Study Thesis

Anomaly detection in data sets generated by the CERN Radiation Monitoring Electronic system CROME to develop predictive maintenance algorithms

Erkennung von Anomalien in Datensätzen des elektronischen Überwachungssystems für Radioaktivität CROME am CERN zur Entwicklung prädiktiver Wartungsalgorithmen

submitted by

Felix Waldhauser, B.Sc.

Geneva / Stuttgart

October 2021

Degree Program	M.Sc. Mechanical Engineering
Student Number	3249826
Examined by	Prof. Dr.-Ing. Bernd Bertsche
Supervised by (CERN)	Dr.-Ing. Hamza Boukabache
Supervised by (IMA)	Dr.-Ing. Martin Dazer
Submitted on	29.10.2021

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
ARCON	ARea CONtroller System
CAU	CROME Alarm Unit
CERN	European Organization for Nuclear Research
CJB	CROME Junction Box
CMPU	CROME Monitoring and Processing Unit
CROME	CERN Radiation Monitoring Electronics
CUPS	CROME Uninterruptible Power Supply
CWT	Continuous Wavelet Transform
DFT	Discrete Fourier Transform
DWT	Discrete Wavelet Transform
EDA	Exploratory Data Analysis
EEG	Electroencephalography
FFT	Fast Fourier Transform
FMEDA	Failure Modes, Effects and Diagnostic Analysis
FPGA	Field Programmable Gate Array
FTA	Fault Tree Analysis
GBDT	Gradient Boosted Decision Trees
iForest	Isolation Forest
iTree	Isolation Tree
k-NN	k-Nearest Neighbors
LSTM	Long Short-Term Memory
LSTM-AE	Long Short-Term Memory Autoencoder
MAE	Mean Absolute Error
ML	Machine Learning
MRA-WT	Multi-Resolution Analysis Wavelet Transform
MSE	Mean Squared Error
NXCALS	Next CERN Accelerators Logging Service
PFH	Probability of Failures per Hour
PHM	Prognostics and Health Management
REMUS	Radiation and Environment Monitoring Unified Supervision
RMS	Root Mean Square
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
RUL	Remaining Useful Life
SNR	Signal-to-Noise Ratio
SoC	System-on-Chip
STFT	Short-Time Fourier Transform
SWAN	Service for Web based Analysis

Kurzfassung

Eine Vielzahl von Experimenten und Teilchenbeschleunigern am CERN, dem größten Labor für Teilchenphysik der Welt, erfordern eine genaue Überwachung ionisierender Strahlung zum Schutz von Mensch und Umwelt. Dies wird durch das am CERN entwickelte System zur Überwachung von Radioaktivität CROME realisiert, das unter anderem für eine zuverlässige Überwachung der Dosisleistung und eine permanente Datenaufzeichnung verantwortlich ist.

Datengetriebene Methoden ermöglichen die Planung von Instandhaltungsmaßnahmen basierend auf Ausfallvorhersagen, um so Ausfallraten sicherheitskritischer Systeme wie CROME auf einem äußerst niedrigen Niveau zu halten. Diese Arbeit soll daher die Eignung der von den CROME-Geräten generierten Datensätze zur Ausfallvorhersage bewerten und die Anwendbarkeit ausgewählter Algorithmen zur Anomalie-Erkennung beurteilen.
Genauer gesagt wurde ein Verfahren zur Anomalie-Erkennung vorgestellt, mit dem Vorläufer von Ausfällen und Fehlfunktionen identifiziert werden können. Erkannte Anomalien können dann zur Zustandsüberwachung oder zur Systemoptimierung genutzt werden. Der vorgestellte Prozess zur Anomalie-Erkennung kombiniert die Extraktion von Rauschsignalen mittels Wavelet-Transformation mit Algorithmen des unüberwachten maschinellen Lernens, um eine Vielzahl von Anomalien erkennbar zu machen. Zudem wurden statistische Merkmale, die das Rauschen und andere Signaleigenschaften beschreiben, als Input für den verwendeten Isolation-Forest-Algorithmus gewählt. Dieser modell-basierte Algorithmus detektierte Proben mit unüblichen Signalverläufen wie beispielsweise Spitzen oder Nulldurchgänge des Signals der Dosisleistung. Außerdem wurde ein Long Short-Term Memory-Autoencoder eingesetzt, um zeitliche Abhängigkeiten in den aufgezeichneten Messdaten zu erlernen. Vom Autoencoder erkannte Proben zeigten unerwartete Korrelationen oder unübliche Messwerte. Dennoch bleibt zu evaluieren, ob es sich bei den festgestellten Anomalien um tatsächliche Vorläufer von Ausfällen oder lediglich um seltene Ereignisse handelt, um die Ergebnisse zur Entwicklung von Strategien zur prädiktiven Wartung nutzen zu können.

Da es in der Literatur keine klaren Richtlinien zur Konfiguration der Wavelet-Transformation gibt, wurde ein Verfahren entwickelt, um verschiedene Konfigurationsmöglichkeiten zu vergleichen. Dabei wurde ein Klassifizierungsalgorithmus darauf trainiert, synthetisch veränderte Proben mithilfe statistischer Merkmale zu erkennen, die für extrahierte Rauschsignale berechnet wurden. Der vorgestellte Prozess ermöglicht die Auswahl der am besten geeigneten Konfiguration für Anwendungen der Wavelet-Transformation zur Rauschextraktion, indem er reale Signalproben verwendet und eine Anpassung der synthetischen Veränderung der Proben erlaubt.

Darüber hinaus wurden zwei Möglichkeiten zur Integration neuer Daten in den Prozess zur Anomalie-Erkennung vorgestellt, bei denen die Daten entweder vor oder nach dem erstmaligen Training des Algorithmus zur Verfügung gestellt werden. Basierend auf diesen Erkenntnissen wurde ein Verfahren vorgeschlagen, das durch eine Feedback-Schleife die Verwendung von Systemwissen in der Anomalie-Erkennung erlaubt.

Abstract

Several experiments and particle accelerators at CERN, the largest laboratory of high-energy particle physics in the world, require accurate monitoring of ionizing stray radiation to protect people and the environment. To meet this requirement, the new in-house developed and produced radiation monitoring system CROME is responsible for reliable monitoring of the ambient dose rate and permanent data logging, among others.

Data-driven methods enable planning of maintenance actions based on failure predictions to keep failure rates of safety-critical systems such as CROME at an extremely low level and ensure high system availability and reliability. The goal of this thesis is therefore to evaluate the usefulness of the datasets generated by CROME devices for failure prediction and to examine the applicability of selected anomaly detection algorithms.
More precisely, an anomaly detection process to identify failure precursors and malfunctions is presented. Detected anomalies can thus be used for condition monitoring or system improvement. The proposed process combines noise extraction using wavelet transform and unsupervised anomaly detection algorithms to increase the detectability of the anomalous system behavior. Moreover, statistical features describing the noise and other signal characteristics were used as input to the employed isolation forest algorithm. This model-based algorithm detected samples with unusual signal shapes such as spikes or zero crossings of the dose rate signal. In addition, a long short-term memory autoencoder model was employed to learn temporal dependencies in the measurement observations. Samples considered as anomalous by the autoencoder showed unexpected correlations or unusual measurement values.
Nevertheless, system experts still need to evaluate whether the anomalies detected are actual failure precursors or solely infrequent occurrences to make the results useful for the development of predictive maintenance strategies.

Due to missing guidelines in the literature on how to configure the wavelet transform, a signal classification process was developed to compare various configurations by analyzing the performance of a supervised machine learning algorithm. This algorithm was trained to detect synthetically modified samples based on statistical measures calculated for extracted noise signals. The proposed signal classification process allows the selection of the most appropriate configuration of the wavelet transform for a given noise extraction use case by relying on real signal samples and providing possibilities to customize the signal modification.

Additionally, two approaches to integrate new data in the anomaly detection process, either before or after the initial training of the model, were presented. Based on these findings, a feedback loop allowing the integration of system knowledge was proposed for further development of the presented anomaly detection process.

Chapter 1

Introduction

In the first part of this introduction, the underlying use case as well as the motivation of the performed analysis is presented. The second part defines the objectives of this thesis based on requirements and expected difficulties.

1.1 Motivation

In order for systems to fulfill safety-critical functions, it is crucial to keep their failure rates at an extremely low level through appropriate maintenance measures. If operational data is available, data-driven methods can be used to monitor the system's condition to evaluate the degradation and to detect malfunctions. This requires the identification of characteristics related to the system's degradation and the detection of failure precursors. Planning maintenance actions based on failure predictions helps to reduce unexpected failures and to increase system availability and reliability.

Radiation protection systems are safety-critical and essential for protecting people and the environment from unjustified exposure to radiation. This is especially important at the European Organization for Nuclear Research (CERN) that hosts the largest laboratory for high-energy particle physics in the world. Collisions of high-energy particles with other particles or stable matter may emit ionizing stray radiation in a variety of experiments at CERN. Therefore, not only to ensure a safe working environment at CERN, but also to comply with safety regulations, accurate monitoring of the ambient dose rate is required.
The system analyzed in this thesis, called CROME, is the new in-house developed radiation monitoring system which is consecutively installed at various high and low radiation areas at CERN. All CROME devices monitor the ambient dose rate and log the dose rate as well as other measurement data continuously on databases. Reliable monitoring of the dose rate and thus ensuring the safety of people working at CERN is only possible if all radiation monitoring devices are properly maintained, thereby keeping failure rates at an extremely low level. However, since many devices are located in restricted, high-radiation areas where engineers cannot access them, it is necessary to monitor the condition of the devices remotely, e.g. using data-driven methods.

1.2 Objectives

This thesis aims at the development of a process to detect anomalies in datasets generated by CROME devices to enable remote, data-driven condition monitoring of the CROME system. Anomalies of interest do not only concern hardware but also involve measurement processing, data logging and device communication issues. The results of the analysis presented will subsequently be used to plan maintenance actions and improve the system based on detected failure precursors or malfunctions. Therefore, the necessary steps to be performed in this thesis are:

- Select and import appropriate datasets
- Pre-process data to increase the detectability of anomalies
- Detect malfunctions in device communication and data logging
- Apply suitable anomaly detection algorithms
- Analyze and interpret detected anomalies
- Evaluate and compare the algorithms used

Since there is currently no known system failure, there is no precise definition of anomalies, nor is the frequency or existence of anomalies known. In addition, no dataset that represents the healthy state of the system is available either. To address these challenges, unsupervised learning algorithms are used to detect anomalies.
To limit the scope of the analysis, the anomaly detection process is focused on noise in signals. Hence, the wavelet transform is chosen to extract the noise from the imported raw signals. However, there are currently no clear guidelines in the literature on how to configure the wavelet transform for noise extraction.
The presented difficulties and limitations result in the following additional tasks:

- Perform hardware tests to increase the understanding of the system's healthy state
- Develop a process to compare various configurations of the wavelet transform and to select the most suitable one for this noise extraction use case
- Integrate statistical measures describing the noise in signals into the anomaly detection process

To ensure wide applicability of the proposed anomaly detection process, it is necessary to include as many scenarios of the device's behavior as possible in the analysis. In addition, it is important to integrate all available variables in the analysis, since it is not known how anomalies can be identified. Besides that, the data acquisition should be modular and dynamic to easily apply the proposed process to other datasets and to run scheduled processes on data from operational devices.
Furthermore, pre-processing the retrieved data aims at the development of useful features helping to distinguish between anomalous and normal observations in order to enhance the performance and the reliability of the anomaly detection algorithms. This includes the quantification of the noise extracted from raw signals using the configured wavelet transforms.

This analysis is intended as a feasibility study to gain insight into the usefulness of the data for anomaly detection and predictive maintenance, and to evaluate the applicability of selected anomaly detection algorithms. Therefore, the results of the algorithms are analyzed in terms of the types of samples that are classified as anomalous and the behaviour of the algorithms. Moreover, results of the algorithms used are compared to derive recommendations for further use and to outline possible improvements.

Chapter 2

Theoretical Background

This chapter provides an overview of the theoretical principles used in this thesis. To increase the detectability of anomalies in the used datasets, signal processing steps are necessary. First, signal processing is explained in general and basic requirements for appropriate signal recording are discussed. Then, time-frequency transforms are introduced and requirements for the configuration of the wavelet transform are outlined. Time-frequency transforms allow further signal processing to unveil specific signal characteristics, e.g. noise in signals. Subsequently, statistical features that describe the characteristics of the processed signals are explained. In addition, the idea of Prognostics and Health Management (PHM) as well as various maintenance strategies are presented. Artificial Intelligence (AI) and general Machine Learning (ML) principles are presented in section 2.4. ML algorithms are required to enable data-driven maintenance strategies based on the processed signals and characteristics. The last section of this chapter introduces the concept of anomaly detection and describes two algorithms used for anomaly detection in this thesis.

2.1 Signal Processing

This section introduces basic ideas of signal processing and gives an overview over available methods and approaches.

Generally speaking, a signal is a function that transfers a message or information [1, 2, 3]. In the field of electronics, a signal is referred to as a description of a parameter developing in time to transmit information either analogously or digitally [4, 5].
A continuous-time signal is a function of continuous time, i.e. it is defined at all instances of time [4]. If both time and amplitude are continuous variables, a signal is called *analog signal* [1, 5]. Discrete-time signals are, however, in general sequences of numerical values with infinite length and are generated by sampling of continuous-time signals at a constant rate [4]. Thus, the elements of a discrete sequence are separated by a constant time increment T_S [4]. A signal is called *digital signal* if both time and amplitude are discrete variables [5].
In order to be capable of sampling a correct discrete representation from a continuous signal the sampling interval T_S must be chosen properly. A correct representation is achieved if a continuous signal can be reconstructed from its discrete version [4, 6]. The Sampling Theorem, also know as *Nyquist-Shannon Theorem*, sets the lower limit for T_S according to B_f which is the highest frequency of the spectrum of the analog signal to be sampled (i.e. upper bandlimit) [4]. Thus, a representative sampling is only possible by satisfying the following inequality, called *Nyquist criterion* according to [4]:

$$B_f < \frac{1}{2T_S} = f_{Ny} \tag{2.1}$$

In other words: the highest frequency in a signal B_f must not exceed the Nyquist Frequency f_{Ny}.

The term *signal processing* is often used to refer to methods and techniques to represent, analyze or modify a signal to make its interpretation easier [1, 4, 5]. Besides that, the term also covers the separation of various contents of a signal (e.g. separate noise from information), or the extraction of components containing information about the state or condition of the underlying system, which can then be used to make decisions about maintenance actions [4, 7].

2.1.1 Time-Frequency Analysis

One main method of signal processing that is widely used to analyze the composition of various types of signals is the time-frequency analysis. It compromises numerous mathematical techniques that use time-frequency shifts (i.e. translations and modulations) to analyze a signal in both time and frequency domain simultaneously [8]. The main goal of time-frequency analyses is according to [7, p. 154] to "determine the energy concentration along the frequency axis at a given time instant". In other words, time-frequency analysis is being used to find a representation combining features of both time and frequency domain in one function called time-frequency representation [7, 8]. The following paragraphs will briefly cover the Fourier transform, Gabor transform and wavelet transform as popular methods of time-frequency analyses.

Fourier Transform

The Fourier transform originated from the Fourier integral theorem as advancement of the Fourier series and is considered as one of the most important theorems in mathematical sciences [9]. The Fourier series is used to represent a periodic signal or function by an infinite but discrete set of harmonic components (i.e. sine and cosine waves). Equation 2.2 shows the Fourier series of a function $f(x)$ being defined on the interval $(-l, l)$, according to [9].

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{i\frac{n\pi}{l}x} \tag{2.2}$$

Here, c_n are the Fourier coefficients defined by Equation 2.3, according to [9, 10].

$$c_n = \frac{1}{2l}\int_{-l}^{l} f(x)e^{-i\frac{n\pi}{l}x}\,dx \tag{2.3}$$

Figure 2.1 illustrates the Fourier series approximation of a square wave for various numbers of coefficients as shown in [11, 12].

However, when applying the Fourier integral theorem to a non-periodic signal, one obtains a signal representation that consists of harmonic components with varying frequency and amplitude [9]. Supposing the function $f(x)$ to be non-periodic and let $l \to \infty$ leads to the Fourier integral theorem as stated in Equation 2.4 according to [9] where ω is the frequency of the harmonic components.

$$f(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{i\omega x}\,d\omega \int_{-\infty}^{\infty} e^{-i\omega t} f(t)\,dt \tag{2.4}$$

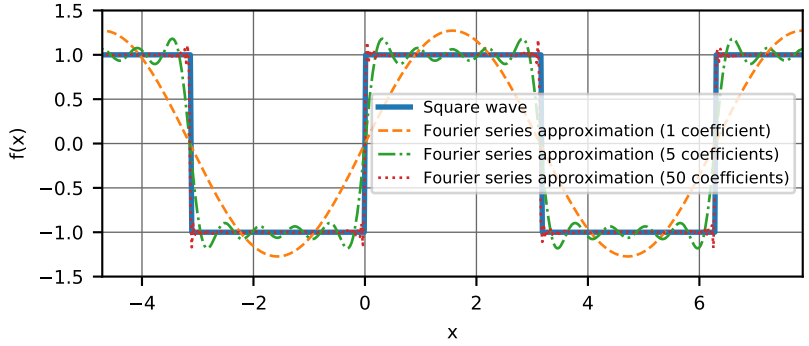


Figure 2.1: Square wave approximated by Fourier series with 1, 5 or 50 coefficients

Consequently, the Fourier transform decomposes a stationary (i.e. time-independent) signal into frequency components using superposition of sine and cosine functions called *sinusoids* leading to the Fourier spectrum [9, 13, 14, 15, 16]. According to [9, 17], the Fourier transform of a continuous signal $f(t)$ is given by Equation 2.5 with $\hat{f}(\omega)$ being a function of the frequency ω. It is most suitable for real-time processing of stationary, harmonic signals if local information is not required since Fourier transforms are time-independent and thus no local information is obtained [9, 14, 16, 18]. This means that the Fourier transform of a signal does not contain any time information of frequency events. Additionally, the Fourier transform allows either to analyze signals in the time or frequency domain but not in both domains simultaneously [9].

$$\mathscr{F}\{f(t)\} = \hat{f}(\omega) = \int_{-\infty}^{\infty} e^{-i\omega t} f(t)\, dt \tag{2.5}$$

The Discrete Fourier Transform (DFT) allows the application of the Fourier transform to discrete signals or functions [4, 9, 14]. In this case, the integral of the continuous Fourier transform (Equation 2.5) is approximated by a sum with a finite number of components supposing the function $f(t_k)$ being sampled at sampling points t_k with $k = 0, 1, .., N$ [9]. The DFT is according to [4, 9, 14] expressed as:

$$\hat{f}(n) = \sum_{k=0}^{N-1} f(t_k) e^{-\frac{2\pi i}{N} kn} \tag{2.6}$$

Where n is an integer related to the frequency $\omega_n = \frac{2\pi n}{b-a}$ and $b = t_{N-1}, a = t_0$ [9].

The Fast Fourier Transform (FFT) optimizes the calculation of the DFT and allows a reduction of the computational effort [4, 9] which made the DFT popular in various fields of applications [4, 9]. There is a variety of FFT methods existing, leading to a reduction of the computational effort from $\mathcal{O}(N^2)$ of DFT to $\mathcal{O}(N \log_2 N)$ of FFT [4, 9].

Gabor Transform

Since most of the signals in real applications are non-stationary (i.e. time-dependent), time-locality of time-frequency analyses is an essential requirement for complete representations of such signals [9]. The Gabor transform is a derivative of the Fourier transform, that localizes the Fourier transform by applying it through a moving window of fixed-size to obtain local information of the frequency components [9, 14, 15, 18]. This leads to a subdivision of the signal into small, sequential, and overlapping data frames in which the signal can be considered stationary, allowing the application of the FFT [16, 19, 20]. According to [9],

Equation 2.7 decomposes the signal $f(t)$ into elementary waveforms that are local in time and frequency using the window function $g(\tau)$ [9, 19]. Under certain conditions (e.g. if the window g is real and symmetric, and thus $g(\tau) = g(-\tau)$), the Gabor transform is also called Short-Time Fourier Transform (STFT) [9].

$$\mathcal{G}\{f(t)\} = \tilde{f}_g(t,\omega) = \int_{-\infty}^{\infty} f(\tau)g(\tau - t)e^{-i\omega\tau}\, d\tau \tag{2.7}$$

Depending on the selection of the window width, the Gabor transform or STFT can either lead to a good resolution for high or low frequencies but not for both simultaneously [19]. This means that there is always a tradeoff between time and frequency resolution, since both cannot be high at the same time [16, 19, 21]. Principally, a large window size leads to a good resolution in the frequency domain and a small window size leads to a good resolution in the time domain [19, 21]. Being restricted to a fixed window size could cause constraints to some applications [21].

Wavelet Transform

A common approach to address the tradeoff between resolution in time and frequency domain is the wavelet transform [16]. The word "wavelet" refers to small wave functions that are used in this time-frequency transform method [22]. Wavelets are functions generated by scaling and translation of a single function called *mother wavelet* using Equation 2.8 [9, 23]. $\Psi(t)$ is the mother wavelet function, a the scaling parameter that affects the compression of the function, and b the translation parameter corresponding to the location of the wavelet [9, 23].

$$\Psi_{a,b}(t) = \frac{1}{\sqrt{|a|}}\Psi\left(\frac{t-b}{a}\right), \quad a, b \in \mathbb{R}, a \neq 0 \tag{2.8}$$

Classifying conditions for wavelet functions are continuity, the mean of the amplitude being equal to zero as well as finite duration and energy [15, 23, 24].

Several types of wavelet functions can be used as mother wavelets such as Daubechies (db), Haar (haar), Symlets (sym) and Coiflets (coif) [25] whereas the Daubechie wavelets are the most commonly used [23]. Functions of the Daubechie, Symlet or Coiflet family allow a selection of different orders [25]. For example, "db4" is a function of the Daubechie family of order 4 as shown in Figure 2.2 created with the PyWavelets package for Python [26]. Basically, the wavelet transform of a signal is based on the correlation between the signal and

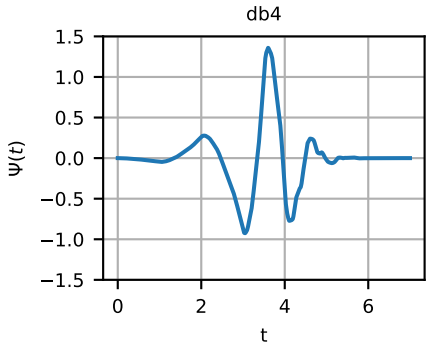


Figure 2.2: Mother wavelet function db4

the wavelet function $\Psi_{a,b}(t)$ [22]. Similar to the Fourier transform, one needs to distinguish between the Continuous Wavelet Transform (CWT) and the Discrete Wavelet Transform

(DWT), whereas the former is analyzed at continuous time and frequency values and the latter at discrete sampling points [23]. Accordingly, the CWT leads to a representation of the signal as weighted integral of continuous wavelet functions. The DWT, however, expresses the signal as weighted sum of shifted and scaled wavelet functions [15, 23, 27].

Since the wavelet functions can be scaled with the parameter a, their time-width can be adjusted to their frequency, allowing a maximum resolution in both the time and frequency domain [9, 14]. Increasing the scaling parameter a compresses the wavelet function in the time domain and thus increases the time resolution in order to correspond to high frequencies [9, 16, 21]. On the contrary, lower values of a stretch the wavelet function in the time domain, increasing the frequency resolution for low frequencies [9, 16, 21]. Compared to the STFT, this allows a change of the window-size over translation in time and thus to achieve a complete time-frequency representation of the signal which is the main reason for the success of wavelet transforms in signal processing and time-frequency analysis [9, 16, 17].

The differences in time and frequency resolution between Fourier transform, STFT and wavelet transform are illustrated in Figure 2.3. It is visible, that Fourier transform loses the time information, STFT has a fixed resolution for both time and frequency and wavelet transform has variable resolutions. Shannon shows in this plot the sampling frequency that limits the resolution.

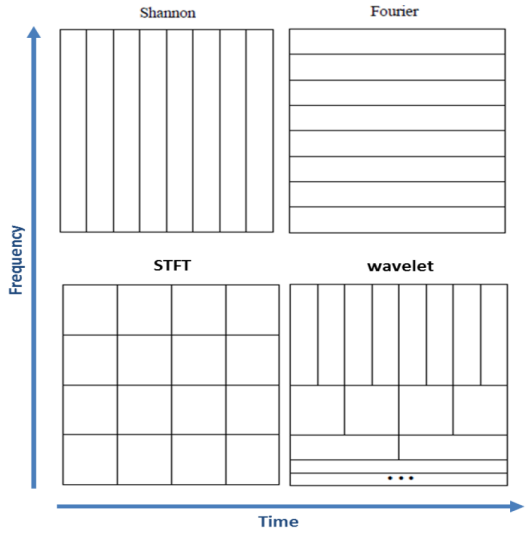


Figure 2.3: Comparison of time and frequency resolutions for time-frequency transforms, taken from [28]

Generally, wavelet transforms can be used to decompose non-stationary, fast-changing signals into their frequency components, often using asymmetric and irregular wavelet functions, which allows a complete representation of these signals [15, 23, 24, 25].

The Mathematical definition of the CWT of a function $f(t)$ is according to [9] as follows:

$$\mathscr{W}_\Psi[f](a,b) = \langle f, \Psi_{a,b} \rangle = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} f(t) \overline{\Psi\left(\frac{t-b}{a}\right)} dt \tag{2.9}$$

Here, $\overline{\Psi}(t)$ is the complex conjugate of the mother wavelet scaled and translated by $a, b \in \mathbb{R}, a \neq 0$.

Subsequently, [24] defines the DWT as stated in Equation 2.10 where cA_i are approximation coefficients, $\varphi_i(t)$ a scaling function, cD_{jk} detail coefficients and $\Psi_{jk}(t)$ a scaled and

translated wavelet function. An efficient implementation of the DWT is the Multi-Resolution Analysis Wavelet Transform (MRA-WT) which decomposes a signal into m decomposition levels by applying the wavelet transform in a hierarchical algorithm with approximation coefficients cA_i and detail coefficients cD_i as output where $i = 1, .., m$ [13, 18, 21]. The approximation coefficients are then used as input to the subsequent decomposition level and are split at each level into cA_i and cD_i consecutively [18]. In special use cases, this decomposition can be considered as a series of lowpass and highpass filters with approximation coefficients as the output of the lowpass filter and the detail coefficients as the ouput of the highpass filter respectively [13, 18, 21]. At each stage of decomposition, the frequency resolution is doubled and the time resolution is halved due to downsampling by the factor of 2 in order to account for scaling [16, 20, 21]. This allows the extraction of increasingly finer details [16]. Reconstruction of the signal from its decomposed representation as approximation coefficients cA_i and detail coefficients cD_i is possible at every level of decomposition [15].

$$x(t) = \sum_{i=-\infty}^{\infty} cA_i \varphi_i(t) + \sum_{j=0}^{\infty} \sum_{k=-\infty}^{\infty} cD_{jk} \Psi_{jk}(t) \tag{2.10}$$

2.1.2 Parameter Selection for Wavelet Transforms

In many use cases, it is necessary to remove noise from signals to extract only the part of the signal that contains information or to separate the noise from the signal to gain insights from analyzing both parts of the signal separately. In the field of signal processing, noise is referred to as unwanted disturbances within a system that could obscure the main information of a signal [1, 5]. Wavelet decomposition is a common method to separate noise from signals by modifying the output coefficients [25, 27]. In the case of the hierarchical MRA-WT, the first level of decomposition can be considered as "noise reduction level" according to [13, p. 610] that separates signal components of high frequencies. Denoising of signals with wavelet transforms is often applied to brainwave signals (Electroencephalography (EEG)) that are used to monitor and identify changes in brain signals [25]. In addition, wavelet transform is commonly applied to extract signal characteristics for the mointoring of the health of a system, see, e.g., Boukabache et al. (2013) [29].

Obtaining a so-called *denoised* version of the original signal, i.e. the remaining signal after noise extraction, can be done by decomposing the signal using the wavelet transform and modifying the approximation and detail coefficients [25]. One approach to modify the wavelet coefficients in order to separate noise from a signal is called *thresholding*. There are several thresholding methods available, e.g. the hard-threshold estimator [30] which is the simplest one or the soft thresholding method *SURE* that finds threshold limits for each level [25]. However, these methods require knowledge of the properties of both signal and noise (e.g. frequency bands) to be able to set correct thresholds. Since this is not the case for the signals used in this thesis, the thresholding methods will not be explained in more detail.

Furthermore, when applying the wavelet transform to decompose signals, one needs to carefully select the mother wavelet function $\Psi(t)$ as well as the level of decomposition to obtain a proper representation of the signal [25]. Currently, there are no clear rules in the literature for selecting the most appropriate mother wavelets for specific applications [25]. If the user has profound knowledge on the properties of the signal, the mother wavelet can be selected according to the waveform based on previous experiences as done in [18] and [15].

Another approach to select the optimal mother wavelet proposed by [24, 25, 31] is based on the correlation coefficient calculated for the original signal and its denoised version. More precisely, the wavelet transform is used to denoise the original signal using a selection of

mother wavelets that are of interest. The correlation coefficient is then calculated for the original signal X and the denoised signal Y using Equation 2.11 according to [24].

$$Corr(X,Y) = \frac{Cov(X,Y)}{\sqrt{Cov(X,X) \cdot Cov(Y,Y)}} \tag{2.11}$$

Correlation is often used as a statistical measure to describe the linear dependence between two variables [31]. Values of the correlation coefficient range from -1 to +1 where -1 and +1 mean a strong correlation, 0 means no correlation and the negative or positive sign refers to the direction of correlation [31]. The mother wavelet achieving the highest positive correlation coefficient is then considered as best suitable for this use case [25].

Generally, mother wavelet functions can be selected based on their compatibility with the characteristics of the signal to be analyzed [25]. For the selection of the most appropriate level of decomposition, Al-Qazzaz et al. (2015) [25] suggest to set this level according to the dominant frequencies present in the signal and the usefulness of features that are generated from the results of the signal decomposition.

2.2 Statistical Features for Signal Classification

The following section introduces statistical measures that can be used in algorithms to classify samples of discrete signals. Signal classification is used for various types of signals and classification tasks, for example in the field of EEG signals to detect physical problems, as shown in [24, 32]. Moreover, statistical features, such as those presented in this section, are often used in data-driven PHM approaches to detect failure precursors [33].

One objective of signal processing is to generate a representation of a signal that allows the calculation of a set of highly discriminative parameters that enable signal classification algorithms to distinguish between various classes of signals [34]. These parameters, called *features*, can be divided into time domain features and frequency domain features [24, 32, 35]. Time domain features are for example mean, median or standard deviation whereas examples for frequency domain features are bandwidth, peak frequency or peak power [24, 32, 35]. In the following, a selection of time domain features will be explained in more detail.
Mean, median, variance and standard deviation are basic statistical measures describing statistical distributions that are often used in signal classification tasks, see e.g. [21, 24, 32, 34]. The calculation of these basic measures is shown in Appendix A. The Root Mean Square (RMS) measures the effective value of a signal and is often used to evaluate the signal's power [5]. It is calculated as the square root of the average of the squared values of a signal, leading to Equation 2.12 for a signal with discrete values X_i and $i = 1, .., N$, according to [35].

$$rms(X) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} X_i^2} \tag{2.12}$$

Moreover, several research papers suggest the usage of the number of zero crossings [24, 32, 35] and the Shannon entropy [24, 35] in signal classification tasks. The Python package *PyAstronomy* contains a function called *zerocross1d* to count the zero crossing events in a one-dimensional dataset [36]. Signal entropy is a measure of complexity of a signal and is based on the probabilities of unique values. Accordingly, Shannon entropy is calculated as shown in Equation 2.13 based on [35, 37, 38] where p_i is the probability of occurrence for each unique value i in X.

$$H(X) = -\sum_{i} p_i \log p_i \tag{2.13}$$

Additional relevant features to evaluate the signal and the distribution of its values are the line length L, the skewness S and the kurtosis K, that are commonly used for signal classification in several research papers, see, e.g., [21, 24, 32, 34]. Line length refers to the length of the curve of a discrete signal X and is calculated as the sum of the absolute distances between two consecutive values (Equation 2.14) [21, 35].

$$L(X) = \sum_{i=2}^{N} |X_i - X_{i-1}| \tag{2.14}$$

Skewness and kurtosis are statistical measures referring to the shape of the data distribution where skewness measures the asymmetry [5, 21, 39] and kurtosis the peakedness of the distribution [5, 39]. Equation 2.15 shows the calculation of the skewness and Equation 2.16 the calculation of the kurtosis, both based on the central moments of the data distribution [39, 40, 41].

$$S(X) = \frac{\sqrt{N(N-1)}}{N-2} g_1 \tag{2.15}$$

$$K(X) = \frac{N-1}{(N-2)(N-3)}((N+1)g_2 + 6) \tag{2.16}$$

Where g_1 and g_2 are defined as follows:

$$g_1 = \frac{m_3}{m_2^{\frac{3}{2}}} \tag{2.17}$$

$$g_2 = \frac{m_4}{m_2^2} - 3 \tag{2.18}$$

Here, m_r is the central moment of order r of the data distribution defined by Equation 2.19 for a signal X with mean value $\mu(X)$ [41].

$$m_r = \frac{1}{N} \sum_{i=1}^{N} (X_i - \mu(X))^r \tag{2.19}$$

When working with noise extraction, it can be useful to compare the signal to the extracted noise signal. One measure to do so is the Signal-to-Noise Ratio (SNR) that is defined as the the power of the signal S divided by the power of the noise signal N [5, 42, 43]. Consequently, the SNR can be calculated as shown in Equation 2.20 based on [42] using RMS which was introduced earlier and represents the power of a signal.

$$SNR = \frac{rms(S)}{rms(N)} \tag{2.20}$$

2.3 Prognostics and Health Management

In the following, basic terms of Prognostics and Health Management (PHM) are introduced and common strategies and methods in this field are explained.

Almost every system is suspect to degradation over time due to stress or load and thus both the system's health and performance can decrease [44]. Companies and manufacturers developed various maintenance strategies in order to keep a system in its original performance

condition and to assure its capability to perform the originally intended function [44, 45]. Here, maintenance means the consolidation of a variety of planned or unplanned activities during a system's life cycle to preserve or to restore the original performance, and to maintain the reliability level [45, 46]. Such activities could be inspection, monitoring, repair, diagnosis and others [45, 46].
The increasing number of adaptive and connected systems allows the calculation, monitoring and optimization of the system's reliability during operation in order to predict and prevent failures based on data containing information about the extent of degradation and the health state of a system [44, 45, 47, 48]. Approaches that aim at optimizing the reliability and availability, reducing both the maintenance and life-cycle costs and the downtime of a system, can be summarized under the term Prognostics and Health Management (PHM) [45, 47, 48]. One main task of PHM is to calculate, evaluate and monitor the Remaining Useful Life (RUL) of a system. RUL defines the remaining lifetime of a system under the assumption that a system performs its intended functionality within appropriate confidence intervals [44, 45, 47]. PHM and thus the assessment of the RUL requires activities in the fields of sensing, diagnostics, prognostics, data acquisition, health management and anomaly detection [44, 47, 48].
Prognostics as the first part of the term PHM covers the process of estimating the system's RUL based on the extent of degradation and the progression of faults [44, 45, 48]. Health Management as the second part of the term PHM is the process of decision making for maintenance actions based on estimates of the degradation and the health state of a system and the severity of possible faults [44, 45, 48]. It also predicts the future use and the associated loads as well as their effect on the system's degradation [48].

PHM strategies can be distinguished according to their temporal occurrence in failure treatment [44, 45]. Corrective maintenance takes place after the failure occurs [44, 45, 46]. Hence, it is reactive, unplanned and passive-only, and does not require monitoring of a system's condition [44, 45]. As a result, failures are hard to predict which leads to long downtimes and high maintenance costs since replacement parts must be available immediately [44].
Another approach is the preventive maintenance which follows an active, planned and periodic maintenance plan in order to prevent failures [44, 45]. Maintenance actions are taken independently from a system's condition, thus healthy parts may be replaced, causing high maintenance costs, but reducing unexpected machine failure [44, 45].
Predictive maintenance, however, also known as condition-based maintenance, takes the degradation of a system as well as its health status into consideration and adapts maintenance actions accordingly [44, 45, 46]. It is proactive and capable to avoid unnecessary replacements in order to decrease maintenance costs and to increase availability and reliability by extending the RUL [44, 45, 47, 49].
An efficient PHM strategy should be able to detect failure precursors early, predict the progression of faults and help to make decisions for maintenance actions in order to decrease costs and increase availability [44, 45].

Generally, there are two main approaches to obtain a representation of a system that allows the calculation of the RUL and the assessment of the health state in the prognostics step [44, 47, 48]. The first approach is to rely on a physical model of a system which requires a full understanding of the system and its failure modes [44, 48]. Measurement data is combined with the physical model to assess the health status and the damage of the system based on experienced loads and stresses in the past [44, 48]. Additionally, the physical model can be used to predict the system's future behavior based on expected usage conditions [44].
If neither a physical model nor knowledge about failure modes and root causes is available, data-driven methods can be used [44, 48]. These methods apply data analytics and machine

learning methods to collected measurement data to identify characteristics of the degradation process, to detect anomalies and to make predictions about the future behavior [44, 48]. Artificial Neural Networks (ANNs) are the most common method for data-driven maintenance and are trained to reproduce outputs for unseen inputs [44]. Since data-driven methods rely on measurements and datasets, their accuracy and performance strongly depend on the available amount of data and the selected features [20, 44, 48].
One main task of data-driven prognostics is the monitoring and detection of failure precursors [48]. Such failure precursors are signs related to the beginning of a failure which could be detected as an anomalous data event, e.g. as change in a measurement variable, and used to predict failures [48]. Early failure prediction can be achieved by model-based characterizations of the healthy system based on variables like current, voltage or temperatures [48]. Pecht and Kang (2018) [48] provide an overview over possible failure precursors for electronic components, e.g., noise for electrolytic capacitors or power dissipation for general-purpose diodes.

2.4 Artificial Intelligence

In order to apply classification algorithms correctly and most efficiently to given datasets, it is crucial to understand the basic idea of Artificial Intelligence (AI) and Machine Learning (ML) algorithms. Thus, important definitions and approaches as well as a selection of algorithms will be covered in this section.

Artificial Intelligence (AI) belongs to the field of computer science and is concerned with the development of techniques to construct algorithms and systems with intelligent behavior using hardware and software [50, 51, 52, 53]. This so-called intelligence should enable the computer system to build knowledge through learning in order to make decisions and solve problems [51]. Since this could be applied to a broad range of fields and problems, it remains unclear what can and what cannot be done by AI algorithms [51]. Exemplary fields where AI is applied are speech processing, object recognition, image understanding and decision tree learning [51].

ML is a sub-field of AI that deals with techniques that allow computers to learn knowledge or skills without being explicitly programmed [48, 51, 52]. This enables algorithms to improve from experience that is in most cases contained in datasets [54, 55]. This means in mathematical terms that a system learns "from experience, E, with respect to a task, T, and a performance measure, P, if its performance on T, as measured by P, improves with experience E" [52, p .2]. ML algorithms thus use information contained in a dataset to create generalized knowledge for performing tasks [56].
Usually, ML algorithms are categorized according to the extent of supervision required during the training phase [48, 51, 56]. Pecht and Kang (2018) [48] distinguish between supervised, unsupervised, semi-supervised and reinforced learning. In the following, supervised and unsupervised learning will be discussed in more detail, as they are used in this thesis. For the remainder please refer to the literature, e.g., [48, 54, 56].

For supervised learning algorithms, it is necessary to provide the results, i.e. labels, for all observations in a dataset [48, 56]. In this case, the model learns the relationship between the input and the output variables, where the input is the observations (e.g. customer data) and the output is the labels (e.g. fraud/non-fraud) to predict the labels for unseen observations [48, 56]. The labels can be either a categorical variable or a numerical variable. In most cases, algorithms used for tasks with categorical labels (e.g. fraud/non-fraud) are called classification algorithms, whereas algorithms used for numerical labels are called regression algorithms (e.g predict the value of a house) [48, 54].

It takes three main stages to construct a model using supervised learning methods: training, testing and prediction [54]. In the training phase, labeled input data is provided to the algorithm to learn patterns and relationships within the input data with respect to the output labels [54]. In some cases, e.g. when training neural networks, the training data is split into training and validation set [54, 56]. The validation set is used to monitor the training progress [56]. Subsequently, the model performance is evaluated in the testing phase by applying the model to an unseen dataset and comparing the predictions to the real outputs [54]. Finally, in the prediction phase, the model is applied to production data and is used for predictions in operational mode [54]. In order to perform both the training and testing phase efficiently and reliably, it is necessary to split the input dataset in a training and a testing set (*train-test-split*) [54, 55]. This ensures that both training and testing dataset have equivalent data quality and that the model has not seen the test data prior to the testing phase, allowing for an unbiased evaluation of the model performance. Figure 2.4 summarizes the proposed process for supervised ML algorithms.

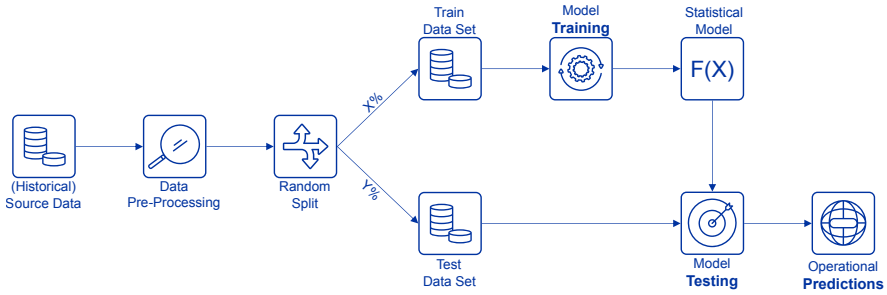


Figure 2.4: Exemplary ML process, including training, testing and prediction phase, based on [54]

Unsupervised algorithms, however, are applied to unlabeled datasets where the solution is unknown [48, 52] and are often employed to learn properties of the underlying system or mechanism that generates the data [54, 56]. Thus, these algorithms are most often used for grouping of similar observations, which is called *clustering*, or to represent the observations in a smaller subspace of the given input variables or features, called *dimensionality reduction* [48, 54, 56]. Another common use case for unsupervised learning is anomaly detection with the objective of identifying observations that do not match the expected normal behavior or pattern [48, 54].

2.4.1 Data Pre-Processing

Real-world datasets are often affected by missing values, noise or inconsistencies [48, 54]. Therefore, data pre-processing is required before applying ML algorithms to improve the performance and prevent false conclusions from the results [48, 54]. The following subchapter will cover the pre-processing steps of data cleaning, scaling, feature engineering and handling of imbalanced data as proposed by [48].

Data Cleaning

The process of data cleaning is mainly concerned with detecting and correcting inaccurate data such as missing values or outliers [48]. ML methods are applied to input datasets in order to extract as much information as possible with respect to the desired target. Hence, the performance and the reliability of these models strongly depend on the quality and completeness of the data that is used. This results in the need to handle missing values in datasets by

either deleting incomplete observations or replacing missing values with appropriate methods [48, 54]. Common methods for replacing missing values include interpolation techniques or predictive models that rely on regression techniques.

Scaling

Many applications of ML algorithms require high-dimensional datasets (i.e. a high number of features) what is also true for PHM of electronics according to [48]. A high number of features often implies a high variety of scales what can cause ML models to be biased towards features that consist of large-scale data [48, 54]. One reason for this is that most classifiers use the Euclidean distance to calculate distances between observations, which leads to large-scale features dominating the distance measure [48]. There exist two main methods to solve this issue by scaling the features, namely min-max normalization and standardization (or z-normalization) [48, 54].
The min-max normalization scales the features according to their minimum and maximum values as shown in Equation 2.21 for feature X [48, 54].

$$X_{i,normalized} = \frac{X_i - X_{min}}{X_{max} - X_{min}} \tag{2.21}$$

The standardization or z-normalization of a feature X can be done using Equation 2.22 where μ is the mean and σ the standard deviation of X [48].

$$X_{i,standardized} = \frac{X_i - \mu(X)}{\sigma(X)} \tag{2.22}$$

Feature Engineering

In general, feature engineering is related to feature construction and feature selection [48]. ML algorithms require features that retain information about the problem being analyzed [48, 54]. Feature engineering is a crucial step in the development of highly reliable and accurate algorithms and requires both time effort and domain knowledge [48, 54]. Most often, features are constructed manually based on expert knowledge and/or the results of an Exploratory Data Analysis (EDA) by processing the input variables (e.g., combining two variables) [48]. Since most algorithms work best with numerical variables, categorical variables need to be converted to numerical variables either by creating Boolean dummy variables or by simply converting levels of categorical variables to a number [54].
Subsequently, a subset of relevant features for the model development needs to be selected with the purpose of improving the model performance and the interpretability by removing irrelevant features causing overfitting or bias, and simplifying the model to better understand the contribution of features to the model predictions [48]. The increasing difficulty of tasks with a high number of variables, or other problems arising from high-dimensional datasets are often associated with the term *curse of dimensionality* [57, 58].

Handling of Imbalanced Data

In the case of classification algorithms, datasets are called imbalanced if the observations of the target classes are not equally distributed among the classes [48, 54, 59]. Both supervised and unsupervised algorithms can learn from imbalanced datasets resulting in model performances similar to those obtained when learning from the same datasets after being balanced by suitable methods [48]. Nevertheless, balancing the datasets can help to improve the model performance [48]. Since the anomaly detection process performed in this thesis is unsupervised and therefore the distribution among the classes not known, reference is made to the literature for information on available methods to balance existing datasets, see e.g. [48, 54].

2.4.2 Model Evaluation

Supervised ML algorithms can be evaluated by comparing the prediction to the actual targets. The performance metrics used to evaluate the model performance differ between classification and regression algorithms due to the different variable types of the labels.
Results of classification algorithms are most often described using a so-called *confusion matrix* where the predictions of the model are classified into true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) [48, 54, 59]. Table 2.1 shows the confusion matrix for a binary classifier according to [48, 54].

		Predicted	
		TRUE	FALSE
Actual	TRUE	True positive (TP)	False negative (FN)
	FALSE	False positive (FP)	True negative (TN)

Table 2.1: Confusion matrix for binary classifier

Based on the confusion matrix, various performance metrics can be calculated to evaluate the predictions of the model. The most common metrics, namely accuracy and precision, will be defined in the following.
Accuracy is related to the overall performance of the classifier and represents the proportion of correct predictions related to all predictions that were made leading to Equation 2.23 according to [48]. Precision measures the proportion of true predictions that where correct (Equation 2.24) [54].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.23}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.24}$$

To evaluate regression models with numeric labels, metrics based on the distance between prediction and actual outcome, such as the Mean Absolute Error (MAE), the Mean Squared Error (MSE) and the Root Mean Square Error (RMSE), are commonly used [48, 54]. The MAE is the mean of the absolute values of the differences between prediction and actual result as shown in Equation 2.25 where $\hat{y}$ is the prediction and y the actual outcome [48, 54]. The MSE is the mean of the squares of the error values (Equation 2.26) [48]. However, the RMSE is the most popular metric to evaluate the accuracy of regression models which is calculated as the square root of the MSE (Equation 2.27) [48].

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i| \tag{2.25}$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{2.26}$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2} \tag{2.27}$$

2.4.3 Classification Algorithms

This section will briefly cover the mechanism of two supervised classification algorithms that will be used in this paper: k-Nearest Neighbors (k-NN) and Gradient Boosted Decision Trees (GBDT). Anomaly detection algorithms are discussed in the next section (section 2.5).

K-Nearest Neighbors (k-NN)

K-Nearest Neighbors (k-NN) is a supervised statistical learning method that is often used to cluster objects according to their similarity with respect to their attributes [51]. It can be used for both classification and regression problems and predicts the result of a new observation based on the k nearest neighbors [51, 54]. If used for classification, this means that the class of a new observation is determined based on the majority class of the k nearest neighbors [51, 54]. Figure 2.5 shows this selection algorithm for $k = 7$ nearest neighbors in a two-dimensional dataset which was created using the Python package *scikit-learn*. Accordingly, the new observation is assigned to class 1 since the majority of the nearest neighbors (4 out of 7) belong to this class. Since the k value can influence the algorithm performance, it needs to be selected carefully. Large k values decrease the impact of noisy patterns but also reduce the sharpness of the model [51, 60]. However, according to a rule of thumb often used in the literature, k can be calculated as $k = \sqrt{N}$ where N is the number of samples [61, 62, 63]. Moreover, it is important to select an odd value for k for binary classification tasks to avoid confusion between classes [54, 61]. The k-NN algorithm *KNeighborsClassifier* used in this thesis is included in the Python package *scikit-learn*.

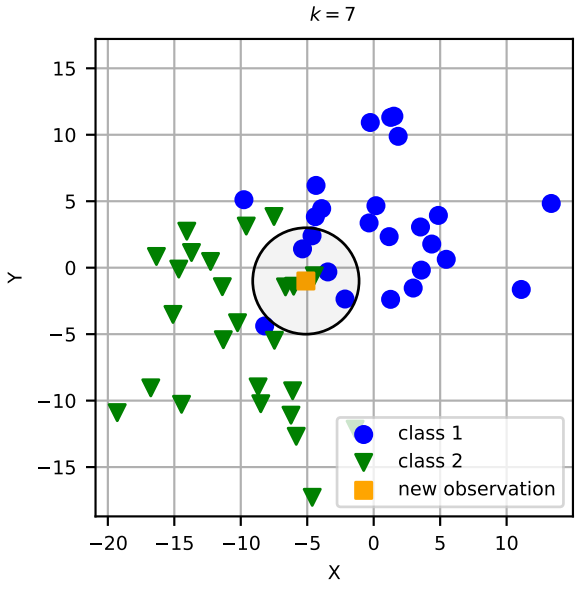


Figure 2.5: k-NN algorithm for $k = 7$ nearest neighbors

Gradient Boosted Decision Trees (GBDT)

Gradient Boosted Decision Trees (GBDT) rely on both decision tree and gradient boosting methods. Decision trees are used in supervised learning algorithms for both classification and regression problems [48]. For classification problems, tree-like structures based on nodes, branches and leafs partition the training dataset recursively into subsets [48, 51, 54]. This requires to determine the variable and the corresponding value that is used to split the dataset at each node [54]. Unseen observations are then classified by starting from the root, testing the variables at each node and repeating the process until the leaf node is reached [51]. The

structure of the decision tree is related to the capability of the features to solve the problem at hand which allows the analysis of the so-called *feature importance* [51, 54, 64]. Thereby, features can be ranked by their impact on the model decision [64].

The term *ensemble learning* refers to methods in which multiple ML algorithms are combined into a single model to overcome the drawbacks of the individual algorithms and to reduce both the variance and bias of the predictions [48]. Boosting algorithms extend this idea and convert multiple weak classifiers into a strong classifier based on a loss function [48, 54]. This loss function is represented in a form appropriate for optimization and is then used by Gradient Boosting algorithms to identify the shortcomings of weak learners in order to optimize the loss function based on the gradient in a sequential manner [54, 65]. More precisely, the model performance improves at each iteration, called *boosting stage*, by focusing on observations that were misclassified in previous iterations [65].
Advantages of GBDT are robustness to overfitting and generalization [66]. Additionally, decision trees are generally able to handle missing values and are not sensitive to outliers in the input dataset [67]. The Python package *scikit-learn* comes with a GBDT classifier called *GradientBoostingClassifier* that is an effective and accurate classification algorithm which can be used in a variety of applications for binary or multi-class classification problems.

All ML models are configured by a set of parameters, called *hyperparameters*, that specify the architecture of the model [68, 69]. Setting these hyperparameters appropriately can increase the efficiency and performance of a ML algorithm [70]. Examples for hyperparameters are the number of nearest neighbors k of the k-NN algorithm or the number of boosting stages for GBDT.

2.5 Anomaly Detection

The concept of anomaly detection as well as the two anomaly detection algorithms used in this thesis, namely Isolation Forest (iForest) and Long Short-Term Memory Autoencoder (LSTM-AE) are presented in this section.

Anomaly detection, also known as outlier detection, is related to the problem of finding patterns or instances in a dataset that are dissimilar to the rest of the dataset and do not conform to the expected normal behavior [48, 71, 72, 73]. These patterns or instances are called anomalies or outliers [71, 72]. Anomaly detection is used in a variety of fields to detect as many outliers as possible based on data-driven methods [71, 73]. It is of great importance in many applications, as significant and critical information about the underlying system can be obtained from detected anomalies [48, 71].
Anomalies are data points which are sufficiently far away from normal patterns or regions and can be caused by errors in the data acquisition or new, previously unknown, processes [71, 72]. Figure 2.6 shows an example of anomalous observations that are distant to the three normal patterns created with the Python package *scikit-learn*. Hawkins (1980) [74] defines an outlier as "an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism" [74, p .1] what leads to the assumption that outliers can indicate changes in the mechanism or malfunctions.

As mentioned in section 2.4, ML algorithms can be categorized into supervised and unsupervised algorithms. Likewise, anomaly detection algorithms are categorized according to the availability of labels [71]. Obtaining an accurate and representative labeled dataset suitable for anomaly detection can be very difficult as the labels need to cover all possible types of behaviors for both normal and anomalous observations [71]. However, even if labeled datasets are available, supervised algorithms do not necessarily lead to better results than unsuper-

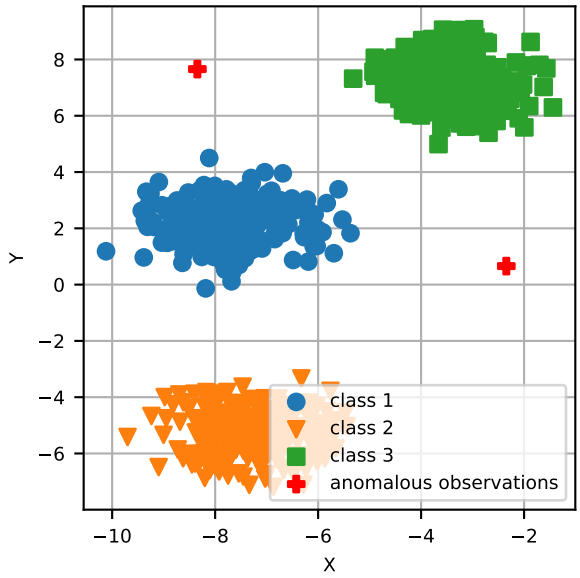


Figure 2.6: Example of anomalies in a two-dimensional dataset

vised algorithms [75]. In addition, mechanisms and behaviors often change dynamically, so new types of observations may arise which are not captured by the labeled data set [71]. Moreover, anomalies are rare in most use cases making the dataset highly imbalanced what affects classification-based anomaly detection approaches [71].

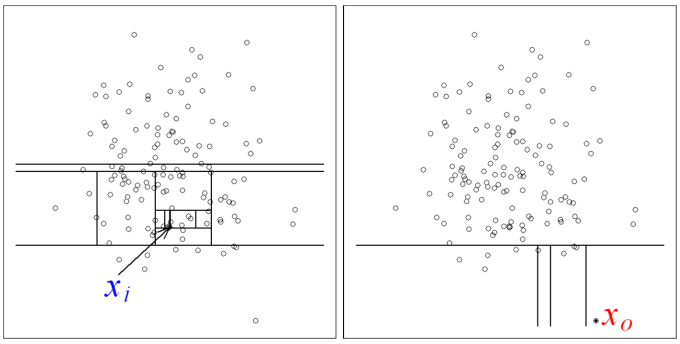
Further, anomaly detection techniques can be categorized into distance-based, clustering-based, classification-based, statistical and spectral techniques [48, 71]. Distance-based techniques calculate the distance between data points using appropriate metrics and assume that anomalies occur far from normal observations [48]. Clustering-based techniques identify anomalies based on their distance to cluster centroids assuming that normal observations belong to the same dense clusters [48, 71]. Classification-based techniques learn from a labeled dataset containing both anomalous and normal observations and classify unseen observations according to their attributes [48, 71]. Moreover, statistical techniques rely on stochastic models that generate the observations assuming that anomalies occur in regions with lower probabilities than normal observations [48, 71]. Spectral techniques, however, assume that the data can be represented in a lower dimensional subspace which allows the distinction between normal and anomalous observations [71].

Since the capability of distance measures to distinguish between anomalous and normal observations decreases drastically with high numbers of dimensions (i.e. number of features), both distance-based and clustering-based techniques are not suitable for anomaly detection in high dimensional datasets [48, 71]. This phenomena is related to the aforementioned curse of dimensionality. In addition, outliers are not necessarily related to an extreme value what could also affect the performance of techniques relying on distance measures [73].

In most cases, anomaly detection algorithms output an anomaly score based on the extent to which each observation is considered anomalous [71]. Hence, observations can be ranked by anomaly score and anomalies can be either identified as top proportion of this ranking or based on an appropriate cutoff threshold [71].

2.5.1 Isolation Forest

Most of the common anomaly detection algorithms rely on constructing a profile of healthy observations and then identify anomalies as observations that are different from this normal profile [48, 76, 77]. Consequently, these algorithms are optimized to learn the normal behavior and not to detect anomalies as such [76]. Moreover, employing distance-based measures to calculate the anomaly score when comparing unseen observations to the healthy profile often leads to high computational complexity which limits the size of the input dataset [48, 76].
The anomaly detection algorithm Isolation Forest (iForest), proposed by Liu et al. (2008) [76], does not compare unseen observations to the normal behavior. Therefore, it can not be clearly categorized in one of the aforementioned classes of anomaly detection algorithms. The iForest algorithm is rather a model-based method with the purpose of isolating anomalies from normal observations assuming that anomalies are rare and distinctly different from normal observations [48, 76]. Thus, anomalies are more likely to be isolated than normal instances [75, 76]. iForest algorithms consist of an ensemble of binary Isolation Trees (iTrees) meaning that each node has exactly two or zero daughter nodes [76, 78]. To train an iTree the input dataset is recursively partitioned until each observation is isolated or a pre-defined maximum tree height is reached [75, 76, 77]. Due to the higher susceptibility to isolation, anomalies are sorted out earlier and thus closer to the root of the iTree, resulting in shorter path lengths compared to normal instances [48, 76]. This behavior is illustrated in Figure 2.7, where the normal observation x_i requires more random partitions of the dataset to be isolated than the outlying observation x_o.
A high number of normal observations arising from large training samples can affect the isolation process, which could reduce the ability to isolate true anomalies [76]. Consequently, iTrees are trained with a subset of the input data selected by random sub-sampling [75, 76, 78, 79]. This also helps to reduce effects of issues like swamping and masking [76, 79]. Swamping refers to false positives, i.e. normal instances identified as anomalies, whereas masking occurs if the number of anomalies in a dataset is too high and thus anomalies are able to mask their presence as normal [76]. Using randomly selected subsamples for the training of the iTrees also allows to detect local and global outliers [75].



(a) Isolating x_i (b) Isolating x_o

Figure 2.7: Isolation of a normal observation x_i and an outlying observation x_o, taken from [76]

The anomaly score of the iForest algorithm is calculated based on the length of the path of one observation from the root node until the final external node after separation [76]. This length is obtained from the number of edges an observation traverses in the tree structure [76]. Since iTrees are trained with random sub-samples, each tree produces a different path length [76]. However, Liu et al. (2008) [76] show that the average path length converges with an increasing number of trees. Using this average path length, the anomaly score $s(x, n)$ is

then calculated as shown in Equation 2.28 according to [76], where $E(h(x))$ is the average of the path length $h(x)$ of an observation x. Furthermore, n is the sub-sampling size and $c(n)$ is the average path length of the sub-sample that is used to normalize $h(x)$ [48, 76, 77]. The resulting anomaly scores range between 0 and 1 and can be interpreted as follows, according to [76]: Anomaly scores very close to 1 imply that an observation is highly likely to be anomalous. Observations with $s \ll 0.5$ are usually considered as normal. If all resulting values are close to 0.5, it is very likely that the entire dataset does not contain any anomalies that can be clearly isolated.

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{2.28}$$

One main advantage of the iForest algorithm is its linear time complexity and low memory requirement [75, 76, 77, 78]. This reduces the computational costs significantly compared to distance-based methods and allows the application of the algorithm to extremely large datasets [75, 76, 78]. Since iForest does not require a definition of a healthy state it is fully unsupervised and also applicable if this normal behavior is unknown [75]. Moreover, iForest is able to handle high-dimensional datasets containing many features with low relevance for anomaly detection [76]. Nevertheless, iForest is also affected by the curse of dimensionality (see subsection 2.4.1) which makes the isolation more difficult [76]. Liu et al. (2008) [76] show that the performance of the iForest algorithm can be improved by reducing the dimensions of the input data based on the kurtosis measure. More precisely, relevant features are selected based on the peakedness of the distribution of the feature values (i.e. the kurtosis).
In addition, iForest is robust to the choice of hyperparameters and requires only the selection of the sub-sampling size n and the number of trees t [75, 76].

For most applications, the number of trees can be set to $t = 100$ since the path lengths usually converge before this number according to [76, 77, 78]. Furthermore, the sub-sampling size is usually set to $n = 256$ (see, e.g., [77, 78, 79]). Ding and Fei (2013) [78] state that this sub-sampling size reduces the effects of swamping and masking whereas Liu et al. (2008) [76] find that $n = 256$ is sufficient for a wide range of anomaly detection applications.

2.5.2 Autoencoder

Insufficient detection rates of common anomaly detection algorithms increased the usage of deep learning algorithms [80]. Deep learning is a sub-field of ML that is used to learn abstract representations of datasets by hierarchically transforming the data to a more abstract level [80, 81]. Deep learning algorithms make use of neural networks and can be trained either supervised or unsupervised whereas the latter is suitable for anomaly detection if no labelled data is available [80, 81].
Autoencoders are unsupervised trained neural networks with a bottleneck layer which learn a low-dimensional data representation and consist in the basic form of an encoder and a decoder part [80, 82]. These networks are often used for dimensionality reduction or anomaly detection and are able to learn non-linear correlations in the input data [82, 83]. Anomaly detection with autoencoders belongs to the classification-based methods. It is based on the assumption that normal instances are correlated and the model tries to find a subspace where normal and anomalous observations are distinguishable [83]. The general principle of autoencoders is to compress the input data to a lower dimensional subspace and to reconstruct the input from this compressed representation [80, 82]. This allows the optimization of autoencoders during training by minimizing the reconstruction error [80, 82, 83, 84].
Encoder and decoder are both parametric functions which are represented as neural networks [82]. The encoder is a function that transforms the input x to a hidden, low-dimensional

representation $h(x)$ and can be described by Equation 2.29 [80]. Here, W_h is the weight matrix of the encoder network, b_h the bias vector and s_f the activation function.

$$h(x) = s_f(W_h x + b_h) \tag{2.29}$$

The decoder, however, generates the reconstruction y from the hidden representation $h(x)$ (Equation 2.30) [80]. Again, W_y is the weight matrix of the decoder network, b_y the bias vector and s_g the activation function of the decoder.

$$y = s_g(W_y h(x) + b_y) \tag{2.30}$$

When describing the encoder as $E(X)$ and the decoder as $D(E(X))$, the autoencoder can be summarized as $Y = D(E(X))$ [84]. Training of autoencoders optimizes the parameters of encoder and decoder in order to minimize the reconstruction error between X and Y [84]. This error is calculated with a specific loss function that is often the MSE or an entropy based measure, e.g. cross-entropy or corr-entropy [82].
The activation function of both encoder and decoder produces an output for each node of the network by summing and weighting the input [82]. To enable the autoencoder to reveal non-linear correlations in the input data, it is necessary to select a non-linear activation function [83, 85]. In most applications of the autoencoder, the Sigmoid function is used as activation function [80, 82]. However, Pawar and Attar (2020) [82] suggest to prefer the hyperbolic tangent function.
To prevent overfitting of autoencoder algorithms, regularization is used [80, 82]. Overfitting would mean in this case that the reconstruction error is very low even if the underlying relations of the input data are not captured by the algorithm [80]. Regularization improves the generalization of the model by ignoring irrelevant components of the weight matrix [82]. It is added to the loss function to make the algorithm perform well on unseen data [80, 82]. For more details, reference is made to [80, 82].
Autoencoders can be categorized into shallow or deep networks [82]. Shallow networks have three layers (input layer, hidden layer, output layer) whereas deep networks have multiple hidden layers (see Figure 2.8) [82].

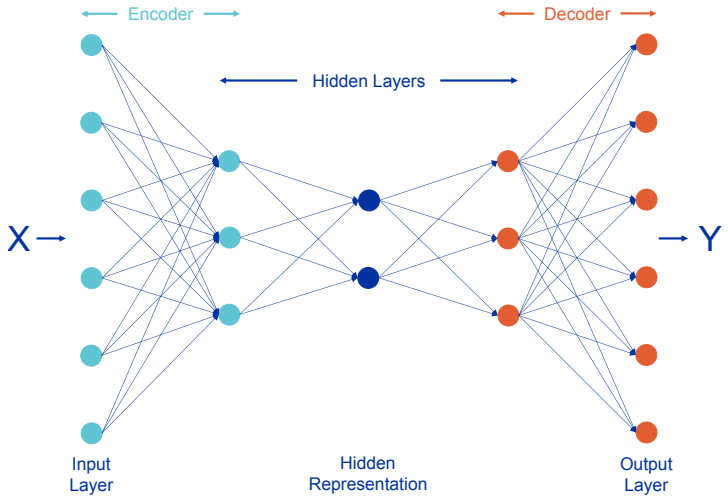


Figure 2.8: Schema of autoencoder with 3 hidden layers, based on [82]

To use autoencoders for anomaly detection, the network is trained on normal data describing the healthy state of a system [83, 86]. This training of an autoencoder network is done in a predefined number of iterations, called epochs. The reconstruction error is then

used as the anomaly score for unseen observations [82, 83]. Instances exceeding a certain threshold are considered as anomalies [82, 83]. This is based on the assumption that the reconstructed data follows the normal pattern of the training dataset and that the model is therefore not able to reconstruct anomalous observations deviating from this pattern [82]. However, it must be noted that the training of an autoencoder is specific for each input dataset and does not allow generalizations to other use cases [82].

Recurrent Neural Networks (RNNs) are a specific class of ANNs that rely on backward connections [85]. Common autoencoders are feed-forward neural networks where each layer uses the output of the previous layer as input [85]. RNNs, however, return the output of a network layer to the same layer, or to the previous layer, in order to address the shortcomings of classical neural networks and to improve accuracy [85]. As a result, RNNs are able to learn from a sequence of time-series data by considering the previous output and the current input at each node [85]. Let X be the input vector, W the weight matrices, b the bias, σ the activation function and z the hidden state. The hidden state z at time t is then calculated using Equation 2.31 where z_{t-1} is the hidden state at time $t-1$ according to [85].

$$z_t = \sigma(W_{xz}X_t + W_{zz}z_{t-1} + b_h) \tag{2.31}$$

Long Short-Term Memory (LSTM) networks are a type of RNNs widely used for temporal analyses to learn sequential dependencies [87]. In LSTM networks, each cell makes use of gates to control the transfer of information within the network [85, 87]. This is done with three types of gates: forget, output and input [85, 87]. The forget gate decides what part of the previous information is retained [85, 87]. The input gate is responsible for the selection of a proportion of the new input and to process this new information as well as the old information obtained from the forget gate [85, 87]. The output gate controls which part of the information received from the forget and the input gate is released as output [85, 87]. All three gates are configured by weights and bias matrices [87]. For more details, please refer to the literature, e.g. [87].

Long Short-Term Memory Autoencoders (LSTM-AEs) use a LSTM network for both encoder and decoder. According to [87], LSTM autoencoders are suitable for time series forecasting and anomaly detection since each network cell is able to learn temporal dependencies. Similarily to classical autoencoders, LSTM-AEs are trained by minimizing the reconstruction error and assume that anomalous observations can be identified by high reconstruction errors [87, 88]. Several studies use LSTM-AEs for anomaly detection, see e.g. [85, 87, 88].

Configurations of LSTM-AE models differ in the literature depending on the use case. Sharma et al. (2020) [88] and Nguyen et al. (2021) [87] use two hidden layers whereas Said Elsayed et al. (2020) [85] use four hidden layers for each encoder and decoder. Common loss function in these studies are MSE or RMSE. Both Sharma et al. (2020) [88] and Said Elsayed et al. (2020) [85] select a learning rate of 0.0001 whereas Nguyen et al. (2021) [87] set the learning rate to 0.01. The learning rate determines the step size of a network when optimizing the weight matrices in order to minimize the loss function and controls the allowed change during one learning step [89]. A higher learning rate allows faster training but comes with the risk of chaotic network training which could prevent the loss function from converging to a low value [89]. Low learning rates, however, can cause slow convergence of the loss function [89].

Chapter 3

The CROME System

The purpose of this chapter is to outline the need for radiation protection at CERN and to give system understanding of the CERN Radiation Monitoring Electronics (CROME). In addition, an exemplary dataset is analyzed to evaluate the data quality and to identify potentially useful characteristics for anomaly detection.

3.1 Radiation Protection at CERN

CERN hosts a variety of particle accelerators and scientific experiments dedicated to study fundamental particles to increase the understanding of the nature of the universe [90, 91]. Due to the interaction of high-energy particles produced by the accelerators with other particles or targets of stable matter, ionizing stray radiation can be emitted [90, 91]. Thus, radiation detectors are installed in the areas of particle accelerators, experiments and targets, to protect people at CERN and the surrounding environment from exposure to radiation by monitoring the ambient dose rate [90, 91].
After 30 years, the former radiation monitoring system, called ARea CONtroller System (ARCON), reached the end of its lifecycle, necessitating the development of an efficient and modular radiation monitoring system to replace ARCON [90, 91]. Consequently, the CROME system was developed in-house as the new generation of radiation monitoring devices used for radiation protection at CERN [90, 91]. It is based on a reconfigurable System-on-Chip (SoC) architecture with an integrated Field Programmable Gate Array (FPGA) to process measurement calculations in real-time [90]. The main responsibilities of the CROME system can be summarized as follows, according to [90]:

- Continuous measurement of the ambient dose rate over a range of multiple decades without scaling
- Trigger interlocks for machinery (e.g. accelerators) and access system in case of high radiation
- Generation of visual and audible radiation alarms
- Long-term, continuous and reliable data logging

Currently, there are 150 CROME devices installed at several areas at CERN. This number will increase to a total of 700 devices in 2027. Each CROME device sends a selection of measurement variables in real-time to a supervision system, called Radiation and Environment Monitoring Unified Supervision (REMUS), which then injects the generated data

to databases. All measurements, including numerous internal variables (e.g. internal voltages), are stored internally on each device and are deleted after seven days if the free memory reaches a minimum [92]. Additionally, a few variables are calculated based on measurements, e.g. the average value of the dose rate [92]. All measurements coming from a CROME device are sampled at a device specific sampling rate (e.g. one sample per second) and are assigned to a specific timestamp [92]. Table B.1 contains a list of all available internal measurement variables. The supervision system REMUS is provided with the variables shown in Table 3.1 [92, 93]. REMUS injects the data to a database called NXCALS with channel names as shown in Table 3.1 where *DEVICE* is the device name. All measurements stored on NXCALS will remain indefinitely. Retrieving data from REMUS is possible through a system called ERGO which also allows data visualization. After selecting device, time period, and required variables, the extracted data can be exported to a local spreadsheet file.
NXCALS stands for Next CERN Accelerators Logging Service and is based on big data technologies. Data can be extracted by directly accessing the NXCALS database without having to use the supervision system REMUS. This can be done, for example, by using the CERN Service for Web based Analysis (SWAN) which is a platform for interactive data analysis in the cloud.
In comparison, however, the internal variables (Table B.1) are not logged remotely and are deleted when the internal memory is full. Nevertheless, these internal variables were recorded manually for a selection of 31 devices in two different areas and can be retrieved from a physical hard drive. Figure 3.1 summarizes the handling of the variables generated by the CROME devices.

Variable Name	Description	Channel Name	Unit
Measurement	Basic dose rate measurement	DEVICE	µSv/h
Average	Dose rate measurement with applied digital filter	DEVICE_AVG	µSv/h
Integral 1	Integral of basic measurement over defined integration time	DEVICE_INT1	µSv
Integral 2	Integral of basic measurement over defined integration time	DEVICE_INT2	µSv
Minimum	Minimum based on exponential moving average algorithm	DEVICE_MIN	µSv/h
Maximum	Maximum based on exponential moving average algorithm	DEVICE_MAX	µSv/h
Temperature	Internal device temperature	DEVICE_TEMP	°C
Humidity	Internal device humidity	DEVICE_HUMD	%
High Voltage	Voltage used to polarize ionisation chamber	DEVICE_HV	V

Table 3.1: Available variables on supervision system REMUS

One of the main goals of the development of CROME was to ensure easy maintenance and replacements of parts by realizing a highly-modular architecture [90]. CROME devices consist of four sub-systems: the CROME Monitoring and Processing Unit (CMPU), the CROME Alarm Unit (CAU), the CROME Uninterruptible Power Supply (CUPS) and the CROME Junction Box (CJB) [90]. The CMPU comprises a metallic housing enclosing the read-out electronics and the control and processing system. Another component of the CMPU is a detector, usually an ionization chamber, that is used to physically detect radiation [90, 94]. In the case of an ionization chamber as detector, it is polarized with a high voltage and generates ultra-low currents which are then used to calculate the dose rate [90, 91, 92].

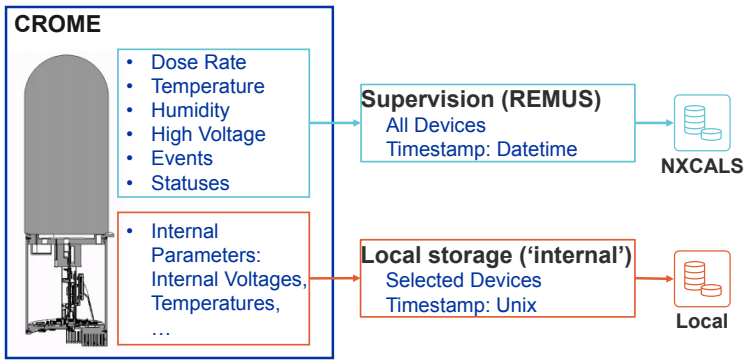


Figure 3.1: Schematic representation of the data storage of CROME variables

Since environmental conditions can affect the electrical current, a correction value is used to compensate influences of temperature and humidity [91]. The CAU is responsible to generate visual and audible alerts of three different modes: *System OK*, *Alert* and *Alarm*. The CUPS is used to provide uninterrupted power supply to CMPU and CAU at 24 V and is able to maintain the supply through an internal battery in case of failure of the main 230 V AC network [90]. The CJB centralizes the interlock signals of multiple CROME devices and communicates with the alarm and interlocking system.

One advantage of the modular architecture is that for high radiation areas the electronics can be decoupled from the detector [90]. This allows to place the electronic parts of the system in a low radiation area and to connect them to the ionization chamber located in high radiation areas. Consequently, there are two possible configurations of CROME: a bulk version for low radiation areas and a rackable version to separate electronics and detector for high radiation areas.

Since CROME is part of the radiation protection at CERN, it is a safety critical system and had to be be verified to fulfil the necessary safety integrity level SIL2 according to IEC 61508 and IEC 60532 [94]. In this context, selected hardware components (e.g. capacitors) have been upgraded to automotive or military grade and redundancies have been implemented for safety-critical subsystems [94, 95]. The verification of SIL2 was done by performing a failure rate prediction, a Failure Modes, Effects and Diagnostic Analysis (FMEDA) and a Fault Tree Analysis (FTA) [94, 95]. After evaluating each component of the CROME system, it was declared to have reached the required safety integrity level SIL2 [94, 95]. In addition, the Probability of Failures per Hour (PFH) was calculated in failures per million hour (fpmh) with the result of $PFH = 8.24 \cdot 10^{-8}\ fpmh$ [94, 95].

3.2 Statistical Analysis of CROME Measurement Data

Before applying anomaly detection algorithms to datasets generated by CROME devices, it is necessary to examine the available datasets and variables as well as the data quality. Furthermore, an exemplary dataset is statistically analyzed to discover potentially useful characteristics for the overall goal of anomaly detection. For this exploratory data analysis, as for all other analyses in this thesis, the high-level programming language Python is used.

To distinguish the data coming from the different sources, data extracted from ERGO is called *ERGO data* whereas data of internal variables obtained from manual logging is called *internal data* in the following. Lastly, datasets extracted directly from NXCALS through the web-interface SWAN are referred to as *NXCALS data*.

Since REMUS processes the data coming from the CROME devices, the timestamp column of the ERGO datasets is already converted to a date-time format in the local time zone whereas the timestamp of the internal measurements is in a raw format called *Unix timestamp*, i.e. seconds (or in this case milliseconds) since 1970/01/01 00:00:00.

All CROME devices are responsible for permanent and reliable data logging and should therefore provide measurement data for all variables without interruption as long as the device is connected to the network and in operational mode. CROME signals are discrete signals that are sampled at a specific sampling frequency. For most devices, the sampling interval of the data logging is either 1.0 s or 1.2 s.

The following data analysis is based on measurement data of one exemplary device randomly chosen out of the 31 devices with available internal data from manual recording. The selected device is a bulk version located in the area *n-TOF* with device name *PAXNT211* and a sampling interval of 1.2 s.
First, both internal and ERGO data is imported from local files for a time period of seven days. Then, the timestamp column of the ERGO dataset is converted to Unix format in order to join the two datasets on numerical timestamps. After the join, the timestamp is converted back to a date-time format allowing an easier interpretation of the results. To complete the data pre-processing, rows with missing values are removed from the dataset. These missing values can occur if the time periods of the extracted internal and ERGO datasets do not match exactly. As a result, a joined dataset with 504,000 rows for the selected time period of seven days is obtained.

To evaluate the data quality, the actual sampling interval of the measurement data is analyzed by calculating the difference of successive timestamps. As shown in Figure 3.2, the resulting sampling interval of the exemplary dataset is generally stable over time with few significant deviations. Hence, monitoring the sampling interval could be useful to detect anomalies related to interruptions in the communication of the devices.

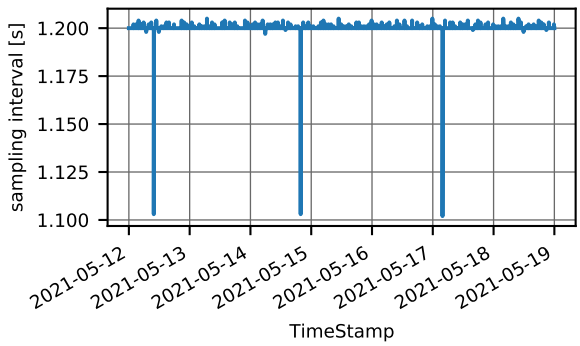


Figure 3.2: Sampling interval of the joined dataset for a period of seven days

In the next step, the correlation between various measurement variables is evaluated by calculating the Spearman correlation coefficient using the Python package *pandas* [96]. The Spearman correlation coefficient is a measure of the monotonic relationship between two variables X_1, X_2 which does not make any assumptions for properties of the underlying data distributions [97, 98]. It is calculated by applying the Pearson correlation equation to a ranked dataset [98], whereby the Pearson correlation coefficient ρ is based on covariance and

standard deviation (Equation 3.1) [99].

$$\rho_{X_1,X_2} = \frac{Cov(X_1, X_2)}{\sigma_1 \sigma_2} \tag{3.1}$$

To evaluate the dependence of the measured dose rate on the environmental parameters and to verify that the temperature compensation fulfils its intended purpose, the correlation matrix shown in Figure 3.4a is calculated. Here, *BaseValue* is the basic dose rate (DR) measurement that is sent to REMUS and *RawValue* the corresponding internal measurement in non-radiological units. The resulting high correlation between the three temperature variables is as expected and is due to the local proximity of the sensors in the devices. Moreover, dose rate measurements are not correlated with environmental variables what leads to the assumption that the temperature compensation is working correctly.

Following these findings, a scatter plot of the radiation measurement *BaseValue* versus the internal temperature *Temperature2* is created to evaluate possible dependencies more precisely (Figure 3.3). The resulting plot meets the expectations of a low correlation which arise from the correlation analysis shown in Figure 3.4a. However, a gap in the distribution of values along the *Temperature2* axis is visible (highlighted in red) leading to the assumption that the shape of the underlying data distribution could be valuable for further analyses in the scope of anomaly detection.

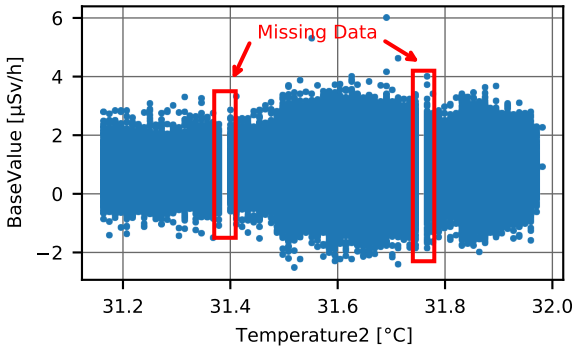
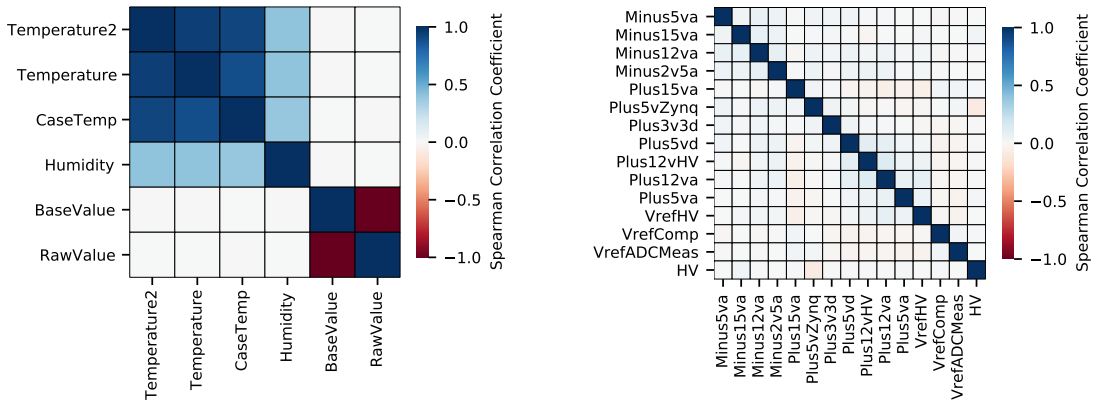


Figure 3.3: Scatter plot of the variables BaseValue and Temperature2

Figure 3.4b shows the correlations between all available voltage variables including low voltages from the internal dataset and the high voltage measurement from the ERGO dataset. It is clearly visible that all voltages are almost entirely independent from each other what could be due to the fact that the internal voltages are permanently controlled and should thus follow a stable profile with very small deviations.

To evaluate the sampling accuracy and to compare data distributions of the measurement variables, the number of unique values of each variable is calculated. For the selected dataset with 504,000 rows, the number of unique values of the internal voltages ranges from 5 to 16. The dose rate measurements of the ERGO and internal datasets have the highest number of unique values, namely 696. These very low figures, especially regarding the internal voltages, could be due to the controlled environment that the device was in during the analyzed period with only very low changes in temperature, humidity and dose rate.

In summary, it has been shown that all temperature variables are highly correlated and that the dose rate measurements are not correlated with environmental parameters. Furthermore, voltage measurements are only slightly correlated. Nevertheless, deviations of the



(a) Dose rate variables and environmental parameters

(b) All voltage variables

Figure 3.4: Correlation matrices of the analyzed variables

sampling interval and gaps in the distribution of the *Temperature2* variable were detected. The latter will be integrated in the anomaly detection process by using statistical measures describing the distribution of the values of a variable. The sampling interval will be used to the monitor data logging and the device communication.

Chapter 4

Configuration of the Wavelet Transform for Noise Extraction

In this chapter, a novel process to select the best configuration of the wavelet transform for noise extraction is presented. First, the process and the corresponding parameters are introduced. In the second section, the process is applied to real signal samples and the results are evaluated in order to select the best configuration.

Due to the absence of hardware related system failures of the CROME system, there is currently no clear definition of anomalous behavior or knowledge on failure precursors. Thus, anomalies in CROME datasets could have a wide range of properties and characteristics and could be related to one or more measurement variables. To limit the scope of the analysis, it has been decided to focus on anomalies related to noise in signals.

To handle this type of anomalies, the wavelet transform (in particular MRA-WT) has been chosen for noise extraction as it is suitable for fast-changing, non-stationary signals (see subsection 2.1.1). CROME measurement signals are generally a function of time and can thus be considered as non-stationary. Consequently, the Fourier transform is not applicable. However, the Gabor transform and the STFT are applicable to non-stationary signals but still have less favorable properties than the wavelet transforms. In particular, the wavelet transform is able to achieve better resolution for high and low frequencies due to shifted and scaled wavelet functions and allows to reconstruct the decomposed signal at every level of decomposition.
To obtain correct representations of the extracted noise signals, it is crucial to define the best combination of mother wavelet and level of decomposition for this use case. Since there is currently no clear framework on how to select these parameters (see subsection 2.1.2), a signal classification process is introduced to make this selection based on the performance and the feature importance of a classification algorithm. This process also uses wavelet transform for noise extraction, as proposed by El Menshawy et al. (2015) [24], but evaluates the extracted noise signal instead of comparing the denoised signal to the raw signal.
In general, the proposed approach is based on the following idea: take an input dataset with signal samples generated by CROME devices and modify some of these signals by adding a synthetically generated noise signal. Then, extract the noise signal for each sample by using the wavelet transform and calculate statistical measures (in the following referred to as *features*) describing these noise signals. Finally, train a GBDT classifier on this binary signal classification task to predict whether the synthetic noise signal is present in a sample or not. Using the model performance to select the best configuration assumes that the noise signal

extracted by the wavelet transform is more suitable for noise quantification if the model precision is higher and the feature importance more balanced.

4.1 Signal Classification Process

In the first step of the signal classification process (see Figure 4.1), data of selected devices and a certain time period is extracted directly from NXCALS. In step 2, the train-test-split is performed by randomly assigning the data of each device to either the training or the testing set. The relative size of the testing set is controlled by the parameter *test_size*. Next, the data is split into hourly samples based on the timestamp (step 3). This is necessary because most of the statistical measures used are calculated for an entire sample and not for individual measurement values. The hourly split provides equally sized samples which allow the comparison of the calculated statistical features.

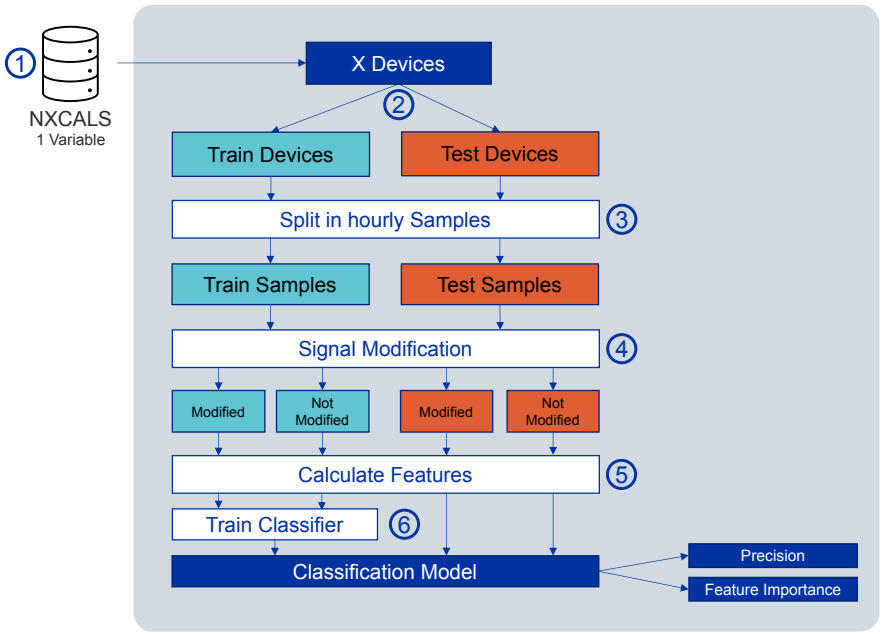


Figure 4.1: Schema of signal classification process

In step 4, a certain number of the hourly samples, controlled by the parameter *noise_prop*, is modified by adding a synthetically generated noise signal. First, the noise of the raw signal of a sample, in the following called *noise_raw*, is extracted using MRA-WT. The raw signal is then divided into 10 equally sized parts, in the following called *slices*. This allows to either modify the entire sample or only a proportion of it by selecting the number of slices to be modified using the parameter *noise_part*, ranging from 1 to 10.
Subsequently, a synthetically generated noise signal, which is based on a sine wave signal, is added to the selected number of slices (see Figure 4.2). More precisely, a sinusoidal signal with frequency $noise\_freq$ is generated taking into account the sampling interval T_s in order to achieve a correct frequency representation (see second plot in Figure 4.2). The amplitude of this sine wave is defined as a function of the standard deviation of *noise_raw* and is calculated as shown in Equation 4.1. Here, $noise\_amplitude\_factor$ is a factor that enables the scaling of the amplitude of the synthetic noise signal.

$$noise\_amplitude = \frac{std(noise\_raw)}{noise\_amplitude\_factor} \tag{4.1}$$

Additionally, a random value drawn from a normal distribution with $\mu = 0$ and $\sigma = noise\_amplitude/10$ is added to each value of the sine wave to create random deviations and to increase variability of the synthetic noise signal. An example of this signal modification is shown in Figure 4.2 with the following parameters: $noise\_part = 5$, $noise\_freq = 0.05$ (in Hertz), $T\_s = 1$ (in seconds) and $noise\_amplitude\_factor = 2$. After the signal modification, each sample is assigned a label where 0 means not modified and 1 means that a synthetic noise signal was added.

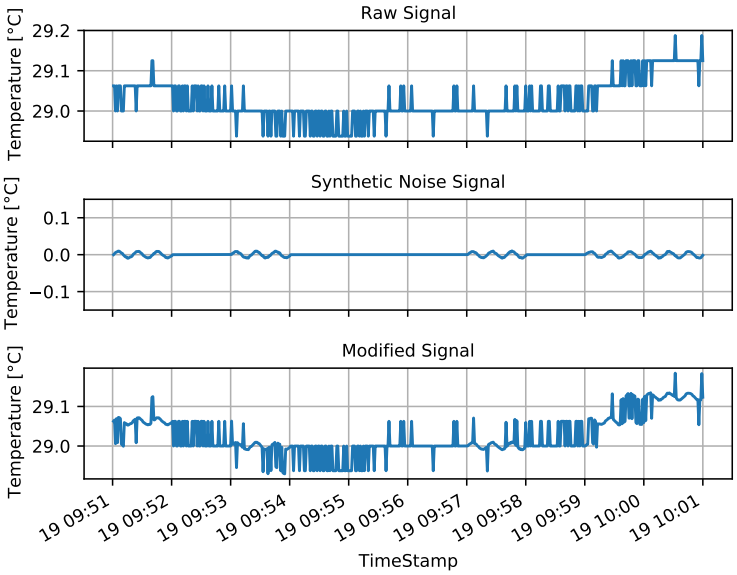


Figure 4.2: Exemplary signal modification of temperature signal: raw signal (top), synthetic noise signal used for modification (middle), modified signal (bottom)

In the following step (step 5 of the signal classification process), MRA-WT is again used with a selected configuration to extract the noise signal from each hourly sample in order to calculate a selection of statistical features which quantify the noise. Those statistical features have been selected according to research results in the field of EEG signal processing and classification (see section 2.2). However, it has been decided not to use the standard deviation for this signal classification process for the following reason: If the mean of a signal $\mu(X) = 0$, which is usually the case for the extracted noise signals as these signals oscillate around zero, then the standard deviation $\sigma(X) = rms(X)$. In this case, the standard deviation does not provide any additional information. Furthermore, the mean is already represented in the feature set. Table 4.1 shows the selected statistical features and the corresponding values of the extracted noise signal of the raw and modified signal shown in Figure 4.2. Here, db1 is used as the mother wavelet with one level of decomposition.

After calculating the statistical features, both training and testing set consist of one row per hourly sample containing the feature values and the corresponding label (0 or 1).

Variance, mean, median, RMS, number of zero crossings and the Shannon entropy are calculated by Python functions that employ the corresponding equations shown in section 2.2. For the calculation of skewness and kurtosis, separate custom functions are developed. First, the central moments of the respective orders are calculated using Equation 2.19 in order to then determine the required substitutions g_1 and g_2 for skewness and kurtosis respectively using Equation 2.17 and Equation 2.18. The functions *calculate_skewness* and *calculate_kurtosis* are then employing Equation 2.15 and Equation 2.16 to calculate the required

Feature	Raw Signal	Modified Signal
Variance	0.0003	0.0003
Mean	0.0000	0.0000
Median	0.0000	0.0000
RMS	0.1740	0.1740
Number of Zero Crossings	102	301
Shannon Entropy	0.8662	3.9965
Line Length	10.0938	10.4399
Skewness	0.0000	0.0000
Kurtosis	0.2378	0.2330

Table 4.1: Statistical features for extracted noise of raw and modified signal

values for skewness and kurtosis. Both functions have been tested on example signals and compared to the results of available out-of-the-box functions to verify the implementation of the calculations. Table 4.2 shows the comparison of resulting values from different functions for a variable X.
Lastly, the line length measure is determined by the custom function *calculate_line_length* using Equation 2.14.

Variable	Result
X	[37, 18, 32, 26, 49]
Skewness (*calculate_skewness*)	0.3727
Skewness (*SKEW*, Microsoft Excel)	0.3727
Skewness (*skew*, SciPy)	0.3727
Kurtosis (*calculate_kurtosis*)	0.1453
Kurtosis (*KURT*, Microsoft Excel)	0.1453
Kurtosis (*kurtosis*, SciPy)	0.1453

Table 4.2: Calculation of skewness and kurtosis using different functions

In the following step 6 of the signal classification process, a ML algorithm is trained on the training dataset to predict whether a sample was modified or not (i.e., whether it contains the synthetic noise or not) based on the calculated statistical features. The trained classifier is a GBDT algorithm chosen because it allows the calculation of the feature importance, which describes the contribution of each feature to the model decision. In this use case, the feature importance could also be interpreted as the ability of a feature to quantify noise in signals. Moreover, a k-NN algorithm is applied to verify the results of the GBDT model.
As the values of the statistical features could be of different orders of magnitude (see for example Table 4.1), it is necessary to scale the features before training the model. This scaling helps to improve the model performance and to remove bias, and is done by applying a standardization algorithm (Equation 2.22) which is implemented in the *scikit-learn* function *StandardScaler*. Once both training and testing datasets are scaled, the GBDT model can be trained with the training dataset and evaluated with the testing dataset.

The main purpose of this signal classification task is the selection of the most suitable configuration of the wavelet transform for this use case. As mentioned in subsection 2.1.2, mother wavelets can be chosen according to their compatibility with the signal characteristics analyzed. Furthermore, the level of decomposition can be defined based on the usefulness of the extracted features, according to [25]. Hence, in this use case, the most suitable combina-

tion of mother wavelet and level of decomposition is selected by comparing the performance and the feature importance of the GBDT classifier for various configurations of the wavelet transform.
Moreover, in order to use wavelet transforms for the extraction of noise from signals, it is also necessary to employ a thresholding technique to modify the coefficients obtained from wavelet decomposition (see subsection 2.1.2). However, as the frequency bands or other properties of the input signals are not known, common thresholding techniques are not applicable. In addition, it is not strictly necessary to obtain a noise signal that is perfectly separated from the information of the raw signal. Rather, the noise signal is extracted under the assumption that the noise is of higher frequency than the information of the signal. Hence, when considering approximation and detail coefficients as outputs of lowpass and highpass filters respectively (see subsection 2.1.1), noise extraction can be done by recomposing the signal using only the detail coefficients of all decomposition levels without modification.

Figure 4.3 shows the wavelet decomposition of the signal used in Figure 4.2 for different combinations of wavelet parameters. It is clearly visible that both the extracted noise signal recomposed from the detail coefficients cD_i and the signal recomposed from the approximation coefficients cA_i are strongly affected by the choice of the wavelet parameters. Therefore, carefully selecting the best combination of mother wavelet and level of decomposition is essential.

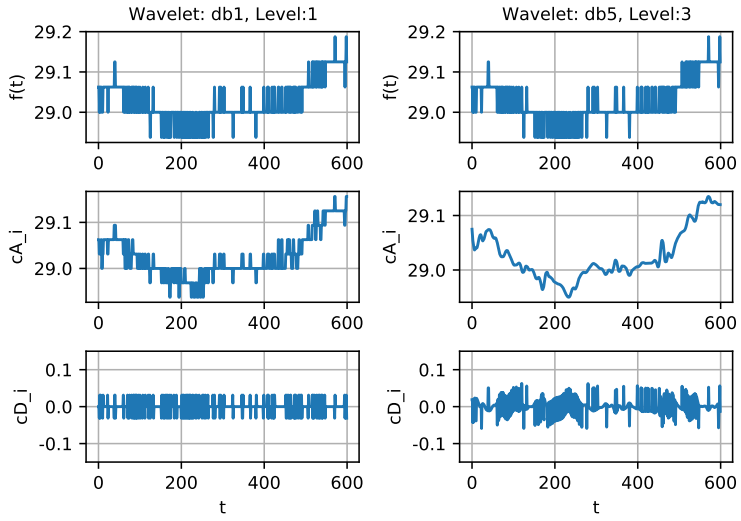


Figure 4.3: Signal decomposition using wavelet transforms with different configurations

4.2 Results and Model Analysis

To ensure that the selected configuration of the wavelet transform is applicable to various types of signals, it is necessary to cover as many scenarios as possible by including data from a variety of devices. Thus, data from a total of 54 devices located in 9 areas covering a time interval of three days (2021/05/12 - 2021/05/15) is used. Data of the four channels dose rate (DR), temperature (TEMP), humidity (HUMD) and high voltage (HV) is obtained directly from the database NXCALS. Average, Integral 1, Integral 2, Minimum and Maximum are not included in the analysis as these variables are derived from the dose rate measurement and do not contain any additional information with respect to the overall goal of anomaly detection. The train-test-split is performed with $test\_size = 0.5$ leading to equally sized

training and testing datasets. This is possible because the amount of data, if insufficient, could be extended by selecting more devices or a longer time period.
Having imported the measurement data, it is necessary to define the parameters for the signal modification step. With the purpose of achieving balanced datasets and thus to improve the model performance (see subsection 2.4.1), 50% of the samples are modified by adding a synthetic noise signal. The corresponding parameter is set accordingly: $noise\_prop = 0.5$. The proportion of noise in one modified sample is defined as $noise\_part = 5$ due to the fact that not all real signals are noisy over the entire sample. Therefore, five randomly chosen slices are modified by adding a synthetic noise signal. Additionally, the frequency and amplitude of the synthetic noise signal have to be defined carefully to create noise signals as realistic as possible. At each modification step, amplitude factor and frequency are drawn randomly from a list of possible values. In total, there are six values for each the amplitude factor and the frequency (Table 4.3) which are then randomly subdivided into separate lists for the training and testing dataset with three elements each. The noise frequency is limited by the *Nyquist Theorem* that was introduced in section 2.1. Among all 54 devices, the highest sampling interval is $T\_s = 1.2\ s$. Accordingly, the upper limit of the noise frequency is

$$f_{upper} = \frac{1}{2 \cdot T\_s} = \frac{1}{2 \cdot 1.2\ s} = 0.42\ Hz \tag{4.2}$$

The selected values for the amplitude factor and the frequency of the synthetic noise signal are listed in Table 4.3.

Property	Possbible Values
Amplitude factor	1, 2, 5, 10, 20, 50
Frequency [Hz]	0.001, 0.005, 0.01, 0.05, 0.1, 0.3

Table 4.3: Possible values for amplitude factor and frequency for synthetic noise signal

All of the above mentioned settings serve the purpose of ensuring that the model quantifies the noise in a signal instead of being trained on a certain unique signal characteristic arising from the synthetic noise signal. Properties of the synthetic noise signal are chosen randomly at each modification step and differ between training and testing dataset. The proposed process is performed for the data of each measurement variable separately to ensure that the model predictions are based on the noise quantification and not on the difference between measurement values.

To select the best suitable combination of mother wavelet and level of decomposition for this use case, the proposed process will be applied to a selection of configurations. The best configuration is then selected based on the resulting model performance and the feature importance. Mother wavelets taken into consideration are Daubechies wavelets of orders 1-20, Symlets wavelets of orders 2-10 and Coiflets wavelets of orders 1-5. The levels of decomposition used in this analysis range from one to three.

As already mentioned, a GBDT classifier is used to detect synthetic noise signals in samples. Since the comparison of the model performances between various combinations of wavelet parameters is the main objective rather than achieving the best absolute model performance, the used *GradientBoostingClassifier* is applied with default hyperparameters. GBDT algorithms are in addition robust to overfitting (see subsection 2.4.3) which allows the application without fine-tuning the hyperparameters. By default, the number of estimators that controls the number of boosting stages is set to $n\_estimators = 100$. Additional default model hyperparameters can be found on the *GradientBoostingClassifier* documentation page [100].

First, the results of the signal classification process are analyzed for using data of the temperature variable. Results for the remaining variables dose rate, humidity and high voltage can be found in the appendix (Figure C.1). Figure 4.4 shows the model precision (see Equation 2.24) for the temperature variable of the testing set for all considered combinations of wavelet parameters. It is clearly visible that the model precision strongly depends on the selected configuration. Level 1 is on average clearly related to higher model precisions. Likewise, the model precision increases in most cases with lower order of the wavelet functions. Consequently, the highest precision can be achieved using combinations of low levels of decomposition and wavelet functions of low order.

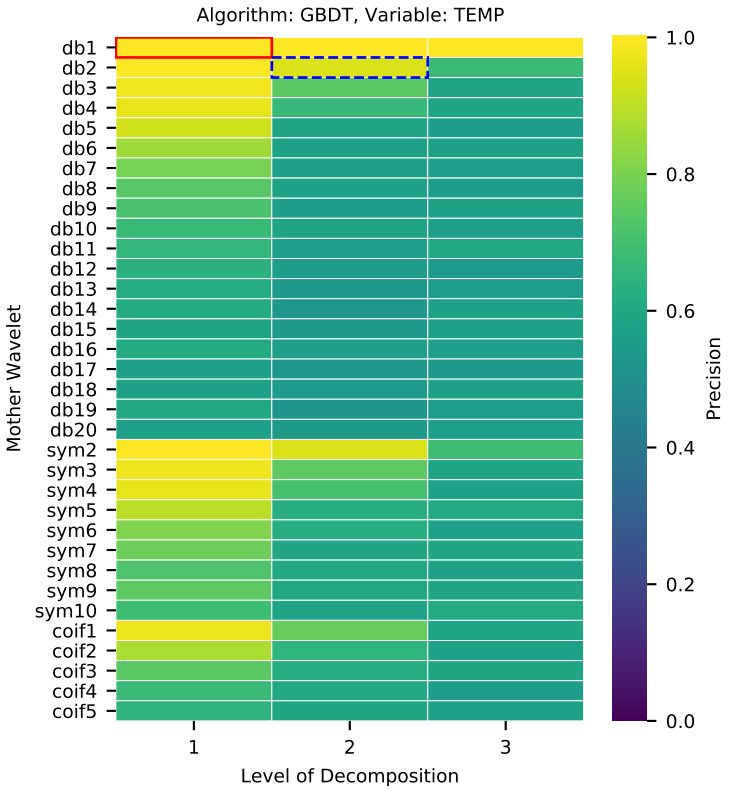


Figure 4.4: Model precision of signal classification for temperature variable (GBDT algorithm)

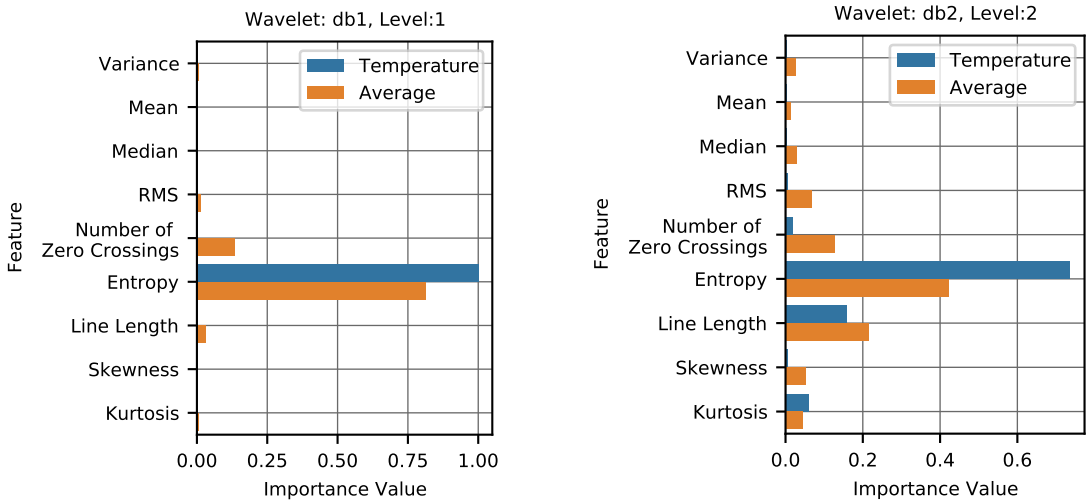
The results of the model precision for the remaining variables show a similar behavior with best precisions for low numbers of decomposition levels and wavelet functions of low orders (see Figure C.1). This is also true when calculating the model accuracy instead of the precision to evaluate the model performance, leading to the assumption that the dataset is well balanced. The resulting confusion matrix of the classification model for the temperature variable using the combination of db2 and level 2 (Table 4.4) confirms this assumption. Both predicted classes are equally sized and have similar numbers of correctly classified samples.

Employing the k-NN algorithm instead of the GBDT results in both similar rankings of wavelet parameter combinations and similar model performances. Here, the *KNeighborsClassifier* of the Python package *scikit-learn* is used with default hyperparameters and $k = \sqrt{N}$. If the resulting k is even, it is increased by 1 to achieve an odd k value. Figure C.2 shows the resulting model precision of the k-NN algorithm for the temperature variable.

Considering only the model precision, Daubechies wavelet db1 with one level of decompo-

		Predicted	
		Modified	Not Modified
Actual	Modified	929	21
	Not Modified	43	951

Table 4.4: Confusion matrix of signal classification (GBDT algorithm)



(a) Mother wavelet: db1, level of decomposition: 1

(b) Mother wavelet: db2, level of decomposition: 2

Figure 4.5: Feature importance of GBDT classifier

sition would be considered as the best suitable combination for this use case (marked in red in Figure 4.4). However, it is also decisive to have a balanced feature importance ensuring that all features contribute to the model decision. This could help to increase the detectability of a variety of noise signals not only limited to the synthetic noise signal used in this classification process. One advantage of the GBDT algorithm is that the feature importance can be evaluated after the training phase. For the combination of db1 and level 1, the feature importance assumes that the model decision is mainly based on the entropy. Especially when analyzing the model results of the temperature variable, but also on average over all four variables, the feature importance is highly unbalanced (see Figure 4.5a). This is also true for most of the other combinations of mother wavelets with one level of decomposition.
Changes in the entropy value are caused by changes in the number of unique values of a variable. As the signal modification adds random values to the raw signal, modified samples are more likely to have a high number of unique values. Other statistical measures that could quantify the noise in a signal only have a slight impact on the model decision for this configuration of the wavelet transform.

Consequently, the configuration with the best combination of high model precision and balanced feature importance, namely db2 with level 2 (marked in blue dashed in Figure 4.4) is selected for further analysis. This configuration results in and an average classification precision of 0.747 for the four variables used. Figure 4.5b confirms that the feature importance is clearly more balanced for this combination of wavelet parameters. This is true when

averaging over all four variables and when analyzing only data of the temperature variable. Under the assumption that a more balanced feature importance is more suitable and reliable for the detection of a variety of noise signals, this combination is selected as most suitable for this use case despite the decrease in model precision by approximately 20% compared to db1 and level 1.
It is noteworthy that the model precisions for this configuration for the dose rate and high voltage variable are relatively low (see Figure C.1). Nevertheless, db2 with level 2 is selected as best configuration as this decision is based on the average model performance. The low precisions for the dose rate and high voltage variable could be due to the fact that the selected properties of the synthetic noise signal allow wide variation (see Table 4.3) and thus make it extremely difficult for the model to classify the samples.

Chapter 5

Anomaly Detection in CROME Datasets

This chapter is focused on the main objective of this thesis: anomaly detection in datasets generated by CROME devices. The algorithms used to detect anomalies are Isolation Forest (iForest) and Long Short-Term Memory Autoencoder (LSTM-AE). Depending on the data sources used as input to the algorithms, two procedures will be described separately in the following subsections. Subsequently, the application of the anomaly detection algorithms to the input data will be explained and the obtained results will be analyzed.

iForest is chosen as anomaly detection algorithm because it is not a classification-based algorithm but rather an algorithm dedicated to actual anomaly detection. It is based on the isolation of rare and different samples which is assumed to be true for anomalies in this use case. Furthermore, iForest is fully unsupervised and does not require a definition of the system's healthy state. It is also applicable to high-dimensional datasets. The second algorithm, LSTM-AE, is chosen because it is commonly used for anomaly detection in time series as it promises to learn temporal dependencies. This could allow the detection of unusual correlations or inconsistencies between variables.

Based on the findings of the statistical analysis of an exemplary CROME dataset in chapter 3, it was decided to include the sampling interval and the statistical distribution of the measurement values in the anomaly detection process. Furthermore, the process proposed in this thesis should focus on anomalies related to noise in signals. Analysis of a classification model trained to detect samples with synthetic noise confirmed the ability of a selection of statistical features to quantify noise. Although not all features contribute to the model decision to the same extent, the entire feature set will be integrated into the anomaly detection approach. This is done because the type and frequency of anomalies in the datasets are not known. Therefore, it might be useful to include a variety of features in the anomaly detection to be able to detect different types of anomalies. Moreover, detected noise signals in the signal classification in chapter 4 are limited to the synthetic noise signal. Thus, features of low importance for this specific signal classification task could still be useful to detect anomalies with different characteristics. In addition, the increase in computational effort that could result from large feature sets is not the focus of this initial anomaly detection analysis. The issue of the curse of dimensionality (subsection 2.4.1) will be approached by employing a feature selection technique to reduce the number of dimensions.

In the first step of each analysis of the iForest algorithm, anomaly detection is performed using statistical features that describe the course and characteristics of the signals. Then, the

raw measurement observations are used as input to the algorithm. This separation pursues the goal of detecting both anomalies related to noise or statistical attributes and anomalies related to inadmissible measurement values or anomalous correlations. The LSTM-AE algorithm is applied to the measurement observations only in order to learn temporal dependencies.

All anomaly detection analyses are performed on distributed cluster computing resources with GPU support and are implemented in the high-level programming language Python using the web-interface SWAN. The Python packages mainly used for anomaly detection are *scikit-learn* [101] for the iForest algorithm and *tensorflow* [102] for LSTM-AE models.

5.1 Data Pre-Processing

Input data that is used in the anomaly detection algorithms is obtained from the database NXCALS or from local files containing internal measurements. As internal data is limited to 31 devices from two areas, the anomaly detection is split into two approaches: usage of data from NXCALS only or usage of both data from NXCALS and internal measurements. Data pre-processing steps of these two approaches will be explained below.

5.1.1 NXCALS data only

If only NXCALS is used as the input source, the input datasets are obtained by extracting data from a selection of devices and variables for a specific time interval. Variables from NXCALS used for anomaly detection are the dose rate measurement (DR), the temperature (TEMP), the humidity (HUMD) and the high voltage (HV).

After importing the data from NXCALS, it is checked for interruptions in the logging by comparing the actual number of extracted rows to the target number of rows. The latter is calculated based on the duration of the selected time period and the device specific sampling frequency, which is the reciprocal value of the sampling interval. If the difference between the actual number of rows and the target number is greater than 10, the corresponding device name as well as the actual and target number of rows is saved in a file that can be used to monitor logging interruptions.
However, devices with logging interruptions are not excluded from the analysis. Rather, these results can be used for root cause analyses to improve logging procedures and reduce interruptions. Nevertheless, the detection of logging interruptions contributes to the overall goal of anomaly detection, not only in terms of hardware, but also in terms of device communication and data logging.
Checking for missing or corrupted data (i.e. NaNs) is done in a later stage of the process.

At this stage of the anomaly detection process, the imported data of each device is still stored in a separate dataset. Before continuing with the data pre-processing, the actual sampling interval of the measurements is calculated and joined as an additional column to the data of each device respectively. The sampling interval column is generated by calculating the difference between successive timestamps. Since no preceding value is available for the first timestamp, the first element of the sampling interval column is assigned the mean value of the remaining sampling intervals. Sampling interval and timestamp are not used as input to anomaly detection algorithms, but are analyzed separately.
In the following step, the data is processed into two datasets: one that contains all available measurements from all selected devices stacked on each other in one single dataset, called *mesval_data*, whereas the other dataset contains hourly samples extracted from all devices, called *feature_data*. The split in hourly samples based on the timestamp is again done due to the fact that most of the statistical features are calculated for an entire sample and not

for single measurements, as mentioned in section 4.1.
The resulting datasets *mesval_data* and *feature_data* are later analyzed separately. Therefore, IDs are assigned to each sample to enable the comparison of the respective results. More precisely, each hourly sample is assigned a unique label that contains information about the device and the respective time period, e.g., DEVICENAME_H1 for the first hour of the analyzed time period. This label is then used for every measurement in *mesval_data* that belongs to this hourly sample.
During the processing of the data into *mesval_data* and *feature_data*, each hourly sample is checked for missing measurements or corrupted data. These missing values can occur if the data logging is interrupted only for one variable while measurements are available for the others. Since it is difficult to replace missing values without interfering with the anomaly detection process, hourly samples with missing values are flagged and sorted out. More precisely, the respective ID is extended by "MISSING" and all IDs containing this expression are removed from both *mesval_data* and *feature_data*. Sample exclusion is possible without much harm, as more data can be included in the process by extending the time period of the analysis, if necessary.

Feature Engineering

In the feature engineering step, selected statistical features are calculated for each hourly sample of *feature_data*. Measurements contained in *mesval_data* are later used directly for anomaly detection without engineering of additional features.

The calculation of the statistical features for *feature_data* is subdivided into two parts: calculate the features for the raw signal and for the extracted noise signal. Here, MRA-WT is used for the noise extraction. According to the results of the noise classification process in chapter 4, the wavelet transform is configured with db2 as mother wavelet and 2 levels of decomposition. Furthermore, also in this case the noise signal is generated by reconstruction of the detail coefficients only.
The selection of the statistical features is generally based on the features used in the noise classification process in chapter 4. As the mean of the raw signal is, unlike the extracted noise signal not generally zero, the standard deviation is included as feature for the raw signal. Moreover, the ratio of the standard deviation to the mean *std_mean* is calculated as an additional feature with Equation 5.1.

$$std\_mean(X) = \frac{\sigma(X)}{\mu(X)} \tag{5.1}$$

Table 5.1 shows the feature set calculated for the raw and the noise signal of each hourly sample. The Signal-to-Noise Ratio (SNR) is calculated with Equation 2.20 to compare the power of the raw signal and the extracted noise signal. The resulting dataset *feature_val* contains the values of all statistical features shown in Table 5.1 calculated for all variables. Calculating this feature set for the four variables obtained from NXCALS leads to a total of 84 features and one row of values per hourly sample.

Scaling

In order to prevent the anomaly detection algorithms from being biased towards features with high absolute values, the datasets are scaled by applying the *StandardScaler* function of the Python package *scikit-learn* which employs Equation 2.22. This scaling is applied to the data of each device separately to focus the scaling on deviations within the data of one single device and not on differences between multiple devices (e.g. different absolute values of the dose rate due to active/inactive beam).

Feature	Raw Signal	Noise signal
Variance	X	X
Standard Deviation	X	
Mean	X	X
std_mean	X	
Median	X	X
RMS	X	X
Number of Zero Crossings	X	X
Shannon Entropy	X	X
Line Length	X	X
Skewness	X	X
Kurtosis	X	X
Signal-to-Noise Ratio (SNR)	X	

Table 5.1: Statistical features used in anomaly detection for raw signal and extracted noise signal

For the *feature_val* dataset, data of one device is extracted, scaled, and saved in a new dataset *features_scaled*. This procedure centers the values of all features and scales them to unit variance. Consequently, deviations from the normal state can be detected regardless of the absolute value of the underlying variables. For example, deviations of features of the dose rate or the temperature are treated equally even if the absolute values of the dose rate are several orders of magnitude lower than those of the temperature.
However, the *mesval_data* dataset is scaled by centering only without scaling to unit variance. Again, data from each device is scaled separately to ensure that the data is centered to the mean of one device and not to the mean of all devices. This allows to use data from high and low radiation areas or different temperature levels in one algorithm by centering the data in the scaling step. However, scaling the data to unit variance could cause false predictions of the anomaly detection algorithms. The reason for this is that the variance of the signals could differ significantly between the devices. Scaling to unit variance would remove this difference and thus remove the information of the absolute values of deviations in the signal. This behavior is clearly visible in Figure 5.1 which shows the raw signal in the top, the signal scaled by centering only in the middle, and the signal scaled by using centering and scaling to unit variance in the lower plot. The underlying data is taken from *mesval_data* which contains measurements of all devices. Concatenating data of different devices causes the erratic deviation that is visible in Figure 5.1. Consequently, scaling to unit variance highly influences the output signal and removes information about absolute deviations of the signals. Thus, it is decided to scale the *mesval_data* dataset by centering only.

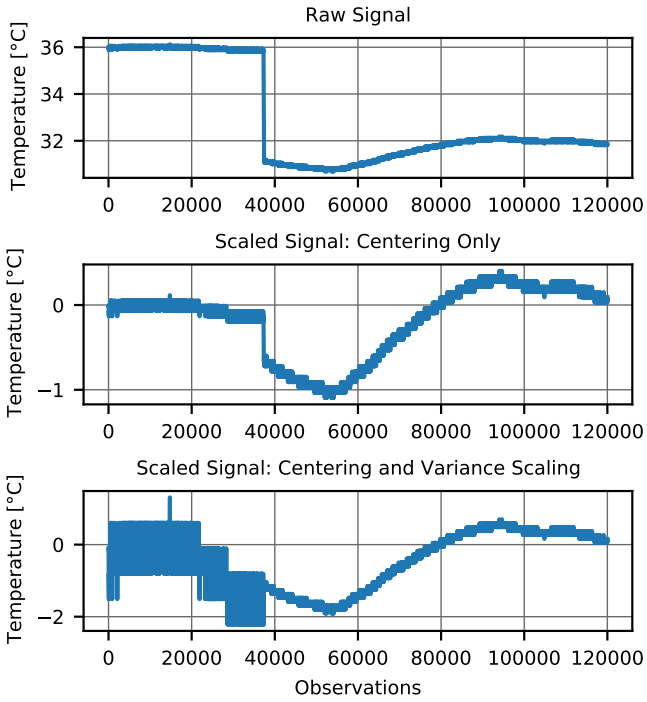


Figure 5.1: Comparison of different scaling techniques for measurement data

5.1.2 NXCALS data and internal data

Internal data containing measurements of various low voltages or other internal parameters is available for 31 devices and can be imported from local files. Since the types or characteristics of anomalies to be detected are not known, it is not known either which variables contain information allowing the identification of anomalies. Thus, it might be beneficial to include as many variables as possible in the anomaly detection process. Therefore, all numeric internal variables are used for anomaly detection except for the timestamp variable (Table B.1).
To include the internal data into the process, internal measurements are imported into the routine from local files. The available time periods are not necessarily the same for all devices due to the manual logging of the internal measurements. The minimum cross section of available time periods is used as reference for the manual selection of the analysis period. In the next step, data is retrieved from NXCALS for a selection of devices and the selected time period.
Similar to subsection 5.1.1, the imported data from NXCALS is then checked for data logging interruptions. Subsequently, the internal data is joined to the NXCALS data on the timestamp column using a left join method, meaning that the variables of the internal data are added for each measurement observation. The following data pre-processing and feature engineering steps are identical to the ones described in subsection 5.1.1. Likewise, the feature set shown in Table 5.1 is calculated for the hourly samples of *feature_data* resulting in *feature_val*. The measurement dataset *mesval_data* contains all measurements of both NXCALS and internal variables. The scaling is analogous to the procedure described in subsection 5.1.1 and results in *mesval_scaled* and *features_scaled*.

Figure 5.2 shows the structure of the resulting scaled datasets for both approaches with and without internal data. One hourly sample is related to one row in *features_scaled* whereas multiple measurement observations belong to one hourly sample in *mesval_scaled*. Each hourly sample has an unique ID which is also used for all measurement observations in

mesval_scaled that belong to this this hourly sample. The scaled datasets *mesval_scaled* and *features_scaled* can then be used in anomaly detection algorithms.

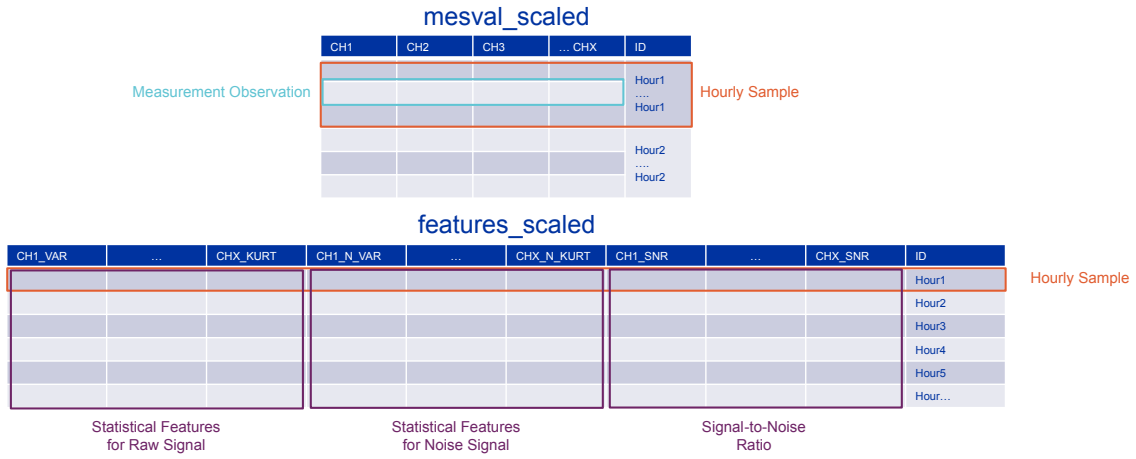


Figure 5.2: Structure of datasets used as input to anomaly detection algorithms

5.2 Application and Results of Anomaly Detection Algorithms

This section will cover the application of anomaly detection algorithms to the pre-processed datasets of the previously described process. Furthermore, results of the respective algorithms will be discussed. Both iForest and LSTM-AE will be applied to two different input datasets:

- RUN1: NXCALS and internal data for 10 devices of two areas and a time period of seven days (2021/05/08 - 2021/05/15)
- RUN2: NXCALS data only for 54 devices of nine areas for a time period of three days (2021/07/15-2021/07/18)

The devices of RUN1 were selected randomly out of the 31 devices with available internal data from manual logging. Reading internal data from local files is restricted to ten devices due to limited memory. Data extraction of RUN1 results in 1,674 hourly samples and 5,019,343 measurement observations for a total of 21 variables. RUN2 results in 3,888 hourly samples and 14,255,986 measurements of four variables.

In the first step of the anomaly detection, the data extracted from NXCALS is checked for logging interruptions by comparing the actual to the target number of rows. The data extracted for RUN1 has an insufficient number of rows for four out of ten devices. In the case of RUN2, no logging interruption is detected meaning that all devices have sufficient numbers of measurement rows.
Additionally, the sampling interval is analyzed by calculating its variance for each hourly sample. This allows to rank the hourly samples by variance of the sampling interval to detect deviations or to analyze possible logging interruptions in more detail. Analysis of the variance of the sampling interval of RUN1 allows to locate the interruptions in data logging. The difference between two successive timestamps reaches for example 1579.2 s for one sample. Contrariwise, sampling intervals of RUN2 show lower deviations with a maximum of 2.4 s.
As aforementioned, no samples are excluded from the analysis based on these findings. Instead, detected logging interruptions or deviations of the sampling interval can be used for root cause analysis to optimize the device communication. Moreover, the device's behavior

after interruptions of data logging (e.g., caused by maintenance) can be a useful scenario to include in the anomaly detection analysis.

Results of RUN1 will be discussed in the following in more detail than the results of RUN2 since datasets with internal measurements are of more relevance for anomaly detection. This is due to the fact that the controlled low voltages of the internal datasets could contain information about malfunctions.

5.2.1 Application of the Isolation Forest Algorithm

Reducing the number of input dimensions can help to improve the performance of the iForest algorithm, as stated in subsection 2.5.1. Calculating the feature set shown in Table 5.1 for all 21 variables of RUN1 leads to more than 400 features. Consequently, a subspace of these features is selected based on the kurtosis of each feature, as suggested by Liu et al. (2008) [76]. This allows a comparison of the information content of the features by evaluating the peakedness of the respective distribution. After ranking the features by kurtosis in descending order, the top 50 features are selected and used in anomaly detection.

The application of the iForest algorithm only requires to set the number of trees and the sub-sampling size. As mentioned in subsection 2.5.1, it is suitable for most applications of the iForest to set the number of trees to $t = 100$ and the sub-sampling size to $n = 256$. The iForest algorithm used in this analysis is implemented in the Python package *scikit-learn* as ensemble algorithm *IsolationForest*. Anomaly scores for a given input dataset are calculated using the custom function *iforest_scores*, which executes the routine shown in Listing 5.1. It should be noted that *iforest_scores* outputs the absolute values of the anomaly scores, since *IsolationForest* calculates the anomaly score according to Equation 2.28 but with a negative sign.

Listing 5.1: Routine of *iforest_scores*

1. Create and configure instance of IsolationForest()
2. Fit model to input data
3. Calculate anomaly scores for the input samples
4. Join anomaly scores and IDs of the input samples to create the output data frame

In the following, the results of RUN1 with both internal and NXCALS data will be discussed. First, the anomaly scores obtained from applying *iforest_scores* to the feature dataset *features_scaled* are analyzed. The histogram in Figure 5.3 shows the distribution of the resulting anomaly scores where a higher score means that a sample is considered as more anomalous. Since all scores are relatively close to 0.5, there is no sample that can clearly be identified as normal or anomalous in this analysis. Due to lack of references to interpret the anomaly scores and lack of true labels, the scores are analyzed by ranking the samples by anomaly score in descending order (Table 5.2).

Each sample is analyzed by graphing the signal curves and comparing the feature values with other samples. The latter is done by calculating the mean difference of a feature value to the values of one preceding and one subsequent reference sample, and dividing this average difference by the mean feature value of the reference samples. This allows to calculate the relative change of a feature value compared to other samples. Features with strongly deviating values can then be selected by ranking the features by the relative difference.

The sample with the highest anomaly score (PMINA0403_H116) has the most significant changes of the feature values in the kurtosis measures (i.e. peakedness) for DR and

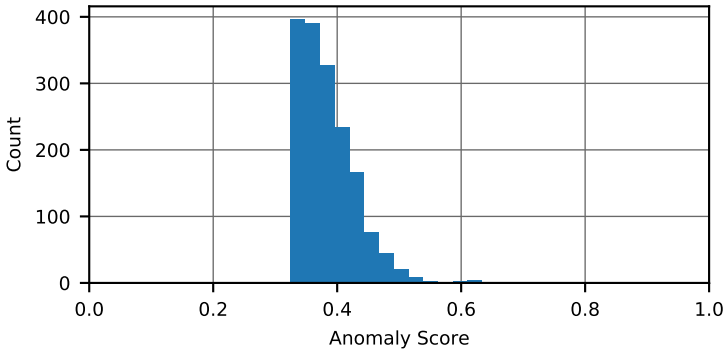


Figure 5.3: Distribution of iForest scores for *features_scaled* (RUN1)

Ranking	Sample	Score
1	PMINA0403_H116	0.6818
2	PMINA0203_H116	0.6565
3	PMIST213_H168	0.6198
4	PMIST214_H116	0.6146
5	PMIST214_H16	0.6144
⋮	⋮	⋮

Ranking	Sample	Score
⋮	⋮	⋮
6	PMIST211_H70	0.6120
7	PMIST213_H70	0.5993
8	PMIST213_H107	0.5915
9	PMINT301_H104	0.5913
10	PMINT201_H5	0.5812

Table 5.2: Samples of *features_scaled* ranked by anomaly score of the iForest algorithm (RUN1)

RawValue for both the raw and the noise signals. When plotting the DR and the RawValue for this signal, it is clearly visible that this change could be due to the spike in the signal (Figure 5.4). This behavior is also true for most of the samples in the top 10 ranking (Table 5.2). However, the three samples of the device PMIST213 show spikes in the direction of the negative dose rate (see PMIST213_H168 in Figure 5.4). The dose rate signal of the sample PMINT301_H104 contains, instead of extreme spikes, multiple negative dose rate measurements across the entire sample. The highest deviations of feature values of the tenth-ranked sample (PMINT201_H5) occur for the skewness and kurtosis measures of the variable *Temperature2* for both raw and noise signal. When analyzing the *Temperature2* signal in detail, it is visible that the measurement consists of two unique values with only a few changes between these values at the end of the sample (Figure 5.4). It is noteworthy that this behavior is only true for *Temperature2* but not for the remaining temperature signals.

Comparing the ten samples with the highest anomaly score leads to the assumption that the iForest algorithm is able to detect samples with single spikes in signals or unusual ranges of values (e.g. negative dose rate) based on the calculated statistical measures. Although negative dose rates are physically impossible and extreme spikes in measurements are rare among the analyzed samples, it still needs to be verified by a system expert whether these samples represent anomalous system behavior or not.

After applying the iForest algorithm to the feature dataset *features_scaled*, it is applied to the measurement dataset *mesval_scaled*. Thus, the anomaly score of each measurement observation is calculated and then integrated in the previous analysis of the feature dataset *features_scaled*. This is used to detect samples that have a high anomaly score for both the feature and the measurement dataset. To combine anomaly scores of both datasets, the scores of *mesval_scaled* have to be reduced to one score per hourly sample. With the

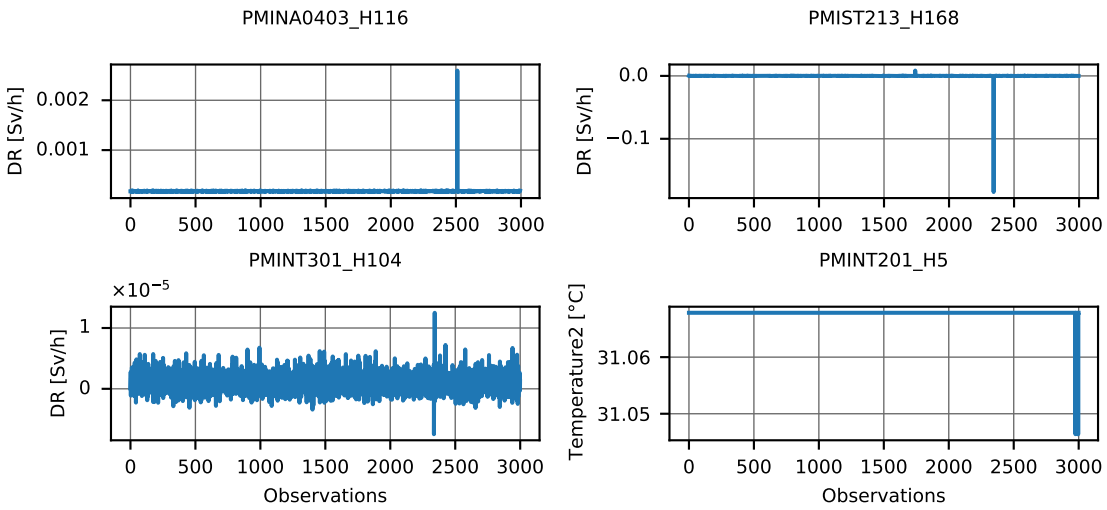


Figure 5.4: Summary of samples detected by iForest algorithm using *features_scaled* (RUN1)

goal of penalizing hourly samples having a high anomaly score for both datasets, the highest anomaly score of all observations within one hourly sample is used to reduce the number of scores to one per hourly sample. The combined score is then calculated by adding the score of *features_scaled* and the highest score per hourly sample of *mesval_scaled*. This penalizes samples that have high scores for both datasets and is possible without affecting the interpretation of the scores as the samples are analyzed by ranking and not by absolute score. Figure 5.5 shows this combination of the scores and the resulting histogram of the combined score.

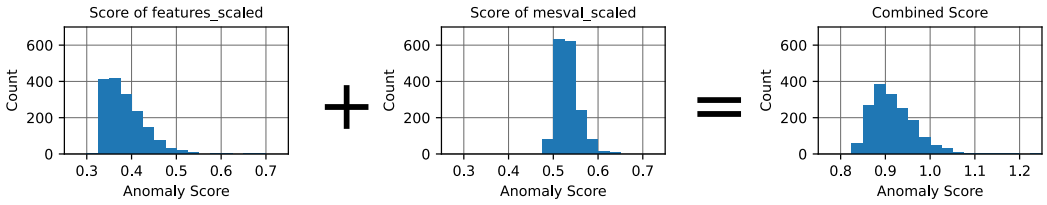


Figure 5.5: Combination of iForest scores of feature and measurement dataset (RUN1)

Following this, all hourly samples are ranked by the resulting combined score *total_score* in order to analyze possibly anomalous samples. When comparing the ranking by *total_score* to the ranking shown in Table 5.2, it is noticeable that the samples are exactly identical and only the order has changed. Therefore, the integration of the anomaly scores of *mesval_scaled* does not affect the type of samples considered as anomalous, but leads to a higher ranking of samples with negative dose rate. The low impact is also due to the narrow distribution of the scores of *mesval_scaled* with values close to 0.5, which means that no sample is clearly identified as outlier. However, if we consider the scores of *mesval_scaled* separately, the ranking of the ten most anomalous samples consists of two types of samples. The first type is related to the device PMINT301 and shows negative dose rate measurements across the entire sample (similar to PMINT301_H104 in Figure 5.4). The second type is related to the device PMINT201 and shows no significant characteristics such as negative dose rates or spikes in measurement signals. Thus, it remains unclear what caused the high anomaly score for these samples.
In summary, the inclusion of the anomaly scores of the measurement dataset *mesval_scaled*

slightly directs the focus to samples with unusual absolute values (e.g. negative dose rate) but has no significant impact on the ranking.

Data of RUN2 is analyzed with the same configuration of the iForest algorithm. The number of input features is again reduced to 50 by selecting relevant features based on the kurtosis measure. First, the results of the iForest algorithm applied to the scaled feature dataset *features_scaled* are discussed. Then, the measurement data from *mesval_scaled* is included in the analysis.

The distribution of the anomaly scores for *features_scaled* is shown in Figure 5.7 (left plot). Compared to the corresponding plot of RUN1 (Figure 5.3), the shape of this distribution is narrower and has fewer and lower outlying values. Similar to the analysis of RUN1, hourly samples are again ranked by anomaly score to analyze the samples considered as most anomalous. Most samples in the top 10 ranking show negative dose rates across the entire sample. This results in deviations of the mean value and the number of zero crossings. For some samples, this is combined with changing variance of the high voltage signal across a sample or spikes in the humidity signal (PAXL5322_H7 and PAXNT211_H13 in Figure 5.6). Other types of samples considered as anomalous show either spikes in the dose rate signal or high numbers of zero crossings of the dose rate signal. One sample in the ranking shows high dose rate measurements across the entire sample. However, when comparing the feature values to other samples of the respective device, feature values related to the dose rate signal show no significant difference except for the number of zero crossings which is unusually high. This leads to the assumption that the anomaly score is caused by the high number of zero crossings and not by the high absolute value of the dose rate.
In summary, the detected samples show similar behaviors to the samples found in the analysis of RUN1 and are mainly related to negative dose rates or spikes in signals.

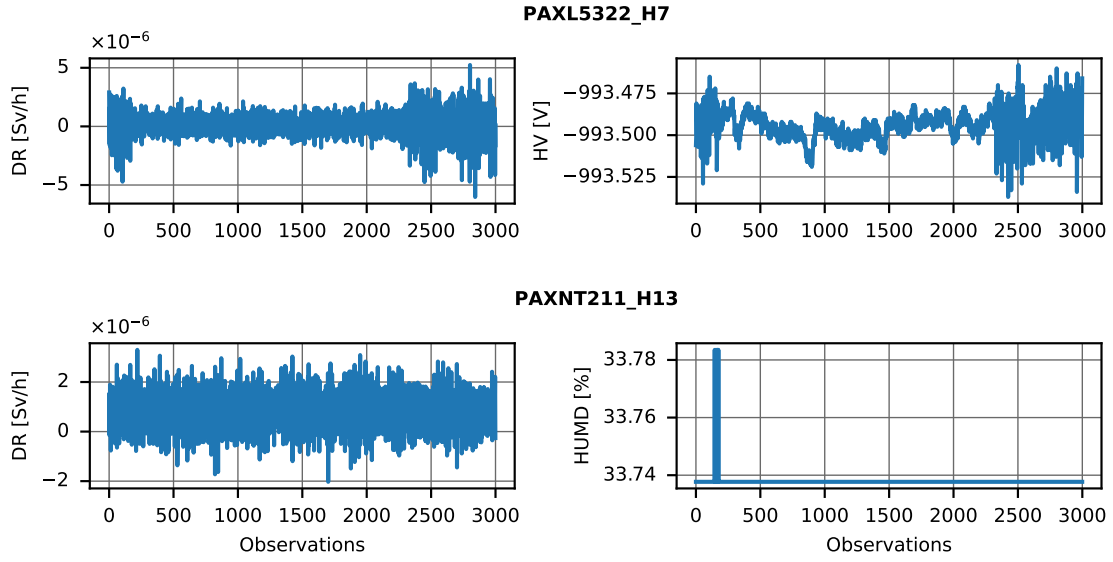


Figure 5.6: Samples of *features_scaled* considered as anomalous by iForest algorithm (RUN2)

The integration of the anomaly scores of *mesval_scaled* has in the case of RUN2 a higher impact on the ranking than in the case of RUN1. The reason for this is that the distribution of the anomaly scores of *mesval_scaled* is wider what results in higher anomaly scores for some samples (see Figure 5.7, histogram of mesval_scaled). Analysis of the anomaly scores of *mesval_scaled* shows that especially samples with high absolute dose rates are considered as

anomalous. Consequently, the focus of the sample ranking is directed towards these samples when including the anomaly scores of *mesval_scaled* in the analysis. The top ten samples ranked by combined score include many samples of the ranking of *features_scaled*, but also new samples which show high dose rate measurements.

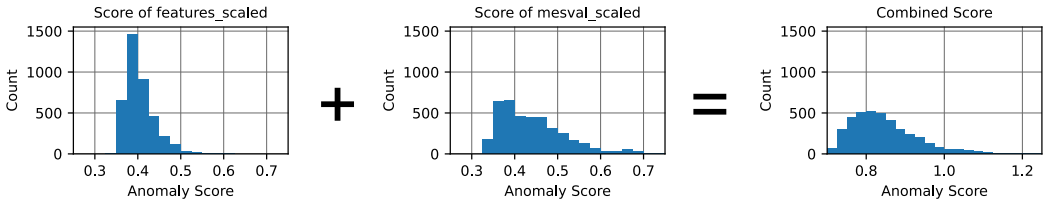


Figure 5.7: Combination of iForest scores of feature and measurement dataset (RUN2)

5.2.2 Application of the LSTM Autoencoder Algorithm

This section discusses the application and results of the LSTM-AE algorithm for the measurement datasets *mesval_scaled* of RUN1 and RUN2. LSTM-AE is not applied to the feature datasets *features_scaled* because it is used to learn temporal dependencies within the signals and not to find samples with unusual feature values.

The configuration of the autoencoder model is mainly based on Sharma et al. (2020) [88] since this research study uses input datasets with similar dimensions. Therefore, the autoencoder model is constructed with two hidden layers for each encoder and decoder resulting in four hidden layers with 16, 4, 4 and 16 neurons. Thus, the input dataset is projected to a four dimensional hidden representation. Other hyperparameters for the setup of the LSTM-AE model shown in Table 5.3 are also mainly taken from [88]. The hyperbolic tangent function is selected as activation function as it is the preferable choice according to Pawar and Attar (2020) [82].

Parameter	Value
Hidden layers	2
Hidden layer size	16,4
Activation function	tanh
Regularization	L2
Learning rate	0.0001
Loss function	MSE
Optimizer	Adam
Batch size	256
Validation split	0.1

Table 5.3: Hyperparameters of the LSTM-AE model (RUN1)

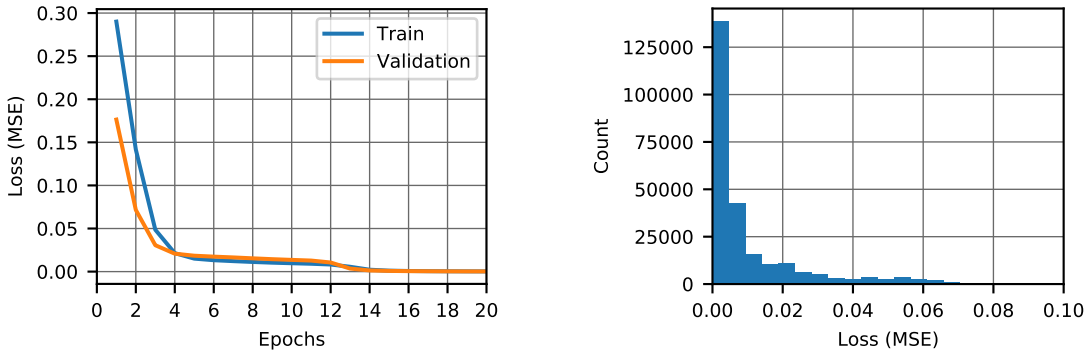
After selecting the hyperparameters of the LSTM-AE, the model architecture can be implemented in a function that creates a model instance called *lstm_ae*. This function constructs the autoencoder model and configures both the activation function and the regularization. First, the input layer is set according to the shape of the input dataset. Then, the hidden layers for encoder and decoder and the output layer with the same size as the input layer are created. Once a model instance is created using *lstm_ae*, the model has to be compiled. In the same step, both optimizer and loss function are defined. Optimizers are

responsible to update the weights and the bias of the model in order to minimize the loss function [103].
Subsequently, the model can be trained on the input data. To configure this training step, the batch size, the validation split, and the number of epochs (i.e. training iterations) need to be defined. The validation split parameter defines the fraction of the training data that is not used for training but to evaluate the model at the end of each training epoch [104]. The batch size defines the number of samples that is used per learning step [104]. The number of epochs is set initially to 20 but will be adapted according to the results of the learning behavior of the model.

Currently, there is no data available that represents the healthy state of the CROME system. All operational devices are taken into account for anomaly detection and can therefore not be considered healthy. Consequently, the autoencoder model is initially trained on pseudo-healthy data that is selected based on the results of the iForest algorithm. More precisely, hourly samples are ranked by the anomaly score of the feature dataset *features_scaled* in ascending order and the samples with the lowest anomaly score will be considered as healthy, and used to train the autoencoder model. It should be mentioned that this is only the proposed solution for this initial anomaly detection process and should not be used in operational applications. In chapter 6, hardware tests with a device considered as healthy will be used to generate more healthy data which will then be included in the training dataset of the autoencoder model.

Accordingly, for the analysis of RUN1, the top 5% of hourly samples with the lowest anomaly score, are extracted from *mesval_scaled* and used as training data. The remaining 95% of samples are used for testing. The resulting training set consists of 500,812 measurement observations whereas the testing set contains 4,518,531 observations. Figure 5.8a shows the learning curve of the LSTM-AE model for 20 epochs for the training and validation set. It is clearly visible that the learning curve converges quite well for both training and validation set after already approximately 15 epochs. Consequently, the number of epochs will be adjusted to 10. This serves also as a stopping criteria to prevent overfitting.

After resetting and training the LSTM-AE model with 10 epochs, it is used to generate the model results of the training dataset in order to calculate the loss of each training sample. The selected loss function is the MSE that is calculated with Equation 2.26. Figure 5.8b shows the distribution of the losses of the training samples.



(a) Learning curve

(b) Histogram of the training losses

Figure 5.8: Training of the LSTM-AE model (RUN1)

Since the training loss is generally close to zero, it can be assumed that the model is well

fitted to the training data. When plotting the training loss over the measurement observations (see appendix: Figure D.1), it is clear that the loss is not device specific and well distributed across the observations, what confirms this assumption. The agglomeration of scores to certain levels which is visible in Figure D.1 could be due to specific hourly samples since the pseudo-healthy training samples are taken from different devices and scenarios and can therefore have different properties.

Subsequently, the results of the trained LSTM-AE model are generated for the testing dataset. Again, the loss is calculated as MSE between the original measurement sample and the reconstructed sample and is used as anomaly score in the case of the testing set. The resulting distribution of the testing loss shows that samples of the testing set result in significantly higher reconstruction errors than the training set (Figure 5.9). Additionally, when plotting the histogram with logarithmic y axis, a local peak is visible for losses around and larger than 0.4. This peak could be an indicative of an accumulation of outliers or could be associated with measurements of few hourly samples with certain characteristics causing a high reconstruction error. Moreover, the analysis of the testing loss across the measurement observations shows that the loss is well distributed and does not form any device-specific patterns (see appendix: Figure D.2).

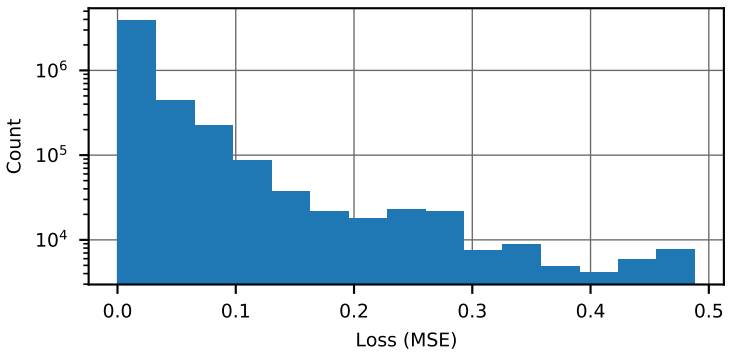


Figure 5.9: Histogram of the testing losses of LSTM-AE (RUN1)

Similarly to the analysis of *mesval_scaled* with the iForest algorithm in the previous section, the highest anomaly score per hourly sample is extracted. The top ten hourly samples ranked by maximum anomaly score in descending order are listed in Table 5.4.

Ranking	Sample	Score
1	PMINT301_H65	0.4682
2	PMINT301_H64	0.4613
3	PMINT301_H66	0.4530
4	PMINT301_H63	0.4475
5	PMINT301_H62	0.4265
⋮	⋮	⋮

Ranking	Sample	Score
⋮	⋮	⋮
6	PMINT301_H61	0.4003
7	PMINT301_H67	0.3972
8	PMINT301_H113	0.3538
9	PMINT301_H114	0.3525
10	PMINT301_H68	0.3495

Table 5.4: Samples of *mesval_scaled* ranked by highest anomaly score per hourly sample according to LSTM-AE algorithm (RUN1)

The hourly sample with the highest anomaly score does not show characteristics such as spikes in the dose rate measurement or very few unique values of a variable, as is the

case with samples detected by the iForest algorithm. However, the first-ranked sample as well as other samples in this ranking have a deviation in the signal of the internal voltage *Plus5vZynq* (see PMINT301_H65 in Figure 5.10). This voltage is usually controlled internally and has very small deviations in other samples. The remaining samples in this ranking (eg. PMINT_H113, PMINT_H114) show a direct correlation of the temperature variables and the humidty measurement. On the contrary, this correlation is inverse for other samples, e.g. the preceding and subsequent samples PMINT_H112 and PMINT_H115. Consequently, high anomaly scores are in this analysis related to unusual deviations of controlled signals or changes in the correlations between variables. Nevertheless, it still needs to be verified by system experts if these samples are just rare and different to other samples or if they actually show an anomalous behavior.

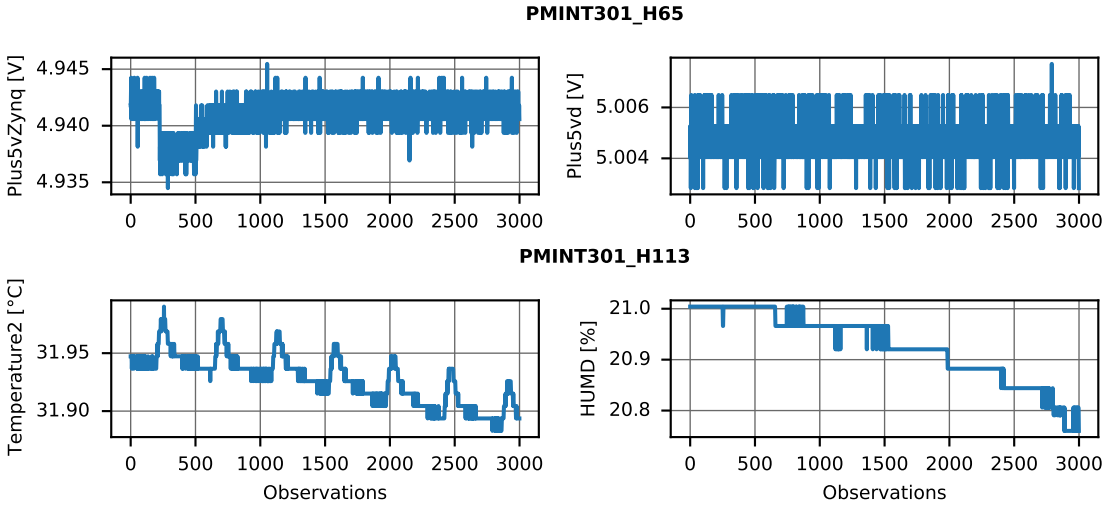


Figure 5.10: Summary of samples detected by LSTM-AE using *mesval_scaled* (RUN1)

Furthermore, autoencoders can also be used for dimensionality reduction by extracting only the encoder part from the trained LSTM-AE model. Hence, the input layer remains the same but the model output is the output of the encoder and thus the hidden representation of the input with reduced dimensions.
For the visualization of samples in the hidden representation, it is necessary to select a limited number of samples to avoid overfull plots. Thus, the hourly samples of the training set are not ranked by maximum loss but by mean loss to select the samples with the lowest average reconstruction error. Hourly samples of the testing set, however, are ranked by maximum loss per sample to select samples that have the measurement observations with the highest reconstruction errors. For both training and testing set, the top 50 hourly samples are selected according to their respective rankings, and the entire hourly samples (including all corresponding measurement observations) are then used to generate the hidden representation.
Figure 5.11 displays the hidden representation of these samples in pairwise plots of the four dimensions. Analysis of the probability densities of the four dimensions (plots on diagonal) reveals intervals in which only the test set has a high probability density. This results in separated patterns of the testing samples in all pairwise plots. Consequently, the analysis of the hidden representation shows that the LSTM-AE model projects some of the testing samples to different areas in the hidden subspace than normal samples.

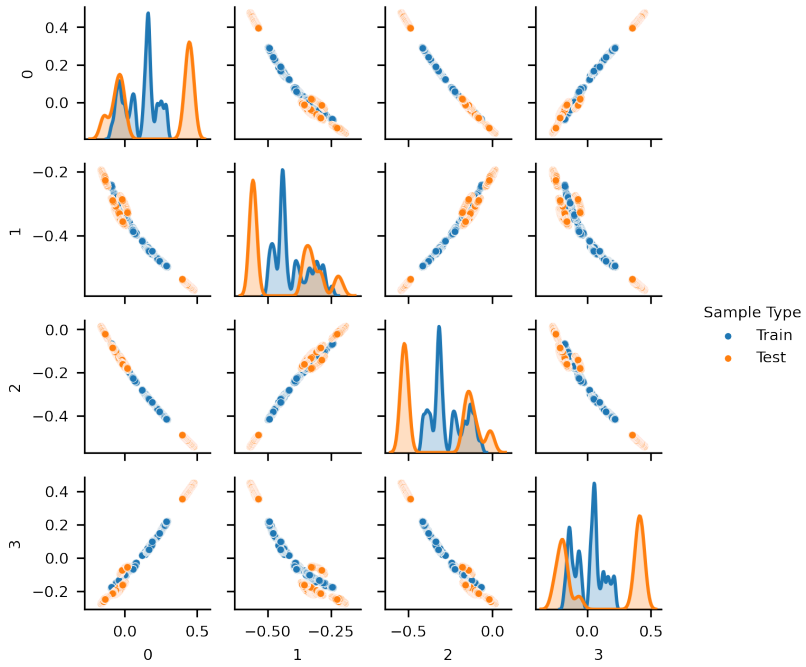


Figure 5.11: Hidden representation of the LSTM-AE model for selected samples of *mesval_scaled* (RUN1)

In the following, the results of the LSTM-AE for *mesval_scaled* of RUN2 will be discussed. The model used for this analysis is constructed with the same hyperparameters as for RUN1 (see Table 5.3), except for the size of the hidden layers which is adapted to the lower number of input dimensions. In RUN2, only the four variables extracted from NXCALS are used. Therefore, the size of the hidden layers is set to 3, 2, 2 and 3 neurons. The first two layers belong to the encoder, while the last two layers belong to the decoder, resulting in a two-dimensional hidden representation. Pseudo-healthy training data is again generated by selecting the top 5% of hourly samples of RUN2 with low anomaly score based on the results of the iForest algorithm. Training the model requires in this case 50 epochs to allow the learning curve to converge. Analysis of the training loss shows further that the model training is less stable than for RUN1, since some training samples result in a clearly higher reconstruction error than the rest of the samples.
The distribution of the testing loss shows some samples with significantly higher reconstruction errors than the remaining samples. These high reconstruction errors are linked to three hourly samples. The significance of the corresponding loss values becomes more clear when analyzing the top 5 samples ranked by maximum reconstruction error per hourly sample (Table 5.5).

Ranking	Sample	Score
1	PMIPSS42_H6	236.1976
2	PMIST212_H26	140.3530
3	PMIST211_H26	83.5858
4	PMIST211_H16	13.0568
5	PMIST211_H17	9.2047

Table 5.5: Samples of *mesval_scaled* ranked by highest anomaly score per hourly sample according to LSTM-AE algorithm (RUN2)

The sample with the highest loss shows a sharp spike of the dose rate in negative direction and also a spike in the humidity signal (PMIPSS42_H6 in Figure 5.12). Similarly, PMIST212_H26, ranked on second place, shows spikes in the humidity signal and a spike in the dose rate signal but in positive direction (Figure 5.12). The sample ranked on the third place shows a constant measurement value of the humidity and also a spike in the dose rate signal (PMIST211_H26 in Figure 5.12). Analysis of the reconstruction error over the measurement observations of the top 3 hourly samples reveals that in these cases the high loss value is always related to the measurement observation containing the spikes in the dose rate. The remaining samples in the ranking do not show any unusual characteristics.

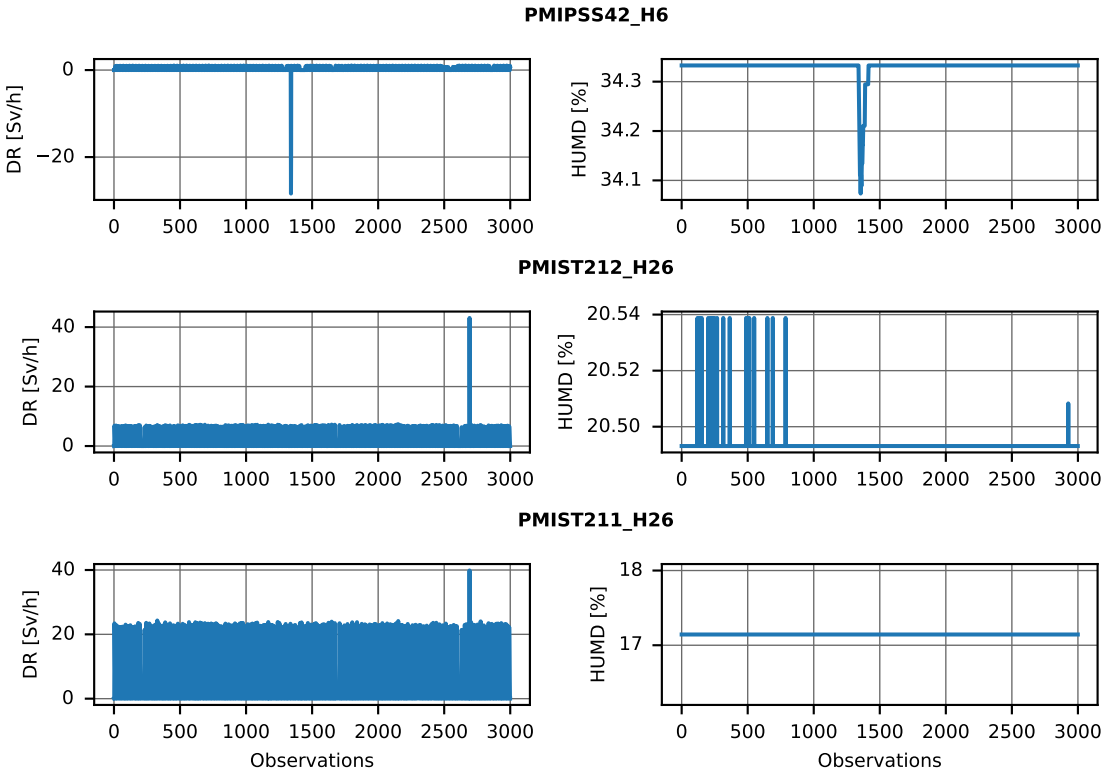


Figure 5.12: Top 3 samples detected by LSTM-AE for *mesval_scaled* of RUN2

Chapter 6

Hardware Tests to Generate Healthy Data

In the previous chapter, the LSTM-AE model was trained on field data that is assumed to represent the healthy state. This selection of pseudo-healthy data was necessary, because there is currently no data available describing the healthy state of the CROME system. One reason for this is that all devices are suspect to anomaly detection. In order to fill this gap, hardware tests are performed to generate data that describes the healthy state of the system during test scenarios. Subsequently, methods to integrate the generated data in the anomaly detection process are presented.

6.1 Test setup

The device used for the hardware tests is a standard CROME device taken from production. It is a bulk version, i.e. the CMPU is directly attached to the detector. Furthermore, it is considered as healthy although no dedicated tests have been performed to confirm this assumption. Both the CROME device and the power supply unit CUPS are placed in a climatic chamber that allows the simulation of various environmental scenarios. All collected measurements are extracted after the tests directly from the local memory of the device. Therefore, no additional data acquisition is necessary during the tests, also because only variables that are also available for operational devices are of interest.

Two test scenarios are developed to generate data of this healthy device. In the first test scenario, the temperature profile shown in Figure 6.1 with a duration of 7.5 hours is applied. This temperature profile was developed after analyzing the temperature sensitivity of the device. It has been found that the temperature profile of the internal CROME temperatures follows the profile of the ambient temperature, but requires a longer cooling phase, which is why the ambient temperature is kept at a low temperature level during the last part of the test. Moreover, the internal temperature of the device is always about 10 $°C$ higher than the ambient temperature (see Figure 6.1), which is due to internal heat generation by power dissipation.
During the second test cycle, which lasts again 7.5 hours, the same temperature profile is applied. In addition, the AC power supply of the CUPS is disconnected for a duration of 3 hours to simulate a downtime of the AC power network. In this case, the power supply of the CROME device is maintained by the integrated battery of the CUPS.
Since both test scenarios are performed without presence of any additional radiation source, the measured dose rates result only from the background radiation in the test laboratory.

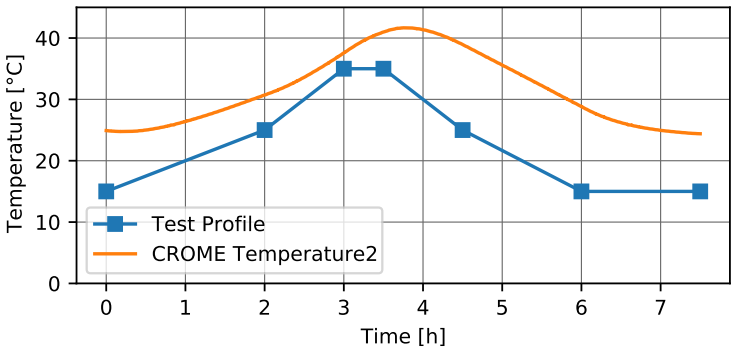


Figure 6.1: Temperature profile of hardware tests and internal CROME temperature of first test scenario

6.2 Integration of Test Data in Anomaly Detection Process

Usage of the generated healthy data is only beneficial for algorithms that learn the healthy state of a system. The results of the hardware tests will therefore only be used for the LSTM-AE algorithm. Before using the healthy data for anomaly detection, it is pre-processed similar to the datasets of RUN1 and RUN2 used in section 5.2. Thus, the data is processed into one dataset with statistical features for both the raw and noise signals, and one dataset containing the measurements. Subsequently, the data is scaled as shown in subsection 5.1.1. Again, the LSTM-AE algorithm is only applied to the measurement dataset.
There are two different ways to integrate the healthy data in the LSTM-AE model. First, it is possible to add the healthy data to the training data before the initial training of the model. In this case, pseudo-healthy data is again obtained based on the results of the iForest algorithm. The healthy data is then added and the model is trained on this combined dataset. The second possibility is to train the model first on the pseudo-healthy data, as shown in subsection 5.2.2. The integration of the generated healthy data is then possible by retraining the model only on this data.

When adding the healthy data to the training data of RUN1, the learning curve of the LSTM-AE model converges again well after 10 epochs. For RUN2, however, the learning curve does not converge and the validation loss even increases after the first few epochs. This could be due to the fact that the input data set has only four variables and the LSTM-AE model has therefore only few neurons. Thus, the number of parameters is significantly lower than for the model of RUN1, which could make the retraining more difficult.
The model results, when using both the pseudo-healthy data and the generated healthy data for the initial training, are similar to the results when using pseudo-healthy data exclusively. The ranking of the most anomalous samples is only slightly affected, which is only reflected in the order, but not in the type of samples considered anomalous. However, the integration of the generated healthy data results in higher values of both the training and the testing losses.
The learning behavior of the model, when retraining with the generated healthy data, is different than when training with the pseudo-healthy data and requires 50 epochs to allow the learning curve to converge. Nevertheless, if the model is retrained until this convergence, the results of both approaches to integrate the healthy data are almost identical. This means that in this particular use case, the behavior of the model is independent of the point in the process at which the data is integrated.

Chapter 7

Results

With the purpose of detecting anomalies related to noise in signals, wavelet transform was selected for noise extraction in order to calculate statistical measures that quantify the noise. Since there are currently no clear guidelines on how to select the best configuration of the wavelet transform, a signal classification process was developed. This process allows to compare different configurations of the wavelet transform by analyzing the performance of a classification model trained to detect synthetic noise signals in samples. The selection of the best configuration is based on the assumption that the extracted noise signal is more suitable for noise quantification if the model performance is better. Both model precision and feature importance are taken into consideration to select the most suitable configuration for this use case. As a result, the mother wavelet db2 and two levels of decomposition are selected for further usage in the anomaly detection process as this configuration results in the best combination of balanced feature importance and high model precision.

Analysis of the data logging and the sampling interval of the CROME datasets extracted from NXCALS proved to be useful to detect malfunctions and interruptions in the data logging. Subsequently, two selected anomaly detection algorithms were applied to the extracted and pre-processed datasets in order to detect observations that are different from other samples or do not comply to the expected behavior.

It has been shown that the iForest algorithm is able to detect samples with rare characteristics based on the calculated statistical features. Samples with high anomaly scores of the feature dataset *features_scaled* of RUN1 and RUN2 are related to negative dose rates, spikes in signals or other unusual behaviors. The detected characteristics are often related to extraordinary values of the statistical features, e.g., high kurtosis values for spikes in signals or high numbers of zero crossings for signals with negative dose rates. However, applying the iForest algorithm to the measurement dataset and combining the anomaly scores of *meseval_scaled* and *features_scaled* directs the focus of the ranking of anomalous samples to samples with extreme measurement values. For RUN2, this even means that some samples with high absolute dose rates are considered as anomalous which is not the intended behavior of the algorithm. Therefore, the integration of the scores of the measurement dataset of the iForest algorithm in the combined score could affect the results negatively. Besides that, anomaly scores of both RUN1 and RUN2 are all relatively close to the value 0.5. Consequently, there is no sample in the datasets of RUN1 or RUN2 that is clearly considered as anomalous or normal by the iForest algorithm.

Compared to the results of the iForest algorithm for RUN1, samples detected by the LSTM-AE model are rather related to correlations of variables or unusual deviations of

signals. Thus, it can be assumed that the model is able to learn temporal dependencies in the input dataset. This leads to higher reconstruction errors if, e.g., the correlation between two variables is not as expected. The losses of the testing set of RUN1 are relatively low and there is no sample with significantly higher reconstruction error. However, the results of RUN2 show three samples with extraordinary high values compared to the other samples. These high reconstruction errors are due to peaks in the dose rate signals with high measurement values, in the case of the sample with the highest loss even towards the negative dose rate.

Although these three samples can be considered as outliers according to the results of the LSTM-AE model, the anomaly scores of the iForest algorithm of these samples are relatively low in comparison. The sample considered as most anomalous by the LSTM-AE model is only ranked on place 28 by the iForest algorithm whereas the other two highly ranked samples are not within the top 100. It is noteworthy that the iForest algorithm does not consider these samples to be highly anomalous even if they have significantly different values for statistical features, e.g., for the kurtosis of the doserate. Nevertheless, it needs to be verified by system experts whether the samples detected by iForest or LSTM-AE show actual anomalous system behavior or if they are rare and different but compliant with the expected behavior.

Both iForest and LSTM-AE were applied to two other datasets similar to the data used in RUN1 in order to validate the results and to analyze the sensitivity of the models. The distribution of the resulting anomaly scores as well as the type of the detected samples were similar to the results of RUN1. In addition, both employed algorithms showed similar behavior as in RUN1. In principle, the results of both algorithms for RUN1 could be validated by applying them to the two other datasets. The results of RUN2, however, have not been validated since this analysis is focused on datasets containing the internal measurements.

Moreover, data that represents the healthy state of the CROME system was generated in two testing scenarios in a controlled environment. In both scenarios, a specific temperature profile is applied, and in one of the two scenarios, an AC power supply failure is simulated additionally. The generated datasets were then integrated in the LSTM-AE algorithm with two different approaches leading to similar results. It was shown that in this specific use case the model results do not depend on the time when the data is integrated in the algorithm.

Chapter 8

Conclusion and Outlook

This thesis shows the development of an anomaly detection process that is used to study the feasibility of anomaly detection for predictive maintenance in datasets generated by the radiation monitoring devices CROME at CERN.

In chapter 3, an exemplary dataset is analyzed to evaluate the data quality and the correlations between available variables. Furthermore, the statistical distributions of the variables are examined. Characteristics of potential interest in terms of anomaly detection are defined and integrated in the proposed anomaly detection process.
In order to limit the scope of the analysis, it is decided to focus on anomalies related to noise in signals. Consequently, the wavelet transform is selected to extract noise from signals. Since there are no clear guidelines in the literature for selecting the best configuration of the wavelet transform, a signal classification process is presented in chapter 4. This process allows the comparison of various configurations of the wavelet transform based on the performance of a classification algorithm that is trained to detect synthetic noise in signal samples. Statistical measures calculated for noise signals extracted by wavelet transform are used as input features for the classifier.
The proposed process allows to select the best suitable configuration of the wavelet transform for a particular use case where the wavelet transform is used for noise extraction, since real samples are used for the classification task. Furthermore, properties of the synthetic noise signals can be adjusted according to application specific requirements. Nevertheless, it should be analyzed if the separation of noise from signals, which is done by using only the detail coefficients of the wavelet transform to reconstruct the noise signal, needs to be adjusted to ensure reliable results for specific use cases. In addition, this process is limited to the synthetic noise signal based on sine waves and therefore not generally applicable.

Chapter 5 proposes a strategy to combine noise extraction using wavelet transforms and unsupervised anomaly detection algorithms. Thus, input datasets are split into two datasets: one dataset contains statistical measures describing the noise and other signal characteristics whereas the other dataset contains all measurement observations. This split allows to detect anomalies that are related to noise or statistical attributes as well as anomalies related to unusual measurement values or unexpected correlations. Data acquisition is done in a modular way via the selection of devices and time periods and can thus be adapted as required for specific analyses.
The selected anomaly detection algorithms are applied to the pre-processed datasets to analyze the types of samples considered as most anomalous. One algorithm used is the Isolation Forest (iForest) algorithm that isolates rare and deviant observations closer to the root in a binary isolation tree than normal observations. The other algorithm is a Long Short-Term

Memory Autoencoder (LSTM-AE) that learns the healthy state of a system and is able to capture temporal dependencies.
In summary, it is found that the iForest algorithm is useful to detect signals with unusual characteristics based on statistical features. Application to the measurement dataset, however, is not recommended since it could negatively affect the iForest algorithm. Although no sample with a high anomaly score clearly connected to noise in signals is detected, the iForest algorithm is able to detect samples with extreme or unusual feature values. It should therefore be useful to detect noise in signals, if present.
On the contrary, the LSTM-AE algorithm is not applicable to the feature dataset but performs better on the measurement dataset than the iForest algorithm. It is useful to detect unexpected deviations, unusual correlations or spikes in signals with extreme values.

Consequently, a combination of both algorithms could help to enhance the anomaly detection performance. One possibility could be to apply the LSTM-AE to the measurement dataset and the iForest to the feature dataset. The combination of the scores of both algorithms as well as the handling of the two algorithms in parallel should therefore be subject of further studies.
Furthermore, it is unclear why three samples clearly identified as outliers by the LSTM-AE algorithm have only a low score for the iForest algorithm, although all three samples show significant changes in values of the statistical features. A possibility to make the model results more comprehensible and explainable could help to address these problems.
In addition, the use of pseudo-healthy data that is selected based on the results of the iForest algorithm is not recommended for operational use of the LSTM-AE algorithm and is only a solution for this feasibility study to overcome the lack of healthy data. Based on the results of the hardware tests, it has been shown that data from healthy devices can be integrated in the process even by updating a model that is already trained. This would allow to integrate new knowledge in an existing algorithm by providing either data generated from healthy devices or samples considered as normal. Therefore, a feedback loop as shown in Figure 8.1 could be used to include system knowledge by manually analyzing samples that have high anomaly scores. Samples classified as normal by a system expert can thus be used to retrain the model. Classifying a sample as anomalous means in this case either that the sample was identified as anomalous in terms of failures, or that the behavior is unusual but not related to failures. In both cases, the samples classified as anomalous are not used to retrain the model. Nevertheless, it still needs to be analyzed if the proposed retraining of the model is beneficial in terms of anomaly detection and how it affects the model in the long term. Further, more hardware tests to generate healthy data that cover a wide variety of scenarios, e.g., by using additional radiation sources, are required to train the LSTM-AE algorithm properly.

Although this analysis shows that the selected anomaly algorithms are useful to detect unusual and outlying characteristics, it is not possible to define or limit the characteristics of the anomalous behavior of the CROME system. To approach this shortcoming, it would be useful to extend the availability of internal measurements of the devices. Since anomalies can not be limited to a selection of variables, it is necessary to include all available measurements in the analysis.

The proposed anomaly detection process provides methods for detecting anomalies as failure precursors that can be used for condition monitoring. However, to use the results of the analysis for data-driven predictive maintenance, it is crucial to evaluate the detected samples. Thus, it should be analyzed if the detected characteristics are related to failures or failure precursors, or if the detected behavior is just rare and different but can be considered as normal. Additionally, to use anomaly scores efficiently to make decisions about maintenance actions, a threshold value needs to be defined. This requires profound knowledge on the

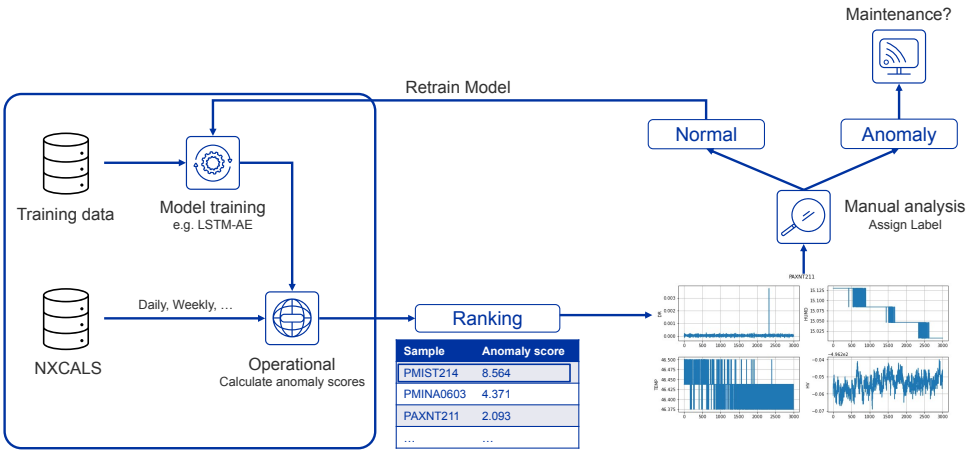


Figure 8.1: Proposed process to integrate expert knowledge in the anomaly detection process

correlation between detected characteristics and absolute anomaly score values. Generally, it is highly recommended to include knowledge from end-of-life tests or knowledge about the physical behavior of the system, before using the results of the anomaly detection algorithms to plan maintenance actions.

Bibliography

[1] R. F. Graf, *Modern dictionary of electronics*, 7th ed., r. Boston: Newnes, 1999.

[2] R. Priemer, *Introductory signal processing*. Singapore and New Jersey: World Scientific, 1990.

[3] M. H. Weik, "signal", in *Computer science and communications dictionary*, M. H. Weik, Ed., Boston and London: Kluwer Academic Publishers, 2000, pp. 1576–1577. DOI: 10.1007/1-4020-0613-6_17334.

[4] S. M. Alessio, *Digital signal processing and spectral analysis for scientists: Concepts and applications*, ser. Signals and communication technology. Cham: Springer, 2015.

[5] A. J. Butterfield and J. Szymanski, Eds., *A Dictionary of Electronics and Electrical Engineering*, 5 ed., ser. Oxford Reference. Oxford: Oxford University Press, 2018, vol. 1. DOI: 10.1093/acref/9780198725725.001.0001.

[6] M. H. Weik, "Nyquist theorem", in *Computer science and communications dictionary*, M. H. Weik, Ed., Boston and London: Kluwer Academic Publishers, 2000, p. 1127. DOI: 10.1007/1-4020-0613-6_12654.

[7] E. Sejdić, I. Djurović, and J. Jiang, "Time-frequency feature representation using energy concentration: An overview of recent advances", *Digital Signal Processing: A Review Journal*, vol. 19, no. 1, pp. 153–183, 2009. DOI: 10.1016/j.dsp.2007.12.004.

[8] K. Gröchenig, *Foundations of Time-Frequency Analysis*, ser. Applied and Numerical Harmonic Analysis. Boston, MA: Birkhäuser Boston, 2001. DOI: 10.1007/978-1-4612-0003-1.

[9] L. Debnath and F. A. Shah, *Wavelet transforms and their applications*. New York: Birkhäuser, 2014.

[10] T. Hsu, *Fourier Series, Fourier Transforms, and Function Spaces : A Second Course in Analysis*. San José, CA: American Mathematical Society, 2020.

[11] H. Takahashi *et al.*, "Low complexity fourier transform using double square-waves", in *2007 International Symposium on Communications and Information Technologies*, 2007, pp. 921–925. DOI: 10.1109/ISCIT.2007.4392147.

[12] Y. Yuchuan Wei, Q. Qishan Zhang, and Q. Zhang, *Common Waveform Analysis : A New and Practical Generalization of Fourier Analysis*. New York, NY: Springer, 2000.

[13] A. Ben Abbes *et al.*, "Comparative study of three satellite image time-series decomposition methods for vegetation change detection", *European Journal of Remote Sensing*, vol. 51, no. 1, pp. 607–615, 2018. DOI: 10.1080/22797254.2018.1465360.

[14] M. R. Canal, "Comparison of Wavelet and Short Time Fourier Transform Methods in the Analysis of EMG Signals", *Journal of Medical Systems*, vol. 34, no. 1, pp. 91–94, 2010. DOI: 10.1007/s10916-008-9219-8.

[15] J. Salvador and H. de Bruin, "The use of the wavelet transform in EMG M-wave pattern classification", *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 2006, pp. 2304–2307, 2006. DOI: 10.1109/IEMBS.2006.259534.

[16] P. Reiter, *Cloud Detection through Wavelet Transforms in Machine Learning and Deep Learning*, Jul. 2020.

[17] S. G. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, 1989. DOI: 10.1109/34.192463.

[18] B. Martínez and M. A. Gilabert, "Vegetation dynamics from NDVI time series analysis using the wavelet transform", *Remote Sensing of Environment*, vol. 113, no. 9, pp. 1823–1842, 2009. DOI: 10.1016/j.rse.2009.04.016.

[19] M. K. Kiymik *et al.*, "Comparison of STFT and wavelet transform methods in determining epileptic seizure activity in EEG signals for real-time application", *Computers in biology and medicine*, vol. 35, no. 7, pp. 603–616, 2005. DOI: 10.1016/j.compbiomed.2004.05.001.

[20] R. Gouriveau, K. Medjaher, and N. Zerhouni, *From Prognostics and Health Systems Management to Predictive Maintenance 1: Monitoring and Prognostics*. Wiley, Oct. 2016, vol. 4, pp. 1–163. DOI: 10.1002/9781119371052.

[21] L. Guo *et al.*, "Automatic feature extraction using genetic programming: An application to epileptic EEG classification", *Expert Systems with Applications*, vol. 38, no. 8, pp. 10 425–10 436, Aug. 2011. DOI: 10.1016/j.eswa.2011.02.118.

[22] A. Abbate, C. M. DeCusatis, and P. K. Das, "Time-Frequency Analysis of Signals", in *Wavelets and Subbands*, Birkhäuser Boston, 2002, pp. 103–187. DOI: 10.1007/978-1-4612-0113-7_3.

[23] H. Adeli, Z. Zhou, and N. Dadmehr, "Analysis of EEG records in an epileptic patient using wavelet transform", *Journal of Neuroscience Methods*, vol. 123, no. 1, pp. 69–87, Feb. 2003. DOI: 10.1016/S0165-0270(02)00340-0.

[24] M. EL Menshawy, A. Benharref, and M. Serhani, "An automatic mobile-health based approach for EEG epileptic seizures detection", *Expert Systems with Applications*, vol. 42, no. 20, pp. 7157–7174, 2015. DOI: 10.1016/j.eswa.2015.04.068.

[25] N. K. Al-Qazzaz *et al.*, "Selection of mother wavelet functions for multi-channel EEG signal analysis during a working memory task", *Sensors*, vol. 15, no. 11, pp. 29 015–29 035, 2015. DOI: 10.3390/s151129015.

[26] G. Lee *et al.*, "PyWavelets: A Python package for wavelet analysis", *Journal of Open Source Software*, vol. 4, no. 36, p. 1237, Apr. 2019. DOI: 10.21105/joss.01237.

[27] C. Dumitrescu *et al.*, "Application of the wavelet transform in machine-learning", *UPB Scientific Bulletin, Series A: Applied Mathematics and Physics*, vol. 76, no. 4, pp. 167–178, 2014.

[28] H. Boukabache *et al.*, "Networked Modular Technology for Integrated Aircraft Health Monitoring: Application to Rotary Structures", in *European Conference of the Prognostics and Health Management Society (PHM2014)*, Nantes, France, Jul. 2014, pp. 672–680. DOI: 10.13140/2.1.3318.7528.

[29] H. Boukabache *et al.*, "Wavlet Decomposition based Diagnostic for Structural Health Monitoring on Metallic Aircrafts: Case of Crack Triangulation and Corrosion Detection", *International Journal of Prognostics and Health Management*, vol. 4, no. 3, Feb. 2013.

[30] X. Ren *et al.*, "Noise reduction based on ICA decomposition and wavelet transform for the extraction of motor unit action potentials", *Journal of Neuroscience Methods*, vol. 158, no. 2, pp. 313–322, Dec. 2006. DOI: 10.1016/j.jneumeth.2006.06.005.

[31] J. Rafiee *et al.*, "Wavelet basis functions in biomedical signal processing", *Expert Systems with Applications*, vol. 38, no. 5, pp. 6190–6201, May 2011. DOI: 10.1016/j.eswa.2010.11.050.

[32] V. Djordjevic *et al.*, "Feature extraction and classification of EEG sleep recordings in newborns", in *2009 9th International Conference on Information Technology and Applications in Biomedicine*, 2009, pp. 1–4. DOI: 10.1109/ITAB.2009.5394439.

[33] K.-L. Tsui *et al.*, "Prognostics and Health Management: A Review on Data Driven Approaches", *Mathematical Problems in Engineering*, vol. 2015, pp. 1–17, May 2015. DOI: 10.1155/2015/793161.

[34] J. Kevric and A. Subasi, "Comparison of signal decomposition methods in classification of EEG signals for motor-imagery BCI system", *Biomedical Signal Processing and Control*, vol. 31, pp. 398–406, Jan. 2017. DOI: 10.1016/j.bspc.2016.09.007.

[35] B. R. Greene *et al.*, "A comparison of quantitative EEG features for neonatal seizure detection", *Clinical Neurophysiology*, vol. 119, no. 6, pp. 1248–1261, Jun. 2008. DOI: 10.1016/j.clinph.2008.02.001.

[36] S. Czesla *et al.*, "PyA: Python astronomy-related packages", *Astrophysics Source Code Library, record ascl:1906.010*, Jun. 2019.

[37] D. Galar and U. Kumar, "Preprocessing and Features", *eMaintenance*, pp. 129–177, Jan. 2017. DOI: 10.1016/B978-0-12-811153-6.00003-8.

[38] H. Zenil *et al.*, "A decomposition method for global evaluation of shannon entropy and local estimations of algorithmic complexity", *Entropy*, vol. 20, no. 8, p. 605, Aug. 2018. DOI: 10.3390/e20080605.

[39] G. J. G. Upton and I. Cook, *A dictionary of statistics*, 2 rev. ed., ser. Oxford paperback reference. Oxford: Oxford University Press, 2008.

[40] M. J. Blanca *et al.*, "Skewness and kurtosis in real data samples", *Methodology*, vol. 9, no. 2, pp. 78–84, Jan. 2013. DOI: 10.1027/1614-2241/a000057.

[41] D. N. Joanes and C. A. Gill, "Comparing measures of sample skewness and kurtosis", *Journal of the Royal Statistical Society Series D: The Statistician*, vol. 47, no. 1, pp. 183–189, Apr. 1998. DOI: 10.1111/1467-9884.00122.

[42] R. Kieser, P. Reynisson, and T. J. Mulligan, "Definition of signal-to-noise ratio and its critical role in split-beam measurements", *ICES Journal of Marine Science*, vol. 62, no. 1, pp. 123–130, Jan. 2005. DOI: 10.1016/j.icesjms.2004.09.006.

[43] M. Welvaert and Y. Rosseel, "On the definition of signal-to-noise ratio and contrast-to-noise ratio for fMRI data", *PLoS ONE*, vol. 8, no. 11, 2013. DOI: 10.1371/journal.pone.0077089. [Online]. Available: www.plosone.org.

[44] N.-H. Kim, D. An, and J.-H. Choi, *Prognostics and Health Management of Engineering Systems*. Cham: Springer International Publishing, Oct. 2017, pp. 1–24. DOI: 10.1007/978-3-319-44742-1.

[45] G. Niu, *Data-Driven Technology for Engineering Systems Health Management*. Singapore: Springer Singapore, Jan. 2017. DOI: 10.1007/978-981-10-2032-2.

[46] DIN Deutsches Institut für Normung e.V., *DIN EN 13306:2018-02*, Berlin, 2018. DOI: https://dx.doi.org/10.31030/2641990.

[47] B. Bertsche, M. Dazer, and M. Henß, "Betriebsfestigkeit und Systemzuverlässigkeit", in *Handbuch Konstruktion*, Carl Hanser Verlag GmbH & Co. KG, Jun. 2018, pp. 107–133. DOI: doi:10.3139/9783446456198.005.

[48] M. G. Pecht and M. Kang, Eds., *Prognostics and Health Management of Electronics*. Chichester, UK: John Wiley and Sons Ltd, Sep. 2018. DOI: 10.1002/9781119515326.

[49] H. M. Hashemian, "State-of-the-Art Predictive Maintenance Techniques", *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 1, pp. 226–236, Jan. 2011. DOI: 10.1109/TIM.2010.2047662.

[50] R. E. Neapolitan and X. Jiang, *Artificial Intelligence : With an Introduction to Machine Learning, Second Edition*. Milton, UK: CRC Press LLC, 2018.

[51] K. R. Chowdhary, *Fundamentals of artificial intelligence*. New Delhi: Springer Nature, 2020.

[52] Z. Nagy, *Artificial intelligence and machine learning fundamentals: Develop Real-World Applications Powered by the Latest AI Advances*. Birmingham, UK: Packt Publishing, 2018.

[53] W. J. Raynor, *The International Dictionary of Artificial Intelligence*. 2020. DOI: 10.4324/9781315074108.

[54] M. Swamynathan, *Mastering Machine Learning with Python in Six Steps*. Berkeley, CA: Apress, 2017. DOI: 10.1007/978-1-4842-2866-1.

[55] L. Felsberger, "Quantitative methods for data driven reliability optimization of engineered systems", Ph.D. dissertation, LMU München, 2021.

[56] O. Simeone, "A Brief Introduction to Machine Learning for Engineers", *Foundations and Trends in Signal Processing*, vol. 12, no. 3-4, pp. 200–431, Sep. 2017.

[57] F. Kuo and I. Sloan, "Lifting the Curse of Dimensionality", *Notices of the AMS*, vol. 52, pp. 1320–1328, Dec. 2005.

[58] M. Verleysen and D. François, "The Curse of Dimensionality in Data Mining and Time Series Prediction", in *Computational Intelligence and Bioinspired Systems*, J. Cabestany, A. Prieto, and F. Sandoval, Eds., Springer Berlin Heidelberg, 2005, pp. 758–770. DOI: 10.1007/11494669_93.

[59] C. Sammut and G. I. Webb, Eds., *Encyclopedia of Machine Learning*. Boston, MA: Springer US, 2010. DOI: 10.1007/978-0-387-30164-8.

[60] A. E. Hassanien, Ed., *Machine Learning Paradigms: Theory and Application*, ser. Studies in Computational Intelligence. Cham: Springer International Publishing, 2019, vol. 801. DOI: 10.1007/978-3-030-02357-7.

[61] P. Chaudhari, H. Agarwal, and V. Bhateja, "Data augmentation for cancer classification in oncogenomics: an improved KNN based approach", *Evolutionary Intelligence*, vol. 14, no. 2, pp. 489–498, 2021. DOI: 10.1007/s12065-019-00283-w.

[62] A. B. Hassanat *et al.*, "Solving the Problem of the K Parameter in the KNN Classifier Using an Ensemble Learning Approach", Sep. 2014.

[63] M. Jirina and M. Jirina, "Classifiers Based on Inverted Distances", in *New Fundamental Technologies in Data Mining*, InTech, Jan. 2011. DOI: 10.5772/13971.

[64] W. Lee, J. Lee, and K. Han, "Finding Potential RNA Aptamers for a Protein Target Using Sequence and Structure Features", in *Intelligent Computing Theories and Application*, D.-S. Huang *et al.*, Eds., Cham: Springer International Publishing, 2018, pp. 888–892.

[65] N. Syam and R. Kaul, *Machine Learning and Artificial Intelligence in Marketing and Sales*. Emerald Publishing Limited, Mar. 2021. DOI: 10.1108/9781800438804.

[66] Y. Liu *et al.*, *Advances in Natural Computation, Fuzzy Systems and Knowledge Discovery*, Y. Liu *et al.*, Eds., ser. Advances in Intelligent Systems and Computing. Cham: Springer International Publishing, Jan. 2020, vol. 1074. DOI: 10.1007/978-3-030-32456-8.

[67] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY: Springer New York, 2009. DOI: 10.1007/978-0-387-84858-7.

[68] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice", *Neurocomputing*, vol. 415, pp. 295–316, Jul. 2020. DOI: 10.1016/j.neucom.2020.07.061.

[69] M. Feurer and F. Hutter, "Hyperparameter Optimization", in, F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., Cham: Springer International Publishing, 2019, pp. 3–33. DOI: 10.1007/978-3-030-05318-5_1.

[70] M. Claesen and B. De Moor, "Hyperparameter Search in Machine Learning", Feb. 2015.

[71] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey", *ACM Computing Surveys*, vol. 41, no. 3, 2009. DOI: 10.1145/1541880.

[72] R. Chalapathy and S. Chawla, "Deep Learning for Anomaly Detection: A Survey", Jan. 2019.

[73] A. Archimbaud, "Statistical methods for outlier detection for high-dimensional data", Ph.D. dissertation, Université de Toulouse, 2018.

[74] D. M. Hawkins, *Identification of Outliers*. Dordrecht: Springer Netherlands, 1980. DOI: 10.1007/978-94-015-3994-4.

[75] E. Barelli and E. Ottaviani, "Unsupervised Anomaly Detection for Hard Drives", *PHM Society European Conference*, vol. 6, no. 1, pp. 7–7, Jun. 2021. DOI: 10.36001/phme.2021.v6i1.2795.

[76] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation forest", *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, 2008. DOI: 10.1109/ICDM.2008.17.

[77] S. Ahmed *et al.*, "Unsupervised Machine Learning-Based Detection of Covert Data Integrity Assault in Smart Grid Networks Utilizing Isolation Forest", *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 10, pp. 2765–2777, Oct. 2019. DOI: 10.1109/TIFS.2019.2902822.

[78] Z. Ding and M. Fei, "An Anomaly Detection Approach Based on Isolation Forest Algorithm for Streaming Data using Sliding Window", in *IFAC Proceedings Volumes*, vol. 3, Elsevier, Jan. 2013, pp. 12–17. DOI: 10.3182/20130902-3-CN-3020.00044.

[79] D. Xu *et al.*, "An Improved Data Anomaly Detection Method Based on Isolation Forest", in *2017 10th International Symposium on Computational Intelligence and Design (ISCID)*, vol. 2, Institute of Electrical and Electronics Engineers Inc., Feb. 2018, pp. 287–291. DOI: 10.1109/ISCID.2017.202.

[80] W. Yan and L. Yu, "On Accurate and Reliable Anomaly Detection for Gas Turbine Combustors: A Deep Learning Approach", in *Proceedings of the Annual Conference of the PHM Society*, vol. 7, Prognostics and Health Management Society, Aug. 2015, pp. 440–447. DOI: https://doi.org/10.36001/phmconf.2015.v7i1.2655.

[81] M.-P. Hosseini *et al.*, "Deep Learning Architectures", in *Deep Learning: Concepts and Architectures*, W. Pedrycz and S.-M. Chen, Eds., Cham: Springer International Publishing, 2020, pp. 1–24. DOI: 10.1007/978-3-030-31756-0_1.

[82] K. Pawar and V. Z. Attar, "Assessment of Autoencoder Architectures for Data Representation", in *Deep Learning: Concepts and Architectures*, W. Pedrycz and S.-M. Chen, Eds., Cham: Springer International Publishing, 2020, pp. 101–132. DOI: 10.1007/978-3-030-31756-0_4.

[83] Z. Chen *et al.*, "Autoencoder-based network anomaly detection", in *Wireless Telecommunications Symposium*, IEEE Computer Society, May 2018, pp. 1–5. DOI: 10.1109/WTS.2018.8363930.

[84] C. Zhou and R. C. Paffenroth, "Anomaly Detection with Robust Deep Autoencoders", in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA: ACM, Aug. 2017, pp. 665–674. DOI: 10.1145/3097983.3098052.

[85] M. Said Elsayed *et al.*, "Network Anomaly Detection Using LSTM Based Autoencoder", in *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, New York, NY, USA: ACM, Nov. 2020, pp. 37–45. DOI: 10.1145/3416013.3426457.

[86] M. Henß and B. Bertsche, "Autoencoder basierte automatisierte Zustandsdiagnose von Wälzlagern", in *VDI Berichte*, vol. 2345, Nürtingen, 2019, pp. 13–28. DOI: 10.51202/9783181023457-13.

[87] H. D. Nguyen *et al.*, "Forecasting and Anomaly Detection approaches using LSTM and LSTM Autoencoder techniques with the applications in supply chain management", *International Journal of Information Management*, vol. 57, Apr. 2021. DOI: 10.1016/j.ijinfomgt.2020.102282.

[88] B. Sharma, P. Pokharel, and B. Joshi, "User Behavior Analytics for Anomaly Detection Using LSTM Autoencoder-Insider Threat Detection", in *ACM International Conference Proceeding Series*, New York, NY, USA: ACM, 2020. DOI: 10.1145/3406601.3406610.

[89] E. J. Rzempoluck, *Neural Network Data Analysis Using Simulnet™*. New York, NY: Springer New York, 1998. DOI: 10.1007/978-1-4612-1746-6.

[90] H. Boukabache *et al.*, "Toward a Novel Modular Architecture for CERN Radiation Monitoring", *Radiation Protection Dosimetry*, vol. 173, no. 1-3, pp. 240–244, Apr. 2017. DOI: 10.1093/rpd/ncw308.

[91] J. M. Szumega, H. Boukabache, and D. Perrin, "Neural network approach for efficient calculation of the current correction value in the femtoampere range for a new generation of ionizing radiation monitors at CERN", *Radiation Physics and Chemistry*, vol. 188, no. 109539, Nov. 2021. DOI: 10.1016/j.radphyschem.2021.109539.

[92] M. Widorski, "CMPU Functional Requirements", CERN, Tech. Rep., 2020.

[93] M. Widorski, "Project definition and User requirements - Integration of CROME into REMUS", CERN, Tech. Rep., 2016.

[94] S. Hurst, H. Boukabache, and D. Perrin, "Overview of a Complete Hardware Safety Integrity Verification According to IEC 61508 for the CERN Next Generation of Radiation Monitoring Safety System", in *Proceedings of the 29th European Safety and Reliability Conference (ESREL)*, Jan. 2019, pp. 4891–4898. DOI: 10.3850/978-981-11-2724-3_0107-cd.

[95] S. Hurst, "SIL Hardware Verification of the CROME System Prototype Q according to the IEC 61508 Standard", CERN, Tech. Rep., 2016.

[96] W. McKinney, "Data Structures for Statistical Computing in Python", in *Python in Science Conference*, 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[97] K. F. Weaver *et al.*, *An Introduction to Statistical Analysis in Research*. Hoboken, NJ, USA: John Wiley & Sons, Inc., Jul. 2017. DOI: 10.1002/9781119454205.

[98] J. Myers, A. Well, and R. Lorch, *Research Design and Statistical Analysis, Third Edition*. Jan. 2010.

[99] N. Mukhopadhyay, "Correlation Coefficient", in *International Encyclopedia of Statistical Science*, M. Lovric, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 315–318. DOI: 10.1007/978-3-642-04898-2_194.

[100] scikit-learn, *sklearn.ensemble.GradientBoostingClassifier - scikit-learn 0.24.2 documentation*, 2020. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html.

[101] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, Jan. 2011.

[102] M. Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems", Mar. 2016.

[103] M. N. Halgamuge, E. Daminda, and A. Nirmalathas, "Best optimizer selection for predicting bushfire occurrences using deep learning", *Natural Hazards*, vol. 103, no. 1, pp. 845–860, Aug. 2020. DOI: 10.1007/s11069-020-04015-7.

[104] F. Chollet *et al.*, *Model training APIs - Keras API reference*, 2021. [Online]. Available: https://keras.io/api/models/model_training_apis/.

Appendix A

Calculation of Basic Statistical Features

For a signal with discrete values X_i and $i = 1, .., N$, the mean is the average of the discrete values [21].

$$mean(X) = \mu(X) = \frac{1}{N} \sum_{i=1}^{N} X_i \tag{A.1}$$

The median, however, is the "middle value of an ordered set of data" [24, p. 7163]. The variance measures the variability of a dataset and is calculated as shown in Equation A.2 according to [39].

$$var(X) = \frac{1}{N} \sum_{j=1}^{N} (X_i - \mu(X))^2 \tag{A.2}$$

The standard deviation can be calculated by taking the square root of the variance [21, 39].

$$std(X) = \sigma(X) = \sqrt{var(X)} = \sqrt{\frac{1}{N} \sum_{j=1}^{N} (X_i - \mu(X))^2} \tag{A.3}$$

Appendix B

Internal Measurement Variables of CROME

Variable Name	Description	Type	Unit
TimeStamp	Timestamp assigned to measurement	Numeric	Unix [ms]
MSID	Device ID	String	-
RawValue	Measured value in non-radiological units as provided by readout circuit	Numeric	-
Temperature2	Internal device temperature	Numeric	°C
CaseTemp	Temperature of CROME housing	Numeric	°C
Minus5va	Internal analog voltage	Numeric	V
Minus15va	Internal analog voltage	Numeric	V
Minus12va	Internal analog voltage	Numeric	V
Minus2v5a	Internal analog voltage	Numeric	V
Plus15va	Internal analog voltage	Numeric	V
Plus5vZynq	Internal voltage	Numeric	V
Plus3v3d	Internal digital voltage	Numeric	V
Plus5vd	Internal digital voltage	Numeric	V
Plus12vHV	Internal voltage	Numeric	V
Plus12va	Internal analog voltage	Numeric	V
Plus5va	Internal analog voltage	Numeric	V
VrefHV	Reference voltage	Numeric	V
VrefComp	Reference voltage	Numeric	V
VrefADCMeans	Reference voltage	Numeric	V
LowVoltagesFaults	Status of low voltages	Boolean	-
SensorConnection	Status of sensor connection	Boolean	-
PowerSupply	Status of power supply	Boolean	-
Battery	Status of the battery	Boolean	-
Charger	Status of the charger unit	Boolean	-
Input1	Status of Input1	Boolean	-
Input2	Status of Input2	Boolean	-
Input3	Status of Input3	Boolean	-
Input4	Status of Input4	Boolean	-
Output1	Status of Output1	Boolean	-
Output2	Status of Output2	Boolean	-
Output3	Status of Output3	Boolean	-
Output4	Status of Output4	Boolean	-

Table B.1: List of all available variables of internal datasets according to [92]

Appendix C

Model Evaluation of Signal Classification Task

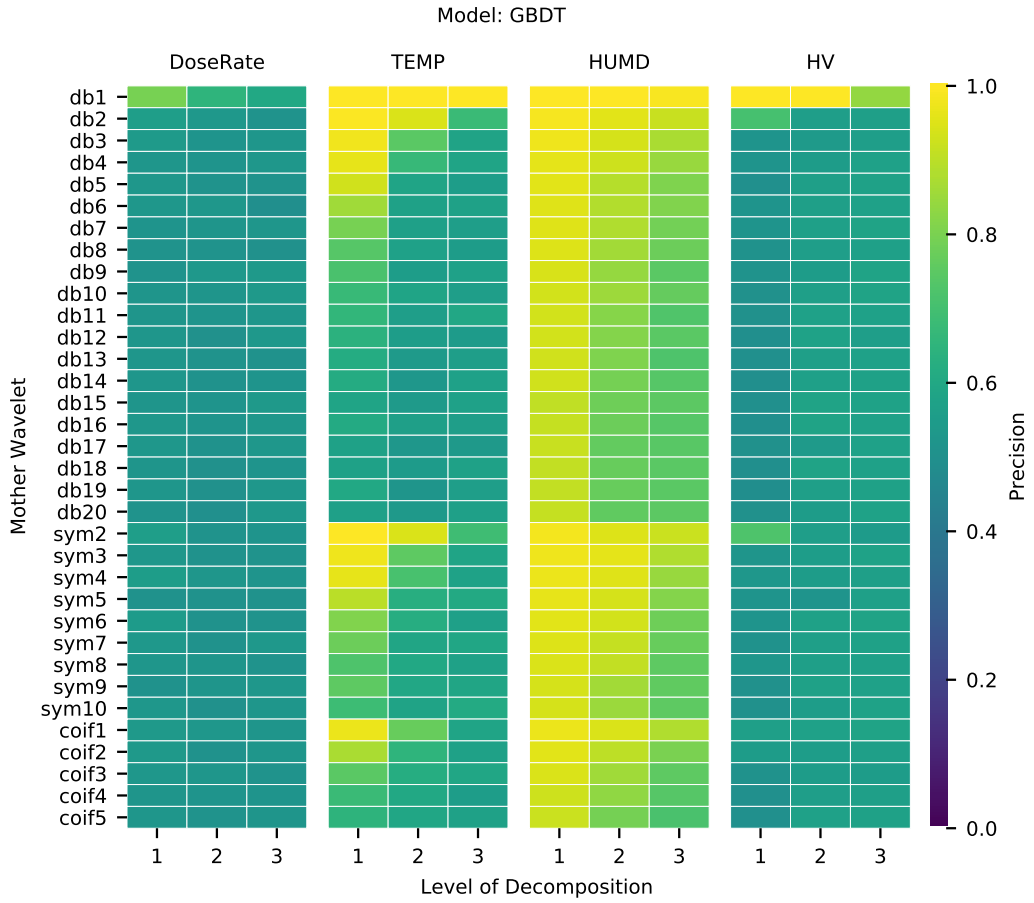


Figure C.1: Model precision of signal classification (GBDT algorithm) for all variables

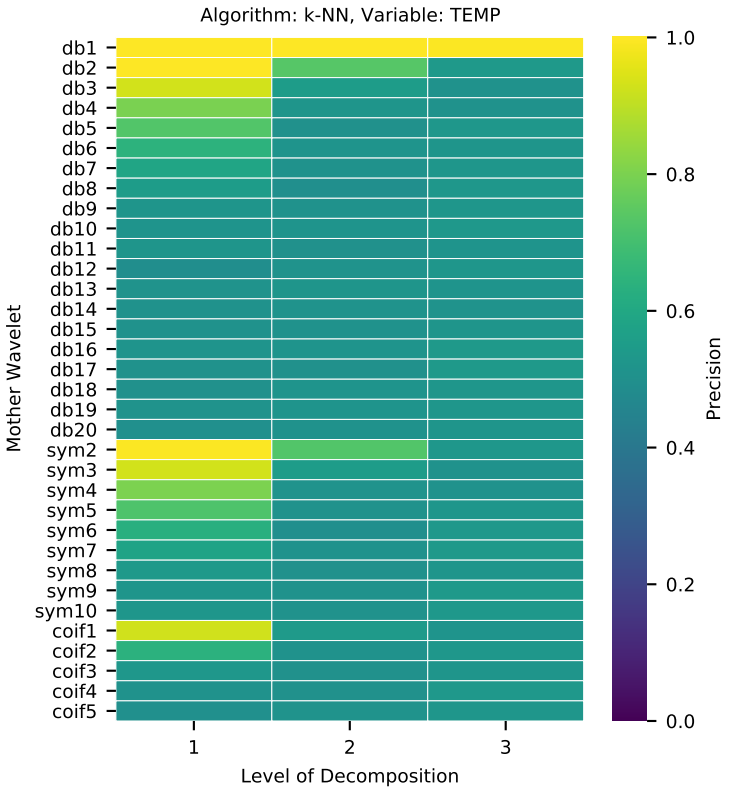


Figure C.2: Model precision of signal classification (k-NN algorithm) for temperature data

Appendix D

Detailed Results of the LSTM-AE Model

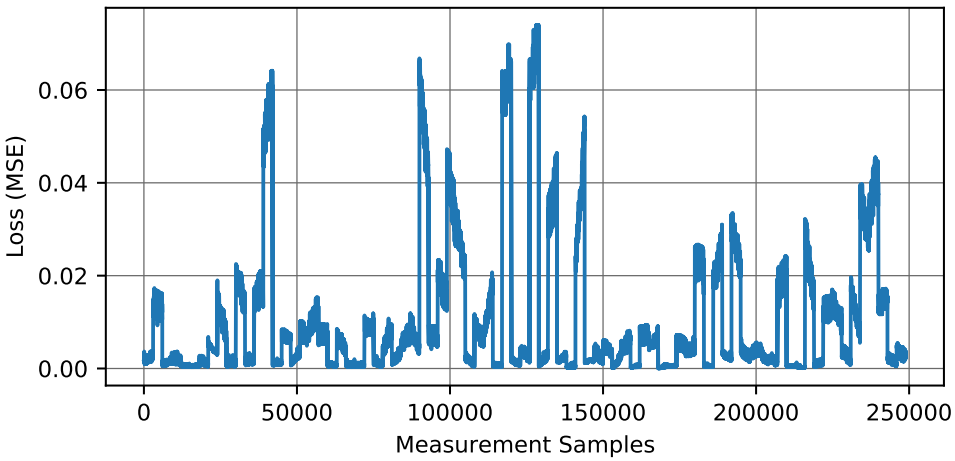


Figure D.1: Training loss over measurement samples LSTM-AE (RUN1)

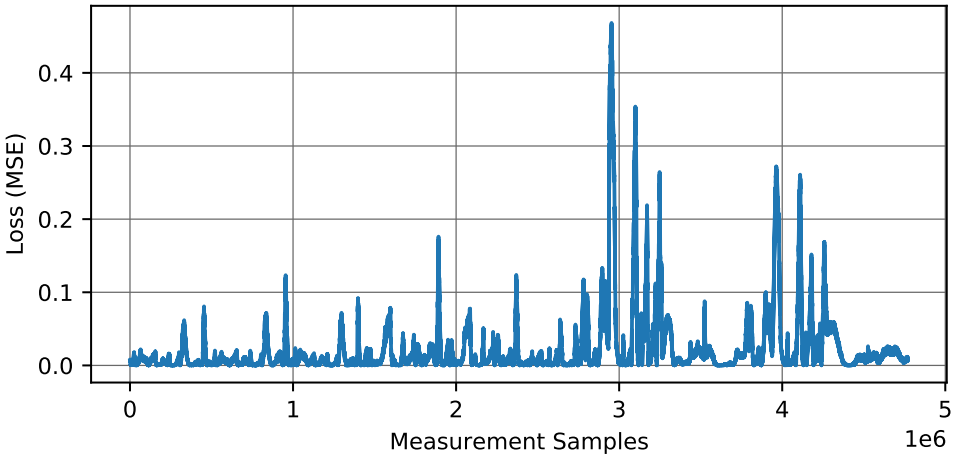


Figure D.2: Testing loss over measurement samples for LSTM-AE (RUN1)