.1 for further details). According to Garg et al., at the expense of manual parameter tuning, the Gauss-D-K aggregation function achieves a statistically significant increase in the F_c-score, this is shown in Figure 5.22a with box-plots. The data is aggregated by datasets, as σ_k is tuned per-dataset. As the figure shows there is a slight increase in mean for the SMAP, SMD and SWAT datasets with the extreme values stretching higher for the Gauss-D-K aggregation in all datasets except for SWAT.

To test whether this is true for the Windowed Evaluation approach I plot the same values using the Normalized Precision for the F-score in Figure 5.22b. The figure clearly shows an even more significant decrease in F-score than the increase was in the case of the F_c-score. The mean, median and overall distribution falls lower on the scale with Gauss-D-K than with Gauss-D. This shows that the empirically tuned σ_k sigma kernel values are not universal and have been optimized to give the best possible results with the F_c-score. An alternative tuning may be possible that attains a similar increase in the F-score, however, this process is out of the scope of this thesis.



(a) Comparison of Gauss-D and Gauss-D-K aggregation using F_c-score



(b) Comparison of Gauss-D and Gauss-D-K aggregation using F-score

Figure 5.22: Comparison of Gauss-D and Gauss-D-K aggregation functions

Evaluation of Tail-probability Thresholding

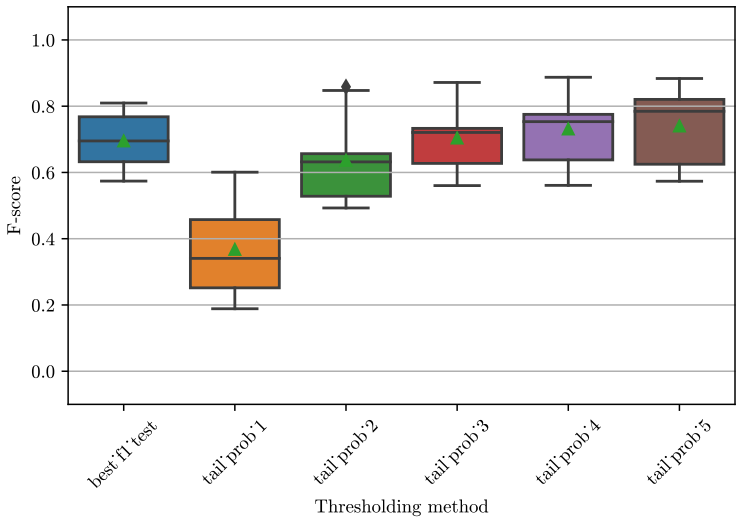


Figure 5.23: Comparison of Best-F-score and Tail-probability thresholding on SWAT

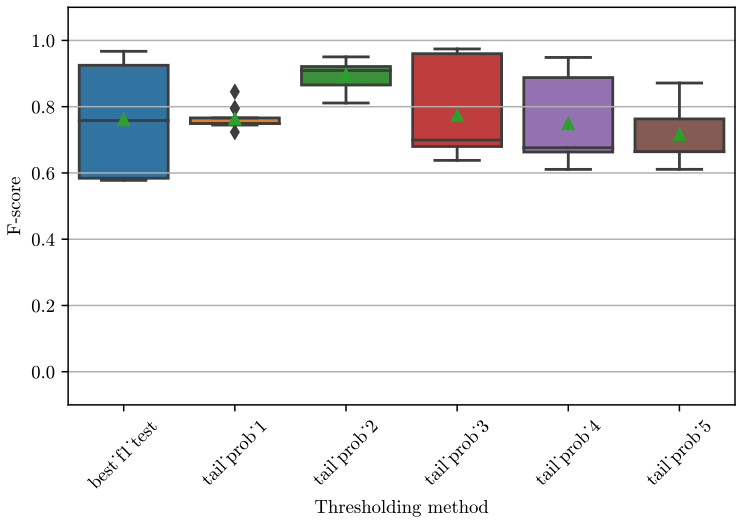


Figure 5.24: Comparison of Best-F-score and Tail-probability thresholding on SKAB

In Figures 5.16 to 5.20 and Tables 5.10 to 5.14 the thresholding was performed by the Best-F-score method. However, this may only be done when anomaly detection evaluation is being performed on labeled data. In real word scenarios, such as an industrial system, the time series is not labeled so an alternative thresholding method needs to be utilized (Section 4.2.2). In this section I investigate how closely can the Tail-probability thresh-

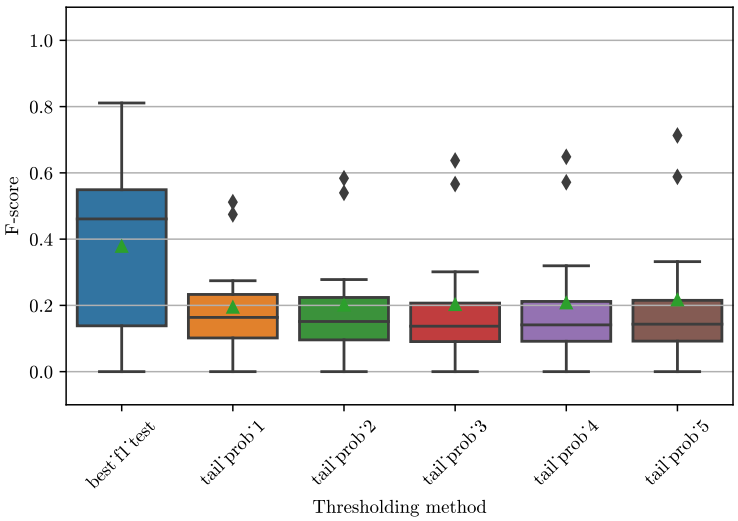


Figure 5.25: Comparison of Best-F-score and Tail-probability thresholding on MSL

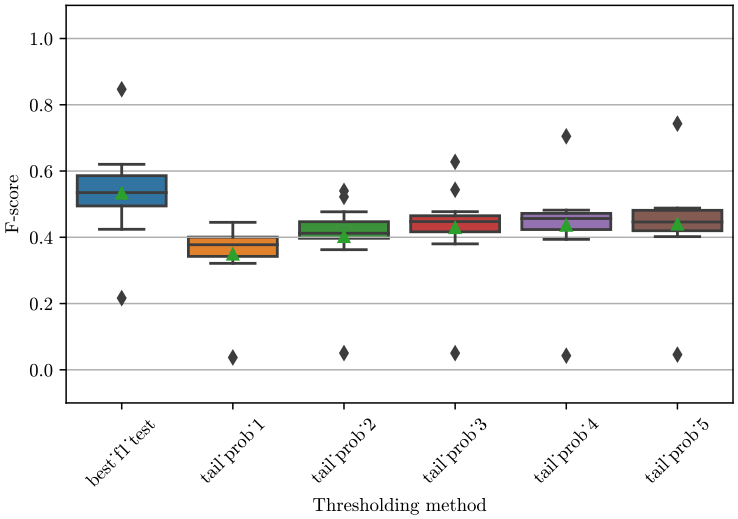


Figure 5.26: Comparison of Best-F-score and Tail-probability thresholding on SMAP

olding approach the F-score achieved by the Best-F-score thresholding. The results are evaluated on a per-dataset (Figures 5.23 to 5.27) and a per-algorithm-group (Figures 5.28 to 5.33) basis where the results of distinctly set up BiDSPOT and FluxEV algorithms are grouped together due to them showing similar distributions when looked at individually.

In the figures the x-axis shows the thresholding method. The first value stands for Best-F-score while the other classes of *tail_prob_x* stand for the version of Tail-probability where $-log_{10}(\epsilon) = x$.

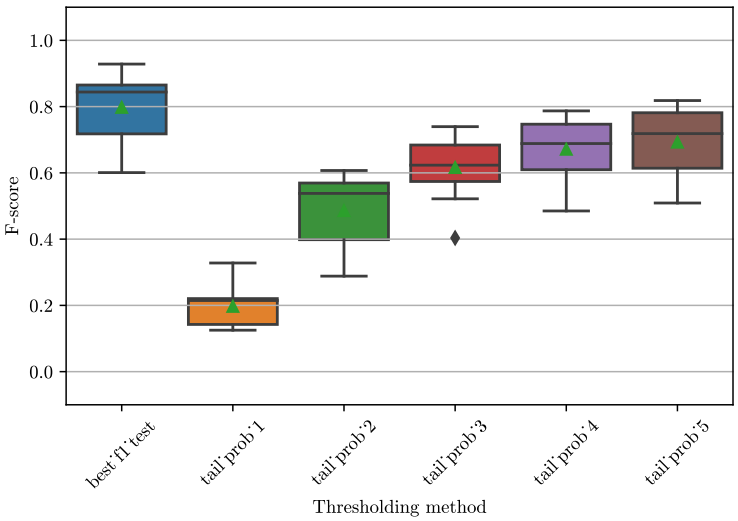


Figure 5.27: Comparison of Best-F-score and Tail-probability thresholding on SMD

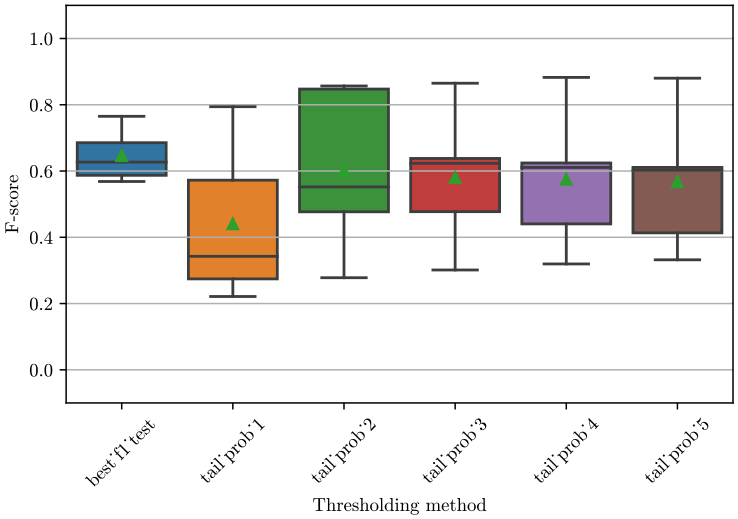


Figure 5.28: Comparison of Best-F-score and Tail-probability thresholding with the Autoregressive model

In the per-dataset evaluation it is shown that for SWAT by choosing $x = 3$, $x = 4$ or $x = 5$, the same F-score distribution it achieved. Therefore, choosing these the F-score will be similar to the result attained by the Best-F-score method. Meanwhile, choosing $x = 1$ or $x = 2$ would almost guarantee a worse F-score as the vast majority of the scores reached by those thresholds lie beneath the distribution shown by Best-F-score.

On SKAB the opposite is true. The highest F-scores can be acquired by the $x = 2$ Tail-probability. The F-scores obtained with this method have a relatively low standard and

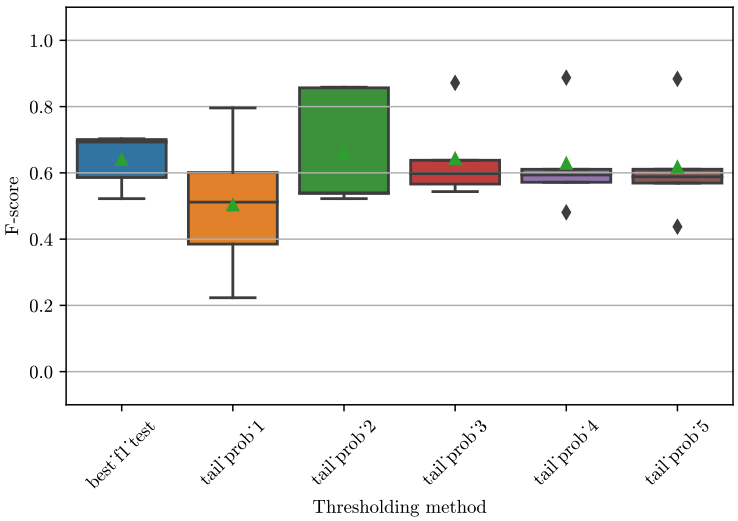


Figure 5.29: Comparison of Best-F-score and Tail-probability thresholding with the Differenced Autoregressive model

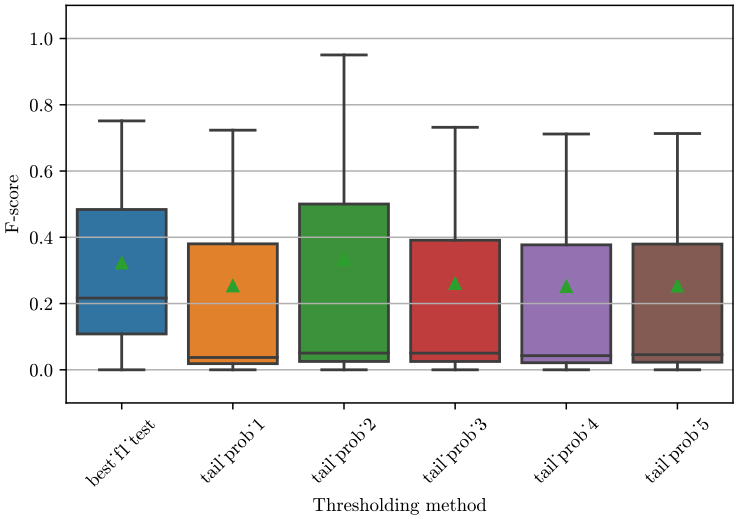


Figure 5.30: Comparison of Best-F-score and Tail-probability thresholding with the Dynamic Autoregressive model

absolute deviation. This combined with a higher average and median than that of the Best-F-score means that even the worst F-score reported by this thresholding function will be higher than the average score given by the Best-F-score thresholding.

On the three remaining datasets – apart from a handful of outliers – only the Tail-probability thresholding method with $x = 5$ in the case of SMD approaches the performance of the Best-F-score. However, this still is subpar with a 75^{th}-percentile falling

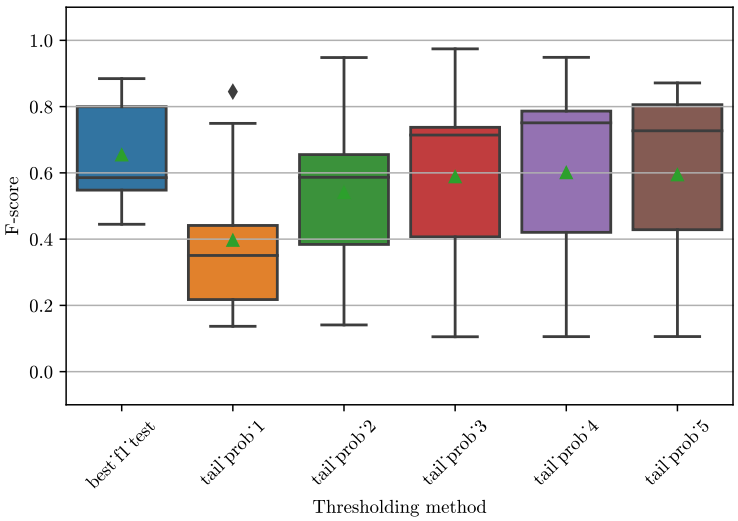


Figure 5.31: Comparison of Best-F-score and Tail-probability thresholding with FluxEV

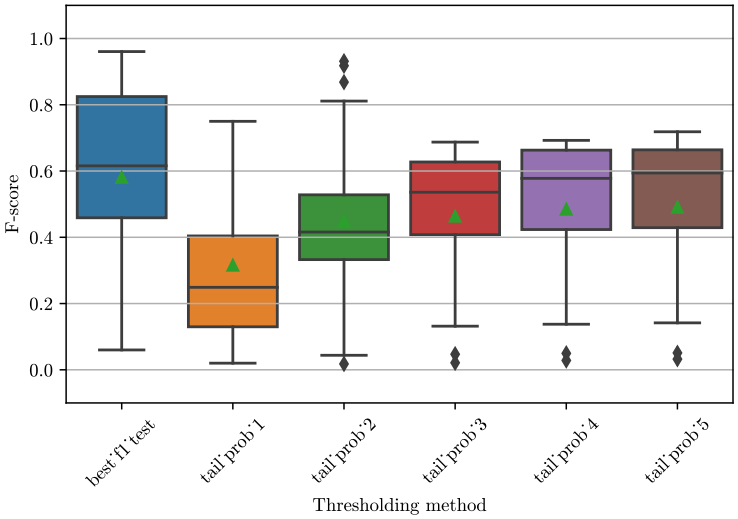


Figure 5.32: Comparison of Best-F-score and Tail-probability thresholding with BiDSPOT

short of the average score of Best-F-score. MSL and SMAP are largely indifferent to the Tail-probability threshold used, as they result in similar F-score distributions.

Regarding the per-algorithm-group evaluation the Autoregressive and differenced Autoregressive models have an optimal thresholding method on these datasets: Tail-probability with $x = 2$. Using this the vast majority of F-scores will be on par or better than can be attained by the Best-F-score method. With BiDSPOT and FluxEV a higher x will achieve the closest but slightly poorer F-score distribution to the distribution of Best-F-score. The

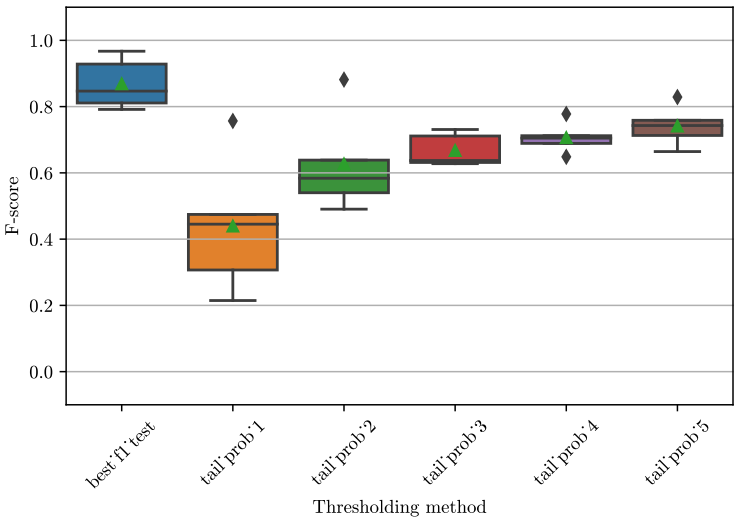


Figure 5.33: Comparison of Best-F-score and Tail-probability thresholding with the Univariate Autoencoder model

dynamic Autoregressive model seems indifferent to the chosen thresholding method with an overall lower achieved average F-score. In the case of the Autoencoder, while the Tail-probability threshold with $x = 5$ approaches the F-score distribution of the Best-F-score method the most, its highest non-outlier score falls below the entire distribution of Best-F-score.

In conclusion the best Tail-probability threshold and whether it can be used effectively is highly dependent on the dataset and the algorithm used.

Possible Combinations of Algorithms for Ensemble Learning

Ensemble learning is an umbrella term for combining outputs of multiple models (also called inducers) with the intention of improving the common detection or prediction performance by using the models to compensate for the errors of each other [36]. The inducer can be any type of prediction or detection algorithm and there are multiple ways of aggregating the model outputs which I do not detail in this thesis.

In this section I investigate which algorithms may be paired to improve their detection performance with ensemble learning. I select the best performing model (by F-score) that uses Gauss-D aggregation from each algorithm group on each dataset. I take the detected anomaly group by each algorithm and assign a letter to it starting from A. The letter reflects the cardinality of the anomaly group. The further a letter is in the alphabet, the larger the detected anomaly group is. The anomaly group cardinality is also reflected in the markersize of the algorithm. I plot the F-score of the selected algorithm (x-axis) and the anomaly group of the algorithm (y-axis) on a scatterplot. Algorithms that detect the exact same anomaly groups are placed under the same y coordinate.

Arrows in a lighter shade of same color as the marker for the algorithm are drawn from Algorithm_1 to Algorithm_2 if A is the anomaly group of Algorithm_1 and B is the anomaly group of Algorithm_2 and $B \subset A$, i.e. B is the proper subset of A. By running these two

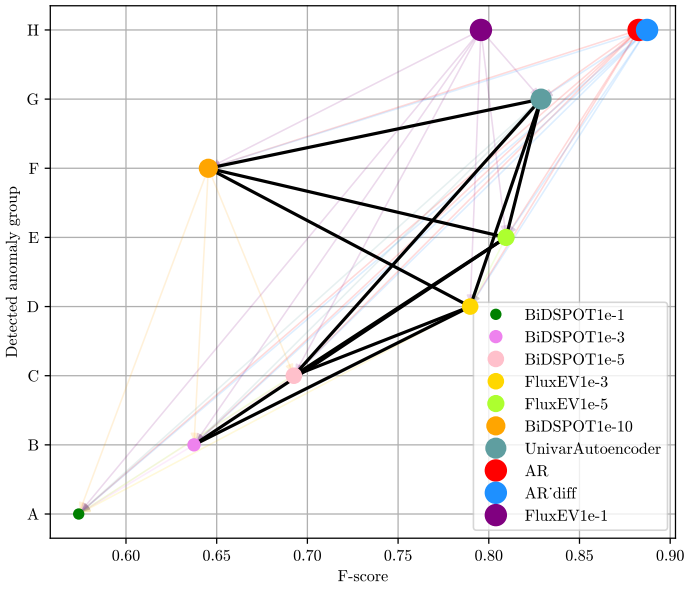


Figure 5.34: Connection between detected anomaly groups and algorithms with possible ensemble learning connections on SWAT

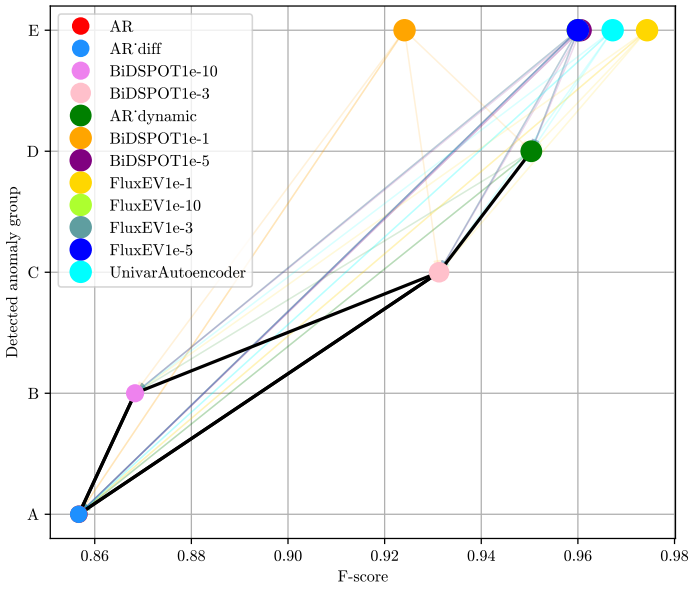


Figure 5.35: Connection between detected anomaly groups and algorithms with possible ensemble learning connections on SKAB

algorithms in an ensemble the set of detected anomalies cannot be extended. However, there is a possibility of increasing precision by mutually influencing the detections of the other algorithm.

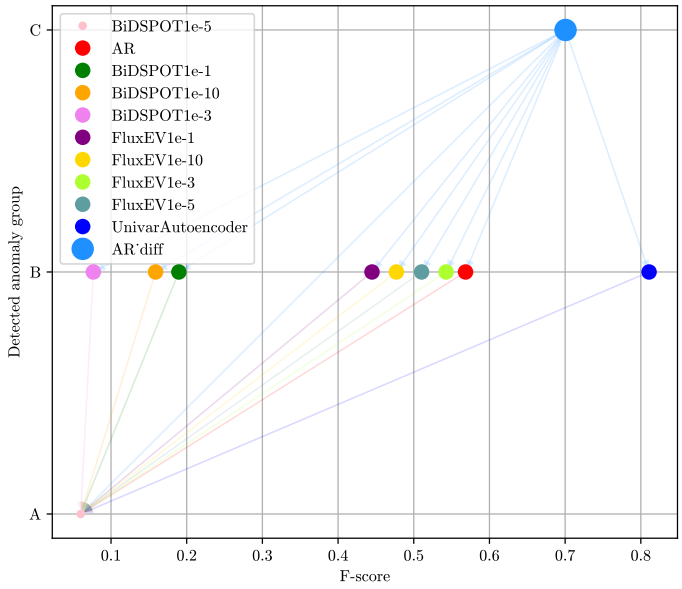


Figure 5.36: Connection between detected anomaly groups and algorithms with possible ensemble learning connections on MSL

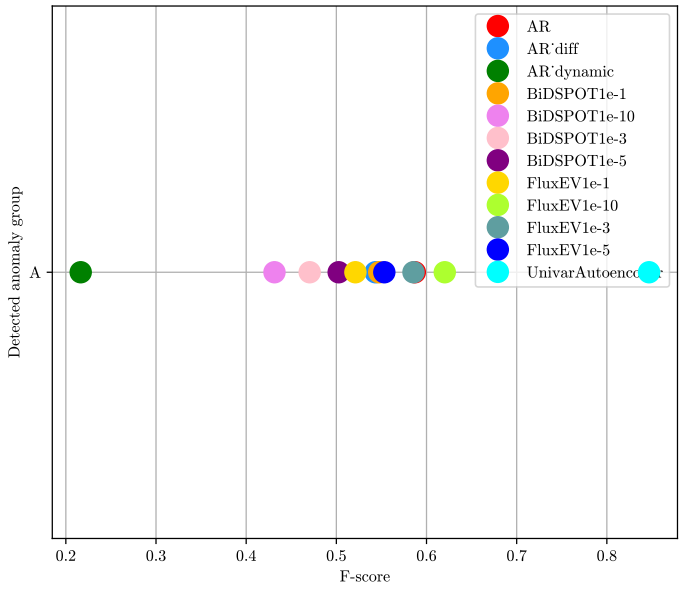


Figure 5.37: Connection between detected anomaly groups and algorithms with possible ensemble learning connections on SMAP

A thick black line connects two algorithms that do not share the same anomaly group, but their anomaly groups are not in a subset relationship. This means that union of the two anomaly groups adds new anomalies to both anomaly groups. These pairings therefore have the potential for expanding the space of detected outliers, however, the increased

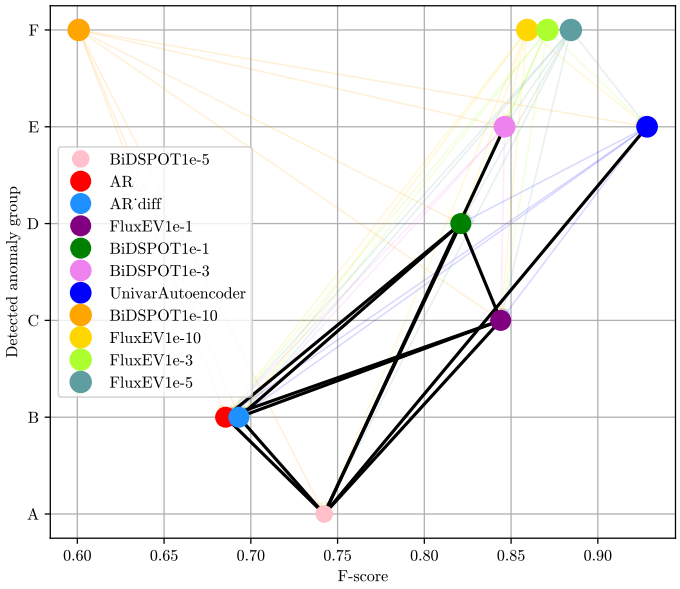


Figure 5.38: Connection between detected anomaly groups and algorithms with possible ensemble learning connections on SMD

Recall may come at the price of a decreased Precision. The possible pairings are listed in Table 5.15.

The algorithms, anomaly groups and their connections are visualized in Figures 5.34 to 5.38.

Table 5.15: Possible pairings of algorithms for ensemble learning

Dataset	Algorithm_1	Algorithm_2
SKAB	AR_dynamic	BiDSPOT1e-3
SKAB	BiDSPOT1e-3	AR
SKAB	BiDSPOT1e-3	AR_diff
SKAB	BiDSPOT1e-3	BiDSPOT1e-10
SKAB	AR	BiDSPOT1e-10
SKAB	AR_diff	BiDSPOT1e-10
SMD	BiDSPOT1e-1	FluxEV1e-1
SMD	BiDSPOT1e-1	BiDSPOT1e-5
SMD	BiDSPOT1e-1	AR
SMD	BiDSPOT1e-1	AR_diff
SMD	FluxEV1e-1	BiDSPOT1e-5
SMD	FluxEV1e-1	AR
SMD	FluxEV1e-1	AR_diff
SMD	BiDSPOT1e-3	BiDSPOT1e-5
SMD	UnivarAutoencoder	BiDSPOT1e-5
SMD	BiDSPOT1e-5	AR
SMD	BiDSPOT1e-5	AR_diff
SWAT	BiDSPOT1e-5	FluxEV1e-5
SWAT	BiDSPOT1e-5	UnivarAutoencoder
SWAT	BiDSPOT1e-5	FluxEV1e-3
SWAT	FluxEV1e-5	BiDSPOT1e-10
SWAT	FluxEV1e-5	UnivarAutoencoder
SWAT	FluxEV1e-5	BiDSPOT1e-3
SWAT	BiDSPOT1e-10	UnivarAutoencoder
SWAT	BiDSPOT1e-10	FluxEV1e-3
SWAT	UnivarAutoencoder	FluxEV1e-3
SWAT	BiDSPOT1e-3	FluxEV1e-3

Chapter 6

Conclusion

The main aim of this thesis was to test the hypothesis of Gers et al. [12] and Polge et al. [34] that simple probabilistic approaches can be effectively used for anomaly detection even in industrial time series. To carry this out I investigated industrial data analysis, the characteristics of distinct outlier types, traits of various anomaly detection approaches and data processing techniques.

In addition I explored the novel features of Industry 4.0 and collected the ways data analysis can be beneficial to manufacturing businesses. I identified the main requirements of industrial data processing algorithms to be real-time operation, fast startup to a detection-ready state and the handling of potential data drift.

Keeping these in mind I selected two probabilistic (BiDSPOT and FluxEV) and one statistical model (Autoregressive Anomaly Detector) to compare against the Autoencoder, the best anomaly detector overall found by Garg et al. [11].

I evaluated the algorithms both in a univariate and in a multivariate scenario. For the latter the channel-wise anomaly aggregation framework by [11] was used. I proposed a data transformation method to make Adaptive Threshold Anomaly Detectors like BiDSPOT and FluxEV work in this framework. As the best aggregation function I opted for Gauss-D, which works in a streaming fashion and adds drift capabilities to each algorithm.

The results in both the univariate and multivariate test cases have shown that the probabilistic or statistical approaches stand as a potential contender against the Autoencoder model, not only matching its performance on the majority of test benchmarks but also outperforming it on several of them while requiring the fraction of the time to run. Moreover, the Tail-probability thresholding also showed promising results on a few datasets and algorithms. This can be used for anomaly detection runs in real-word unsupervised scenarios.

The experiments have validated the hypothesis, simple probabilistic anomaly detectors can replace more complex Deep Learning approaches. The only datasets where the Autoencoder performed substantially better were the ones with one-hot-encoded values that require complex models to learn the intricate details and connections among them. The conclusion of the evaluation is that if computation resources are not limited, a good anomaly detection result can be achieved by the Autoencoder model, however, should the available resources be constrained, for instance in the Edge devices of the IIoT Architecture, probabilistic or statistical methods can be used for a prompt, local anomaly detection.

I also investigated the possibility of the tested algorithms to form a potential ensemble of approaches in order to further improve detection performance. I plan to continue this work in the future. In addition, to automatize detection a method needs to be developed to automatically set the the threshold for the Tail-probability thresholding. Although DSPOT did not have good performance in this study for this particular task, I only carried out a limited number of experiments. A more extensive study with rigorous parameter tuning ought to be conducted. As future work I plan to explore this in detail.

Acknowledgements

I would like to express my deepest gratitude to my two thesis advisors, Dr.Adrián Pekár and Dr. Károly Farkas for guiding me along the way, providing helpful feedback throughout the entirety of my thesis project and for having a flexible attitude towards me.

I would also like to convey my earnest gratefulness towards my family for supporting me in every aspect of my life and encouraging me during my studies.

List of Figures

List of Tables

Bibliography

[1] Charu C. Aggarwal. *Outlier Analysis*. Springer, Cham, 2017. ISBN 978-3-319-47578-3. DOI: https://doi.org/10.1007/978-3-319-47578-3.

[2] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017. ISSN 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2017.04.070. URL https://www.sciencedirect.com/science/article/pii/S0925231217309864. Online Real-Time Learning Strategies for Data Streams.

[3] Amossys-Team. Spot source-code, 2017. URL https://github.com/Amossys-team/SPOT.

[4] Jan Beirlant, Yuri Goegebeur, Johan Segers, and Jozef L. Teugels. *Statistics of Extremes: Theory and Applications*. John Wiley & Sons, Ltd, 2004. ISBN 0471976474.

[5] Jan Beirlant, Yuri Goegebeur, Johan Segers, Jozef L. Teugels, Daniel De Waal (Contributions by), and Chris Ferro (Contributions by). *Statistics of Extremes: Theory and Applications*. John Wiley & Sons, Ltd, 2006. ISBN 978-0-470-01237-6.

[6] Paul Boniol, Michele Linardi, Federico Roncallo, Themis Palpanas, Mohammed Meftah, and Emmanuel Remy. Unsupervised and scalable subsequence anomaly detection in large data series. *The VLDB Journal*, 03 2021. ISSN 0949-877X. DOI: 10.1007/s00778-021-00655-8. URL https://doi.org/10.1007/s00778-021-00655-8.

[7] Mohammad Braei and Sebastian Wagner. Anomaly detection in univariate time-series: A survey on the state-of-the-art, 2020.

[8] Leo Breiman. Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3):199 – 231, 2001. DOI: 10.1214/ss/1009213726. URL https://doi.org/10.1214/ss/1009213726.

[9] Alberto Diez-Olivan, Javier Del Ser, Diego Galar, and Basilio Sierra. Data fusion and machine learning for industrial prognosis: Trends and perspectives towards industry 4.0. *Information Fusion*, 50:92–111, 2019. ISSN 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2018.10.005. URL https://www.sciencedirect.com/science/article/pii/S1566253518304706.

[10] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231, 1996.

[11] Astha Garg, Wenyu Zhang, Jules Samaran, Ramasamy Savitha, and Chuan-Sheng Foo. An evaluation of anomaly detection and diagnosis in multivariate time series. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–10, 2021. DOI: 10.1109/TNNLS.2021.3105827.

[12] Felix A. Gers, Douglas Eck, and Jürgen Schmidhuber. Applying lstm to time series predictable through time-window approaches. In Georg Dorffner, Horst Bischof, and Kurt Hornik, editors, *Artificial Neural Networks — ICANN 2001*, pages 669–676, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-44668-2.

[13] Jonathan Goh, Sridhar Adepu, Khurum Junejo, and Aditya Mathur. A dataset to support research in the design of secure water treatment systems. 10 2016.

[14] Klaus Greff, Rupesh K. Srivastava, Jan Koutnik, Bas R. Steunebrink, and Jurgen Schmidhuber. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, Oct 2017. ISSN 2162-2388. DOI: 10.1109/tnnls.2016.2582924. URL http://dx.doi.org/10.1109/TNNLS.2016.2582924.

[15] Scott D. Grimshaw. Computing maximum likelihood estimates for the generalized pareto distribution. *Technometrics*, 35(2):185–191, 1993. DOI: 10.1080/00401706.1993.10485040. URL https://www.tandfonline.com/doi/abs/10.1080/00401706.1993.10485040.

[16] D. Hawkins. *Identification of Outliers*. Springer, Dordrecht, 1980. ISBN 978-94-015-3994-4. DOI: https://doi.org/10.1007/978-94-015-3994-4.

[17] Jordan Hochenbaum, Owen S. Vallis, and Arun Kejariwal. Automatic Anomaly Detection in the Cloud Via Statistical Learning, 2017.

[18] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 387–395, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355520. DOI: 10.1145/3219819.3219845. URL https://doi.org/10.1145/3219819.3219845.

[19] J. Stuart Hunter. The exponentially weighted moving average. *Journal of Quality Technology*, 18(4):203–210, 1986. DOI: 10.1080/00224065.1986.11979014. URL https://doi.org/10.1080/00224065.1986.11979014.

[20] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice, 2nd edition*. OTexts: Melbourne, Australia., 2018. URL OTexts.com/fpp2.

[21] IBM. Ibm spss modeler help | time series outliers, 2014. URL https://www.ibm.com/docs/en/spss-modeler/18.3.0?topic=series-outliers. Accessed: 2021-08-12.

[22] James Pickands III. Statistical Inference Using Extreme Order Statistics. *The Annals of Statistics*, 3(1):119 – 131, 1975. DOI: 10.1214/aos/1176343003. URL https://doi.org/10.1214/aos/1176343003.

[23] iTrust, Centre for Research in Cyber Security, Singapore University of Technology and Design. iTrust Datasets, 2021. URL https://itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info/.

[24] Iurii D. Katser and Vyacheslav O. Kozitsin. Skoltech anomaly benchmark (skab). https://www.kaggle.com/dsv/1693952, 2020.

[25] Keras. Keras Pyhton Library, 2021. URL https://keras.io/.

[26] Denis Kwiatkowski, Peter C.B. Phillips, Peter Schmidt, and Yongcheol Shin. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics*, 54(1):159–178, 1992. ISSN 0304-4076. DOI: https://doi.org/10.1016/0304-4076(92)90104-Y. URL https://www.sciencedirect.com/science/article/pii/030440769290104Y.

[27] A. Lavin and S. Ahmad. Evaluating Real-Time Anomaly Detection Algorithms – The Numenta Anomaly Benchmark. In *Proceedings of the IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 38–44, 2015. DOI: 10.1109/ICMLA.2015.141.

[28] Jia Li, Shimin Di, Yanyan Shen, and Lei Chen. Fluxev: A fast and effective unsupervised framework for time-series anomaly detection. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 824–832, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450382977. DOI: 10.1145/3437963.3441823. URL https://doi.org/10.1145/3437963.3441823.

[29] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Lstm-based encoder-decoder for multi-sensor anomaly detection, 2016.

[30] Roy A Maxion and Rachel R Roberts. *Proper use of ROC curves in Intrusion/Anomaly Detection*. University of Newcastle upon Tyne, Computing Science Tyne, UK, 2004.

[31] Numenta. Numenta Anomaly Bencmark Datasets, 2016. URL https://github.com/numenta/NAB/tree/master/data.

[32] Numenta, Inc. NAB: Numenta Anomaly Benchmark [Online code repository], 2017. URL https://github.com/numenta/NAB.

[33] Keith Ord. Outliers in statistical data: V. barnett and t. lewis, 1994, 3rd edition, (john wiley & sons, chichester), 584 pp., [uk pound]55.00, isbn 0-471-93094-6. *International Journal of Forecasting*, 12(1):175–176, 1996. URL https://EconPapers.repec.org/RePEc:eee:intfor:v:12:y:1996:i:1:p:175-176.

[34] Julien Polge, Jérémy Robert, and Yves Le Traon. A case driven study of the use of time series classification for flexibility in industry 4.0. *Sensors*, 20:7273, 12 2020. DOI: 10.3390/s20247273.

[35] David Professor Hardt. Mill talk: What is industry 4.0 and how did we get here? with mit professor david hardt, 2020. URL https://youtu.be/ttfMEXGdh1s.

[36] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *WIREs Data Mining and Knowledge Discovery*, 8(4):e1249, 2018. DOI: https://doi.org/10.1002/widm.1249. URL https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1249.

[37] Skipper Seabold and Josef Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010. URL https://www.statsmodels.org/dev/examples/notebooks/generated/autoregressions.html.

[38] Alban Siffer, Pierre-Alain Fouque, Alexandre Termier, and Christine Largouet. Anomaly detection in streams with extreme value theory. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 1067–1075, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450348874. DOI: 10.1145/3097983.3098144. URL https://doi.org/10.1145/3097983.3098144.

[39] Nidhi Singh and Craig Olinsky. Demystifying numenta anomaly benchmark. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1570–1577, 2017. DOI: 10.1109/IJCNN.2017.7966038.

[40] 4.0 Solutions, 2021. URL https://www.iiot.university/.

[41] statsmodels.org. Statsmodels Python Library, 2021. URL https://www.statsmodels.org/dev/examples/notebooks/generated/autoregressions.html.

[42] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 2828–2837, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. DOI: 10.1145/3292500.3330672. URL https://doi.org/10.1145/3292500.3330672.

[43] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Server machine dataset, 2019. URL https://github.com/NetManAIOps/OmniAnomaly/tree/master/ServerMachineDataset.

[44] Daniel Vajda, Adrian Pekar, and Karoly Farkas. Towards Machine Learning-based Anomaly Detection on Time-Series Data. *Infocommunications Journal*, XIII(1):36–44, 3 2021. DOI: 10.36244/ICJ.2021.1.5.

[45] Ke Wang. Fluxev source-code, 2021. URL https://github.com/cnKeWang/FluxEV_reproduce.

[46] Xing Wang, Jessica Lin, Nital Patel, and Martin Braun. A self-learning and online algorithm for time series anomaly detection, with application in cpu manufacturing. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, page 1823–1832, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450340731. DOI: 10.1145/2983323.2983344. URL https://doi.org/10.1145/2983323.2983344.

[47] Xing Wang, Jessica Lin, Nital Patel, and Martin Braun. Exact variable-length anomaly detection algorithm for univariate and multivariate time series. *Data Mining and Knowledge Discovery*, 32(6):1806–1844, 10 2018. ISSN 1573-756X. DOI: 10.1007/s10618-018-0569-7. URL https://doi.org/10.1007/s10618-018-0569-7.

[48] Yahoo! Webscope. Yahoo! Webscope dataset ydata-labeled-time-series-anomalies-v1_0, 2015. URL http://labs.yahoo.com/Academic_Relations.

Appendix

Table A.1: Evaluation metrics on NAB datasets on a per-dataset basis

Algorithm	Dataset	Precision	Recall	F-score	AUC
AR_comb_3sigma	ambient_temperature_system_failure.csv	0.375	0.500	0.429	0.707
AR_comb_F-score	ambient_temperature_system_failure.csv	0.000	0.000	0.000	0.651
AR_diff_3sigma	ambient_temperature_system_failure.csv	0.000	0.000	0.000	0.483
AR_diff_F-score	ambient_temperature_system_failure.csv	0.000	0.000	0.000	0.651
AR_dynamic_3sigma	ambient_temperature_system_failure.csv	0.326	0.500	0.395	0.551
AR_dynamic_F-score	ambient_temperature_system_failure.csv	0.007	0.500	0.013	0.614
Autoencoder_3sigma	ambient_temperature_system_failure.csv	0.429	0.500	0.462	0.580
Autoencoder_best_F-score	ambient_temperature_system_failure.csv	0.004	0.500	0.008	0.522
FluxEV1e-1	ambient_temperature_system_failure.csv	0.004	1.000	0.008	0.507
FluxEV1e-2	ambient_temperature_system_failure.csv	0.004	1.000	0.008	0.507
FluxEV1e-3	ambient_temperature_system_failure.csv	0.004	1.000	0.008	0.507
SPOT1e-1	ambient_temperature_system_failure.csv	0.010	1.000	0.020	0.878
SPOT1e-2	ambient_temperature_system_failure.csv	0.010	1.000	0.020	0.878
SPOT1e-3	ambient_temperature_system_failure.csv	0.010	1.000	0.020	0.878
AR_comb_3sigma	cpu_utilization_asg_misconfiguration.csv	0.026	0.500	0.049	0.779
AR_comb_F-score	cpu_utilization_asg_misconfiguration.csv	0.000	0.000	0.000	0.650
AR_diff_3sigma	cpu_utilization_asg_misconfiguration.csv	0.112	1.000	0.201	0.992
AR_diff_F-score	cpu_utilization_asg_misconfiguration.csv	0.000	0.000	0.000	0.903
AR_dynamic_3sigma	cpu_utilization_asg_misconfiguration.csv	0.000	0.000	0.000	0.772
AR_dynamic_F-score	cpu_utilization_asg_misconfiguration.csv	0.008	1.000	0.015	0.906
Autoencoder_3sigma	cpu_utilization_asg_misconfiguration.csv	0.165	1.000	0.283	0.996
Autoencoder_best_F-score	cpu_utilization_asg_misconfiguration.csv	0.006	1.000	0.011	0.993
FluxEV1e-1	cpu_utilization_asg_misconfiguration.csv	0.002	1.000	0.003	0.881
FluxEV1e-2	cpu_utilization_asg_misconfiguration.csv	0.002	1.000	0.003	0.881
FluxEV1e-3	cpu_utilization_asg_misconfiguration.csv	0.002	1.000	0.003	0.881
SPOT1e-1	cpu_utilization_asg_misconfiguration.csv	0.004	1.000	0.008	0.832
SPOT1e-2	cpu_utilization_asg_misconfiguration.csv	0.004	1.000	0.008	0.832
SPOT1e-3	cpu_utilization_asg_misconfiguration.csv	0.004	1.000	0.008	0.832
AR_comb_3sigma	ec2_request_latency_system_failure.csv	0.726	1.000	0.841	0.999
AR_comb_F-score	ec2_request_latency_system_failure.csv	0.000	0.000	0.000	0.652
AR_diff_3sigma	ec2_request_latency_system_failure.csv	0.714	1.000	0.833	0.999
AR_diff_F-score	ec2_request_latency_system_failure.csv	0.804	1.000	0.891	0.999
AR_dynamic_3sigma	ec2_request_latency_system_failure.csv	0.682	0.667	0.674	0.898
AR_dynamic_F-score	ec2_request_latency_system_failure.csv	0.052	1.000	0.100	0.969
Autoencoder_3sigma	ec2_request_latency_system_failure.csv	0.227	0.333	0.270	0.830
Autoencoder_best_F-score	ec2_request_latency_system_failure.csv	0.026	1.000	0.050	0.907
FluxEV1e-1	ec2_request_latency_system_failure.csv	0.011	1.000	0.023	0.838
FluxEV1e-2	ec2_request_latency_system_failure.csv	0.011	1.000	0.023	0.838
FluxEV1e-3	ec2_request_latency_system_failure.csv	0.011	1.000	0.023	0.838
SPOT1e-1	ec2_request_latency_system_failure.csv	0.064	1.000	0.120	1.000
SPOT1e-2	ec2_request_latency_system_failure.csv	0.064	1.000	0.120	1.000
SPOT1e-3	ec2_request_latency_system_failure.csv	0.064	1.000	0.120	1.000
AR_comb_3sigma	machine_temperature_system_failure.csv	0.000	0.000	0.000	0.312
AR_comb_F-score	machine_temperature_system_failure.csv	0.000	0.000	0.000	0.253
AR_diff_3sigma	machine_temperature_system_failure.csv	0.122	0.250	0.164	0.705
AR_diff_F-score	machine_temperature_system_failure.csv	0.000	0.000	0.000	0.503
AR_dynamic_3sigma	machine_temperature_system_failure.csv	0.096	0.250	0.138	0.606
AR_dynamic_F-score	machine_temperature_system_failure.csv	0.005	0.750	0.010	0.648
Autoencoder_3sigma	machine_temperature_system_failure.csv	0.000	0.000	0.000	0.290
Autoencoder_best_F-score	machine_temperature_system_failure.csv	0.001	0.250	0.003	0.285
FluxEV1e-1	machine_temperature_system_failure.csv	0.003	1.000	0.005	0.802

Continued on next page

Table A.1: Evaluation metrics on NAB datasets on a per-dataset basis

Algorithm	Dataset	Precision	Recall	F-score	AUC
FluxEV1e-2	machine_temperature_system_failure.csv	0.003	1.000	0.005	0.802
FluxEV1e-3	machine_temperature_system_failure.csv	0.003	1.000	0.005	0.802
SPOT1e-1	machine_temperature_system_failure.csv	0.003	1.000	0.006	0.815
SPOT1e-2	machine_temperature_system_failure.csv	0.003	1.000	0.006	0.815
SPOT1e-3	machine_temperature_system_failure.csv	0.003	1.000	0.006	0.815
AR_comb_3sigma	nyc_taxi.csv	0.000	0.000	0.000	0.667
AR_comb_F-score	nyc_taxi.csv	0.000	0.000	0.000	0.651
AR_diff_3sigma	nyc_taxi.csv	0.435	0.400	0.417	0.662
AR_diff_F-score	nyc_taxi.csv	0.000	0.000	0.000	0.505
AR_dynamic_3sigma	nyc_taxi.csv	0.000	0.000	0.000	0.635
AR_dynamic_F-score	nyc_taxi.csv	0.012	0.600	0.024	0.645
Autoencoder_3sigma	nyc_taxi.csv	0.000	0.000	0.000	0.353
Autoencoder_best_F-score	nyc_taxi.csv	0.000	0.000	0.000	0.555
FluxEV1e-1	nyc_taxi.csv	0.062	0.200	0.095	0.751
FluxEV1e-2	nyc_taxi.csv	0.062	0.200	0.095	0.751
FluxEV1e-3	nyc_taxi.csv	0.062	0.200	0.095	0.751
SPOT1e-1	nyc_taxi.csv	0.036	0.200	0.061	0.577
SPOT1e-2	nyc_taxi.csv	0.036	0.200	0.061	0.577
SPOT1e-3	nyc_taxi.csv	0.036	0.200	0.061	0.577
AR_comb_3sigma	rogue_agent_key_hold.csv	0.000	0.000	0.000	0.724
AR_comb_F-score	rogue_agent_key_hold.csv	0.000	0.000	0.000	0.652
AR_diff_3sigma	rogue_agent_key_hold.csv	0.000	0.000	0.000	0.691
AR_diff_F-score	rogue_agent_key_hold.csv	0.017	1.000	0.033	0.527
AR_dynamic_3sigma	rogue_agent_key_hold.csv	0.000	0.000	0.000	0.462
AR_dynamic_F-score	rogue_agent_key_hold.csv	0.000	0.000	0.000	0.509
Autoencoder_3sigma	rogue_agent_key_hold.csv	0.000	0.000	0.000	0.417
Autoencoder_best_F-score	rogue_agent_key_hold.csv	0.000	0.000	0.000	0.504
FluxEV1e-1	rogue_agent_key_hold.csv	0.017	1.000	0.033	0.528
FluxEV1e-2	rogue_agent_key_hold.csv	0.017	1.000	0.033	0.528
FluxEV1e-3	rogue_agent_key_hold.csv	0.017	1.000	0.033	0.528
SPOT1e-1	rogue_agent_key_hold.csv	0.017	1.000	0.034	0.317
SPOT1e-2	rogue_agent_key_hold.csv	0.017	1.000	0.034	0.317
SPOT1e-3	rogue_agent_key_hold.csv	0.017	1.000	0.034	0.317
AR_comb_3sigma	rogue_agent_key_updown.csv	0.000	0.000	0.000	0.587
AR_comb_F-score	rogue_agent_key_updown.csv	0.000	0.000	0.000	0.651
AR_diff_3sigma	rogue_agent_key_updown.csv	0.000	0.000	0.000	0.500
AR_diff_F-score	rogue_agent_key_updown.csv	0.000	0.000	0.000	0.507
AR_dynamic_3sigma	rogue_agent_key_updown.csv	0.000	0.000	0.000	0.498
AR_dynamic_F-score	rogue_agent_key_updown.csv	0.009	0.500	0.018	0.590
Autoencoder_3sigma	rogue_agent_key_updown.csv	0.000	0.000	0.000	0.510
Autoencoder_best_F-score	rogue_agent_key_updown.csv	0.000	0.000	0.000	0.348
FluxEV1e-1	rogue_agent_key_updown.csv	0.000	0.000	0.000	0.493
FluxEV1e-2	rogue_agent_key_updown.csv	0.000	0.000	0.000	0.493
FluxEV1e-3	rogue_agent_key_updown.csv	0.000	0.000	0.000	0.493
SPOT1e-1	rogue_agent_key_updown.csv	0.006	1.000	0.011	0.456
SPOT1e-2	rogue_agent_key_updown.csv	0.006	1.000	0.011	0.456
SPOT1e-3	rogue_agent_key_updown.csv	0.006	1.000	0.011	0.456

Table A.2: Evaluation metrics on SWAT dataset with Gauss-D and Gauss-D-DK aggregation with Best-F-score and Tail-probability thresholding

Algorithm	Aggregation	Thresholding method	Point-wise Precision	Normalized Precision	Recall	F-score	F_c-score	AUC
AR	Gauss-D	best_f1_test	0.651	**0.998**	0.457	0.627	0.537	0.542
AR	Gauss-DK	tail_prob_5	0.65	0.8	0.171	0.282	0.271	0.757
AR	Gauss-DK	tail_prob_3	0.591	0.808	0.457	0.584	0.515	0.757
AR	Gauss-DK	tail_prob_2	0.52	0.72	0.6	0.655	0.557	0.757
AR	Gauss-DK	tail_prob_1	0.169	0.227	0.829	0.357	0.28	0.757
AR	Gauss-DK	best_f1_test	0.479	0.636	0.714	0.673	0.573	0.757
AR	Gauss-DK	tail_prob_4	0.607	0.802	0.257	0.389	0.361	0.757
AR	Gauss-D	tail_prob_4	0.123	0.853	0.914	0.883	0.216	0.542
AR	Gauss-D	tail_prob_3	0.113	0.821	0.914	0.865	0.201	0.542
AR	Gauss-D	tail_prob_2	0.101	0.77	0.943	0.847	0.182	0.542
AR	Gauss-D	tail_prob_1	0.072	0.406	**0.971**	0.572	0.133	0.542
AR	Gauss-D	tail_prob_5	0.136	0.875	0.886	0.88	0.236	0.542
AR_diff	Gauss-DK	tail_prob_4	0.603	0.783	0.229	0.354	0.332	0.761
AR_diff	Gauss-D	tail_prob_2	0.088	0.788	0.943	0.859	0.161	0.546
AR_diff	Gauss-D	tail_prob_3	0.089	0.833	0.914	0.872	0.163	0.546
AR_diff	Gauss-D	tail_prob_4	0.093	0.862	0.914	**0.887**	0.169	0.546
AR_diff	Gauss-D	tail_prob_5	0.103	0.881	0.886	0.884	0.184	0.546
AR_diff	Gauss-DK	best_f1_test	0.493	0.649	0.686	0.667	0.573	0.761
AR_diff	Gauss-DK	tail_prob_1	0.165	0.223	0.829	0.351	0.275	0.761
AR_diff	Gauss-DK	tail_prob_2	0.529	0.729	0.6	0.658	0.562	0.761
AR_diff	Gauss-DK	tail_prob_3	0.576	0.785	0.4	0.53	0.472	0.761
AR_diff	Gauss-D	tail_prob_1	0.075	0.435	**0.971**	0.601	0.14	0.546
AR_diff	Gauss-DK	tail_prob_5	0.651	0.801	0.171	0.282	0.271	0.761
AR_diff	Gauss-D	best_f1_test	0.562	0.997	0.543	0.703	0.553	0.546
BiDSPOT1e-1	Gauss-DK	tail_prob_5	0.544	0.659	0.371	0.475	0.441	0.801
BiDSPOT1e-1	Gauss-DK	tail_prob_3	0.525	0.615	0.4	0.485	0.454	0.801
BiDSPOT1e-1	Gauss-DK	tail_prob_2	0.48	0.519	0.429	0.469	0.453	0.801
BiDSPOT1e-1	Gauss-DK	tail_prob_1	0.083	0.09	0.914	0.164	0.152	0.801
BiDSPOT1e-1	Gauss-DK	best_f1_test	0.477	0.54	0.486	0.512	0.481	0.801
BiDSPOT1e-1	Gauss-DK	tail_prob_4	0.531	0.631	0.371	0.468	0.437	0.801
BiDSPOT1e-1	Gauss-D	tail_prob_4	0.543	0.726	0.457	0.561	0.496	0.773
BiDSPOT1e-1	Gauss-D	tail_prob_3	0.429	0.615	0.514	0.56	0.468	0.773
BiDSPOT1e-1	Gauss-D	tail_prob_2	0.326	0.504	0.657	0.571	0.436	0.773
BiDSPOT1e-1	Gauss-D	tail_prob_1	0.092	0.105	0.914	0.189	0.167	0.773
BiDSPOT1e-1	Gauss-D	best_f1_test	**0.724**	0.869	0.429	0.574	0.538	0.773
BiDSPOT1e-1	Gauss-D	tail_prob_5	0.721	0.867	0.429	0.573	0.538	0.773
BiDSPOT1e-10	Gauss-DK	tail_prob_5	0.479	0.578	0.457	0.511	0.468	0.806
BiDSPOT1e-10	Gauss-DK	tail_prob_3	0.394	0.5	0.571	0.534	0.466	0.806
BiDSPOT1e-10	Gauss-DK	tail_prob_2	0.383	0.422	0.686	0.523	0.491	0.806
BiDSPOT1e-10	Gauss-DK	tail_prob_1	0.091	0.093	0.829	0.167	0.163	0.806
BiDSPOT1e-10	Gauss-DK	best_f1_test	0.428	0.497	0.657	0.566	0.519	0.806
BiDSPOT1e-10	Gauss-D	tail_prob_5	0.546	0.724	0.543	0.62	0.545	0.771
BiDSPOT1e-10	Gauss-D	tail_prob_4	0.453	0.641	0.6	0.62	0.516	0.771
BiDSPOT1e-10	Gauss-D	tail_prob_3	0.349	0.555	0.771	0.645	0.481	0.771
BiDSPOT1e-10	Gauss-D	tail_prob_2	0.261	0.365	0.829	0.506	0.397	0.771
BiDSPOT1e-10	Gauss-DK	tail_prob_4	0.435	0.52	0.457	0.486	0.446	0.806
BiDSPOT1e-10	Gauss-D	best_f1_test	0.573	0.752	0.543	0.631	0.557	0.771
BiDSPOT1e-10	Gauss-D	tail_prob_1	0.122	0.143	0.914	0.247	0.215	0.771
BiDSPOT1e-3	Gauss-DK	tail_prob_5	0.533	0.641	0.371	0.47	0.438	0.787
BiDSPOT1e-3	Gauss-D	best_f1_test	0.66	0.838	0.514	0.637	0.578	0.768
BiDSPOT1e-3	Gauss-D	tail_prob_1	0.118	0.145	0.914	0.251	0.209	0.768
BiDSPOT1e-3	Gauss-D	tail_prob_2	0.211	0.356	0.8	0.493	0.333	0.768
BiDSPOT1e-3	Gauss-D	tail_prob_3	0.284	0.502	0.657	0.569	0.397	0.768
BiDSPOT1e-3	Gauss-DK	tail_prob_4	0.497	0.598	0.371	0.458	0.425	0.787
BiDSPOT1e-3	Gauss-DK	best_f1_test	0.608	0.755	0.371	0.498	0.461	0.787
BiDSPOT1e-3	Gauss-DK	tail_prob_1	0.088	0.097	0.857	0.175	0.159	0.787
BiDSPOT1e-3	Gauss-DK	tail_prob_2	0.388	0.427	0.543	0.478	0.453	0.787
BiDSPOT1e-3	Gauss-DK	tail_prob_3	0.389	0.478	0.429	0.452	0.408	0.787
BiDSPOT1e-3	Gauss-D	tail_prob_4	0.506	0.716	0.543	0.618	0.524	0.768
BiDSPOT1e-3	Gauss-D	tail_prob_5	0.669	0.837	0.486	0.615	0.563	0.768
BiDSPOT1e-5	Gauss-D	tail_prob_1	0.126	0.148	0.914	0.255	0.222	0.79
BiDSPOT1e-5	Gauss-DK	tail_prob_5	0.478	0.594	0.429	0.498	0.452	0.82
BiDSPOT1e-5	Gauss-DK	tail_prob_4	0.447	0.559	0.457	0.503	0.452	0.82
BiDSPOT1e-5	Gauss-DK	tail_prob_3	0.431	0.531	0.543	0.537	0.48	0.82
BiDSPOT1e-5	Gauss-DK	tail_prob_2	0.396	0.442	0.686	0.537	0.502	0.82
BiDSPOT1e-5	Gauss-D	best_f1_test	0.549	0.761	0.629	0.688	0.586	0.79
BiDSPOT1e-5	Gauss-DK	best_f1_test	0.443	0.517	0.657	0.579	0.529	0.82
BiDSPOT1e-5	Gauss-D	tail_prob_5	0.582	0.773	0.543	0.638	0.562	0.79
BiDSPOT1e-5	Gauss-D	tail_prob_4	0.524	0.732	0.657	0.693	0.583	0.79
BiDSPOT1e-5	Gauss-D	tail_prob_3	0.362	0.55	0.714	0.621	0.48	0.79
BiDSPOT1e-5	Gauss-D	tail_prob_2	0.271	0.373	0.829	0.514	0.408	0.79
BiDSPOT1e-5	Gauss-DK	tail_prob_1	0.099	0.108	0.886	0.192	0.179	0.82
FluxEV1e-1	Gauss-D	best_f1_test	0.418	0.992	0.543	0.702	0.473	0.564
FluxEV1e-1	Gauss-D	tail_prob_1	0.119	0.314	**0.971**	0.474	0.213	0.564
FluxEV1e-1	Gauss-D	tail_prob_2	0.154	0.514	0.914	0.658	0.264	0.564
FluxEV1e-1	Gauss-D	tail_prob_3	0.168	0.613	0.914	0.734	0.283	0.564
FluxEV1e-1	Gauss-D	tail_prob_4	0.147	0.663	0.914	0.768	0.253	0.564
FluxEV1e-1	Gauss-DK	tail_prob_1	0.094	0.118	0.943	0.209	0.172	0.769
FluxEV1e-1	Gauss-DK	best_f1_test	0.463	0.612	0.629	0.62	0.533	0.769
FluxEV1e-1	Gauss-DK	tail_prob_2	0.34	0.436	0.8	0.564	0.477	0.769
FluxEV1e-1	Gauss-DK	tail_prob_3	0.411	0.527	0.686	0.596	0.514	0.769
FluxEV1e-1	Gauss-DK	tail_prob_5	0.465	0.597	0.514	0.553	0.488	0.769
FluxEV1e-1	Gauss-D	tail_prob_5	0.125	0.704	0.914	0.796	0.219	0.564
FluxEV1e-1	Gauss-DK	tail_prob_4	0.448	0.581	0.629	0.604	0.523	0.769
FluxEV1e-3	Gauss-DK	tail_prob_5	0.446	0.581	0.486	0.529	0.465	0.761
FluxEV1e-3	Gauss-D	tail_prob_5	0.186	0.704	0.886	0.784	0.308	0.567
FluxEV1e-3	Gauss-D	best_f1_test	0.364	0.989	0.657	0.79	0.469	0.567
FluxEV1e-3	Gauss-D	tail_prob_1	0.093	0.232	**0.971**	0.374	0.17	0.567

Continued on next page

Table A.2: Evaluation metrics on SWAT dataset with Gauss-D and Gauss-D-DK aggregation with Best-F-score and Tail-probability thresholding

Algorithm	Aggregation	Thresholding method	Point-wise Precision	Normalized Precision	Recall	F-score	F_c-score	AUC
FluxEV1e-3	Gauss-D	tail_prob_2	0.166	0.468	0.943	0.625	0.283	0.567
FluxEV1e-3	Gauss-D	tail_prob_3	0.177	0.586	0.914	0.714	0.297	0.567
FluxEV1e-3	Gauss-D	tail_prob_4	0.189	0.652	0.886	0.751	0.311	0.567
FluxEV1e-3	Gauss-DK	best_f1_test	0.456	0.607	0.629	0.618	0.529	0.761
FluxEV1e-3	Gauss-DK	tail_prob_1	0.075	0.092	0.943	0.167	0.139	0.761
FluxEV1e-3	Gauss-DK	tail_prob_2	0.32	0.41	0.8	0.542	0.457	0.761
FluxEV1e-3	Gauss-DK	tail_prob_3	0.388	0.515	0.743	0.608	0.51	0.761
FluxEV1e-3	Gauss-DK	tail_prob_4	0.435	0.566	0.629	0.596	0.514	0.761
FluxEV1e-5	Gauss-DK	tail_prob_5	0.467	0.584	0.486	0.53	0.476	0.766
FluxEV1e-5	Gauss-DK	tail_prob_4	0.443	0.579	0.629	0.603	0.52	0.766
FluxEV1e-5	Gauss-D	best_f1_test	0.357	0.988	0.686	0.81	0.47	0.593
FluxEV1e-5	Gauss-D	tail_prob_1	0.104	0.259	**0.971**	0.408	0.188	0.593
FluxEV1e-5	Gauss-D	tail_prob_2	0.18	0.499	0.943	0.652	0.303	0.593
FluxEV1e-5	Gauss-D	tail_prob_3	0.17	0.604	0.914	0.727	0.287	0.593
FluxEV1e-5	Gauss-D	tail_prob_4	0.177	0.659	0.886	0.756	0.295	0.593
FluxEV1e-5	Gauss-D	tail_prob_5	0.171	0.705	0.886	0.785	0.286	0.593
FluxEV1e-5	Gauss-DK	best_f1_test	0.458	0.615	0.629	0.622	0.53	0.766
FluxEV1e-5	Gauss-DK	tail_prob_1	0.079	0.096	0.943	0.175	0.146	0.766
FluxEV1e-5	Gauss-DK	tail_prob_2	0.329	0.42	0.8	0.551	0.467	0.766
FluxEV1e-5	Gauss-DK	tail_prob_3	0.392	0.513	0.714	0.597	0.506	0.766
UnivarAutoencoder	Gauss-DK	tail_prob_3	0.561	0.658	0.514	0.577	0.537	**0.824**
UnivarAutoencoder	Gauss-DK	tail_prob_2	0.527	0.59	0.629	0.609	0.574	**0.824**
UnivarAutoencoder	Gauss-DK	tail_prob_1	0.087	0.095	0.943	0.173	0.159	**0.824**
UnivarAutoencoder	Gauss-DK	best_f1_test	0.56	0.644	0.629	0.636	0.593	**0.824**
UnivarAutoencoder	Gauss-D	tail_prob_5	0.513	0.802	0.857	0.829	0.642	0.78
UnivarAutoencoder	Gauss-D	best_f1_test	0.716	0.936	0.686	0.792	**0.701**	0.78
UnivarAutoencoder	Gauss-D	tail_prob_3	0.372	0.622	0.886	0.731	0.523	0.78
UnivarAutoencoder	Gauss-D	tail_prob_2	0.315	0.499	0.886	0.638	0.465	0.78
UnivarAutoencoder	Gauss-D	tail_prob_1	0.136	0.183	0.943	0.307	0.238	0.78
UnivarAutoencoder	Gauss-DK	tail_prob_4	0.588	0.679	0.457	0.546	0.514	**0.824**
UnivarAutoencoder	Gauss-D	tail_prob_4	0.429	0.712	0.857	0.778	0.572	0.78
UnivarAutoencoder	Gauss-DK	tail_prob_5	0.606	0.697	0.429	0.531	0.502	**0.824**

Table A.3: Evaluation metrics on SKAB dataset with Gauss-D and Gauss-D-DK aggregation with Best-F-score and Tail-probability thresholding

Algorithm	Aggregation	Thresholding method	Point-wise Precision	Normalized Precision	Recall	F-score	F_c-score	AUC
AR	Gauss-D	best_f1_test	0.372	0.62	**1.0**	0.765	0.542	0.504
AR	Gauss-DK	tail_prob_4	0.068	0.903	0.029	0.057	0.041	0.508
AR	Gauss-DK	tail_prob_3	0.158	0.96	0.118	0.21	0.135	0.508
AR	Gauss-DK	tail_prob_2	0.279	0.976	0.441	0.608	0.342	0.508
AR	Gauss-DK	tail_prob_1	0.366	0.69	**1.0**	0.817	0.536	0.508
AR	Gauss-DK	best_f1_test	0.374	0.437	**1.0**	0.609	0.545	0.508
AR	Gauss-DK	tail_prob_5	0.034	0.932	0.029	0.057	0.032	0.508
AR	Gauss-D	tail_prob_4	0.343	0.992	0.441	0.611	0.386	0.504
AR	Gauss-D	tail_prob_3	0.322	0.99	0.471	0.638	0.383	0.504
AR	Gauss-D	tail_prob_2	0.301	0.974	0.765	0.857	0.432	0.504
AR	Gauss-D	tail_prob_1	0.37	0.659	**1.0**	0.794	0.54	0.504
AR	Gauss-D	tail_prob_5	0.349	**0.993**	0.441	0.611	0.39	0.504
AR_diff	Gauss-DK	tail_prob_5	0.0	0.0	0.0	0.0	0.0	0.502
AR_diff	Gauss-D	tail_prob_2	0.283	0.974	0.765	0.857	0.414	0.501
AR_diff	Gauss-D	tail_prob_3	0.318	0.99	0.471	0.638	0.38	0.501
AR_diff	Gauss-D	tail_prob_4	0.338	0.992	0.441	0.611	0.383	0.501
AR_diff	Gauss-D	tail_prob_5	0.355	**0.993**	0.441	0.611	0.393	0.501
AR_diff	Gauss-DK	best_f1_test	0.372	0.414	**1.0**	0.585	0.542	0.502
AR_diff	Gauss-DK	tail_prob_1	0.363	0.694	**1.0**	0.82	0.533	0.502
AR_diff	Gauss-DK	tail_prob_2	0.284	0.977	0.441	0.608	0.346	0.502
AR_diff	Gauss-DK	tail_prob_3	0.16	0.961	0.118	0.21	0.136	0.502
AR_diff	Gauss-DK	tail_prob_4	0.068	0.903	0.029	0.057	0.041	0.502
AR_diff	Gauss-D	tail_prob_1	0.366	0.661	**1.0**	0.796	0.536	0.501
AR_diff	Gauss-D	best_f1_test	0.371	0.414	**1.0**	0.586	0.541	0.501
AR_dynamic	Gauss-D	tail_prob_2	0.328	0.931	0.971	0.95	0.49	0.499
AR_dynamic	Gauss-D	tail_prob_1	0.365	0.567	**1.0**	0.723	0.535	0.499
AR_dynamic	Gauss-DK	tail_prob_5	0.205	0.963	0.176	0.298	0.19	0.5
AR_dynamic	Gauss-DK	tail_prob_4	0.283	0.974	0.382	0.549	0.326	0.5
AR_dynamic	Gauss-DK	tail_prob_3	0.327	0.961	0.471	0.632	0.386	0.5
AR_dynamic	Gauss-DK	tail_prob_2	0.332	0.937	0.735	0.824	0.458	0.5
AR_dynamic	Gauss-DK	best_f1_test	0.372	0.517	**1.0**	0.682	0.542	0.5
AR_dynamic	Gauss-D	best_f1_test	0.369	0.602	**1.0**	0.751	0.539	0.499
AR_dynamic	Gauss-D	tail_prob_5	0.312	0.985	0.559	0.713	0.401	0.499
AR_dynamic	Gauss-D	tail_prob_4	0.318	0.98	0.559	0.712	0.405	0.499
AR_dynamic	Gauss-DK	tail_prob_1	0.37	0.561	**1.0**	0.719	0.54	0.5
AR_dynamic	Gauss-D	tail_prob_3	0.327	0.969	0.588	0.732	0.42	0.499
BiDSPOT1e-1	Gauss-DK	tail_prob_5	0.431	0.989	0.471	0.638	0.45	0.528
BiDSPOT1e-1	Gauss-DK	tail_prob_4	0.431	0.984	0.5	0.663	0.463	0.528
BiDSPOT1e-1	Gauss-DK	tail_prob_3	0.405	0.977	0.529	0.687	0.459	0.528
BiDSPOT1e-1	Gauss-DK	tail_prob_1	0.391	0.604	**1.0**	0.753	0.562	0.528
BiDSPOT1e-1	Gauss-DK	best_f1_test	0.421	0.874	**1.0**	0.933	0.593	0.528
BiDSPOT1e-1	Gauss-DK	tail_prob_2	0.429	0.953	0.559	0.705	0.485	0.528
BiDSPOT1e-1	Gauss-D	tail_prob_3	0.402	0.979	0.529	0.687	0.457	0.523
BiDSPOT1e-1	Gauss-D	tail_prob_2	0.388	0.953	0.706	0.811	0.501	0.523
BiDSPOT1e-1	Gauss-D	tail_prob_1	0.382	0.598	**1.0**	0.749	0.552	0.523
BiDSPOT1e-1	Gauss-D	best_f1_test	0.398	0.859	**1.0**	0.924	0.57	0.523
BiDSPOT1e-1	Gauss-D	tail_prob_5	0.37	0.99	0.5	0.664	0.426	0.523
BiDSPOT1e-1	Gauss-D	tail_prob_4	0.388	0.985	0.5	0.663	0.437	0.523
BiDSPOT1e-10	Gauss-D	tail_prob_1	0.374	0.6	**1.0**	0.75	0.544	0.513
BiDSPOT1e-10	Gauss-D	best_f1_test	0.38	0.718	**1.0**	0.836	0.55	0.513
BiDSPOT1e-10	Gauss-D	tail_prob_2	0.328	0.958	0.794	0.868	0.464	0.513
BiDSPOT1e-10	Gauss-D	tail_prob_3	0.335	0.978	0.529	0.687	0.41	0.513
BiDSPOT1e-10	Gauss-D	tail_prob_4	0.338	0.985	0.5	0.663	0.403	0.513
BiDSPOT1e-10	Gauss-D	tail_prob_5	0.339	0.989	0.5	0.664	0.404	0.513
BiDSPOT1e-10	Gauss-DK	best_f1_test	0.381	0.511	**1.0**	0.676	0.551	0.517
BiDSPOT1e-10	Gauss-DK	tail_prob_1	0.375	0.601	**1.0**	0.751	0.546	0.517
BiDSPOT1e-10	Gauss-DK	tail_prob_2	0.361	0.954	0.559	0.705	0.438	0.517
BiDSPOT1e-10	Gauss-DK	tail_prob_3	0.363	0.975	0.5	0.661	0.42	0.517
BiDSPOT1e-10	Gauss-DK	tail_prob_5	0.313	0.984	0.265	0.417	0.287	0.517
BiDSPOT1e-10	Gauss-DK	tail_prob_4	0.39	0.984	0.5	0.663	0.438	0.517
BiDSPOT1e-3	Gauss-DK	tail_prob_5	**0.444**	0.989	0.471	0.638	0.457	0.526
BiDSPOT1e-3	Gauss-DK	tail_prob_4	0.4	0.984	0.5	0.663	0.444	0.526
BiDSPOT1e-3	Gauss-D	best_f1_test	0.385	0.865	**1.0**	0.928	0.556	0.52
BiDSPOT1e-3	Gauss-D	tail_prob_1	0.378	0.593	**1.0**	0.745	0.549	0.52
BiDSPOT1e-3	Gauss-D	tail_prob_2	0.38	0.952	0.912	0.931	0.537	0.52
BiDSPOT1e-3	Gauss-D	tail_prob_3	0.366	0.975	0.5	0.661	0.423	0.52
BiDSPOT1e-3	Gauss-D	tail_prob_4	0.362	0.986	0.5	0.663	0.42	0.52
BiDSPOT1e-3	Gauss-DK	best_f1_test	0.402	0.868	**1.0**	0.929	0.573	0.526
BiDSPOT1e-3	Gauss-DK	tail_prob_1	0.386	0.596	**1.0**	0.747	0.557	0.526
BiDSPOT1e-3	Gauss-DK	tail_prob_2	0.39	0.947	0.559	0.703	0.459	0.526
BiDSPOT1e-3	Gauss-DK	tail_prob_3	0.393	0.975	0.5	0.661	0.44	0.526
BiDSPOT1e-3	Gauss-D	tail_prob_5	0.386	0.989	0.5	0.664	0.436	0.52
BiDSPOT1e-5	Gauss-D	tail_prob_1	0.372	0.6	**1.0**	0.75	0.542	0.513
BiDSPOT1e-5	Gauss-D	tail_prob_2	0.367	0.956	0.882	0.918	0.519	0.513
BiDSPOT1e-5	Gauss-D	tail_prob_3	0.305	0.974	0.529	0.686	0.387	0.513
BiDSPOT1e-5	Gauss-D	tail_prob_4	0.312	0.983	0.5	0.663	0.385	0.513
BiDSPOT1e-5	Gauss-D	tail_prob_5	0.325	0.988	0.5	0.664	0.394	0.513
BiDSPOT1e-5	Gauss-DK	best_f1_test	0.393	0.856	**1.0**	0.922	0.565	0.518
BiDSPOT1e-5	Gauss-DK	tail_prob_1	0.378	0.604	**1.0**	0.753	0.548	0.518
BiDSPOT1e-5	Gauss-DK	tail_prob_2	0.391	0.949	0.588	0.726	0.47	0.518
BiDSPOT1e-5	Gauss-DK	tail_prob_3	0.327	0.972	0.5	0.66	0.396	0.518
BiDSPOT1e-5	Gauss-DK	tail_prob_4	0.348	0.982	0.5	0.663	0.41	0.518
BiDSPOT1e-5	Gauss-DK	tail_prob_5	0.398	0.988	0.441	0.61	0.419	0.518
BiDSPOT1e-5	Gauss-D	best_f1_test	0.382	0.924	**1.0**	0.961	0.553	0.513
FluxEV1e-1	Gauss-D	best_f1_test	0.384	0.515	**1.0**	0.68	0.555	0.501
FluxEV1e-1	Gauss-D	tail_prob_1	0.347	0.732	**1.0**	0.845	0.515	0.501
FluxEV1e-1	Gauss-D	tail_prob_2	0.332	0.901	**1.0**	0.948	0.498	0.501
FluxEV1e-1	Gauss-D	tail_prob_3	0.276	0.95	**1.0**	**0.974**	0.433	0.501

Continued on next page

Table A.3: Evaluation metrics on SKAB dataset with Gauss-D and Gauss-D-DK aggregation with Best-F-score and Tail-probability thresholding

Algorithm	Aggregation	Thresholding method	Point-wise Precision	Normalized Precision	Recall	F-score	F_c-score	AUC
FluxEV1e-1	Gauss-D	tail_prob_4	0.207	0.956	0.794	0.868	0.328	0.501
FluxEV1e-1	Gauss-DK	best_f1_test	0.371	0.414	**1.0**	0.586	0.541	0.491
FluxEV1e-1	Gauss-D	tail_prob_5	0.162	0.951	0.588	0.727	0.255	0.501
FluxEV1e-1	Gauss-DK	tail_prob_5	0.101	0.934	0.412	0.571	0.163	0.491
FluxEV1e-1	Gauss-DK	tail_prob_4	0.131	0.931	0.412	0.571	0.198	0.491
FluxEV1e-1	Gauss-DK	tail_prob_2	0.314	0.941	0.912	0.926	0.467	0.491
FluxEV1e-1	Gauss-DK	tail_prob_1	0.363	0.71	**1.0**	0.831	0.533	0.491
FluxEV1e-1	Gauss-DK	tail_prob_3	0.176	0.926	0.441	0.598	0.252	0.491
FluxEV1e-10	Gauss-DK	tail_prob_5	0.331	0.979	0.618	0.758	0.431	0.51
FluxEV1e-10	Gauss-DK	tail_prob_4	0.313	0.967	0.676	0.796	0.428	0.51
FluxEV1e-10	Gauss-DK	tail_prob_3	0.294	0.932	0.735	0.822	0.42	0.51
FluxEV1e-10	Gauss-DK	tail_prob_2	0.352	0.862	0.971	0.913	0.517	0.51
FluxEV1e-10	Gauss-DK	best_f1_test	0.376	0.554	**1.0**	0.713	0.547	0.51
FluxEV1e-10	Gauss-D	tail_prob_5	0.295	0.965	0.794	0.871	0.43	0.516
FluxEV1e-10	Gauss-D	tail_prob_4	0.34	0.956	0.941	0.949	0.5	0.516
FluxEV1e-10	Gauss-D	tail_prob_3	0.348	0.923	**1.0**	0.96	0.517	0.516
FluxEV1e-10	Gauss-D	tail_prob_2	0.355	0.834	**1.0**	0.909	0.524	0.516
FluxEV1e-10	Gauss-D	tail_prob_1	0.37	0.599	**1.0**	0.749	0.541	0.516
FluxEV1e-10	Gauss-D	best_f1_test	0.382	0.407	**1.0**	0.578	0.553	0.516
FluxEV1e-10	Gauss-DK	tail_prob_1	0.375	0.545	**1.0**	0.706	0.546	0.51
FluxEV1e-3	Gauss-DK	tail_prob_5	0.333	0.98	0.618	0.758	0.433	0.51
FluxEV1e-3	Gauss-D	tail_prob_5	0.295	0.965	0.794	0.871	0.43	0.515
FluxEV1e-3	Gauss-D	best_f1_test	0.382	0.407	**1.0**	0.578	0.553	0.515
FluxEV1e-3	Gauss-D	tail_prob_1	0.37	0.599	**1.0**	0.749	0.541	0.515
FluxEV1e-3	Gauss-D	tail_prob_2	0.355	0.834	**1.0**	0.909	0.524	0.515
FluxEV1e-3	Gauss-D	tail_prob_3	0.349	0.923	**1.0**	0.96	0.517	0.515
FluxEV1e-3	Gauss-D	tail_prob_4	0.34	0.956	0.941	0.949	0.5	0.515
FluxEV1e-3	Gauss-DK	best_f1_test	0.376	0.554	**1.0**	0.713	0.547	0.51
FluxEV1e-3	Gauss-DK	tail_prob_1	0.375	0.545	**1.0**	0.706	0.546	0.51
FluxEV1e-3	Gauss-DK	tail_prob_2	0.353	0.862	0.971	0.913	0.517	0.51
FluxEV1e-3	Gauss-DK	tail_prob_3	0.295	0.932	0.735	0.822	0.421	0.51
FluxEV1e-3	Gauss-DK	tail_prob_4	0.313	0.967	0.676	0.796	0.428	0.51
FluxEV1e-5	Gauss-DK	tail_prob_5	0.333	0.98	0.618	0.758	0.433	0.51
FluxEV1e-5	Gauss-DK	tail_prob_4	0.313	0.967	0.676	0.796	0.428	0.51
FluxEV1e-5	Gauss-D	best_f1_test	0.382	0.406	**1.0**	0.578	0.553	0.515
FluxEV1e-5	Gauss-D	tail_prob_1	0.37	0.599	**1.0**	0.749	0.541	0.515
FluxEV1e-5	Gauss-D	tail_prob_2	0.355	0.834	**1.0**	0.909	0.524	0.515
FluxEV1e-5	Gauss-D	tail_prob_3	0.349	0.923	**1.0**	0.96	0.517	0.515
FluxEV1e-5	Gauss-D	tail_prob_4	0.34	0.956	0.941	0.949	0.5	0.515
FluxEV1e-5	Gauss-D	tail_prob_5	0.295	0.965	0.794	0.871	0.43	0.515
FluxEV1e-5	Gauss-DK	best_f1_test	0.376	0.554	**1.0**	0.713	0.547	0.51
FluxEV1e-5	Gauss-DK	tail_prob_1	0.375	0.545	**1.0**	0.706	0.546	0.51
FluxEV1e-5	Gauss-DK	tail_prob_2	0.353	0.862	0.971	0.913	0.517	0.51
FluxEV1e-5	Gauss-DK	tail_prob_3	0.295	0.932	0.735	0.822	0.421	0.51
UnivarAutoencoder	Gauss-DK	tail_prob_3	0.411	0.977	0.529	0.687	0.463	**0.536**
UnivarAutoencoder	Gauss-DK	tail_prob_2	0.421	0.95	0.588	0.727	0.49	**0.536**
UnivarAutoencoder	Gauss-DK	tail_prob_1	0.391	0.608	**1.0**	0.756	0.562	**0.536**
UnivarAutoencoder	Gauss-DK	best_f1_test	0.438	0.904	**1.0**	0.95	**0.609**	**0.536**
UnivarAutoencoder	Gauss-D	tail_prob_5	0.364	0.989	0.5	0.664	0.421	0.53
UnivarAutoencoder	Gauss-D	best_f1_test	0.401	0.936	**1.0**	0.967	0.572	0.53
UnivarAutoencoder	Gauss-D	tail_prob_3	0.391	0.978	0.559	0.711	0.46	0.53
UnivarAutoencoder	Gauss-D	tail_prob_2	0.4	0.948	0.824	0.882	0.539	0.53
UnivarAutoencoder	Gauss-D	tail_prob_1	0.387	0.609	**1.0**	0.757	0.558	0.53
UnivarAutoencoder	Gauss-DK	tail_prob_4	0.429	0.986	0.529	0.689	0.474	**0.536**
UnivarAutoencoder	Gauss-D	tail_prob_4	0.385	0.986	0.529	0.689	0.446	0.53
UnivarAutoencoder	Gauss-DK	tail_prob_5	0.416	0.989	0.5	0.664	0.454	**0.536**

Table A.4: Evaluation metrics on MSL dataset with Gauss-D and Gauss-D-DK aggregation with Best-F-score and Tail-probability thresholding

Algorithm	Aggregation	Thresholding method	Point-wise Precision	Normalized Precision	Recall	F-score	F_c-score	AUC
AR	Gauss-D	best_f1_test	0.431	0.567	0.685	0.568	0.466	0.511
AR	Gauss-DK	tail_prob_4	0.15	0.158	0.204	0.156	0.154	0.601
AR	Gauss-DK	tail_prob_3	0.143	0.176	0.241	0.181	0.157	0.601
AR	Gauss-DK	tail_prob_2	0.149	0.189	0.296	0.21	0.182	0.601
AR	Gauss-DK	tail_prob_1	0.096	0.129	0.407	0.177	0.14	0.601
AR	Gauss-DK	best_f1_test	0.458	0.485	0.722	0.511	0.488	0.601
AR	Gauss-DK	tail_prob_5	0.156	0.168	0.204	0.165	0.161	0.601
AR	Gauss-D	tail_prob_4	0.144	0.291	0.389	0.32	0.186	0.511
AR	Gauss-D	tail_prob_3	0.135	0.27	0.389	0.301	0.175	0.511
AR	Gauss-D	tail_prob_2	0.121	0.244	0.389	0.278	0.157	0.511
AR	Gauss-D	tail_prob_1	0.121	0.235	0.5	0.274	0.159	0.511
AR	Gauss-D	tail_prob_5	0.155	0.306	0.389	0.332	0.198	0.511
AR_diff	Gauss-DK	tail_prob_5	0.162	0.204	0.296	0.228	0.19	0.624
AR_diff	Gauss-D	tail_prob_2	0.226	0.477	0.685	0.54	0.298	0.515
AR_diff	Gauss-D	tail_prob_3	0.225	0.519	0.667	0.566	0.294	0.515
AR_diff	Gauss-D	tail_prob_4	0.216	0.53	0.648	0.572	0.28	0.515
AR_diff	Gauss-D	tail_prob_5	0.239	0.553	0.648	0.588	0.307	0.515
AR_diff	Gauss-DK	best_f1_test	0.508	0.581	0.87	0.629	0.565	0.624
AR_diff	Gauss-DK	tail_prob_1	0.157	0.232	0.667	0.321	0.228	0.624
AR_diff	Gauss-DK	tail_prob_2	0.247	0.294	0.5	0.343	0.295	0.624
AR_diff	Gauss-DK	tail_prob_3	0.193	0.242	0.389	0.284	0.24	0.624
AR_diff	Gauss-DK	tail_prob_4	0.192	0.228	0.352	0.267	0.232	0.624
AR_diff	Gauss-D	tail_prob_1	0.219	0.429	0.778	0.512	0.295	0.515
AR_diff	Gauss-D	best_f1_test	0.466	0.666	0.852	0.701	0.528	0.515
AR_dynamic	Gauss-D	tail_prob_2	0.0	0.0	0.0	0.0	0.0	0.5
AR_dynamic	Gauss-D	tail_prob_1	0.0	0.0	0.0	0.0	0.0	0.5
AR_dynamic	Gauss-DK	tail_prob_5	0.0	0.0	0.0	0.0	0.0	0.514
AR_dynamic	Gauss-DK	tail_prob_4	0.0	0.0	0.0	0.0	0.0	0.514
AR_dynamic	Gauss-DK	tail_prob_3	0.0	0.0	0.0	0.0	0.0	0.514
AR_dynamic	Gauss-DK	tail_prob_2	0.0	0.0	0.0	0.0	0.0	0.514
AR_dynamic	Gauss-DK	best_f1_test	0.06	0.061	0.185	0.087	0.087	0.514
AR_dynamic	Gauss-D	best_f1_test	0.0	0.0	0.0	0.0	0.0	0.5
AR_dynamic	Gauss-D	tail_prob_5	0.0	0.0	0.0	0.0	0.0	0.5
AR_dynamic	Gauss-D	tail_prob_4	0.0	0.0	0.0	0.0	0.0	0.5
AR_dynamic	Gauss-DK	tail_prob_1	0.0	0.0	0.0	0.0	0.0	0.514
AR_dynamic	Gauss-D	tail_prob_3	0.0	0.0	0.0	0.0	0.0	0.5
BiDSPOT1e-1	Gauss-DK	tail_prob_5	0.096	0.097	0.167	0.114	0.111	0.563
BiDSPOT1e-1	Gauss-DK	tail_prob_4	0.093	0.093	0.167	0.109	0.108	0.563
BiDSPOT1e-1	Gauss-DK	tail_prob_3	0.09	0.089	0.167	0.104	0.105	0.563
BiDSPOT1e-1	Gauss-DK	tail_prob_1	0.053	0.06	0.185	0.081	0.073	0.563
BiDSPOT1e-1	Gauss-DK	best_f1_test	0.21	0.221	0.444	0.27	0.258	0.563
BiDSPOT1e-1	Gauss-DK	tail_prob_2	0.066	0.075	0.167	0.091	0.086	0.563
BiDSPOT1e-1	Gauss-D	tail_prob_3	0.121	0.124	0.167	0.132	0.127	0.534
BiDSPOT1e-1	Gauss-D	tail_prob_2	0.088	0.101	0.167	0.113	0.097	0.534
BiDSPOT1e-1	Gauss-D	tail_prob_1	0.092	0.105	0.204	0.12	0.105	0.534
BiDSPOT1e-1	Gauss-D	best_f1_test	0.16	0.17	0.259	0.19	0.179	0.534
BiDSPOT1e-1	Gauss-D	tail_prob_5	0.128	0.133	0.167	0.142	0.135	0.534
BiDSPOT1e-1	Gauss-D	tail_prob_4	0.125	0.129	0.167	0.138	0.132	0.534
BiDSPOT1e-10	Gauss-D	tail_prob_1	0.121	0.123	0.185	0.131	0.128	0.547
BiDSPOT1e-10	Gauss-D	best_f1_test	0.147	0.154	0.167	0.159	0.154	0.547
BiDSPOT1e-10	Gauss-D	tail_prob_2	0.122	0.122	0.167	0.129	0.129	0.547
BiDSPOT1e-10	Gauss-D	tail_prob_3	0.124	0.125	0.167	0.132	0.132	0.547
BiDSPOT1e-10	Gauss-D	tail_prob_4	0.13	0.131	0.167	0.139	0.139	0.547
BiDSPOT1e-10	Gauss-D	tail_prob_5	0.133	0.134	0.167	0.143	0.142	0.547
BiDSPOT1e-10	Gauss-DK	best_f1_test	0.196	0.207	0.315	0.238	0.23	0.593
BiDSPOT1e-10	Gauss-DK	tail_prob_1	0.059	0.057	0.167	0.077	0.08	0.593
BiDSPOT1e-10	Gauss-DK	tail_prob_2	0.066	0.063	0.167	0.084	0.087	0.593
BiDSPOT1e-10	Gauss-DK	tail_prob_3	0.072	0.069	0.167	0.09	0.094	0.593
BiDSPOT1e-10	Gauss-DK	tail_prob_5	0.082	0.08	0.167	0.102	0.104	0.593
BiDSPOT1e-10	Gauss-DK	tail_prob_4	0.077	0.075	0.167	0.098	0.1	0.593
BiDSPOT1e-3	Gauss-DK	tail_prob_5	0.022	0.021	0.056	0.03	0.031	0.547
BiDSPOT1e-3	Gauss-DK	tail_prob_4	0.019	0.018	0.056	0.027	0.028	0.547
BiDSPOT1e-3	Gauss-D	best_f1_test	0.075	0.082	0.111	0.077	0.071	0.527
BiDSPOT1e-3	Gauss-D	tail_prob_1	0.01	0.012	0.074	0.02	0.017	0.527
BiDSPOT1e-3	Gauss-D	tail_prob_2	0.011	0.011	0.056	0.018	0.018	0.527
BiDSPOT1e-3	Gauss-D	tail_prob_3	0.013	0.013	0.056	0.021	0.021	0.527
BiDSPOT1e-3	Gauss-D	tail_prob_4	0.019	0.019	0.056	0.028	0.028	0.527
BiDSPOT1e-3	Gauss-DK	best_f1_test	0.129	0.14	0.259	0.159	0.15	0.547
BiDSPOT1e-3	Gauss-DK	tail_prob_1	0.01	0.009	0.056	0.015	0.017	0.547
BiDSPOT1e-3	Gauss-DK	tail_prob_2	0.012	0.01	0.056	0.017	0.019	0.547
BiDSPOT1e-3	Gauss-DK	tail_prob_3	0.016	0.013	0.056	0.021	0.024	0.547
BiDSPOT1e-3	Gauss-D	tail_prob_5	0.022	0.023	0.056	0.032	0.031	0.527
BiDSPOT1e-5	Gauss-D	tail_prob_1	0.041	0.042	0.074	0.046	0.043	0.509
BiDSPOT1e-5	Gauss-D	tail_prob_2	0.041	0.041	0.056	0.044	0.044	0.509
BiDSPOT1e-5	Gauss-D	tail_prob_3	0.044	0.044	0.056	0.047	0.047	0.509
BiDSPOT1e-5	Gauss-D	tail_prob_4	0.047	0.047	0.056	0.05	0.05	0.509
BiDSPOT1e-5	Gauss-D	tail_prob_5	0.048	0.048	0.056	0.051	0.051	0.509
BiDSPOT1e-5	Gauss-DK	best_f1_test	0.114	0.115	0.278	0.146	0.144	0.542
BiDSPOT1e-5	Gauss-DK	tail_prob_1	0.018	0.016	0.056	0.024	0.026	0.542
BiDSPOT1e-5	Gauss-DK	tail_prob_2	0.022	0.02	0.056	0.029	0.031	0.542
BiDSPOT1e-5	Gauss-DK	tail_prob_3	0.027	0.024	0.056	0.033	0.036	0.542
BiDSPOT1e-5	Gauss-DK	tail_prob_4	0.028	0.026	0.056	0.035	0.037	0.542
BiDSPOT1e-5	Gauss-DK	tail_prob_5	0.03	0.029	0.056	0.038	0.039	0.542
BiDSPOT1e-5	Gauss-D	best_f1_test	0.056	0.058	0.093	0.06	0.059	0.509
FluxEV1e-1	Gauss-D	best_f1_test	0.335	0.383	0.864	0.445	0.407	0.596
FluxEV1e-1	Gauss-D	tail_prob_1	0.061	0.129	0.148	0.137	0.075	0.596
FluxEV1e-1	Gauss-D	tail_prob_2	0.066	0.135	0.148	0.141	0.08	0.596
FluxEV1e-1	Gauss-D	tail_prob_3	0.066	0.137	0.148	0.142	0.078	0.596

Continued on next page

Table A.4: Evaluation metrics on MSL dataset with Gauss-D and Gauss-D-DK aggregation with Best-F-score and Tail-probability thresholding

Algorithm	Aggregation	Thresholding method	Point-wise Precision	Normalized Precision	Recall	F-score	F_c-score	AUC
FluxEV1e-1	Gauss-D	tail_prob_4	0.067	0.139	0.148	0.143	0.079	0.596
FluxEV1e-1	Gauss-DK	best_f1_test	0.345	0.352	0.883	0.415	0.411	0.598
FluxEV1e-1	Gauss-D	tail_prob_5	0.068	0.139	0.148	0.143	0.08	0.596
FluxEV1e-1	Gauss-DK	tail_prob_5	0.027	0.046	0.093	0.058	0.042	0.598
FluxEV1e-1	Gauss-DK	tail_prob_4	0.028	0.043	0.093	0.056	0.042	0.598
FluxEV1e-1	Gauss-DK	tail_prob_2	0.028	0.041	0.111	0.055	0.044	0.598
FluxEV1e-1	Gauss-DK	tail_prob_1	0.027	0.06	0.148	0.078	0.044	0.598
FluxEV1e-1	Gauss-DK	tail_prob_3	0.028	0.045	0.111	0.06	0.044	0.598
FluxEV1e-10	Gauss-DK	tail_prob_5	0.052	0.067	0.111	0.082	0.068	0.584
FluxEV1e-10	Gauss-DK	tail_prob_4	0.05	0.064	0.111	0.079	0.066	0.584
FluxEV1e-10	Gauss-DK	tail_prob_3	0.047	0.059	0.111	0.075	0.063	0.584
FluxEV1e-10	Gauss-DK	tail_prob_2	0.054	0.073	0.148	0.095	0.076	0.584
FluxEV1e-10	Gauss-DK	best_f1_test	0.399	0.417	0.846	0.461	0.444	0.584
FluxEV1e-10	Gauss-D	tail_prob_5	0.077	0.102	0.111	0.106	0.08	0.56
FluxEV1e-10	Gauss-D	tail_prob_4	0.077	0.102	0.111	0.106	0.079	0.56
FluxEV1e-10	Gauss-D	tail_prob_3	0.078	0.101	0.111	0.105	0.081	0.56
FluxEV1e-10	Gauss-D	tail_prob_2	0.091	0.147	0.185	0.162	0.102	0.56
FluxEV1e-10	Gauss-D	tail_prob_1	0.137	0.181	0.364	0.219	0.17	0.56
FluxEV1e-10	Gauss-D	best_f1_test	0.393	0.428	0.827	0.477	0.446	0.56
FluxEV1e-10	Gauss-DK	tail_prob_1	0.093	0.105	0.29	0.146	0.134	0.584
FluxEV1e-3	Gauss-DK	tail_prob_5	0.081	0.117	0.167	0.135	0.099	0.638
FluxEV1e-3	Gauss-D	tail_prob_5	0.106	0.169	0.185	0.176	0.124	0.635
FluxEV1e-3	Gauss-D	best_f1_test	0.441	0.497	0.864	0.543	0.504	0.635
FluxEV1e-3	Gauss-D	tail_prob_1	0.104	0.17	0.389	0.21	0.145	0.635
FluxEV1e-3	Gauss-D	tail_prob_2	0.114	0.2	0.204	0.198	0.134	0.635
FluxEV1e-3	Gauss-D	tail_prob_3	0.106	0.168	0.185	0.175	0.123	0.635
FluxEV1e-3	Gauss-D	tail_prob_4	0.106	0.169	0.185	0.176	0.124	0.635
FluxEV1e-3	Gauss-DK	best_f1_test	0.398	0.433	0.846	0.474	0.453	0.638
FluxEV1e-3	Gauss-DK	tail_prob_1	0.102	0.13	0.407	0.18	0.151	0.638
FluxEV1e-3	Gauss-DK	tail_prob_2	0.078	0.114	0.185	0.133	0.102	0.638
FluxEV1e-3	Gauss-DK	tail_prob_3	0.072	0.1	0.167	0.12	0.093	0.638
FluxEV1e-3	Gauss-DK	tail_prob_4	0.076	0.108	0.167	0.128	0.096	0.638
FluxEV1e-5	Gauss-DK	tail_prob_5	0.08	0.101	0.111	0.1	0.09	0.607
FluxEV1e-5	Gauss-DK	tail_prob_4	0.069	0.096	0.111	0.097	0.085	0.607
FluxEV1e-5	Gauss-D	best_f1_test	0.414	0.445	0.901	0.51	0.483	0.588
FluxEV1e-5	Gauss-D	tail_prob_1	0.121	0.165	0.296	0.191	0.144	0.588
FluxEV1e-5	Gauss-D	tail_prob_2	0.108	0.183	0.259	0.206	0.125	0.588
FluxEV1e-5	Gauss-D	tail_prob_3	0.098	0.161	0.185	0.171	0.11	0.588
FluxEV1e-5	Gauss-D	tail_prob_4	0.097	0.164	0.185	0.173	0.108	0.588
FluxEV1e-5	Gauss-D	tail_prob_5	0.1	0.165	0.185	0.174	0.111	0.588
FluxEV1e-5	Gauss-DK	best_f1_test	0.37	0.392	0.864	0.444	0.423	0.607
FluxEV1e-5	Gauss-DK	tail_prob_1	0.082	0.102	0.278	0.139	0.119	0.607
FluxEV1e-5	Gauss-DK	tail_prob_2	0.069	0.101	0.148	0.114	0.093	0.607
FluxEV1e-5	Gauss-DK	tail_prob_3	0.062	0.089	0.111	0.093	0.08	0.607
UnivarAutoencoder	Gauss-DK	tail_prob_3	0.293	0.435	0.772	0.502	0.389	**0.77**
UnivarAutoencoder	Gauss-DK	tail_prob_2	0.291	0.396	0.87	0.498	0.398	**0.77**
UnivarAutoencoder	Gauss-DK	tail_prob_1	0.21	0.275	0.944	0.381	0.307	**0.77**
UnivarAutoencoder	Gauss-DK	best_f1_test	0.658	0.755	0.944	0.793	0.706	**0.77**
UnivarAutoencoder	Gauss-D	tail_prob_5	0.387	0.655	0.858	0.713	0.479	0.667
UnivarAutoencoder	Gauss-D	best_f1_test	**0.71**	**0.808**	0.92	**0.811**	**0.727**	0.667
UnivarAutoencoder	Gauss-D	tail_prob_3	0.32	0.541	0.926	0.637	0.419	0.667
UnivarAutoencoder	Gauss-D	tail_prob_2	0.301	0.486	0.944	0.584	0.393	0.667
UnivarAutoencoder	Gauss-D	tail_prob_1	0.245	0.376	**0.963**	0.475	0.331	0.667
UnivarAutoencoder	Gauss-DK	tail_prob_4	0.299	0.424	0.704	0.494	0.384	**0.77**
UnivarAutoencoder	Gauss-D	tail_prob_4	0.337	0.563	0.87	0.648	0.436	0.667
UnivarAutoencoder	Gauss-DK	tail_prob_5	0.299	0.377	0.574	0.424	0.36	**0.77**

Table A.5: Evaluation metrics on SMAP dataset with Gauss-D and Gauss-D-DK aggregation with Best-F-score and Tail-probability thresholding

Algorithm	Aggregation	Thresholding method	Point-wise Precision	Normalized Precision	Recall	F-score	F_c-score	AUC
AR	Gauss-D	best_f1_test	0.371	0.549	0.806	0.587	0.429	0.454
AR	Gauss-DK	tail_prob_4	0.146	0.154	0.185	0.162	0.157	0.516
AR	Gauss-DK	tail_prob_3	0.141	0.149	0.185	0.157	0.151	0.516
AR	Gauss-DK	tail_prob_2	0.174	0.197	0.241	0.207	0.19	0.516
AR	Gauss-DK	tail_prob_1	0.137	0.278	0.593	0.332	0.187	0.516
AR	Gauss-DK	best_f1_test	0.417	0.474	0.827	0.519	0.468	0.516
AR	Gauss-DK	tail_prob_5	0.133	0.141	0.167	0.148	0.142	0.516
AR	Gauss-D	tail_prob_4	0.173	0.407	0.519	0.44	0.212	0.454
AR	Gauss-D	tail_prob_3	0.152	0.439	0.586	0.477	0.199	0.454
AR	Gauss-D	tail_prob_2	0.135	0.421	0.667	0.477	0.185	0.454
AR	Gauss-D	tail_prob_1	0.129	0.279	0.685	0.342	0.175	0.454
AR	Gauss-D	tail_prob_5	0.184	0.389	0.463	0.413	0.217	0.454
AR_diff	Gauss-DK	tail_prob_5	0.152	0.157	0.179	0.163	0.16	0.572
AR_diff	Gauss-D	tail_prob_2	0.152	0.461	0.685	0.522	0.204	0.505
AR_diff	Gauss-D	tail_prob_3	0.158	0.51	0.633	0.543	0.207	0.505
AR_diff	Gauss-D	tail_prob_4	0.178	0.455	0.537	0.481	0.218	0.505
AR_diff	Gauss-D	tail_prob_5	0.148	0.416	0.481	0.437	0.188	0.505
AR_diff	Gauss-DK	best_f1_test	0.383	0.459	0.858	0.507	0.449	0.572
AR_diff	Gauss-DK	tail_prob_1	0.149	0.273	0.627	0.34	0.207	0.572
AR_diff	Gauss-DK	tail_prob_2	0.15	0.195	0.269	0.211	0.177	0.572
AR_diff	Gauss-DK	tail_prob_3	0.13	0.135	0.185	0.15	0.146	0.572
AR_diff	Gauss-DK	tail_prob_4	0.138	0.143	0.179	0.153	0.15	0.572
AR_diff	Gauss-D	tail_prob_1	0.148	0.312	0.731	0.385	0.2	0.505
AR_diff	Gauss-D	best_f1_test	0.312	0.494	0.772	0.522	0.374	0.505
AR_dynamic	Gauss-D	tail_prob_2	0.009	0.039	0.093	0.05	0.013	0.536
AR_dynamic	Gauss-D	tail_prob_1	0.008	0.028	0.093	0.037	0.013	0.536
AR_dynamic	Gauss-DK	tail_prob_5	0.0	0.0	0.0	0.0	0.0	0.53
AR_dynamic	Gauss-DK	tail_prob_4	0.0	0.0	0.0	0.0	0.0	0.53
AR_dynamic	Gauss-DK	tail_prob_3	0.0	0.017	0.019	0.018	0.001	0.53
AR_dynamic	Gauss-DK	tail_prob_2	0.004	0.022	0.037	0.026	0.007	0.53
AR_dynamic	Gauss-DK	best_f1_test	0.187	0.209	0.574	0.267	0.247	0.53
AR_dynamic	Gauss-D	best_f1_test	0.152	0.179	0.444	0.216	0.189	0.536
AR_dynamic	Gauss-D	tail_prob_5	0.006	0.04	0.056	0.046	0.009	0.536
AR_dynamic	Gauss-D	tail_prob_4	0.009	0.037	0.056	0.043	0.013	0.536
AR_dynamic	Gauss-DK	tail_prob_1	0.009	0.019	0.074	0.027	0.013	0.53
AR_dynamic	Gauss-D	tail_prob_3	0.009	0.041	0.074	0.05	0.013	0.536
BiDSPOT1e-1	Gauss-DK	tail_prob_5	0.309	0.348	0.426	0.37	0.341	0.615
BiDSPOT1e-1	Gauss-DK	tail_prob_4	0.31	0.353	0.444	0.374	0.343	0.615
BiDSPOT1e-1	Gauss-DK	tail_prob_3	0.312	0.367	0.469	0.386	0.347	0.615
BiDSPOT1e-1	Gauss-DK	tail_prob_1	0.186	0.233	0.63	0.29	0.237	0.615
BiDSPOT1e-1	Gauss-DK	best_f1_test	0.481	0.519	0.836	0.538	0.513	0.615
BiDSPOT1e-1	Gauss-DK	tail_prob_2	0.249	0.328	0.506	0.367	0.298	0.615
BiDSPOT1e-1	Gauss-D	tail_prob_3	0.254	0.419	0.593	0.461	0.297	0.557
BiDSPOT1e-1	Gauss-D	tail_prob_2	0.224	0.351	0.593	0.402	0.269	0.557
BiDSPOT1e-1	Gauss-D	tail_prob_1	0.189	0.284	0.63	0.342	0.235	0.557
BiDSPOT1e-1	Gauss-D	best_f1_test	0.44	0.523	0.821	0.548	0.478	0.557
BiDSPOT1e-1	Gauss-D	tail_prob_5	0.314	0.455	0.562	0.483	0.351	0.557
BiDSPOT1e-1	Gauss-D	tail_prob_4	0.28	0.449	0.58	0.482	0.319	0.557
BiDSPOT1e-10	Gauss-D	tail_prob_1	0.233	0.418	0.463	0.402	0.276	0.518
BiDSPOT1e-10	Gauss-D	best_f1_test	0.339	0.417	0.611	0.424	0.353	0.518
BiDSPOT1e-10	Gauss-D	tail_prob_2	0.245	0.426	0.463	0.411	0.289	0.518
BiDSPOT1e-10	Gauss-D	tail_prob_3	0.298	0.433	0.463	0.419	0.325	0.518
BiDSPOT1e-10	Gauss-D	tail_prob_4	0.318	0.438	0.463	0.426	0.337	0.518
BiDSPOT1e-10	Gauss-D	tail_prob_5	0.321	0.443	0.463	0.431	0.34	0.518
BiDSPOT1e-10	Gauss-DK	best_f1_test	0.459	0.514	0.685	0.523	0.485	0.567
BiDSPOT1e-10	Gauss-DK	tail_prob_1	0.258	0.363	0.463	0.374	0.296	0.567
BiDSPOT1e-10	Gauss-DK	tail_prob_2	0.269	0.381	0.463	0.389	0.306	0.567
BiDSPOT1e-10	Gauss-DK	tail_prob_3	0.275	0.393	0.463	0.4	0.313	0.567
BiDSPOT1e-10	Gauss-DK	tail_prob_5	0.281	0.413	0.463	0.418	0.319	0.567
BiDSPOT1e-10	Gauss-DK	tail_prob_4	0.279	0.404	0.463	0.41	0.317	0.567
BiDSPOT1e-3	Gauss-DK	tail_prob_5	0.281	0.36	0.417	0.368	0.309	0.541
BiDSPOT1e-3	Gauss-DK	tail_prob_4	0.28	0.375	0.444	0.384	0.308	0.541
BiDSPOT1e-3	Gauss-D	best_f1_test	0.359	0.454	0.682	0.47	0.39	0.493
BiDSPOT1e-3	Gauss-D	tail_prob_1	0.202	0.38	0.5	0.376	0.241	0.493
BiDSPOT1e-3	Gauss-D	tail_prob_2	0.212	0.389	0.463	0.382	0.249	0.493
BiDSPOT1e-3	Gauss-D	tail_prob_3	0.295	0.418	0.463	0.409	0.311	0.493
BiDSPOT1e-3	Gauss-D	tail_prob_4	0.301	0.424	0.463	0.416	0.318	0.493
BiDSPOT1e-3	Gauss-DK	best_f1_test	0.42	0.489	0.701	0.502	0.451	0.541
BiDSPOT1e-3	Gauss-DK	tail_prob_1	0.247	0.322	0.5	0.34	0.277	0.541
BiDSPOT1e-3	Gauss-DK	tail_prob_2	0.261	0.342	0.457	0.357	0.289	0.541
BiDSPOT1e-3	Gauss-DK	tail_prob_3	0.279	0.364	0.444	0.375	0.306	0.541
BiDSPOT1e-3	Gauss-D	tail_prob_5	0.302	0.429	0.463	0.422	0.32	0.493
BiDSPOT1e-5	Gauss-D	tail_prob_1	0.219	0.41	0.54	0.411	0.269	0.528
BiDSPOT1e-5	Gauss-D	tail_prob_2	0.233	0.423	0.522	0.424	0.281	0.528
BiDSPOT1e-5	Gauss-D	tail_prob_3	0.321	0.454	0.522	0.454	0.341	0.528
BiDSPOT1e-5	Gauss-D	tail_prob_4	0.336	0.465	0.522	0.466	0.354	0.528
BiDSPOT1e-5	Gauss-D	tail_prob_5	0.339	0.476	0.522	0.477	0.358	0.528
BiDSPOT1e-5	Gauss-DK	best_f1_test	0.474	0.518	0.67	0.535	0.497	0.589
BiDSPOT1e-5	Gauss-DK	tail_prob_1	0.278	0.358	0.531	0.379	0.316	0.589
BiDSPOT1e-5	Gauss-DK	tail_prob_2	0.294	0.381	0.522	0.401	0.331	0.589
BiDSPOT1e-5	Gauss-DK	tail_prob_3	0.311	0.412	0.522	0.432	0.35	0.589
BiDSPOT1e-5	Gauss-DK	tail_prob_4	0.322	0.433	0.522	0.451	0.361	0.589
BiDSPOT1e-5	Gauss-DK	tail_prob_5	0.33	0.426	0.494	0.44	0.368	0.589
BiDSPOT1e-5	Gauss-D	best_f1_test	0.448	0.493	0.66	0.503	0.466	0.528
FluxEV1e-1	Gauss-D	best_f1_test	0.302	0.465	0.815	0.521	0.353	0.521
FluxEV1e-1	Gauss-D	tail_prob_1	0.071	0.312	0.63	0.379	0.106	0.521
FluxEV1e-1	Gauss-D	tail_prob_2	0.071	0.347	0.63	0.414	0.105	0.521
FluxEV1e-1	Gauss-D	tail_prob_3	0.074	0.368	0.63	0.434	0.109	0.521

Continued on next page

Table A.5: Evaluation metrics on SMAP dataset with Gauss-D and Gauss-D-DK aggregation with Best-F-score and Tail-probability thresholding

Algorithm	Aggregation	Thresholding method	Point-wise Precision	Normalized Precision	Recall	F-score	F_c-score	AUC
FluxEV1e-1	Gauss-D	tail_prob_4	0.075	0.382	0.63	0.447	0.111	0.521
FluxEV1e-1	Gauss-DK	best_f1_test	0.334	0.402	0.833	0.444	0.381	0.528
FluxEV1e-1	Gauss-D	tail_prob_5	0.076	0.39	0.63	0.455	0.113	0.521
FluxEV1e-1	Gauss-DK	tail_prob_5	0.114	0.206	0.287	0.227	0.142	0.528
FluxEV1e-1	Gauss-DK	tail_prob_4	0.109	0.196	0.287	0.217	0.137	0.528
FluxEV1e-1	Gauss-DK	tail_prob_2	0.089	0.241	0.454	0.284	0.129	0.528
FluxEV1e-1	Gauss-DK	tail_prob_1	0.073	0.178	0.648	0.232	0.111	0.528
FluxEV1e-1	Gauss-DK	tail_prob_3	0.09	0.184	0.296	0.211	0.121	0.528
FluxEV1e-10	Gauss-DK	tail_prob_5	0.168	0.339	0.485	0.38	0.218	0.568
FluxEV1e-10	Gauss-DK	tail_prob_4	0.159	0.323	0.503	0.371	0.209	0.568
FluxEV1e-10	Gauss-DK	tail_prob_3	0.131	0.283	0.503	0.334	0.176	0.568
FluxEV1e-10	Gauss-DK	tail_prob_2	0.104	0.249	0.531	0.306	0.144	0.568
FluxEV1e-10	Gauss-DK	best_f1_test	0.473	0.594	0.929	0.634	0.53	0.568
FluxEV1e-10	Gauss-D	tail_prob_5	0.125	0.446	0.596	0.488	0.159	0.512
FluxEV1e-10	Gauss-D	tail_prob_4	0.11	0.421	0.596	0.468	0.142	0.512
FluxEV1e-10	Gauss-D	tail_prob_3	0.107	0.393	0.596	0.441	0.136	0.512
FluxEV1e-10	Gauss-D	tail_prob_2	0.101	0.354	0.596	0.405	0.127	0.512
FluxEV1e-10	Gauss-D	tail_prob_1	0.103	0.301	0.596	0.351	0.127	0.512
FluxEV1e-10	Gauss-D	best_f1_test	0.445	0.606	0.883	0.62	0.503	0.512
FluxEV1e-10	Gauss-DK	tail_prob_1	0.089	0.205	0.596	0.262	0.121	0.568
FluxEV1e-3	Gauss-DK	tail_prob_5	0.125	0.248	0.355	0.277	0.166	0.539
FluxEV1e-3	Gauss-D	tail_prob_5	0.111	0.438	0.596	0.481	0.151	0.549
FluxEV1e-3	Gauss-D	best_f1_test	0.38	0.567	0.806	0.586	0.436	0.549
FluxEV1e-3	Gauss-D	tail_prob_1	0.102	0.356	0.596	0.4	0.138	0.549
FluxEV1e-3	Gauss-D	tail_prob_2	0.108	0.392	0.596	0.437	0.147	0.549
FluxEV1e-3	Gauss-D	tail_prob_3	0.11	0.41	0.596	0.455	0.149	0.549
FluxEV1e-3	Gauss-D	tail_prob_4	0.111	0.425	0.596	0.469	0.151	0.549
FluxEV1e-3	Gauss-DK	best_f1_test	0.44	0.547	0.83	0.565	0.479	0.539
FluxEV1e-3	Gauss-DK	tail_prob_1	0.094	0.221	0.596	0.28	0.134	0.539
FluxEV1e-3	Gauss-DK	tail_prob_2	0.121	0.29	0.475	0.329	0.167	0.539
FluxEV1e-3	Gauss-DK	tail_prob_3	0.106	0.211	0.355	0.245	0.142	0.539
FluxEV1e-3	Gauss-DK	tail_prob_4	0.114	0.229	0.355	0.261	0.153	0.539
FluxEV1e-5	Gauss-DK	tail_prob_5	0.149	0.244	0.336	0.263	0.18	0.566
FluxEV1e-5	Gauss-DK	tail_prob_4	0.14	0.227	0.336	0.247	0.171	0.566
FluxEV1e-5	Gauss-D	best_f1_test	0.379	0.546	0.753	0.553	0.426	0.591
FluxEV1e-5	Gauss-D	tail_prob_1	0.097	0.289	0.491	0.321	0.131	0.591
FluxEV1e-5	Gauss-D	tail_prob_2	0.099	0.329	0.485	0.363	0.137	0.591
FluxEV1e-5	Gauss-D	tail_prob_3	0.102	0.348	0.485	0.38	0.141	0.591
FluxEV1e-5	Gauss-D	tail_prob_4	0.104	0.365	0.485	0.394	0.144	0.591
FluxEV1e-5	Gauss-D	tail_prob_5	0.104	0.375	0.485	0.402	0.145	0.591
FluxEV1e-5	Gauss-DK	best_f1_test	0.416	0.493	0.799	0.52	0.463	0.566
FluxEV1e-5	Gauss-DK	tail_prob_1	0.092	0.182	0.491	0.223	0.13	0.566
FluxEV1e-5	Gauss-DK	tail_prob_2	0.129	0.255	0.401	0.279	0.173	0.566
FluxEV1e-5	Gauss-DK	tail_prob_3	0.129	0.21	0.336	0.231	0.161	0.566
UnivarAutoencoder	Gauss-DK	tail_prob_3	0.488	0.589	0.812	0.628	0.549	**0.77**
UnivarAutoencoder	Gauss-DK	tail_prob_2	0.367	0.493	0.914	0.576	0.456	**0.77**
UnivarAutoencoder	Gauss-DK	tail_prob_1	0.253	0.307	**1.0**	0.392	0.332	**0.77**
UnivarAutoencoder	Gauss-DK	best_f1_test	**0.782**	**0.88**	0.951	**0.874**	**0.798**	**0.77**
UnivarAutoencoder	Gauss-D	tail_prob_5	0.433	0.684	0.92	0.743	0.508	0.63
UnivarAutoencoder	Gauss-D	best_f1_test	0.719	0.86	0.948	0.847	0.752	0.63
UnivarAutoencoder	Gauss-D	tail_prob_3	0.32	0.539	0.981	0.628	0.402	0.63
UnivarAutoencoder	Gauss-D	tail_prob_2	0.292	0.452	0.981	0.54	0.37	0.63
UnivarAutoencoder	Gauss-D	tail_prob_1	0.272	0.365	**1.0**	0.445	0.343	0.63
UnivarAutoencoder	Gauss-DK	tail_prob_4	0.575	0.63	0.775	0.649	0.609	**0.77**
UnivarAutoencoder	Gauss-D	tail_prob_4	0.382	0.625	0.957	0.705	0.465	0.63
UnivarAutoencoder	Gauss-DK	tail_prob_5	0.563	0.61	0.728	0.627	0.592	**0.77**

Table A.6: Evaluation metrics on SMD dataset with Gauss-D and Gauss-D-DK aggregation with Best-F-score and Tail-probability thresholding

Algorithm	Aggregation	Thresholding method	Point-wise Precision	Normalized Precision	Recall	F-score	F_c-score	AUC
AR	Gauss-D	best_f1_test	0.398	0.815	0.662	0.685	0.449	0.632
AR	Gauss-DK	tail_prob_5	0.295	0.637	0.491	0.499	0.332	0.721
AR	Gauss-DK	tail_prob_4	0.269	0.585	0.523	0.496	0.321	0.721
AR	Gauss-DK	tail_prob_3	0.238	0.526	0.593	0.502	0.311	0.721
AR	Gauss-DK	tail_prob_1	0.083	0.137	0.955	0.228	0.148	0.721
AR	Gauss-DK	best_f1_test	0.425	0.749	0.605	0.622	0.449	0.721
AR	Gauss-DK	tail_prob_2	0.196	0.44	0.755	0.514	0.291	0.721
AR	Gauss-D	tail_prob_4	0.237	0.634	0.695	0.624	0.33	0.632
AR	Gauss-D	tail_prob_3	0.208	0.569	0.783	0.623	0.309	0.632
AR	Gauss-D	tail_prob_2	0.156	0.439	0.881	0.552	0.253	0.632
AR	Gauss-D	tail_prob_1	0.066	0.132	0.969	0.221	0.122	0.632
AR	Gauss-D	tail_prob_5	0.259	0.673	0.626	0.602	0.337	0.632
AR_diff	Gauss-D	tail_prob_4	0.226	0.607	0.686	0.594	0.32	0.657
AR_diff	Gauss-D	best_f1_test	0.376	0.827	0.647	0.693	0.441	0.657
AR_diff	Gauss-D	tail_prob_1	0.07	0.134	0.968	0.223	0.127	0.657
AR_diff	Gauss-D	tail_prob_2	0.148	0.425	0.894	0.538	0.246	0.657
AR_diff	Gauss-D	tail_prob_3	0.197	0.55	0.778	0.598	0.301	0.657
AR_diff	Gauss-D	tail_prob_5	0.248	0.643	0.611	0.569	0.33	0.657
AR_diff	Gauss-DK	best_f1_test	0.357	0.721	0.66	0.648	0.43	0.745
AR_diff	Gauss-DK	tail_prob_1	0.083	0.135	0.955	0.225	0.148	0.745
AR_diff	Gauss-DK	tail_prob_2	0.183	0.411	0.745	0.478	0.276	0.745
AR_diff	Gauss-DK	tail_prob_3	0.226	0.508	0.605	0.488	0.304	0.745
AR_diff	Gauss-DK	tail_prob_4	0.256	0.572	0.536	0.485	0.316	0.745
AR_diff	Gauss-DK	tail_prob_5	0.281	0.625	0.511	0.489	0.328	0.745
BiDSPOT1e-1	Gauss-DK	tail_prob_4	0.288	0.52	0.854	0.605	0.393	0.796
BiDSPOT1e-1	Gauss-DK	tail_prob_3	0.234	0.419	0.898	0.535	0.345	0.796
BiDSPOT1e-1	Gauss-DK	tail_prob_2	0.147	0.258	0.952	0.375	0.239	0.796
BiDSPOT1e-1	Gauss-DK	tail_prob_1	0.069	0.082	0.989	0.144	0.124	0.796
BiDSPOT1e-1	Gauss-DK	best_f1_test	0.554	0.79	0.744	0.735	0.587	0.796
BiDSPOT1e-1	Gauss-D	tail_prob_5	0.301	0.595	0.891	0.681	0.416	0.759
BiDSPOT1e-1	Gauss-D	tail_prob_4	0.274	0.534	0.923	0.641	0.393	0.759
BiDSPOT1e-1	Gauss-D	tail_prob_3	0.216	0.426	0.941	0.55	0.327	0.759
BiDSPOT1e-1	Gauss-D	tail_prob_2	0.133	0.256	0.965	0.376	0.219	0.759
BiDSPOT1e-1	Gauss-D	tail_prob_1	0.067	0.085	0.995	0.15	0.121	0.759
BiDSPOT1e-1	Gauss-D	best_f1_test	0.609	0.879	0.799	0.821	0.649	0.759
BiDSPOT1e-1	Gauss-DK	tail_prob_5	0.326	0.575	0.807	0.626	0.419	0.796
BiDSPOT1e-10	Gauss-D	best_f1_test	0.7	0.815	0.58	0.601	0.568	0.614
BiDSPOT1e-10	Gauss-D	tail_prob_1	0.043	0.071	0.923	0.125	0.08	0.614
BiDSPOT1e-10	Gauss-DK	tail_prob_5	0.352	0.546	0.454	0.424	0.299	0.632
BiDSPOT1e-10	Gauss-DK	tail_prob_4	0.274	0.452	0.475	0.398	0.265	0.632
BiDSPOT1e-10	Gauss-DK	tail_prob_2	0.077	0.187	0.673	0.265	0.128	0.632
BiDSPOT1e-10	Gauss-DK	tail_prob_1	0.045	0.07	0.913	0.124	0.083	0.632
BiDSPOT1e-10	Gauss-DK	tail_prob_3	0.167	0.325	0.544	0.352	0.204	0.632
BiDSPOT1e-10	Gauss-D	tail_prob_5	0.347	0.598	0.544	0.509	0.34	0.614
BiDSPOT1e-10	Gauss-D	tail_prob_4	0.264	0.502	0.589	0.485	0.291	0.614
BiDSPOT1e-10	Gauss-D	tail_prob_3	0.158	0.357	0.634	0.404	0.208	0.614
BiDSPOT1e-10	Gauss-D	tail_prob_2	0.073	0.202	0.73	0.288	0.124	0.614
BiDSPOT1e-10	Gauss-DK	best_f1_test	0.589	0.72	0.54	0.542	0.494	0.632
BiDSPOT1e-3	Gauss-DK	tail_prob_1	0.062	0.075	0.983	0.133	0.112	0.774
BiDSPOT1e-3	Gauss-DK	tail_prob_5	0.49	0.684	0.74	0.654	0.519	0.774
BiDSPOT1e-3	Gauss-DK	tail_prob_4	0.424	0.618	0.79	0.634	0.487	0.774
BiDSPOT1e-3	Gauss-DK	tail_prob_3	0.307	0.503	0.851	0.571	0.402	0.774
BiDSPOT1e-3	Gauss-DK	tail_prob_2	0.155	0.301	0.935	0.409	0.244	0.774
BiDSPOT1e-3	Gauss-DK	best_f1_test	0.747	0.895	0.748	0.789	0.722	0.774
BiDSPOT1e-3	Gauss-D	tail_prob_2	0.141	0.305	0.961	0.421	0.227	0.739
BiDSPOT1e-3	Gauss-D	tail_prob_4	0.391	0.64	0.875	0.688	0.485	0.739
BiDSPOT1e-3	Gauss-D	tail_prob_3	0.277	0.517	0.926	0.608	0.38	0.739
BiDSPOT1e-3	Gauss-D	tail_prob_1	0.06	0.076	0.991	0.135	0.108	0.739
BiDSPOT1e-3	Gauss-D	best_f1_test	0.777	0.938	0.795	0.846	0.77	0.739
BiDSPOT1e-3	Gauss-D	tail_prob_5	0.464	0.703	0.828	0.718	0.542	0.739
BiDSPOT1e-5	Gauss-D	tail_prob_4	0.357	0.591	0.728	0.595	0.388	0.688
BiDSPOT1e-5	Gauss-D	best_f1_test	0.754	0.89	0.68	0.742	0.683	0.688
BiDSPOT1e-5	Gauss-D	tail_prob_1	0.052	0.071	0.989	0.126	0.095	0.688
BiDSPOT1e-5	Gauss-D	tail_prob_2	0.104	0.238	0.864	0.347	0.176	0.688
BiDSPOT1e-5	Gauss-D	tail_prob_3	0.222	0.442	0.793	0.522	0.303	0.688
BiDSPOT1e-5	Gauss-D	tail_prob_5	0.45	0.701	0.675	0.626	0.439	0.688
BiDSPOT1e-5	Gauss-DK	best_f1_test	0.71	0.833	0.607	0.667	0.618	0.713
BiDSPOT1e-5	Gauss-DK	tail_prob_1	0.053	0.07	0.981	0.125	0.097	0.713
BiDSPOT1e-5	Gauss-DK	tail_prob_2	0.114	0.229	0.82	0.331	0.188	0.713
BiDSPOT1e-5	Gauss-DK	tail_prob_3	0.233	0.406	0.712	0.465	0.296	0.713
BiDSPOT1e-5	Gauss-DK	tail_prob_4	0.366	0.557	0.646	0.531	0.368	0.713
BiDSPOT1e-5	Gauss-DK	tail_prob_5	0.457	0.654	0.581	0.548	0.413	0.713
FluxEV1e-1	Gauss-DK	tail_prob_3	0.218	0.491	0.976	0.615	0.34	0.727
FluxEV1e-1	Gauss-DK	tail_prob_2	0.172	0.39	0.988	0.522	0.279	0.727
FluxEV1e-1	Gauss-DK	tail_prob_1	0.096	0.185	0.995	0.293	0.169	0.727
FluxEV1e-1	Gauss-DK	best_f1_test	0.629	0.91	0.732	0.794	0.636	0.727
FluxEV1e-1	Gauss-D	tail_prob_5	0.232	0.617	0.98	0.727	0.357	0.662
FluxEV1e-1	Gauss-D	tail_prob_4	0.208	0.572	0.986	0.691	0.327	0.662
FluxEV1e-1	Gauss-D	tail_prob_3	0.182	0.508	0.988	0.636	0.292	0.662
FluxEV1e-1	Gauss-D	tail_prob_2	0.144	0.409	0.992	0.545	0.24	0.662
FluxEV1e-1	Gauss-D	tail_prob_1	0.087	0.211	0.995	0.328	0.155	0.662
FluxEV1e-1	Gauss-D	best_f1_test	0.6	0.945	0.782	0.844	0.65	0.662
FluxEV1e-1	Gauss-DK	tail_prob_4	0.251	0.551	0.952	0.662	0.378	0.727
FluxEV1e-1	Gauss-DK	tail_prob_5	0.279	0.593	0.938	0.692	0.409	0.727
FluxEV1e-10	Gauss-D	best_f1_test	0.665	0.959	0.79	0.859	0.709	0.624
FluxEV1e-10	Gauss-D	tail_prob_3	0.255	0.63	0.974	0.732	0.382	0.624
FluxEV1e-10	Gauss-D	tail_prob_4	0.3	0.697	0.962	0.782	0.435	0.624
FluxEV1e-10	Gauss-D	tail_prob_5	0.331	0.739	0.944	0.804	0.468	0.624

Continued on next page

Table A.6: Evaluation metrics on SMD dataset with Gauss-D and Gauss-D-DK aggregation with Best-F-score and Tail-probability thresholding

Algorithm	Aggregation	Thresholding method	Point-wise Precision	Normalized Precision	Recall	F-score	F_c-score	AUC
FluxEV1e-10	Gauss-DK	best_f1_test	0.668	0.933	0.782	0.837	0.692	0.697
FluxEV1e-10	Gauss-DK	tail_prob_1	0.066	0.12	0.993	0.202	0.122	0.697
FluxEV1e-10	Gauss-DK	tail_prob_2	0.193	0.434	0.977	0.561	0.302	0.697
FluxEV1e-10	Gauss-DK	tail_prob_3	0.29	0.602	0.944	0.698	0.422	0.697
FluxEV1e-10	Gauss-DK	tail_prob_4	0.342	0.668	0.909	0.735	0.475	0.697
FluxEV1e-10	Gauss-DK	tail_prob_5	0.374	0.699	0.867	0.737	0.496	0.697
FluxEV1e-10	Gauss-D	tail_prob_1	0.06	0.132	0.993	0.22	0.11	0.624
FluxEV1e-10	Gauss-D	tail_prob_2	0.169	0.459	0.989	0.587	0.27	0.624
FluxEV1e-3	Gauss-D	tail_prob_5	0.349	0.747	0.961	0.818	0.488	0.626
FluxEV1e-3	Gauss-DK	tail_prob_3	0.3	0.606	0.948	0.703	0.433	0.699
FluxEV1e-3	Gauss-D	best_f1_test	0.681	**0.965**	0.807	0.871	0.724	0.626
FluxEV1e-3	Gauss-D	tail_prob_1	0.059	0.128	0.995	0.216	0.109	0.626
FluxEV1e-3	Gauss-D	tail_prob_2	0.179	0.471	0.989	0.598	0.283	0.626
FluxEV1e-3	Gauss-D	tail_prob_3	0.263	0.634	0.975	0.735	0.39	0.626
FluxEV1e-3	Gauss-D	tail_prob_4	0.313	0.704	0.964	0.787	0.449	0.626
FluxEV1e-3	Gauss-DK	best_f1_test	0.688	0.936	0.782	0.841	0.708	0.699
FluxEV1e-3	Gauss-DK	tail_prob_1	0.066	0.118	0.995	0.201	0.121	0.699
FluxEV1e-3	Gauss-DK	tail_prob_2	0.202	0.448	0.979	0.575	0.315	0.699
FluxEV1e-3	Gauss-DK	tail_prob_4	0.354	0.673	0.921	0.744	0.488	0.699
FluxEV1e-3	Gauss-DK	tail_prob_5	0.39	0.713	0.903	0.766	0.521	0.699
FluxEV1e-5	Gauss-DK	tail_prob_5	0.386	0.713	0.908	0.765	0.515	0.706
FluxEV1e-5	Gauss-DK	tail_prob_4	0.354	0.674	0.924	0.745	0.487	0.706
FluxEV1e-5	Gauss-D	tail_prob_1	0.062	0.124	**0.996**	0.21	0.113	0.634
FluxEV1e-5	Gauss-D	tail_prob_2	0.185	0.481	0.988	0.607	0.29	0.634
FluxEV1e-5	Gauss-D	tail_prob_3	0.266	0.637	0.978	0.739	0.394	0.634
FluxEV1e-5	Gauss-D	tail_prob_4	0.316	0.704	0.966	0.786	0.451	0.634
FluxEV1e-5	Gauss-D	tail_prob_5	0.346	0.743	0.95	0.807	0.481	0.634
FluxEV1e-5	Gauss-DK	best_f1_test	0.665	0.935	0.779	0.84	0.692	0.706
FluxEV1e-5	Gauss-DK	tail_prob_1	0.068	0.113	0.995	0.193	0.123	0.706
FluxEV1e-5	Gauss-DK	tail_prob_2	0.215	0.458	0.982	0.585	0.329	0.706
FluxEV1e-5	Gauss-DK	tail_prob_3	0.302	0.607	0.948	0.703	0.432	0.706
FluxEV1e-5	Gauss-D	best_f1_test	0.651	0.957	0.838	0.884	0.716	0.634
UnivarAutoencoder	Gauss-DK	best_f1_test	0.765	0.915	0.875	0.89	0.808	**0.856**
UnivarAutoencoder	Gauss-DK	tail_prob_3	0.327	0.518	0.975	0.634	0.462	**0.856**
UnivarAutoencoder	Gauss-DK	tail_prob_2	0.232	0.365	0.992	0.494	0.352	**0.856**
UnivarAutoencoder	Gauss-DK	tail_prob_1	0.101	0.123	0.993	0.207	0.176	**0.856**
UnivarAutoencoder	Gauss-D	tail_prob_5	0.423	0.671	0.967	0.759	0.56	0.825
UnivarAutoencoder	Gauss-DK	tail_prob_4	0.407	0.618	0.953	0.708	0.541	**0.856**
UnivarAutoencoder	Gauss-D	tail_prob_3	0.297	0.511	0.992	0.631	0.429	0.825
UnivarAutoencoder	Gauss-D	tail_prob_2	0.208	0.36	0.993	0.49	0.323	0.825
UnivarAutoencoder	Gauss-D	tail_prob_1	0.098	0.128	0.995	0.215	0.17	0.825
UnivarAutoencoder	Gauss-D	best_f1_test	**0.808**	0.955	0.912	**0.928**	**0.848**	0.825
UnivarAutoencoder	Gauss-D	tail_prob_4	0.37	0.609	0.985	0.712	0.507	0.825
UnivarAutoencoder	Gauss-DK	tail_prob_5	0.465	0.679	0.934	0.747	0.591	**0.856**