TU WIEN Informatics

Anomaly Detection in Redundant Sensor Systems

A Comparative Study Using Photovoltaic Power Plant Data

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Data Science

eingereicht von

Alexander Stefitz, BSc
Matrikelnummer 11817192

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dr. Ivona Brandić
Mitwirkung: Projektass.(FWF) Alessandro Tundo, PhD

Wien, 21. Oktober 2024

Alexander Stefitz	Ivona Brandić

TU WIEN Informatics

Anomaly Detection in Redundant Sensor Systems

A Comparative Study Using Photovoltaic Power Plant Data

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Data Science

by

Alexander Stefitz, BSc
Registration Number 11817192

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dr. Ivona Brandić
Assistance: Projektass.(FWF) Alessandro Tundo, PhD

Vienna, October 21, 2024

Alexander Stefitz	Ivona Brandić

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Alexander Stefitz, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 21. Oktober 2024

Alexander Stefitz

Acknowledgements

I would like to express my sincere thanks to my supervisors, Ivona Brandić and Alessandro Tundo. Your guidance and quick responses to my questions and concerns were crucial to the success of this thesis. My thanks also go to my colleagues at the supporting company, who made me feel welcome from the very first day of my internship. Special thanks go to my supervisor, Matthias, for always being there to answer my questions and provide valuable advice.

I am truly grateful to my parents, whose unconditional support, in every possible way, made this possible. I would not be where I am today without you. My heartfelt thanks also go to my brother, Sebastian, who was always there to listen to my concerns and needs.

Finally, I want to thank all my friends who I was lucky to get to know at many different points in my life, and my girlfriend Sophie, for always being on my side and giving me strength when things were not going as expected. Special thanks to those who, in the course of my studies at TU Wien, have turned from colleagues into true friends. Without you, this journey would have been much harder - and a lot less fun!

Alexander Stefitz

Kurzfassung

In den letzten Jahren gab es weltweit erhebliche Fortschritte beim Ausbau der Energieerzeugung durch Photovoltaik (PV), und dieser Trend wird voraussichtlich auch in Zukunft anhalten. Um den Betriebszustand und die Effizienz eines großen PV-Kraftwerks zu überwachen, wird die Sonneneinstrahlung mithilfe redundanter Sensoren an verschiedenen Standorten am Kraftwerksgelände gemessen. Diese Daten werden genutzt, um festzustellen, ob die tatsächliche Leistung des Kraftwerks den Erwartungen entspricht oder ob Probleme vorliegen, welche die Leistung beeinträchtigen. Allerdings können auch die Messgeräte selbst fehlerhaft sein, was zu widersprüchlichen oder uneindeutigen Daten führt, bei denen unklar ist, welche Messwerte der Realität entsprechen.

An diesem Punkt setzt diese Arbeit an. Es werden zwei Methoden zur Validierung und Fehlererkennung in den Messdaten der Sensoren verglichen. Grundlage der Analyse sind die Messdaten von zwei großen PV-Kraftwerken in Europa, die im Zeitraum von Jänner 2023 bis Juli 2024 erhoben wurden. Da die Anzahl der fehlerhaften Messdaten nicht ausreichend war, wurde ein Framework mit künstlich erzeugten Anomalien eingesetzt, die auf real auftretenden Problemen basieren, um die Methoden zu evaluieren.

Mit PRADA wird eine neue Methode zur Fehlererkennung in redundanten Sensorsystemen vorgestellt. Dabei wird eine Regression zwischen jeweils zwei Sensoren durchgeführt und anhand des Regressionskoeffizienten das Verhalten der Sensoren bewertet. Dieses Verhalten wird über mehrere Tage hinweg beobachtet, um festzustellen, ob ein Sensor fehlerhaft ist. PRADA zeichnet sich durch Flexibilität, Transparenz und geringe Anforderungen an die Hardware aus. Als Vergleichsverfahren dient ein LSTM-Autoencoder, eine etablierte Methode zur Anomalieerkennung.

In verschiedenen Experimenten erzielt PRADA konstant hohe Erkennungsraten mit einem F1-Score zwischen 0.94 und 0.975. Damit übertrifft PRADA in fünf von sechs Experimenten das Ergebnis des LSTM-Autoencoders, welcher mit einem F1-Score zwischen 0.84 und 0.97 weniger konsistente Ergebnisse liefert. Darüber hinaus kann PRADA deutlich schneller trainiert werden. Bei der Anwendung befinden sich die Laufzeiten beider Methoden in der gleichen Größenordnung, wenn für den LSTM-Autoencoder eine GPU verwendet wird, ohne GPU zeigt PRADA signifikant kürzere Laufzeiten.

Abstract

In recent years, there has been significant progress globally in the expansion of photovoltaic (PV) energy generation, and this trend is expected to continue in the future. To monitor the operational status and efficiency of an utility-scale PV power plant, solar irradiance is measured using redundant sensors located at different locations on the site of the plant. This data is used to determine whether the electric power of the plant meets the expectations or if there are issues affecting its performance. However, the sensors themselves can sometimes malfunction, leading to inconsistent or ambiguous data where it is unclear which measurements reflect reality.

This is where this work comes in. It compares two methods for validating and detecting anomalies in the sensor data. The analysis is based on irradiance data collected from two utility-scale PV power plants in Europe between January 2023 and July 2024. Since the number of occurrences of anomalies in the data was insufficient, a framework with artificially generated anomalies, based on real-world anomalies, was used for evaluation.

With PRADA, a new method for anomaly detection in redundant sensor systems is introduced. It performs regressions between pairs of sensors, and based on the regression coefficient, evaluates the behaviour of the sensors. This behaviour is monitored over several days to determine whether a sensor is faulty. PRADA is characterized by its flexibility, transparency, and low hardware requirements. An LSTM Autoencoder is used as the comparison method, which is an established method for anomaly detection.

In various experiments, PRADA consistently achieves high detection rates with an F1 score between 0.94 and 0.975. PRADA outperforms the LSTM Autoencoder in five out of six experiments, where the LSTM Autoencoder produces less consistent results, with an F1 score ranging from 0.84 to 0.97. Furthermore, PRADA can be trained significantly faster than the LSTM Autoencoder. During execution, the runtimes of both methods are comparable when a GPU is used for the LSTM Autoencoder. Without a GPU, PRADA shows significantly shorter runtimes.

Contents

CHAPTER 1

Introduction

In recent years, there has been considerable growth in the global expansion of electricity generation through photovoltaics. Between 1998 and 2015, the global installed capacity increased by an average of 38% per year [1]. This trend has accelerated even further in recent years, with Austria seeing a 69% increase in total installed capacity from 2022 to 2023 [2]. It is expected that the global installed capacity will exceed 2 terawatts by 2024, even though the 1 terawatt mark was only exceeded in 2022 [3]. This significant increase can likely be attributed to two reasons:

On one hand, many industrialised nations worldwide have set ambitious goals to rely more on renewable energy sources to reduce greenhouse gas emissions and combat climate change. In some cases, there are even binding obligations or agreements, such as the Paris Climate Agreement, which aims to limit the increase in the global average temperature to well below 2°C. In this context, great hopes are placed on wind power and photovoltaics, since these have more growth potential than other energy sources and are also cheaper and faster to build [4]. There are also high growth rates and great potential for photovoltaics in developing countries, where population growth, urbanisation and industrialisation continue to drive a rapid increase in energy demand. This increase can be quickly and easily satisfied with photovoltaics, especially since many of these countries are located in geographical locations with a lot of potential for the use of solar energy [5].

The second crucial reason for this growth is the price drop of photovoltaic panels and other required components such as inverters. The average cost of a PV power plant, for example, has fallen from \$5.12/Wp in 2010 to \$0.88/Wp in 2022, both based on 2022 purchasing power [4]. In the years prior to 2010, prices were even higher. In Germany, prices for rooftop-systems dropped by 90% from 1990 to 2023 [6]. In 2023, the International Energy Agency stated that, under some circumstances, photovoltaic is the cheapest source of electricity in history [7].

The energy crisis in the years 2021-2023 has further accelerated this trend: As a result, the price of electricity also increased considerably [8], which has significantly reduced the payback period of renewable energy sources and thus motivated investment in these forms of energy.

However, as the number of PV power plants continues to grow, so does the need to maintain them. For the power plant operator, this is essential to avoid a loss of income. However, it also serves the general public, which benefits from a more stable power grid when power plants do not experience failures.

The performance of photovoltaic power plants depends on the incoming irradiance, which is converted into electricity by the photovoltaic cells. Compared to other power plants, especially fossil fuel power plants, PV power plants are relatively simple in design. In addition, PV power plants scale not by increasing the size of individual components, as it is common with other power plants, but by increasing the number of components [9]. Therefore, larger power plants typically use the same components, only in greater quantities.

However, the ability of PV power plants to independently detect irregularities in operation or to assess their own efficiency is limited. To do so, it is required to know environmental variables, primarily irradiance. This knowledge can then be used to assess whether the electricity production fits the given circumstances [10]. If the production is lower than expected, this is a sign that some parts of the power plant are not working correctly.

Therefore, it is common to install sensors at different locations on site which measure the irradiance and other environmental variables [11]. The number of sensors installed depends on the size of the power plant. On the one hand, it is desirable to install as few sensors as possible to save costs. On the other hand, multiple sensors are necessary to be better protected against failures and to take different conditions within the power plant (e.g. different orientations of the panels, shading) into account. In any case, the expectation is that all sensors having the same orientation also measure the same values. This is called redundant sensor setup. Furthermore, there are standards that recommend a certain number of sensors depending on the size of the power plant [12].

However, this leads to a new problem: In order to use the measurement data of the environmental variables to monitor the state of the power plant, these must be considered to be correct. Otherwise, it is not clear whether a possible discrepancy between the measurement data and the power produced occurs due to a problem with the power plant or due to incorrect measurement data. Furthermore, it is possible that the redundant sensor setup produces ambiguous values and it is not clear which of the measurements can be trusted.

This is where this work steps in: It aims to develop and evaluate methods that verify and validate the measurement data of the environmental variables by performing anomaly detection so that their correctness can be assumed for further monitoring of the entire power plant. Irradiance measurements are used as a use case within this work.

Two methods have been explored in this thesis: PRADA is a novel method, an LSTM Autoencoder serves as comparison method. PRADA focuses very strongly on the redundancy of the sensors, implying that the measurements should be very similar. This is verified by performing a regression between the measurements of each pair of sensors. It also checks whether the behaviour of sensors has changed over time. PRADA is also characterised by transparency and traceability as well as low hardware requirements. The LSTM Autoencoder combines an Autoencoder, a proven method in the field of anomaly detection in various applications [13, 14, 15, 16], with an LSTM network, a deep learning model widely used for time series data [17]. It tries to learn the correct behaviour of data by encoding it and then reconstructing it again. The size of the reconstruction error can then be used to assess whether the data shows the usual behaviour or contains an anomaly. In contrast to PRADA, it focuses less on the direct comparison of two sensors, and more on the inspection of the general behaviour of the data. As a deep learning model, the LSTM Autoencoder is a black box model and it can be difficult to understand the behaviour in detail. Furthermore, the hardware requirements are usually higher.

The two proposed methods are evaluated using real-world data from two utility-scale power plants in Europe. They both have a redundant sensor setup for measuring irradiance, but differ in the number and orientation of sensors. Due to a scarcity of anomalies in the real-world data, a framework of artificial anomalies is introduced in this thesis. These artificial anomalies are based on the anomalies found in the real-world data, as well as on reports from experts and literature [18, 19, 20]. They enable a significantly more accurate evaluation compared to using only the few anomalies present in the data.

Generally, the evaluation is performed using the F1 score, but also other factors are taken into account: Undetected anomalies are examined more closely to determine weaknesses of the methods, and the runtimes for training and applying the methods are assessed. Finally, an experiment with Transfer Learning is carried out to determine whether it is necessary to operate a separate model for each power plant. Various experiments have shown that the novel approach PRADA is superior for the given problem: In five out of six situations, it was able to achieve a higher F1 score than the LSTM Autoencoder. Furthermore, it proved to be more easily applicable. However, both methods have different strengths and weaknesses and the exact situation must be taken into account. Additionally, PRADA proved to be more stable against changes in hyperparameters, as well as when applied to data from a new, previously unknown location.

This thesis is structured as follows: Chapter 2 presents and discusses the background of this research, and also introduces the considered power plants and artificial anomalies. This is followed by Chapter 3, where relevant and related literature is presented. Chapter 4 and Chapter 5 present the considered approaches and all parts of their workflow. Chapter 6 presents the setup of the comparative study, which includes a descriptive analysis of the data and its preprocessing steps, as well as the exact implementation of the artificial anomalies. The results and relevant findings are presented in Chapter 7, where Section 7.6 provides a short summary of all numerical results. Chapter 8 concludes the work and provides an outlook on further research that can be conducted based on this thesis.

CHAPTER 2

Background and Motivation

Photovoltaics refers to the conversion of light energy in the form of sunlight into electrical energy. This is achieved by using semiconductor materials that exhibit the photovoltaic effect: Light irradiation releases electrons, creating an electrical current. The photovoltaic cells used today are usually silicon-based, but materials research is constantly working on other promising approaches that could lead to lower prices and greater efficiency [21].

2.1 Large-Scale Photovoltaic Power Plants

When it comes to generating electricity using photovoltaics, a distinction is made between distributed solar and utility-scale solar [9, 10]:

- **Distributed Solar** refers to installations on the roofs of single-family homes, apartment buildings, commercial buildings or factory halls. This category also includes implementations such as parking lot canopies. This means that they are often a desirable additional use or source of income alongside the main purpose of the property. As a result, power plants of this type are very space-saving and prevent the creation of further impervious surfaces.

  However, the growth of distributed solar is limited because there are only a limited number of suitable buildings, and not all theoretically available sites are economically viable. The size of individual plants is therefore limited, so that the output is usually between a few kWp and a few MWp. The p in kWp and MWp stands for peak and refers to the nominal output that the plant can achieve under standard test conditions. In practice, the actual output is almost always lower.

- **Utility-Scale Solar** refers to systems that are installed in open areas such as fields, deserts or other large areas of land. Sometimes they can be combined with

agriculture (e.g. animal farming) but usually they are characterised by power generation as their main use.

Since there is no limitation due to roof space or similar, these can be designed to be significantly larger than distributed solar systems. Even small power plants of this type usually have an output of several MWp, and power plants with an output of more than 1 GWp have been in existence for several years [22]. In 2024, the currently largest plant, with a capacity of 3.5 GWp, was put into operation in China [23].

In the US, it is estimated that in 2022 63% of the installed capacity was provided by utility-scale solar, in this case defined as having more than 5 MWp and ground-mounted modules [24]. In the future, this is expected to shift further in favour of utility-scale solar, with an expected share of 73% by 2033.

Globally, however, the influence of distributed solar should not be underestimated: The share of utility-scale solar in 2022 was 57%, which was the lowest value since 2012. This is certainly also due to the many subsidies that governments have offered around this time [25]. Given those numbers, it can be assumed that both types of photovoltaic power plants will be relevant in the future.

Photovoltaic power plants are very simple compared to other types of power generation, especially compared to fossil fuel power plants, and consist of relatively few different components [26]. Figure 2.1 gives an overview about those: The lowest level shows single panels, while the top represents the whole power plant.

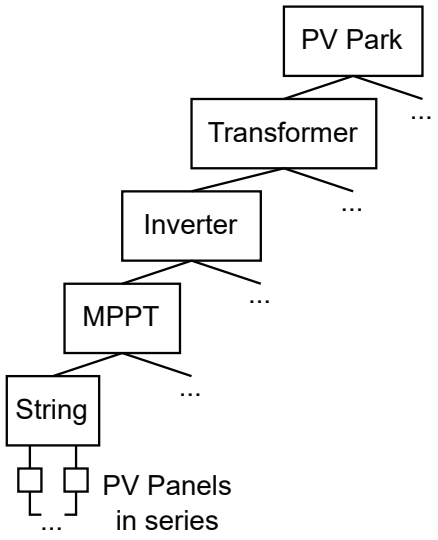

Figure 2.1: Typical structure of a PV power plant

Figure 2.1 should be read from bottom to the top: A low double-digit number of PV panels are connected in series and form a string. One or more of these are connected

to an MPPT, which stands for Maximum Power Point Tracker. This device optimises voltage and current so that the strings are operated at the Maximum Power Point (MPP), i.e. the point with the highest power.

One or more MPPTs are then connected to an inverter, which converts the direct current (DC) generated by the panels into alternating current (AC). However, the voltage is usually still too low to be fed into the grid, which is why a transformer is needed to bring the voltage up to the required level.

Even though different power plants can vary greatly in terms of power and size, the structure is more or less always the same. Often, especially for small power plants, it is common for certain components to be combined, for example MPPT and inverter [26]. Furthermore, it is possible to omit the transformer if the electricity is not fed into the high-voltage grid but into the normal household grid.

It is also important to emphasise that photovoltaic power plants do not scale by increasing the size or capacity of individual components, but rather by increasing their quantity. A larger power plant will therefore have more panels, more strings, and more MPPTs, but typically uses the same types of components found in smaller systems.

2.2 Monitoring of Photovoltaic Power Plants

The operator of a photovoltaic power plant not only has to build it, but is also responsible for its maintenance. Compared to other power plants, PV power plants are relatively low-maintenance [9]. Even large power plants are usually not staffed. Still, it may happen that parts of the power plant do not operate correctly, especially after a few years of operation. In such cases, the power plant must be inspected on site and parts of it may need to be replaced. However, because the power plants are often located in remote areas, as much monitoring as possible should be done remotely and an employee should only travel to the power plant if there is a confirmed problem [26].

It is therefore common for larger power plants to continuously record and collect operating data that can be used to draw conclusions about the condition of the power plant [11]. There are recommendations about how this monitoring process is implemented [12]. However, the exact implementation of this is arbitrary and varies depending on the operator, size and time of construction.

For some components such as MPPT, inverters or transformers, monitoring procedures based on operational values are available which allows the detection of failures such as electronic defects or overheating parts without additional data or measurements: [27] provides an exhaustive review about different possibilities. The components of some manufacturers are capable of detecting certain problems on their own [28].

However, there is still a fundamental problem that cannot be detected in this way: Without external information, a power plant can not know whether the power produced corresponds to the meteorological conditions on site: The power plant is therefore not

able to evaluate its current efficiency and it is therefore theoretically possible that on a clear day in the middle of summer, the power plant only produces half of its theoretically possible output, but is considered fully functional because it is not aware of this deviation from the target value and all the observed operating parameters are within the normal range.

Therefore, in order to be able to make a statement about the efficiency of the power plant, environmental variables must be measured. Usually, a selection of different variables is recorded [27], but irradiance is the most important one. These measurements can then be used to calculate the theoretically achievable power output. If this deviates significantly from the actual production, it is likely to indicate a problem.

2.3 Measuring Irradiance

Irradiance can be defined as the amount of radiant flux incident on a given surface area, either real or imaginary, and is typically expressed in units of $\frac{W}{m^2}$ [29]. Different devices can be used to measure irradiance, the two most known ones being pyranometers (shown in Figure 2.2) and silicon sensors. The first ones are considered to be more accurate, but also having downsides like higher costs and longer response times [30].

Figure 2.2: Pyranometer of type Hukseflux SR20. Image credit: [31]

Irradiance can be measured not only by different types of devices, but also in different ways [32], including:

- **GHI (Global Horizontal Irradiance):** Total solar irradiance on a horizontal surface, including diffused light.
- **DNI (Direct Normal Irradiance):** Solar irradiance directly from the sun on a perpendicular surface.
- **DHI (Diffuse Horizontal Irradiance):** Scattered solar irradiance from the sky on a horizontal surface.
- **POA (Plane of Array):** Solar irradiance on a tilted surface, including reflected and shadowed components.

Due to the different measurement techniques, the measured values can differ significantly, but some of the variables are composed of others and there are methods to convert measurements between them [33].

For photovoltaic power plants, the Plane of Array Irradiance (at the same angle as the PV panels) is the most relevant, as this measures exactly the irradiance that reaches the panels. On the right side of Figure 2.3 one can see a silicon sensor that was installed in Plane of Array (POA).

Figure 2.3: Photovoltaic panels and a silicon sensor installed in the Plane of Array (POA). Image credit: [34]

Utility-scale power plants typically feature a redundant sensor setup: This means that several irradiance sensors are installed at various locations on the site. For each orientation, there should be at least two sensors which are expected to report the same values, hence the term *redundant*. Which irradiance sensors are installed and where they are positioned on site must be considered during the planning of the power plant, as this is crucial for reliable operation. Furthermore, these measuring instruments also regularly show defects, which is why checking and maintaining the sensors is essential [35]. There are also different norms and recommendations regarding the number of sensors and their types.

2.4 Problem Description

In the previous sections, it was pointed out that measurements of environmental variables, especially irradiance, are essential to determine the operating state and efficiency of a PV power plant. Therefore, sensors are usually installed at the plant site in a redundant setup to use this measurement data for monitoring.

However, this creates a new problem: How can certainty that the irradiance measurements are correct be achieved? Only then, correct conclusions about the state of the power plant can be made, because otherwise it is not known whether the cause of an anomaly is a problem with the power plant or with the irradiance measurement data. In the worst case, a problem of the power plant could be accompanied by false measurement data in such a way that none of them is detected. It is also possible, for example, that several sensors of the redundant setup measure ambiguous values and it is not clear which of them are correct.

This thesis aims to solve this problem: Methods which validate the measurements of the environmental variables and detect irregularities in them should be investigated and compared. The focus should be on data-driven approaches, i.e. judgements should be based on available historical and present data, without the use of physical models or external data. As key variable for photovoltaics, irradiance should be used for the experiments, but no assumptions should be taken to make methods work only with data of this variable. The methods under consideration should work in such a way that at the end of the day, they receive the measurement data from all sensors of the redundant setup, and then make a statement as to whether all sensors are operating properly. If this is not the case, the methods should indicate which sensors are anomalous.

To achieve this goal, this work presents the novel approach PRADA, which uses regression methods to evaluate the behaviour of the sensors. An LSTM Autoencoder serves as comparison method, which combines an Autoencoder, a proven method in the field of anomaly detection, with an LSTM network, which is a neural network suitable for time series data. For the comparative study, irradiance measurements of two power plants operated by an Austrian energy provider are available. These power plants are located in Austria and Spain and are both classified as utility-scale power plants. More relevant information about the two considered power plants can be seen in Table 2.1.

	Plant in Austria	**Plant in Spain**
Short name	`p0`	`p1`
Orientation	East (`o0`) and West (`o1`)	South (`o0`)
Irradiance sensors	10 (`o0_s0`-`o0_s4`, `o1_s0`-`o1_s4`)	21 (`s0`-`s20`)
Example sensor name	`p0_o1_s4`	`p1_o0_s17`

Table 2.1: Comparison of the photovoltaic power plants considered in this study

The data availability for `p0` ranges from January 2023 to July 2024, while the data for `p1` is only available from mid-April 2023. A descriptive data analysis is presented in Section 6.1. For all sensors, the measured variable is Plane of Array (POA) irradiance. The sensors are mainly silicon sensors, but at least one pyranometer per plant and orientation was installed as well. In this thesis, there is no differentiation between those two types of sensors, since this would be an additional assumption restricting the use of the approaches. Also in daily operation, they are usually not handled differently.

With the help of the real data from these two power plants, it is evaluated which of

the two approaches is better suitable for anomaly detection in the daily operation of a power plant operator. Primarily, the performance in detecting anomalies is considered. However, other factors are also included in the evaluation, for example, the simplicity of implementation, the required computational effort or the applicability to new power plants.

A major challenge of this work is a scarcity of true anomalies in the available data: The number of irregularities in the data is low, certainly also due to the fact that both power plants were only built a few years ago. The existing anomalies are not sufficient to carry out an objective comparative study of the two methods. Therefore, an alternative approach is taken, which resulted in the following methodology: After carefully analysing the available data and performing a descriptive data analysis, all known true anomalies are completely removed. The complete absence of anomalies can, of course, never be entirely ruled out, especially if they are so weak that they don't have a significant impact and remain undetected. However, for the purposes of this thesis, the assumption of anomaly-free data is applied from now on.

Afterwards, artificial anomalies are systematically added to the data. For this purpose, based on the real detected anomalies and reports from literature and domain experts, three different categories of errors were determined, which occurred or are plausible when measuring irradiance. Each added anomaly belongs to one of these three categories. However, a wide range of variation is also possible within the categories in order to be as realistic as possible. The three error types considered are presented below, together with the abbreviation used to refer to each type in the next chapters.

- **Measurement is constant (`const`):** Instead of delivering measurements representing the reality, a sensor just reports a constant value over a significant period of time. This happens regularly in daily operation, since there are various possibilities for this to happen: For example, a broken measuring element in an otherwise (from an electronic point of view) functioning sensor, so that no irradiance can be measured. Another possibility would be a mount breaking and therefore the sensor lying on the floor facing down, which prevents any light from reaching the sensor. However, it does not only happen that 0 W/m^2 is constantly measured, but domain experts also reported incidents in which sensors have constantly measured very high or very low values. In one case, this value was the largest integer that can be represented with a 32-bit register width ($2^{32} - 1 = 4,294,967,295$), which suggests that an error occurred in an internal verification or correction process. In another case, it was a very large negative value.
- **Measurement is close to reality, but distorted (`deter`):** The second type of artificial anomalies is deterioration, meaning that the sensor measures data that is correlated with the reality, but not completely representing it. A prime example of this would be a layer of dust or snow covering the sensor, which reduces the incoming irradiance, but does not block it completely (like for anomalies of type `const`).

However, there may be other, less temporary and easily explainable causes, such as electronic defects or a calibration that has failed or not been carried out. This behaviour also occurs in real world [19], as it has been observed by domain experts multiple times. Also one of the few anomalies present in the data used in this thesis was of this type, which is explained in more detail in Section 6.2.

When adding artificial errors in this work, a deterioration ratio between 0 and 1 is chosen, as well as one of three further behaviours:

1. **Constant deterioration:** For the duration of the anomaly, the measured value is a constant weakening of reality. On a sunny day with 1,000 W/m^2 of real irradiance, for example, only 500 W/m^2 are measured, and a few days later (cloudy) 200 instead of 400 W/m^2. However, the ratio always remains the same. An example of this would be a permanent error, such as a clouding of the glass on the measuring element or humidity in the sensor [18].
2. **Increasing deterioration:** In this case, the weakening does not remain constant, but becomes increasingly stronger. As in the previous case, the anomaly begins with a certain deterioration, but then becomes stronger over time until it reaches 100% and the sensor only measures a constant 0 W/m^2. An example of this would be a sensor next to construction work, where more and more dust is deposited on the sensor due to the sand being stirred up, until it no longer lets any light through. A similar situation was reported, where the degradation increased over the first days, but then appeared constant [20].
3. **Decreasing deterioration:** In this third case, the opposite occurs: It starts again with a certain weakening, but here it decreases until the sensor measures values that correspond to reality again. In this case, the anomaly also ends. This case occurs, for example, during heavy snowfall and a subsequent melt due to sunshine in the following days.

- **Measurement is random (`rand`):** The third type of anomaly considered in this thesis occurs when the sensor reports values that fluctuate but are not reflective of actual conditions. In practice, this can happen, for instance, when the connection (particularly in the case of analogue connections) is disrupted, resulting in the sensor transmitting only noise instead of meaningful data.

Artificial errors are added with a certain probability that is chosen in advance. Generally, the artificial errors are considered to be independent of each other. However, problems of the sensors usually have an underlying cause, thus the probability for the occurrence of a sensor experiencing an anomaly is generally increased if another anomaly is currently active. Therefore, in this implementation, it is also more likely that a sensor fails if another one already has an active anomaly. The exact implementation of this, as well as all available and selected parameters, are described in detail in Section 6.3. Examples of all the artificial anomalies considered are also provided there in the form of figures and tables.

It is possible, in theory, that additional types of errors exist beyond the three categories introduced so far. However, these categories capture all the real errors identified in the available data and relevant literature. Therefore, it can be assumed that they provide a sufficiently accurate representation of the real situation. This concludes the background and objectives of this thesis, allowing for the discussion of the considered approaches and their implementation in the following chapters.

CHAPTER 3

Related Work

This chapter presents relevant literature that is thematically linked to this thesis. Anomaly detection has been a field of great interest for many years. Even though it is several years old, [36] still provides a valuable overview of this field. It addresses various challenges and also offers insights into different methods and their application. The rise of deep learning enabled significantly more complex use cases and also accelerated further research in this field: [37] introduces a taxonomy to differentiate them and also discusses future opportunities. The applications are highly diverse, ranging from medical problems [38] to data from the Internet of Things (IoT) [39]. Generally, a distinction is made between supervised and unsupervised problems. The latter often better reflect real-world scenarios but pose unique challenges for evaluation [40].

Another key application of anomaly detection is in time series, as it is the case in this work. Time series data appears across various domains and is often of special interest due to its complex, sequential nature. Significant research has been dedicated to this area in recent years: In [41], a comprehensive study was conducted to explore various methods and use cases. As for anomaly detection in general, recent developments have increasingly focused on deep learning-based approaches [42]. Also here, it is differentiated between supervised and unsupervised problems and methods, with [43] serving as an example for an unsupervised deep learning based method. However, statements about continuously improving methods must be viewed with caution in this case, since the benchmarks often used for this purpose are considered to be unreliable [44].

There have also been advancements in anomaly detection concerning sensor data. [45] provides an overview of this field. In this context, a distinction is made between model-driven (also referred to as conventional) and data-driven methods. The former address the problem from a physical perspective, by modelling the situation according to physical models or checking for violation of known physical limits. With data-driven methods, which are the focus of this work, the physical properties about the measured data move into the background and the data is validated by only using past data and experience,

or by comparing it with additional data. This might be other measurements of the same variable (e.g. second sensor at different location at the plant), as well as possibly correlated data of different variables (e.g. module temperature and irradiance).

The area of energy generation is also an important one: [46] introduces a data-driven condition monitoring and anomaly detection of a wind turbine, in [47] data of nuclear power plants is used. When considering PV power plants and irradiance data, some literature is available about validation using additional data: In [48], a validation of ground irradiance measurements only using satellite data is introduced, and extended in [49]. [50] summarises different data-related problems regarding PV power plants and attempts to predict or validate measurements using other available variables.

However, literature about irradiance validation primarily focused on GHI (Global Horizontal Irradiance), while for PV power plants POA (Plane of Array) irradiance (sensor tilted in the same angle as the solar panels) is relevant. Generally, the approach of only using the past time series of the same variable for anomaly detection, as considered in this work, has not gained a lot of attention in research in the context of PV power plants or irradiance yet.

LSTM networks are already being used in combination with irradiance data, including in connection with PV power plants. However, the focus here is not on detecting anomalies, but on predicting measurements [51, 52]. If such an LSTM network is combined with an autoencoder to an LSTM Autoencoder, it can be used for anomaly detection and has proven itself in many other domains: [53] gives an overview about different use cases of LSTM Autoencoders as an unsupervised anomaly detection method. In [54], it is used to detect anomalies in an automotive context, while [55] uses it with energy data. This method is also already being used in the field of transportation, for example with data from metro vehicles [56] or in monitoring safety in networks of cars, including self-driving ones [16]. [57] places a special focus on the application for multi-sensor time series, but uses a different network structure instead of LSTM.

As a novel approach, there is no literature directly providing comparison values or use cases of PRADA. However, it still relates to other, existing literature: [58] considers a similar situation, also related to power plants, but without pairwise comparison. In [59], quasi-redundant sensors are considered, so the expectation of identical measurement values is less strong than in the case of this thesis. However, it also considers a linear relationship. [60] describes a different initial situation, but shows another way in which linear regression can be used for anomaly detection.

Model-driven monitoring approaches have also been used to validate irradiance measurements. Those are usually based on Baseline Surface Radiation Network QC tests, which have been introduced in [61] and expanded in [62]. They are easy to execute, but show weaknesses because the intervals of accepted values are wide and irradiance as a weather-dependent variable always shows a lot of fluctuation. There have been attempts to narrow the intervals using statistical values [63] or other meteorological or environmental values [64, 65, 66], but the general problem of not detecting small measuring errors

persists, because slightly wrong measurements are still within the permitted interval.

Although not directly in the scope of this work, photovoltaic systems in general are of central importance and related to the experiments and data considered in this work. In recent years, a significant amount of literature has been published on this topic. [11], for example, provides a comprehensive overview of various photovoltaic systems and different monitoring aspects. [67] also provides a detailed overview and addresses possible faults and their sources. What regards the monitoring of a photovoltaic power plant as a whole, model-based approaches can provide reliable anomaly detection [68], if desired also on a very low level, such as string [69] or MPPT [70]. A good overview is provided by [71], where different models are compared. However, there have also been achievements with data-driven approaches: [72] shows how to utilise the data recorded in daily operation to detect anomalies on string-level. In [73] different machine learning models are used and compared to detect an anomaly of the whole plant.

CHAPTER 4

PRADA: Pairwise Regression-based Anomaly Detection Approach

PRADA is a novel method proposed in this work and the first of two approaches covered in this thesis to detect anomalies in the situation described in Chapter 2. As the acronym implies, the underlying idea is to use regression methods to detect anomalies in the data. The data from all sensors is not used all at once, i.e. no multiple regression is performed, but the various redundant sensors are compared in pairs. The method is based on the assumption that the measurements of any two sensors that are aligned in the same direction are equal or very similar, or mathematically expressed to be linearly dependent. If the measurements of one sensor are regressed against those of the other (always using regression without intercept), it is expected that the regression coefficient β is 1 or close to 1. If the result deviates from this, it may be a sign of an anomaly.

However, simply comparing the coefficient with 1 is not sufficient for several reasons: Firstly, it is possible that two sensors are aligned in the same way and both function properly, but still do not measure exactly the same. An example of this would be a sensor that is placed on the edge of the photovoltaic power plant next to a forest or building and therefore suffers from shading at certain times of the day and year, which then results in lower values. This would happen on several consecutive days and, if the comparison was strict, would be recognised immediately as an anomaly, even though in fact it is not. Another possibility would be if sensors of different types were used, which have different recalibration frequencies and at some point deviate from each other by a few percent or tenths of a percent.

Secondly, a strict decision based only on the coefficient would not allow any statement to be made as to which of the two sensors considered in the regression is faulty, because if

the two sensors were swapped in the regression (exchanging the explained and explaining variables), the result would simply be the inverse of the previously obtained regression coefficient and the proportional deviation from 1 would be equal.

The approach presented here, PRADA, attempts to solve all of the problems just discussed and thus to offer a robust solution for anomaly detection in redundant sensor systems. The core of the approach relies on well-tried and relatively non-computationally intensive methods, enabling a use on low-performance devices, possibly directly at the control unit of the power plant. Furthermore, every step and every decision in this approach is transparent and reproducible, so that the basis for a decision that is not directly comprehensible to the human observer can be found easily.

Compared to the previously mentioned simple check of the coefficient, improvements have been made at various points in PRADA. The coefficient is not simply checked for each day and for each sensor combination, but it is also monitored how the coefficient has developed over the last few days. If the deviation between the sensors in the past few days was as large as it is today, then this is less likely to be an anomaly than if the deviation has suddenly occurred. Of course, it is taken into account that not all real anomalies occur suddenly.

Although the use of regression might suggest it, PRADA does not include a learning process. In an optimisation process, the best hyperparameters for given data must be found, but for fixed hyperparameters, PRADA is applied directly to the data and provides a result without the need for prior training. This is one fundamental difference to the second approach presented in this work, LSTM Autoencoder, which is presented in Chapter 5 and includes a training process.

Furthermore, PRADA can be used to make a statement about which sensor is faulty by carefully tracking the comparisons, coefficients and their deviation from the target values using error points. A value is also provided for each detected error, which can be interpreted as error certainty. This is not used further in this work, but can be of great benefit in real-world applications and can be used to implement further improvements in the future.

Before going into detail on all aspects of PRADA and introducing the necessary theory, a brief summary of the procedure: For each pair of sensors with the same orientation and for each day, two regressions are performed. One with the values of the day under consideration, and the other with the data from a certain number of past days (in this thesis, a fixed duration between 3 days and 4 weeks). To keep track, accounts containing error points are set up for each of the sensors, which will then be used to assess the status of the sensors.

First, the coefficient of the day under consideration is checked, which is expected to be approximately 1. If it deviates significantly, both sensors in question are given an error point. In addition, the ratio of the coefficient to the coefficient of the previous days is considered. This is to check whether any difference between the sensors was already present in the last few days. This ratio is also expected to be 1 if the sensors

are functioning properly, meaning that the behaviour today is the same as in the days before. If this ratio deviates significantly, both sensors again receive an error point.

The error points received by the sensors are then divided by the theoretically achievable number of error points. This results in values between 0 and 1 for each sensor and day, with a high value indicating the existence of an error. If a certain limit value, which was previously defined or determined, is exceeded, the sensor is declared faulty for that day. One or more consecutive days of error then result in an anomaly. Depending on the selected hyperparameters, these must have a certain minimum length and an anomaly is not interrupted if a single day is recognised as not being faulty in the context of faulty days.

Now that the basic concept is clear, the details of the implementation will be discussed.

4.1 Regression Methods

This section presents the two regression methods which are used in this implementation of PRADA. In this case, these are the conventional OLS (Ordinary Least Square) regression and a robust regression method, which should reduce the influence of outliers.

4.1.1 Ordinary Least Squares (OLS) Regression

OLS is the regression method that is usually referred to when linear regression is spoken of. It attempts to establish a linear relationship between a dependent variable y and one or more independent variables $x_i,\ i = 1, \ldots, n,\ n \in \mathbb{N}$ by minimising the sum of squared residuals, which are the differences between observed and predicted values. The description of this method is based on [74].

Having n observations and p independent variables, the model can be written as

$$y = X\beta + \varepsilon$$

with the following variables:

- y is the vector representing the dependent variable, having shape $n \times 1$.
- X is the matrix of the independent variables. If the model contains an intercept, the first column only consists of the values 1 and X has shape $n \times (p + 1)$. Otherwise, the shape is $n \times p$.
- β is the vector of regression coefficients, one for each independent variable. OLS is used to obtain these values, which represent the linear relationship with the smallest sum of the squared residuals. The shape of β depends on whether the model contains an intercept, therefore it is either $p \times 1$ or $(p + 1) \times 1$.

- ε is the vector representing the residual for every observation, therefore it has the shape $n \times 1$.

For given X and y, β can then be estimated as

$$\hat{\beta}_{OLS} = \arg\min_{\beta} \sum_{i=1}^{n} (y_i - X_i\beta)^2 \tag{4.1}$$

where X_i represents the i-th row of X. Note that β represents the true, unknown parameter, while $\hat{\beta}$ represents the estimated parameter based on given data. This may seem like a hard optimisation problem to solve, but the optimal solution can be derived directly with the following matrix equation:

$$\hat{\beta}_{OLS} = (X^\top X)^{-1} X^\top y \tag{4.2}$$

To calculate $\hat{\beta}$ using Equation 4.2, it is necessary to calculate the inverse of the Gram matrix ($X^\top X$, shape $n \times n$). This requires a certain amount of computing power, but nowadays this is no longer a problem even with weak devices, so this is not a limitation.

Beside its simplicity, the OLS estimator has some statistical properties which can be useful in some cases. One of those is the fact that it is BLUE, which stands for Best Linear Unbiased Estimator. This means that, if some assumptions are fulfilled, it has the smallest variance among all linear unbiased estimators. Another property would be asymptotic normality, which makes the OLS estimates normally distributed when the sample size is growing. Also OLS is consistent, which means that the estimator converges to the true parameter for increasing sample size.

However, these properties do not apply in every case, but certain requirements must be met. These vary slightly depending on the literature, but the most important ones are shortly summarised below.

1. **Linearity:** The relationship between the dependent variable and the independent variables must be linear.
2. **Homoscedasticity:** The residuals must be normally distributed with mean 0 and constant variance σ^2.
3. **Independence of Residuals:** The error terms of each observation must be independent.
4. **No perfect multicollinearity:** No independent variable may be a linear combination of other independent variables.

If the assumptions are not fulfilled, the positive properties of the OLS estimator cannot be guaranteed. In the setting considered in this thesis, the Assumption 1 can be deemed to be fulfilled, as this is the case by definition if the sensors are functioning properly and also in many other cases. Assumption 4 is obsolete in the case considered here, as there is always only a single independent variable under consideration.

The fulfilment of Assumption 2 and Assumption 3 is less obvious. These, especially the independence of the residuals, are difficult to fulfil for data with a temporal relationship (time series), as is the case here, and therefore cannot be guaranteed in this case either.

However, this does not create too much of a problem in the scope of this work. If the assumptions are not met, the validity of the above-mentioned desirable properties of the OLS estimator cannot be guaranteed, but this does not invalidate the entire model.

If the model is used for predictions of unknown data points and the highest possible model quality is important, these properties are necessary and therefore great attention must be paid to the fulfilment of the conditions. However, none of these properties are required in this work, and the model is no longer used for further predictions once the coefficient has been determined. Instead, the regression methods are only used as a tool to determine a relationship between two data series once. Therefore, the fulfilment of the conditions is not as essential as in other settings and is thus assumed. In addition, a further regression method (robust regression) is investigated, which has weaker assumptions.

Connection to the Correlation

The problem statement and definition of PRADA raises the question whether this problem can also be solved with correlation methods instead of regression methods. Indeed, there is a close mathematical relationship between OLS regression and the Pearson correlation, which measures the linear correlation between data points. The Pearson correlation coefficient is defined as

$$\rho_{x,y} = \frac{\mathrm{Cov}(x,y)}{\sqrt{\mathrm{Var}(x) \cdot \mathrm{Var}(y)}},$$

where $\mathrm{Cov}(x,y)$ stands for the covariance between x and y, while $\mathrm{Var}(x)$ represents the variance of x. The OLS coefficient, when performing simple linear regression (only one independent variable), can be expressed as

$$\beta = \frac{\mathrm{Cov}(x,y)}{\mathrm{Var}(x)}.$$

This results in

$$\beta = \rho_{x,y} \cdot \sqrt{\frac{\mathrm{Var}(y)}{\mathrm{Var}(x)}}$$

being the relation between the regression and correlation coefficients. With the definition range $[-1, 1]$ and the relationship to regression just described, the correlation coefficient can certainly be used in a similar way. However, there is significantly less methodical diversity in calculating a correlation than there is in regression methods, where each method, even if it was usually developed for other or complex applications, can be used. Furthermore, the coefficient β with the domain $\mathbb{R}$ provides significantly more information than the regression coefficient. Therefore, regression methods are used for PRADA.

4.1.2 Robust Regression (M-Estimator)

While the OLS estimator can be calculated very easily and quickly, it also has some downsides, one of which is the vulnerability to outliers. This can be especially problematic in the use case from this thesis, where it even occurred in the data considered in this thesis (see Section 6.2, Figure 6.6) that there was a significant point outlier on an otherwise normal day. In this case, one single outlier could falsify the results of the whole day.

To counteract this, the second regression method considered is robust, meaning it is less sensitive to outliers and violations of assumptions, providing more reliable estimates in challenging data conditions.

The estimator presented now is also referred to as M-Estimator and the following description is based on [75]. It works by replacing the square function in Equation 4.1 with a generic loss function. This loss function can then be used to add different weights to the observations, for example giving the possibility to use the square for small residuals, but only the absolute values for large ones. This leads to the generic estimator

$$\hat{\beta}_L = \arg\min_{\beta} \sum_{i=1}^{n} L(y_i - X_i\beta), \tag{4.3}$$

where L is any loss function. There are countless different loss functions with different strengths and weaknesses. In this work, the choice fell on Huber's T-estimator, as it was both widespread and easily available in the libraries used. The idea is to combine the best features of OLS and LAD. LAD stands for Least Absolute Deviations and in this case, the absolute value is used instead of the square. For Huber's T-estimator, the loss function is given by

$$L(z) = \begin{cases} \frac{1}{2}z^2 & \text{if } |z| \leq t, \\ t(|z| - \frac{1}{2}t) & \text{if } |z| > t. \end{cases} \tag{4.4}$$

Using such a loss function leads to observation having different weights, which can be determined with Equation 4.5

$$W(z) = \begin{cases} 1 & \text{if } |z| \leq t, \\ \frac{t}{|z|} & \text{if } |z| > t. \end{cases} \tag{4.5}$$

Both functions have been depicted in Figure 4.1 to get a better idea how they look and what the effects are. In this case, $t = 1.345$ has been chosen as threshold. When looking at the loss function, one can clearly see that the function is quadratic around 0. For small residuals, the loss therefore corresponds to that of OLS. Above the threshold t (into the positive and negative), the loss function becomes linear, so the influence of such a large residual decreases in comparison to OLS.

The question may arise why Equation 4.4 is not simply the square function and the absolute value function, but instead these are also provided with factors and constant terms. This is necessary so that the loss function is continuous around the threshold t and the loss function remains consistent with other losses. However, as none of this is dependent on the residual, these terms have no influence on the behaviour.

This is also reflected in the weights function. For $z \leq |t|$ the weight is constantly 1, so all residuals have the same importance. However, the larger the residual becomes, the smaller the weight and thus the influence.

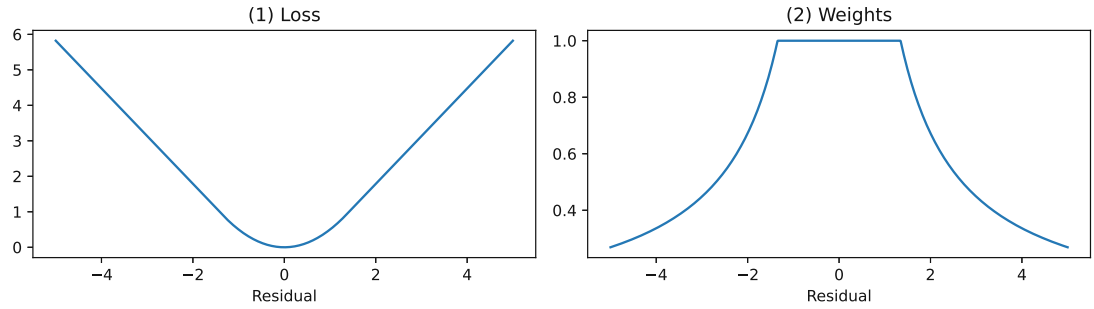

Figure 4.1: Huber's T loss and weight functions with $t = 1.345$

The optimal value for t depends on the data considered, or more precisely on the magnitude of the expected errors, and should therefore be chosen appropriately.

However, using robust regression is not without its drawbacks. While the OLS estimator can be calculated relatively easily, this is often more complicated with robust regression. This is usually solved using IRLS (Iteratively Reweighted Least Squares). With this method, the robust estimator $\hat{\beta}_L$ is determined by repeatedly weighting the residuals and then calculating the OLS estimator on them. This makes it necessary to calculate a large number of OLS estimators to determine a single robust estimator, which can cause problems on low-performance systems or take a relatively long time.

Until now, all definitions have referred to multiple regression, i.e. a dependent variable is explained by several independent variables, possibly also an intercept. However, this is not necessary in this case. As described above, the data is always considered in pairs and one sensor is arbitrarily chosen as the independent variable and the other as the dependent variable. This also does not require an intercept. Therefore, the univariate case without intercept is always used in the following, so $p = 1$ applies here. This means

that X and y have the same shape $(n \times 1)$, and with shape 1×1, β becomes a number from $\mathbb{R}$. This is also known as simple linear regression.

Although the focus of this work is on comparing PRADA with the LSTM Autoencoder, the decision was made at this point to use and evaluate two different regression methods (OLS and robust regression with Huber's T-Loss), as this highlights the flexibility of PRADA. There is no restriction to these two regression methods, and although it was not explicitly tested in the course of this work, it can be assumed that it will work with any other method, whether similar or very different, as long as it has a single coefficient as a result.

4.2 Calculating Error Probabilities

This section goes more into detail how PRADA uses the results of the regressions to decide whether an anomaly is taking place for a specific day and sensor. The first step is to determine the pairs to be considered. All sensors of one orientation are compared with each other, but sensors of different orientations are not in the current implementation. In this case, the regression is inverse symmetrical, so the coefficient of regressing sensor `s0` with sensor `s1` is the inverse of the coefficient when regressing `s1` with sensor `s0`. However, the percentage deviation of the coefficient from 1 is the same in both cases, so only one of the two comparisons needs to be made. If $n_{\text{orientations}}$ is the number of different orientations and n_i is the number of sensors of the orientation i, the total number of comparisons to be carried out is described by Equation 4.6:

$$n_{\text{pairs}} = \sum_{i=1}^{n_{\text{orientations}}} \binom{n_i}{2} \tag{4.6}$$

The number of comparisons to be made is therefore determined by the binomial coefficient and corresponds to the sum of the possible pairs of each orientation. It also applies that each sensor of orientation i is compared with $n_i - 1$ sensors.

For a given day and all sensors, it should now be checked whether anomalies are present. To do this, the pairs to be compared are first specified and noted. The regressions are then carried out: For each of the pairs, two regressions are performed, on the one hand the measurements of the day in question and on the other hand the measurements of the previous days, also known as *lookback days*. The hyperparameter n_{lookback} determines the number of days that are looked back into the past. It is also possible that different methods are used for the two regressions. This is discussed in more detail in Section 4.4.

The fixed number of days that are looked back into the past results in a *rolling window* so that comparisons are always made with the most recent past while the older past is discarded. However, there is also the option of using an expanding window. In this case, n_{lookback} only determines how many days should be used at the start of the data basis (e.g. if $n_{\text{lookback}} = 14$ and the start date is January 1st, the analysis can be started on

January 15th). However, all available data from the past is used for the regression of the days before. $\delta_{\text{expanding}}$ is used as hyperparameter defining whether the window is rolling or expanding.

Both variants have different strengths: The expanding window naturally has the advantage that more data and therefore more information is used, but the ever-increasing amount of data also increases the computing time for the regression. It is also possible that certain effects are drowned out by the long duration or that PRADA does not become accustomed to a certain trend in the behaviour. The rolling window with a fixed length has the advantage of a constant calculation duration, as exactly the same amount of data is always used. Also, it can get used to a trend better, as it is always compared with the days directly before. However, if a certain behaviour has already occurred exactly one year ago, for example, this information cannot be accessed. In this paper, both possibilities are analysed, more details are given in Section 4.4 and Section 4.5.

Regardless of the variant selected, the result for each pair is two coefficients: β_{day} and β_{base}, which describe the relationship of the pair on the considered day and on the previous days, respectively. This is then used to determine the relationship to the previous days:

$$\beta_{\text{ratio}} = \frac{\beta_{\text{day}}}{\beta_{\text{base}}} \tag{4.7}$$

The values β_{day} and β_{ratio} are now used further: The expectation is that both values are 1, which is why the proportional deviation from 1 is considered. The hyperparameters α_{day} and α_{ratio} define what the maximum permitted deviation of the coefficients is in order to be considered to be functioning normally. Using the function

$$f_{\text{thr}}(x, \alpha) = \begin{cases} 0 & \text{if } x \in \left(\frac{1}{1+\alpha}, 1+\alpha\right) \\ 1 & \text{otherwise,} \end{cases} \tag{4.8}$$

it can then be determined whether the deviation is acceptable (0 is returned) or not (1 is returned). The denominator in the lower interval limit was deliberately chosen due to the inverse symmetry of the regression.

Once all the calculations are available, the next step is to draw conclusions. To do this, an *error account* is set up for each sensor and $f_{\text{thr}}(\beta_{\text{day}}, \alpha_{\text{day}})$ and $f_{\text{thr}}(\beta_{\text{ratio}}, \alpha_{\text{ratio}})$ are calculated for all pairs.

For filling the error accounts, there are three possible cases for every pair:

- $f_{\text{thr}}(\beta_{\text{day}}, \alpha_{\text{day}}) = f_{\text{thr}}(\beta_{\text{ratio}}, \alpha_{\text{ratio}}) = 0$: In this case, everything seems to be in order and no error points are given.
- $f_{\text{thr}}(\beta_{\text{day}}, \alpha_{\text{day}}) = 1$, $f_{\text{thr}}(\beta_{\text{ratio}}, \alpha_{\text{ratio}}) = 0$ The regression of the considered day detected a problem, but the comparison with the days before did not notice unusual behaviour. Therefore only one error point is assigned to both sensors.

- $f_{\text{thr}}(\beta_{\text{day}}, \alpha_{\text{day}}) = 0$, $f_{\text{thr}}(\beta_{\text{ratio}}, \alpha_{\text{ratio}}) = 1$: The regression of the day under consideration gave normal results, so the two sensors measured approximately the same, but the behaviour changed compared to the lookback days. As before, one point is assigned to each of the two sensors.

- $f_{\text{thr}}(\beta_{\text{day}}, \alpha_{\text{day}}) = f_{\text{thr}}(\beta_{\text{ratio}}, \alpha_{\text{ratio}}) = 1$: Both regressions detected unusual behaviour and therefore both sensors receive two error points each.

This is repeated for all pairs. Each error account then has a value between 0 and $2(n_i - 1)$. The value is then divided by $2(n_i - 1)$, resulting in $[0, 1]$ as the definition range. This value in this interval is the final result for a sensor on a day and can be interpreted as the error probability.

Since both sensors of a pair always receive error points when a limit value is exceeded (it is not possible to determine which of the two sensors is affected), this naturally also means that sensors that are functioning correctly receive error points and therefore also have an error probability greater than 0.

However, this is not a problem, it is intentional and this is precisely how the affected sensor can ultimately be determined with a high degree of reliability: If a sensor is defective, this is noticeable in every or at least almost every pair, and this affected sensor gets error points with every comparison. However, a functioning sensor only scores points in the comparisons with the defective sensor. It does not receive any points in the comparisons with the other functioning sensors, so the points account remains relatively low and the probability of error is close to 0.

The question naturally arises if including β_{ratio} in PRADA is necessary or if the approach just presented with the error points and β_{day} alone is not sufficient. This was investigated in more detail, and in fact β_{day} is more indicative of the condition of the sensor, but looking at the previous days still adds value: It allows more error points to be scored, enabling a finer distinction (clearer separation between sensors with high and low scores) to be made. This also makes it easier to recognise the beginning and end of an anomaly, as at the end both coefficients usually do not change to the normal state at the same time, but first β_{day}, then β_{ratio}.

Almost all steps of PRADA have now been defined. What has been presented so far can be applied to all data in the past, up to the latest available day, and an error probability in the interval $[0, 1]$ is given for each day and for each sensor. Only the first n_{lookback} days are excluded from this, as there is not yet enough data for the calculation of β_{base} and thus also β_{ratio}. If an error probability is also necessarily required in these days, the regression can be carried out with less than n_{lookback} days. Only on the first day can the calculation not be performed.

There are various ways to implement PRADA. Which method is the best depends on the requirements, e.g. the available computing power and how important efficiency is. Section 6.6 provides details on the programming and execution of the implementation in

this work, while Algorithm 4.1 outlines how PRADA was conceptually implemented and successfully applied in the experiments.

Algorithm 4.1: PRADA: Efficient implementation

Input: Data matrix D with dimension $(n_{\text{days}} \cdot n_{\text{daily}}) \times n_{\text{sensors}}$,
Regression functions $R_{\text{day}}, R_{\text{base}}$

Parameters: α_{day}, α_{ratio} and n_{lookback}

Functions: $f_{\text{thr}}(x, \alpha) = \begin{cases} 0 & \text{if } x \in \left(\frac{1}{1+\alpha}, 1+\alpha\right) \\ 1 & \text{otherwise} \end{cases}$

Output: Matrix E with error probabilities
with dimension $(n_{\text{days}} - n_{\text{lookback}}) \times n_{\text{sensors}}$

1 $c = 0$ (dimension $n_{\text{sensors}} \times 1$)
2 **for** $d = (n_{\text{lookback}} + 1)$ **to** n_{days} **do**
3 $\quad \beta_{\text{day}} = \beta_{\text{base}} = \beta_{\text{ratio}} = 1$ (dimension $n_{\text{sensors}} \times n_{\text{sensors}}$)
4 $\quad v_{\text{day}} = D[d, :]$
5 $\quad v_{\text{base}} = D[(d - n_{\text{lookback}}) : (d-1), :]$
6 $\quad$ **for** $s_1 = 1$ **to** n_{sensors} **do**
7 $\quad\quad$ **for** $s_2 = (s_1 + 1)$ **to** n_{sensors} **do**
8 $\quad\quad\quad$ **if** orientation of s_1 and s_2 is different **then**
9 $\quad\quad\quad\quad$ **break for**
10 $\quad\quad\quad \beta_{\text{day}}[s1, s2] = R_{\text{day}}(v_{\text{day}}[s_1], v_{\text{day}}[s_2])$
11 $\quad\quad\quad \beta_{\text{base}}[s1, s2] = R_{\text{base}}(v_{\text{day}}[s_1], v_{\text{base}}[s_2])$
12 $\quad\quad\quad c[s1] = c[s1] + 2$
13 $\quad\quad\quad c[s2] = c[s2] + 2$
14 $\quad \beta_{\text{day}} = \beta_{\text{day}} + \beta_{\text{day}}^{\top}$
15 $\quad \beta_{\text{base}} = \beta_{\text{base}} + \beta_{\text{base}}^{\top}$
16 $\quad \beta_{\text{ratio}} = \beta_{\text{day}} / \beta_{\text{base}}$
17 $\quad E[d, :] = (\sum_{s=1}^{n_{\text{sensors}}} (f_{\text{thr}}(\beta_{\text{day}}, \alpha_{\text{day}}) + f_{\text{thr}}(\beta_{\text{ratio}}, \alpha_{\text{ratio}}))[:, s]) / c[s]$
18 **return** $E[n_{\text{lookback}} :, :]$

In the first lines of code (1-5) the coefficients and data is initialised. The variable c is a counter for the maximum number of error points that a sensor can receive. As mentioned earlier, this is $2(n_i - 1)$, where n_i is the number of sensors in alignment i. Then (6-9) the pairs are chosen. In this implementation, this works by looping over the sensors. For the first sensor (s_1), it is iterated over all sensors, while for the second (s_2) only the ones coming after s_1 are chosen. This covers all pairs of interest, while discarding the ones with different orientation or pairs that already occurred in different order.

In lines 10 and 11, the regression takes place and the coefficients are stored in a matrix. Due to the strategy, this results in an upper triangular matrix, with 1 on the diagonal. The counter is increased in lines 12 and 13. The number 2 is added because of the two executed regressions (behaviour today and behaviour today compared to the days before) and therefore 2 possible error points to be assigned.

This matrices already contain all required data, but it is added with the transposed version of itself in lines 14 and 15 in order to simplify further calculations: Now all information about one sensor can be accessed by selecting one row or column, not both. Also, due to the inverse symmetry it is possible to add the transposed version without inverting its entries, since the result of f_{thr} is the same. In line 16, β_{ratio} is calculated and in line 17, the error probabilities are calculated by dividing the error points by the count. At the end, the final result is a matrix containing error probabilities for each sensor and all days except the first n_{lookback}.

4.3 Defining Anomalies

The procedure described so far is already sufficient in many situations. In this work, however, the requirement is to obtain a list of detected anomalies (start, duration) in order to compare the performance of PRADA and the LSTM Autoencoder. This works according to the following simple logic:

A certain minimum length d_{min} for detected anomalies is defined. In this work, this is one, two or three days, but it can be set higher if required. Furthermore, a limit value α_{anomaly} is defined, above which a sensor is considered defective. This must naturally be in the interval $(0, 1)$, values above 0.6 were typical in preliminary experiments.

All sequences of days that exceed the limit value and fulfil the minimum length are then defined as an anomaly. An additional rule applies to make the procedure more robust: It is permitted within an anomaly for the error probability to be below the threshold value on one day if the limit value is exceeded again on the following day. This prevents an anomaly from being recognised as two separate anomalies due to a downward outlier.

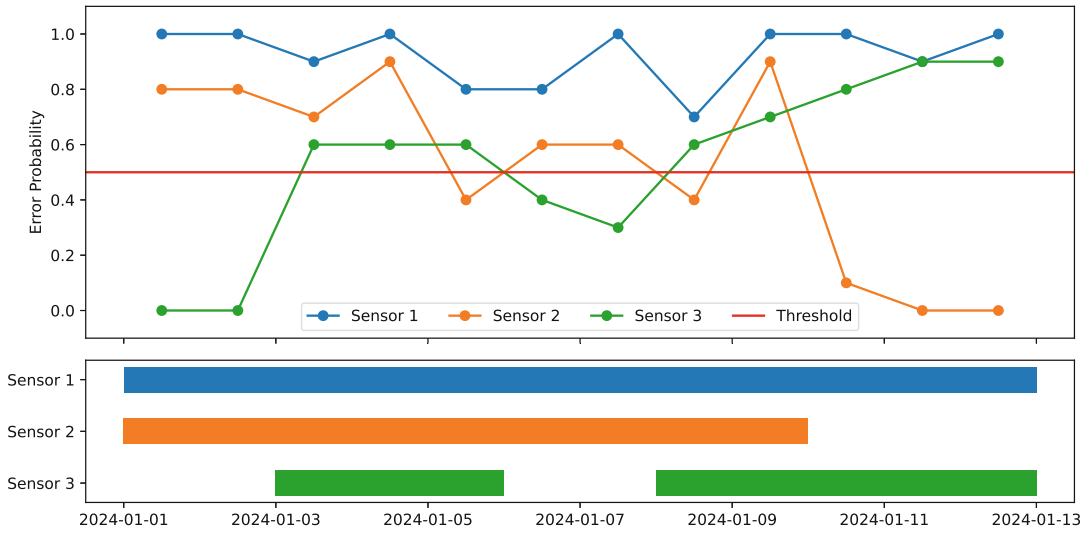

Figure 4.2: Example of different error probabilities and the resulting anomalies

Figure 4.2 gives examples how anomalies are defined given different error probabilities. The upper plot shows error probabilities over time for three sensors. The red line stands for the threshold α_{anomaly}. Sensor 1 has an error probability above the threshold for the whole observation period, so an anomaly is defined for the whole observation period. Sensor 2 shows more fluctuating behaviour: On two days, the error probability falls below the threshold. However, in both cases the threshold is exceeded again on the next day, which results in an continuous anomaly. This can not be said about Sensor 3: The error probability falls below the threshold for two consecutive days, resulting in two independent anomalies being defined.

Again, there are various options of implementation, the one of this work is visible in Algorithm 4.2. For reasons of simplicity and clarity, it is looped over the sensors and the days. If efficiency is the most important priority, it is also possible to solve this using vector operations (for example by adding shifted time series), but this has a negative impact on comprehensibility. In addition, no computationally complex calculations, only value checks, are carried out here, which keeps the influence of the loops low.

Algorithm 4.2: Defining anomalies from error probabilities

Input: Error probabilities matrix E with dimension $(n_{\text{days}} - n_{\text{lookback}}) \times n_{\text{sensors}}$
Parameters: Threshold α_{anomaly}, Minimum days d_{min}
Output: List of anomalies: start day, duration and average error probability

```
for s = 1 to n_sensors do
    count = 0
    for t = n_lookback to n_days do
        if E[t, s] > α_anomaly then
            if anomaly not active then
                begin anomaly at t
        else if E[t, s] ≤ α_anomaly then
            if E[t + 1, s] > α_anomaly then
                continue
            if anomaly is active then
                if duration of anomaly ≥ d_min then
                    end anomaly, save with start day, duration and average error
                     probability
                    count = count + 1
                else
                    end anomaly, discard
    if anomaly is active then
        if duration of anomaly ≥ d_min then
            end anomaly, save with start day, duration and average error
             probability
            count = count + 1
return List of anomalies
```

The algorithm can be summarised as follows: It is looped over the sensors and days. If the threshold value for a sensor is exceeded on a day, the system checks whether an anomaly is already active. If not, one is started, otherwise the loop simply continues. If a day is reached on which the limit value is not exceeded, it is checked what the value is on the following day. If this is not exceeded either, the anomaly is ended and saved if longer than d_{min}, otherwise discarded. If there is an exceedance again on the next day, the loop continues to run normally and the anomaly is still active.

The algorithm is designed for the case that data of a longer period of time is checked collectively. This is the reason for the query at the end, which takes care of anomalies still active at the end of the time series. If PRADA is used live, i.e. every day new data will arrive and PRADA is executed for the day before, this query can be omitted.

4.4 Hyperparameters

The aim of this chapter is to summarise and clearly present the hyperparameters and options for varying PRADA, in particular for the following explanations of the optimisation process. When PRADA is used, the following decisions can be made:

- **Regression models R_{day} and R_{base}:** Regression models which are used to calculate the coefficients β_{day} and β_{base}, on which PRADA is based on. It is also possible to use different models for β_{day} and β_{base}. In this work, OLS and robust regression is considered, but generally the choice is arbitrary.

- **Thresholds t_{day} and t_{base}:** Those hyperparameters are only relevant if the corresponding model (R_{day} and R_{base}) is a robust regression model. In this case, t_{day} and t_{base} describe the threshold above which residuals have less weight compared to the OLS estimator, as presented in Subsection 4.1.2. In this work, thresholds in the interval $(0.5, 5)$ have been considered since residuals have typically been in this magnitude or smaller in preliminary experiments.

- **Thresholds α_{day} and α_{ratio}:** Those thresholds define the maximum percentage deviation of β_{day} and β_{base} from 1. Therefore, they must be from the interval $(0, 1)$. The coefficient is then expected to be in the interval $\left(\frac{1}{1+\alpha}, 1+\alpha\right)$.

- **Window type $\delta_{\text{expanding}}$:** The default version of PRADA has a fixed size window which it looks in the back to calculate β_{base}. However, there is also the possibility to always use all available past data, in which it is an expanding window.

- **Size of window n_{lookback}:** When using a fixed size window, n_{lookback} defines the size of it. If using an expanding window, it defines the size of the first windows, in other words after how many days the analysis starts. In this work, integer values from $[3, 28]$ days were considered, but generally it can be any positive integer.

- **Minimum anomaly length d_{min}:** This variable defines the minimum length of a detected anomaly. In this work, integer values from $[1, 3]$ days were considered. Generally it can be any positive integer, but it should not be too long to avoid false negatives.
- **Threshold α_{anomaly}:** If this value is exceeded for a sensor on a day, it counts as anomalous. Must be in $(0, 1)$, also this whole interval has been considered in this work.

4.5 Optimisation Procedure

To obtain the best possible hyperparameters for the underlying problem and not just for a certain set of data, it is important to ensure that the model is generalising during optimisation. For this reason, this work uses a train-validation-test split, as common in related problems. There are three disjoint data sets that are used for different purposes:

- The training set is used to train the model. The optimiser aims to maximise the performance of the model on this data.
- The validation set is used to compare the performance of different models using previously unseen data within an optimisation run. Therefore, the model never receives this data for training, but is only evaluated on it after training.
- The test set is used to compare the model that was determined to be the best using the validation set with other methods or approaches (in this case the LSTM Autoencoder, see Chapter 5).

How the division into those three sets is executed in this work for the experiments is explained in Section 6.5, where the exact implementation of the optimisation process in this work is also discussed.

One important thing must be noted again at this point: Even though the optimisation process of PRADA is often referred to as training, there is no training process in the classical sense as known from many other machine learning methods. The optimisation process attempts to select the best possible hyperparameters described in Section 4.4. However, if PRADA is then applied to a dataset with chosen hyperparameters, the result is unambiguous and deterministic. There is no process in which further weights or parameters are determined.

Not all hyperparameters are relevant at the same time in PRADA. The hyperparameters R_{day}, R_{base}, α_{day}, α_{ratio}, n_{lookback} and $\delta_{\text{expanding}}$ must be set at the beginning, as they define the regression models and therefore have an influence on the process from the very start. However, the remaining hyperparameters d_{min} and α_{anomaly} only have an influence at a later point in time.

Therefore, it makes sense to optimise the hyperparameters that only become relevant at a later stage separately. This avoids having to consider multiple hyperparameter combinations that only differ in d_{min} and α_{anomaly}, thus repeating many identical computationally expensive steps.

The optimisation of the hyperparameters described in Section 4.4 therefore takes place in two stages: At the beginning of an optimisation run, R_{day}, R_{base}, α_{day}, α_{ratio}, n_{lookback} and $\delta_{\text{expanding}}$ are defined. These are then used to determine the error probabilities per sensor and day on the training set.

Hyperparameters d_{min} and α_{anomaly} only become relevant at this point. Now, the optimal values of these two hyperparameters given the others are determined and stored with the help of the already available error probabilities.

The entire process is repeated as long as possible in order to find the hyperparameters yielding the best results. Afterwards, the model with the best performance on the validation set is selected.

CHAPTER 5

LSTM Autoencoder

While PRADA is a conceptually new approach to solving the problem of anomaly detection for redundant sensor systems, the LSTM Autoencoder is a proven method in the field of anomaly detection and consists of two parts: One of them is the autoencoder, which consists of an encoder and a decoder. The former is intended to compress the input data into a representation of different dimension, while the latter is supposed to restore it. The goal is for the restored data to be as close as possible to the original data.

Which model is used as the encoder and decoder is generally arbitrary and depends on the data. In this case, an LSTM model is used. This stands for Long Short-term Memory and is a type of Recurrent Neural Network (RNN) that specialises in learning long-term dependencies in sequential data and is therefore particularly suitable for time series.

Together, these form the LSTM Autoencoder and can be used for anomaly detection. The model is trained with data that do not contain errors or anomalies, so the model learns patterns in the data and is able to reconstruct the data well, which means that the input and output are very similar.

To apply the model, is provided with data that it does not yet know. If the model is able to reconstruct this data, this indicates that the data follows the usual pattern and does not contain any anomalies. However, if the difference between input and output is large, i.e. the model is not yet familiar with this behaviour, then this indicates that the input data contains anomalies.

In this implementation, the model is not applied directly to the data from all sensors simultaneously, but rather, as in PRADA, they are compared in pairs. After that, a reconstruction error is determined for each sensor and day, and that is used to define the anomalies.

5.1 LSTM: Long Short-term Memory

Long Short-term Memory networks are a specialised version of Recurrent Neural Networks (RNN). Their development was a major step in the development of deep learning methods. They process sequential data by maintaining a hidden state that captures information from previous time steps. This makes them effective for tasks involving temporal patterns, but they often struggle with vanishing gradients which results in them having difficulty retaining long-term dependencies. LSTM was developed in 1997 [76] and tries to overcome this problem with a gating mechanism, which allows to keep information over many time steps.

Generally, a LSTM cell (also referred to as unit) has three key components [15]:

- **Cell State (C_t):** This acts as the memory of the network and keeps the information over the numerous time steps. Therefore, it takes care of the long-term dependencies.
- **Hidden State (h_t):** This is the current output of the cell at a specific point in time and handles the short-term memory.
- **Gating mechanism:** This regulates the flow of information and is explained below.

Another important component when creating an LSTM network is the input data x_t, although this is not part of the cell. The gating mechanism controls how information is updated and maintained within a memory cell over time. It consists of three gates:

1. **Forget Gate:** This gate controls how much of the previous cell state C_{t-1} is relevant and should be therefore be kept. Mathematically, this can be expressed as

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f).$$

   In the following, W always stands for weight matrices belonging to different steps or gates, while the different b represent biases. $[h_{t-1}, x_t]$ represents the previous hidden state concatenated with the input data of the current time step. σ is the sigmoid function, so $\sigma(x) = \frac{1}{1+e^{-x}}$. It ensures that the value of f_t is in the interval $[0, 1]$, where 0 means that everything should be forgotten and 1 that all information should be retained.

2. **Input Gate:** The input gate has two purposes: Firstly, it checks how much new information (previous hidden state and current input data) should be kept in the cell state. This is also called called candidate cell state and can be expressed as

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C).$$

The tangens hyperbolicus is used as activation function since its values are in $[-1, 1]$, and the negative values are used to reduce the impact. $\tilde{C}_t$ is also called candidate cell state.

Secondly, it checks which new information should be added. This can be expressed as

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i).$$

The new cell state C_t is then determined by

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t.$$

3. **Output Gate:** This gate determines the next hidden state h_t. First, the gate activation o_t is calculated by

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o).$$

The new hidden state can then be calculated by

$$h_t = o_t \cdot \tanh(C_t).$$

For a better understanding, the whole process has been depicted in Figure 5.1. When fitting the model, all input data must be processed by all LSTM cells, so the process of going through the three gates is repeated (C_t is becoming C_{t-1}, same for h_t and x_t) until this is fulfilled.

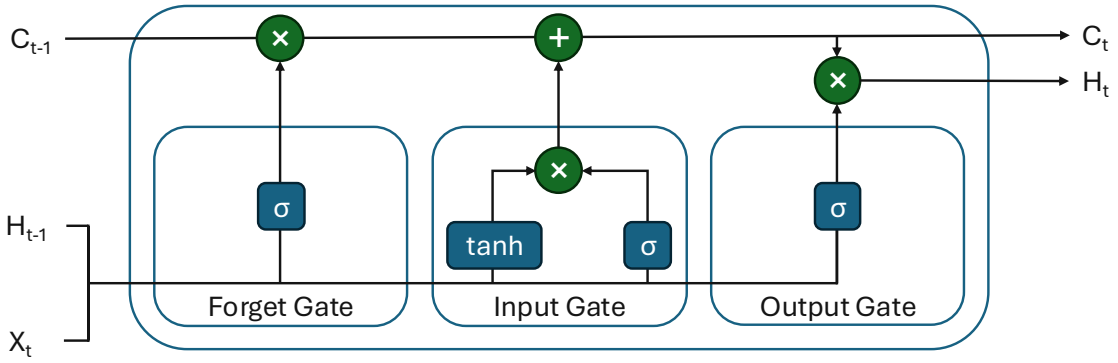

Figure 5.1: Structure of an LSTM cell

However, to form a functional network, more than one LSTM cell is necessary. For this reason, a specific number of LSTM cells are grouped together to form a layer. Within this layer, the cells act independently: Each has its own weights, cell state, and hidden state, but they receive the same input values x_t at each time step t.

It is not only possible to group multiple cells into one layer, but typically a network consists of multiple layers. This structure creates a flow from input to output. The first

layer processes the original input values x_t, while the second layer receives the hidden states h_t from the cells of the first layer as input. It's important to note that each cell in the second layer receives the hidden states of all cells from the previous layer. This layered architecture allows for the stacking of an arbitrary number of layers, potentially with a different number of cells each.

In this structure, the different layers often take over distinct roles. While the first layer might focus on extracting basic features or short-term dependencies, the following layers refine this information and capture more abstract representations or longer-term relationships. This hierarchy of layers allows the network to learn complex temporal dynamics.

It is also possible to add extra layers between the LSTM layers that serve additional functions, such as normalisation or regularisation. A commonly used method, which is also applied in this work, is dropout. There, a certain percentage of the data passed from one layer to the next one is set to zero. This helps to prevent the model from overfitting to the input data, and therefore improves generalisation. In this work, a dropout layer with a rate of 20% was added between all LSTM layers.

5.2 Autoencoder

Autoencoders are an unsupervised learning method suitable for different tasks, among them dimensionality reduction or anomaly detection [77]. They first compress the input data to a space with different dimension, also called latent space or *bottleneck*. This step is called encoding. Often, this latent space has a lower dimension than the original data, but this is not a requirement and depends on the use case. Then, in the decoding step, it is reconstructed back to the original input dimensions.

When training an autoencoder, the model tries to adjust all possible parameters so that the difference between input and output is as small as possible. This should help the model to learn which features of the data are necessary and which can be discarded during encoding. It is also possible that additional features and patterns which have been hidden in the data can be extracted. Since no ground truth or other labels are available, but the input and output data are the same, autoencoders fall into the category of unsupervised learning models.

In general, an autoencoder therefore consists of the following parts [13]:

1. **Input Layer:** This layer represents the values x that the autoencoder receives as an input. This depends on the use case and can be multidimensional.

2. **Encoder:** In this layer, x gets encoded by an encoder function e to a different-dimensional space. The function e is generally arbitrary and depends on the type of data and use case. In this work, LSTM models are used as encoders.

3. **Latent Space:** This space contains the encoded representation of x, $z = e(x)$. Even if z has a smaller dimension than x, it should still contain all relevant information.

4. **Decoder:** Here, z gets decoded back to the original dimension by the decoder function d. As for the encoder, also the function d is arbitrary. However, usually the same functions (with different parameters) are used for encoder and decoder. Therefore, in this work, LSTM models are also used as decoders.

5. **Output Layer:** This final layer represents the result of the autoencoder, $\hat{x} = d(z) = d(e(x))$.

The difference between x and $\hat{x}$ is called reconstruction error. How exactly this difference is defined depends again on the data and its format. In the typical case that input and output are vectors or matrices, the element-wise difference is often used together with a norm, e.g. the Euclidean norm. In the case that x and thus also $\hat{x}$ are vectors with length n, the reconstruction error is then given by $\sum_{i=1}^{n}(x_i - \hat{x}_i)^2$.

When training an autoencoder, it is the goal to change the parameters of the encoder and decoder functions $e(x)$ and $d(z)$ so that the total reconstruction error gets minimised.

After completing the training, the autoencoder can be used: Which part of the autoencoder is then used in the downstream task depends on the use case. If it is used for dimensionality reduction, the representation in the latent space is relevant. In the case of this work, where the goal is anomaly detection, the reconstruction error is of interest.

Figure 5.2 illustrates the structure of the autoencoder again, in this case LSTM is also used as the encoder and decoder, corresponding to the use in this work.

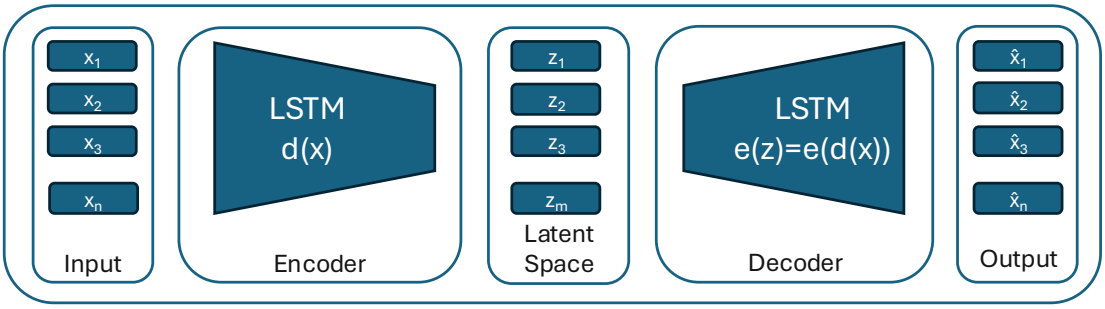

Figure 5.2: Structure of an LSTM Autoencoder

5.3 Calculating Reconstruction Errors

This section explains how the LSTM Autoencoder was implemented in this work. As already mentioned in the introduction to this chapter, the model is not applied directly to the data or slices of it, but the sensors are again considered in pairs. There are several reasons for this:

Firstly, an LSTM Autoencoder model is not flexible in the dimensions of the data after it has been defined. This means that if the model was defined and trained with the 10 sensors of the power plant `p0`, then the model only accepts data with 10 columns as input. This is very restrictive if sensors are added or removed. Furthermore, it makes it impossible to use a model across different power plants, as it is also examined in Section 6.5, Experiment 3.

Secondly, experiments with such an implementation have shown that the model is able to detect anomalies, but is poorly able to link them to a sensor. However, this is essential for the problem in this work. In cases where two or more anomalies are active at the same time, such an implementation was no longer able to distinguish between the different anomalies.

Thirdly, it is much more important to have a large amount of training data for the LSTM Autoencoder, since, in contrast to PRADA, a real training of the model takes place here. Especially with neural networks as LSTM, there is the problem that there is a very large number of parameters, but the available data may be insufficient to determine all of them. Splitting the data into pairs increases the amount of data: Assuming there are 10 sensors of the same orientation, a data set with 10 columns becomes $\binom{10}{2} = 45$ data sets with two columns each, which is nine times the amount of individual measurements. This is very helpful for training large deep learning models like this one.

Before the data is divided into pairs, it is scaled since many machine learning methods, especially neural networks, benefit from it [78]. Usually one of the intervals $[0, 1]$ or $[-1, 1]$ is used. Generally, it is common practice to scale each variable individually for multi-dimensional data sets. However, since it is assumed that all sensors usually measure the same, in this case the decision has been made in favour of global scaling. This also serves the purpose of simplicity. Experiments with scaling on a variable basis were carried out and led to almost identical results. Furthermore, $[0, 1]$ was selected as the new interval, as this is much more obvious, since the irradiance can only be positive.

At this point, it should also be noted that the parameters for scaling (minimum and maximum) must not be determined on the entire data set, but only on the training set. Then, the same parameters must also be used for the validation and test sets. This is the only way to prevent information from the validation and test sets from being included in the training process.

The next step is to break the time series down into sequences of a certain length. The length of these sequences is defined by the hyperparameter $n_{\text{timesteps}}$, which is also optimised at a later stage. This step also massively increases the number of data points available for the LSTM Autoencoder again, since the measurements of all timestamps, except for those at the very beginning and end, occur in $n_{\text{timesteps}}$ different sequences. Figure 5.3 gives an example how the data is split into sequences.

These sequences can then be passed to the LSTM Autoencoder, which always receives data in the shape $n_{\text{timesteps}} \times 2$ as input. During training, the reconstruction error between the input and output sequence is calculated directly and the model is adapted to minimise

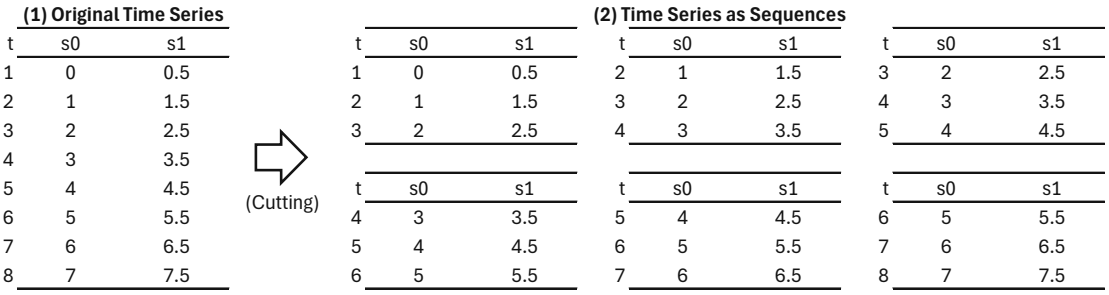

Figure 5.3: Example of converting a single time series into sequences

it. The applied library normally handles this automatically and the optimisation process is described in more detail in Section 5.6 and Section 6.6.

After successful training and optimisation, the LSTM Autoencoder is then used for anomaly detection. For this purpose, the model receives the sequences with two sensors each (i.e. shape $n_{\text{timesteps}} \times 2$) as input, encodes and decodes them, and returns data in the same shape as the input as a result. For anomaly detection, the reconstruction error is required. For an analysis of which sensor has a problem, however, these must first be aggregated. So the data in pairwise sequence format must be converted back into the table format, so that only 1 value (instead of $(n_i - 1) \cdot n_{\text{timesteps}}$) is available per sensor and timestamp. This is a multi-step process and works as follows:

First, the corresponding output from the LSTM Autoencoder is taken for each individual input and the absolute error is calculated for each element. From now on, the reconstruction errors are used instead of irradiance measurements.

Next, the sequences are converted back into a long time series so that only one value per pairwise comparison is available for each sensor at each point in time. The mean is used for this, i.e. the average over all reconstruction errors of a measurement across the sequences is calculated. There are various options for the exact technical implementation of this. In the case of this work, the sequences are stored as a three-dimensional array. This allows the mean to be calculated very efficiently by shifting along an axis and then calculating the mean along that axis. However, the exact implementation is arbitrary.

Now the data is in the format of a single time series again instead of many short sequences, but it is still separated into pairs. The next step is therefore to aggregate the values from each pair into one value for each sensor. Various methods were evaluated for this and it was found to be advantageous to use the minimum function. This is because the reconstruction error is often increased for both sensors in a pair, even if only one of them is affected by an anomaly. Similar to the procedure for PRADA, it can be assumed that all pairs of the sensor are affected by an error. The minimum function takes this into account: a sensor only has a high reconstruction error if that is the case in all pairwise comparisons.

After this step, the data has the original format again, so there is one value per sensor

and timestamp, in this case a reconstruction error. Figure 5.4 illustrates this process again for a better understanding using example values.

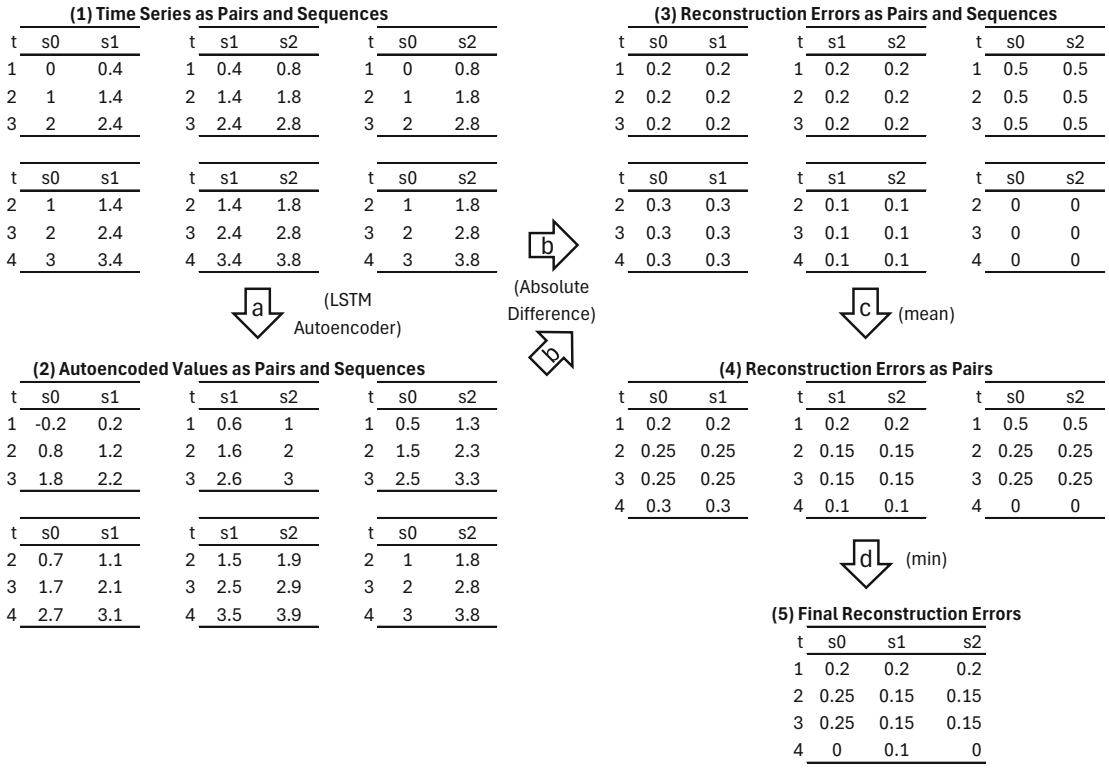

Figure 5.4: Example of calculating the reconstruction error from sequences and converting them back into a single time series

First, in step (a), the sequences are fed to the autoencoder which returns the reconstructed values with the same shape. Then, the original sequences and the reconstructed values are used to calculate the absolute reconstruction error in step (b). Afterwards, in step (c), the sequences can be merged by taking the mean of all timestamps which occur in multiple sequences. As a last step (d), the pairs are merged by taking the minimum reconstruction error of each combination of sensor and timestamp, which results in the final reconstruction errors on a timestamp basis.

As with PRADA, a statement is to be made on a daily basis in the course of this work. The reconstruction errors just obtained per sensor and timestamp must therefore be aggregated again. This work considers two strategies for doing this:

- **mean:** In this case, the aggregated daily value is calculated as the mean of all individual reconstruction errors of a day. No relation to the other sensors is taken.

The reconstruction error at day d of sensor s is thus defined by

$$r_{ds} = \frac{1}{|T_d|} \sum_{t \in T_d} r_{ts},$$

where $|.|$ denotes the cardinality of a set, T_d the set of all timestamps on day d and r_{ts} the reconstruction error of sensor s at timestamp t.

- **`normalised_mean`:** When using this strategy, the mean reconstruction error of one sensor is related to the mean error of all other sensors. This is to prevent days being declared as erroneous on which all sensors have increased reconstruction errors due to external circumstances. In the case of this work, the external circumstances might be, for example, a day with a rarely occurring weather phenomenon which the model therefore has never seen. The daily reconstruction error is then given by

$$r_{ds} = \frac{\frac{1}{|T_d|} \sum_{t \in T_d} r_{ts}}{\frac{1}{|S|-1} \sum_{\tilde{s} \in S \setminus s} \frac{1}{|T_d|} \sum_{t \in T_d} r_{t\tilde{s}}}, \tag{5.1}$$

  where S denotes the set of all sensors. The numerator in Equation 5.1 therefore describes the average reconstruction error of the sensor s, and the denominator that of all other sensors.

As desired, a reconstruction error is now provided for each sensor and day and the next step can be taken, the definition of anomalies.

5.4 Defining Anomalies

The procedure for defining anomalies in the LSTM Autoencoder is very similar to that of PRADA, which is described in Section 4.3. Again, a minimum length d_{min} and a limit value α_{anomaly} need to be defined.

However, there is a difference here: Since the LSTM Autoencoder considers reconstruction errors and instead of error probabilities, there is no upper bound. Nevertheless, they are always positive, as always the absolute error is considered.

Which values are typical for the reconstruction errors depends on the selected aggregation strategy: With the strategy `mean`, values between 0 and 0.3 are common, while with the strategy `normalised_mean` they are usually between 1 and 3. Accordingly, a suitable α_{anomaly} must also be in this interval.

The further procedure for defining anomalies with the help of reconstruction errors is identical to the procedure for PRADA with the error probability, which is why at this point it is again referred to Figure 4.2 and Algorithm 4.2.

5.5 Hyperparameters

This section intends to present a clear overview of all hyperparameters that can be adjusted in this work for the LSTM Autoencoder, to provide more context and to justify any related decisions. With LSTM networks in particular, the exact hyperparameters available also depend on the implementation used (in this work, see Section 6.6), but the most important ones are mentioned here:

- **Aggregation method** $\delta_{\text{aggregate}}$**:** This hyperparameter defines which of the two methods presented at the end of Section 5.3 should be used to aggregate the reconstruction values by day.

- **LSTM layers** n_{layers}**:** This hyperparameter defines how many layers the LSTM encoder and decoder models should have. In this work, encoder and decoder always use the same architecture, therefore they have the same amount of layers and units. n_{layers} refers to both encoder and decoder, so the LSTM Autoencoder model as a whole has $2 \cdot n_{\text{layers}}$ LSTM layers. Generally, this can be any positive integer number, in this work 2 to 6 layers have been considered to allow for a great variety of models. More layers would yield too complicated models for the given data availability.

- **LSTM units** n_{units}**:** This hyperparameter defines how much cells/units each of the LSTM layers has. While every positive integer is possible theoretically, it is common to use powers of 2, so the values $[16, 32, 64, 128]$ have been considered in this work.

- **Learning Rate** η**:** During optimisation, the learning rate controls the size of the steps taken to minimise the loss function. It can be any real value between 0 and 1, but small values are common. In this work the interval $[0.00001, 0.1]$ was chosen.

- **Time Steps** $n_{\text{timesteps}}$**:** This defines how long the sequences that the LSTM Autoencoder takes as input should be. In theory, it can be any positive integer, but the amount of available timestamps must be considered. In this work, the values $[4, 8, 16, 24, 32, 48]$ were considered. These are all multiples of 4, since the data used in this work is given at 15-minute intervals and the sequences are therefore always full hours.

- **Minimum anomaly length** d_{min}**:** This defines the minimal length of an anomaly. As for PRADA, 1, 2 or 3 days have been considered, but it is generally arbitrary.

- **Threshold** α_{anomaly}**:** A sensor counts as anomalous if the reconstruction error exceeds this threshold on a given day. In the case of the LSTM Autoencoder, α_{anomaly} can be any positive real value.

5.6 Optimisation Procedure

As for the definition of anomalies, the optimisation procedure of the LSTM Autoencoder is very similar to the one of PRADA, introduced in Section 4.5. Again, it's very important to ensure generalisation of the model: Therefore, the data is split into training, validation and test set. Special care is taken that no data is spilled from validation or test set into the training data.

However, remember that real training process takes place in the LSTM Autoencoder. Even after the hyperparameters mentioned in Section 5.5 have been selected, the training process is only initiated afterwards, which can be different depending on the implementation. Often, this also includes a stochastic component, whereby the result can differ slightly for different runs.

Also for the LSTM Autoencoder, not all hyperparameters are relevant at the same time. The hyperparameters $\delta_{\text{aggregate}}$, $f_{\text{activation}}$, n_{layers}, n_{units}, η and $n_{\text{timesteps}}$ need to be defined before any training of the model starts, because they define very basic properties of the underlying model. The two remaining hyperparameters however, d_{min} and α_{anomaly}, only become relevant when the aggregated reconstruction error is already available and the only task left is defining anomalies.

Therefore, equivalent to PRADA, a two-stage optimisation of the hyperparameters presented in Section 5.5 is used here as well, to prevent multiple identical calculations and to make the optimisation process more efficient.

Also here, the optimisation process is carried out again as long as possible to try a wide variety of hyperparameter combinations. Afterwards, the model with the best performance on the validation set is selected for further comparisons with other approaches.

CHAPTER 6

Experimental Setup

This chapter presents all necessary details of the experiments comparing the two different approaches from Chapter 4 and Chapter 5 performed in the thesis. As a first step, the considered data is explained in detail, followed by the executed preprocessing steps. To ensure objectivity and allow for sensible comparisons, artificial errors are added after data cleaning. Those have already been mentioned in Section 2.4 from a technical perspective, while the exact implementation is described in this chapter. This is followed by a description of the evaluation methods and a precise formulation of the experiments carried out.

6.1 Characteristics of Data

This work presents and evaluates the anomaly detection approaches with the help of data from two photovoltaic power plants. As introduced in Section 2.4, they are located in Austria and Spain. Some characteristics about the power plants have already been shown in Table 2.1, while this chapter focuses on the available data.

Generally, the expected shape of irradiance data is highly dependent of the geographic location, which influences all aspects like length of the day (period of non-zero irradiance), difference between the seasons, maximum possible irradiance level and so on. In any case, the expected irradiance level is 0 W/m^2 during the night (between sunset and sunrise), and positive during the day. Note that there are also regions on the earth where the sun does not set or rise at some time of the year - in this case, the expected behaviour is accordingly.

This thesis considers data from a power plant in Austria (abbreviated as `p0`) and one in Spain (`p1`). For the plant in Austria, data is available from January 2023 until July 2024, while for the plant in Spain, data coverage begins mid-April 2023. Figure 6.1 shows

the mean irradiance (averaged across all sensors of one plant) for the two existing power plants.

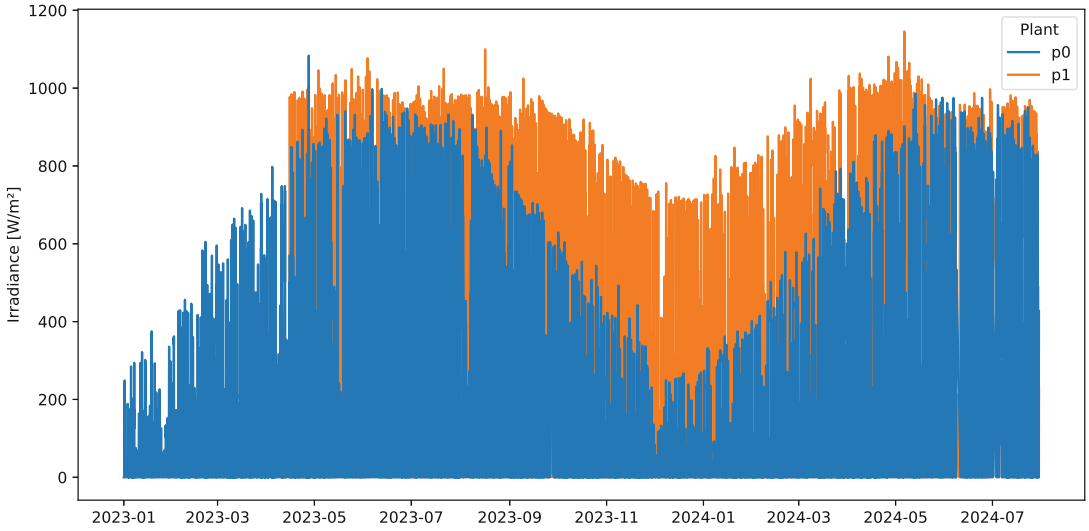

Figure 6.1: Average irradiance measurements over the entire observation period for power plants `p0` and `p1`

The graph shows clear differences between the two time series. Throughout the whole year, the power plant in Spain measures more irradiance than the one in Austria. In summer, more than 1,000 W/m^2 are measured in Spain, while Austria lags behind by about 10% with around 900 W/m^2.

The difference between the seasons can be clearly observed. In Spain, the measured irradiance drops from about 1,000 to 800 W/m^2 on days with good weather, which represents a reduction of 20%. In Austria, on the other hand, the measured values drop to about 300 W/m^2 in winter, which represents a drop of 67% compared to the summer value. The differences between the seasons are therefore much more distinct in Austria than in Spain, which can be explained by the geographical location.

These statements are also confirmed by Table 6.1. This shows the average measured irradiance values for the two power plants across all sensors and time points. The significantly lower values are therefore due to the fact that these also include measurements from the night. Here again, the general advantage of the power plant in Spain can be seen, as well as the different behaviour between the seasons.

When looking closer at single days more interesting behaviour can be noticed. Figure 6.2 shows the irradiance levels for all sensors of the Austrian plant for four different days. It is immediately visible that they are significantly different: Day 1 is a clear-sky day in summer, showing high irradiance levels and a steady rise and fall during the day. Day 2 also features a lot of sunshine (comparable irradiance levels with day 1), but much more unstable, which is a sign of light cloud cover. On Day 3, the day begins with sunshine,

	Plant in Austria (p0)	**Plant in Spain (p1)**
Whole period	145.33	232.13
Spring	171.90	251.33
Summer	236.43	289.48
Autumn	103.43	203.31
Winter	49.36	152.29

Table 6.1: Average irradiance measurements by season of power plants `p0` and `p1`

but around noon it changes rapidly and the remaining day is very unstable. On Day 4, the whole day has very low irradiance levels, which indicates a constant strong cloud cover. It should be noted that the time on the x-axes of Figure 6.2 (and also all the following graphs) is given in UTC and not in local time.

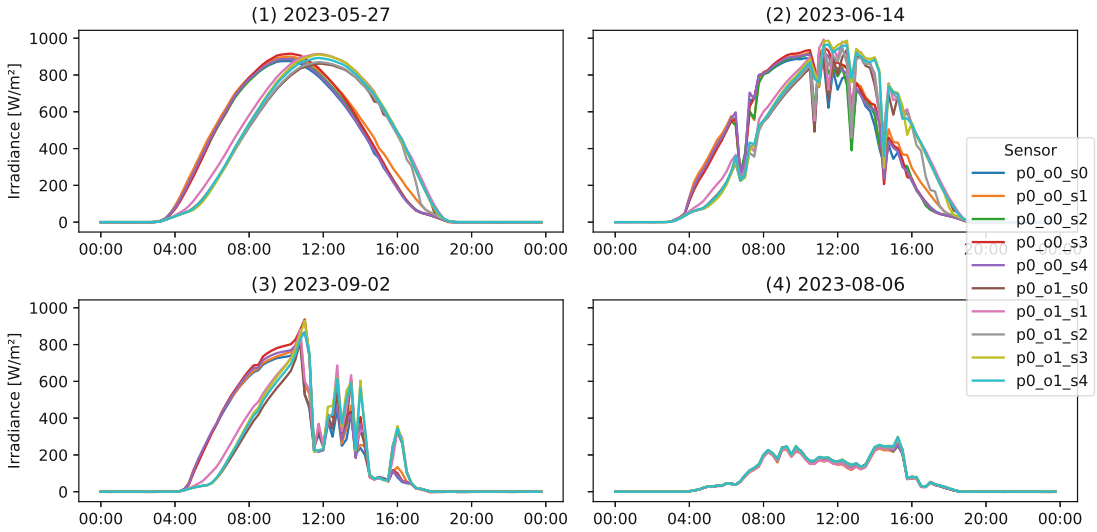

Figure 6.2: Various weather conditions and corresponding irradiance measurements of the Austrian power plant `p0`

Another interesting fact to notice is the difference between sensor orientations. Especially visible on Day 1, the peak in irradiance of the sensors facing to the east (sensor names containing `o0`) is reached about 2-3 hours earlier than of the sensors facing west (sensor names containing `o1`). On days with low irradiance levels (like Day 4), this effect can not be seen because the irradiance mainly comes from the diffuse light due to the lack of direct sun.

Note that even though all days in Figure 6.2 show completely different behaviour, this is expected and none of those records feature an anomaly: All sensors of one orientation measure approximately the same, also the relationship between the different orientations is coherent. On Day 4, which was a very cloudy day, there is not even a difference

between the two orientations.

The data set has a granularity of 15 minutes, which equals 96 data points per sensor and day. As for most real-world data sets, the data set is not complete, but contains some missing values. The number and proportion of missing values can be seen in Table 6.2.

	Plant in Austria (`p0`)	**Plant in Spain (`p1`)**
Full data set	553,230	952,203
Existing data points	528,900	893,484
Missing values (absolute)	24,330	58,719
Missing values (ratio)	4.40%	6.17%

Table 6.2: Overview of the data quantity and missing values before data preparation

6.2 Data Preparation

As a first step of data preparation, a visual inspection by the author of this work together with domain experts from the operator of the two power plants was performed. This is necessary to ensure good data quality and fair conditions for the comparative study.

Starting with the power plant in Austria, `p0`, the number of data points showing unusual behaviour is low, but there are some points in time or intervals that could show an anomaly and need to be looked at more closely.

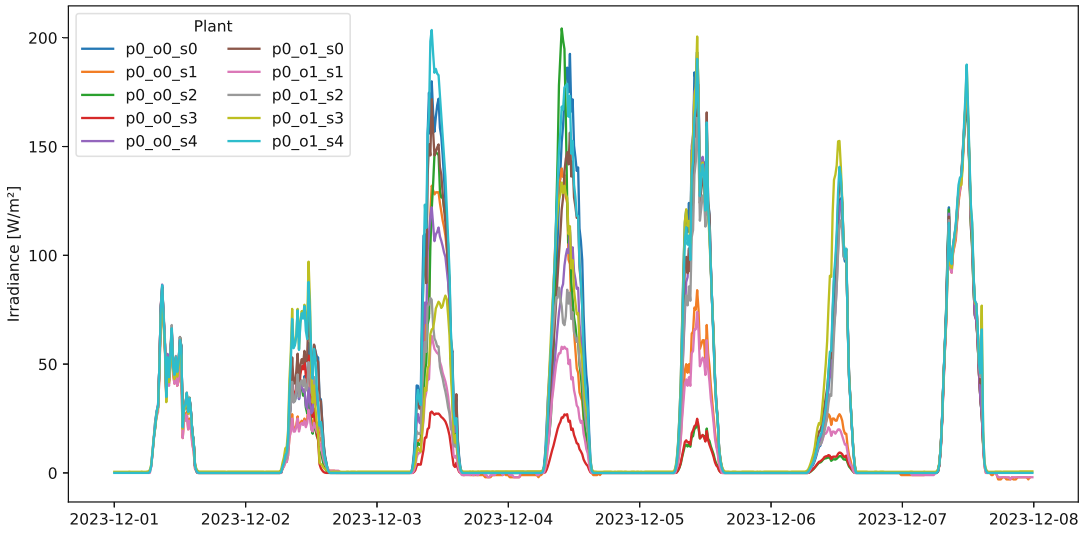

Figure 6.3: Unexpected behaviour due to snowfall at the Austrian power plant `p0`

From December 2nd, 2023 until December 6th, 2023, the data does not show the expected behaviour. As visible on Figure 6.3, the irradiance levels of the sensors facing east (`o0`) tend to be much lower, but also sensors facing the same direction show significant

variances. Together with domain experts and historical weather records [79], this was due to heavy snowfall, and the data of the affected days was removed.

On February 7th, 2023 and January 10th, 11th and 21st, 2024, anomalous behaviour of the sensors `p0_o0_s1` and `p0_o1_s1` was observed. Two of the affected days are depicted in Figure 6.4. The cause could not be clearly determined, but to ensure good data quality, the affected days were also removed completely.

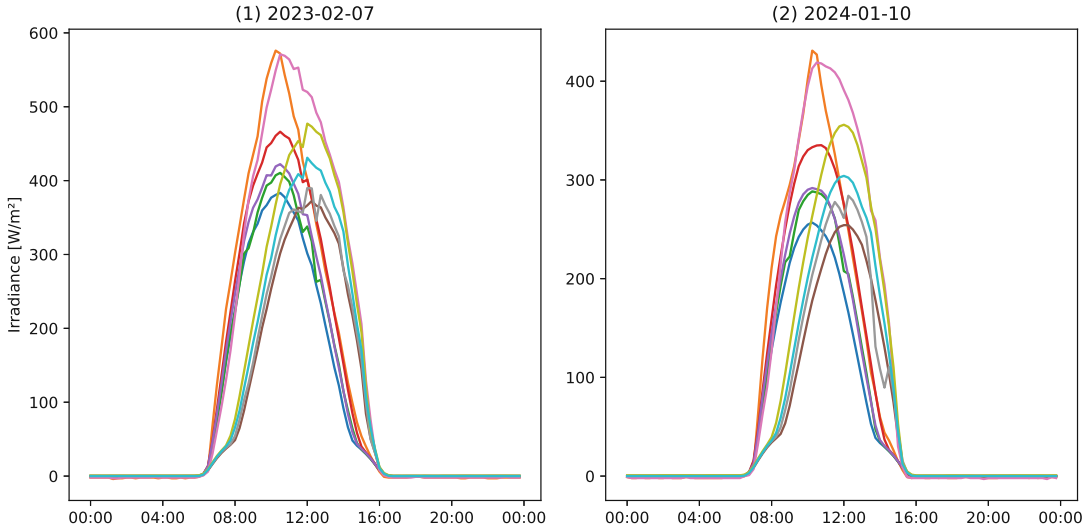

Figure 6.4: Unexpected behaviour on two specific days at the Austrian power plant `p0`

When looking at the power plant in Spain, `p1`, in more detail, there are a few very obvious anomalies. Firstly, irradiance sensor `p1_o0_s20` constantly measured 0 W/m^2 throughout the whole time series and all data of it was removed as a consequence.

Another interesting behaviour was shown by the sensors `p1_o0_s1` and `p1_o0_s9` and is depicted on Figure 6.5. From autumn 2023 until summer 2024, the measured irradiance is always 30-50% lower than all other sensors of the same orientation. While this is generally possible due to shadowing or different terrains [80], the magnitude, duration and period of this deterioration is unexpected, especially since it also occurred on days with heavy clouds and thus no direct sunlight. No exact cause could be determined, but due to the uncertain data quality, it was decided to remove all data from these two sensors, which reduces the total number of sensors of `p1` to 18.

Even though the focus of this work lies on anomalies that span across a longer period of time (both methods make their statements about whole days), the data should also be free of very short anomalies (called point anomalies, if they are only visible in one measurement, or very short subsequence anomalies [81]).

For the plant `p0` in Austria, the visual check could not detect such anomalies, but for the Spanish plant `p1` three anomalies of this type could be found. In these cases, the

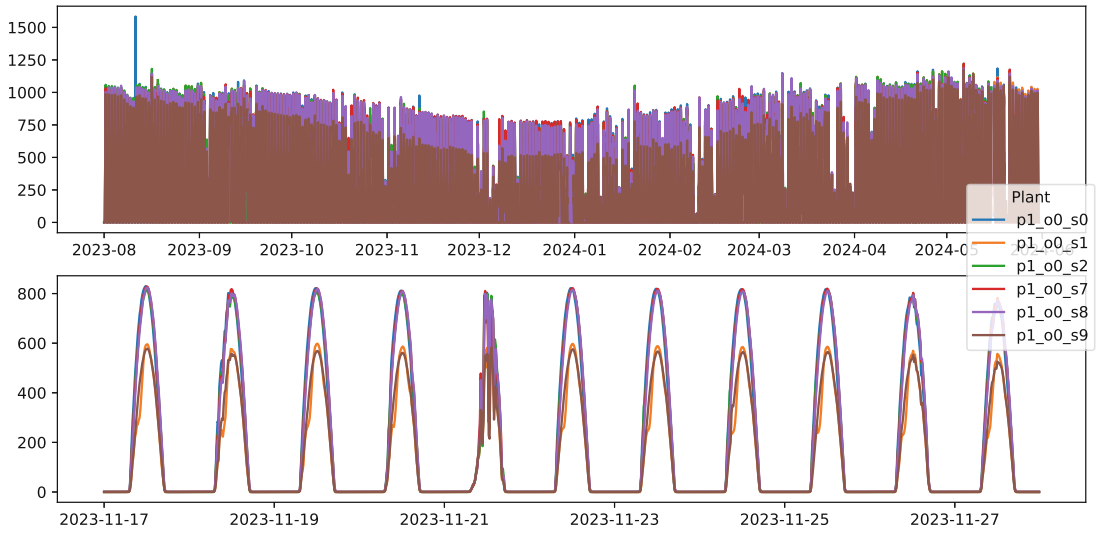

Figure 6.5: Unexpected behaviour over an extended period at the Spanish power plant `p1`

measured values are physically impossible to achieve, since they massively exceed the highest possible energy arriving at the atmosphere [82]:

1. **August 11th, 2023 05:15-05:30**: `p1_o0_s0` and `p1_o0_s3` measured up to 1,584.05 W/m^2 before sunrise
2. **December 8th, 2023 22:00-22:15**: `p1_o0_s3` measured up to 2,740.51W/m^2 after sunset
3. **January 10th, 2024 21:30-22:00**: `p1_o0_s10` measured up to 3,276.70 W/m^2 after sunset

Anomaly 2 can also be seen in Figure 6.6. Since those anomalies seem to appear very randomly, no direct cause could be determined by the domain experts, and all affected data points have been removed from the data.

The complete absence of further anomalies naturally can never be guaranteed, but all reasonable measures for this have now been taken and the next step of data preprocessing can be carried out.

While PRADA has no problem with missing data, as long as each combination of same-oriented sensor has at least one measurement at the same time, the LSTM Autoencoder can not handle missing data. In daily operation, this prerequisite can be achieved with a simple check before applying the method and depending on the extent, imputation of values or removal of timestamps or sensors, while taking note of the missing values, since repeated occurrences can also be a sign of a defect.

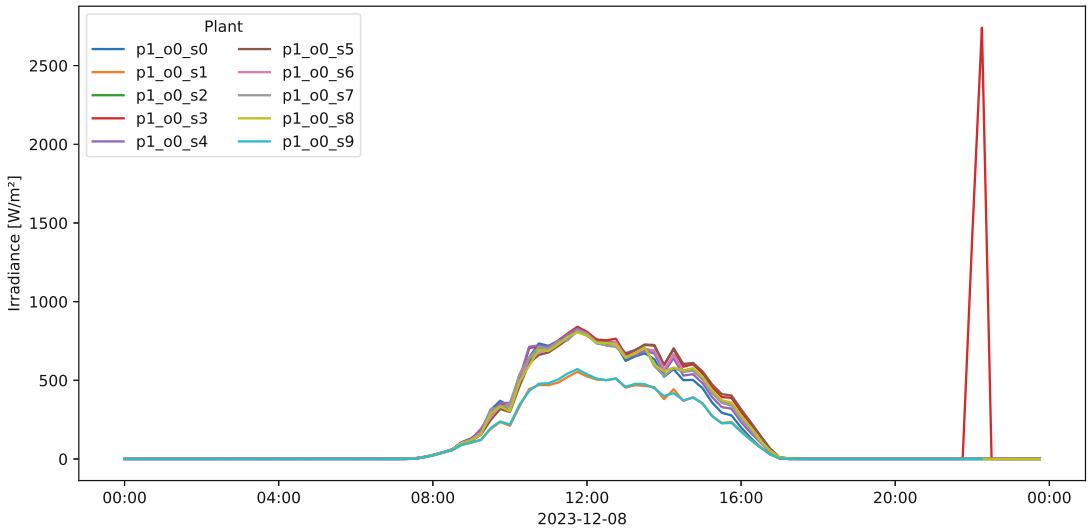

Figure 6.6: Unexpected behaviour at a specific timestamp at the Spanish power plant `p1`

However, in order to ensure a fair comparative study of the two approaches, the assumption of complete data must be established for both algorithms. To achieve a complete data set, while still having a data set representative of the reality and being reasonably sized, the structure of missing values in the data set is studied in greater detail.

The first row of Figure 6.7 shows the number of anomalies during the observation period for both power plants. It is evident that there are a few hotspots of missing values, but aside from that, the missing values are distributed fairly evenly. Interestingly, there was a significant increase in missing values in the spring and summer of 2024 for both power plants. In Austria, missing values became more frequent, while there were still regular instances of all sensors functioning with zero missing values at some timestamps. In Spain, however, five sensors (`p1_o0_s2` through `p1_o0_s6`) failed completely in May 2024, raising the minimum number of missing values per timestamp to five from that point onward. Due to the nearly simultaneous emergence of the increase in missing values, it is also possible that the cause lies not with the power plants themselves, but rather with the data collection process. However, this could not be determined with certainty.

The second row of Figure 6.7 shows the distribution of missing values across the different sensors. In Austria, there are quite significant differences: Sensors `p0_o0_s2` and `p0_o1_s2` failed to report values much more frequently than the others. Compared to these two, the values of the remaining sensors are significantly lower, but there are still substantial differences among them.

The situation in Spain is more balanced: The five sensors, which have not been reporting data since May 2024, are of course clearly at the top of the statistics with around 7,500 missing data points each, but they are quite similar to each other. The remaining sensors are well below that with around 1,500 missing values and are also quite balanced.

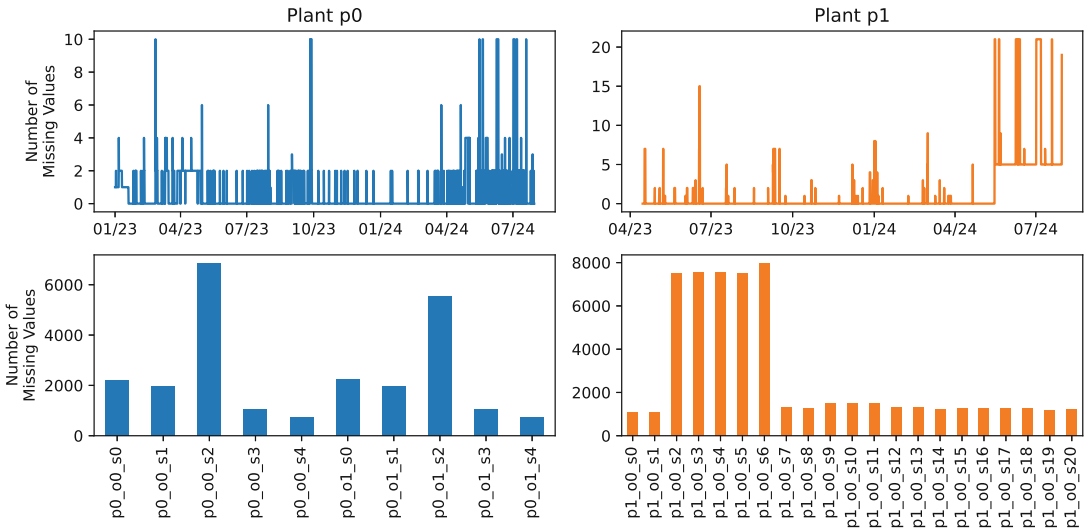

Figure 6.7: Structure of missing data of both power plants `p0` and `p1`

Note that the assumption of complete data was introduced, meaning that all considered sensors should have data for all timestamps of a day (e.g. for the Austrian plant `p0`, a full day consists of $10 \cdot 96 = 960$ data points per day). A simple and obvious way to achieve this goal would be to remove all entire days for which at least one data point is missing. This would guarantee data completeness without having to perform imputations and thus making further assumptions.

However, in this case only 324 and 331 of the originally available 568.60 (`p0`) and 462.52 (`p1`) days would remain. This is too restrictive, which is why a more liberal strategy must be adopted, which also requires the careful use of imputation.

The next step was to try removing all days that exceed a certain ratio of missing data points. This makes imputation necessary, but by limiting the maximum ratio of missing data points, the influence on the results should be kept as low as possible.

Various minimum quotas for the available data between 70% and 95% were tested. The difference between the various levels turned out to be smaller than expected: For example, a reduction in the quota from 90% to 85% for the power plant in Austria added about 22 days, while in Spain the difference was even smaller and only 4 days were added.

In the end, a minimum availability of 90% was the best compromise between as many data points as possible for the comparative study and as little missing data as possible, and was therefore set at this level. The resulting missing and available data points used for the experiments can be seen in Table 6.3. Note that the data of `p1` now ends in May 2024 due to the failure of 5 sensors, as described above and visible in Figure 6.7.

Now that only days with at least 90% data availability remain, the only thing missing for a complete data set that meets our requirements is imputation.

	Plant in Austria (p0)	**Plant in Spain (p1)**
Possible data points after preprocessing	460,800	667,008
Existing data points	457,724	664,262
Missing values (absolute)	3,076	2,746
Missing values (ratio)	0.67%	0.41%

Table 6.3: Overview of the data quantity and missing values after data preparation

In general, there are many different ways to impute missing data points. A very simple option is to simply fill the missing value with the mean value of the respective variable or with another value that is determined to be plausible [83]. More complicated examples would be using different machine learning approaches [84] or bootstrapping [85] to replace missing values. There is also a trend towards multiple imputation, in which missing values are replaced by multiple plausible estimates in order to better reflect the uncertainty caused by the missing data and to achieve more robust results [86].

If the underlying data is a time series, this brings new possibilities on the one hand, but on the other hand, some methods may no longer be applicable. New possibilities could be back-fill/front-fill (in which a gap is filled with the closest available data point in the past/future) or interpolation [87]. This involves determining a plausible value to be inserted on the basis of the existing data points around the gap. Typically, a linear interpolation is used, but there are many other possibilities depending on the application, e.g. polynomial or spline interpolation. Also methods like an LSTM networks, which is used for anomaly detection in this work, can be used for imputing [88].

Often back-fill or front-fill are the natural choices for time series, but they are not in this case. As also visible in Figure 6.2, the irradiance is expected to steadily rise from sunrise until about noon, then start to decrease again until sunset. Only during the night, the value is expected to be constantly around 0 W/m^2, which is therefore the only time where those method provide realistic data.

Hence interpolating is the much better choice in this case, since it considers both values before and after the gap and can therefore imitate the increase and decrease during the day, while still staying constantly 0 during the night (assuming the points before and after the gap also did so). Due to those properties, interpolating was chosen as imputation method. Different interpolation functions were considered. However, due to both simplicity and the following key reason, linear interpolation was chosen: On days where the irradiance graph has a steady shape (like Day 1 in Figure 6.2) it can be considered approximately linear for short time periods because the data granularity (15 minutes), while on days with unstable weather, interpolating might produce results further away from reality, but still does not perform worse than other methods.

However, this is rarely necessary due to another measure: Remember that this work considers a redundant sensor setup, which yields the assumption of all sensors having the

same orientation measuring approximately the same value. Therefore, it is reasonable to fill data gaps based on the measurements of the other sensors. To put this into practice, it is achieved by calculating the mean of all sensors with available data of the same orientation and using this value for imputing.

For example, if the sensor `p0_o1_s0` of the power plant in Austria has a missing data point, the average of the sensors `p0_o1_s1`-`p0_o1_s4` is used for imputing. Since the power plant `p1` in Spain only has one orientation, all available sensors are used for the calculation. This ensures that the imputed data is much closer to reality compared other methods and at the same time also prevents the introduction of new anomalies.

Interpolation is then only used if all the sensors of an orientation (i.e. all the sensors in Spain) have failed simultaneously at a certain point in time. For the data used in this thesis, after removing the days with more than 10% missing data, this is never the case in Spain (there is at least one measurement at every timestamp), while in Austria there are two events of all sensors failing, in total affecting 7 timestamps. Due to the negligible number of points affected and the fact that most of them are located at night, it can be assumed that interpolation has no influence on the results and can be carried out as described.

This concludes the data preprocessing and leaves the following data for the next steps: For the plant `p0`, data of 482 days is available (which equals 46272 observations of 10 sensors each), while for the `p1` the data comprises 386 days (37056 observations and 18 sensors).

6.3 Adding Artificial Anomalies

With the preprocessing presented in Section 6.2, the assumptions of complete and anomaly-free data can now be considered fulfilled. However, in order to carry out the comparative study and compare the performance of the two algorithms, anomalies in the data are required. For this purpose, domain experts and past experience (see also the errors found in Section 6.2, or [18, 89]) were used to discuss which errors can occur in th redundant sensor setup of the considered type.

As part of this work, a framework was developed that adds artificial errors to a given data set. While this is optimised for the conditions of this thesis and their experiments, it is generally very modular in design and can be extended as desired (e.g. by adding further error types). Also the occurrence and variation of the individual errors can be controlled very precisely. To define the anomalies that should be added to the data, a dictionary (referred to as `ERROR_CONFIG`) must be provided. The key-value pairs of this dictionary define all parameters and provide, assuming the data does not change and a seed is set, a unique and reproducible error setup.

The most important key in the `ERROR_CONFIG` dictionary is `ERROR_PROB`, which controls the probability that an anomaly occurs. If no anomaly is currently active at one sensor, one is generally beginning with the probability provided by `ERROR_PROB`.

However, experience has shown that different anomalies do not tend to be completely independent, so if one sensor is currently experiencing a failure, chances are increased that another one also does so as well because of external reasons. For this reason, the following formula was implemented which determines the probability of an anomaly starting. In this case p_{base} represents the base probability provided in the dictionary, n_t stands for the number of currently active anomalies and p_t for the actual error probability at timestamp t.

$$p_t = p_{base} \cdot 2^{n_t}$$

In other words, the probability of an error occurring is doubled with each additional sensor that is currently faulty.

An artificial error always has a specified duration, which is determined by the key `ERROR_DAYS_INTERV` in the `ERROR_CONFIG` dictionary. The corresponding key has the shape `{"low": min_dur, "high": max_dur}`, defining the minimum and maximum duration of the anomaly. For each added anomaly, the duration is then uniformly distributed from the interval `(min_dur, max_dur)`.

There is one special case in which this does not apply: The assumption of complete days was introduced, which made it necessary to remove data of some days and therefore create gaps between days. This raises the issue how to handle the combination of errors and data gaps. Several options were considered, but it was concluded that it is best to simply avoid having artificial anomalies overlapping with data gaps. This works as follows:

Before adding an error, it is checked how far the next data gap is in the future. If it is further away than `max_dur`, the error can be added in any case without special attention. If the next data gap is closer than `max_dur`, but further away than `min_dur`, the random duration must be shorter than the duration until the gap. If this is not the case, the random draw for the duration is repeated until the condition is met. Only if the next data gap is closer than `min_dur`, the insertion of the error be cancelled.

This naturally raises the concern if the knowledge that an anomaly cannot extend beyond a data gap might also be learned by the algorithms and thus distort the results. However, at least in all the cases considered in this work, this can be ruled out because neither of the two algorithms has access to data that lies in the future, and in both cases the date information is removed before the calculations, so the algorithms do not know if there is a gap between the days under consideration or if they were consecutive.

Another very important entry in the `ERROR_CONFIG` dictionary defines the different types of errors that can be added and their distribution. For the key `ERROR_TYPE_PROB`, the value should be another dictionary which has the different types of anomalies (as string) as keys, and their corresponding distribution (must sum up to 1) as values.

In this work, three different types of artificial errors are considered. Those have been presented from an operators perspective in Section 2.4, in the following the implementation and possible variations of the errors are discussed.

6.3.1 Measurement is Constant (`const`)

This type of anomaly occurs, for example, when the measuring element of a sensor is defective or the sensor has fallen from its mounting to the floor. Another possibility would be a problem with the electronics of the sensor. This artificial anomaly is characterised by the fact that the sensor reports only one constant value during the entire duration of the anomaly. In many cases, this is 0, but it can also be a different value.

Figure 6.8 gives two examples how an artificial anomaly of type `const` looks like: Example 1 shows an average cloudy day, where the weather gets worse as the day progresses. In the morning, one can see the difference between the two orientations, but in general the measurements are very similar. However, one sensor (`p0_o0_s0`) clearly deviates from this, as it constantly measures 0. While this is generally a plausible value, it is not at this time of day and given the results of the other sensors. In Example 2, the defective sensor is `p0_o1_s1`, which constantly measures -1,175.85 W/m^2, which is physically impossible in any situation.

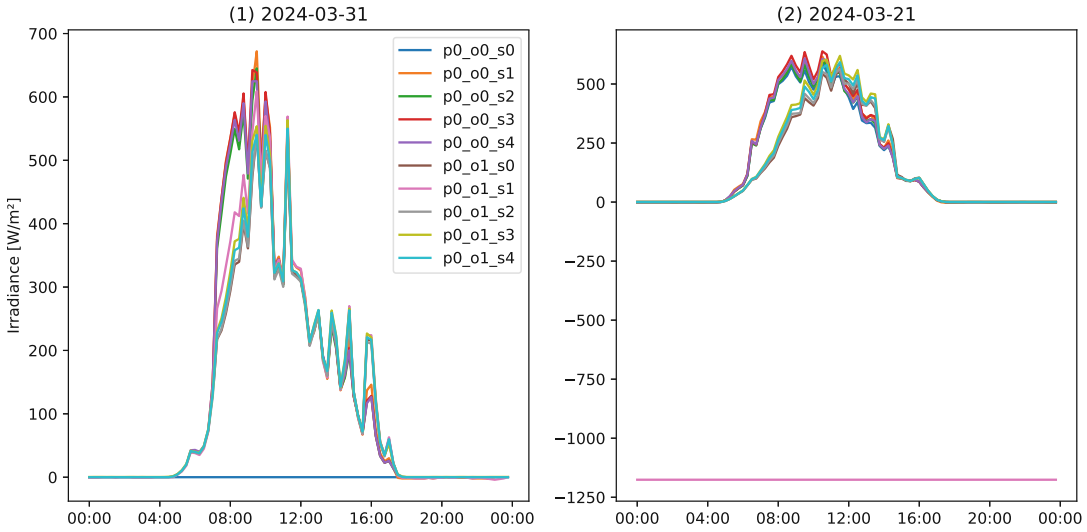

Figure 6.8: Examples for artificial error of type `const`

In a later chapter, more details will be given on how information about the artificial errors is stored. For this purpose, the two example errors just discussed were used, among others, and are therefore visible with IDs 60 and 61 in Table 6.7.

For this anomaly type, another key-value pair is added to the configuration dictionary `ERROR_CONFIG`: The record `CONST_VALUE_PROB` defines which values are feasible to

be added as constant errors and their corresponding probabilities. The value is again a dictionary with the format `{0: prob_0, (low, up): prob_interv}`, where the probabilities should add up to 1. In this case, there are two possibilities: With probability `prob_0`, the constant value is 0 W/m^2, while with probability `prob_interv` it is a random value from the interval `(low, up)`, drawn using an uniform distribution.

However, since the framework is modular, the keys of the dictionary are arbitrary and can be any number or interval (as a tuple), also multiple times. Thus it is possible to define very specific mathematical sets, using a combination of (disjoint or overlapping) intervals and single numbers. To give an example, a dictionary including all of the possibilities would be `{0: 0.1, 1: 0.1, (4,5): 0.3, (4.5,5.5): 0.2, (5,6): 0.3}`, which in mathematical representation equals

$$\texttt{const\_value} = \begin{cases} 0 & \text{with probability } 0.10, \\ 1 & \text{with probability } 0.10, \\ U[4, 4.5] & \text{with probability } 0.15, \\ U[4.5, 5.5] & \text{with probability } 0.50, \\ U[5.5, 6] & \text{with probability } 0.15. \end{cases}$$

6.3.2 Measurement is Close to Reality, but Distorted (`deter`)

This type of anomaly is characterised by the fact that the measurements correlate with reality, but do not represent it correctly. An example of this would be a sensor that is covered with a layer of dust and therefore measures less irradiance than actually arrives. This means there is a proportional deterioration of reality.

Each anomaly of this type falls into one of three categories, which differ in the development of the deterioration:

1. **Constant deterioration:** The deterioration remains constant throughout the entire duration of the anomaly, so the measured value and the true value are always in the same ratio to each other.
2. **Increasing deterioration:** In this case, the deterioration becomes more and more severe until the sensor eventually always measures 0 W/m^2.
3. **Decreasing deterioration:** This case is the opposite of the one just described. The deterioration becomes weaker over time, i.e. the measured value approaches the true value. When the anomaly ends, the sensor again measures values that correspond to reality.

Figure 6.9 gives examples how those three types of deterioration anomalies can look like: The first graph shows `p0_o1_s3` having an anomaly with constant deterioration. While it is active, it reports 33% of the real value. On November 28th, the error ended and

the sensor operates again as intended. The second example in the plot shows two more of those anomalies: On April 29th, an anomaly starts for sensor `p0_o1_s2` (13% of original value). On May 2nd, `p0_o1_s0` starts to fail with deterioration to 39%.

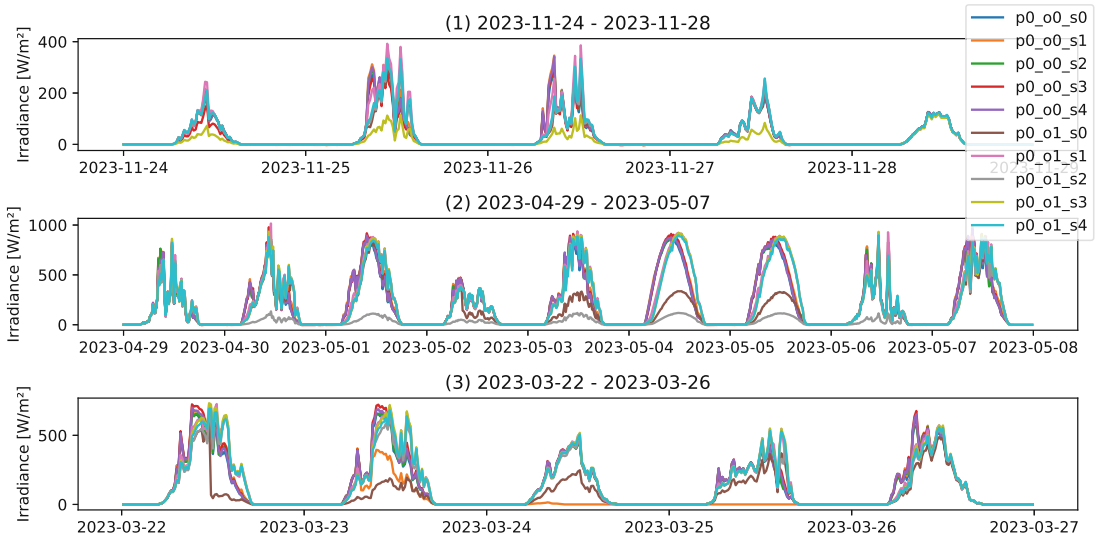

Figure 6.9: Examples for artificial error of type `deter`

Until now, only examples with constant deterioration have been considered. Example 3 covers those two remaining types: On March 22nd, `p0_o1_s0` begins an anomaly with decreasing deterioration. It starts with deterioration to about 11%, but then steadily gets closer to the original value which it reaches after 4.5 days. On March 23rd, `p0_o0_s1` starts to experience an anomaly with increasing deterioration. It first falls to 63% of the original value, then starts to decrease and reaches 0 after a bit more than a day (45% of the total time), and stays like this for about 27 more hours. The just described anomalies can be also be found in Table 6.7 with the IDs 45 (Example 1), 10 (Example 2), 8 and 9 (Example 3).

Table 6.4 shows which key-value pairs need to be added to the `ERROR_CONFIG` dictionary and describes their use.

6.3.3 Measurement is Random (`rand`)

This type of artificial anomaly causes the values measured by the sensor to be random, so they have no relation to the actual irradiance value, but they are also not constant. This can occur if there is a problem during the transmission of the measurement data and the actual measurements are replaced by noise.

In this thesis, this was implemented using a random walk [90]. Starting from 0, a random number drawn from an uniform distribution is added or subtracted in each timestep. Mathematically, this can be expressed as follows:

Key	Value	Description
DETER_INTERV	{"low": min_deter, "high": max_deter}	Deterioration ratio is chosen uniformly distributed from provided interval
DETER_CHANGE	{None: prob_none, "up": prob_up, "down": prob_down}	Probabilities that decide if the deterioration stays constant or increases/decreases
DETER_DOWN_RATE	{"low": min_deter_down, "high": max_deter_down}	For increasing deterioration, ratio (uniformly distributed from provided interval) of the total time after which 100% deterioration should be reached

Table 6.4: Entries in the ERROR_CONFIG dictionary for the anomaly type deter

$$X_t = X_{t-1} + x_t \text{ where } x_t \sim U(l, u),\ X_0 = 0 \text{ and } t \geq 1 \tag{6.1}$$

This raises the need for another entry in the ERROR_CONFIG dictionary: The value provided for the key RANDOM_BASE is a dictionary, defines the parameters for the random walk, denoted as l and u in Equation 6.1, and has the format (analogous to entries made before) {"low": rand_walk_min, "high": rand_walk_max}.

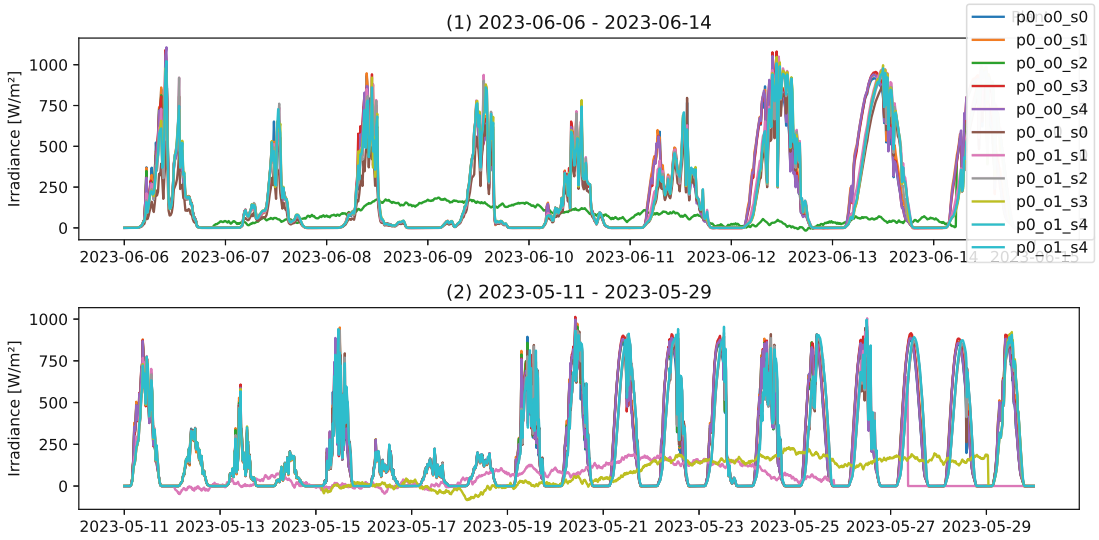

Figure 6.10: Examples for artificial error of type rand

As for the previous anomaly types, examples how those artificial anomalies can look like are provided in Figure 6.10. As the name suggests, this method yields very unique paths of the time series: Sensor p0_o0_s2 in example 1 and sensors p0_o1_s1 and

p0_o1_s3 all look different. Also other paths are possible, walks reaching over 1,000 W/m^2 have also been observed, while others stay close to 0.

Also those artificial errors can be found in Table 6.7 with the IDs 16, 12 and 13.

6.3.4 Choice of Parameters

In the previous pages, a large number of parameters were introduced that must be selected for the experiments in this work. The selection of these is not trivial, it was tried to create as realistic conditions as possible. Another goal was to allow as much diversity in errors as possible in order to better evaluate the strengths and weaknesses of the algorithms.

The following shows the standard configuration of ERROR_CONFIG used in this work. Some experiments presented in Section 6.5 deviate from this, but unless otherwise stated, these parameters were used. This also serves as an overview and summary of all the configuration parameters introduced in Section 6.3.

```
ERROR_CONFIG =
    {"ERROR_PROB":         0.01/96,
     "ERROR_DAYS_INTERV": {"low": 1, "high": 14},
     "ERROR_TYPE_PROB":   {"const": 0.25,
                           "deter": 0.5,
                           "rand": 0.25},
     "CONST_VALUE_PROB":  {0: 0.5, (-10000, 10000): 0.5},
     "DETER_INTERV":      {"low": 0.1, "high": 0.9},
     "DETER_CHANGE":      {None: 0.5, "up": 0.3, "down": 0.2},
     "DETER_DOWN_RATE":   {"low": 0.1, "high": 0.7},
     "RANDOM_BASE":       {"low": -10, "high": 10}}
```

The following reasons were key to the choice of parameters:

- ERROR_PROB: Various error probabilities were examined in order to obtain as many anomalies as possible for a good comparative study, while still remaining realistic. An error rate of 0.5% to 1.5% per sensor and day turned out to be optimal, so the middle of this interval was chosen. Domain experts have confirmed that, when considering all types of failures and problems together, this value is realistic, albeit rather high. Given that this work benefits from a larger dataset to evaluate the proposed approaches, this is considered acceptable. The value is then divided by 96 (number of timestamps per day), since the probability here does not refer to a day, but to a timestamp.
- ERROR_DAYS_INTERV: The lower limit (1 day) was chosen because this work focuses exclusively on longer anomalies, as described in Chapter 2, and not on point anomalies.

The upper limit was chosen so that, on the one hand, a relatively large number of different error lengths would arise, but on the other hand, there would also be a clear differentiation from permanent malfunctions. It can also be assumed that most problems will be resolved by the operator of such a power plant within 14 days.

- `ERROR_TYPE_PROB`: For this parameter, the focus was on creating the greatest possible variety of different errors. Since the error type `deter` has 3 different characteristics, the probability of this error was set at 50%, with the other types sharing the rest equally. This also makes sense in view of the real circumstances, since this type of error is to be expected several times a year (snowfall in winter, no rain for a longer period and therefore soiling in summer).
- `CONST_VALUE_PROB`: For the error type `const`, the value 0 or any other value (determined by an equal distribution) in the interval $[-10^4, 10^4]$ is chosen with equal probability. The value 0 W/m^2 was explicitly picked out because it is the most common in reality, as described in Section 2.4. The interval was chosen to often remain in the same order of magnitude as the real measured values and thus to be a greater hurdle for the algorithms, but sometimes also provide impossible values, which should be easier to detect.
- `DETER_INTERV`: For the values of the possible deterioration, an attempt was again made to leave all possibilities open. Only deteriorations to less than 10% or more than 90% of the original value were excluded, as these are too close to a constant 0 or to the real measurement data.
- `DETER_CHANGE`: These parameters were chosen as a compromise between a wide range of error selection and reality. The neutral case, in which the deterioration remains constant, was therefore chosen as the highest at 50%, with the rest was distributed with a slight tendency towards a smaller weakening, as this case is more regularly expected due to snowfall in winter.
- `DETER_DOWN_RATE`: The aim here was to introduce as wide a range of different errors as possible, and the interval was therefore deliberately chosen to be large. Values below 0.1 were excluded because this would correspond to an unrealistically fast degradation process. Values above 0.7 were not considered because it is unlikely that the error would reach 100% deterioration exactly at the end of the error, rather it should end with a constant phase in any case.
- `RANDOM_BASE`: For the random walk, various parameters for the change in each timestamp were examined. The aim here was for most of the curves to be in the same order of magnitude as the real irradiance measurements, but also to go above and beyond them at times. This was achieved very well with the interval $[-10, 10]$.

To conclude this section, Figure 6.11 shows the distribution of errors over the first 7 months of the data when the parameters just described are chosen and 10 is used as the

seed. These are also the same errors that have been used as examples in this chapter, and some of which are listed in Table 6.7 and detected in Table 6.8. It is visible that the anomalies are definitely present and there should be enough of them to carry out reasonable experiments, but on the other hand (it should be remembered that a line in Figure 6.11 always refers to only one sensor) the sensors are operating in the normal state for most of the time.

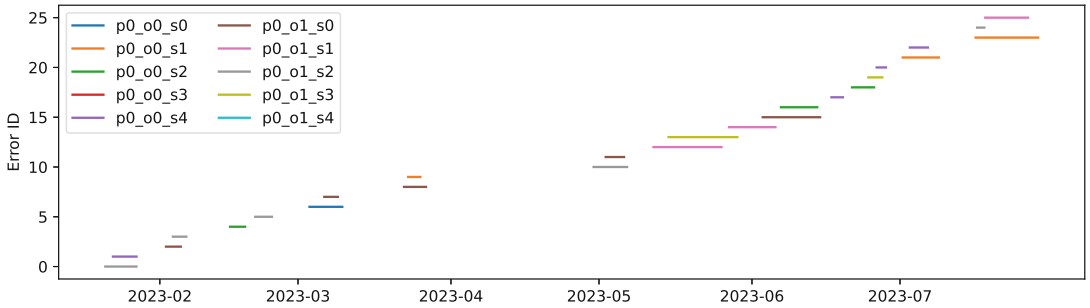

Figure 6.11: Timeline of artificial anomalies using parameters described in Section 6.3

6.4 Evaluation Metrics

Before the peculiarities of the evaluation in this work are discussed, a basic overview of classification and its evaluation methods is given. Generally, a distinction is made between binary, multi-class, multi-labelled and hierarchical classification [91]. In the following, only binary classification is discussed, as this is the only relevant case for the given problem.

In this case, each data point belongs to exactly one group, and in the classification, each data point (using external information, without using the real group membership) is predicted to belong to one of the two groups. In this case, it is assumed that the two groups are *positive* and *negative*, although this is arbitrary. Each of the data points then falls into one of the following 4 categories:

- **TP (True Positive)**: Points that are positive and were also predicted as such.
- **TN (True Negative)**: Points that are negative and were also predicted as such.
- **FP (False Positive)**: Points that are negative, but were predicted as positive.
- **FN (False Negative)**: Points that are positive, but were predicted as negative.

The quantities of these 4 categories are usually represented by a confusion matrix. Table 6.5 shows one possible way how a confusion matrix is often shown, also a representation

		Actual	
		Positive	**Negative**
Predicted	**Positive**	TP	FP
	Negative	FN	TN

Table 6.5: Definition of a confusion matrix for binary classification

as figure or list is possible. In some literature, the transposed version is preferred, so that FP and FN are also swapped.

The quantities of the four categories (TP, TN, FP and FN) can then be used to calculate various evaluation metrics [92], some of them shown in Table 6.6. Again, the information is limited to what is relevant for this work.

Evaluation Metric	**Definition**	**Description**
Accuracy	$\frac{TP+TN}{TP+FP+TN+FN}$	Proportion of correct predictions out of all predictions
Precision (P)	$\frac{TP}{TP+FP}$	Proportion of true positives among predicted positives
Recall (R)	$\frac{TP}{TP+TN}$	Proportion of true positives among actual positives
F1-Score	$\frac{2 \cdot P \cdot R}{P+R} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$	Harmonic mean of precision and recall

Table 6.6: Selected evaluation metrics for binary classification

The various metrics are all based on the values from the confusion matrix, but they have different implications and different metrics are appropriate depending on the situation. For example, accuracy is a commonly used metric when the focus is on the total number of correct classifications. However, it can quickly become misleading, especially when dealing with imbalanced groups of different sizes. For example, it is possible that a classifier has 98% accuracy, but fails to detect any relevant data point [93].

Precision is particularly relevant when false-positive classifications have serious consequences, while false-negative classifications are less problematic (e.g. recognising spam emails). In contrast, recall is more important when avoiding false-negative classifications is more important than minimising false-positive classifications (e.g. cancer screening) [94]. F1 score is a good choice when it is unclear whether precision or recall is more important, as it balances both metrics.

Now that all the general requirements for the evaluation have been met, the focus is once again on the problem considered in this work. When artificial anomalies are added, not only the modified time series is stored, but also information about the added errors. These are, in any case, the affected sensor, the start time, the duration and the type of error. In addition, there are the characteristics of the respective error type, e.g. for an error `const` the value that is assumed to be constant, or for the type `deter` the

deterioration level and whether this remains constant or not. All this information is referred to in the following as ground truth.

If PRADA or the LSTM Autoencoder is applied, the result is very similar: The affected sensor, the start time and the length in days are reported for the detected errors. In this case, the start time is always midnight and the length is always an integer, because the algorithms both work on a daily basis, but this assumption is not necessary in the following. Depending on the algorithm, further values such as the reconstruction error are reported, but not used further in this work.

Table 6.7 gives an example how the table containing information about the artificially added errors looks like. Those were added with seed 10, and have also been used to give examples for the different error types considered in this work: Anomalies with ID 60 and 61 can be seen in Figure 6.8, 45, 10, 8 and 9 can be seen in Figure 6.9, and 16, 12 and 13 can be seen in Figure 6.10.

ID	Sensor	Type	Begin	Days	const_ value	deter_ base	deter_ change	deter_ down_ value
60	p0_ o1_s1	const	2024-03-19 02:45	4.56	-1,175.85	-	-	-
61	p0_ o0_s0	const	2024-03-26 04:45	7.47	0.00	-	-	-
45	p0_ o1_s3	deter	2023-11-24 01:15	3.84	-	0.33	-	-
10	p0_ o1_s2	deter	2023-04-29 21:45	6.78	-	0.14	-	-
8	p0_ o1_s0	deter	2023-03-22 11:45	4.45	-	0.11	up	-
9	p0_ o0_s1	deter	2023-03-23 07:30	2.48	-	0.63	down	0.45
16	p0_ o0_s2	rand	2023-06-06 21:00	7.35	-	-	-	-
12	p0_ o1_s1	rand	2023-05-12 01:00	13.77	-	-	-	-
13	p0_ o1_s3	rand	2023-05-15 02:30	13.94	-	-	-	-
68	p0_ o1_s0	deter	2024-05-14 10:30	5.38	-	0.56	up	-

Table 6.7: Example of artificially added errors

The result received after applying one of the two algorithms considered in this work looks very similar. One example is given in Table 6.8, where PRADA has been applied to the

data just described. As for the added errors, this shows only the relevant part of the result.

ID	Sensor	Begin	Days	Error probability
67	p0_o1_s1	2024-03-19 00:00	5	1.00
69	p0_o0_s0	2024-03-27 00:00	6	1.00
49	p0_o1_s3	2023-11-24 00:00	4	1.00
6	p0_o1_s2	2023-04-30 00:00	7	0.98
4	p0_o1_s0	2023-03-23 00:00	2	1.00
5	p0_o0_s1	2023-03-23 00:00	3	1.00
16	p0_o0_s2	2023-06-10 00:00	4	0.97
10	p0_o1_s1	2023-05-15 00:00	11	0.90
9	p0_o1_s3	2023-05-15 00:00	15	0.84
60	p0_o1_s4	2024-02-08 00:00	5	0.62

Table 6.8: Example of detected errors after applying PRADA or the LSTM Autoencoder

This raises the question of how the evaluation is carried out after or during an experiment having those two tables. The theory just presented cannot be used directly, since here not every data point/timestamp is classified, but rather errors that extend over a certain period of time. However, with a little adaptation, the introduced evaluation metrics can still be used.

All entries in both tables are checked and assigned to one of the following three cases:

- **TP (True Positive)**: This applies to all errors that occur in both tables, i.e. those that have been added artificially and those that have been found. This raises the question of how the matching between added and found errors works, since these are not normally completely identical (the added ones have an arbitrary start time and duration, while the found ones always have midnight or integer, respectively).

  This was solved as follows: First, a check is made to see whether there is a suitable pair of entries at all, i.e. which concerns the same sensor and for which the affected periods (start + duration) overlap. If this is not the case, one of the other two cases applies. If such a pair exists, the system checks how large the overlap between the added and found error is. A minimum overlap of 25% was defined, which means that the overlapping period must account for at least 25% of the total duration of both the added and found error.

  This prevents errors from being classified as identified even though they only overlap in a very small area by chance. If this is the case and the overlap is less than 25%, the points fall into the other categories (FN or FP). If there is at least 25% overlap, the pair is counted as a correctly identified error (TP).

  Figure 6.12 shows some examples of added and detected errors: In Example 1, the anomaly is detected one day after its start, but is considered active two days longer

than it really is. However, there is still a huge overlap (87.5% for the ground truth and 75% for the detected error, so the overlap is definitely sufficient to consider the anomaly as detected. In Example 2, the detected error is much shorter than the ground truth, but still completely within its duration. Even though the duration is not correct, it is still enough to clearly detect that the sensor is not working as expected, which is the goal in this thesis. In Example 3 however, the overlap between the true anomaly and the detected error is very short. Here it is not possible to say whether the error was detected or this was just a random occurrence, so in this case, the anomaly would not be considered detected.

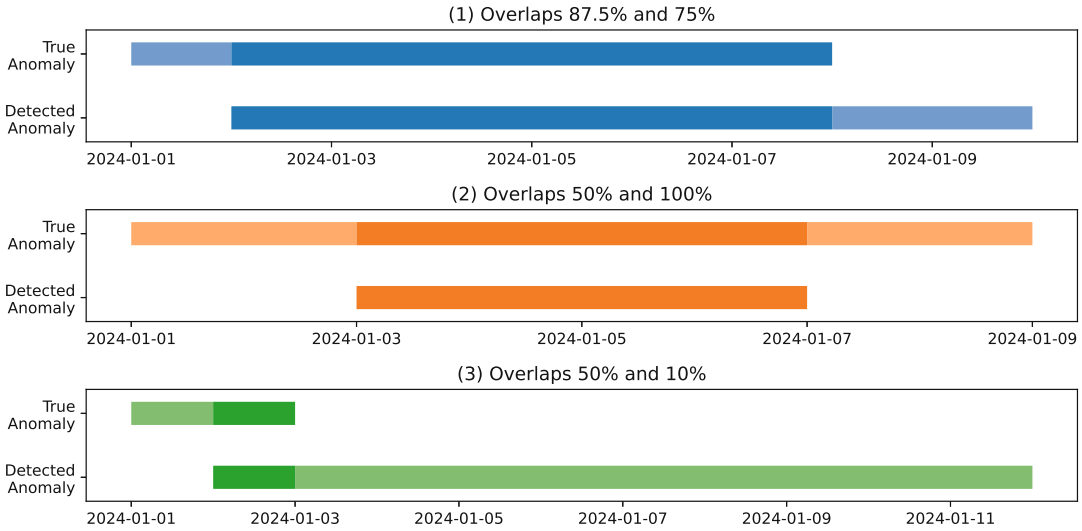

Figure 6.12: Examples of error pairs (ground truth and detected) with different overlaps

The minimum overlap of 25% was deliberately chosen because in the setting of this work, it does not depend on a single day. It is much more important that the error is reliably detected than the exact start and end time. However, this procedure ensures that no pairs are identified as correct in which the added and detected error have no connection.

In the example tables Table 6.7 and Table 6.8, the first 9 entries of each table form pairs (e.g. ID 60 from Table 6.7 forms a pair with ID 67 from Table 6.8) and are therefore considered TP. The overlap is not specified explicitly, but can be checked easily: For the first pair, the true anomaly is completely covered by the detected anomaly (100% overlap), which is only about 11 hours longer in total than the real error ($\sim$ 90% overlap). This therefore clearly fulfils the condition of minimum overlap.

- **FN (False Negative)**: This category describes errors that were added artificially but could not be detected, or not with sufficient overlap. This class therefore

includes entries from the ground truth. In Table 6.7, ID 68 would be an example of this case, since it does not have a counterpart in the table of detected errors.

- **FP (False Positive)**: This category contains time periods that were detected as anomalous, but for which there is no counterpart that was added artificially. This therefore includes entries from the set of detected errors, and in Table 6.8 ID 60 would be an example.

The evaluation situation at hand resembles the general case, but there are notable differences: Not every data point or timestamp is classified, but a connected unit (errors that span a sensor and at least 96 timestamps). For this very reason, the class of true negatives (TN) does not exist, which prevents the use of certain metrics such as accuracy.

In the situation at hand, there is no clear tendency whether the consequences of false positives or negatives are more serious: False negatives are problematic because they lead to lost revenue, and the failure to recognise a problem might make it more serious and expensive to resolve. However, false positives are just as problematic, as they tie up valuable personnel resources that could be better used elsewhere. Since there is no clear trend here, the F1 score was chosen as evaluation method for the following experiments, as it does not require the number of false negatives and creates a good balance between the two types of errors and therefore ensures a fair comparative study.

6.5 Experimental Plan

The fundamental question of this work is which of the two algorithms, PRADA and the LSTM Autoencoder, is better suited for the use case presented in Chapter 2. The aim is to provide an overall assessment, as well as to look at individual aspects in more detail in order to be able to better justify the decision and to be prepared for any changes that may occur. Since it is not possible to carry out all experiments in parallel for both power plants, efforts have been made to design the experiments in order to maximize the advantages offered by the existence of two datasets, including the application of Transfer Learning.

Overall, the focus of the experiments is on the power plant in Austria, `p0`. Although it has fewer sensors than `p1`, it is also sufficiently large. In addition, it covers a longer period of time, adds additional challenges with the two different orientations, and its data was available at an earlier stage in the development process of this work. This also makes the calculations less complex, as the algorithms scale much better in terms of the time considered than in terms of the number of sensors due to the pairwise comparison.

Another difficulty is the question of the train-validation-test split. As described in Chapter 4, PRADA does not actually train a model, but only a selection of hyperparameters has to be determined. For this, very small training and validation sets would be sufficient, which would create a lot of room for a test set that is as large as possible and thus contains more anomalies. In contrast, the LSTM Autoencoder learns the correct behaviour using

an error-free training set. In this case, the process benefits from having more training data, so the split should favour the training set as much as possible. The problem of those two diametrically opposed tendencies can be resolved by the fact that the validation and the test set can be used multiple times with different artificial errors generated through different seeds. Consequently, the dataset split can prioritize the training set while still ensuring adequate evaluation on the validation and test sets. Table 6.9 shows the resulting threshold dates used for the split of the dataset and the corresponding ratios for both power plants `p0` and `p1`.

	Plant	Training	Validation	Test
Period	**Austria (p0)**	01/2023-12/2023	01/2024-03/2024	04/2024-07/2024
	Spain (p1)	04/2023-12/2023	01/2024-02/2024	03/2024-05/2024
Ratio	**Austria (p0)**	60.63%	18.33%	21.04%
	Spain (p1)	65.80%	15.03%	19.17%

Table 6.9: Threshold dates for splitting data into training, validation and test sets, with resulting ratios

In all of the following experiments, the validation and test sets are considered eight times with different anomalies by setting a different seed. After a detailed assessment of the situation and the performance of diverse intermediate experiments, the decision was taken to set up the comparative study as follows:

1. **Training and Evaluation on Heterogeneous Anomalies:** This experiment should assess the general anomaly detection capabilities of the two approaches. For doing so, artificial errors of all three types are added to the data with the parameters introduced in Section 6.3. A detailed search for the best hyperparameters is then carried out on the training data and the best model is selected based on the performance on the validation set. The score for comparison with other methods or experiments is determined on the test set. This experiment is executed on both datasets, `p0` and `p1`, whereby the focus is on `p0` due to the significantly shorter durations of training and application.

2. **Evaluation on Homogeneous Anomalies:** This experiment is intended to test whether one of the approaches is particularly suitable for one or more of the three error types. For this purpose, the best models of Experiment 1 are evaluated on a test set which only contains anomalies of one of the types `const`, `deter` or `rand`.

3. **Transfer Learning: Cross-Dataset Evaluation:** Both Experiment 1 and Experiment 2 consider the two power plants separately, so the models are only applied to data of the same power plant on which they were trained on. This raises the question of whether this separation of models by power plant or at least region with the same irradiance is necessary, or whether the models obtained generalise well enough to be used at any power plant.

To assess this, the models that performed best on `p0` are applied on data of `p1`, and the performance is compared to the one of the native model, meaning the model that was trained on and performed best on `p1`. The findings are essential for the application of the approaches in real-world scenarios, as they attempt to answer the question of whether an operator should try to achieve the best possible model, but then use it centrally and without any adjustments, or whether a training process for each power plant is more sensible.

6.6 Implementation

All programs and experiments described in this work were implemented with Python 3.10.14 on a machine running Ubuntu 20.04.6 LTS. It has 2 AMD EPYC 7452 processors with 32 cores each and 1 TB RAM. Two GPUs are available for tasks that can make use of CUDA (in the case of this work only the LSTM Autoencoder): An NVIDIA Quadro RTX 8000 with 50GB memory and an NVIDIA Quadro RTX 5000 with 16GB memory.

Optuna [95] 3.6.1 was used to optimise the hyperparameters for both PRADA and the LSTM Autoencoder. `TPESampler` was used as the sampler because it is suitable for the combination of numerical and categorical variables in this case. TPE stands for Tree-structured Parzen Estimator and works by fitting two Gaussian Mixture Models in each trial and for each hyperparameter and then selecting the most promising value [96].

In both algorithms, the optimisation takes place in two stages, so certain hyperparameters are not determined by Optuna, but are only optimised when the error probabilities (for PRADA) or reconstruction errors (LSTM Autoencoder) are available. This was implemented using a grid search. A grid is defined for all relevant hyperparameters. To do this, approximately 100 equidistant values are selected in the definition range. After that, the best value is determined and the process is repeated four times in the closer range of the previously best hyperparameter (about 10% of the previous range).

PRADA was largely implemented from scratch, with the exception of the `OLS` and `RLM` functions used for regression, which were provided by the Statsmodels library [97] 0.14.2. While most of the detection logic of the LSTM Autoencoder was also programmed directly, Keras [98] 3.4.1 and Tensorflow [99] 2.17.0 were used for the underlying models.

CHAPTER 7

Results and Discussion

In this chapter, the results of the experimental part of this thesis are presented. First, it follows the structure of the three experiments presented in Section 6.5. Afterwards, findings about the practicability of the approaches such as the run or training times are discussed in Section 7.4 and possible limitations in Section 7.5. It is concluded with Section 7.6, where an overview of all numerical results is provided.

7.1 Training and Evaluation on Heterogeneous Anomalies

For the experiments described in this section, both methods, PRADA and the LSTM Autoencoder, were trained on both available data sets (power plants `p0` and `p1`) before being applied and evaluated. In this case, the two power plants were considered independently of each other, so training and evaluation were carried out on the data from the same power plant (but separated into training, validation and test sets, as described in Section 6.5). The resulting outcomes represent the main message of the comparative study, as this experiment tries to represent reality as close as possible. Despite the limitations in terms of time and computing resources, the focus was therefore placed on conducting these experiments.

	p0	**p1**
PRADA	1009	241
LSTM Autoencoder	286	26

Table 7.1: Number of hyperparameter combinations considered to find the best-performing model on the validation set

The number of hyperparameter combinations, which were considered in order to find the best model, can be found in Table 7.1. The relatively small number of considered

combinations of the LSTM Autoencoder on the data from power plant `p1` can be attributed to the long runtime, which is discussed in more detail in Section 7.4.

Out of these considered hyperparameter combinations, the model that performed best on the validation set was then selected. The highest-ranked combinations were considered and some combinations were excluded, because they showed irregularities (e.g., high validation F1 but very low training F1, which indicates a poorly generalising model). The F1 value on the test set was then calculated from the selected best hyperparameter combination. For the power plant `p0`, the results are shown in Table 7.2.

	Training				**Validation**			
	F1	**TP**	**FP**	**FN**	**F1**	**TP**	**FP**	**FN**
PRADA	0.9778	44	2	0	0.9099	101	0	20
LSTM Autoencoder	0.9524	40	0	4	0.8649	96	5	25

	Test			
	F1	**TP**	**FP**	**FN**
PRADA	0.9416	121	0	15
LSTM Autoencoder	0.9004	113	2	23

Table 7.2: F1 scores and confusion matrices of the best models for power plant `p0`

It can be seen that both methods generally perform well and detect a majority of the added artificial anomalies: On the test set, PRADA detects eight more anomalies than the LSTM Autoencoder and produces no false positives, whereas the LSTM Autoencoder generates two.

The false negatives were examined more closely: Of the 15 false negatives from PRADA and the 23 from the LSTM Autoencoder, eight overlap, indicating that these anomalies were missed by both methods. Seven anomalies could only be detected by the LSTM Autoencoder, 15 only by PRADA.

Furthermore, there are differences in the detection rates between the three different error types. As can be seen in Figure 7.1, the detection rate for anomalies of type `const` and `rand` is significantly higher than for errors of type `deter` for both approaches.

For the errors of type `deter`, it was found that the undetected errors often either had a short duration or only a slight deterioration: Across all `deter` errors in this test set, the average duration is 5.54 days and on average it is deteriorated to 50.77% of the original value (described by the parameter `deter_base`). However, the ones not detected by PRADA are only deteriorated to 71.54% on average (while having approximately the same average duration). For the LSTM Autoencoder, it seems to be the opposite: With an average deterioration to 57.32% there is only a slight difference, but the false negatives only have an average duration of 4.18 days. The two approaches therefore seem to have different strengths and weaknesses: In this case, PRADA is worse at recognising errors with low deterioration, while the LSTM Autoencoder struggles with shorter anomalies.

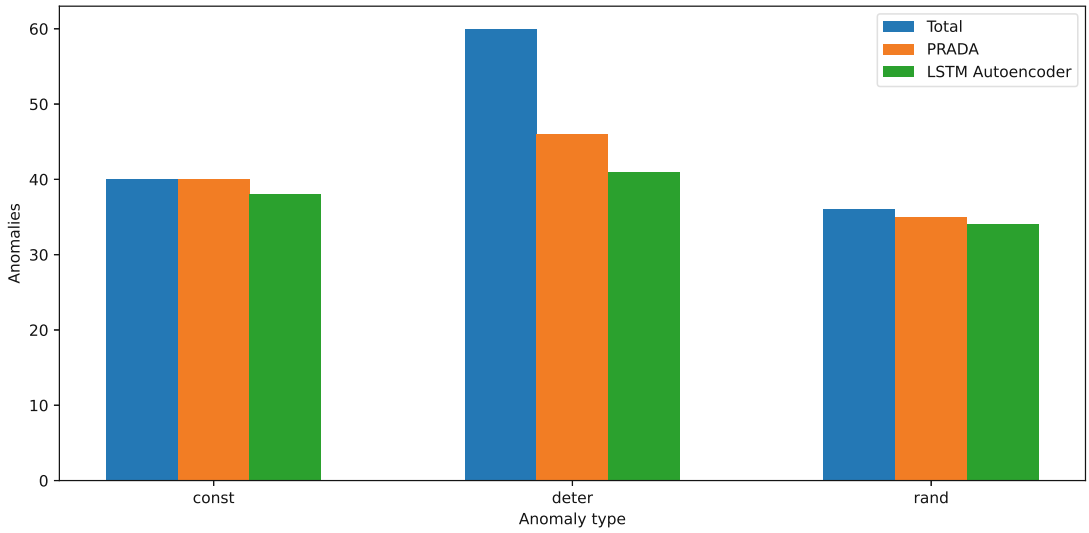

Figure 7.1: Distribution of anomalies in the test data of `p0` and their detection rates

These experiments were repeated with the data from the power plant `p1` and the results are shown in Table 7.3. Regarding the performance, the results are comparable to the ones of `p0`: Both approaches detect the majority of anomalies, again PRADA detects six more than the LSTM Autoencoder on the test set. However, it is noticeable that in this case, PRADA generates more false positives. Regarding the F1 value, the difference between the two methods is smaller than for `p0`, but PRADA can still show a slightly higher value.

	Training				**Validation**			
	F1	**TP**	**FP**	**FN**	**F1**	**TP**	**FP**	**FN**
PRADA	0.9778	44	2	0	0.9886	87	1	1
LSTM Autoencoder	0.9535	41	1	3	0.9708	83	0	5

	Test			
	F1	**TP**	**FP**	**FN**
PRADA	0.9435	117	12	2
LSTM Autoencoder	0.9367	111	7	8

Table 7.3: F1 scores and confusion matrices of the best models for power plant `p1`

Again, the undetected anomalies were examined more closely: In this case, there is one anomaly that is not recognized by either approach, one is only recognized by the LSTM Autoencoder and seven only by PRADA. Figure 7.2 shows the distribution of the three anomaly types and their detection rates for the `p1` dataset. In this case, the detection rate is very high for both methods for all types of anomalies. In some cases, even all anomalies were detected: PRADA detects all anomalies of type `const`, and the LSTM

Autoencoder all of the type `rand`.

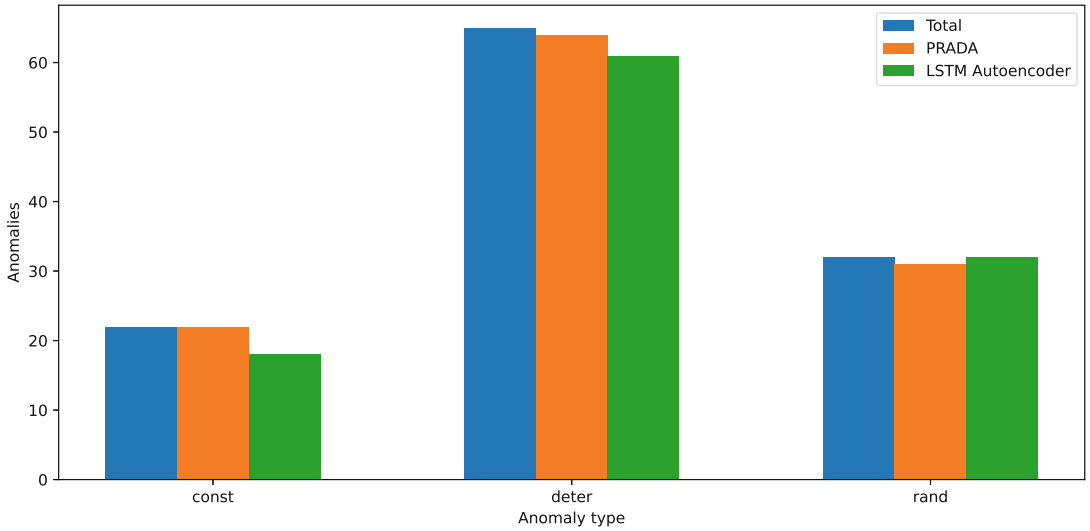

Figure 7.2: Distribution of anomalies in the test data of `p1` and their detection rates

Interestingly, the tendency observed with `p0` that `deter`-type anomalies tend to be detected worse than the other types is much less prominent here. However, the hypothesis that undetected `deter`-type anomalies are characterised by a short duration or low deteriorations can be confirmed: Of the four different false negatives of type `deter`, three have only very weak deteriorations (`deter_base` of 85% or higher), while the fourth is relatively short, with a duration of only 2.78 days. However, the number of false negatives is too small to confirm the statement that PRADA and the LSTM Autoencoder have different strengths and weaknesses in detecting `deter` anomalies.

Due to the unusually high number of false positives compared to other experiments, these were also examined more closely: It was found that the 19 false positives in total extend to only four different day-sensor combinations (remember that the test set is repeated eight times with different anomalies). This indicates that at the affected points in time, there is a behaviour that is not entirely normal, but shows only very slight deviations and no clear signs of an anomaly, and was therefore not recognised during data preprocessing. This is also supported by the fact that it was not recognised in all, but only in some of the eight runs for both methods. In any case, it does not limit the validity of this result, since it affects both PRADA and the LSTM Autoencoder equally.

In summary, PRADA demonstrates higher detection rates and better F1 scores for both power plants, though the margin of improvement over the LSTM Autoencoder is relatively small. There is no clear tendency that one of the two methods is less susceptible to the generation of false positives, but their number is small compared to the number of detected anomalies in any case.

The hyperparameters which were considered best and have therefore been used for the experiments in this section can be seen in Table 7.4.

Method	Hyperparameter	p0	p1
PRADA	R_{day}	OLS	Robust Regression
	t_{day}	-	0.53
	α_{day}	0.1	0.04
	R_{base}	Robust Regression	Robust Regression
	t_{base}	4.89	0.53
	α_{ratio}	0.18	0.04
	$\delta_{\text{expanding}}$	True	True
	n_{lookback}	3	3
	d_{min}	1	1
	α_{anomaly}	0.88	0.98
LSTM Autoencoder	$\delta_{\text{aggregate}}$	`normalised_mean`	`mean`
	n_{layers}	4	5
	n_{units}	128	128
	η	0.00106148	0.00073712
	$n_{\text{timesteps}}$	8	8
	d_{min}	2	1
	α_{anomaly}	1.53	0.009

Table 7.4: Hyperparameters of the best models for both approaches and both power plants

7.2 Evaluation on Homogeneous Anomalies

In contrast to the experiments carried out so far, where all three anomaly types were always present at the same time, data with homogeneous artificially added errors is considered here. This means that all errors added in an experiment are of the same type. This is done for all three types `const`, `deter` and `rand`. This may not correspond as closely to reality, but it can provide further information about which of the two methods is better suited for which situations and where possible weaknesses lie. Due to resource constraints, these experiments were only carried out on the data from the power plant `p0`, using the same hyperparameters as before, presented in Table 7.4.

	F1	TP	FP	FN
PRADA	0.9565	132	0	12
LSTM Autoencoder	0.8425	107	3	37

Table 7.5: F1 scores and confusion matrices for anomalies of type `const` only

Table 7.5 shows the results for anomalies of type `const`. It can be seen clearly that PRADA performs significantly better than the LSTM Autoencoder: It detects 25 more

anomalies without generating any false positives, while the LSTM Autoencoder falsely detects three of those. This is also visible in the F1 score, where the LSTM Autoencoder therefore has a significantly lower value.

A closer look at the false negatives of the two methods shows that they are almost disjoint: Only a single anomaly is not recognised by either of the two approaches. 11 are detected by the LSTM Autoencoder but not by PRADA. On the other hand, there are 36 anomalies that are detected by PRADA but not by the LSTM Autoencoder.

Interesting behaviour can also be observed when considering which value is constantly reported during the anomaly. As can be seen in Figure 7.3, the LSTM Autoencoder is significantly worse at detecting anomalies whose `const_value` is 0. A possible explanation for this is that the LSTM Autoencoder not only considers the behaviour of sensors in relation to each other, but also the fundamental behaviour of the time series. Since it is normal for irradiance data to be constantly 0 during the night, it is more difficult for the LSTM Autoencoder to detect this type of anomaly.

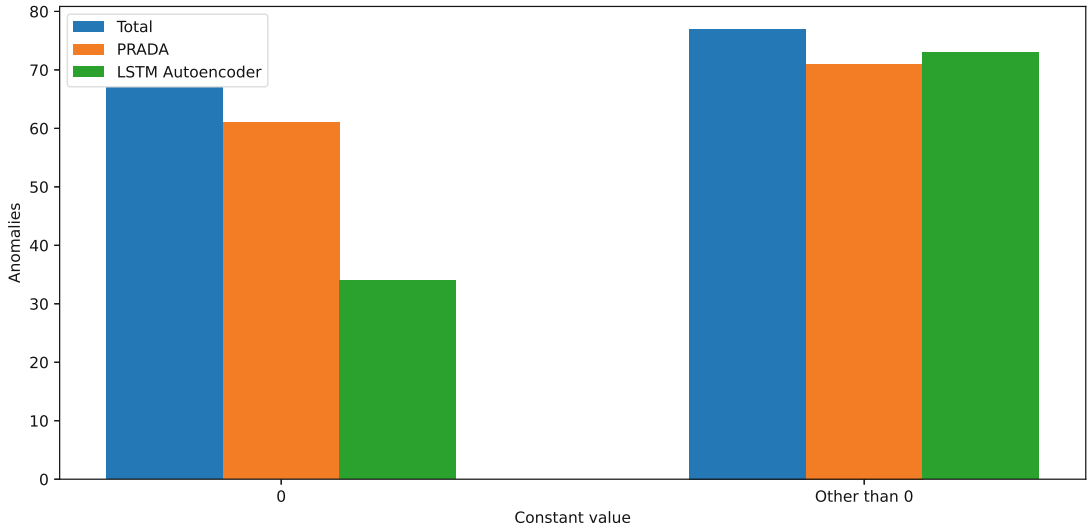

Figure 7.3: Detection rates by constant value for both approaches

Summarising, it can be said that PRADA has a higher detection rate and a better F1 score than the LSTM Autoencoder, and that it does not show any particular weakness for a certain variant of constant anomalies.

The results of the evaluation of a data set containing only anomalies of type `deter` can be found in Table 7.6. In this case, the difference between the two methods is smaller than with `const`: PRADA detects four more anomalies than the LSTM Autoencoder, in addition the LSTM Autoencoder detects five false positives. Considering the F1 score, therefore, PRADA also performs better in this case.

The overlap between the false negatives is relatively small in this case, at only five.

	F1	TP	FP	FN
PRADA	0.9485	129	0	14
LSTM Autoencoder	0.9157	125	5	18

Table 7.6: F1 scores and confusion matrices for anomalies of type `deter` only

However, the false negatives can be used to confirm the theory proposed in Section 7.1 that the two methods have different strengths when it comes to detecting errors of type `deter`: Across all 143 anomalies, the average duration is 4.98 days, and on average, the measured value is deteriorated to 49.44%.

Considering the 14 anomalies not detected by PRADA, the average duration is very similar to that of all anomalies, at 4.76 days, but on average it is only deteriorated to 70.16%. On the other hand, for the LSTM Autoencoder, the 18 undetected anomalies have an average `deter_base` of 46.26%, which represents more deterioration compared to the overall average. However, the undetected anomalies are only 1.86 days long on average.

This confirms that the strength of PRADA lies in detecting short anomalies, but it is prone to not detecting errors with only weak deterioration. For the LSTM Autoencoder, it is the other way around, so the detection of short anomalies of type `deter` is its weakness.

In summary, it can be said that PRADA performs better as it detects more anomalies and at the same time does not generate any false positives. However, the two methods have different strengths for errors of the type `deter`, so that in special situations the LSTM Autoencoder might be preferable.

Finally, a dataset is considered that only contains anomalies of type `rand`. The results of this can be seen in Table 7.7.

	F1	TP	FP	FN
PRADA	0.9591	129	0	11
LSTM Autoencoder	0.9603	133	4	7

Table 7.7: F1 scores and confusion matrices for anomalies of type `rand` only

When only considering anomalies of type `rand`, both methods perform very well, with the LSTM Autoencoder outperforming PRADA by detecting four more anomalies. Although it generates four false positives, it still has a small advantage in terms of the F1 score.

The overlap between the 11 false negatives of PRADA and the seven of the LSTM Autoencoder is very small: Only two errors are not detected by either method. Furthermore, it is noticeable that the undetected anomalies of both methods are characterised by a short duration: Across all 140 anomalies, the average duration is 5.39 days. However, for the undetected errors, the average duration is only 2.43 days for PRADA and 2.13 days for the LSTM Autoencoder. Not a single undetected error has a duration of five days or

longer. Both methods are thus significantly worse at detecting short errors than longer ones.

Summarising across all three error types, PRADA consistently offers good results, detects most of the anomalies, and does not generate false positives. The performance of the LSTM Autoencoder is more inconsistent: For `const` anomalies, it performs significantly worse, for `deter` the difference to PRADA is smaller and it even shows particular strength with one variant of this error. For `rand` it even performs slightly better than PRADA. In all cases, however, it is prone to generating false positives.

7.3 Transfer Learning: Cross-Dataset Evaluation

This experiment is designed to assess whether the optimal hyperparameters of the models depend on the data they are trained on, and if so, to what extent. In practice, this determines whether it makes sense to train and set up a separate model for each power plant or whether this can be done centrally with a single high-quality model.

To examine this, the models from PRADA and the LSTM Autoencoder are used, which performed best on `p0` (Table 7.4, first column). These are then evaluated on the validation and test set of `p1`. As before, the test set serves as the basis for decision-making. The validation set is also considered, because the exact hyperparameter combination that performed best on `p0` was not necessarily also considered in the course of optimising `p1`. By looking at the F1 scores on the validation set, it can therefore be determined whether the hyperparameters would also have been considered as the best hyperparameters for `p1`.

The results of the experiment are shown in Table 7.8.

	Validation				**Test**			
	F1	**TP**	**FP**	**FN**	**F1**	**TP**	**FP**	**FN**
PRADA	0.9708	83	0	5	0.9741	113	0	6
LSTM Autoencoder	0.9136	74	0	14	0.8679	92	1	27

Table 7.8: F1 scores and confusion matrices of the best models for `p0`, evaluated on `p1`

Generally, it can be seen PRADA performs significantly better than the LSTM Autoencoder: On the test set, PRADA detects 21 more anomalies, and the LSTM Autoencoder also creates one false positive.

The overlap between the false negatives of the two approaches is six in this case: This means that all six anomalies that PRADA does not detect are also not detected by the LSTM Autoencoder. Additionally, there are 21 anomalies that are detected by PRADA but not by the LSTM Autoencoder.

Figure 7.4 shows the detection rates of the two methods categorised by error type. It can be seen that for all three error types, PRADA is better at detecting anomalies than the

LSTM Autoencoder. For errors of the type `deter`, the detection rate is the lowest and the difference between the two methods is the highest. Once again, the weaknesses of the two methods in detecting `deter` errors, which have been mentioned repeatedly, are evident: For PRADA, the false positives are characterised by a very low deterioration (on average only to 84.89%), while those of the LSTM Autoencoder are more than two days shorter than the overall average in this experiment.

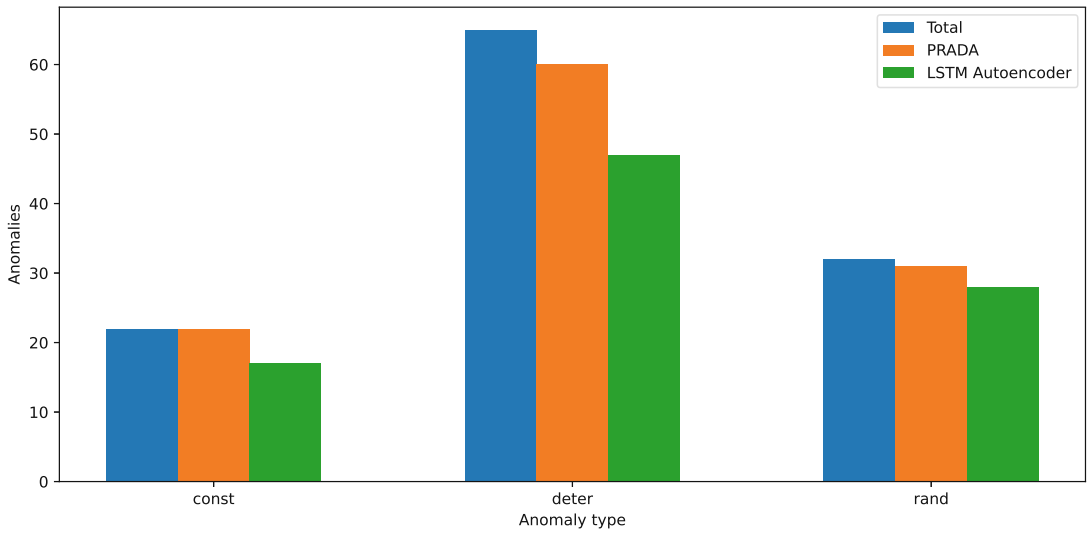

Figure 7.4: Distribution of anomalies in the test data of `p1` and their detection rates using the model trained on `p0`

Finally, it is important to see how the two models compare to those that were trained natively on the `p1` data. Those results were already presented in Section 7.1. For a better overview, they are now shown together with the results from Table 7.8 in Figure 7.5.

It can be seen that on the validation set, both Transfer Learning models perform slightly worse than the native models. This means that the hyperparameters would not have been selected as best models, since there were better performing hyperparameter combinations. However, when looking at the test set, interesting behaviour can be observed: For the LSTM Autoencoder, the Transfer Learning model performs significantly worse than the model, which has been natively trained on this data. For PRADA however, the F1 score of the Transfer Learning model is even higher than of the native one.

The results suggest that the problem at hand is generally well suited for Transfer Learning and that the methods also work well with the hyperparameters or models of other power plants. However, there are also differences between the methods. For example, PRADA is more tolerant of hyperparameter changes and better suited for Transfer Learning than the LSTM Autoencoder. This also showed during optimisation, where there have been many models performing equally good with very different hyperparameters. At this point, however, it should also be noted that in these experiments the Transfer Learning

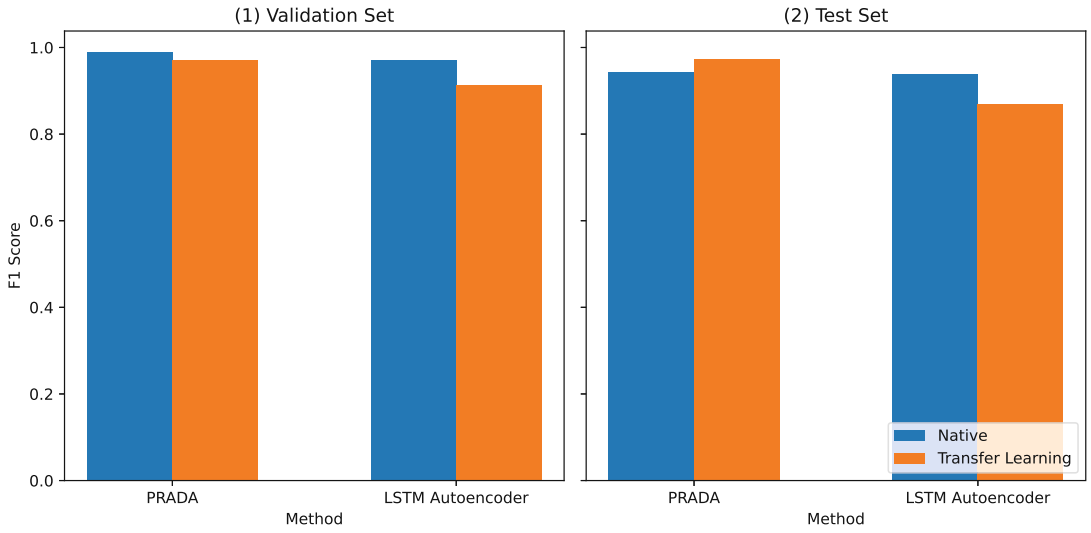

Figure 7.5: F1 scores on validation and test set of native `p1` models compared to Transfer Learning

models were trained only with the data from one power plant. In the case of real-world application of the methods by an operator, a model can be trained using the data of multiple existing power plants, so that the model already has more knowledge about the possible behaviour of different power plants and thus performs better on the new power plant.

7.4 Operational Efficiency and Runtime Evaluation

For a final judgement on which of the two methods is better suited for use in anomaly detection for irradiance data at an operator, not only the detection performance is important, but also the duration of implementing the approach and the detection duration. For this reason, this is discussed in detail in this chapter.

First, the duration required to put one of the two methods into operation is considered. To do this, a large number of hyperparameter combinations must be tested to determine the best one. Table 7.9 indicates the average duration of a run. The specified time refers to both training and evaluating of a hyperparameter combination. The number of runs performed in this work has already been presented in Table 7.1.

	p0	**p1**
PRADA	00:16:00	01:12:20
LSTM Autoencoder (GPU)	00:57:04	06:51:14

Table 7.9: Average duration of one optimisation run

It can be seen that PRADA is significantly faster than the LSTM Autoencoder for both power plants: The LSTM Autoencoder takes almost four times as long for `p0` and almost six times as long for `p1`. The GPUs presented in Section 6.6 were used for all runs of the LSTM Autoencoder. Training with the CPU is much slower and was rejected after some preliminary attempts. For training and evaluation with PRADA, generally only the CPU is used.

When fixing one of the two methods and looking at the difference between `p0` and `p1`, PRADA takes about 4.5 times longer, while the run time of the LSTM Autoencoder increases by a factor greater than seven. However, these results are consistent with the differences in size between the data sets: `p0` has two orientations of five sensors each, which results in 20 pairwise comparisons, while `p1` has 18 sensors after data preprocessing, resulting in 153 pairwise comparisons.

In conclusion, while all run times remain within a manageable range, they should still be considered carefully when planning to implement either method in a power plant.

After the successful implementation of one of the two methods, these are used daily to validate the measured data from the previous day and to detect anomalies. Also here, the runtime is relevant, especially if these calculations are not carried out centrally in a data center of the operator, but possibly locally with low-performance hardware directly at the power plant. For this reason, this was also tested when using the CPU instead of the GPU. Table 7.10 shows the average duration of validating the data of one day in seconds.

	p0	**p1**
PRADA	1.89	26.73
LSTM Autoencoder (GPU)	2.31	20.41
LSTM Autoencoder (CPU)	9.64	93.00

Table 7.10: Average duration (in seconds) for anomaly detection per sensor and day

When looking at the run times for validating the data of one day for the `p0` power plant, it is noticeable that PRADA is the fastest method. However, when the LSTM Autoencoder is run on a GPU, it is only about 20% slower. For a power plant with more sensors, like `p1`, the situation is reversed, with the LSTM Autoencoder being 30% faster. This can be explained by the fact that using a GPU introduces some overhead, which only pays off with larger datasets. If the LSTM Autoencoder is used without a GPU, the computation time is significantly higher compared to both PRADA and the LSTM Autoencoder with a GPU. However, since the run times, even in the worst-case scenario, are still in the range of minutes, this should not pose a significant issue in any practical application.

7.5 Limitations

Despite the contributions of this work to the field of anomaly detection in PV power plants and irradiance data, a number of limitations must be acknowledged. First, the evaluation was conducted using artificial anomalies. Those were designed to mimic real-world situations and are based on multiple reports of true anomalies, but it is not guaranteed that they perfectly represent real operational anomalies. Another limitation relates to the selection of hyperparameters: In this study, hyperparameters were optimized based on the F1 score, but the influence of different hyperparameter configurations on the performance of the model was not deeply explored.

Additionally, both regression techniques considered for PRADA, OLS and Robust Regression, currently do not account for temporal dependencies within the data of one day. While PRADA can detect even small discrepancies between the measurements of two sensors, completely atypical behaviour of the irradiance data within one day could go unnoticed if both sensors show it. Another critical point of PRADA is the reliance on fixed intervals for evaluating whether the regression coefficient is within the normal range, without incorporating statistical tests or more dynamic approaches. This lack of flexibility could affect the ability to adapt to changes that occur over time.

These current limitations highlight areas where improvements may be necessary. Addressing these issues in future research could significantly improve the applicability of PRADA and the LSTM Autoencoder in the situation considered in this work as well as similar scenarios.

7.6 Overview of Experimental Results

All relevant results described in more detail in the sections above are summarized again in Table 7.11. The first two columns show the power plant used for training and evaluation of the model, while the third column specifies the configuration of artificial anomalies on which the model was tested.

Optimized on	Tested on	Anomaly setup	PRADA	LSTM Autoencoder
p0	p0	Heterogeneous	0.9416	0.9004
p1	p1	Heterogeneous	0.9435	0.9367
p0	p0	Homogeneous const	0.9565	0.8425
p0	p0	Homogeneous deter	0.9485	0.9157
p0	p0	Homogeneous rand	0.9591	0.9603
p0	p1	Heterogeneous	0.9741	0.8679

Table 7.11: Overview of the results from all evaluations conducted on test sets

CHAPTER 8

Conclusion and Outlook

The work presented in this thesis addresses the issue of anomaly detection in redundant sensor systems, with irradiance measurements from photovoltaic power plants as an example use case. This is becoming more and more necessary due to the constantly growing power generation from photovoltaics, both to avoid loss of income for the operator due to undetected or late detection of faults, as well as to ensure a stable power supply in the future. This thesis introduced a new method, PRADA, and conducted a comparative study with a second method, an LSTM Autoencoder, which is a deep-learning based method and proven in the field of anomaly detection.

To conduct the comparative study, measurement data from two power plants in Europe was used: One in Austria, which features sensors with two different orientations and has a high data availability of over 1.5 years, the other one in Spain, which has more sensors of only one orientation, but only slightly more than a year of data. The available data contains a few irregularities that indicate anomalies, but clearly not enough to evaluate the detection performance of the two approaches.

To overcome this problem, a framework making use of artificial anomalies was developed: The data was first cleaned to meet the assumption of being free of anomalies. Then, artificial anomalies of the types `const`, `deter` and `rand` were added. They represent different real-world problems and are based on the few irregularities originally present in the data as well as on reports from literature and domain experts. With the addition of artificial errors, detailed information (such as exact start and end dates) is available for each error, transforming the problem into a supervised one.

The optimisation framework Optuna was used to find the best possible hyperparameters. The F1 score is used to determine the best model, and the data was divided into training, validation, and test sets. To boost data availability and allow for more variation between different models, the validation and test sets were used multiple times with different artificial anomalies. This is possible since the model never learns any information from

those. The best model was then selected on the validation set and evaluated on the test set for further comparisons.

In terms of performance metrics, PRADA showed a higher F1 score than the LSTM Autoencoder in five out of six scenarios. It demonstrated consistent performance with an F1 score between 0.94 and 0.975 across all considered scenarios. The LSTM Autoencoder was significantly less consistent with an F1 score between 0.84 and 0.97. Only in the case where only anomalies of type `rand` are considered, the LSTM Autoencoder was able to detect more anomalies than PRADA. Furthermore, experiments with Transfer Learning were carried out, i.e. models were applied to data from power plants that were completely new to them and may show different behaviour. Also in this case PRADA performed significantly better: The transferred model was even able to outperform the native one. For the LSTM Autoencoder, the transferred model could not keep up with the performance of the native model, which supports the finding that the LSTM Autoencoder places more emphasis on the behaviour of the time series itself rather than on the comparison among the redundant sensors.

The empirical evaluation also showed that overall PRADA is less susceptible to producing false positives, making it more reliable in environments where false alarms could result in unnecessary costs or interventions. The runtimes required for training and applying the approaches were also examined. Finding the optimal hyperparameters is significantly faster with PRADA and no GPU is required, while the training of the LSTM Autoencoder requires one to achieve justifiable run times. When it comes to applying the methods, PRADA is on the same order of magnitude as the LSTM Autoencoder when a GPU is used. An application of the LSTM Autoencoder without a GPU, for example directly at the power plant location, is possible, but it takes significantly longer. This means that PRADA can be used more flexibly, possibly also in less conventional settings such as edge-based systems.

In summary, PRADA has proven to be the superior method in the scenario of this thesis, as it delivers consistently good results and also has advantages in terms of applicability. However, in some situations the LSTM Autoencoder performed better and was also able to highlight certain strengths, so it should also be considered in a real application.

While the methods presented in this thesis offer valuable insights to the field of anomaly detection in redundant sensor systems, there are still several promising avenues for future research. One of the most important next steps would be to obtain more data containing real-world anomalies. These can then be used, on the one hand, to test PRADA and the LSTM Autoencoder in a completely real-world scenario. On the other hand, the findings about the additional anomalies can be used to refine the definition of the artificial anomalies and repeat the experiments of this work with them.

Another key area for future research is a comprehensive sensitivity analysis of the hyperparameters of both methods. In this work, the selection of hyperparameters was based on optimizing the F1 score. Exploring how different hyperparameter choices affect the performance could provide valuable insights. This could go hand in hand with

expanding the regression methodology of PRADA, which currently does not consider the temporal relationships between individual measurements. Using regression techniques which take these temporal dependencies into account could improve the performance of PRADA by placing more focus on the behaviour of the time series itself, not only the pairwise comparison. Another possibility for improvement would be to use a statistical test to determine whether the coefficient significantly deviates from 1, instead of using a fixed interval. This could provide more robustness and, at the same time, reduce the number of hyperparameters to be optimised, which facilitates finding the best model.

To make both methods even more applicable in real-world scenarios, further work could be done to develop a measure of the certainty of an anomaly. PRADA already calculates an error probability and the LSTM Autoencoder an average reconstruction error. At the moment, however, these are only compared with a threshold and not used further. In the future, they could be used to calculate a degree of certainty that an error is occurring. This could allow for more adaptive and context-aware detection systems and thus help operators to decide whether an intervention or a more detailed investigation is appropriate.

In summary, the methods introduced and experiments conducted in this thesis represent significant progress in anomaly detection in redundant sensor systems. While the novel method PRADA generally outperforms the LSTM Autoencoder, both methods demonstrate strengths in different areas, and future research should aim to further refine these approaches, particularly by incorporating real-world data and more adaptive detection mechanisms. This will help enhance the practical implementation of these methods and improve their reliability and robustness.

List of Figures

List of Tables

References

[1] Felix Creutzig et al. „The underestimated potential of solar energy to mitigate climate change“. en. In: *Nature Energy* 2.9 (Aug. 2017), p. 17140. ISSN: 2058-7546. DOI: 10.1038/nenergy.2017.140. URL: https://www.nature.com/articles/nenergy2017140 (accessed on Sept. 5, 2024).

[2] Bundesministerium für Klimaschutz, Umwelt, Energie, Mobilität, Innovation und Technologie (BMK). *Weiterhin Boom bei Photovoltaik und hohe Verkaufszahlen bei Wärmepumpen*. de. 2024. URL: https://energie.gv.at/erneuerbare-energie/energiewende-schreitet-voran (accessed on Sept. 5, 2024).

[3] Solarserver. *Solarstrom weltweit: Knackt die Photovoltaik 2024 die 2-TW-Marke?* de. June 2024. URL: https://www.solarserver.de/2024/06/18/solarstrom-weltweit-knackt-die-photovoltaik-2024-die-2-tw-marke/ (accessed on Sept. 5, 2024).

[4] International Renewable Energy Agency. *Renewable Power Generation Costs in 2022*. en. Tech. rep. Abu Dhabi, 2023, p. 208. URL: https://www.irena.org/-/media/Files/IRENA/Agency/Publication/2023/Aug/IRENA_Renewable_power_generation_costs_in_2022.pdf (accessed on Sept. 6, 2024).

[5] Marcel Suri et al. *Global Photovoltaic Power Potential by Country*. en. Tech. rep. Washington, D.C.: World Bank Group, Energy Sector Management Assistance Program (ESMAP), 2020. URL: http://documents.worldbank.org/curated/en/466331592817725242/Global-Photovoltaic-Power-Potential-by-Country.

[6] Fraunhofer Institute for Solar Energy Systems, ISE. *Photovoltaics Report*. en. Tech. rep. Freiburg, July 2024. URL: https://www.ise.fraunhofer.de/en/publications/studies/photovoltaics-report.html (accessed on Sept. 12, 2024).

[7] Sunil Prasad Lohani and Andrew Blakers. „100% renewable energy with pumped-hydro-energy storage in Nepal“. en. In: *Clean Energy* 5.2 (June 2021), pp. 243–253. ISSN: 2515-4230, 2515-396X. DOI: 10.1093/ce/zkab011. URL: https://academic.oup.com/ce/article/5/2/243/6275217 (accessed on Sept. 6, 2024).

[8] Eurostat. *Electricity & gas hit record prices in 2022*. en. Apr. 2023. URL: https://ec.europa.eu/eurostat/web/products-eurostat-news/w/ddn-20230426-2 (accessed on Oct. 8, 2024).

[9] Yaman Abou Jieb and Eklas Hossain. *Photovoltaic Systems: Fundamentals and Applications*. en. Cham: Springer International Publishing, 2022. ISBN: 978-3-030-89779-6. DOI: 10.1007/978-3-030-89780-2. URL: https://link.springer.com/10.1007/978-3-030-89780-2 (accessed on Oct. 8, 2024).

[10] Augustin Joseph McEvoy, T. Markvart, and Luis Castañer. *Practical Handbook of Photovoltaics: Fundamentals and Applications*. en. 2nd ed. Waltham, MA: Academic Press, 2012. ISBN: 978-0-12-385934-1.

[11] Siva Ramakrishna Madeti and S.N. Singh. „Monitoring system for photovoltaic plants: A review". en. In: *Renewable and Sustainable Energy Reviews* 67 (Jan. 2017), pp. 1180–1207. ISSN: 13640321. DOI: 10.1016/j.rser.2016.09.088. URL: https://doi.org/10.1016/j.rser.2016.09.088 (accessed on Sept. 9, 2024).

[12] Hukseflux Thermal Sensors. *How many monitoring systems on a PV solar power plant?* en. Version 2403. 2024. URL: https://www.hukseflux.com/applications/solar-energy-pv-system-performance-monitoring/how-many-monitoring-systems-on-a-pv-power-plant (accessed on Oct. 8, 2024).

[13] H.D. Nguyen et al. „Forecasting and Anomaly Detection approaches using LSTM and LSTM Autoencoder techniques with the applications in supply chain management". en. In: *International Journal of Information Management* 57 (Apr. 2021), p. 102282. ISSN: 02684012. DOI: 10.1016/j.ijinfomgt.2020.102282. URL: https://linkinghub.elsevier.com/retrieve/pii/S026840122031481X (accessed on Sept. 3, 2024).

[14] Zhaomin Chen et al. „Autoencoder-based network anomaly detection". en. In: *2018 Wireless Telecommunications Symposium (WTS)*. Phoenix, AZ: IEEE, Apr. 2018, pp. 1–5. ISBN: 978-1-5386-3395-3. DOI: 10.1109/WTS.2018.8363930. URL: https://ieeexplore.ieee.org/document/8363930/ (accessed on Oct. 8, 2024).

[15] Yuanyuan Wei et al. „LSTM-Autoencoder-Based Anomaly Detection for Indoor Air Quality Time-Series Data". en. In: *IEEE Sensors Journal* 23.4 (Feb. 2023), pp. 3787–3800. ISSN: 1530-437X, 1558-1748, 2379-9153. DOI: 10.1109/JSEN.2022.3230361. URL: https://ieeexplore.ieee.org/document/10011213/ (accessed on Sept. 3, 2024).

[16] Javed Ashraf et al. „Novel Deep Learning-Enabled LSTM Autoencoder Architecture for Discovering Anomalous Events From Intelligent Transportation Systems". en. In: *IEEE Transactions on Intelligent Transportation Systems* 22.7 (July 2021), pp. 4507–4518. ISSN: 1524-9050, 1558-0016. DOI: 10.1109/TITS.2020.3017882.

URL: https://ieeexplore.ieee.org/document/9198908/ (accessed on Sept. 10, 2024).

[17] Greg Van Houdt, Carlos Mosquera, and Gonzalo Nápoles. „A review on the long short-term memory model“. en. In: *Artificial Intelligence Review* 53.8 (Dec. 2020), pp. 5929–5955. ISSN: 0269-2821, 1573-7462. DOI: 10.1007/s10462-020-09838-1. URL: https://link.springer.com/10.1007/s10462-020-09838-1 (accessed on Oct. 8, 2024).

[18] Dirk C Jordan, Bill Sekulic, and Sarah Kurtz. „Impact and Detection of Pyranometer Failure on PV Performance“. en. In: Golden CO, Feb. 2013. URL: https://www.energy.gov/eere/solar/articles/impact-and-detection-pyranometer-failure-pv-performance (accessed on Aug. 22, 2024).

[19] Leonardo Micheli, Michael G. Deceglie, and Matthew Muller. „Predicting photovoltaic soiling losses using environmental parameters: An update“. en. In: *Progress in Photovoltaics: Research and Applications* 27.3 (Mar. 2019), pp. 210–219. ISSN: 1062-7995, 1099-159X. DOI: 10.1002/pip.3079. URL: https://onlinelibrary.wiley.com/doi/10.1002/pip.3079 (accessed on Aug. 22, 2024).

[20] D. Feuermann and A. Zemel. „Dust-induced degradation of pyranometer sensitivity“. en. In: *Solar Energy* 50.6 (June 1993), pp. 483–486. ISSN: 0038092X. DOI: 10.1016/0038-092X(93)90109-2. URL: https://linkinghub.elsevier.com/retrieve/pii/0038092X93901092 (accessed on Aug. 22, 2024).

[21] Justyna Pastuszak and Paweł Węgierek. „Photovoltaic Cell Generations and Current Research Directions for Their Development“. en. In: *Materials* 15.16 (Aug. 2022), p. 5542. ISSN: 1996-1944. DOI: 10.3390/ma15165542. URL: https://www.mdpi.com/1996-1944/15/16/5542 (accessed on Oct. 8, 2024).

[22] European Bank for Reconstruction and Development. *Benban, Africa's largest solar park, completed.* en. Oct. 2019. URL: https://www.ebrd.com/news/video/benban-africas-largest-solar-park-completed.html (accessed on Sept. 6, 2024).

[23] Vincent Shaw. *World's largest solar plant goes online in China.* en. June 2024. URL: https://www.pv-magazine.com/2024/06/06/worlds-largest-solar-plant-goes-online-in-china-2/ (accessed on Sept. 6, 2024).

[24] Mark Bolinger et al. *Utility-Scale Solar, 2023 Edition: Empirical Trends in Deployment, Technology, Cost, Performance, PPA Pricing, and Value in the United States.* en. Tech. rep. Lawrence Berkeley National Laboratory, Oct. 2023. URL: https://emp.lbl.gov/sites/default/files/utility_scale_solar_2023_edition_slides.pdf (accessed on Sept. 12, 2024).

[25] International Energy Agency. *Solar PV.* en. 2023. URL: https://www.iea.org/energy-system/renewables/solar-pv (accessed on Sept. 12, 2024).

[26] Weidong Xiao. *Photovoltaic Power System: Modelling, Design and Control*. en. 1st ed. Wiley, July 2017. ISBN: 978-1-119-28034-7. DOI: 10.1002/9781119280408. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/9781119280408 (accessed on Sept. 12, 2024).

[27] A. Mellit, G.M. Tina, and S.A. Kalogirou. „Fault detection and diagnosis methods for photovoltaic systems: A review“. en. In: *Renewable and Sustainable Energy Reviews* 91 (Aug. 2018), pp. 1–17. ISSN: 13640321. DOI: 10.1016/j.rser.2018.03.062. URL: https://linkinghub.elsevier.com/retrieve/pii/S1364032118301370 (accessed on Oct. 8, 2024).

[28] Inverter Service BV. *HUAWEI-error codes*. en. URL: https://omvormerservice.be/index.html/en/error_code_huawei-en/ (accessed on Oct. 8, 2024).

[29] International Electrotechnical Commission. *IEV number 845-21-053: "irradiance"*. Dec. 2020. URL: https://www.electropedia.org/iev/iev.nsf/display?openform&ievref=845-21-053 (accessed on Aug. 4, 2024).

[30] M.J. Rivera Aguilar and C. Reise. „Silicon Sensors vs. Pyranometers – Review of Deviations and Conversion of Measured Values“. en. In: *37th European Photovoltaic Solar Energy Conference and Exhibition; 1449-1454* (2020), 6 pages, 7726 kb. DOI: 10.4229/EUPVSEC20202020-5BV.3.3. URL: https://userarea.eupvsec.org/proceedings/EU-PVSEC-2020/5BV.3.3/ (accessed on Aug. 4, 2024).

[31] Hukseflux Thermal Sensors. *SR20 pyranometer 1.jpg*. Nov. 2012. URL: https://commons.wikimedia.org/wiki/File:SR20_pyranometer_1.jpg (accessed on Sept. 7, 2024).

[32] Marc A. N. Korevaar. „Measuring Solar Irradiance for Photovoltaics“. en. In: *Solar Radiation - Measurement, Modeling and Forecasting Techniques for Photovoltaic Solar Energy Applications*. IntechOpen, Aug. 2022. ISBN: 978-1-83968-859-1. DOI: 10.5772/intechopen.105580. URL: https://www.intechopen.com/chapters/82468 (accessed on Aug. 21, 2024).

[33] Michael Gostein et al. „Evaluating a model to estimate GHI, DNI, & DHI from POA irradiance“. en. In: *2016 IEEE 43rd Photovoltaic Specialists Conference (PVSC)*. June 2016, pp. 0943–0946. DOI: 10.1109/PVSC.2016.7749749. URL: https://ieeexplore.ieee.org/abstract/document/7749749 (accessed on Aug. 4, 2024).

[34] Arthurcala. *Photovoltaic pyranometer on POA.jpg*. Mar. 2021. URL: https://commons.wikimedia.org/wiki/File:Photovoltaic_pyranometer_on_POA.jpg (accessed on Sept. 7, 2024).

[35] Solargis APAC Pte. Ltd. *On-Site Measurements in Large-Scale Solar Projects*. en. Mar. 2022. URL: https://solargis.com/resources/blog/best-practices/growing-pain-3-on-site-measurements-in-large-scale-solar (accessed on Sept. 7, 2024).

[36] Varun Chandola, Arindam Banerjee, and Vipin Kumar. „Anomaly detection: A survey“. en. In: *ACM Computing Surveys* 41.3 (July 2009), 15:1–15:58. ISSN: 0360-0300. DOI: 10.1145/1541880.1541882. URL: https://doi.org/10.1145/1541880.1541882 (accessed on Feb. 27, 2024).

[37] Guansong Pang et al. „Deep Learning for Anomaly Detection: A Review“. en. In: *ACM Computing Surveys* 54.2 (Mar. 2022), pp. 1–38. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3439950. URL: https://dl.acm.org/doi/10.1145/3439950 (accessed on Oct. 9, 2024).

[38] Tharindu Fernando et al. „Deep Learning for Medical Anomaly Detection – A Survey“. en. In: *ACM Computing Surveys* 54.7 (Sept. 2022), pp. 1–37. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3464423. URL: https://dl.acm.org/doi/10.1145/3464423 (accessed on Oct. 9, 2024).

[39] Imtiaz Ullah and Qusay H. Mahmoud. „Design and Development of a Deep Learning-Based Model for Anomaly Detection in IoT Networks“. In: *IEEE Access* 9 (2021), pp. 103906–103926. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3094024. URL: https://ieeexplore.ieee.org/document/9469914/ (accessed on Oct. 9, 2024).

[40] Nicolas Goix. *How to Evaluate the Quality of Unsupervised Anomaly Detection Algorithms?* en. arXiv:1607.01152 [cs, stat]. July 2016. DOI: 10.48550/arXiv.1607.01152. URL: http://arxiv.org/abs/1607.01152 (accessed on Feb. 26, 2024).

[41] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. „Anomaly detection in time series: a comprehensive evaluation“. en. In: *Proceedings of the VLDB Endowment* 15.9 (May 2022), pp. 1779–1797. ISSN: 2150-8097. DOI: 10.14778/3538598.3538602. URL: https://dl.acm.org/doi/10.14778/3538598.3538602 (accessed on Oct. 9, 2024).

[42] Kukjin Choi et al. „Deep Learning for Anomaly Detection in Time-Series Data: Review, Analysis, and Guidelines“. en. In: *IEEE Access* 9 (2021), pp. 120043–120065. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3107975. URL: https://ieeexplore.ieee.org/document/9523565/ (accessed on Oct. 9, 2024).

[43] Julien Audibert et al. „USAD: UnSupervised Anomaly Detection on Multivariate Time Series“. en. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Virtual Event CA USA: ACM, Aug. 2020, pp. 3395–3404. ISBN: 978-1-4503-7998-4. DOI: 10.1145/3394486.3403392. URL: https://dl.acm.org/doi/10.1145/3394486.3403392 (accessed on Oct. 9, 2024).

[44] Renjie Wu and Eamonn Keogh. „Current Time Series Anomaly Detection Benchmarks are Flawed and are Creating the Illusion of Progress“. en. In: *IEEE Transactions on Knowledge and Data Engineering* (2021), pp. 1–1. ISSN: 1041-4347, 1558-2191, 2326-3865. DOI: 10.1109/TKDE.2021.3112126. URL: https://ieeexplore.ieee.org/document/9537291/ (accessed on Oct. 9, 2024).

[45] L. Erhan et al. „Smart anomaly detection in sensor systems: A multi-perspective review“. en. In: *Information Fusion* 67 (Mar. 2021), pp. 64–79. ISSN: 15662535. DOI: 10.1016/j.inffus.2020.10.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S1566253520303717 (accessed on Oct. 9, 2024).

[46] Ling Xiang et al. „Condition monitoring and anomaly detection of wind turbine based on cascaded and bidirectional deep learning networks“. en. In: *Applied Energy* 305 (Jan. 2022), p. 117925. ISSN: 03062619. DOI: 10.1016/j.apenergy.2021.117925. URL: https://linkinghub.elsevier.com/retrieve/pii/S0306261921012368 (accessed on Oct. 9, 2024).

[47] Xiangyu Li et al. „Research on anomaly detection method of nuclear power plant operation state based on unsupervised deep generative model“. en. In: *Annals of Nuclear Energy* 167 (Mar. 2022), p. 108785. ISSN: 03064549. DOI: 10.1016/j.anucene.2021.108785. URL: https://linkinghub.elsevier.com/retrieve/pii/S0306454921006629 (accessed on Oct. 9, 2024).

[48] Ruben Urraca et al. „Quality control of global solar radiation data with satellite-based products“. en. In: *Solar Energy* 158 (Dec. 2017), pp. 49–62. ISSN: 0038-092X. DOI: 10.1016/j.solener.2017.09.032. URL: https://www.sciencedirect.com/science/article/pii/S0038092X17308046 (accessed on Feb. 13, 2024).

[49] Ruben Urraca, Andres Sanz-Garcia, and Iñigo Sanz-Garcia. „BQC: A free web service to quality control solar irradiance measurements across Europe“. en. In: *Solar Energy* 211 (Nov. 2020), pp. 1–10. ISSN: 0038-092X. DOI: 10.1016/j.solener.2020.09.055. URL: https://www.sciencedirect.com/science/article/pii/S0038092X20310082 (accessed on Feb. 13, 2024).

[50] Sascha Lindig et al. „Outdoor PV System Monitoring—Input Data Quality, Data Imputation and Filtering Approaches“. en. In: *Energies* 13.19 (Jan. 2020), p. 5099. ISSN: 1996-1073. DOI: 10.3390/en13195099. URL: https://www.mdpi.com/1996-1073/13/19/5099 (accessed on Feb. 15, 2024).

[51] Victor Hugo Wentz et al. „Solar Irradiance Forecasting to Short-Term PV Power: Accuracy Comparison of ANN and LSTM Models“. en. In: *Energies* 15.7 (Mar. 2022), p. 2457. ISSN: 1996-1073. DOI: 10.3390/en15072457. URL: https://www.mdpi.com/1996-1073/15/7/2457 (accessed on Oct. 9, 2024).

[52] Yunjun Yu, Junfei Cao, and Jianyong Zhu. „An LSTM Short-Term Solar Irradiance Forecasting Under Complicated Weather Conditions“. en. In: *IEEE Access* 7 (2019), pp. 145651–145666. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2946057. URL: https://ieeexplore.ieee.org/document/8864021/ (accessed on Oct. 9, 2024).

[53] Oleksandr I. Provotar, Yaroslav M. Linder, and Maksym M. Veres. „Unsupervised Anomaly Detection in Time Series Using LSTM-Based Autoencoders“. en. In: *2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT)*. Dec. 2019, pp. 513–517. DOI: 10.1109/ATIT49449.2019.9030505. URL: https://ieeexplore.ieee.org/abstract/document/9030505 (accessed on Feb. 17, 2024).

[54] Stefano Longari et al. „CANnolo: An Anomaly Detection System Based on LSTM Autoencoders for Controller Area Network“. en. In: *IEEE Transactions on Network and Service Management* 18.2 (June 2021), pp. 1913–1924. ISSN: 1932-4537. DOI: 10.1109/TNSM.2020.3038991. URL: https://ieeexplore.ieee.org/abstract/document/9262960 (accessed on Sept. 10, 2024).

[55] Hong-Soon Nam, Youn-Kwae Jeong, and Jong Won Park. „An Anomaly Detection Scheme based on LSTM Autoencoder for Energy Management“. en. In: *2020 International Conference on Information and Communication Technology Convergence (ICTC)*. ISSN: 2162-1233. Oct. 2020, pp. 1445–1447. DOI: 10.1109/ICTC49870.2020.9289226. URL: https://ieeexplore.ieee.org/abstract/document/9289226 (accessed on Sept. 10, 2024).

[56] Jaeyong Kang et al. „Anomaly Detection of the Brake Operating Unit on Metro Vehicles Using a One-Class LSTM Autoencoder“. en. In: *Applied Sciences* 11.19 (Jan. 2021), p. 9290. ISSN: 2076-3417. DOI: 10.3390/app11199290. URL: https://www.mdpi.com/2076-3417/11/19/9290 (accessed on Sept. 10, 2024).

[57] Yuxin Zhang et al. „Unsupervised Deep Anomaly Detection for Multi-Sensor Time-Series Signals“. de. In: *IEEE Transactions on Knowledge and Data Engineering* (2021), pp. 1–1. ISSN: 1041-4347, 1558-2191, 2326-3865. DOI: 10.1109/TKDE.2021.3102110. URL: https://ieeexplore.ieee.org/document/9507359/ (accessed on Oct. 9, 2024).

[58] S.C. Lee. „Sensor value validation based on systematic exploration of the sensor redundancy for fault diagnosis KBS“. en. In: *IEEE Transactions on Systems, Man, and Cybernetics* 24.4 (Apr. 1994), pp. 594–605. ISSN: 00189472. DOI: 10.1109/21.286380. URL: http://ieeexplore.ieee.org/document/286380/ (accessed on Sept. 10, 2024).

[59] J. Frolik, M. Abdelrahman, and P. Kandasamy. „A confidence-based approach to the self-validation, fusion and reconstruction of quasi-redundant sensor data“. en. In: *IEEE Transactions on Instrumentation and Measurement* 50.6 (Dec. 2001), pp. 1761–1769. ISSN: 00189456. DOI: 10.1109/19.982977. URL: http://ieeexplore.ieee.org/document/982977/ (accessed on Sept. 10, 2024).

[60] Mark Rafferty et al. „Local Anomaly Detection by Application of Regression Analysis on PMU Data“. en. In: *2018 IEEE Power & Energy Society General Meeting (PESGM)*. Portland, OR: IEEE, Aug. 2018, pp. 1–5. ISBN: 978-1-5386-7703-2. DOI: 10.1109/PESGM.2018.8586320. URL: https://ieeexplore.ieee.org/document/8586320/ (accessed on Sept. 10, 2024).

[61] Charles N. Long and Ellsworth G. Dutton. „BSRN Global Network recommended QC tests, V2.x“. en. In: (2010). URL: https://epic.awi.de/id/eprint/30083/1/BSRN_recommended_QC_tests_V2.pdf (accessed on Feb. 19, 2024).

[62] C. N. Long and Y. Shi. „An Automated Quality Assessment and Control Algorithm for Surface Radiation Measurements“. en. In: *The Open Atmospheric Science Journal* 2.1 (Apr. 2008). URL: https://benthamopen.com/ABSTRACT/TOASCJ-2-23 (accessed on Feb. 17, 2024).

[63] S. Younes, R. Claywell, and T. Muneer. „Quality control of solar radiation data: Present status and proposed new approaches“. en. In: *Energy*. Measurement and Modelling of Solar Radiation and Daylight- Challenges for the 21st Century 30.9 (July 2005), pp. 1533–1549. ISSN: 0360-5442. DOI: 10.1016/j.energy.2004.04.031. URL: https://www.sciencedirect.com/science/article/pii/S0360544204002233 (accessed on Feb. 14, 2024).

[64] Dazhi Yang, Gokhan Mert Yagli, and Hao Quan. „Quality Control for Solar Irradiance Data“. en. In: *2018 IEEE Innovative Smart Grid Technologies - Asia (ISGT Asia)*. May 2018, pp. 208–213. DOI: 10.1109/ISGT-Asia.2018.8467892. URL: https://ieeexplore.ieee.org/abstract/document/8467892/authors#authors (accessed on Feb. 13, 2024).

[65] Daniel Perez-Astudillo, Dunia Bachour, and Luis Martin-Pomares. „Improved quality control protocols on solar radiation measurements“. en. In: *Solar Energy* 169 (July 2018), pp. 425–433. ISSN: 0038-092X. DOI: 10.1016/j.solener.2018.05.028. URL: https://www.sciencedirect.com/science/article/pii/S0038092X18304614 (accessed on Feb. 15, 2024).

[66] Michel Journée and Cédric Bertrand. „Quality control of solar radiation data within the RMIB solar measurements network“. en. In: *Solar Energy* 85.1 (Jan. 2011), pp. 72–86. ISSN: 0038-092X. DOI: 10.1016/j.solener.2010.10.021. URL: https://www.sciencedirect.com/science/article/pii/S0038092X10003270 (accessed on Feb. 14, 2024).

[67] Ying-Yi Hong and Rolando A. Pula. „Methods of photovoltaic fault detection and classification: A review“. en. In: *Energy Reports* 8 (Nov. 2022), pp. 5898–5929. ISSN: 23524847. DOI: 10.1016/j.egyr.2022.04.043. URL: https://linkinghub.elsevier.com/retrieve/pii/S2352484722008022 (accessed on Sept. 9, 2024).

[68] Radu Platon et al. „Online Fault Detection in PV Systems“. en. In: *IEEE Transactions on Sustainable Energy* 6.4 (Oct. 2015), pp. 1200–1207. ISSN: 1949-3037. DOI: 10.1109/TSTE.2015.2421447. URL: https://ieeexplore.ieee.org/abstract/document/7098398 (accessed on Feb. 17, 2024).

[69] Santiago Silvestre et al. „Analysis of current and voltage indicators in grid connected PV (photovoltaic) systems working in faulty and partial shading conditions“. en. In: *Energy* 86 (June 2015), pp. 42–50. ISSN: 0360-5442. DOI: 10.1016/j.energy.2015.03.123. URL: https://www.sciencedirect.com/science/article/pii/S0360544215004727 (accessed on Feb. 20, 2024).

[70] Jubaer Ahmed and Zainal Salam. „An Accurate Method for MPPT to Detect the Partial Shading Occurrence in a PV System“. en. In: *IEEE Transactions on Industrial Informatics* 13.5 (Oct. 2017), pp. 2151–2161. ISSN: 1941-0050. DOI: 10.1109/TII.2017.2703079. URL: https://ieeexplore.ieee.org/document/7926354 (accessed on Feb. 19, 2024).

[71] I. de la Parra et al. „PV performance modelling: A review in the light of quality assurance for large PV plants“. en. In: *Renewable and Sustainable Energy Reviews* 78 (Oct. 2017), pp. 780–797. ISSN: 1364-0321. DOI: 10.1016/j.rser.2017.04.080. URL: https://www.sciencedirect.com/science/article/pii/S1364032117305920 (accessed on Feb. 18, 2024).

[72] Qi Liu et al. „Hierarchical context-aware anomaly diagnosis in large-scale PV systems using SCADA data“. en. In: *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*. ISSN: 2378-363X. July 2017, pp. 1025–1030. DOI: 10.1109/INDIN.2017.8104914. URL: https://ieeexplore.ieee.org/abstract/document/8104914 (accessed on Feb. 18, 2024).

[73] Mariam Ibrahim et al. „Machine Learning Schemes for Anomaly Detection in Solar Power Plants“. en. In: *Energies* 15.3 (Jan. 2022), p. 1082. ISSN: 1996-1073. DOI: 10.3390/en15031082. URL: https://www.mdpi.com/1996-1073/15/3/1082 (accessed on Feb. 20, 2024).

[74] Alvin C Rencher and G Bruce Schaalje. *Linear models in statistics*. en. John Wiley & Sons, 2008. ISBN: 978-0-471-75498-5.

[75] Peter Filzmoser. „Multivariate Statistik“. de. Wien, Oct. 2020.

[76] Sepp Hochreiter and Jürgen Schmidhuber. „Long Short-Term Memory“. en. In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco.1997.9.8.1735. URL: https://direct.mit.edu/neco/article/9/8/1735-1780/6109 (accessed on Sept. 3, 2024).

[77] Pengzhi Li, Yan Pei, and Jianqiang Li. „A comprehensive survey on design and application of autoencoder in deep learning“. en. In: *Applied Soft Computing* 138 (May 2023), p. 110176. ISSN: 15684946. DOI: 10.1016/j.asoc.2023.110176. URL: https://linkinghub.elsevier.com/retrieve/pii/S1568494623001941 (accessed on Oct. 10, 2024).

[78] Christopher M. Bishop. *Neural networks for pattern recognition*. en. Oxford : New York: Clarendon Press ; Oxford University Press, 1995. ISBN: 978-0-19-853864-6.

[79] GeoSphere Austria. *Messstationen Tagesdaten v2*. de. 2024. DOI: 10.60669/GS6W-JD70. URL: https://data.hub.geosphere.at/dataset/klima-v2-1d (accessed on Aug. 22, 2024).

[80] Achim Woyte, Johan Nijs, and Ronnie Belmans. „Partial shadowing of photovoltaic arrays with different system configurations: literature review and field test results“. en. In: *Solar Energy* 74.3 (Mar. 2003), pp. 217–233. ISSN: 0038092X. DOI: 10.1016/S0038-092X(03)00155-5. URL: https://linkinghub.elsevier.com/retrieve/pii/S0038092X03001555 (accessed on Aug. 22, 2024).

[81] Ane Blázquez-García et al. „A Review on Outlier/Anomaly Detection in Time Series Data“. en. In: *ACM Comput. Surv.* 54.3 (Apr. 2021), 56:1–56:33. ISSN: 0360-0300. DOI: 10.1145/3444690. URL: https://doi.org/10.1145/3444690 (accessed on Aug. 5, 2024).

[82] O. Coddington et al. „A Solar Irradiance Climate Data Record“. en. In: *Bulletin of the American Meteorological Society* 97.7 (July 2016), pp. 1265–1282. ISSN: 0003-0007, 1520-0477. DOI: 10.1175/BAMS-D-14-00265.1. URL: https://journals.ametsoc.org/doi/10.1175/BAMS-D-14-00265.1 (accessed on Aug. 22, 2024).

[83] Sebastian Jäger, Arndt Allhorn, and Felix Bießmann. „A Benchmark for Data Imputation Methods“. en. In: *Frontiers in Big Data* 4 (July 2021), p. 693674. ISSN: 2624-909X. DOI: 10.3389/fdata.2021.693674. URL: https://www.frontiersin.org/articles/10.3389/fdata.2021.693674/full (accessed on Aug. 22, 2024).

[84] José M. Jerez et al. „Missing data imputation using statistical and machine learning methods in a real breast cancer problem“. en. In: *Artificial Intelligence in Medicine* 50.2 (Oct. 2010), pp. 105–115. ISSN: 09333657. DOI: 10.1016/j.artmed.2010.05.002. URL: https://linkinghub.elsevier.com/retrieve/pii/S0933365710000679 (accessed on Aug. 22, 2024).

[85] Bradley Efron. „Missing Data, Imputation, and the Bootstrap“. en. In: *Journal of the American Statistical Association* 89.426 (June 1994), pp. 463–475. ISSN: 0162-1459, 1537-274X. DOI: 10.1080/01621459.1994.10476768. URL: https://www.tandfonline.com/doi/full/10.1080/01621459.1994.10476768 (accessed on Aug. 22, 2024).

[86] James R Carpenter et al. *Multiple imputation and its application*. en. John Wiley & Sons, 2023.

[87] Steffen Moritz and Thomas Bartz-Beielstein. „imputeTS: time series missing value imputation in R.“ en. In: *R Journal* 9.1 (2017).

[88] Hyun Ahn, Kyunghee Sun, and Kwanghoon Pio Kim. „Comparison of Missing Data Imputation Methods in Time Series Forecasting“. en. In: *Computers, Materials & Continua* 70.1 (2022), pp. 767–779. ISSN: 1546-2226. DOI: 10.32604/cmc.2022.019369. URL: https://www.techscience.com/cmc/v70n1/44403 (accessed on Aug. 22, 2024).

[89] D. L. Elwell et al. „Ongoing Experience with Ohios Automatic Weather Station Network“. en. In: *Applied Engineering in Agriculture* 9.5 (1993), pp. 437–441. ISSN: 1943-7838. DOI: 10.13031/2013.26006. URL: http://elibrary.asabe.org/abstract.asp??JID=3&AID=26006&CID=aeaj1993&v=9&i=5&T=1 (accessed on Aug. 22, 2024).

[90] Sheldon M. Ross. *Introduction to probability models*. de. Eleventh edition. Amsterdam ; Boston: Elsevier, 2014. ISBN: 978-0-12-407948-9.

[91] Marina Sokolova and Guy Lapalme. „A systematic analysis of performance measures for classification tasks“. en. In: *Information Processing & Management* 45.4 (July 2009), pp. 427–437. ISSN: 0306-4573. DOI: 10.1016/j.ipm.2009.03.002. URL: https://www.sciencedirect.com/science/article/pii/S0306457309000259 (accessed on Aug. 10, 2024).

[92] Mohammad Hossin and Md Nasir Sulaiman. „A Review on Evaluation Metrics for Data Classification Evaluations“. en. In: *International Journal of Data Mining & Knowledge Management Process* 5.2 (Mar. 2015), pp. 01–11. ISSN: 2231007X, 22309608. DOI: 10.5121/ijdkp.2015.5201. URL: http://www.aircconline.com/ijdkp/V5N2/5215ijdkp01.pdf (accessed on Aug. 10, 2024).

[93] Arvind Kumar et al. „A Review on Unbalanced Data Classification“. en. In: *Proceedings of International Joint Conference on Advances in Computational Intelligence*. Ed. by Mohammad Shorif Uddin, Prashant Kumar Jamwal, and Jagdish Chand Bansal. Singapore: Springer Nature Singapore, 2022, pp. 197–208. DOI: 10.1007/978-981-19-0332-8_14. URL: https://link.springer.com/10.1007/978-981-19-0332-8_14 (accessed on Aug. 22, 2024).

[94] David M. W. Powers. „Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation“. en. In: (2020). DOI: 10.48550/ARXIV.2010.16061. URL: https://arxiv.org/abs/2010.16061 (accessed on Aug. 22, 2024).

[95] Takuya Akiba et al. „Optuna: A Next-generation Hyperparameter Optimization Framework“. en. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019.

[96] James Bergstra et al. „Algorithms for Hyper-Parameter Optimization“. en. In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor et al. Vol. 24. Curran Associates, Inc., 2011. URL: https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

[97] Skipper Seabold and Josef Perktold. „statsmodels: Econometric and statistical modeling with python“. en. In: *9th Python in Science Conference*. 2010.

[98] François Chollet et al. *Keras*. 2015. URL: https://keras.io.

[99] TensorFlow Developers. *TensorFlow*. July 2024. DOI: 10.5281/ZENODO.4724125. URL: https://zenodo.org/doi/10.5281/zenodo.4724125 (accessed on Sept. 2, 2024).