Detection and Classification of Faults in Residential PV Systems with a Synthetic PV Training Database

A Machine Learning-Based Approach Using the PVMD Toolbox to Generate Synthetic PV Yield Data

D. de Mooy

TU Delft

Detection and Classification of Faults in Residential PV Systems with a Synthetic PV Training Database

A Machine Learning-Based Approach Using the PVMD Toolbox to Generate Synthetic PV Yield Data

by

D. de Mooy

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday June 14, 2022 at 12:30 PM.

Student number:	4449894	
Project duration:	March 29, 2021 – June 14, 2022	
Supervisors:	Dr. ir. R. Santbergen,	TU Delft
	Dr. J. L. Cremer,	TU Delft
	Dr. H. Ziar,	TU Delft
Thesis committee:	Prof. Dr. O. Isabella,	TU Delft
	Dr. Ir. A. Shekhar,	TU Delft

TU Delft

Abstract

In this thesis, a new photovoltaic fault detection and classification method is proposed. It combines the generation of a synthetic photovoltaic training database and the use of a machine learning model to detect and classify faults in small-scale residential PV systems. The database was generated in Matlab, and the machine learning modeling was done with the scikit-learn library for Python. From the modeled PV systems, solely power yield is used as an indicator, combined with system age and meteorological conditions. Using these features, four types of machine learning models are used to detect malfunctioning PV systems and classify short-circuit faults and open-circuit faults. This thesis also shows the benefit of a synthetic PV training base as opposed to alternative methods, with increasing performance due to control of database balance.

The result of this thesis is a method that can be used to construct a model for detection and classification of photovoltaic faults, specific to a single residential PV system. Malfunctioning systems can be detected with an accuracy of 80.4% using a random forest algorithm. For fault type classification, an F1-score of 0.759 was achieved, also using a random forest.

D. de Mooy
Delft, June 2022

Preface

This thesis was fulfilled as a requirement to obtain the degree Master of Sustainable Energy Technology at Delft University of Technology and is the result of my graduation project research. The project was carried out as part of both the Photovoltaic Materials and Devices group and the Intelligent Electrical Power Grids group. Although it was a long and challenging project, and the pandemic prevented physical meetings in the first months, it was an excellent opportunity to apply the knowledge obtained during the master program. This result would not have been achieved if it wasn't for a few people that helped me throughout the process. First of all, I would like to thank my supervisors Dr. ir. Rudi Santbergen, Dr. Jochen L. Cremer and Dr. Hesan Ziar for all their help and feedback. They helped me a lot with going in the right direction and to complete the thesis.

I would like to thank Prof. Dr. Olindo Isabella and Dr. Ir. Aditya Shekhar for being a part of my thesis committee. I would also like to thank Mert Karaçelebi for helping me with the machine learning modeling, which was completely new to me.

Finally, I want to thank my family and friends for their support during my research and my roommates for their company during study sessions.

D. de Mooy
Delft, June 2022

Contents

1 Introduction

1.1. Background

As the energy transition continues and the share of renewable energy keeps rising, the application of photovoltaic (PV) modules will increase. In 2020 the renewable energy production in the Netherlands increased by 40%, and the installed PV capacity increased by more than 3 GW to a total of more than 10 GW installed capacity [1]. Compared to other renewable energy sources, photovoltaic technologies have the advantage of applicability to be installed at a smaller scale i.e. household PV systems. This causes the existing electricity system to shift towards a more decentralized system, where people start to become prosumers rather than just consumers. This shift will generate new problems regarding grid stability and maintaining the energy balance regarding supply and demand. Besides that, there will also be a higher likelihood of underperforming systems, as these household PV systems are oftentimes not monitored sufficiently, and because the PV power output varies with the varying irradiance conditions, a reduction of power output due to a defect is difficult to detect. Consequently, the efficiency and reliability of systems decrease. To give an idea of the significance of energy production that is lost, average energy losses of 5.1% were reported for typical residential systems [2].

To ensure an effective and reliable energy source in the energy transition, it is of importance to digitalize the system. By doing so, it is possible to detect malfunctioning systems so that maintenance can be scheduled. This is to the benefit of research as an accurate performance evaluation is important to identify future challenges, but also benefits the prosumer financially. Furthermore, a solid performance assessment and maintenance plan allow photovoltaic power to reach its full potential, thereby increasing the confidence of the public in PV systems. Due to the lack of monitoring devices in these household PV systems, we are mostly left with just the power output of the system. There are, however methods that can automatically identify malfunctioning PV systems that use a PV yield database, using only output power of the PV systems [3]. To improve on such methods, machine learning (ML) algorithms could be applied to increase the performance. Limited study has been done on combining a PV yield database and machine learning that focuses on PV failure detection.

1.2. PV fault prediction

With the increasing presence of PV in the energy mix, several monitoring strategies have been developed, as well as fault detection methods. As the evolution of PV systems keeps continuing, more advanced techniques are being investigated [4]. Depending on the faults that are to be detected, different methods should be selected, as no method can detect and differentiate every type of fault.

1.2.1. Overview of PV Faults

The most common PV faults appearing in literature are listed below:

- Inverter failure

- Ground faults, or zero efficiency faults. Occurs when there is a short circuit between a conductor and the ground.
- Arc faults, often due to insulation breakdown in conductors, also leading to a short circuit.
- Line to line faults, a short circuit in the PV module between two strings, or two points of the same string.
- Broken string(s) or module(s)
- Bypass diode failure

As the behavior of a PV system responds differently to different failures, it is possible to distinguish various faults from one another. For example, a too-small inverter could completely limit the power output of a system above a certain threshold, while a broken string will result in an amount of power loss of the number of broken strings divided by the total number of strings. After a fault or failure in one part of the PV system occurs, the system remains deteriorated and depending on the type of fault or failure, the system will behave differently. This has to be considered when one wants to detect and distinguish what kind of fault occurs. Other faults appearing in literature are shading, soiling and degradation, but these happen naturally and are not the result of a fault or failure in the PV system.

1.2.2. PV System Model

To give a better idea of where the most common faults occur in the PV system, the PV generator model should be introduced, shown in figure 1.1.

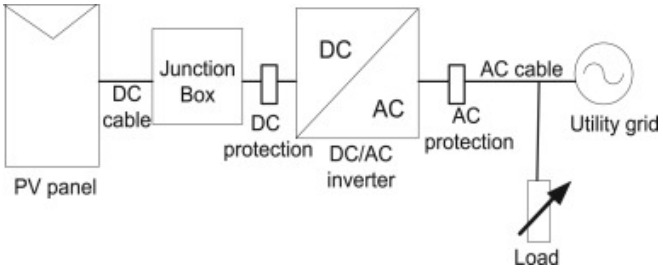


Figure 1.1: Schematic overview of grid-connected PV system [4].

Faults in a PV system are most likely to occur in the PV panel part of the system or due to a malfunctioning inverter. Most of the time, the PV panel part represents the array of PV modules. Every module consists of a string of PV cells or multiple strings. In a string, the cells are connected in series, and the strings themselves are connected parallel to one another. Multiple PV modules combined, again connected in a combination of series and strings, to form a PV array. One of the most used models to describe the PV panel part of the PV system is the one-diode model, as shown in figure 1.2.

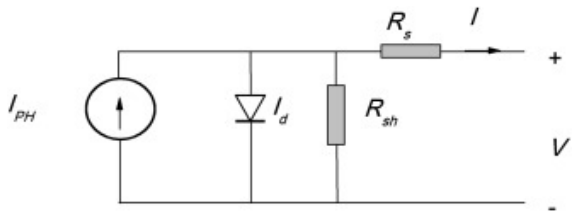


Figure 1.2: Schematic overview of one-diode model [5].

The model consists of the following parameters: The photo-generated current I_{PH}, a diode with diode current I_d, a parallel shunt resistance R_{sh} and series resistance R_s, and is expressed by:

$$I = I_{PH} - I_0(e^{\frac{V+R_s}{nV_t}} - 1) - (\frac{V + R_s I}{R_{sh}})$$

where n is the ideality factor, V_t is the thermal voltage and I_o is the reverse saturation current. The photo-generated current I_{PH} is impacted the temperature and irradiance, whereas the reverse saturation current I_o

is only influenced by temperature. By only measuring the output current, voltage, irradiance and temperature, and knowing the cell characteristics, all other parameters can be calculated.

The one-diode model describes only a single PV cell but can be scaled to a module level by scaling I, V, R_{sh} and R_s accordingly to the number of cells in series and strings in parallel. An illustration of the one-diode model at the module level is shown in figure 1.3 where N_P is the number of (parallel) strings, and N_S is the number of cells (in series) in each string. I_{PH} is multiplied by N_P, V should be multiplied by N_S which is not shown in this figure, and both R_s and R_{sh} are multiplied by N_S/N_P. Subsequently, this can be scaled up to array level and hence the one-diode model is a very convenient model. A more detailed description on the one-diode model is given by [5].

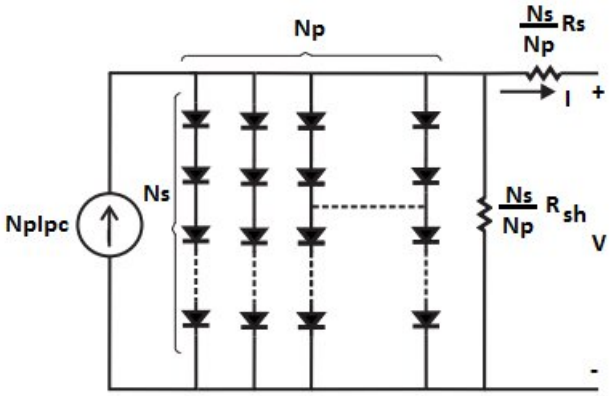


Figure 1.3: Schematic overview of one-diode model at module level [6].

1.2.3. Overview of Fault Prediction Methods

As mentioned in the introduction of this section, a lot of PV fault prediction techniques have been developed, and are still under development. These techniques can be classified in several groups or combinations of the following groups:

Electrical Circuit Simulation

In this technique, a model of the system is recreated using modeling software. A big advantage is that there is accurate knowledge of all electrical parameters of the system. When using this model for real-time simulation, it is often hard to accurately model the yield as atmospheric conditions are unknown. This can be controlled with the approach of recreating a system using modeling software. One could use modeling software to manually introduce faults so that the changing behavior of the system can be studied offline and used as reference data of a 'malfunctioning system' [7].

Comparison between Measured and Modeled Output

Some studies, such as look at the difference between modeled/predicted output parameters of a system and the actual measured output parameters [2], [8], [9]. Oftentimes, the difference in power between the modeled system and the real system is analyzed to make predictions on the state of the system. Also, output voltage and current can be considered as extra indicators [10]. Some studies even use satellite data as a replacement for on-site monitoring systems [11]. The modeled system is considered to be under normal operation. The difference in power relative to the measured system could then be considered lost due to either faults or degradation.

Statistical Analysis

Statistical quality control, often used in the manufacturing industry, monitors and ensures the quality of a product. Changes in operation of certain outliers can be detected. By using statistical charts that for example set control limits on certain variables, or look at the weighted average such methods can also be applied in the field of PV fault detection [12].

Artificial Intelligence

Machine learning-based approaches are very powerful tools that automate model building. A variety of algorithms can be applied and is already used in literature, also for PV fault detection [10], [12] and [13], [14], [15], [16], [8]. A machine learning-based approach should be paired with either a real experimental PV system or a modeled PV system to obtain data on which an analysis can be performed. In the next section, a short introduction to machine learning is given.

1.3. Machine Learning

Machine learning (ML) is a part of artificial intelligence that applies computer algorithms to given data to improve itself automatically. These algorithms are thus self-learning and are mainly used for two things:

- Classification of data, such as automatic detection of spam-mail or face recognition.
- Making predictions, for example the prediction of stock market prices.

To make predictions or do classification, so-called "training data" is used. The algorithm then uses this training data to iteratively improve itself so that the prediction or classification becomes more accurate. The idea is that this improved algorithm then can also correctly predict new outcomes or classify new data. ML algorithms can be divided into "supervised learning" and "unsupervised learning".

Supervised Learning

Supervised learning algorithms need training data that consists of both input and expected output. When new "test data" is given of which the expected output is not yet known, the algorithm tries to correctly predict the expected output given the input of the test data. The most popular supervised learning algorithms are regression and classification. Regression is used when the output has a continuous-valued output, whereas classification has a discrete-valued output e.g. binary classification.

Unsupervised Learning

For unsupervised learning, there is no expected outcome but only input. The training data is not "labeled". The most common unsupervised learning algorithm is clustering, which aims to group or cluster data based on similar or related variables.

As this study aims to predict malfunctioning of PV systems and identify the type of fault that is ongoing, the data is labeled (fault/no-fault or type of fault). Therefore supervised learning is the preferable type of algorithm for this study. Consequently, labeled data is necessary: data of both PV systems that work correctly and malfunctioning PV systems, preferably with correctly classified fault types. A straightforward method would be the use of a large enough database, as was done with an analytical approach [3]. However, as far as the author's knowledge goes, there is limited study that links machine learning with purely a PV databases, and oftentimes there is a real-time PV system used for data gathering [8], [10], [12], [14], [15], [16].

1.4. PV Database

The use of a database to train the ML model is an interesting idea, but a problem that immediately occurs is the limited access to PV data. Oftentimes, this data is limited to solar system installation companies and distribution network operators. However, with the introduction of the Internet, it has become easy to collect enormous amounts of data worldwide. This can be used to create open databases for a wide variety of scientific research fields and thus also for PV yield prediction. However, to harvest this data, owners of the PV systems should actively participate in registering their system, and data should be shared in real-time. Such participation of the public in science is referred to as 'citizen science'. As the European Commission realized, citizen science can significantly contribute to finding solutions for challenges we face; a budget of 462 million euros was earmarked for SwafS (Science with and for Society) [17]. One of the listed projects is the GRECO project, which aims to apply this open science approach in the PV field. One of the products of the GRECO project is the Generation Solar App, which is an open database of small-scale PV systems [18]. Unfortunately, lots of data is lacking, with the database essentially only consisting of system parameters, rather than actual output data. Also, the number of installations so far is only 132, showing the lack of information of available data in the PV field. Other studies involving PV databases show similar flaws: not enough systems and/or lack of useful data (output power etc.) [19], [20] and [21]. Another problem with these open databases is that it

contains lots of data input errors as most citizens are no experts in the PV field and might make mistakes with correctly putting in the data [22]. The best open database available, especially in the Netherlands, is PVoutput.org with 10528 registered installations, accounting for 68 MW installed capacity, including real-time data. However, this database shows the same problems as mentioned earlier: data input errors. To correct all the input errors, lots of unnecessary work has to be done regarding pre-processing of the data and verifying the system parameters, which is not desirable. Therefore, it can be concluded that these open databases in their current state are not suitable for accurate fault detection and classification.

An alternative approach would be to create synthetic data so that all necessary data is available and can be controlled. By modeling PV systems and introducing faults, a database with useful training data could be generated, which can then be used to train an ML model for detection and classification of faults in PV systems. A synthetic training database comes with multiple advantages compared to open databases with historic data. First of all, to some extent there is no reliance on PV system owners to correctly input their system design parameters. Having control over the data generation process ensures correct input of system parameters and results in complete data without input errors. Secondly, a synthetic training database can be balanced better regarding healthy systems and malfunctioning systems. Besides that, there is often a lack of knowledge on historic data of the operational state of a PV system i.e. there is no indication of a system being healthy or malfunctioning, historic data tends to be skewed heavily towards healthy operating systems, when there would actually be information on the operational state of systems. This is especially the case for a specific fault type that rarely occurs. This type of fault might however significantly impact the performance and safety of a PV system. By manually controlling the balance in the training database, it is expected that an ML model becomes better in detecting and correctly classifying faults, especially the more rare fault types. Thirdly, with a synthetic database it is easy to quickly generate a lot of data for a specific system design. The ML model can then be trained specifically on this particular PV-system which hypothetically also increases the performance of detection and classification.

1.5. Objective

The objective of this thesis is to test the performance of machine learning models for PV system fault detection and classification, trained on a synthetic PV database. To achieve the objective, four steps have to be taken: identifying the common PV faults, generating a synthetic PV database, designing machine learning algorithms for PV detection and classification, and comparing the performance of the different machine learning models. The research question is formulated as follows:

- How accurate can an ML-based model detecting and classifying faults in residential PV systems get when using a synthetic PV training database?

Additionally, the research question is divided into sub-questions addressing the different aspects of this study. Each sub-question will be answered by a separate chapter:

- What are the most common PV faults in residential PV systems and what is the impact of these faults on a system? (Chapter 2)
- How can a synthetic PV training database be generated for PV fault detection and fault classification in residential PV systems? (Chapter 3)
- How to design a machine learning algorithm for PV fault detection and classification? (Chapter 4)
- What is the performance of different machine learning algorithms trained on a synthetic PV database for fault detection and classification? (Chapter 5)

After answering the sub-questions and main research question, it is expected to have a better understanding of PV failure behavior of PV systems and have a database that can be used for further studies. In practice, PV system owners could receive a trained fault detection and classification model specifically trained on their PV system. In return the PV system owner only has to provide the necessary system parameters of their PV system. Another idea would be to have a company use this method for periodic maintenance checks of PV systems of customers. Also, this database could act as a starting point for the creation of a PV monitoring platform where system owners can actively share their system data so that they can be informed when their system is underperforming of malfunctioning. The key for all these ideas in practice is that the system parameters should be correct, which in the case of an involved company could be easier to do.

1.6. Outline of Thesis

Chapter 2 discusses the most common and impactful faults in residential PV systems and the effect each fault has on the PV system, and how this affects the physical model of the system. In chapter 3 the whole process of generating the synthetic PV training database is described. These two chapters result in the training database and form the first half of this study. In chapter 4 a more elaborate introduction on machine learning is given, the used algorithms will be explained and the necessary theory is explained. In chapter 5 the results of the designed machine learning models, trained on the synthetic PV database, are discussed. In chapter 6 the findings from earlier chapters are put into perspective with respect to the economics and scalability of the proposed method. The thesis concludes with conclusions and recommendations in chapter 7.

2

PV Faults

In this chapter, the sub-question "What are the most common PV faults in residential PV systems and what is the impact of these faults on a system?" is answered. When working with fault detection, it is important to know with what type of faults we are working, what their impact is on the PV system and how it changes the electric variables in the system. As mentioned in section 1.2, not every fault detection method is able to detect all types of faults. Depending on the technique and available system data, different methods could be applied. The two key aspects of faults with respect to fault detection and classification are the frequency of a specific fault occurring and the impact of the specific fault on the system. As for the frequency of occurrence, it is important that a specific fault that occurs more often is easily detected. However, there might be faults that almost never occur but still have a large impact on the system. This could lead to a large decrease in power output or damage to the system. For example, fire hazards are a growing concern, especially for residential PV systems, due to the lack of monitoring. Although a photovoltaic system catching fire is a rare occasion, especially when the system is properly installed by an accredited installer, there is still a small chance something disastrous could happen. These events should be avoided at all costs and therefore it is important the detection method also takes these possibly more rare faults into account.

This chapter will discuss the common faults occurring in photovoltaic systems. In section 2.1 an overview of PV faults is given, and the faults that are taken into account in this study are discussed. Section 2.2 describes the impact these faults have on the physical system i.e. how the system and its variables change compared to a healthy operating system. Finally in section 2.3 the first sub-question is answered.

2.1. Overview of PV Faults

To increase the effectiveness of the fault detection method, a literature review was carried out to identify what are the common faults in PV systems. As most research has been done on large-scale systems, this is not perfectly accurate for residential systems only, but it gives a good indication of what type of faults can be expected. The faults that are included in this study are the following:

- Ground fault; short-circuit when a conducting part of the system makes contact with a grounded part of the system (subsection 2.2.1).
- Line-to-line fault; short-circuit when two conducting part with different potential make contact (subsection 2.2.2).
- Broken string fault; when an open-circuit is created, for example by a broken cable or loose solder joint (subsection 2.2.3).
- Inverter failure or zero efficiency fault; when the inverter trips and shuts down the system. Output power drops to zero. Can have many different causes (subsection 2.2.4).

Ground faults are by far the most occurring faults on the DC-side of PV systems [10], [23], [24], [25]. After that, line-to-line faults (or phase-phase faults) and arc faults are the main reasons for failure. Other faults mentioned in literature are mainly component failures: broken interconnections, broken cells or modules,

bypass diode failure etc. [26], [27], [28]. These component failures can have different causes, i.e. a broken cell can be caused by poor manufacturing quality but also by inherent shading leading to hot spots that can damage the cells.

Besides faults, another common cause of decreased efficiency of PV systems is shading. Shading can be divided into three types [29]:

- Fixed shading: shading caused by soiling, bird droppings or leaves having a constant impact on the system performance.
- Variable shading: shading caused by trees, poles or other buildings that block sunlight depending on the sun position.
- Intermittent shading: random shading, mainly caused by clouds.

Shading is not in the scope of this study. Intermittent shading is a problem that can't really be tackled and fixed shading is generally something that can better be visually inspected whereas variable shading can easily be done by horizon construction. In this study the focus will also lie on detecting failures in the system rather than the cause of this failure. Consequently, once a failure has been detected, further investigation of the system could be carried out to identify what has caused the specific failure.

Around 85% of PV system failures are due to electronic devices (inverter, DC power combiner, PV optimizer, conversion devices) which are on the AC-side of the PV systems [30]. This aligns with another study [31] which discusses a collection and analysis of PV faults and failures occurring in PV power plants but also distributed PV systems. The distributed PV systems are bundled in one portfolio and when ignoring grid failures (which are outside our reach), inverter- and dc combiner failures make up exactly 85% of failures, with the inverter accounting for around 66.7%. The DC-side accounts for the other 15% of fault or failures. A broken string is present in the 3-4% range of systems [3], [31]. One other cause of decreased power output which is not actually a fault is inverter clipping i.e. max power output being capped by the inverter size. It is reported that 22% of residential PV systems are inverter clipped [3]. The impact on performance/power out, frequency and risk of damage/hazards of the faults of this study are summarized in a qualitative table 2.1. The scales are relative with very high being the most likely fault to occur or the fault with the highest effect on power output. In terms of power impact, the ground fault, line-to-line fault and inverter fault can eventually all result in zero power output, therefore categorised as very high. In terms of frequency, the inverter failure is most common, therefore categorized as very high. Meanwhile, a broken string is least likely, but there are faults not taken into account being even less likely, resulting in the category low. The qualitative scale is arbitrary but is a good indicator of the different impacts of the faults included.

Table 2.1: Impact and frequency of faults on a relative scale from very low to very high with very high being the most common fault type.

Fault	Power Impact	Frequency	Danger Level
Ground Fault	Very High - Low	High	High
Line-to-Line Fault	Very High - Low	Medium	Very High
Broken String	Medium	Low	Low
Inverter Failure	Very High	Very High	Very Low

2.2. Impact on Physical Model

In this section, the included faults that were listed in 2.1 are explained in more detail, and the impact of the fault on the physical model of the PV system is discussed.

2.2.1. Ground Fault

Ground faults occur when a conducting part of the PV system accidentally makes direct contact with a grounded part of the system. The current is essentially short-circuited, and depending on the impedance of the fault this results in a very high current, potentially resulting in fire hazards. In most systems, ground-fault protection devices, GFPDs e.g. fuses, are included. A GFPD detects there is a leakage current somewhere in the system and, when large enough, will break the circuit. GFPDs have a certain rating corresponding to the

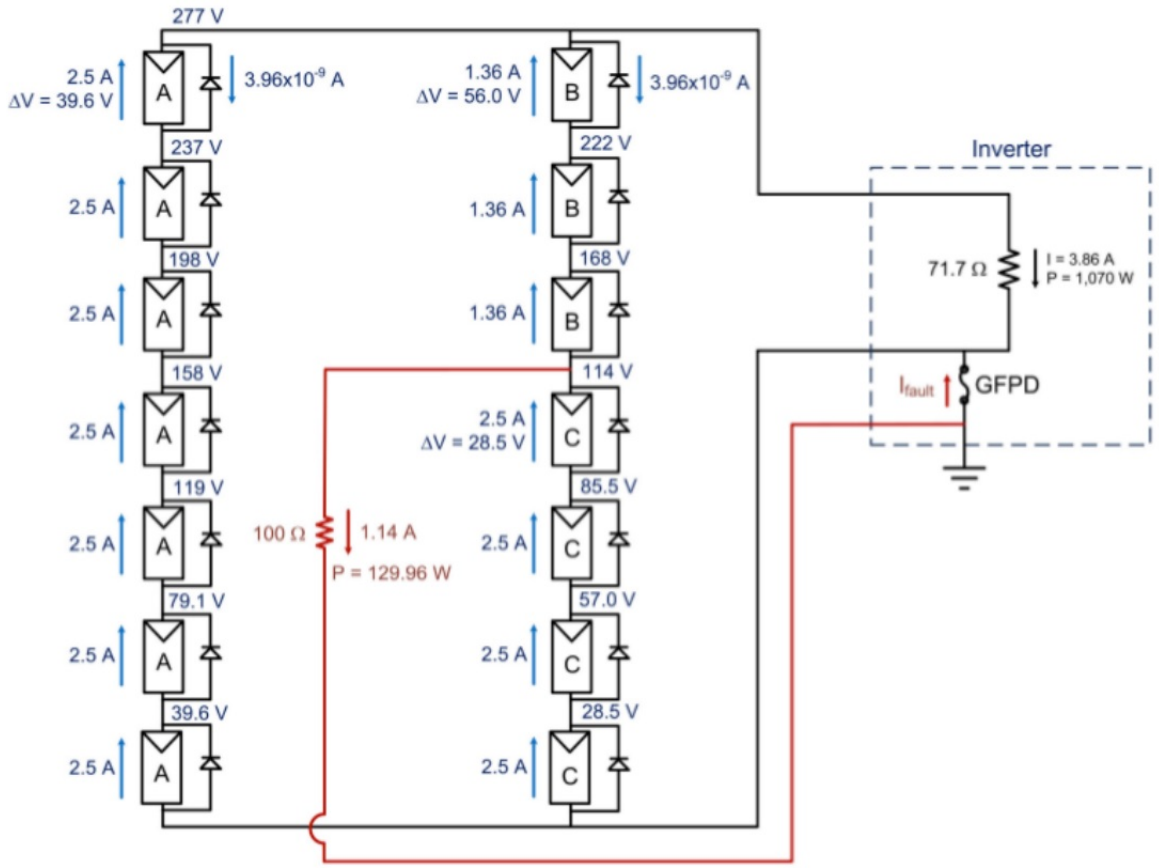


Figure 2.1: Schematic overview of a ground fault in a two-string array with 14 modules [32].

current at which they start operating. For example, a 1 A fuse will "blow" when the current through the fuse reaches 1 A or more.

There is a wide variation in ground faults, depending on the location in the system and the impedance of the fault. Ground faults can occur within modules or in conductors between modules. When continuous direct contact is made, a fault current flows, which can be modeled as a simple resistance [32]. An example of a ground fault in a small-scale PV system can be seen in figure 2.1. In this case, the fault occurs between the 4th and 5th module of string 2. The fault is modeled as a controllable switch and a series resistance representing the fault resistance. The voltage point between the 4th and 5th module is equal to the product of fault current and fault resistance, making it so that the modules below the fault decrease in voltage and shift on the I-V curve towards short-circuit operation. The modules in string 2 above the fault will increase in applied voltage, and their operation shifts towards open-circuit operation. Due to the fault, also the modules in string 1 will have decreased operating voltage as the applied voltage of the complete array is decreased. In this specific case, the power output of the array is only 64.7% of the maximum power output of a healthy operating system.

When more modules are involved in the fault i.e. the fault occurs "higher" in the string, the fault current will be higher. According to Kirchoff's current law, the fault current will be the difference between the current of modules below the fault and the current of modules above the fault. Consequently, the array voltage and current will be lower, and therefore the array power will be decreased. When the fault occurs "lower", fewer modules are involved and fault current will be lower, leading to a smaller decrease in array power. Especially the faults only involving one module are interesting for monitoring, as these might go unnoticed by the GFPD, depending on the fault resistance. It is also important to notice that PV systems are almost never

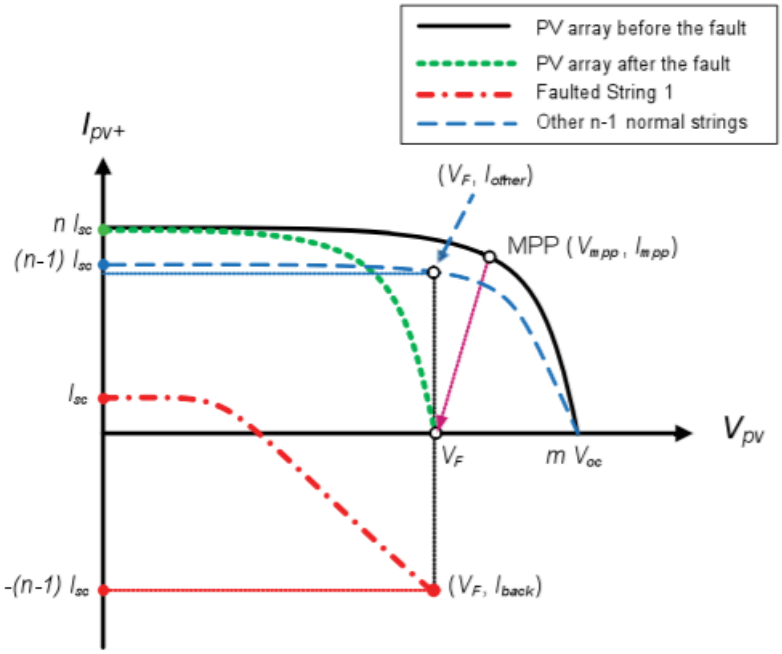


Figure 2.2: I-V curve of a ground fault with backfed current into faulted string [33].

under STC conditions. Generated current can be much lower under real conditions, when irradiance is low. Consequently, the fault current will be lower and therefore a ground fault can go unnoticed while having a significant impact on the system performance.

When enough modules are faulted and fault resistance is low enough, a phenomenon called backfed current occurs. Because the fault resistance is low, the fault is near ground potential so that the complete array voltage is applied to the modules above the fault. The operational voltage exceeds the open-circuit voltage and the modules start to act as a load in the 4th quadrant of the I-V curve [33]. The I-V curve for a ground fault with backfeeding is shown in figure 2.2. A negative current is flows in the faulted string fed by the other strings in the system. This backfed current flows into the fault. Meanwhile, the faulted modules flow short-circuit current into the fault. Generally, when a GFPD is in place, this will be cleared as the fault current is high enough due to the back feeding and therefore, the fuse will blow. There is little information on fault conditions, their resistance and how they progress over time [32]. Resistances can range from below 0.1 Ω to relatively large resistances, but the simulations show that for small-scale systems, ground faults larger than 1000 Ω have little to no impact.

It can also occur that two conductors with different potentials approach each other to such a distance that current flows via the dielectric breakdown of gas between the two conductors. This type of fault causes an electric arc and is therefore also called an arc fault. An arc fault is not restricted to contact between a current-conducting part and grounded part of a PV system (i.e. ground arc fault) but can also occur between two conducting parts (i.e. phase-phase arc fault). An arc ground fault and regular ground fault can be modeled the same way [32]. Although arc faults show highly non-linear characteristics, there was no analytical difference in transient effects when running simulations.

Generally, ground faults in PV systems don't pose a risk of any severe damage to the system because in most cases, the GFPD will clear the fault and if not, the fuse does not blow because the fault current is very low. There is however a special case that has caused multiple fires: the ground fault happens to be in a detection 'blind spot' i.e. goes unnoticed. As soon as a second ground fault in the system occurs, the traditional protection device is not able to clear the fault due to the already existing fault, which can lead to fires.

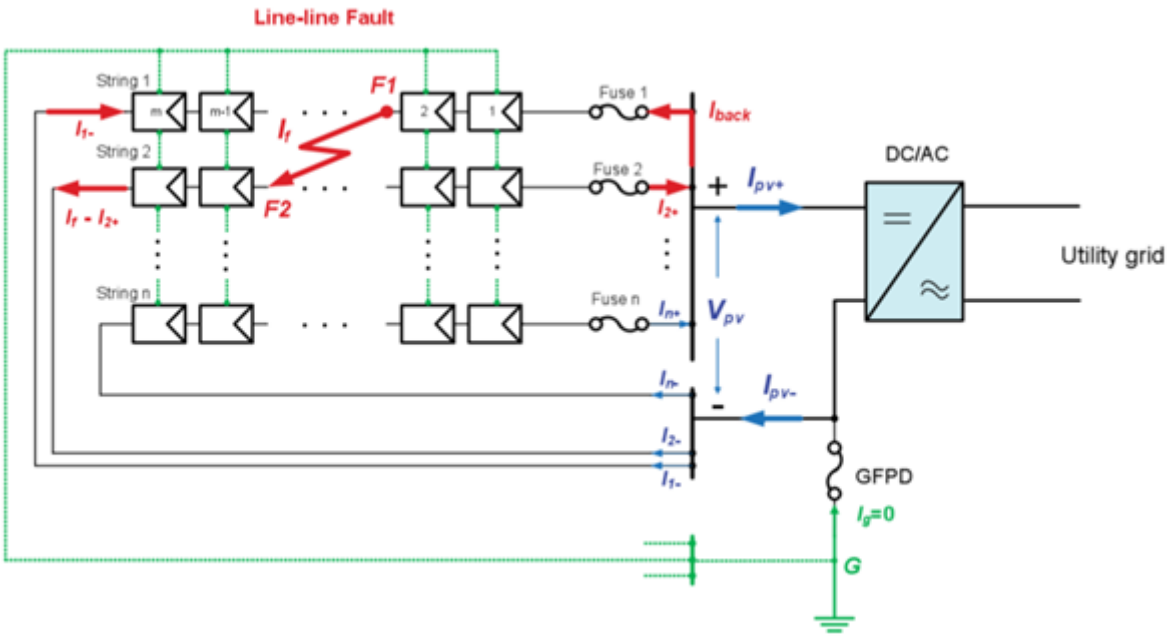


Figure 2.3: Schematic diagram of array under a line-to-line fault [34].

2.2.2. Line-to-Line Fault

Another type of short-circuit fault commonly occurring in PV systems is a line-to-line fault, or phase-to-phase fault. A line-to-line fault is similar to a ground fault and can be the result of insulation failure of cables, incidental contact between two current conductors or damaged DC junction box [34]. A line-to-line fault can happen either between two separate strings or inside the same string, referred to as cross-string and intra-string respectively [35]. The fault current depends on the different potential of the two connected points. Just as discussed in section 2.2.1, a backfed current can occur depending on the magnitude of the potential difference. To prevent any damage done to the system by a line-to-line fault, over-current protection devices (OCPD) should be included in the circuit. Usually, fuses are utilized as OCPD in series with the PV modules. A single fuse per string would be enough as the backfed current will always flow through the OCPD as long as the system is grounded. OCPDs are usually rated based on the short-circuit module current, at 1.56 I_{SC}.

A schematic diagram of the PV array under a cross-string line-to-line fault is shown in figure 2.3. Due to the contact between points F1 and F2, the modules left of the fault of strings 1 and 2 are now in parallel, therefore sharing the same voltage. Thus, the voltage of the (m-2) modules of string 1 are down to the same voltage of the two modules of string 2 left of point F2. A similar situation occurs on the other side of the fault. Just as described in section 2.2.1, the disruption of the array might cause modules to operate beyond their open-circuit voltage when the potential difference is large enough. Similarly, this string will start acting as a load and a backfed current starts flowing into the string, fed by the unfaulted strings. When the backfed current is large enough the OCPD will trip. The situation might also occur where the potential difference is small, when there is only one module-level difference between points F1 and F2 [36]. In this case, there is no backfed current in the faulted string. A line-to-line fault with a small voltage difference will thus not trip the OCPD. While the fault goes unnoticed, this will definitely have a negative impact on the system performance.

The corresponding I-V curve for the line-to-line fault shown in figure 2.3 can be seen in 2.4. The behavior is similar to the ground-fault I-V curve in figure 2.2 but the operating voltage of the array is the same before and after faulting, at MPP voltage. Contrary to a ground fault, the operating voltage of the system does not change. Again it is clear that the faulted string operates as a load, as the I-V curve is in the 4th quadrant. After a line-to-line fault has been cleared by an OCPD, there is no longer a current backfeeding into the faulted string. However, the fault path still exists, and a fault current is still running, although reduced by the fuse. A schematic diagram of the PV array under cleared conditions is shown in figure 2.5.

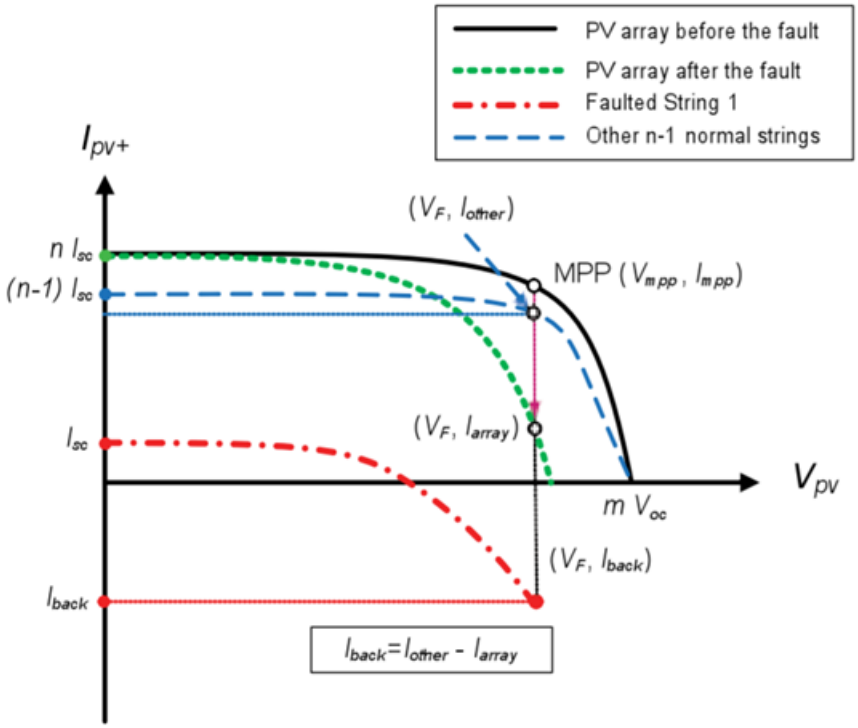


Figure 2.4: IV-curve of line-to-line fault [34].

When systems are properly protected, line-to-line faults should not have a high risk of fire hazards. Depending on the potential difference, the fault current can be very high but will be reduced by the proper operation of an OCPD. Still, line-to-line faults are with ground faults the most likely to do severe system damage, as these two fault types can cause arc faults. The electric discharge of an arc fault goes accompanied by lots of heat which melts the surrounding insulation and trigger fires.

2.2.3. Broken String Fault

An open-circuit fault can occur either inside a PV module, i.e. a broken cell string, or in a module string and is also referred to as a broken string fault. Essentially, this type of fault occurs when a component in the system fails such that an open circuit is created. Several reasons can cause an open-circuit fault: broken cells, physical breakdown of cable joints, loose connections or aging of power cables, amongst other reasons [37]. To lower the impact of open-circuit faults on the PV system performance, bypass diodes are included in modules.

Bypass diodes in PV systems redirect the current in case of an open circuit, so that circuit continuity is maintained. These bypass diodes will also prevent damage to cells in case of a current mismatch due to partial shading. The current will flow past the affected cells or modules, depending on how many bypass diodes are installed. Consequently, the voltage of the string decreases as fewer modules in series contribute to the complete string voltage, while the string current is unaffected. The total system power is therefore decreased. Ideally, every solar cell has its own bypass diode, but this is rather expensive and therefore modules oftentimes have either one or three bypass diodes.

There are multiple different cases of an open-circuit fault:

- The first case is an open-circuit fault such that a complete string of modules is faulted, for example when the interconnection between two modules fails. Therefore the power contribution of the faulted string to the inverter will be zero. The PV system power is decreased by

$$\frac{n_{fault}}{n_{total}} \tag{2.1}$$

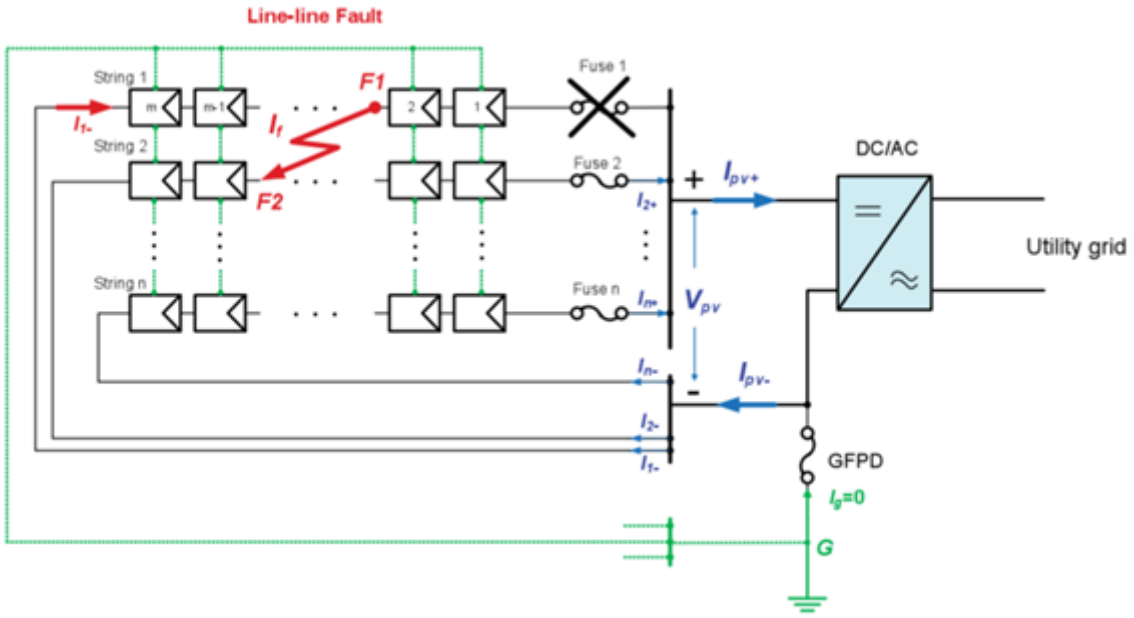


Figure 2.5: Schematic diagram of array after cleared line-to-line fault [34].

where n_{fault} corresponds to the number of faulted PV strings and n_{total} is the total number of strings in the PV array [3]. The array voltage is unaffected by the fault, but rather the array current is decreased.

- In case of an open-circuit fault inside a module with a single bypass diode, the complete module is essentially excluded of the string when the module is affected by a broken component, as current will bypass the module via the bypass diode. The string voltage, and thus string power, will be decreased by approximately [38]

$$\frac{m_{fault}}{m_{string}} \tag{2.2}$$

where m_{fault} corresponds to the number of faulted modules and m_{string} is the total number of modules in the string. This is true under STC but can be extrapolated for varying temperature and irradiance levels. Consequently, a voltage mismatch between the PV strings of the array is present. The MPPT will then enable the other strings to decrease their operating voltage so that the string voltage will match the faulted string voltage [39]. Thus, the power output of the full system decreases similarly to the decrease in power of the faulted string.

- Lastly, there is the case of an open-circuit fault inside a module with three bypass diodes. With three bypass diodes, the module essentially consists of three strings of cells, one of which is faulted. Only two-thirds of the module will now contribute to the string, and the generated power of the module is thus decreased by one-third. A similar case, discussing shaded cells rather than open-circuit faults, is described in literature [39]. The other modules in the string are not affected and remain operating at MPP (maximum power point). Similar to the previous case, the voltage of the string containing the faulted module will be lower, but only by one-third of the module voltage. Again, the MPPT takes care of the voltage mismatch, and the system power will be decreased by approximately

$$\frac{m_{fault}}{3 * m_{string}} \tag{2.3}$$

For the second and third case, it is clear that the decrease in power after an open-circuit fault inside a module is independent of the number of strings but rather depends on the number of modules per string. Another fault that might occur is the failure of a bypass diode. A shorted bypass diode will have a similar effect as the last two cases of an open-circuit fault mentioned above. Current will flow through the path

of least resistance, the shorted bypass diode, therefore bypassing the affected cells or modules. Damage to the system involving open-circuit faulting is limited as the fault will prevent all current flow, as opposed to a short-circuit fault where the current will rise significantly and can cause lots of heat and the risk of fire. However, the impact on system performance is still significant, and a lot of potential power generation is lost.

2.2.4. Inverter Faults

The photovoltaic inverter (PVI) is a critical component of PV systems converting the DC current generated by the PV array to AC current that can be used for residential purposes or fed to the grid. Inverters often have maximum power point tracking (MPPT) to optimize the efficiency and ensure maximum power delivery of the PV system. The inverter samples the I-V curve of the modules and applies the right resistance accordingly so the modules can operate at maximum power. As temperature, irradiance and other parameters change, the inverter uses an algorithm to make sure the array keeps performing at the new MPP.

The problem with inverter faults is the wide variety of faults, as the inverter is a complex part of the system consisting of multiple components. Inverter failure causes can be cooling issues, fan failure, IGBT failure or control software issues. However, ground faults and line-to-line faults will often also trip the inverter, which is then often marked as inverter fault, while the problem lies elsewhere in the PV system. Moreover, once an inverter trips it will be easily noticed as power production drops to zero. Using fault detection techniques is therefore not of much relevance as on the one hand it is near impossible to pinpoint the fault causing the inverter to fail; on the other hand, the output power is zero and it's clear to anyone something is wrong. Consequently, a 0% efficiency fault category could be included in the detection method. In terms of hazards involving inverter faults, the risk is low. A failing inverter will shut down the system and essentially removes any risk. Some system damage could happen, mainly to the inverter itself, but it is low compared to the potential damage of a ground fault or line-to-line fault.

Inverter Clipping

One inverter-related phenomenon however, that has a significant impact on system performance and can also be detected is inverter clipping. Inverter clipping occurs when the DC power generated by the PV array exceeds the maximum inverter power. At this point, the inverter will shift the operating point of the array by increasing the DC voltage, therefore reducing the DC current, such that the DC power is at maximum inverter power [40]. Thus, the absorbed solar energy is not fully converted to electricity.

When designing a PV system, the appropriate DC/AC inverter ratio, or inverter load ratio (IRL), is based on meteorological conditions and financial considerations. Usually, a ratio around 1.13-1.30 is chosen, such that the inverter is slightly undersized compared to the PV array [41]. For example, a 5 kWp system could be paired with a 4.5 kW inverter. Most of the time, this is not an issue as the DC side is rated at peak power i.e. under STC conditions. Especially in the Netherlands, STC irradiance of 1000 W/m2 is rarely encountered. Therefore, the clipping losses are minimal even with an undersized inverter. Besides having a lower investment cost, a smaller inverter also ramps up faster in the morning and ramps down slower in the afternoon, leading to a slightly higher power production over the day.

Although some power clipping over the year is not a concern, it is not uncommon for residential PV systems to have an inverter that is actually undersized too much. As was shown, 22% of systems were inverter clipped [3]. It is not clear by how much these systems were clipped and if it was 'acceptable' clipping i.e. inverter clipping that is to be expected under ideal conditions. However, the fact that all these systems are located in the Netherlands makes it unlikely that all clipped systems fall in this category of 'acceptable' clipping. Therefore, this is an issue that is of concern while also being easily detected. Due to the ease of detection of inverter clipping, this fault is not considered in this study as there are easier methods to detect this problem. The generated database will however naturally contain some cases of inverter clipping.

2.3. Conclusion

The first sub-question "What are the most common PV faults in residential PV systems and what is the impact of these faults on a system?" can now be answered. The most common PV faults occur due to failure of the inverter, DC power combiner, PV optimizer and other conversion devices. On the DC-side of the PV system, the most common faults are open-circuit faults (broken string or broken cells), short-circuit faults (ground

fault or line-line fault) or the failure of bypass-diodes. While inverter failures and open-circuit faults have little impact on the safety of a system, short-circuit faults can be more severe with the risk of fire hazards as a consequence.

With respect to power output decrease, open-circuit faults have the most significant effect. The operating voltage of the string in which the fault occurs is decreased and consequently the operating voltage of the system. The operating current is unaffected. Short-circuit faults have either a very small impact on the power output loss or will cause the power output to drop to zero. In the case of a ground fault, both system current and voltage are changed, where for a line-line fault the operating system voltage remains similar to before the fault. Inverter failure will result in zero power output, but oftentimes inverter shutdown is caused by a short-circuit fault. Therefore a zero-efficiency fault category is more logical to be included.

3

Synthetic Database Generation

In this chapter the sub-question "How can a synthetic PV training database be generated for PV fault detection and fault classification in residential PV systems?" is answered. As described in section 1.4, open-source initiatives are a great contribution to science, but unfortunately in the field of PV monitoring, qualitative options are lacking. Synthetic generation of data is a good alternative, but to be able to correctly implement this approach, it is of uttermost importance that the generated data is realistic, resembling real data. Modeling of PV energy yield is already widely explored, and several software tools are available for this purpose, such as MATLAB [42]. The Photovoltaic Materials and Devices (PVMD) group from Delft University of Technology built an Energy Yield Prediction Model for PV systems: the PVMD Toolbox. In the PVMD Toolbox, 6 units should be consecutively run: 'CELL', 'MODULE', 'WEATHER', 'THERMAL', 'ELECTRICAL' and 'CONVERSION'. A complete PV system is modeled from the ground up, starting with modeling the individual cells, with options for tandem cells and bifacial cells. In the 'WEATHER' unit, the meteorological aspect is taken into account, including the reconstructed horizon at the location, to account for shading. The 'THERMAL' unit deals with the cell temperatures. After the first four units i.e. the module characteristics, irradiance and temperature, the electric simulation can be performed to obtain the I-V curve, energy yield and more. The last unit, 'CONVERSION', deals with the number of modules, choice of inverter and cable losses. The PVMD Toolbox is still under development and new features are continuously added. This toolbox can be used to generate data so a database can be built with PV system output data. Unfortunately, as of today, there is not yet an option to introduce faults in the system to get data that can be labeled 'faulty'.

This chapter will outline how a well-constructed synthetic database can be generated using the PVMD toolbox, and is subdivided into four sections. Section 3.1 discusses the general input parameters of our system and how to sample these. This allows for the generation of data for healthy systems. Section 3.2 describes how to sample the fault parameters so that, in combination with the general input parameters, data for faulted systems can be generated. Section 3.3 gives a description of how the faults are implemented in the Toolbox. Section 3.4 concludes the generation of the synthetic PV training database. In the final section 3.5, the second sub-question is answered.

3.1. Normal System Operation

In this section, all the input parameters of PVMD Toolbox specific to this study are discussed for normal system operation. For this study, normal system operation is considered to be a healthy operating system without faults or defects.

3.1.1. System Parameters

For ease of implementation, the PV system design will be fixed when creating the database. The idea is that when a future customer wants to run our failure detection model, the design of their system should be delivered, and the model would be run accustomed to their PV system specifically. The end product would be a failure detection ML model specific to the customer's system. The system design parameters used for this study are described for each part of the PVMD Toolbox, consisting of six parts: cell, module, weather, thermal, electrical and conversion. The exact value for every parameter can be found in appendix A.

- The **cell** part of the simulation simulates the individual solar cell, taking into account the complete geometry of all layers as well as the optic parameters of the materials used. For this part, a monocrystalline PERC (passivated emitter- and rear contact) solar cell is considered. An 'old save' of the output of this part will be loaded, which saves a lot of computational time.

- The **module** part of the toolbox considers the figuration of the cells and the geometry of the module. The parameters included are module tilt, module azimuth, module height, cells in series, cell strings in parallel, module thickness, cell spacing, edge spacing, module side spacing, module row spacing, cell length and -width, the albedo of the cell. Ray tracing is performed so that the possible shading of the surroundings is taken into account. Again for the ray tracing an old save is loaded using the monocrystalline PERC cell.

- In the **irradiation** part, meteorological data at the specific location is used as input to simulate the generated photocurrent of each cell of the module. This includes global horizontal irradiance (GHI), diffuse horizontal irradiance (DHI), direct normal irradiance (DNI), ambient temperature T_{amb}, wind speed FF, sun elevation and sun azimuth. In contrast to the cell- and module parameters, the meteorological data will be varied when generating the synthetic data.

- The **thermal** part of the toolbox simulates the hourly cell temperature for all individual cells, which is a very important variable impacting the efficiency of a module. As input it takes the thermal cell efficiency, thermal cell efficiency temperature coefficient and the glass thickness of the module. Again these are all design parameters.

- In the **electrical** part of the simulation, the IV curve is modeled, and the electrical variables (currents, voltages, maximum power point etc.) are determined. As input, data-sheet values of the manufacturer can be used, or a simulated IV curve, which again significantly reduces computation time. For this study, a simulated IV curve for a silicon solar cell is used. An option in this part of the toolbox is also to calculate optical shading losses and resistance due to metalization, using a percentage area loss and the electric resistance of the metal. Also, the number of bypass diodes can be specified, and this part of the model can also perform simulations for tandem cells.

- In the final part of the model, the **conversion** step is simulated. The number of modules in series and -parallel and the percentage of cable loss should be given. A specific inverter can be chosen from a database with the option of a central-, micro-, string inverter or a power optimizer.

In the remainder of this report, all of the design decisions of the system will be referred to as parameters. Parameters are identical for every simulation. Everything that changes when part of the input is changed will be referred to as variables. This includes the changing input variables for weather but also electrical variables such as currents and voltages, or cell temperatures.

3.1.2. Sampling Weather Data

For normal operation of the PV system, the only input variables are the weather variables GHI, DHI, DNI, T_{amb}, wind speed, sun elevation and sun azimuth. Changing weather conditions are used to generate random power output profiles of the PV system. Meteonorm was used as the source for meteorological data. Meteonorm gives hourly values of irradiation, temperature etc., at any given location. It does not provide actual historical measured data but uses nearby weather stations and interpolates based on monthly average data at the given location. Subsequently, a stochastic weather generator simulates a typical year of hourly data using the interpolated monthly data. Although data from Meteonorm is not actual measured data, it suits the purpose of generating realistic weather profiles for the simulation.

For this study, hourly weather data from 2015 to 2020 was used based on the monthly data of the weather station in Cabauw, which can be assumed to be similar to the climate in Delft. By taking data from six different years, enough data is included to generate varied conditions. From this, two consecutive days of weather are randomly sampled as input variables for each run of the toolbox, resulting in 48-hour output profiles. It could occur that a day with very low irradiance, for example due to heavy shading, is sampled, which results in complications in fault detection. By sampling an additional day, the likelihood of this happening is lowered. A more detailed explanation of this is given in section 5.2.1.

3.2. Fault Sampling

Faults in PV systems do not occur randomly. Some faults happen more often early in the lifetime of a system e.g. manufacturing issues. Other faults tend to happen as time increases after the system starts degrading, for example due to corrosion. Therefore, age is an important variable to include in the fault detection model. This section is divided into two subsections. Subsection 3.2.1 discusses how the probability of a fault event is determined by using failure probability distributions, and subsection 3.2.2 describes how these distributions are used to sample system age for the training database generation.

3.2.1. Failure Probability Distribution

Fault occurrence can oftentimes be described by an underlying failure probability distribution. In the case of a continuous distribution, it is represented by a probability density function (PDF). These failure distributions can be constructed by using maintenance data collected for specific components and determining the time to failure (TTF). From the TTF, a best fit reliability distribution can be constructed, which is a distribution of a specific failure against time. Typically, two parameters characterize the distribution, except for an exponential distribution which only has one parameter. The likelihood of a fault occurring changes over time following the failure probability distribution specific to the type of fault. The following distributions are commonly used to describe fault- and failure distributions [31]:

- **Normal distribution**. A bell-shaped curve described by the parameters mean μ and standard deviation σ. The normal distribution has the following PDF:

$$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \qquad x \in \mathbb{R} \tag{3.1}$$

  The mean is the location of the top of the probability density function, and the standard deviation controls the shape of the curve i.e. height and width. A normal distribution can be used for faults and failures of which the failure rate is likely to increase over time, but a decreasing failure rate later, such as corrosion or other failures resulting from chemical processes [43].

- **Lognormal distribution**. A distribution of which the logarithm is normally distributed. Similar to the normal distribution the lognormal distribution is described by a mean and standard deviation. The distribution has the following PDF:

$$f(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left(-\frac{(\ln(x)-\mu)^2}{2\sigma^2}\right), \qquad x > 0 \tag{3.2}$$

  The lognormal distribution is useful for events that are more likely to happen early during the lifetime, and the probability decreases later on. This is represented by the curve having a peak on the left with a tail to the right. It is most often used for repair distributions rather than fault- or failure distributions.

- **Gamma distribution**. This distribution is described by a shape parameter α and a scale parameter θ, or rate parameter β being the inverse of θ. The gamma distribution has the following PDF:

$$f(x;\alpha,\beta) = \frac{\beta^\alpha x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)}, \qquad x \geq 0 \tag{3.3}$$

  where $\Gamma(\alpha) = (\alpha-1)!$, represents the gamma function. The gamma distribution can be used to describe infant mortality failures but is less common in fault- or failure distributions than the other distributions.

- **Exponential distribution**. The exponential distribution is a particular case of the gamma distribution for when $\alpha = 1$. The rate parameter for the exponential distribution is denoted by λ. The exponential distribution has the following PDF:

$$f(x;\lambda) = \lambda e^{-\lambda x}, \qquad x \geq 0 \tag{3.4}$$

  A shape parameter of 1 corresponds to a constant failure rate over time, which is a common assumption made for component failure due to the simplified derivations. The exponential distribution is the only continuous distribution that is memoryless i.e. the probability of an event is independent of the history of a process.

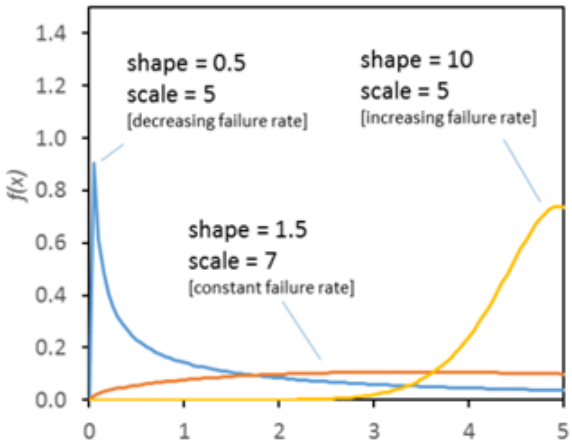


Figure 3.1: Example of Weibull probability density functions used to create a bathtub curve [31].

- **Weibull distribution**. The most versatile distribution of all is the Weibull distribution, as it can deal with decreasing-, constant- and increasing failure rates over time, and can be used in place of other distributions in many cases. Again, this distribution is described by a shape parameter k and a scale parameter λ. The Weibull distribution has the following PDF:

$$f(x;\lambda,k) = \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(\frac{x}{\lambda})^k}, \qquad x \geq 0 \tag{3.5}$$

  A decreasing failure rate is given by $\kappa < 1$, a constant failure rate for $\kappa = 1$ and a increasing failure rate for $\kappa > 1$. A common application of the Weibull distribution is the bathtub curve that is often used in reliability analysis, being a visual representation of the failure rate of a product over time. It is a combination of infant mortality failures, random failures and wear-out failures. The shape resembles a bathtub, as can be seen in figure 3.1.

From literature, appropriate failure probability distributions for the faults described in section 2.2 were found. As the research done on fault statistics is limited, an appropriate distribution was not found for every fault type. Literature mainly provides single component failure rates, which imply an exponential distribution. However, provides best-fit distributions are provided for some components: connector (exponential), inverter (lognormal), bypass diode (Weibull), module (exponential) [27]. Unfortunately, no parameter values for these distributions are listed. Most inverter failure modes are best described by a Weibull distribution [31]. For ground faults and line-to-line faults, the available data is even more limited, and only four specific cases of ground-arc faults are listed, being normally distributed in three of four cases [31].

For a broken module string, the PDF of a connector failure is chosen as failure distribution, whereas for a broken cell string, the PDF of a cell failure is used. Both of these have an exponential distribution best-fit. For the ground- and line-to-line fault, a normal distribution was picked in line with other literature [31], but values are guesstimates. The same was done for inverter failure but with a Weibull distribution. Consequently, the failure distributions for these faults have somewhat arbitrary values but were chosen in line with the limited data available from literature. An overview of the faults with corresponding distributions and parameter values is shown in table 3.1.

Table 3.1: Failure distribution with parameter values of included faults.

Fault	Distribution	Mean/Shape	Scale/St. Dev.	Time Scale
Ground Fault	Normal	~320	~110	days
Line-to-Line Fault	Normal	~320	~110	days
Broken Module String	Exponential	-	$1/(0.00065 * 2\ m_{total})$	years
Broken Cell String	Exponential	-	$1/(0.00087 * m_{total} * c_{total})$	years
Inverter Failure	Weibull	>1	1-2	years

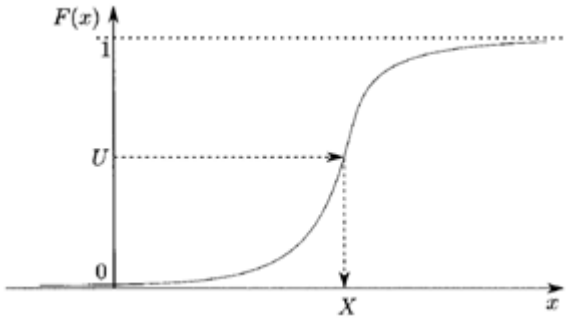


Figure 3.2: Inverse transform method illustrated [44].

In table 3.1, m_{total} is the number of modules in the system and c_{total} the number of cells per module. For these failures, the PV system contains a multiple of the same component, therefore having a higher likelihood of occurring. Faults could be interdependent, and also multiple faults of similar- or different type could occur simultaneously, but for this study, it was assumed not to be the case.

3.2.2. Failure Distribution Sampling

From the failure distributions, the system ages at which the failure occurs are sampled using the inverse-transform method. This method uses random uniform number generation to sample from a probability distribution and works as follows:

Let X be a random variable with PDF f and cumulative distribution function (CDF) F. As F is a non-decreasing function, its inverse F^{-1} can be defined by:

$$F^{-1}(y) = \inf\{x : F(x) \geq y\}, \qquad 0 \leq y \leq 1 \tag{3.6}$$

Subsequently, let $U \sim U(0,1)$ from which random uniform numbers $U_1, ..., U_k$ are generated, with k being the amount of generated numbers. The CDF of the inverse transform $F^{-1}(U)$ is then given by

$$\mathbb{P}(F^{-1}(U) \leq x) = \mathbb{P}(U \leq F(x)) = F(x) \tag{3.7}$$

Thus, to generate a random variable X from PDF f, first get the CDF F. Then take a random uniform sample U and set $X = F^{-1}(U)$. An illustration of this process is shown in figure 3.2.

Where the PDF is at its highest, the slope of the corresponding CDF will be most steep. It is around this point where most numbers are generated as this part of the CDF covers most of the domain of U. If k is made large enough, the distribution of generated variables follows the original PDF f. Consequently, a set of system ages at which failure is most likely to occur is sampled from the probability distribution specific to a certain fault type or failure.

However, there is no such probability distribution for healthy systems to sample from. Therefore, data on the age of existing installed system power is used [45]. The data can be approximated with an exponential distribution with $\mu = 3.0568$. The data is very much skewed to younger systems because most PV systems were installed in recent years. As the aim of this study is to develop a detection algorithm for existing systems, this should not be a concern. It is good to keep in mind a large number of the installed power comes from non-residential systems: for 2021, the subsidized installed power for non-rooftop systems is already half of the total installed power of 2020 [45]. From the rooftop systems, a large share of installed systems is also non-residential. However, the assumption is made that the installed residential system power follows a similar trend as the total installed PV power.

Note on Broken Module String Probability Distribution

Using the failure distribution for a broken module string given the parameters in table 3.1, results in a range of failure ages far beyond the usual warranty of 25 years. The time scale is entirely different from the already

installed systems used for the healthy system sampling. In contrast, the other fault distributions have a relatively similar time scale to healthy systems. A new probability distribution has been constructed to address this problem by taking the product of the broken module string PDF and the healthy system PDF. Given two gamma distributions

$$f_1(x_1;k_1,\theta_1) = \frac{x_1^{k_1-1}e^{-x_1/\theta_1}}{\theta_1^{k_1}\Gamma(k_1)} \tag{3.8}$$

$$f_2(x_2;k_2,\theta_2) = \frac{x_2^{k_2-1}e^{-x_2/\theta_2}}{\theta_2^{k_2}\Gamma(k_2)} \tag{3.9}$$

where k_1, k_2 and θ_1, θ_2 are the shape and scale parameters of distribution 1 and 2 respectively, and Γ is the Gamma function. Subsequently, the product of these two distributions $z = x_1 x_2$ is given by

$$p_Z(z) = \frac{2}{\Gamma(k_1)\Gamma(k_2)} \frac{z^{\frac{k_1+k_2}{2}-1}}{(\theta_1\theta_2)^{\frac{k_1+k_2}{2}}} K_{k_1-k_2}\left(2\sqrt{\frac{z}{\theta_1\theta_2}}\right) \tag{3.10}$$

which in the case of two exponential distributions simplifies to

$$p_Z(z) = \frac{2}{(\lambda_1\lambda_2)} K_0\left(2\sqrt{\frac{z}{\lambda_1\lambda_2}}\right) \tag{3.11}$$

where λ_1 and λ_2 are the rate parameters of the exponential distributions, and K is a second kind modified Bessel function. Distribution 3.10 is also referred to as a K-distribution.

The system age is implemented in the simulation by including a degradation rate factor. Degradation is highly dependent on climate conditions and therefore varies significantly for different locations. Irradiance and temperature are two of the important factors of degradation, and the rate of change of these two variables. For this study, a simplified degradation factor is used, with a rate in the range of -1.05%/year to -1.16%/year is used [46]. This study about degradation of residential rooftop systems in the UK can be used as a rough degradation estimate as the climate is comparable to the Netherlands. The assumption is made that the degradation of a single system is consistent through the years. Degradation rate is randomly picked under the assumption that degradation rate is normally distributed with μ being the average of the found range and σ derived from the range rule for standard deviation:

$$\sigma = \frac{\text{Max - Min}}{4} \tag{3.12}$$

The inclusion of a more complex degradation model dependent on climate conditions in the PVMD Toolbox is currently ongoing and can thus be used in future research for a more accurate implementation of degradation for the synthetic data generation.

3.3. Faulty System Operation

The PVMD Toolbox was developed to precisely calculate the annual energy yield of a PV module, including all types of (novel) PV technologies and technologies that are still in development. It involves very in-depth optic simulations and is ideal for going from cell to module level. At the system level however, it is not yet very extensive; the modeled module is essentially extrapolated to the system level with no distinction between the different modules in the system. A suitable inverter can be chosen from a large library, but those are all the options for the system-level design. As all faults described in 2.2 involve irregular module operation across the system, different solutions to incorporate these faults in the toolbox were implemented. The solution for each fault will be discussed in the remainder of this section.

3.3.1. Ground Fault

Modeling a ground fault and its impact on the PV system starts by determining the fault current I_{fault}, which is also the current going through the GFPD. By applying Kirchoff's voltage law (KVL) and Ohm's law on the circuit, the following equation for I_{fault} is derived:

$$I_{fault} = \frac{I_{MPP}(R_x + pR_{comb}) - nV_C}{R_{GFPD} + R_{EGC} + R_{fault} + R_x + R_{comb}} \tag{3.13}$$

where R_{GFPD}, R_{EGC}, R_x, R_comb and R_{fault} are the resistances of the GFPD, equipment grounding conductor (EGC), cabling up until fault point, combiner box and fault respectively. I_{MPP} is the string current which is approximately at maximum power point, n the number of faulted modules, P the number of strings, and V_C the voltage of the modules below the fault. For the remainder of this section, modules in a non-faulted string will be assigned the letter A, modules in the faulted string above fault point the letter B, and modules in the faulted string below fault point the letter C. By varying R_{fault} and also varying n, a range of different ground faults is generated. Using Kirchoff's current law (KCL) as well as KVL and Ohm's law, the rest of the circuit is modeled consequently:

$$V_{fault} = I_{fault} * R_{fault} \tag{3.14}$$

$$V_C = \frac{I_{MPP}(R_x + pR_{comb}) - I_{fault}(R_{fault} + R_{GFPD} + R_{EGC} + R_{fault} + R_x + R_{comb})}{n} \approx \frac{V_{fault}}{n} \tag{3.15}$$

$$V_A = \frac{nV_C + (m-n)V_B}{m} \tag{3.16}$$

$$I_B = I_C - I_{fault} \tag{3.17}$$

where m is the number of modules per string. The initial conditions are that I_A and I_C operate at I_{SC} and V_B is equal to V_{OC}. As can be seen in figure 3.3 the operating point of the modules shift towards short circuit operation for A and C, while for B, it shifts towards open circuit operation. For every time step, the IV curve is constructed, and after the above equations, I_A, I_C and V_B are recalculated by taking the closest point on the IV curve for that specific time step for the calculated V_A, V_C and I_B respectively. Consequently, the parameters of the complete array are:

$$I = I_A + I_B \tag{3.18}$$

$$V = mV_A \tag{3.19}$$

The above procedure is carried out for every time step i.e. for every hour. The IV curve will change constantly depending on the meteorological conditions and time of the day. Therefore all current- and voltage parameters are time dependent. This incorporates the idea that some faults can go unnoticed due to low irradiance levels while still having a significant impact on the power output of the system.

As one might have noticed, I_{fault} is dependent on V_C but also vice versa as given by equations 3.13 and 3.15. To properly calculate I_{fault} dependent on R_{fault}, n and V_C, extra information is necessary but can't be provided by the rest of the circuit. Also, solving iteratively. the system appears not to converge. To solve this problem, we identify the desired results for these four parameters in extreme cases:

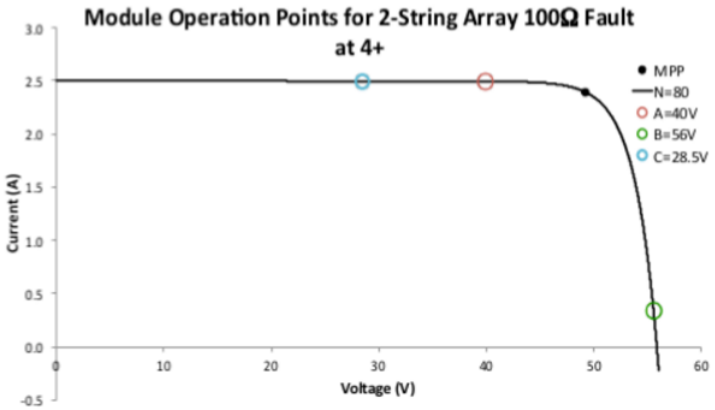


Figure 3.3: IV-curve of 2-string array with ground fault involving 4 modules of 100 Ω [32].

$$\lim_{R_{fault}\to 0} I_{fault} = \frac{n}{m} p I_{SC} \tag{3.20}$$

$$\lim_{R_{fault}\to 0} V_C = 0 \tag{3.21}$$

$$\lim_{R_{fault}\to \infty} I_{fault} = 0 \tag{3.22}$$

$$\lim_{R_{fault}\to \infty} V_C = V_{MPP} \tag{3.23}$$

The former two equations yield that for an infinitesimal fault resistance, the fault current approaches short-circuit current times the number of strings and decreases linearly with the number of involved modules. In this case, the current of all modules in the complete system is fed into the fault. This also means that the modules below the fault operate at short-circuit current and thus at zero voltage. The latter two equations yield that the fault current approaches zero for an infinitely large fault resistance, as there is essentially no fault in place. Therefore the limit of the fault voltage is of indeterminate form and can be any number. However, in this case the modules below the fault should operate normally and thus at MPP voltage. With this in mind, the fault voltage could be determined. By combining the knowledge of these extreme cases with the former equations describing the system, a good representation of the ground fault behavior can be modeled at any given time step. An example of the results of the ground fault simulation at a single time step is shown in figure 3.4

For this study, the interest lies in faults with I_{fault} smaller than the fuse rating, as these faults persist and decrease the array efficiency without going noticed by the installed GFPD. As can be seen in figure 3.4, with a fuse size of $1\,A$, this would be in the range of 20Ω and higher and is far more likely when fewer modules are involved in the ground fault. Of course, when irradiance is lower, the I-V curve at that instance is different, and the fault current will be lower. Consequently, the resistance range of interest shifts towards lower resistance. Besides the condition of I_{fault} being smaller than the fuse rating, the fault resistance R_{fault} was sampled in a range from 1 Ω to 1000 Ω. Outside this resistance range, the system behavior does not significantly change anymore, as is visible in figure 3.4.

The backfeeding phenomenon described in section 2.2.1 will not be of importance for scenarios created for the purpose of this study, since this only occurs for large enough faults which the GFPD will already detect. During the simulation of the Toolbox, once the fault current exceeds the fuse rating, a new R_{fault} and n are randomly picked, repeating the simulation with these new parameters, such that a scenario in the scope of this study is generated.

3.3.2. Line-to-Line Fault

For the modeling of line-to-line faults, a procedure similar to that of a ground fault described in section 3.3.1 was followed. By varying R_{fault} and the location of the fault, a range of line-to-line faults is generated. Compared to a ground fault, the difference here is the lack of a connection with ground, which was used as a reference voltage potential. Now, contact is made between two strings, complicating the circuit, as modules of both strings are involved. Simply combining Ohm's law, KVL one arrives at the following equation for I_{fault}

$$I_{fault} = \frac{kV_C - nV_A}{R_{fault}} \tag{3.24}$$

where n and k are the number of modules below the fault in strings 1 and 2 respectively, and V_A and V_C are the corresponding voltages of these modules. Once n and k are not equal, there will be a potential difference between the strings when operating at MPP. For this study, n was chosen always to be the smaller number than k. Consequently, just as for the ground fault, the operation point of the modules will shift away from their MPP. As n is smaller than k, modules A will shift to a higher operating voltage ($V_A \in [V_{MPP}, V_{OC}]$) while modules C will lower the operational voltage and shift towards I_{SC} ($V_C \in [0, V_{MPP}]$). The severity of a line-to-line fault is, besides fault resistance, mainly determined by the difference between n and k and hence the potential difference between strings at the fault point. The OCPD is highly likely to trip when the difference is higher than one [36]. The open-circuit results in a significantly lower power output, especially for a 2-string system. Similarly to the ground faults, interest lies in the faults that are going undetected by the conventional

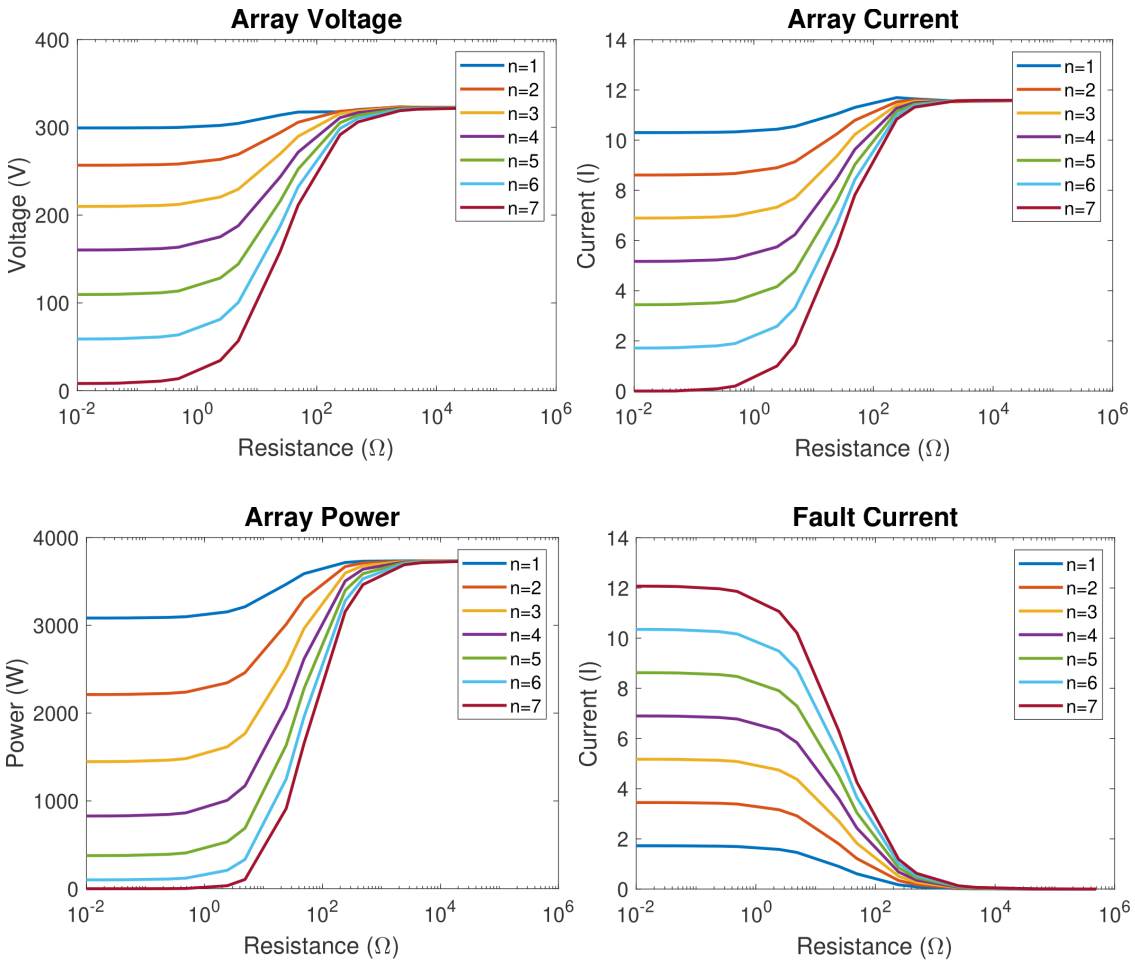


Figure 3.4: Results of Toolbox simulation of a ground fault in a 2-string array involving 1 up to 7 faulted modules at a random time step. For this specific time step the module parameters are: $I_{MPP} = 5.80A$, $I_{SC} = 6.04A$, $V_{MPP} = 46.04$ and $V_{OC} = 51.65$. The system operates close to optimal for faults with a resistance of 1000 Ω or up. For lower resistance faults, both array current, voltage and power decrease significantly, approaching zero for faults smaller than 1 Ω. Fault current increases almost linearly with the number of faulted modules and approaches 2 I_{SC} in worst-case scenario of low fault resistance and maximal modules involved in the ground fault.

protection devices, but that could potentially be harmful when the fault develops over time, rather than the faults that are cleared already and are easy to detect without the need of an elaborate model.

Once V_A and V_C are known, I_A and I_C are obtained by projection of V_A and V_C on the I-V curve and taking the closest point. The rest of the circuit is calculated by the following equations using KCL and Ohm's law:

$$I_B = I_A + I_{fault} \tag{3.25}$$

$$I_D = I_C - I_{fault} \tag{3.26}$$

$$I = I_B + I_D \tag{3.27}$$

$$V_1 = nV_A + (m-n)V_B \tag{3.28}$$

$$V_2 = nV_C + (m-k)V_D \tag{3.29}$$

where I_B and I_D are the current of the modules above the line-to-line fault for strings 1 and 2 respectively. V_B and V_D are once again obtained by projection on the I-V curve. The string voltages V_1 and V_2 are calculated by summation of modules A and B and modules C and D respectively. To find P_{DC} for a given line-to-line fault, an optimization problem has to be solved for variables V_A and V_C where the objective function is:

$$\max_{V_A, V_C} \quad P_{DC} = IV = (I_B + I_D)(nV_A + (m-n)V_B) \tag{3.30a}$$

$$\text{subject to} \quad V_1 = V_2 = V \tag{3.30b}$$

$$V_{MPP} \leq V_A \leq V_{OC} = \tag{3.30c}$$

$$0 \leq V_C \leq V_{MPP} = \tag{3.30d}$$

In other words, the combination of V_A and V_C that satisfies the equality constraint 3.30b as well as boundary conditions 3.30c and 3.30d is found by maximizing the power produced by the system P_{DC} for objective function 3.30a. All variables in the optimization problem are indirectly dependent on either V_A, V_C or a combination of both via equations 3.24 - 3.29 and the intermediate steps of projection on the I-V curve.

The whole procedure described in this section is followed for every time step as the I-V curve is changes over time. Simulation shows that for small values of R_{fault}, especially in combination with low irradiance levels, a solution can not always be found. As the procedure works perfectly for all other conditions, the minimum irradiance for this procedure was set at 50 W/m^2. This almost only occurs either for the hour after sunrise and the hour before sunset under two conditions: low irradiance (cloudy/winter) or when the first or last hour in practice is far less than a full hour (sunrise on a clear July 28th in Delft occurs at 05:58 so the sixth hour of the day will only have sun for two minutes out of the whole hour). To account for the decrease in power as a result, the current and voltage at these time steps are calculated by multiplication of I_{MPP} and V_{MPP} by the ratio of I_{MPP} to I and V_{MPP} and V of the closest hour with enough irradiance. This would be the next time step for the hour during dawn or the prior time step for the hour during dusk. This is not a perfect solution, but the ratio of V_{MPP} and V is fairly consistent throughout the day. The ratio of I_{MPP} and I varies more where it is close to 1 except for the hours of concern with very low irradiance. By taking the closest hour i.e. an hour after dawn or an hour before dusk, this is covered partially as for these hours, irradiance will still be relatively low due to the low sun position.

3.3.3. Broken String Fault

The broken string fault and its impact on the physical system are rather straightforward compared to the ground- and line-to-line faults. In the toolbox, the generated current density was repeated for the number of strings and modules in series. In this case, the current density array size is 48x72: 72 cells in a module (columns) and 48 hours simulated (rows). After repeating the array this is increased to 2x7 blocks of 48x72 i.e. 96x504. By correct indexing, the decrease in power of the affected cells/modules/string is accounted for by setting the current density to 0 i.e. no power is produced. To account for the voltage mismatch for fault cases

2 and 3 of section 2.2.3, the current density is decreased by the corresponding voltage decrease. The number of by-pass diodes is included in the model being 3 for the studied system..

For the toolbox to be able to do the further necessary simulations, the array has to be of the same size as before again, 48x72. The mean of all 14 modules is taken for every cell and time respectively. Consequently, a module representing system's average is used to run the rest of the simulation. As there is only the current density array to work with, the voltage can't be changed, so this is done by adjusting the current density. In the end, the power generated by the array is just the multiplication of current and voltage, therefore resulting in the same power reduction. However, by doing so, the temperature simulation of the toolbox will not be correct completely. The approach of repeating the array, manipulating of the current density, and combining the modules again is not completely necessary, but it gives a good representation and visualization of how the system is impacted, as can be seen in figure 3.5

Note on Broken Cell String Fault

For the broken cell string fault, the expected time to failure is low. In the system used for this study it is 1.14 years. Because this type of fault is likely to occur multiple times during the lifetime of a PV system , a different approach for this type of fault is considered. The system age is sampled similarly to the age sampling for healthy systems described in section 3.2.2. Consequently, failure ages are sampled from the broken cell string probability distribution and added until the sampled system age is reached. Because a new broken cell string might occur in the same cell string that has already been affected by an earlier fault, each time the chance of a broken cell string fault impacting the system performance slightly decreases. Hence the failure age of consecutive faults slightly increases. The occurrence of multiple broken cell strings is also accounted for in the resulting voltage mismatch between the module strings. Consider a 2-string system with a single broken cell string in each module string. In this case, the string voltage of both strings is decreased equally, and there is no voltage mismatch. In this case, a correction in string voltages would not be necessary. The new fault compensates for the string voltage correction resulting from the initial fault in the other string. This results in no change in performance after the second fault compared to the situation before the second fault. In fact, the string voltage correction is linearly dependent on the difference in number of broken cell string faults between the specific string and the string with the highest number of broken cell string faults. In the given example this difference goes from 1 to 0 after the second fault occurs.

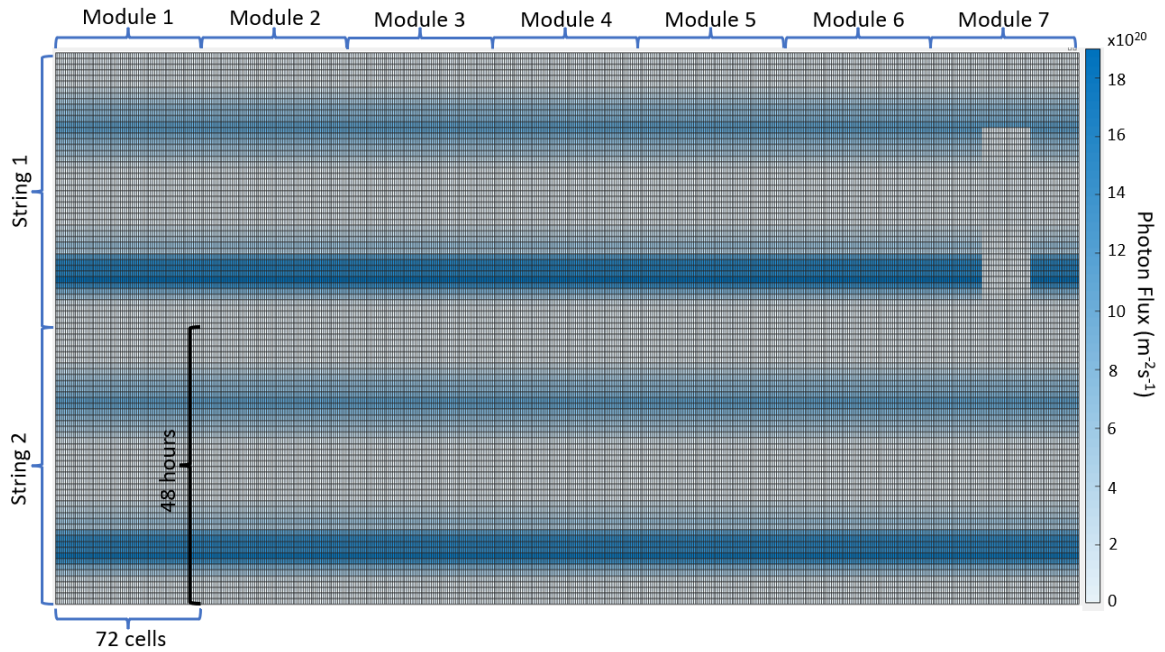


Figure 3.5: Visual representation of the photo-generated current density of the PV system with a broken cell string in string 1, module 7, cell string 2, with three by-pass diodes per module. A fault event occurs after 14 hours. The photon flux in string 2 is slightly lower than string 1 after 14 hours due to the string voltage mismatch but is invisible to the eye in this representation.

3.3.4. Inverter Faults

As discussed in section 2.2.4, inverter failure-related causes vary widely, and the suggestion of introducing a 0% efficiency fault category, rather than incorporating all the different inverter faults, was made. This fault category includes all faults that would trip the inverter, from lightning strikes to inverter component failures. Unless the cause for inverter failure is already shown by the inverter itself, a complete shutdown of power output of the system is reason enough to have a thorough investigation of the system. Therefore this fault category does not necessarily have to focus on the cause upfront. Incorporation of the 0% fault category in the Toolbox is thus very easy: $P_{DC} = 0$.

3.4. Synthetic Database Result

With the implementation of faults in the PVMD Toolbox described in section 3.3, the different system scenarios can all be simulated. In figure 3.6 the different fault classes are compared for the same system under the same weather conditions. For every scenario, the relevant parameters for the training of the machine learning model are extracted from the simulation results, including the meteorological conditions, system power output and system age. The final training database consists of 25000 different scenarios. For each scenario, system status (healthy or type of fault), a 48-hour weather profile including irradiance, sun elevation, sun azimuth and ambient temperature and a 48-hour power output profile is stored in the database. These profiles consist of hourly values for the respective variables. Of the database, 60% comprises healthy operating systems while the other 40% concerns faulty systems. The faulty scenarios involve ground faults, line-line faults, broken string faults and broken cell faults, all accounting for 25% of the faulty data points. Zero efficiency faults were also included in the database in early stages of this study but have been removed as the performance on this fault class was indeed 100% and therefore not very interesting.

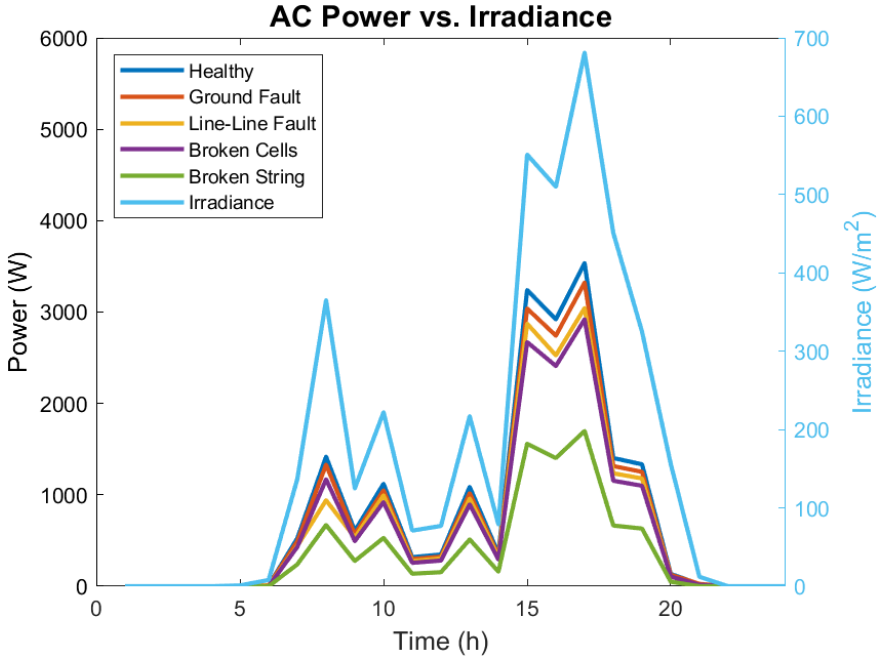


Figure 3.6: AC Power output for different operating states compared to irradiance (in $\frac{W}{m^2}$). Ground fault is at location n=3 and R_{fault} = 5 Ω. Line-line fault is at location n=1 and R_{fault} = 5 Ω. Broken cell fault involves 4 broken cells of which 2 in each string. All states are under the exact same weather conditions and zero degradation.

3.5. Conclusion

The second sub-question "How can a synthetic PV training database be generated for PV fault detection and fault classification in residential PV systems?" can now be answered. By using the PVMD Toolbox in Matlab, a representative healthy PV system can be modeled, and the power output can be modeled accordingly. By varying the meteorological conditions and system age that affects, different scenarios for the same system

can be simulated. Weather data is taken from Meteonorm, and 48-hour weather profiles are sampled. The system age is sampled from the failure probability distribution of specific faults or from the age curve of already installed residential PV capacity for healthy systems. Manipulating the data in between simulation steps of the Toolbox makes it possible to implement faults in the system, and hence faulty systems can be simulated. By repeating the simulation of a single system multiple times, for varying conditions and varying system status, a database can be constructed with many different scenarios of the same system. For every scenario, the relevant parameters for the machine learning algorithm are stored in the database, including system age and 48-hour profiles for irradiance, sun elevation, sun azimuth, ambient temperature and power output. By balancing the amount of simulated healthy and faulty systems, this synthetic PV training database can be used for machine learning model training for fault detection and classification.

4

Machine Learning Model Theory

As alluded to in chapter 1, machine learning can be an extremely powerful tool for making predictions and do classifications. For this reason, the generated synthetic data as described in 3 can be classified as healthy/faulty and subdivided into several fault types using ML algorithms. This chapter aims to answer the sub-question "How to design a machine learning algorithm for PV fault detection and classification?" and provides theory on machine learning algorithms, performance metrics and model optimization. This chapter does not show any results, and all results are discussed in chapter 5.

Basic theory and concepts of machine learning that are necessary to understand the machine learning model design and results in chapter 5 are explained in section 4.1. Section 4.2 discusses the proposed machine learning algorithms for this study and also describes how to optimize the machine learning models. In section 4.3 the third sub-question is answered.

4.1. Theory

Machine learning or automated learning refers to the automatic detection of patterns in data and has the ability to learn from data. It is especially purposeful for large data sets that are often too complex for humans to extract useful information from. Learning is something observed everywhere in nature with. The best known example is Pavlov's classical conditioning experiment: dogs would learn that the ringing of a bell was synonymous with receiving food and consequently, after repeating the experiment, would already start to salivate hearing only the sound of the bell. The dogs learned through their memorized previous experiences. Similarly, a machine could be taught specific tasks through experience. For example, automated spam filter detection is able to label a new e-mail as spam or not, based on previous e-mails being labeled as spam or not. When the new e-mail is similar to a previously received e-mail that was marked as spam, the spam filter recognizes that this new e-mail is likely also to be spam and puts it in the spam folder. Based on certain words and combinations, a completely new and never seen before e-mail can be marked as spam successfully, as specific words are more likely to be contained in spam than in regular e-mails or vice versa. Other indications of spam could be mails without a subject or obscure mail addresses.

Features, Training- and Test Data

The indicators for a spam filter to detect an e-mail as spam or not are referred to as features. Based on a set of features, a machine performs its task accordingly. Some features might have more correlation with the outcome than others. In our example, as more and more e-mails are received and evaluated by the spam filter, the detection accuracy improves. As more data is used, the machine learns more and consequently, the performance should theoretically increase. This is called the training of a model. Training data is used to train the model so that the model can perform its task on new unknown data. This new data is referred to as test data.

A training data set used to train an ML model is denoted by X being a matrix with entries $X_{i,j}$, with X being and $n * p$ matrix. Then i are the data points/samples/observations, and j are the features. The total number of observations equals n, and the number of features equals p. In thespam filter example, there are n e-mails used to learn from and p indicators for spam, such as specific words or word combinations.

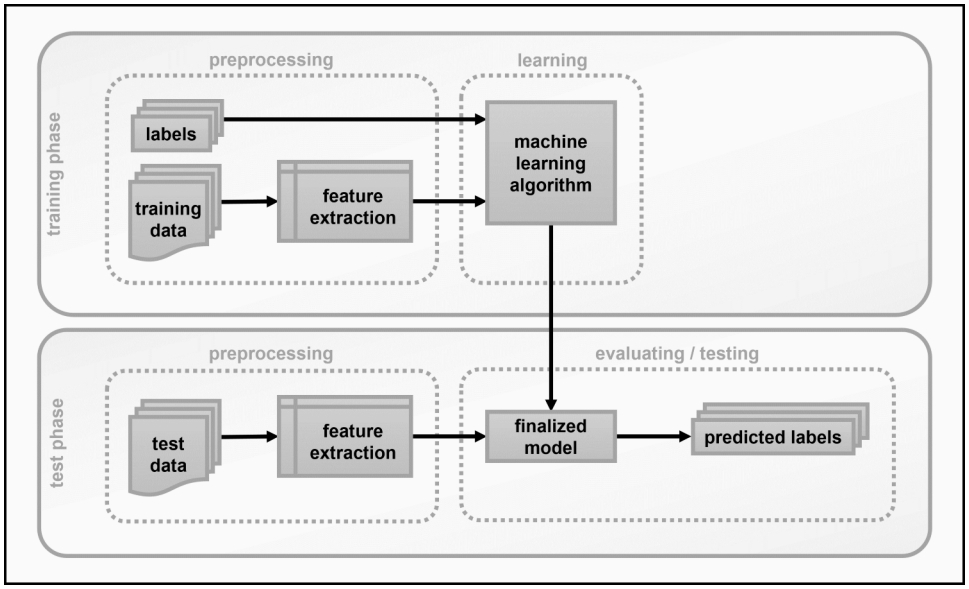


Figure 4.1: Workflow diagram of machine learning model [31].

The output of an ML model is depicted by vector y of length n containing the known output of the training samples. This output is also referred to as labels. Similarly there should also be a test data set, which looks similar to the training data set except without labels: the output vector is empty, and the labels have to be determined by the model. A schematic workflow diagram of an ML model is shown in figure 4.1

4.1.1. Supervised- versus Unsupervised Learning

In machine learning, a distinction can be made based on how a model learns from data: supervised learning and unsupervised learning. For supervised learning, training data is always labeled. For example, a spam filter has been trained with many e-mails that are labeled beforehand as either spam or non-spam. Labels can be discrete as in our spam filter example, but can also be continuous. The test data is also labeled. The task of a model is to label the test data accordingly, and by comparing the model predicted labels with the actual labels, the performance can be evaluated. Contrary to supervised learning, unsupervised learning algorithms are trained using unlabeled data. Therefore, unsupervised learning is more fuzzy and tries to cluster the data in groups with similar features. An example of a task that could be performed using an unsupervised learning algorithm is anomaly detection. The unsupervised learning model tries to find patterns in the data set, and depending on whether the test data fits the pattern, it can be marked as an anomaly.

4.1.2. Classification and Regression

Supervised learning can be further subdivided into two categories. The difference lies in the type of output data in y. A regression problem deals with quantitative, continuous output data. Consequently, the outcome of a regression problem is also quantitative. Examples of regression are power generation forecasting and market price forecasting. With a classification problem, output data is discrete and falls into a finite set of categories. Output data does not always have to be quantitative but could also be qualitative. An example of qualitative classification is the spam filter example described earlier. Output is either "spam" or "non-spam". Usually, qualitative data is translated to quantitative data, in this case [0,1], where "0" represents "non-spam" and 1 represents "spam". This is a binary classification problem where output can be one of two classes. A problem with more than two classes is referred to as multiclass classification .

This study aims to detect faulty operation in residential PV systems and classify the type of fault. Therefore, both binary and multiclass classification is necessary. First, the different fault types are grouped together as one 'faulty' class and the regular operating systems as the other 'healthy' class. With binary classification, it is evaluated if it is possible to detect and separate these two classes. After that, the different fault types are all considered separate classes. With multiclass classification, it is identified if the proposed method can also classify the correct fault type. In the following section, the ML algorithms considered are summarized.

4.2. Machine Learning Algorithms

Four different machine learning algorithms were tested for this study: decision tree, random forest, support vector machine and neural network. All these algorithms are capable of both binary and multiclass classification. The decision tree is the most simple method, and in the field of PV fault detection with machine learning is often referred to as a benchmark method [10]. The other three are more advanced, which compared to a decision tree should theoretically improve performance at the cost of computation time and comprehensibility. In section 4.2.1 a brief explanation on each algorithm is given. After the construction of an ML model, the model is optimized to increase model performance, which is discussed in section 4.2.2.

4.2.1. Machine Learning Model Overview

In this section, a summary of the machine learning algorithms of this study is given.

Decision Tree

One of the simplest machine learning algorithms to understand is the decision tree, as it's easily visualized. A decision tree starts with a root node, representing the complete dataset. The root node is then split into two internal nodes. These internal nodes are split again and again until a stopping criterion is met. The last nodes are called the leaf nodes. Splits are determined by a cut-off value of a particular feature: all data points with a value lower than the cut-off value for this feature are put in one internal node, and all data points with a value higher than the cut-off in the other node. An example of a simple decision tree is shown in figure 4.2.

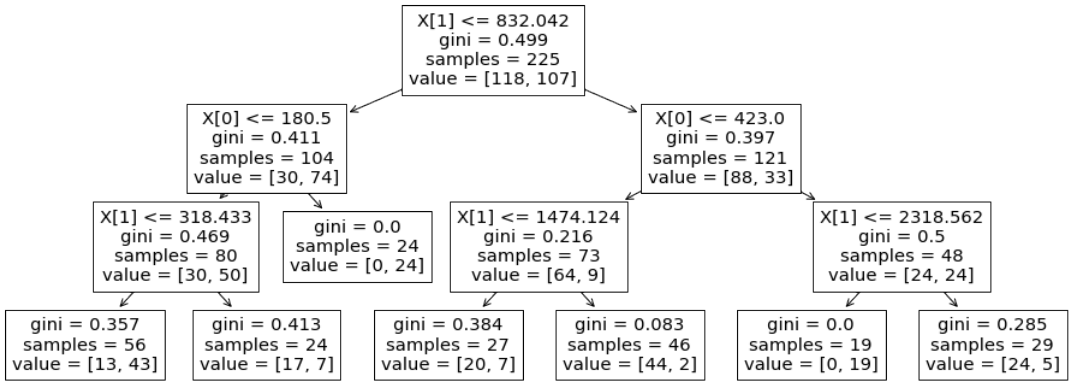


Figure 4.2: Schematic of a decision tree with two features X[0] and X[1] with a depth of three.

The first split is created based on the value of feature X[1] being higher or lower than 832.042. In this case, the total number of data points or samples is 225, with 118 belonging to one class and 107 to the other class. The first internal nodes consist of 104 and 121 samples respectively, and are further split two more times, except for one node on the left side of the tree, which is already a so-called pure leaf. A pure leaf consists of data points of only a single class. In this example, it consists of 0 samples of one class and 24 of the other class. There is no need for further splitting in this case. Splits are not created randomly. At every node, the best splitting value is found for every feature, and from these, the best feature to split the node is chosen. The best splitting is based on the Gini-value, which is a measure of the purity of the node. A Gini of 0 implies a pure node only consisting of one class. On the other hand, a Gini of 0.5 is the most impure node with a fifty-fifty split between classes. The idea is that with each split, the weighted average Gini value of new nodes decreases and subsequently, the nodes get purer. In theory, the splitting procedure could proceed until all leaf nodes are perfectly pure. In the case of the decision tree in figure 4.2 however, the stopping criteria was a tree depth of three. Other stopping criteria could be a minimum number of samples in a node required to split the node or a minimum number of samples in a leaf node.

The decision tree from figure 4.2 can also be visualized by the decision space shown in figure 4.3. The features X[0] and X[1] are respectively the irradiance incident on a PV system and the AC power produced by this system. Therefore, the first split was made based on the system AC power being higher or lower than 832.042 W. This decision boundary is visible in figure 4.3 as a horizontal line and splits the decision space

into two half-planes. Consequently, subsequent splits are also visible: the two vertical lines are the splits based on irradiance at 180.5 W/m^2 at the left side of the decision tree and 423.0 W/m^2 at the right side of the decision tree. As the number of splits increases, not all splits can be visualized anymore in a single figure as the half-planes overlap. Therefore, the split at AC power of 1474.124 W is not visible in figure 4.3. The dot colors represent the actual operating status of systems being either 'regular' or 'faulty'. All systems above the decision boundary are predicted to be 'regular' while all systems below are predicted to be 'faulty'. Clearly, not all systems are predicted correctly, but with increasing complexity of the decision tree, the performance increases.

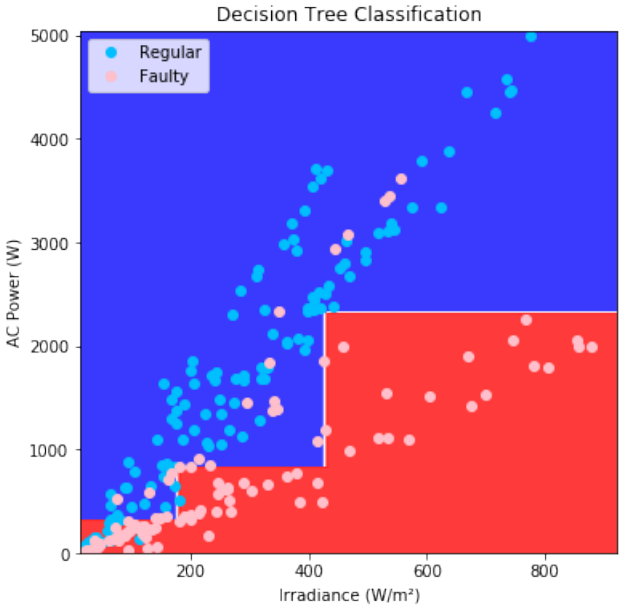


Figure 4.3: Decision space of a decision tree classifier with two features X[0] and X[1].

Random Forest

A random forest is a collection of randomized decision trees. The trees are randomized in two ways. First, every tree is trained on only a subset of the features. Second, every tree is not trained on the complete number of samples, but samples are drawn by bootstrapping i.e. random re-sampling of the training data with replacement. Eventually, the results of all decision trees are combined and for every sample, the most predicted class across all trees is picked as label for this sample. Compared to a decision tree, a random forest generally is more accurate. Because the subset of features is picked randomly, the random forest is less dependent on a specific subset of features than a decision tree would be. As for a decision tree, every split is made to improve the purity, the decision tree will perform very good on training data. However, this often causes the decision tree to generalize poorly and performance on test data is worse than a random forest, due to better generalization capability.

Support Vector Machine

A support vector machine (SVM) works differently from a decision tree, and classes are separated by a multidimensional plane or so-called hyperplane. With a feature space of size p, the hyperplane would be $p-1$-dimensional. The hyperplane can take different shapes depending on the kernel type, which could be linear, polynomial or more. Figure 4.4 shows a representation of the decision space of an SVM with a linear kernel and only two features. Thus, the hyperplane is one-dimensional.

In this example, the same data has been used as for the decision tree in figure 4.3. However, the feature values have been scaled, which is explained in 5.1. In this case, the data is not linearly separable. For linearly inseparable data, there will always be some points on the wrong side of the hyperplane. The larger the distance between these points and the hyperplane, the larger the loss. The combined loss of all incorrectly predicted data points is tried to be minimized in order to find the best hyperplane. For this study, the

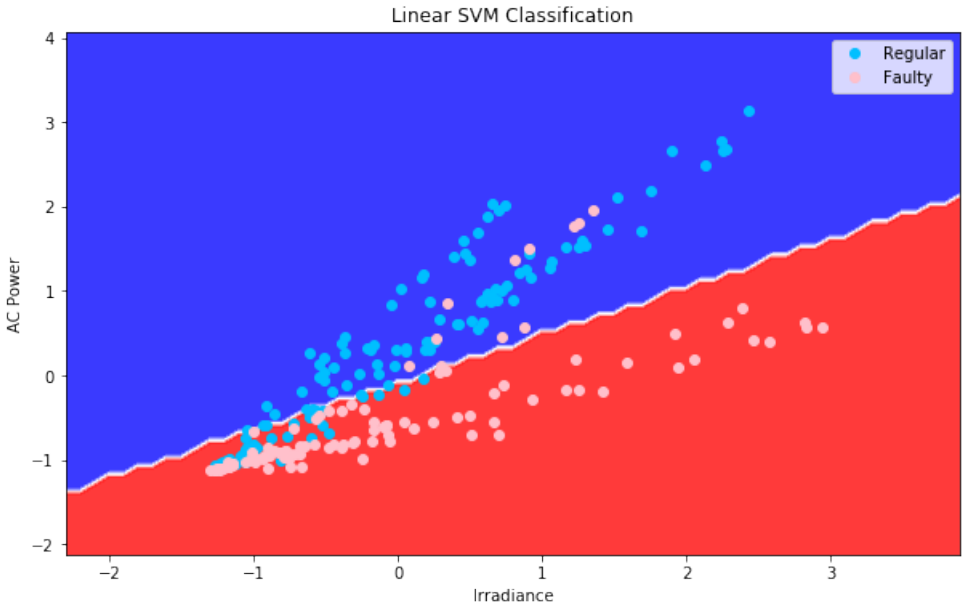


Figure 4.4: Schematic of a linear SVM classifier with two features X[0] and X[1].

Gaussian RBF kernel is used as it outperformed the other kernel options. Comparing the decision space of the SVM compared to that of a decision tree in figure 4.3 the difference between these algorithms becomes clear. The support vector machine only uses a single decision boundary, while the decision tree can use a theoretically infinite number of decision boundaries. However, the decision tree is always limited to vertical and horizontal decision boundaries based on single feature values, which is not the case for a support vector machine.

Neural Network

Lastly, the neural network is an algorithm that seeks to mimic the working of the human brain where a complex network of neurons transmits information, and doing so is capable of learning from given information. The neural network type used in this study is a multilayer perceptron (MLP) feedforward artificial neural network (ANN) using backpropagation as learning technique. This type of neural network consists of an input layer, one or multiple hidden layers and an output layer. The input layer consists of p inputs, equal to the number of features. The hidden layer consists of a manually chosen number of neurons K and the output layer consists of a number of neurons equal to the number of classes. Each neuron has inputs and produces a single output. A neuron in the first hidden layer has an input from all neurons in the input layer and, through an activation function, produces a single output. The activation function is an essential part of the learning process of a neural network and decides what is to be communicated to the next neuron. The most simple activation function is a step function where the neuron either fires all information to the next neuron or no information at all. Other activation functions are for example the hyperbolic tan function f(x) = tanh(x) or the rectified linear unit (ReLu) f(x) = max(0,x). In this study, ReLu was chosen as the activation function as it is the most commonly used activation function in machine learning. ReLu gives a similar output as input when input is positive and zero when input is negative. The process of a single neuron in the hidden layer is as follows:

$$x = (x_1, ..., x_n)^T \in \mathbb{R}^n \tag{4.1}$$

$$w = (w_1, ..., w_p)^T \in \mathbb{R}^n \tag{4.2}$$

$$b \in \mathbb{R} \tag{4.3}$$

$$a_i = g(w_i \cdot x + b_i), \qquad i = 1, ..., K \tag{4.4}$$

where x is a vector containing all inputs from the input layers, w_i a vector containing the weights of the p input neurons connected to a neuron i in the hidden layer, b_i the bias of neuron i and g the activation function. The output layer acts similar to the hidden layer, and its output is the predicted class to which a data point belongs as follows:

$$a = g(W \cdot x + b) \tag{4.5}$$

$$f(x) = g^{out}(w^{out} \cdot a + b^{out}) \tag{4.6}$$

where a is a vector of outputs of the neurons in the hidden layer, W a weight matrix, b a vector of the bias of the neurons in the hidden layer, g^{out} the activation function of a neuron in the output layer, w^{out} a weight vector of the outputs of the neurons in the hidden layer and b^{out} the bias of the neuron in the output layer. Then, $f(x)$ gives us the predicted label Y. The described process is for a multilayer perceptron neural network (MLP-NN) with a single hidden layer. However, neural networks can exist with multiple hidden layers. A neural network with two hidden layers is shown in figure 4.5 which also illustrates the described process above.

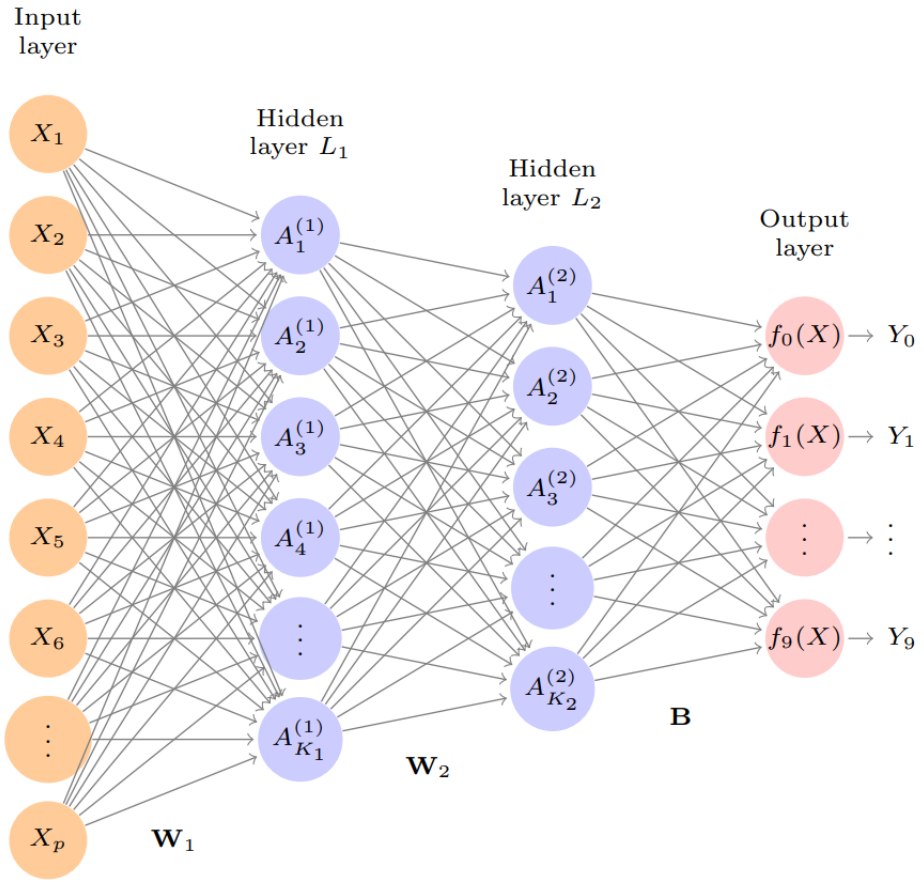


Figure 4.5: A neural network diagram with two hidden layers L_1 and L_2. Input layer consist of p units, L_1 of K_1 neurons, L_2 of K_2 neurons and the output layer of 10 units [47].

To every connection between neurons, a weight is assigned. The initial weights and biases of the network are small random numbers determined by random number generation. The neural network processes all data points of the training data, and the performance is evaluated. In this study, the weights and biases are then adjusted using the optimization algorithm *Adam*, short for Adaptive Moment Estimation. The *Adam* algorithm is a stochastic gradient-based optimizer proposed by Kingma and Ba, and a detailed description of the algorithm can be found in their paper [48]. Gradient-descent is an optimization algorithm used to

minimize an objective function in the direction of steepest descent i.e. the direction in which the objective function is minimized mostly. Compared to regular stochastic gradient-descent, *Adam* converges a lot faster for larger datasets. One important parameter in gradient-descent is the learning rate. The learning rate or step size determines how much weights and biases are adjusted. A high learning rate makes the model converge faster, while a lower learning rate often leads to a more optimal solution. For standard stochastic gradient-descent, the learning rate is constant, while for *Adam* the learning rate adapts over time.

Neural connections that are more important for the final label prediction get assigned a higher weight, and connections that are less important a lower weight. One single cycle of processing all training data and adjusting the weights is called an epoch. This process is repeated multiple times or multiple epochs. Consequently, connections stemming from the most important features will end up having the highest weights. On the other hand, connections stemming from features that hold essentially no information ends up with a very small or even zero weight. It could happen that this neuron never gets activated and will no longer contribute during the training process. This is called a dead neuron. Eventually, once a stopping criterion is met, the model training stops. Examples of stopping criteria for a neural network are a maximum number of epochs or not achieving a certain threshold in a performance increase for a few consecutive epochs. There are also different types of NNs besides ANNs, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs). A CNN is suitable for spatial data such as image processing. To predict pixel values, also neighbouring pixels are considered as features, so patterns in images are incorporated in the prediction process. An RNN is suitable for temporal or sequential data and can use its internal memory to process variable-length sequences of inputs. Essentially, the prediction of a time step also considers the previous time steps. In this study, an ANN was used as it is the most universal and accessible.

4.2.2. Machine Learning Model Evaluation

Optimization should be performed once an ML model is constructed, to achieve the best model performance. This is done by evaluating the performance of a model through a multitude of performance metrics and consequently adjusting the parameters specific to the algorithm aiming to increase the performance. These algorithm-specific parameters are called hyperparameters. Therefore this adjustment process is also referred to as hyperparameter tuning. In this section, the performance metrics used in this study are discussed. Subsequently, the important hyperparameters of each algorithm are explained.

Performance Metrics

A very important concept for performance evaluation of classification problems is the confusion matrix. A confusion matrix visualizes the actual labels y of the dataset versus the predicted labels $\hat{y}$ by the model. It is not a performance metric but forms the basis for other metrics. In the case of this study, we want to predict whether a system is faulty or not. In terms of the confusion matrix of a faulty system, the outcome we want to detect is a positive or 1. On the contrary, a non-faulty system, thus healthy, is a negative or 0. Hence $y = \{0, 1\}$. The machine learning model will predict the labels $\hat{y}$. If $y = \hat{y}$, a correct prediction is made while if $y \neq \hat{y}$ a false prediction is made. Figure 4.6 is an example of a confusion matrix of a decision tree model.

In the bottom-right quadrant the True Positives (TP) ($y = 1$ and $\hat{y} = 1$) are shown. These are actual faulty systems that are also predicted to be faulty. The top-right shows the False Positives (FP) ($y = 0$ and $\hat{y} = 1$). These are the systems that are predicted to be faulty, but in reality are healthy systems. In the top-left are the healthy systems that are predicted to be healthy, the True Negatives (TN) ($y = 0$ and $\hat{y} = 0$). The bottom-left quadrant concerns the False Negatives (FN) ($y = 1$ and $\hat{y} = 0$), faulty systems that are predicted to be healthy. Generally, the higher the number of True Positives and True Negatives compared to the number of False Positives and False Negatives, the better the performance of a model. From this confusion matrix, the following performance metrics can be determined:

- **Accuracy** is defined by the correct predictions versus the total number of predictions:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.7}$$

- **Recall** is the ratio between the correctly detected faulty systems compared to the actual number of faulty systems present:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.8}$$

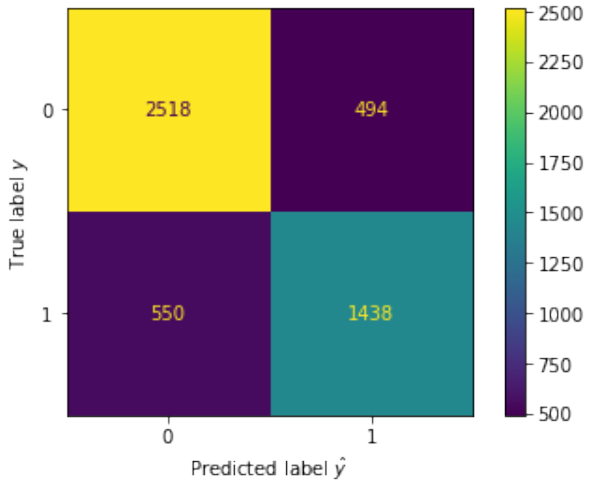


Figure 4.6: Confusion matrix of decision tree classifier for faulty PV systems.

- **Precision** is the ratio between correctly detected faulty systems compared to all the predicted faulty systems. This concerns the healthy systems that are falsely predicted to be faulty.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.9}$$

- **Specificity** is similar to recall, but for the healthy systems. It is the number of correctly predicted healthy systems compared to the actual number of healthy systems:

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{4.10}$$

- **F1-score** combines precision and recall and is the harmonic mean of these two metrics. It is a widely used metric as a high F1-score implies both a good precision and recall and is a good indicator of a good performance.

$$\text{F1-score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} \tag{4.11}$$

The last metric used in this study is the Area Under ROC-Curve (AUC), which is dependent on the Receiver Operatic Characteristics (ROC) curve. The ROC curve plots the False Positive Rate (FPR), which is 1-specificity, against the True Positive Rate (TPR), which corresponds to the recall, at different thresholds. The ROC-curve corresponding to the confusion matrix in figure 4.6 is shown in figure 4.7.

A classifier predicting randomly would have an ROC-curve in a straight line from (0,0) to (1,1). The AUC would be 0.5. Every model with any predictive capability should have an AUC above 0.5. An ideal model with 100% accuracy would show a square curve where a TPR of 1.0 would be reached immediately for an FPR of 0.0. It can be seen that, with increasing the TPR, slowly the FPR starts to increase and the model makes some wrong predictions. Figure 4.7 shows two ROC curves, one for the training data and one for validation data. Validation data acts similar to test data as labels for this set are not fed to the model but have to be predicted by the model. The purpose of a validation set is explained in more detail in section 5.1. In figure 4.7 there is a slight discrepancy between the curve for training data and validation data. This means that the trained model not perfectly generalizes to new data. Here an interesting trade-off is to be made that is referred to as the bias-variance trade-off.

Bias-Variance Trade-off

In terms of machine learning, bias is the error in the learning algorithm that can cause the algorithm to miss the relation between features and the output (labels) when bias is too high. When a model is unable to accurately capture the relationship between features and output, this is called underfitting. An underfit

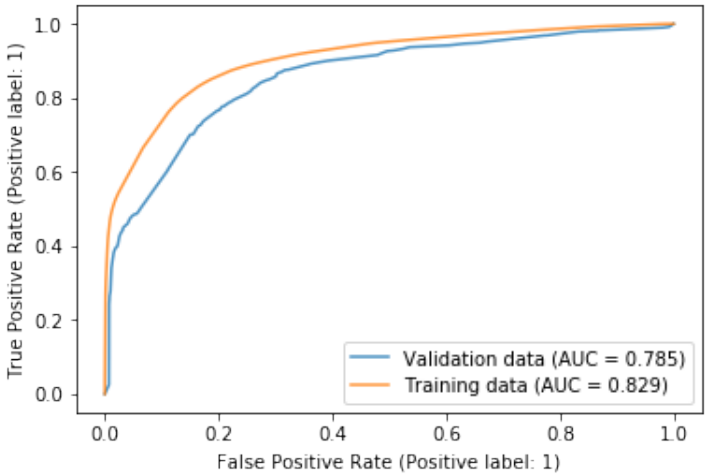


Figure 4.7: ROC-curve of decision tree classifier for faulty PV systems for training set and validation set.

model is not complex enough and therefore does not accurately fit to the training data. Hence, the model will also generalize poorly to the test data. Variance is another error in machine learning due to a model that is overtrained on the training data. Variance is the amount that an estimate of the target function will change if a different training set was used. An overfit model is too complex and fits too much to the training data, capturing relations specific to only this particular training set. When subjecting an overfit model to new data, the model performs poorly as it does not generalize well. Extreme cases of an underfit and overfit model are illustrated by figure 4.8.

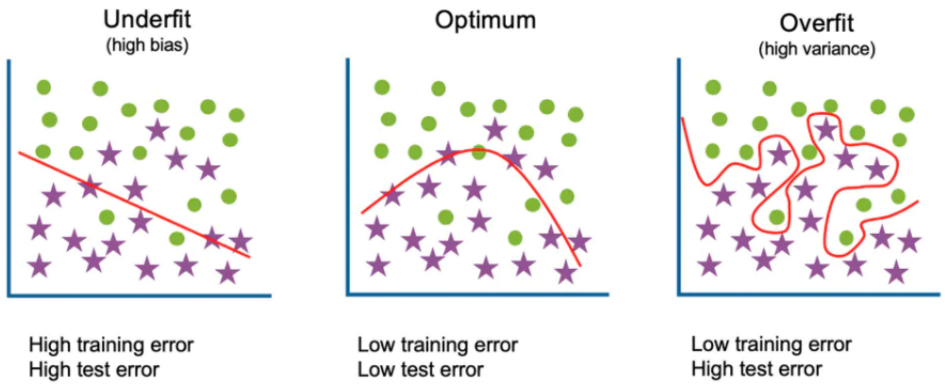


Figure 4.8: Schematic of an underfit model, good fit and overfit model on a training set [49].

Generally, when bias decreases, the variance increases and the other way around. The bias-variance tradeoff is important when a model is desired that performs well on the training data but also generalizes well to new data. In the case of the decision tree classifier used in figure 4.7 it would be possible to get a perfect performance on the training data with a 1.0 score for all metrics by making the decision tree as large and complex as necessary. However, this would significantly decrease the performance on the validation or test data. The model is extremely overfit. The other way around, the decision tree could be made less complex, but the model would not learn the relevant relations in the data, and performance goes down for training data but also for validation or test data. The model would be underfit. Generally, when the ROC-curve of training data and validation or test data are relatively close, it means the model generalizes properly. However, the model will always perform slightly better on training data. By changing the decision tree and performing hyperparameter tuning, the performance can be optimized. and a balance can be found between bias and variance.

Hyperparameter Tuning

Hyperparameter tuning aims to find the combination of hyperparameters yielding a good model performance. There are several hyperparameter tuning methods, such as manual search, random search and grid search. With grid search, a predefined set of values for each hyperparameter is defined, and all combinations are tried. With random search, a similar procedure is followed but with randomly picked values (within a range) rather than a predefined grid of values. In this study, a grid search was performed manually where the initial values were the default values of the scikit-learn library for Python. Subsequently, the following hyperparameters have been tuned for each model, with the parameter names as used in the scikit-learn library:

- Decision Tree
  - max_depth: maximum depth of the decision tree i.e. layers of the tree.
  - min_samples_split: minimum number of samples to split an internal node.
  - min_samples_leaf: minimum number of samples at a leaf node.
  - max_leaf_nodes: maximum number of leaf nodes.
  - ccp_alpha: Complexity parameter for minimal cost-complexity pruning. Pruning is a specific term for algorithms using decision trees aiming to reduce the complexity of the tree(s) to avoid overfitting. The other tuned hyperparameters are also part of the pruning process e.g. reducing the depth of the tree or limiting the number of leaf nodes.
- Random Forest
  - n_estimators: number of decision trees.
  - All parameters listed for decision tree.
- Support Vector Machine
  - C: regularization parameter controlling the error of the model.
  - gamma: a coefficient of the Gaussian RBF kernel controlling the curvature of the hyperplane.
- Neural Network
  - hidden_layer_sizes: the number of neurons in the hidden layer(s).
  - activation: the activation function used.
  - solver: algorithm controlling the weight optimization
  - learning_rate_init: the initial learning rate which controls the size of the weight updates.
  - n_iter_no_change: number of epochs after which no noticeable improvement is made. The training process stops when this number is reached.

For all parameters involved in the tuning process, a range of values was predefined, and combinations of the parameters were tried. One example would be the regularization parameter C of the support vector machine algorithm. A logarithmic range from 10^{-4} to 10^{3} was used. First, the best validation performance was achieved for C in the range of 10^{2} to 10^{3}. Then, a new smaller grid found the best performance was achieved with C = 200. For all other parameters not involved in the tuning process, the value was kept the same as the default value of scikit-learn.

4.3. Conclusion

The third sub-question "How to design a machine learning algorithm for PV fault detection and classification?" can now be answered. First of all, the type of data available should be evaluated, and the correct type of machine learning algorithm should be picked. The data used for PV fault detection and classification in this study is labeled data and the labels are discrete. Therefore a supervised learning algorithm should be used which can perform classification tasks. From literature, the most common machine learning algorithms in the field of PV fault detection and classification were found to be a decision tree, random forest, support vector machine and neural network.
Consequently, each of these four algorithms is trained on the synthetic PV training database constructed in

chapter 3 and the performance can be evaluated using performance metrics. The performance metrics used in this study are accuracy, recall, precision, specificity, F1-score and AUC, alongside confusion matrices for visualization of the results. Finally, the hyperparameters specific to each algorithm can be changed aiming to increase the performance of the models using the performance metrics. This process is called hyperparameter tuning. The machine learning modeling performed in this study was primarily done using the scikit-learn library for Python. The results of the hyperparameter tuning i.e. the final hyperparameters are discussed in chapter 5.

5

Machine Learning Model Performance

This chapter will answer the last and fourth sub-question "What is the performance of different machine learning algorithms trained on a synthetic PV database for fault detection and classification?". Several steps were followed and are discussed in this chapter to test the proposed method of an ML model to detect and classify faults in residential PV systems using a synthetic training database. Preprocessing of the data retrieved from the PVMD Toolbox is discussed in section 5.1. The performance of binary classification is evaluated in section 5.2 and in section 5.3 the multiclass classification is discussed. Finally in section 5.4 the last sub-question is answered.

5.1. Preprocessing

Before the actual testing of the proposed ML algorithms, a preprocessing step is required. This involves feature scaling or data normalization and splitting of the data into a training, validation, and test set.

The first preprocessing step is feature scaling. Most ML algorithms calculate distances between points as part of the objective function, and if different features have a significant variation in range, not all features will contribute equally. Feature scaling will scale all features separately such that they end up having a similar range. Now each feature will contribute proportionately. For neural networks specifically, feature scaling has the additional benefit of significantly faster model training as gradient-descent algorithms converge faster with feature scaling. Also it prevents the algorithm from getting stuck in a local optimum. For support vector machines, it reduces the time of finding the support vectors.

The two common feature scaling methods are normalization and standardization. Normalization scales the features to a [0,1] interval with 0 corresponding to the minimum value of a certain feature and 1 to the maximum value. This is also called min-max scaling. Standardization scales features to a normal distribution with mean $\mu = 0$ and standard deviation $\sigma = 1$. In this study, standardization is performed as most features already follow a normal distribution. This is also the most common scaling method for support vector machines and neural networks. Decision trees and random forests do not necessarily require feature scaling, but doing so will not impact the result.

After the features are standardized, data is split into three sets. The purpose of a training and test set was already discussed in 4.1. When optimizing an ML model, the performance on the test set is tried to be optimized. However, when optimizing the hyperparameters of an ML model with only a training and test set, the hyperparameters will automatically be biased towards this test set. Consequently, the performance will be higher on the test set than on similar different representative datasets. A third set should be added to prevent this biasing issue: a validation set. The validation set is used to optimize the hyperparameters of a model and maximize the model performance. Consequently, the model will be biased towards this validation set just as the model would be to the test set in case there is no validation set. However, this is no longer an issue as the test set can now be used to test the performance of the final model. This way, the test set will be unbiased and give a more accurate representation of the model performance. Commonly, the training set makes up 60% of the data, the validation set 20%, and the test set 20%. The split is created randomly, so the balance between

classes is not exactly the same in the three sets. However, with a large enough dataset, they should not be significantly different.

To check whether the three sets are representative, the mean and standard deviation of every feature of every set was calculated as shown in tables 5.1 and 5.2 for the raw data and standardized data respectively. Comparing these two tables shows the effect of standardization. For example, the range of the feature AC power is completely different from the range of system age prior to feature scaling. Consequently, AC power has much more weight than the other features distorting the results. After standardization, all features can contribute equally to the model training.

Table 5.1: Mean μ and standard deviation σ of the original features of the training-, validation- and test set denoted as $[\mu, \sigma]$.

Dataset	Irradiance (W/m^2)	Temperature (°C)	Azimuth (°)	Elevation (°)	Power (W)	Age (yr)
Training	[238,206]	[12.4,6.8]	[0.1,57.4]	[24.2,15.4]	[1429,1337]	[3.45,4.23]
Validation	[243,209]	[12.5,6.8]	[-2.3,57.4]	[24.5,15.5]	[1431,1327]	[3.53,4.29]
Test	[238,206]	[12.4,6.8]	[2.5,57.3]	[24.3,15.5]	[1430,1337]	[3.43,4.21]

Table 5.2: Mean μ and standard deviation σ of the original features of the standardized training-, validation- and test set denoted as $[\mu, \sigma]$.

Dataset	Irradiance	Temperature	Azimuth	Elevation	Power	Age
Training	[0.00,1.00]	[0.00,1.00]	[0.00,1.00]	[-0.01,0.99]	[0.00,1.00]	[0.00,1.00]
Validation	[0.02,1.02]	[0.01,0.99]	[-0.04,1.00]	[0.02,1.01]	[0.00,0.99]	[0.02,1.03]
Test	[-0.01,0.99]	[0.00,1.00]	[0.04,0.99]	[0.00,1.01]	[0.00,1.00]	[0.00,0.99]

Univariate unpaired t-tests were performed between all set combinations for each feature. It was found that for all features except azimuth, there was no significant difference between the sets. This implies the three sets are similar and therefore representative, except for the azimuth in this case. After another random split was created, the azimuth between the three sets was also not significantly different, implying there is no inherent issue with the process of feature scaling and data splitting.

In the early stages of the model, wind speed was also considered a feature, being part of the meteorological conditions. A chi-square test was performed to evaluate the relevance of each feature. The chi-square test is used in statistics to evaluate the independence of two variables. A low chi-square value corresponds to two variables being independent. However, in this case, the interest lies in a high chi-square value between a feature and the labels. The high chi-square value implies the feature is more dependent on the labels and vice versa, thus being a good feature to select for model training.

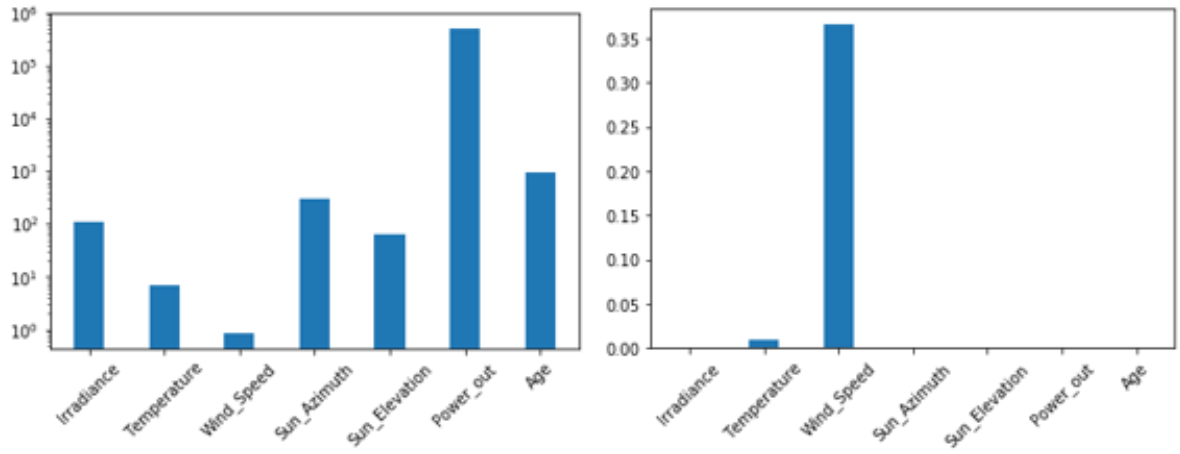
The chi-square test results in 5.1 show that wind speed has a very low chi-square value and p-value far above 0.05. Thus, it is concluded wind speed and the model outcome are independent, and wind speed was removed as a feature. System power output is shown to be the most crucial feature in predicting the status of a system, which is to be expected.

5.2. Binary Classification

After the preprocessing process, binary classification was performed to see whether a model is capable of detecting faulty operating systems and regular healthy operating systems or not. The binary classification is divided into two cases. The first case, case 1, considers the addition of extra features to enhance model performance. Here, hyperparameter tuning was also performed alongside the addition of the new features. The second case, case 2, concerns the balance in the database between healthy and faulty systems and evaluates the impact of this balance in the database. In case 2, the trained models of case 1 were used, and no new hyperparameter tuning was performed.

5.2.1. Case Study Considerations and Settings

The initial six features discussed in section 5.1 make up the initial database. Every data point i corresponds to a single point in time with an average hourly value of each feature j, where the time is a randomly picked



(a) Chi-square value.

(b) P-value.

Figure 5.1: Chi-square value and p-value of chi-square test performed on initial 7 features.

hour between sunrise and sunset. Initially, the idea was to consider the time element in the model. Faults would be introduced at a random hour during the day, and the change in power output should be noticed by the model and aid in the model's predictive capability. However, including a complete time sequence as feature for the model proved to be too complex for the scope of this study, but only on a single hourly data point. Therefore the first models were trained on the initial database without any time element. Since the point in time is picked randomly, a random hour might be picked with very low irradiance. Consequently this leads to a relatively low power output, for example early in the morning or close to sunset. From literature, it is clear that the detection of faults is more complex when irradiance is low [32]. Therefore, besides the original 6 features, it was decided to introduce new features: irradiance and power output values from the day before at the same hour and at 10:00, 13:00 and 16:00 the day before. This would partially solve the issue of accidentally sampling an hour with very low irradiance, for example by sampling the irradiance and power at 13:00, which is generally the hour with the highest irradiance. Also, adding these features includes some form of time element, which is not as complex as working with complete time sequences.

Besides these 8 additional features, a new feature was added in the form of the ratio of system output power to irradiance (power ratio). This ratio should be lower for faulty systems independent of the power and irradiance itself and should add extra value. Furthermore, this power/irradiance ratio was also determined for the aforementioned points in time: 10:00, 13:00 and 16:00 of the day before, as well as the same hour as the randomly sampled hour the day before. This adds 5 new features, meaning there are 13 additional features and hence 19 features in total.

5.2.2. Case 1: Feature Selection

Case 1 compares the model performances for the 6 original features against using 13 features and 19 features. Here, the 19-feature case includes all the additional features, whereas the 13-feature case leaves out the irradiance, power output and power/irradiance ratio at both 10:00 and 16:00 for each variable. An overview of the features used for the three different cases is shown in table 5.3

In table 5.4 the performance of the random forest model is shown for the three cases. Training and testing of the models are repeated ten times, and the mean and standard deviation are both shown in the results notated as μ (σ). The process of repetition decreases inaccuracy in results due to random initialization of hyperparameters. All results shown in this chapter have a similar notation to table 5.4 and are all the result of ten repetitions in training and testing.

It is clear that model performance increases significantly for all metrics by adding the extra features. Especially the inclusion of the first additional features proves to add significant value to the database as the increase in performance increases by over 5% for all metrics. Also, the last 6 extra features still have a positive, yet very small, impact on the performance, resulting in an extra performance increase of less than 1%.

Table 5.3: The three sub-cases and their respective features used for model training.

Case (no. of features)	Features
1a (6)	Irradiance, Sun azimuth, Sun elevation, Ambient Temperature, System Power, System Age
1b (13)	All features of case 1a, Irradiance day before, Irradiance at 1pm, System power day before, System power at 1pm, Power ratio, Power ratio day before, Power ratio 1pm
1c (19)	All features of case 1a & 1b, Irradiance at 10am, Irradiance at 4pm, System power at 10am, System power at 4pm, Power ratio at 10am, Power ratio at 4pm

Table 5.4: Performance of random forest model on test set for different number of features.

Metric	1a (6)	1b (13)	1c (19)
Accuracy	0.743 (0.003)	0.803 (0.003)	0.805 (0.002)
Recall	0.648 (0.006)	0.713 (0.008)	0.710 (0.006)
Precision	0.687 (0.004)	0.774 (0.005)	0.780 (0.004)
Specificity	0.806 (0.002)	0.862 (0.004)	0.868 (0.004)
F1	0.667 (0.005)	0.742 (0.005)	0.744 (0.003)
AUC	0.727 (0.004)	0.788 (0.004)	0.789 (0.002)

Performance of the decision tree model, SVM model and NN model can be found in appendix C.1 shown in tables C.1-C.4 and the corresponding confusion matrices in figures C.1-C.12. For all models, the 13 feature case significantly outperforms the 6 feature case.

Adding the power ratio in combination with irradiance, power and power ratio of the previous day, as well as irradiance, power and power ratio at 1 PM of the previous day, appears to be advantageous for fault detection. However, for the decision tree model and neural network model, the performance is slightly worse for the 19 feature case. In contrast, for the random forest model and support vector machine model, the performance slightly improves. Adding even more values of the previous day (10 AM and 4 PM) for irradiance, power and power ratio at 10 AM appears not to significantly impact the performance.

As mentioned, in case 1 the model's hyperparameters have been tuned in order to enhance model performance. An overview of all hyperparameter values for the final binary classification models can be found in appendix B.1. The random forest model outperformed the other models, when hyperparameter tuning was performed using the validation set. As the random forest model appears to benefit slightly from the additional features, it was decided to keep all 19 features. Therefore, for all subsequent results in this study, 19 features were included in model training and testing. Since the difference in performance between case 1b and case 1c is very small and for other models even the other way around, picking the better option is quite arbitrary. However, the training time is shorter for case 1b due to fewer features, which is most relevant for the support vector machine model and neural network model as the decision tree model and random forest model are already very fast. This could be considered in future research. Also, different feature combinations of the 19 features could be tried, possibly resulting in better performance.

5.2.3. Case 2: Database balance

In the introduction, it was mentioned that one advantage of a synthetic database is the ability to control the balance of classes. Historic databases, under the condition that all data is complete and correct, typically are very skewed towards healthy operating systems. Therefore, in case 2, the balance of the database is varied, and theperformance of the models is evaluated. Case 2 is further divided into four subcases. In case 2a, the completed dataset balance is varied, including training, validation, and test set. Case 2b considers the balance from the original set for the training set while keeping the number of faulty systems compared to the number of healthy systems low in the validation and test, such that the test set mimics the skewed balance of a historical database. Case 2c compares the model performance of the four models found in case 2a at different database balances to find the best model for a certain balance. Finally, case 2d looks into the effect of dataset size.

Case 2a: Varying Database Balance

In the first subcase, the ratio of faulty systems to the total number of systems has been varied from 1% to 70%. To ensure an equal comparison, the size of the dataset was kept equal for all ratios. From the original dataset of 25000 data points, faulty and healthy systems were individually sampled without replacement such that a new subset of the data was created with 15000 data points with the desired balance. Only for the 70% case, the dataset size is slightly smaller with 14286 data points as there were not enough faulty systems in the original data set to create a dataset with 15000 data points and 70% faulty systems. Table 5.5 shows the performance of the neural network model for varying faulty system ratios.

Table 5.5: Performance of neural network model on test set for varying ratio of faulty systems in dataset.

Metric	1%	5%	10%	20%	30%	40%	50%	60%	70%
Accuracy	0.993 (0.001)	0.956 (0.007)	0.918 (0.005)	0.853 (0.006)	0.808 (0.005)	0.790 (0.005)	0.793 (0.009)	0.798 (0.005)	0.812 (0.005)
Recall	0.509 (0.061)	0.365 (0.027)	0.466 (0.015)	0.529 (0.041)	0.631 (0.030)	0.723 (0.021)	0.804 (0.026)	0.854 (0.016)	0.889 (0.008)
Precision	0.602 (0.117)	0.562 (0.108)	0.617 (0.044)	0.678 (0.038)	0.713 (0.017)	0.759 (0.014)	0.795 (0.008)	0.822 (0.010)	0.847 (0.005)
Specificity	0.997 (0.002)	0.984 (0.008)	0.968 (0.007)	0.935 (0.015)	0.887 (0.014)	0.837 (0.017)	0.780 (0.016)	0.712 (0.024)	0.637 (0.017)
F1	0.540 (0.039)	0.436 (0.029)	0.530 (0.012)	0.592 (0.018)	0.669 (0.013)	0.740 (0.007)	0.799 (0.012)	0.837 (0.005)	0.868 (0.004)
AUC	0.753 (0.030)	0.675 (0.011)	0.717 (0.005)	0.732 (0.014)	0.759 (0.010)	0.780 (0.005)	0.792 (0.008)	0.783 (0.006)	0.763 (0.007)

It becomes clear that increasing the number of data points representing faulty systems greatly increases model performance, largely due to the increased recall. Recall is very low for a highly imbalanced dataset. Similarly, precision increases for the neural network model when increasing the number of faulty systems which is not necessarily the case for all models: the random forest model shows highest precision for lower ratio of faulty systems, as can be seen in appendix C.2 in table C.6. When recall increases, consequently F1-score is greatly increased. On the contrary, when the dataset is skewed to healthy systems, almost all of the test set is correctly labeled as healthy systems. Model accuracy decreases when increasing the balance in the dataset and increases again once the dataset gets more skewed towards faulty systems. This is due to the decreasing specificity with an increasing number of faulty systems. However, when the faulty system ratio crosses the 50%-point, the number of correctly predicted faulty systems starts to outweigh the lower number of correctly predicted healthy systems, and accuracy goes up again slightly. Overall, a good balance across all metrics is found exactly for a perfectly balanced dataset. At this point, the AUC is at its highest. Results for the other models can be found in appendix C.2 in tables C.5-C.8 and the corresponding confusion matrices for the varying dataset balances for all models in figures C.13-C.48.

Case 2b: Optimal Training Set Balance with Low Fault Ratio in Test Set

In this subcase, an approach is tried where the balance in training data is manipulated, but the balance in test data is kept low. This way, the test data is very skewed, similarly to historical data, while having the advantage of a synthetic training dataset in manually balancing the training data. Training was performed on the dataset from case 1, having a balance of 40% faulty systems. Performance on test data with low ratios of 1%, 2%, 5% and 10% was tested. Results of the decision tree model are shown in table 5.6.

Table 5.6: Performance of decision tree model on test set for varying ratio of faulty systems in test set while keeping 40% ratio in training set.

Metric	1%	2%	5%	10%
Accuracy	0.830 (0.001)	0.831 (0.001)	0.829 (0.001)	0.826 (0.001)
Recall	0.567 (0.000)	0.683 (0.000)	0.724 (0.003)	0.723 (0.003)
Precision	0.033 (0.000)	0.077 (0.001)	0.187 (0.002)	0.331 (0.003)
Specificity	0.833 (0.001)	0.834 (0.001)	0.835 (0.001)	0.838 (0.001)
F1	0.062 (0.000)	0.139 (0.001)	0.297 (0.002)	0.454 (0.003)
AUC	0.700 (0.001)	0.759 (0.001)	0.780 (0.002)	0.780 (0.002)

When the results in table 5.6 are compared with the results from case 1, it is clear that precision is very low, and this also heavily impacts the F1-score, it being the harmonic mean of recall and precision. The performance on the other metrics is similar to case 1 or even slightly better. Looking at both recall and specificity, it is apparent that the model is still capable of classifying most of the healthy systems correctly as healthy systems and faulty systems as faulty systems. However, due to the extreme class imbalance in the test set, the number of false negatives is a lot higher than the number of true negatives, leading to a very poor precision score. The confusion matrix illustrates this in figure 5.2. For example, a very low precision of only 0.187 for the 5% balance case is achieved. This is due to the very skewed test set causing the number of healthy systems that are misclassified as faulty to outweigh the number of correctly classified faulty systems.

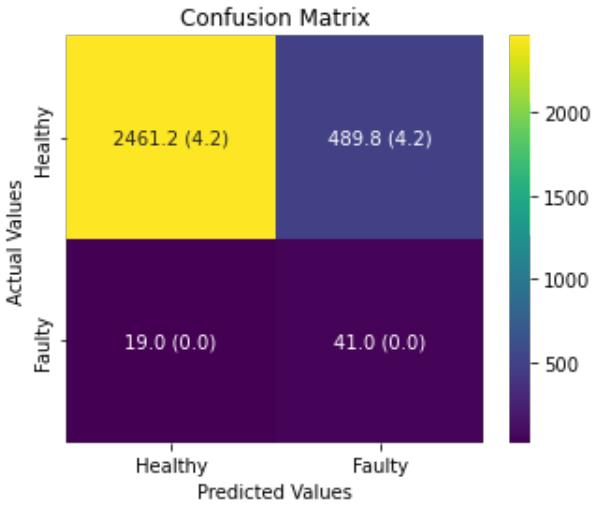


Figure 5.2: Confusion matrix of decision tree model for 40% faulty systems in training set and 2% faulty systems in test set.

However, this doesn't mean the model's performance is bad. In the case of 5% faulty systems in the test set, the decision tree model can predict with an accuracy of 82.9%, and 72.4% of the faulty systems are correctly classified. Comparing this to the results of the decision tree model of case 2a in table 5.7, for similar faulty systems ratios, it shows that recall is greatly improved. In this case study, recall is the most interesting metric to look at, and it shows that with a very skewed dataset, only balancing the training data already greatly improves the performance in detecting faulty systems correctly. For a 5% balance, the recall is more than doubled when balancing the training set compared to not balancing the training set.

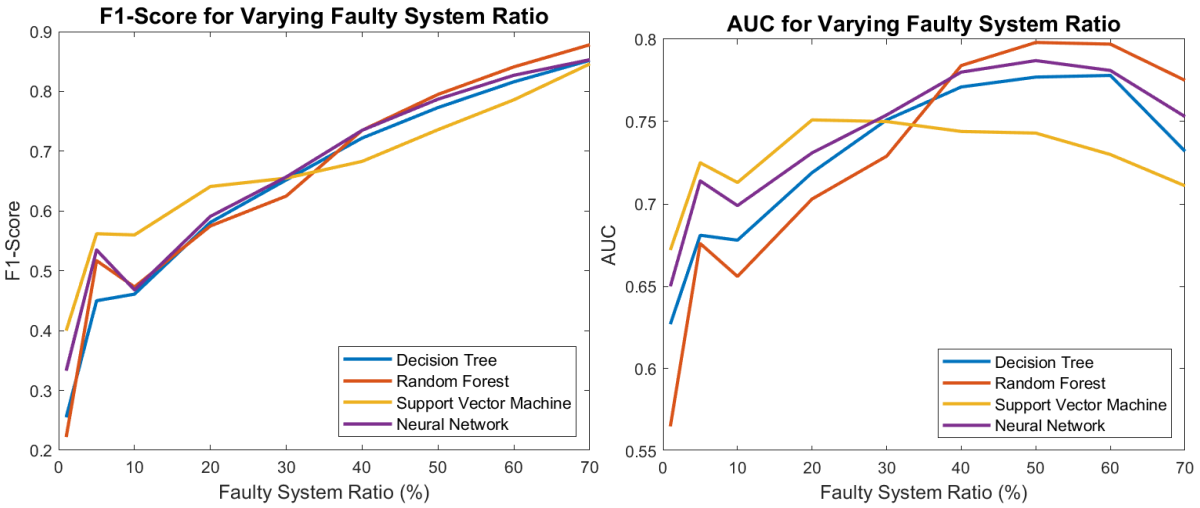
Table 5.7: Performance of decision tree model from case 2a, for varying ratio of faulty systems in both test set and in training set.

Metric	1%	5%	10%
Accuracy	0.988 (0.001)	0.956 (0.001)	0.917 (0.001)
Recall	0.296 (0.013)	0.319 (0.007)	0.421 (0.005)
Precision	0.289 (0.034)	0.549 (0.025)	0.619 (0.008)
Specificity	0.994 (0.001)	0.987 (0.001)	0.972 (0.001)
F1	0.292 (0.023)	0.403 (0.010)	0.501 (0.005)
AUC	0.645 (0.007)	0.653 (0.003)	0.696 (0.002)

Results for the other models can be found in appendix C.3 in tables C.9-C.12 and the corresponding confusion matrices for the varying dataset balances for all models in figures C.49-C.64.

Case 2c: Best Model for Different Database Balances

In this subcase of binary classification, the four models are compared for varying balances in the dataset (both training and test set having the same balance) to see which model performs best at certain faulty system ratios in the dataset. Depending on the metric, the best model slightly differs, but overall the trend is clearly illustrated when comparing the models with their respective F1-score and AUC as shown in figure 5.3a and 5.3b.



(a) F1-score.

(b) AUC.

Figure 5.3: F1-score and AUC for all models with varying faulty system ratio in the dataset.

The SVM model outperforms the other models when the dataset is poorly balanced (0% - 20% fault ratio). However, the performance of the SVM model starts to decrease rapidly relative to the other models when the dataset is more well balanced and is clearly the worst model of the 4 in these cases. With a balance of 30% the neural network is the best model, whereas the random forest model starts to outperform for balanced datasets or when the fault ratio is even higher (50% - 70%).

Case 2d: Dataset Size

In this last case of binary classification, the impact of the size of the dataset is evaluated. The initial dataset of case 1 had a faulty system ratio of 40% and a size of 25000. However, in case 2a, the dataset size was decreased to 15000 as explained in section 5.2.3. Table 5.8 compares the results of case 1 with the results for 40% faulty system ratio of case 2a.

Metric	DT	RF	SVM	NN
Accuracy	0.798 (0.001)	0.794 (0.002)	0.785 (0.000)	0.790 (0.005)
Recall	0.711 (0.001)	0.678 (0.003)	0.652 (0.000)	0.723 (0.021)
Precision	0.738 (0.001)	0.795 (0.005)	0.792 (0.000)	0.759 (0.014)
Specificity	0.821 (0.001)	0.876 (0.004)	0.879 (0.000)	0.837 (0.017)
F1	0.724 (0.001)	0.732 (0.003)	0.715 (0.000)	0.740 (0.007)
AUC	0.766 (0.001)	0.777 (0.002)	0.765 (0.000)	0.780 (0.005)

(a) Dataset size of 15000.

Metric	DT	RF	SVM	NN
Accuracy	0.791 (0.000)	0.805 (0.002)	0.797 (0.000)	0.810 (0.004)
Recall	0.724 (0.002)	0.710 (0.006)	0.637 (0.000)	0.771 (0.022)
Precision	0.743 (0.001)	0.780 (0.004)	0.811 (0.000)	0.756 (0.016)
Specificity	0.835 (0.001)	0.868 (0.004)	0.902 (0.000)	0.835 (0.018)
F1	0.733 (0.001)	0.744 (0.003)	0.714 (0.000)	0.763 (0.006)
AUC	0.779 (0.000)	0.789 (0.002)	0.770 (0.000)	0.803 (0.005)

(b) Dataset size of 25000.

Table 5.8: Model performance on test set with dataset size of 15000 and 25000, both with faulty system ratio of 40%.

With increasing dataset size, all models increase in performance. However, when comparing the models, some appear to respond better to increasing dataset size than others. While the DT, RF and SVM models only slightly increase, the NN model increases by around 2% for almost every metric. During the hyperparameter tuning process using the validation set, this difference between models with respect to dataset size was

enough for the NN model to even overtake the RF model as the best performing model. Although a dataset of size 15000 seems sufficiently large, it is clear that there is still a lot of room for an increase in performance by simply increasing the dataset size.

5.3. Multiclass Classification

Where binary classification aims to identify faulty operating residential PV systems, multiclass classification in this study aims also to detect what type of fault is causing the system to malfunction. The multiclass classification is divided into 4 subcases, aiming to improve the performance of the classification model. In the first case, 3a, both the one-versus-one and one-vs-rest approaches are tested and compared. Also, the hyperparameter tuning for the multiclass classification models is performed in case 3a. An overview of all hyperparameter values for the final multiclass classification models can be found in appendix B.2 In case 3b, a different approach to balance the dataset is evaluated. In case 3c the fault classes of ground faults and line-line faults are combined into one class. In the last case, 3d, the fault resistance range for the data generation of ground fault scenarios and line-line fault scenarios is adjusted, and a partially new dataset is tested.

Case 3a: One-Versus-One classifier vs. One-Versus-Rest Classifier

Contrary to binary classification, not all machine learning algorithms are inherently capable of dealing with multiple classes. Therefore there are two approaches to overcome this issue, which essentially transform the multiclass classification problem into a binary classification problem: one-versus-rest (OVR) classification and one-vs-one (OVO) classification. A one-versus-rest classification of K classes transforms the problem into K binary classification problems where a single class is compared against all the other classes combined as the other class. Thus, there is one classifier for each class. When making predictions, all classifiers are applied to the test data and the classifier predicts the label with the highest confidence score. For one-versus-one classification, a multiclass classification problem of K classes is split into $K(K-1)/2$ binary classification problems, where each binary classifier is trained on a single pair of classes. Again, all classifiers are applied to the test data, and the class with the highest number of predictions is picked as the final prediction.

For the multiclass classification, a subset of the original dataset has been taken such that every class has a similar size. Since the original dataset consisted of 2500 samples of each fault type, the subset for multiclass classification has a size of 12500. The five classes (healthy, ground fault, line-line fault, broken string fault, broken cells fault) all make up 20% of the dataset. All four models have been trained using this new dataset, both for OVO-classification and OVR-classification. For the random forest model and support vector machine model, the OVO-classifier outperformed the OVR-classifier, wheareas for the neural network model, the OVR-classifier performed better. The random forest model is the best performing model for both multiclass classification strategies. In table 5.9, the performance of both classifiers using the random forest algorithm is shown.

The performance for the multiclass classification shows very poor results. For some classes, the prediction capability is even below 0.5. This is due to two problems, more clearly illustrated by the confusion matrix. The confusion matrix for the OVO-classifier using a random forest model is shown in figure 5.4. The first problem is mostly caused because the ground and line-line faults are hard to distinguish from each other. Many ground faults are classified as line-line faults and vice versa, as is clear from the confusion matrix. This is due to the similar behavior in power decrease as an effect of these types of faults. Additionally, the same failure distribution has been used for the two types of faults due to a lack of research on this topic. The second problem that is illustrated by these results is the limited performance in classifying healthy operating systems. As described in sections 3.3.1 and 3.3.2, only ground and line-line faults that do not trip traditional protection devices are considered in this study. Therefore, the reduced power output of the considered fault cases is limited. This also affects the distinction between these short-circuit fault cases and healthy systems. All results of case 3a are shown in appendix D.1 in tables D.1-D.8 and confusion matrices in figures D.1-D.4. The next three subcases aim to tackle these two problems.

Case 3b: Varying Balance of Healthy and Faulty Systems for Multiclass Classification

In the previous case, the dataset was balanced such that every class was equally represented. In order to increase the performance of correct classification of healthy operating systems, for case 3b, the balance is changed to a similar balance as in the binary classification: half of the dataset is made up of healthy systems

Class	Precision	Recall	F1
Healthy	0.710 (0.008)	0.550 (0.008)	0.620 (0.005)
Line-Line	0.516 (0.007)	0.508 (0.013)	0.512 (0.009)
Ground	0.443 (0.008)	0.633 (0.009)	0.521 (0.007)
Broken Cell	0.728 (0.009)	0.605 (0.009)	0.661 (0.008)
Broken String	0.929 (0.004)	0.952 (0.003)	0.940 (0.003)
Average	0.665 (0.004)	0.650 (0.003)	0.651 (0.004)

(a) One-versus-one.

Class	Precision	Recall	F1
Healthy	0.712 (0.008)	0.526 (0.011)	0.605 (0.006)
Line-Line	0.512 (0.011)	0.496 (0.015)	0.504 (0.011)
Ground	0.440 (0.008)	0.654 (0.016)	0.526 (0.010)
Broken Cell	0.716 (0.009)	0.586 (0.008)	0.645 (0.005)
Broken String	0.920 (0.003)	0.952 (0.003)	0.936 (0.002)
Average	0.660 (0.005)	0.643 (0.005)	0.643 (0.005)

(b) One-versus-rest.

Table 5.9: Multiclass classification performance on test set using random forest model.

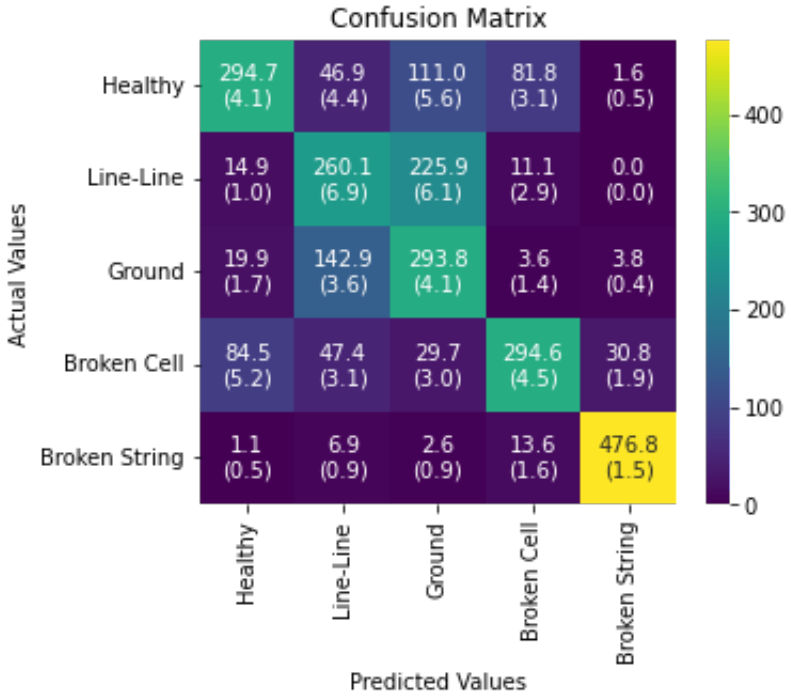


Figure 5.4: Confusion matrix of OVO-classifier using random forest model on test set.

and the remaining half of faulty systems, and thus each fault type contributes to one-eighth of the dataset. The faulty system ratio in the training and test set is assumed to be the same. For this case, the OVO-classifier using an NN-model gives the best results. A comparison with the OVO-classifier with NN-model where all classes are balanced equally (from case 3a) is shown in table 5.10. The classification of healthy systems is certainly improved, with recall going up by more than 30 %. However, this is at the expense of the classification capability of all other classes. Especially the performance on ground faults and line-line faults is decreased, with recall dropping 12% and 26% respectively for OVO-classification using a neural network model. All results of case 3b are shown in appendix D.4 in tables D.9-D.16 and the confusion matrices in figures D.9-D.12.

Class	Precision	Recall	F1
Healthy	0.787 (0.011)	0.833 (0.014)	0.809 (0.005)
Line-Line	0.441 (0.023)	0.306 (0.029)	0.360 (0.018)
Ground	0.299 (0.014)	0.380 (0.053)	0.334 (0.030)
Broken Cell	0.679 (0.022)	0.561 (0.018)	0.614 (0.011)
Broken String	0.963 (0.003)	0.962 (0.008)	0.963 (0.004)
Average	0.634 (0.009)	0.608 (0.009)	0.616 (0.007)

(a) Healthy and faulty balanced equally.

Class	Precision	Recall	F1
Healthy	0.688 (0.017)	0.529 (0.023)	0.597 (0.013)
Line-Line	0.474 (0.015)	0.419 (0.029)	0.445 (0.020)
Ground	0.400 (0.009)	0.577 (0.024)	0.472 (0.010)
Broken Cell	0.681 (0.016)	0.642 (0.013)	0.661 (0.007)
Broken String	0.952 (0.004)	0.963 (0.006)	0.958 (0.005)
Average	0.639 (0.005)	0.626 (0.005)	0.627 (0.005)

(b) All classes balanced equally.

Table 5.10: Multiclass classification performance on test set of OVO-classifier using NN model for two different dataset balances.

Case 3c: Combined Short-Circuit Fault Class for Ground and Line-Line Faults

In this case, the first problem described in section 5.3 is tackled, where ground faults and line-line faults are often misclassified as the other type of short-circuit fault and thus hard to distinguish. Therefore the ground faults and line-line faults are combined in a single class: short-circuit faults. As is clear from the confusion matrix in figure 5.4 of case 3a, these two fault types or hard to distinguish from each other. To maintain class balance, only half of the data points from this combined class are used. Thus the size of the dataset is 10000. By combining the two, the overall model performance should hypothetically be increased, whereas in practice, this would not make much of a difference: as the proposed method in this study does not identify the location of a fault in the PV system, once a system is classified as malfunctioning due to a ground or line-line fault, on-site investigation has to be performed either way, to determine the location of the short-circuit fault. The OVO-classifier and OVR-classifier, both using a random forest algorithm or neural network algorithm, are all close in performance for the best-performing model. Table 5.11 compares the OVO-classifier using a random forest for the combined-short circuit class with the results from case 3a.

Class	Precision	Recall	F1
Healthy	0.723 (0.007)	0.565 (0.007)	0.635 (0.007)
Short-Circuit	0.653 (0.002)	0.928 (0.003)	0.767 (0.002)
Broken Cell	0.774 (0.011)	0.572 (0.005)	0.658 (0.006)
Broken String	0.925 (0.003)	0.978 (0.001)	0.950 (0.002)
Average	0.769 (0.004)	0.761 (0.003)	0.752 (0.003)

(a) Combined short-circuit fault class.

Class	Precision	Recall	F1
Healthy	0.710 (0.008)	0.550 (0.008)	0.620 (0.005)
Line-Line	0.516 (0.007)	0.508 (0.013)	0.512 (0.009)
Ground	0.443 (0.008)	0.633 (0.009)	0.521 (0.007)
Broken Cell	0.728 (0.009)	0.605 (0.009)	0.661 (0.008)
Broken String	0.929 (0.004)	0.952 (0.003)	0.940 (0.003)
Average	0.665 (0.004)	0.650 (0.003)	0.651 (0.004)

(b) Ground and line-line fault as separate classes.

Table 5.11: Multiclass classification performance on test set of OVO-classifier using RF model for a combined short-circuit fault class and for ground fault and line-line fault as separate classes.

The results confirm the hypothesis of increased model performance when combining the two types of

short-circuit faults into a single class. The overall performance expressed in precision, recall and F1-score is increased by 10%. The performance of the unchanged classes is mostly unaffected, with even a small increase for the healthy and broken cell class. Unfortunately, the performance of classifying healthy operating systems is still low compared to other classes. As can be seen in the confusion matrix in figure 5.5, a high number of healthy systems are still misclassified as belonging to the short-circuit fault class. All results of case 3d are shown in appendix D.3 in tables D.17-D.24 and confusion matrices in figures D.17-D.20. The next and final case aims to decrease the number of misclassified healthy systems as systems with a short-circuit fault to improve overall classification performance.

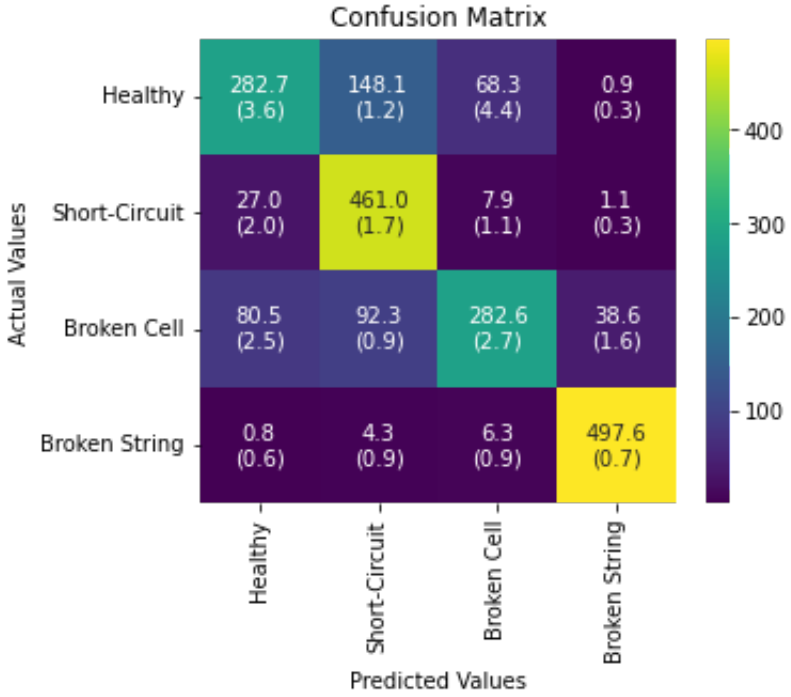


Figure 5.5: Confusion matrix of OVO-classifier using random forest model on test set with short-circuit fault class.

Case 3d: Lower Fault Resistance for Short-Circuit Faults

As mentioned in section 5.3, the problem of misclassification of healthy operating systems as short-circuit faults is caused by the limited effect on the power output of these faults. By only investigating short-circuit faults that are not detected by traditional protection devices, the fault currents are small, and hence the decrease in system power output is limited. In the original database simulation (used for all previous cases), fault resistances for short-circuit faults of up to 1000 Ω were included. The fault resistance sampling was done such that more cases with relatively low fault resistances were sampled. Histograms of the fault resistances for ground faults and line-line faults of the original dataset are shown in figure 5.6. Since high fault resistance results in a lower decrease in power, the cases with high fault resistance are hypothetically the hardest to detect by the model. Therefore, in case 3d, the maximum fault resistance in the database generation is lowered to 100 Ω. Removing the high resistance cases above 100 Ω, which result in the smallest power decrease, could therefore have a positive effect on the model performance. Histograms of the fault resistances used for case 3d are shown in figure 5.7. In both figure 5.6 and figure 5.7, the fault resistance for line-line faults is less skewed towards low resistances than for ground faults and is more spread out over the resistance range. This is due to the data generation model for line-line fault cases often not being able to find a solution when a low resistance was sampled. Consequently, a new fault resistance is sampled from the given resistance range leading to more cases with a fault resistance on the higher end of the range.

The best performing model of case 3d, OVR-classifier using RF-algorithm, is slightly better compared to case 3c as shown in table 5.12. The healthy system class contributes most to the performance increase, while also an increase in the short-circuit class is noticeable. This is more clear when comparing the confusion matrices of both cases shown in figures 5.8 and 5.9 where the difference, although very limited, is almost

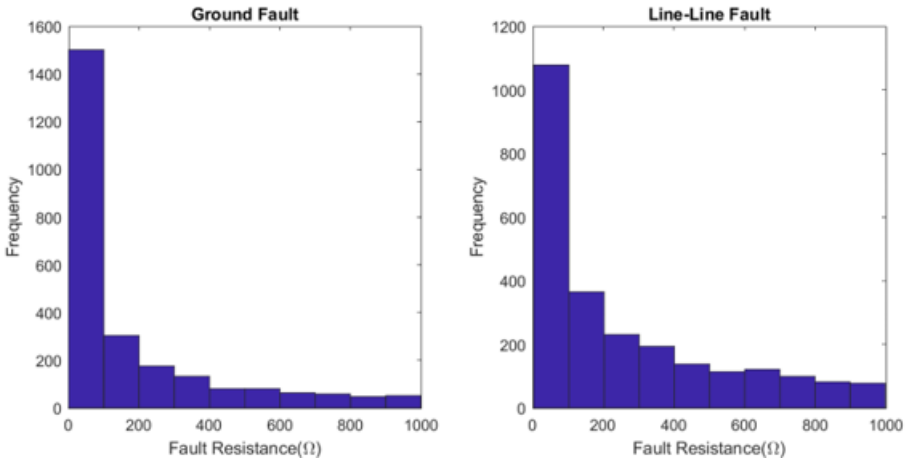


Figure 5.6: Fault resistance histograms of ground faults and line-line faults of original dataset with $R_{fault,max} = 1000\,\Omega$.

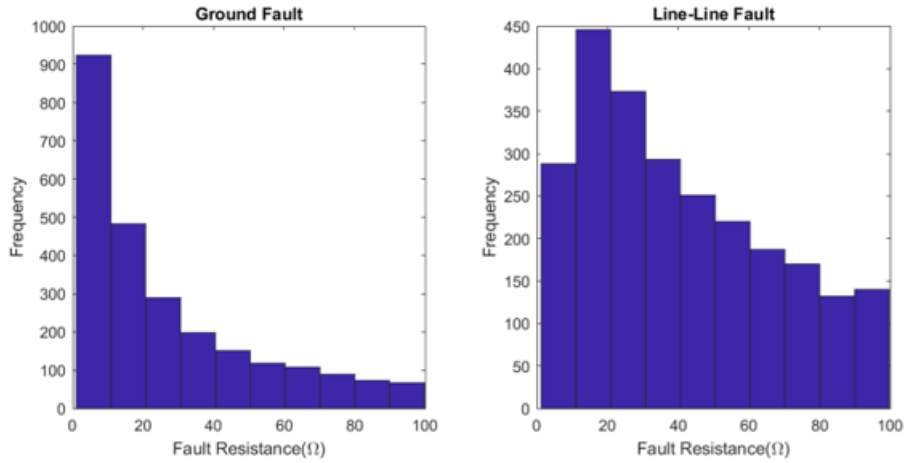


Figure 5.7: Fault resistance histograms of ground faults and line-line faults of new dataset with $R_{fault,max} = 100\,\Omega$.

solely due to the decreasing number of misclassified healthy systems as short-circuit faults. Ultimately, the OVR-classifier using an RF-algorithm of case 3d is the best overall performing multiclass model achieved in this study. Besides the performance increase for both OVO and OVR-classifier combined with the random forest algorithm, only the OVR-classifier with support vector machine algorithm performs better in case 3d than in case 3c. The models perform slightly worse for all other options when lowering the maximum fault resistance in the synthetic data generation process. All results of case 3d are shown in appendix D.4 in tables D.25-D.32 and confusion matrices in figures D.25-D.28. It can not be concluded that lowering the maximum fault resistance for short-circuit faulting in the database generation process has a positive effect on the ML model performance in general. As was mentioned in section 2.2.1, little information on fault resistances and their progression over time is known. When more information is available on this topic, a more accurate analysis could be done on this parameter in the synthetic fault data generation process and its impact on the model performance.

5.4. Conclusion

The final sub-question "What is the performance of different machine learning algorithms trained on a synthetic PV database for fault detection and classification?" can now be answered. For binary classification, the best performance was obtained by the random forest model with a dataset balance of 50% faulty system and 19 features: an accuracy of 0.804, recall of 0.827, precision of 0.797, specificity of 0.778, F1-score of 0.812 and AUC of 0.803. By increasing the ratio of faulty systems to 70%, the results could be considered even better if detecting faulty systems is considered more important: accuracy of 0.828, recall of 0.922, precision of 0.844,

Class	Precision	Recall	F1
Healthy	0.720 (0.006)	0.584 (0.009)	0.645 (0.006)
Short-Circuit	0.672 (0.003)	0.910 (0.003)	0.773 (0.003)
Broken Cell	0.760 (0.008)	0.592 (0.009)	0.666 (0.006)
Broken String	0.928 (0.002)	0.976 (0.003)	0.952 (0.002)
Average	0.770 (0.003)	0.766 (0.003)	0.759 (0.003)

(a) $R_{fault,max} = 100\,\Omega$.

Class	Precision	Recall	F1
Healthy	0.725 (0.007)	0.563 (0.008)	0.634 (0.007)
Short-Circuit	0.655 (0.002)	0.932 (0.005)	0.769 (0.002)
Broken Cell	0.773 (0.007)	0.574 (0.007)	0.659 (0.007)
Broken String	0.926 (0.003)	0.978 (0.003)	0.951 (0.001)
Average	0.770 (0.003)	0.762 (0.003)	0.753 (0.003)

(b) $R_{fault,max} = 1000\,\Omega$.

Table 5.12: Multiclass classification performance on test set of OVR-classifier using RF model for a combined short-circuit fault class with $R_{fault,max} = 100\,\Omega$ and $R_{fault,max} = 1000\,\Omega$.

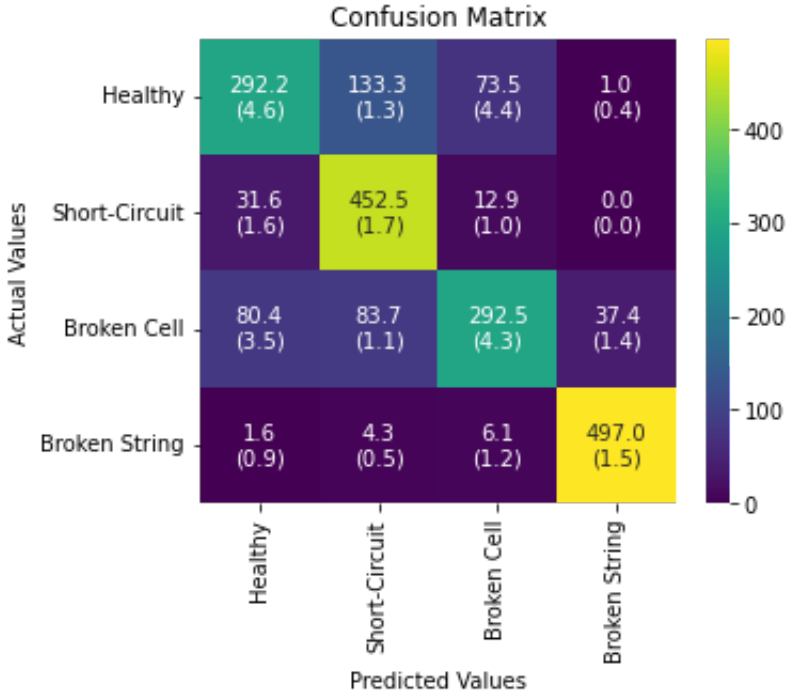


Figure 5.8: Confusion matrix of OVR-classifier using RF-algorithm for a combined short-circuit fault class with $R_{fault,max} = 1000\,\Omega$.

specificity of 0.614, F1-score 0.881 and AUC of 0.768. The best performance for multiclass classification was also achieved by using a random forest algorithm with a one-vs-rest classifier, using a combined short-circuit fault class instead of separate ground fault and line-line fault classes. Using 19 features and perfectly balanced classes, the average results for all classes were a recall of 0.766, precision of 0.770 and F1-score of 0.759.

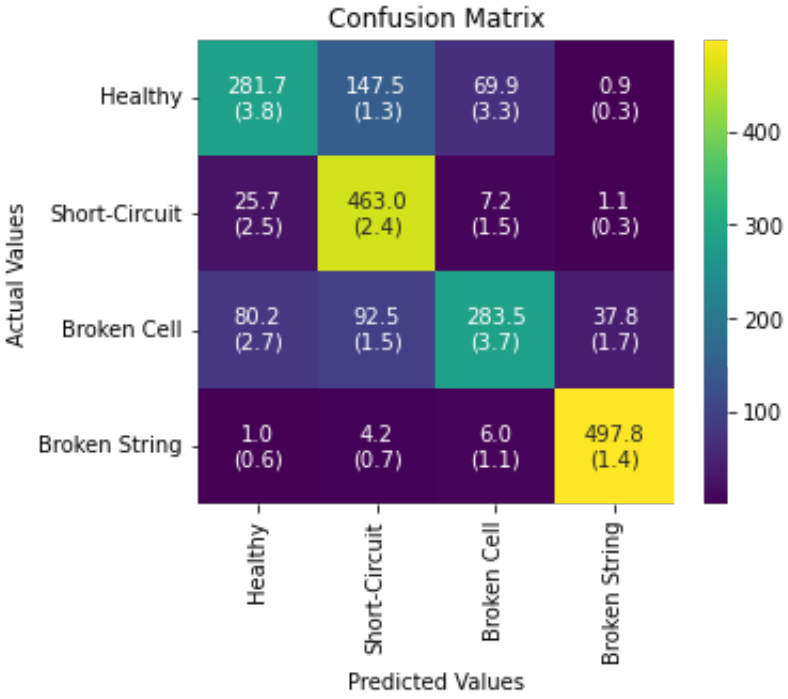


Figure 5.9: Confusion matrix of OVR-classifier using RF-algorithm for a combined short-circuit fault class with $R_{fault,max} = 100\,\Omega$ and $R_{fault,max} = 1000\,\Omega$.

6

Economics and Scalability

This chapter puts the findings of earlier chapters in perspective in terms of economics and scalability of the proposed method and also provides some points of discussion. In section 6.1 the economics of proposed method are discussed and section 6.2 discusses the scalability of this method.

6.1. Economics

The healthy systems in this study produce an average power of 6871 kWh per year. In order to know how much money could be saved by detecting and classifying faults in residential PV systems, the power lost due to faulting should be identified. However, it appears that the systems suffering from short-circuit faults in this study actually produce slightly more power on average: 6992 kWh per year. This is mainly because these systems have a lower age on average, as these faults occur relatively early in the system's lifetime. Despite this fact, the ML model is still capable of distinguishing most healthy systems and systems with short-circuit faulting. Also, fault resistance could decrease over time when such fault persists, and hence the power loss might increase.

If a power loss of 5.1% due to faulting is considered as stated in section 1.1, and the cost of electricity in the Netherlands is €0.226 per kWh in March 2021, a total of €79.20 per year per system could be saved. Considering these faults can go undetected for longer periods of time, this could add up significantly. Also, the risk of more severe faulting with system damage as result, is lowered. Bottlenecks in energy supply due to reopening economies after the pandemic, combined with the political conflict with Russia, caused soaring electricity prices. The average electricity price in the Netherlands in February 2022 has already increased to €0.378 per kWh. With such prices, savings of €132.46 per year per system could be reached. Average capital costs for a typical residential system as used in this study are €7350. Depending on the electricity price, over a 25-year system lifetime, 26.9-45.1% of capital cost could be saved.

The Netherlands is not a country with a high average irradiance. Compared to a typical system in Spain, where the average power production is 11000 kWh per year on average, the power produced by a system in the Netherlands is significantly lower. As the electricity price in Spain is also roughly 25% higher, the savings that could be achieved with fault detection, classification and subsequent maintenance, are two times the savings in the Netherlands. It is clear that fault detection and classification methods have a higher effect in countries with higher irradiance, higher electricity prices or a combination of the two. In the Netherlands, a total capacity of around 5500 MWp residential PV systems is currently installed. Would the power loss due to faulting be negated by correct detection and fault classification, savings up to €85M to €143M could be achieved.

All numbers mentioned in this section assume that all malfunctioning systems can be detected and repaired. In practice, this is not possible, which the results found in chapter 5 confirm. Also, the proposed method does not include all potential faults that could occur in a system. Therefore, a system might be misclassified and might suffer from a fault that is not included in this method. Of course, this is an issue that is very hard to solve, but by including the most common fault types in this study, this should be a minor issue.

However, both factors mentioned here will negatively impact the savings that could be achieved.

On the other hand, the hyperparameter tuning process in this study could be more lengthy. An extensive random search could lead to a solution closer to the optimal solution. Also, more hyperparameters could be included in the tuning process where currently a few hyperparameters are set to default values. The performance results presented in this thesis provide an approximation of the optimal results that could most likely be exceeded by a slight margin with a more detailed hyperparameter tuning process.

6.2. Scalability

The database generation model in the PVMD Toolbox is constructed to be scalable with respect to all system parameters. However, some parameters (number of cells, number of bypass diodes, number of modules/strings) have a significant impact on the relative power loss caused by the faults considered in this study. Therefore, the system's behavior under faulty operating conditions compared to a regular operating system might differ more or less than for the PV system in this study. This will most definitely impact the ML model performance.

The model does not consider the occurrence of multiple faults acting simultaneously. However, a combination of multiple ground- and line-line faults will likely already disrupt the system enough to trip one of the installed protection devices. Multiple broken strings (in a system with more than two strings) will most likely also be classified as a broken string fault by the machine learning system as the power loss is large compared to the other faults included in this study. Moreover, a broken string fault is highly unlikely to occur more than once in a relatively short period, and therefore not an interesting case to consider. The most problematic fault considered would be one of broken cells, which is currently already modeled to include multiple broken cells simultaneously. To tell how many cells are actually faulted is presently not predicted by the proposed method.

The scalability of data is the most concerning issue. The database used in this study consist of 25000 data points. This would take around 12 hours to generate and data exporting has to be done to be able to use the database for the ML model training. Especially the line-line fault cases take up a long time to generate. Decreasing the size of the database is not advisable, as can be seen in table 5.8. To reduce computation time, the code could be cleaned up, and the pipeline of using MATLAB to generate the training database and Python to perform the machine learning modeling could be streamlined.

Besides the long computation time, generating line-line faults scenarios the way it has currently been implemented in the Toolbox sometimes results in termination of the database generation. This occurs when irradiance is very low throughout the whole 48-hour period. The database generation process has to be manually restarted. This also causes the variation in meteorological conditions to be less for line-line fault scenarios than for the other fault classes and healthy systems.

7

Conclusion and Recommendations

In this final chapter, the conclusion is discussed, and future recommendations are proposed. In section 7.1 conclusions based on the results from chapters 2-5 are drawn and the sub-questions listed in section 1.5 are answered to ultimately answer the main research question. After that, section 7.2 provides some future recommendations to follow up on this study.

7.1. Conclusions

The objective of this study was to develop a method of detection and classification of faults in residential PV systems by training a machine learning model on a synthetically generated PV database and evaluating how accurate this method would be. Four research sub-questions were set up with one chapter devoted to each question to achieve this objective. In this section, these sub-questions and, subsequently the main research question are answered

7.1.1. Sub-Questions

What are the most common PV faults in residential PV systems and what is the impact of these faults on a system?

The most common PV faults that occur in residential PV systems are due to the failure of electronic components in the system: inverter, DC power combiner, PV optimizer and conversion devices. These failures account for 85% of PV system failures, of which 66% are due to inverter faults. The other 15% of faults or failures occur on the DC-side of the PV system. These are caused by open-circuit faults due to broken PV cells, PV modules or failure of bypass diodes, or short-circuit faults such as ground faults and line-to-line faults caused by accidental contact between two conducting parts of the system.

Open-circuit faults mainly impact the power output of a system. Due to bypass diodes included in PV systems, open-circuits can be bypassed, and the PV system can still operate in good conditions. Despite the decreasing efficiency of the system, open-circuit faults are relatively harmless in terms of damage to the system. That is why these faults can go unnoticed for extended periods of time, while the decrease in power production can be significant. By including two types of open-circuit faults in the form of broken module strings and broken cells, this unnecessary loss in system power output could be reduced with the proposed method in this study.

Short-circuit faults differ from open-circuit faults in the fact that these could be harmful to a PV system with ultimate risk of fire hazards. That already is a valid reason to include these faults in this study. Short-circuit faults also come in different forms, of which ground faults and line-to-line faults are included in this study. Short-circuit faults have a varying effect on the system depending on the fault resistance. The lower the fault resistance is, the larger the decrease in power and the higher the risk of damage to the system. Luckily, traditional protection devices installed in PV systems prevent system damage when the fault current is above a certain threshold. However, short-circuit faults can progress over time, making it more interesting to look into short-circuit faults that are not yet detected by these installed protection devices. Only short-circuit faults that do not trigger the protection devices are included in this study. As the fault current for these faults

is low, the output power loss of the system is significantly lower than for the open-circuit faults.

Inverter faults account for the largest part of PV system faults. It is one of the most important and most complex components of the system. Hence there is a very wide variety in inverter-related faults, and therefore it is hard to include these in this study. Also, short-circuit faults can make the inverter trip which is then often incorrectly marked as an inverter fault, when the inverter did not actually cause the failure. Therefore the number of inverter faults is likely to be inflated.

How can a synthetic PV training database be generated for PV fault detection and fault classification in residential PV-systems?
The first step in creating a synthetic PV database is finding a tool capable of modeling a PV system. In this study, the PVMD Toolbox of The Photovoltaic Materials and Devices (PVMD) group from Delft University of Technology was used to do PV yield predictions. Subsequently, a representative healthy PV system can be modeled with all the correct design parameters of the system. Once a healthy PV system is modeled, and the system's power output can be simulated, varying scenarios should be created such that a database can be constructed. By randomly sampling weather data profiles from different days, using Meteonorm data, varying weather conditions can be simulated. System age was varied by sampling the age from failure probability distributions for faulty systems or from an age distribution of existing installed PV systems for healthy systems. Different faults have different failure probability distributions: some faults are more likely to occur in the early stages, whereas other faults occur more often at the end of the system lifetime. After the modeling of a healthy PV system in the PVMD Toolbox, and creating varying system scenarios by varying the weather conditions and the system age, the implementation of faulty PV systems in the Toolbox was done by data manipulation in between different simulation steps: voltages and current were changed to match the effect of certain faults on the system.

Finally, the PV yield simulation was performed a total of 25000 times for 25000 different scenarios of the same system. Of all simulations, 15000 times a healthy system was simulated, and 10000 times it was a faulty system. Every simulation results in a 48-hour system power profile of hourly values. Alongside the power profiles, a 48-hour weather profile including the irradiance, sun elevation, sun azimuth and ambient temperature are stored. After every simulation, a random hour is picked for every simulation from which the AC power, irradiance, sun elevation and azimuth, and temperature is taken. Combined with the system age, the six variables are stored as the six standard features for the machine learning model. Together with these six features, the system status is added as label, being one of the following: healthy, ground fault, line-line fault, broken cells fault or broken string fault. Therefore the synthetically generated PV training database has 25000 rows, with each row containing a value for irradiance, temperature, sun elevation, sun azimuth, system AC power, system age, and the seventh column containing the system status label. By picking a random hour, the produced scenarios will vary more in terms of meteorological conditions. In the later stages of the machine learning model design, extra features are: the irradiance and power production on the same hour the day before, as well as the irradiance and power at 10 AM, 1 PM and 4 PM at the day before. At all these hours, the ratio of irradiance and power (power ratio) were also used as features, adding up to a total of 19 features. By adding these extra features, the time element is taken into account in some form without including complete 48-hour profiles.

How to design a machine learning model for PV fault detection and classification?
The first step in building a machine learning model is to identify the type of data to be used, and subsequently find the corresponding learning type. In this study, labeled data was created with a discrete output. Therefore classification, a type of supervised learning, is the right approach. For PV fault detection, binary classification was used, and for the fault classification, multiclass classification was performed.

Subsequently, a machine learning algorithm has to be picked. In the field of PV fault detection and classification with machine learning, mostly four algorithms are used: decision trees, random forests, support vector machines, and multilayer perceptron neural networks. Therefore four different models were designed using these algorithms. Subsequently, the hyperparameter values of the algorithms should be defined. Hyperparameters differ for each algorithm, and there are optimal hyperparameter values for the best model performance. When starting the model training, default starting values for the hyperparameters can be used. Once the models have been built, the model optimization starts.

The generated training database is split into three sets: training, validation and test. The training set is used to train the four models. The validation set is used to test the performance of the models during the hyperparameter tuning process. By changing the hyperparameter values, the model performance changes. Once a good performance on the validation set is found, the hyperparameter values can be fixed. Afterward, the test set can be used to get unbiased results, as the model has never 'seen' the test set data. Model performance is evaluated by the following metrics: accuracy, precision, recall, specificity, F1-score and AUC, combined with confusion matrices and ROC curve plots. Increasing the score for these metrics means the models can better detect and classify the PV faults.

What is the performance of different machine learning algorithms trained on a synthetic PV database for fault detection and classification?
A complete overview of all results can be found in appendix C and D. Here the best performing models are highlighted. For fault detection, using binary classification, the random forest model achieved the best performance with a dataset balance of 50% faulty system and 19 features: an accuracy of 80.4% was achieved. By increasing the ratio of faulty systems to 70%, the results could be considered even better if detecting faulty systems is considered more important: an accuracy of 82.8%.

In case 1, with the addition of additional features and inclusion of data from the previous day, the performance increases significantly across all performance metrics for all models. Case 2a greatly shows the benefit of the proposed method of a synthetic training database as it is shown that optimal model performance is obtained for a dataset with around 50%. Case 2b showcases that only balancing the training set already greatly increases the number of correctly detected faulty systems for a skewed dataset. Case 2c shows that different models are more suitable depending on the balance in the dataset. The support vector machine model outperforms all other models for very skewed datasets, whereas the random forest model outperforms all other models when the percentage of faulty systems in the dataset increases to 50% and higher. Case 2d concludes the binary classification with the result that showcases dataset size indeed greatly affects all models' performance. The neural network model performance increases more than the other models. With larger datasets, a neural network might outperform the random forest model and be the optimal model for PV fault detection.

For fault classification, using multiclass classification, the best performance was also achieved by the random forest algorithm with a one-vs-rest classifier, while combining ground faults and line-line faults into a single short-circuit fault class. Again, 19 features were used, and the classes were perfectly balanced. The average results for all classes were a recall of 0.766, precision of 0.770 and F1-score of 0.759. Broken string faults are classified most easily with an F1-score of 0.952. After that comes the short-circuit fault class with a 0.773 F1-score. The healthy and broken cell fault classes are classified worst with an F1-score of 0.645 and 0.666 respectively.

7.1.2. Main Research Question

How accurate can an ML-based model detecting and classifying faults in residential PV systems get when using a synthetic PV training database?
An ML-based model used for the detection and classification of faults in residential PV systems trained on a synthetic PV database, as proposed in this study, can get 80.4% accurate in detecting malfunctioning systems and can achieve a recall of 0.766, precision of 0.770 and F1-score of 0.759 for fault classification. In practice, this performance is on the lower side, and lots of systems would still be misclassified. However, considering the very limited resources necessary for this approach, where a PV system owner should only provide the system parameters and PV yield data, this is a good first approach towards a fault detection and classification method for residential PV systems specifically.

Moreover, this study shows the benefit of using a synthetic database as the dataset balance can be controlled, achieving better results. This is true for balancing the training set only as well as balancing both training and test set. Also a database can be generated specifically towards a single system, which theoretically should outperform a model trained on a database with lots of different systems.

It is hard to put the model performance obtained in this study into perspective as there is no literature

available using a combination of a similar setup with similar fault types for varying irradiance levels. A study by Zhao also involving line-line faults ($R_{fault} = 20\,\Omega$) and broken string faults (on a 2-string system) reports fault detection and fault classification accuracy of 93.56% and 85.43% respectively, for a simple decision tree model of 11 nodes [10]. However, this study used 764529 data points, which is more than 30 times as much. Also, besides similar features being irradiance, system power and ambient temperature, other features were included, such as operating temperature, array current, array voltage, fill factor and more. Most interesting of this study is the performance on the classification of on line-line faults with $R_{fault} = 20\,\Omega$ as it is similar to the approach of considering short-circuit faults with relatively high fault resistance and thus a very minimal decrease in system power. The classification accuracy on this class for the simple DT model was reported to be 0%. Unfortunately, no classification accuracy is reported for the individual classes for the more complex DT models. The general accuracy for detection and classification reaches 99.98% and 99.8% respectively, for a very complex DT model with 1637 nodes and 819 leaf nodes. Again, it is tough to compare the results with the results obtained in the study by Zhao as they used a lot more data points and more features that are not available in most residential PV systems.

A study by Chine uses a similar ANN to detect the following faults: short-circuit fault in any bypass diode or cell or module, inversed bypass diode fault or cell or module, shunted bypass diode fault or cell or module, connection resistance between PV modules [8]. A general classification accuracy of 90.3% is reached. However, the availability of system voltage and current was again used to train the model.

7.2. Recommendations

This section presents recommendations for possible future research building on this study. These recommendations are either looking to improve the model's accuracy in the detection and classification of PV failures, improve the quality of the synthetic data and its resemblance to realistic data, or expand on the proposed framework in this study.

First of all, another approach to increase the realisticness of the fault impact on PV systems would be to expand the Toolbox so that individual cells, modules etcetera, can be defined. As of now, cell and module parameters are defined and then extrapolated to a complete PV system. Therefore, it is not yet possible to introduce a fault in one of the cells or modules only, and creative ways to implement such a fault had to be found such that the final output of the system would resemble a system with such a fault. If individual components could be defined separately, this would increase the accuracy of the fault impact on a system, for example, due to a correct system temperature calculation, which is not homogeneous throughout a real system. However, computation time would be significantly longer when simulating all cells and modules separately. If such a feature would be added to the Toolbox so that the computation time is not increased too much, this could benefit the model's accuracy in detecting and classifying faults of real PV systems.

Secondly, there is a great lack of information on failure probability distributions in literature, and more research on this topic could increase the performance of this method. Currently, the age of systems with ground faults and line-line faults are sampled from the same distribution, and moreover, this distribution is somewhat arbitrary. Suppose more knowledge on this topic would be available. In that case, system age as a feature could better contribute to the ML model performance in the detection and classification of PV faults in residential systems. Would this method be used in the future, data of faulty systems could be gathered to improve the proposed method and hence the performance. A more accurate degradation model would also benefit from the resemblance of realistic PV data, which is already worked on currently and could be added for future research.

Thirdly, additional research could be done on the inclusion of system current and voltage as features for the machine learning model. Some inverters do also show this information, but the question is how much inverters do this, and what the effect on the model performance would be. Hypothetically ground faults and line-line faults could be kept as separate classes in the multiclass classification process, as these two faults impact PV systems differently with respect to voltage and current. This could also benefit the correct classification of healthy systems.

Fourthly, it would be interesting to use an RNN instead of an ANN so that complete 48-hour profiles could

be included in the detection and classification process, as an RNN can deal with temporal data. This might result in a performance increase for the neural network model and possibly gives better results than the random forest model, which outperformed the ANN used in this study.

Fifthly, the proposed method could be tested on a real PV system, to confirm accurate operation of the proposed method. Although the synthetic PV training database was carefully generated to resemble a real PV system, it is never possible to exactly mimic a real PV system.

Sixthly, it would be interesting to develop a model that quantifies the mismatch between system parameters of residential PV systems in an open database or cloud, compared to the actual parameters of a system in place. In this study, the system parameters are assumed to be completely correct, whereas open databases often have missing or incorrect data. Insight into this could possibly be useful for the database generation process.

Lastly, a better cost analysis could be performed for practical reasons. What are the labor costs and repair costs, and how does this impact the results of this study? Does a misclassified system cost more than is gained from a correctly classified system? With this knowledge, weights could be assigned to the true positives, true negatives, etcetera. This could considerably change the conclusions drawn in this study. A different database balance might be preferable to what is currently considered to be optimal.

A

System Parameters

- Cell and module parameters
  - Cell type = Monocrystalline silicon passivated emitter and rear cell (Mono PERC)
  - Module tilt = 40°
  - Module azimuth = 0°
  - Module height above ground = 50 cm
  - Number of cell rows = 12
  - Number of cell columns = 6
  - Module thickness = 0.5 cm
  - Module cell spacing = 0.3 cm
  - Module edge spacing = 1 cm
  - Cell length = 15.675 cm
  - Cell width = 15.675 cm
  - Module albedo = 0.2
- Weather and thermal parameters
  - Longitude = 52.011753
  - Latitude = 4.359364
  - Efficiency of PV module = 0.25 (only used for temperature calculation)
  - Temperature Coefficient = -0.0035 $\frac{1}{°C}$
  - Glass thickness = 0.4 cm
- Electrical and conversion parameters
  - Shading loss due to metallization = 3%
  - Metal grid resistance = 0.0058 Ω
  - Number of bypass diodes = 3
  - Inverter type = central
  - Inverter capacity = 5000 W
  - Number of modules in parallel = 2
  - Number of modules in series = 7
  - Cable losses = 1%

B

Machine Learning Model Hyperparameters

B.1. Binary Classification Models

- Decision Tree
  - max_depth: 15
  - min_samples_split: 2
  - min_samples_leaf: 1
  - max_leaf_nodes: 150
  - ccp_alpha: 0.00001
- Random Forest
  - n_estimators: 50
  - max_depth: 15
  - min_samples_split: 2
  - min_samples_leaf: 1
  - max_leaf_nodes: 400
  - ccp_alpha: 0.00001
- Support Vector Machine
  - C: 200
  - gamma: 0.05
- Neural Network
  - hidden_layer_sizes: 20, 20, 10
  - activation: ReLu
  - solver: adam
  - learning_rate_init: 0.007
  - max_iter: 2000
  - n_iter_no_change: 25

B.2. Multiclass Classification Models

- Decision Tree
  - max_depth: 14
  - min_samples_split: 2
  - min_samples_leaf: 1
  - max_leaf_nodes: 100
  - ccp_alpha: 0.00001
- Random Forest
  - n_estimators: 70
  - max_depth: 14
  - min_samples_split: 2
  - min_samples_leaf: 1
  - max_leaf_nodes: 100
  - ccp_alpha: 0.00001
- Support Vector Machine
  - C: 200
  - gamma: default
- Neural Network
  - hidden_layer_sizes: 15, 15
  - activation: ReLu
  - solver: adam
  - learning_rate_init: 0.004
  - max_iter: 2000
  - n_iter_no_change: 25

C

Results Binary Classification

C.1. Case 1

Table C.1: Performance of decision tree model on test set for different number of features.

Metric	6 features	13 features	19 features
Accuracy	0.746 (0.000)	0.796 (0.001)	0.791 (0.000)
Recall	0.682 (0.001)	0.744 (0.001)	0.724 (0.002)
Precision	0.680 (0.000)	0.743 (0.001)	0.743 (0.001)
Specificity	0.788 (0.000)	0.830 (0.001)	0.835 (0.001)
F1	0.681 (0.000)	0.743 (0.001)	0.733 (0.001)
AUC	0.735 (0.000)	0.787 (0.001)	0.779 (0.000)

Table C.2: Performance of random forest model on test set for different number of features.

Metric	6 features	13 features	19 features
Accuracy	0.743 (0.003)	0.803 (0.003)	0.805 (0.002)
Recall	0.648 (0.006)	0.713 (0.008)	0.710 (0.006)
Precision	0.687 (0.004)	0.774 (0.005)	0.780 (0.004)
Specificity	0.806 (0.002)	0.862 (0.004)	0.868 (0.004)
F1	0.667 (0.005)	0.742 (0.005)	0.744 (0.003)
AUC	0.727 (0.004)	0.788 (0.004)	0.789 (0.002)

Table C.3: Performance of support vector machine model on test set for different number of features.

Metric	6 features	13 features	19 features
Accuracy	0.722 (0.000)	0.787 (0.000)	0.797 (0.000)
Recall	0.621 (0.000)	0.625 (0.000)	0.637 (0.000)
Precision	0.659 (0.000)	0.796 (0.000)	0.811 (0.000)
Specificity	0.788 (0.000)	0.894 (0.000)	0.902 (0.000)
F1	0.639 (0.000)	0.700 (0.000)	0.714 (0.000)
AUC	0.704 (0.000)	0.760 (0.000)	0.770 (0.000)

Table C.4: Performance of neural network model on test set for different number of features.

Metric	6 features	13 features	19 features
Accuracy	0.791 (0.007)	0.814 (0.007)	0.810 (0.004)
Recall	0.723 (0.027)	0.763 (0.024)	0.771 (0.022)
Precision	0.745 (0.016)	0.769 (0.015)	0.756 (0.016)
Specificity	0.837 (0.018)	0.848 (0.015)	0.835 (0.018)
F1	0.733 (0.011)	0.765 (0.011)	0.763 (0.006)
AUC	0.780 (0.008)	0.805 (0.009)	0.803 (0.005)

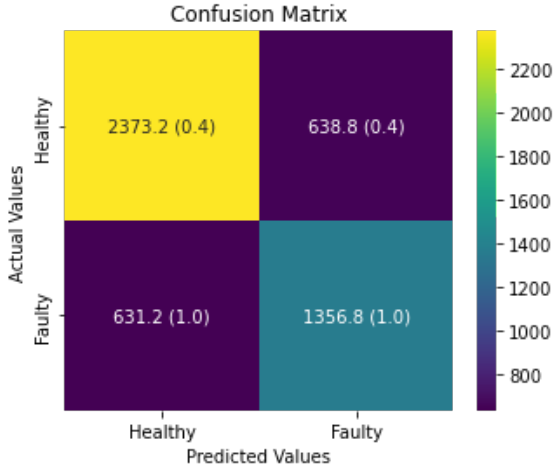


Figure C.1: Confusion matrix of decision tree model on test set for 6 features.

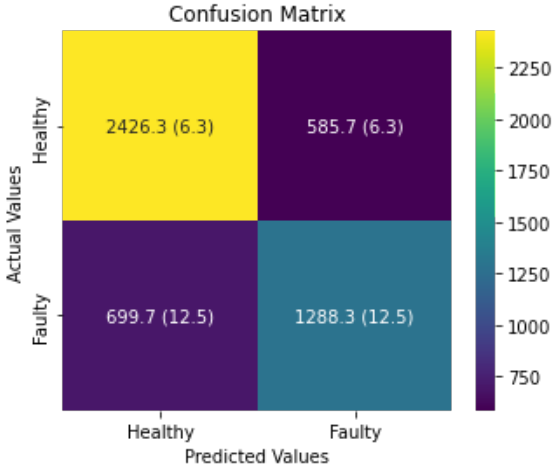


Figure C.2: Confusion matrix of random forest model on test set for 6 features.

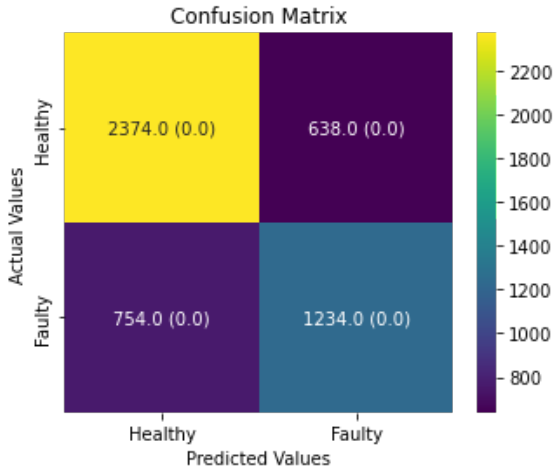


Figure C.3: Confusion matrix of support vector machine model on test set for 6 features.

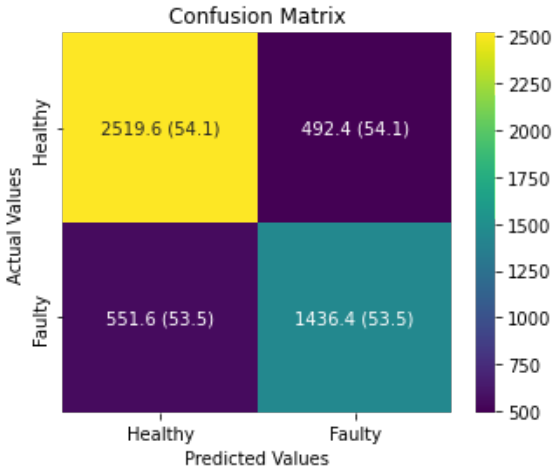


Figure C.4: Confusion matrix of neural network model on test set for 6 features.

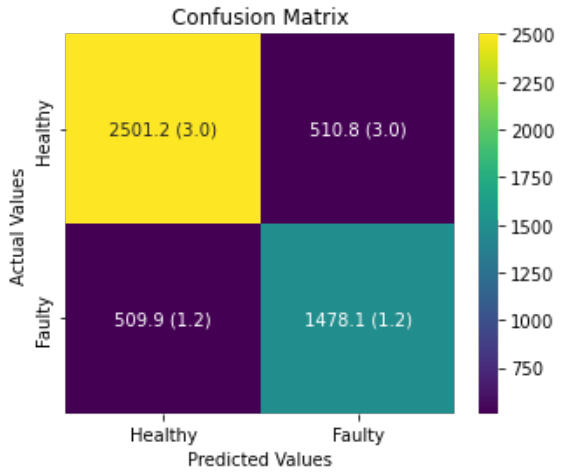


Figure C.5: Confusion matrix of decision tree model on test set for 13 features.

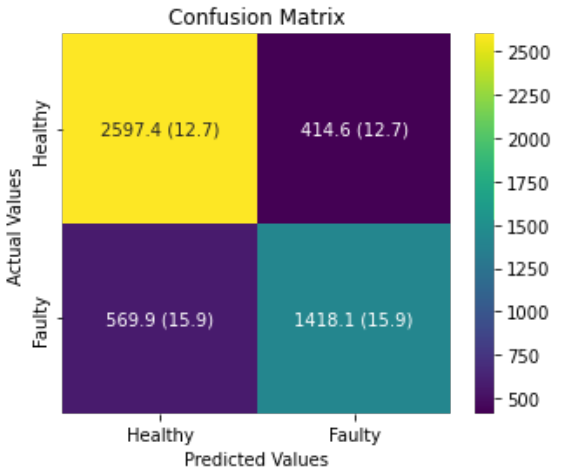


Figure C.6: Confusion matrix of random forest model on test set for 13 features.

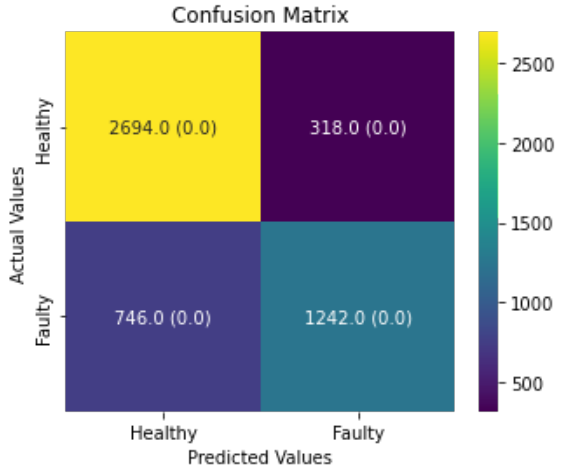


Figure C.7: Confusion matrix of support vector machine model on test set for 13 features.

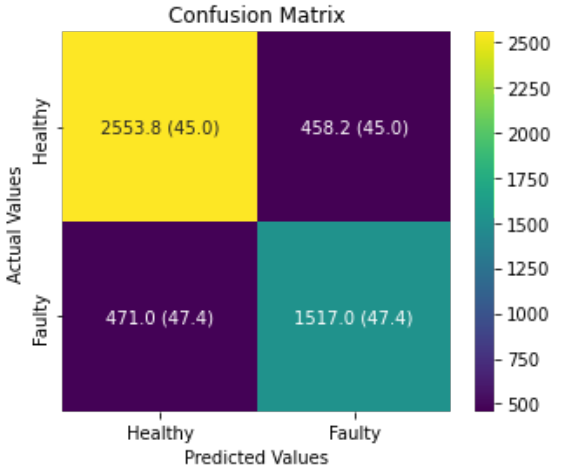


Figure C.8: Confusion matrix of neural network model on test set for 13 features.

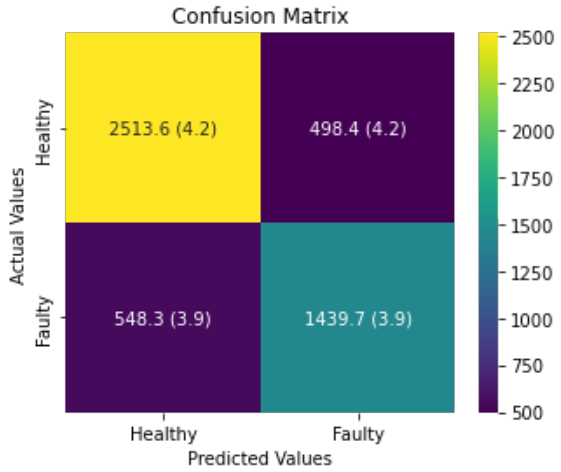


Figure C.9: Confusion matrix of decision tree model on test set for 19 features.

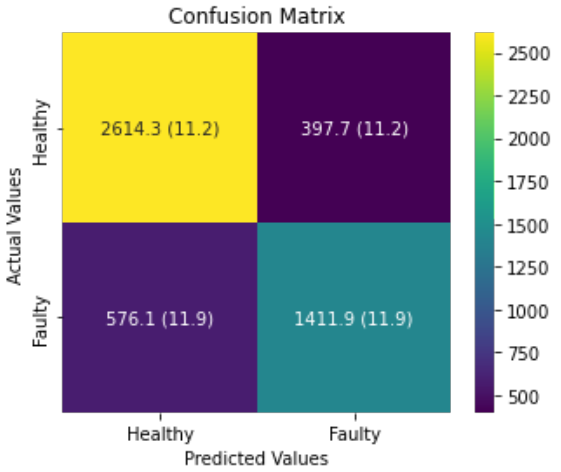


Figure C.10: Confusion matrix of random forest model on test set for 19 features.

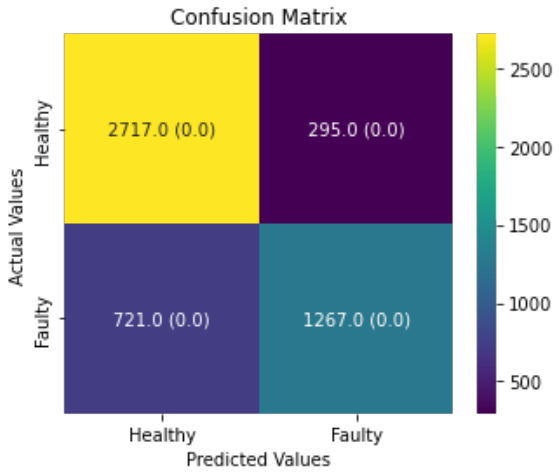


Figure C.11: Confusion matrix of support vector machine model on test set for 19 features.

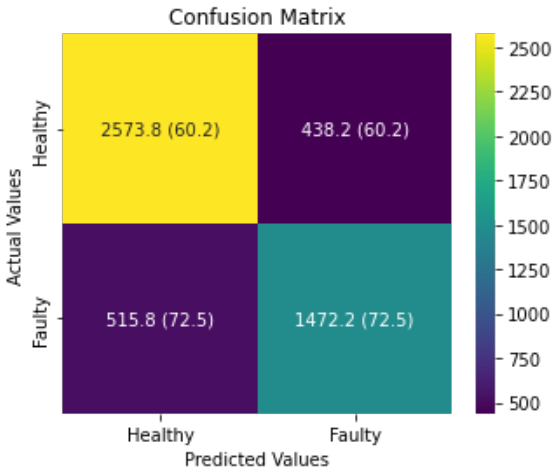


Figure C.12: Confusion matrix of neural network model on test set for 19 features.

C.2. Case 2a

Table C.5: Performance of decision tree model on test set for varying ratio of faulty systems in dataset.

Metric	1%	5%	10%	20%	30%	40%	50%	60%	70%
Accuracy	0.988 (0.001)	0.956 (0.001)	0.917 (0.001)	0.844 (0.001)	0.798 (0.001)	0.776 (0.001)	0.795 (0.001)	0.788 (0.001)	0.805 (0.001)
Recall	0.296 (0.013)	0.319 (0.007)	0.421 (0.005)	0.416 (0.003)	0.606 (0.002)	0.711 (0.001)	0.816 (0.001)	0.835 (0.001)	0.895 (0.001)
Precision	0.289 (0.034)	0.549 (0.025)	0.619 (0.008)	0.693 (0.004)	0.696 (0.001)	0.738 (0.001)	0.791 (0.000)	0.820 (0.001)	0.836 (0.001)
Specificity	0.994 (0.001)	0.987 (0.001)	0.972 (0.001)	0.953 (0.001)	0.883 (0.001)	0.821 (0.001)	0.772 (0.000)	0.716 (0.002)	0.602 (0.003)
F1	0.292 (0.023)	0.403 (0.010)	0.501 (0.005)	0.520 (0.003)	0.648 (0.001)	0.724 (0.001)	0.803 (0.001)	0.827 (0.001)	0.864 (0.000)
AUC	0.645 (0.007)	0.653 (0.003)	0.696 (0.002)	0.685 (0.002)	0.744 (0.001)	0.766 (0.001)	0.794 (0.001)	0.775 (0.001)	0.749 (0.001)

Table C.6: Performance of random forest model on test set for varying ratio of faulty systems in dataset.

Metric	1%	5%	10%	20%	30%	40%	50%	60%	70%
Accuracy	0.994 (0.000)	0.966 (0.000)	0.934 (0.001)	0.868 (0.001)	0.808 (0.003)	0.794 (0.002)	0.804 (0.003)	0.821 (0.002)	0.828 (0.002)
Recall	0.292 (0.000)	0.271 (0.003)	0.349 (0.005)	0.373 (0.003)	0.457 (0.011)	0.678 (0.003)	0.827 (0.003)	0.884 (0.003)	0.922 (0.001)
Precision	1.000 (0.000)	0.980 (0.020)	0.967 (0.009)	0.934 (0.014)	0.847 (0.013)	0.795 (0.005)	0.797 (0.004)	0.832 (0.002)	0.844 (0.002)
Specificity	1.000 (0.000)	1.000 (0.000)	0.999 (0.000)	0.993 (0.002)	0.963 (0.004)	0.876 (0.004)	0.778 (0.005)	0.724 (0.005)	0.614 (0.006)
F1	0.452 (0.000)	0.425 (0.004)	0.512 (0.006)	0.533 (0.004)	0.594 (0.008)	0.732 (0.003)	0.812 (0.003)	0.857 (0.002)	0.881 (0.001)
AUC	0.646 (0.000)	0.636 (0.002)	0.674 (0.003)	0.683 (0.002)	0.710 (0.004)	0.777 (0.002)	0.803 (0.003)	0.804 (0.002)	0.768 (0.003)

Table C.7: Performance of support vector machine model on test set for varying ratio of faulty systems in dataset.

Metric	1%	5%	10%	20%	30%	40%	50%	60%	70%
Accuracy	0.994 (0.000)	0.967 (0.000)	0.930 (0.000)	0.875 (0.000)	0.810 (0.000)	0.785 (0.000)	0.767 (0.000)	0.767 (0.000)	0.793 (0.000)
Recall	0.500 (0.000)	0.393 (0.000)	0.438 (0.000)	0.499 (0.000)	0.556 (0.000)	0.652 (0.000)	0.739 (0.000)	0.806 (0.000)	0.889 (0.000)
Precision	0.706 (0.000)	0.809 (0.000)	0.751 (0.000)	0.808 (0.000)	0.763 (0.000)	0.792 (0.000)	0.792 (0.000)	0.810 (0.000)	0.826 (0.000)
Specificity	0.998 (0.000)	0.995 (0.000)	0.984 (0.000)	0.970 (0.000)	0.923 (0.000)	0.879 (0.000)	0.796 (0.000)	0.707 (0.000)	0.576 (0.000)
F1	0.585 (0.000)	0.529 (0.000)	0.553 (0.000)	0.617 (0.000)	0.643 (0.000)	0.715 (0.000)	0.765 (0.000)	0.808 (0.000)	0.856 (0.000)
AUC	0.749 (0.000)	0.694 (0.000)	0.711 (0.000)	0.735 (0.000)	0.740 (0.000)	0.765 (0.000)	0.767 (0.000)	0.756 (0.000)	0.733 (0.000)

Table C.8: Performance of neural network model on test set for varying ratio of faulty systems in dataset.

Metric	1%	5%	10%	20%	30%	40%	50%	60%	70%
Accuracy	0.993 (0.001)	0.956 (0.007)	0.918 (0.005)	0.853 (0.006)	0.808 (0.005)	0.790 (0.005)	0.793 (0.009)	0.798 (0.005)	0.812 (0.005)
Recall	0.509 (0.061)	0.365 (0.027)	0.466 (0.015)	0.529 (0.041)	0.631 (0.030)	0.723 (0.021)	0.804 (0.026)	0.854 (0.016)	0.889 (0.008)
Precision	0.602 (0.117)	0.562 (0.108)	0.617 (0.044)	0.678 (0.038)	0.713 (0.017)	0.759 (0.014)	0.795 (0.008)	0.822 (0.010)	0.847 (0.005)
Specificity	0.997 (0.002)	0.984 (0.008)	0.968 (0.007)	0.935 (0.015)	0.887 (0.014)	0.837 (0.017)	0.780 (0.016)	0.712 (0.024)	0.637 (0.017)
F1	0.540 (0.039)	0.436 (0.029)	0.530 (0.012)	0.592 (0.018)	0.669 (0.013)	0.740 (0.007)	0.799 (0.012)	0.837 (0.005)	0.868 (0.004)
AUC	0.753 (0.030)	0.675 (0.011)	0.717 (0.005)	0.732 (0.014)	0.759 (0.010)	0.780 (0.005)	0.792 (0.008)	0.783 (0.006)	0.763 (0.007)

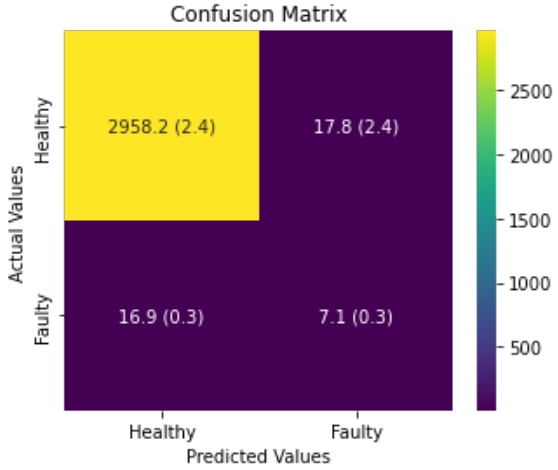


Figure C.13: Confusion matrix of decision tree model on test set for 1% of faulty systems in dataset.

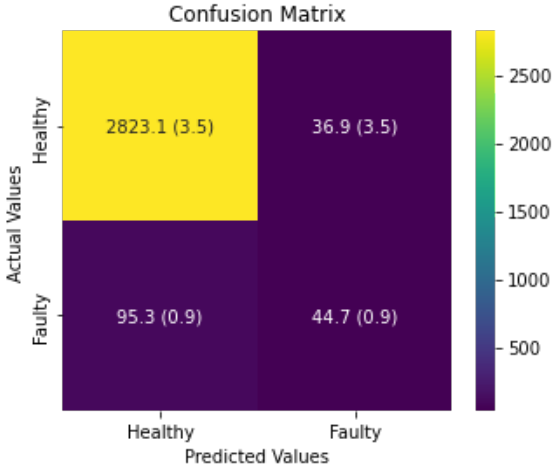


Figure C.14: Confusion matrix of decision tree model on test set for 5% of faulty systems in dataset.

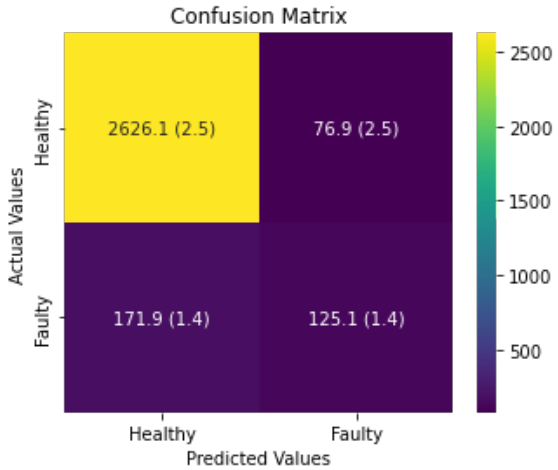


Figure C.15: Confusion matrix of decision tree model on test set for 10% of faulty systems in dataset.

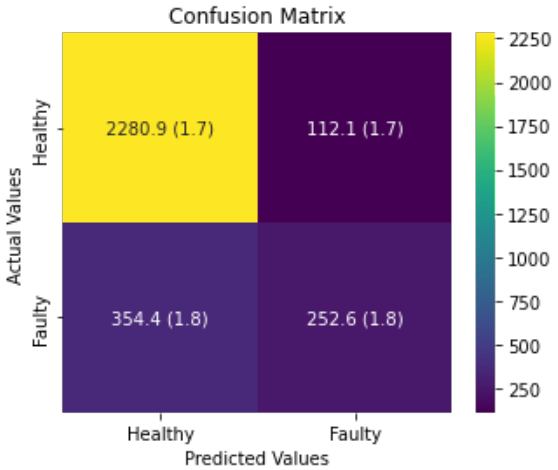


Figure C.16: Confusion matrix of decision tree model on test set for 20% of faulty systems in dataset.

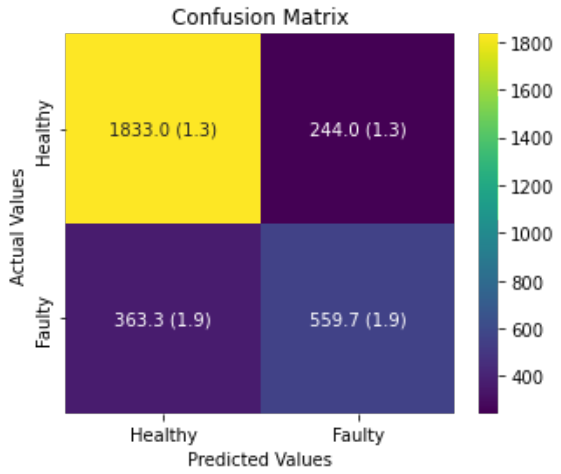


Figure C.17: Confusion matrix of decision tree model on test set for 30% of faulty systems in dataset.

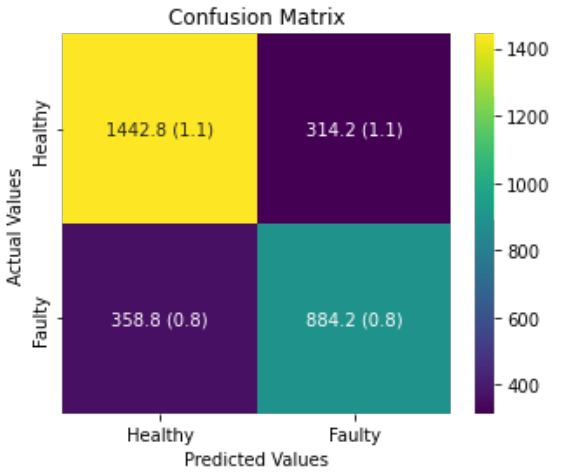


Figure C.18: Confusion matrix of decision tree model on test set for 40% of faulty systems in dataset.

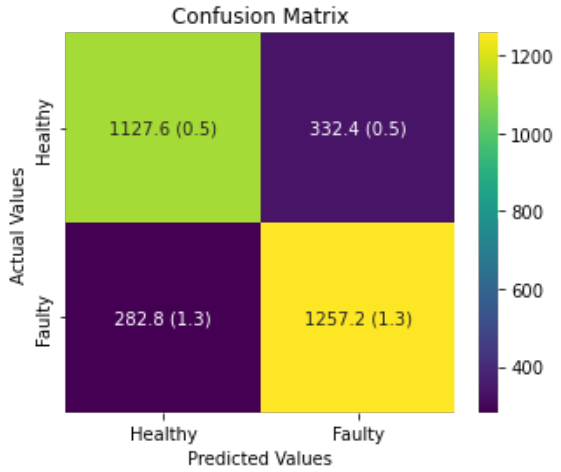


Figure C.19: Confusion matrix of decision tree model on test set for 50% of faulty systems in dataset.

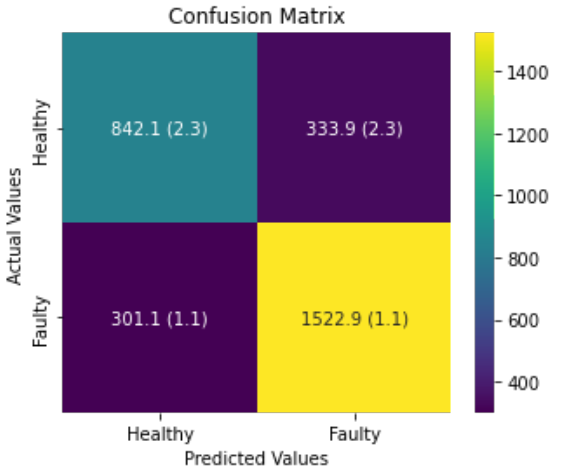


Figure C.20: Confusion matrix of decision tree model on test set for 60% of faulty systems in dataset.

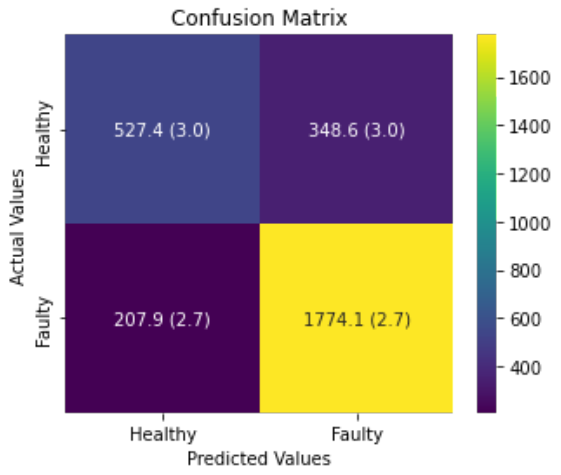


Figure C.21: Confusion matrix of decision tree model on test set for 70% of faulty systems in dataset.

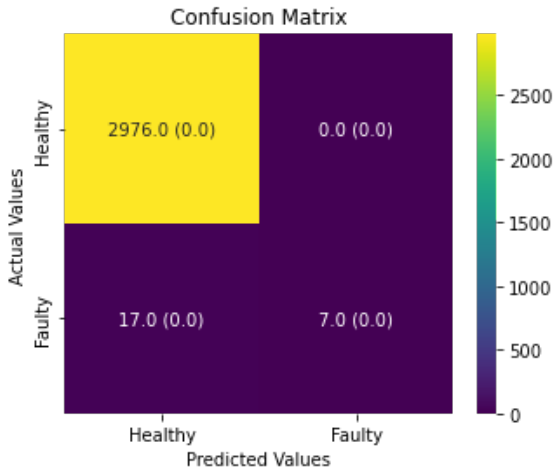


Figure C.22: Confusion matrix of random forest model on test set for 1% of faulty systems in dataset.

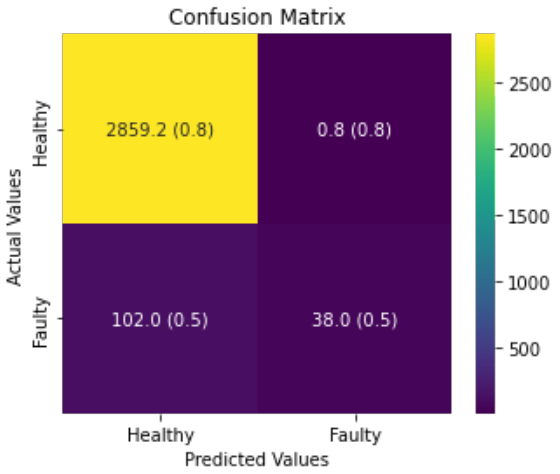


Figure C.23: Confusion matrix of random forest model on test set for 5% of faulty systems in dataset.

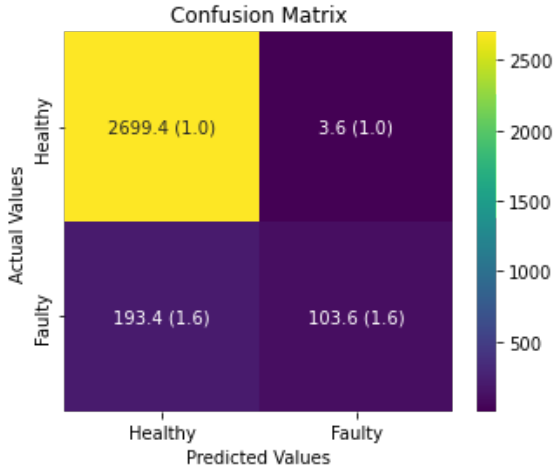


Figure C.24: Confusion matrix of random forest model on test set for 10% of faulty systems in dataset.

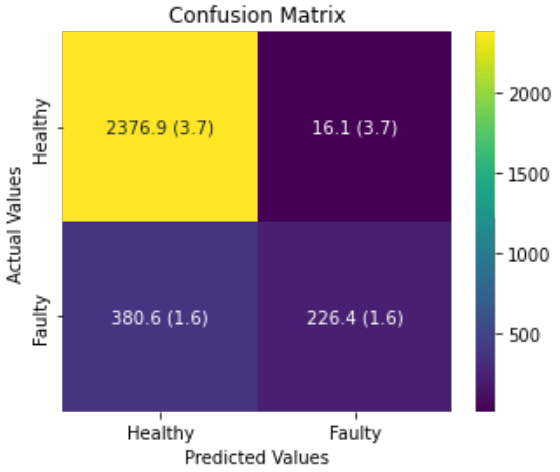


Figure C.25: Confusion matrix of random forest model on test set for 20% of faulty systems in dataset.

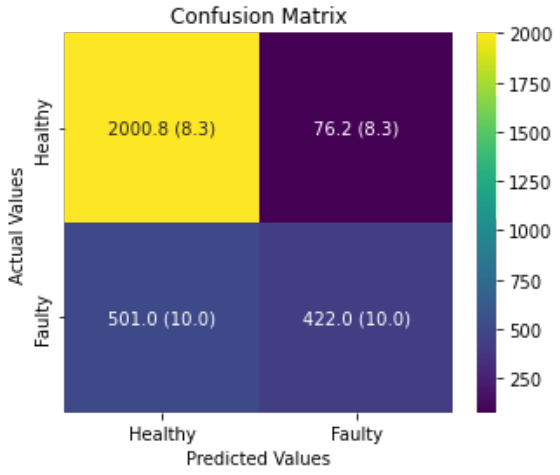


Figure C.26: Confusion matrix of random forest model on test set for 30% of faulty systems in dataset.

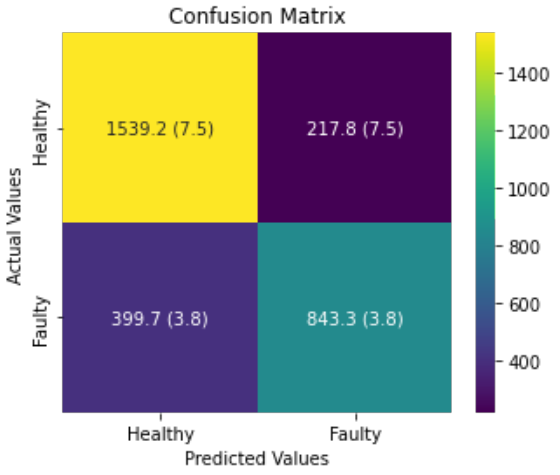


Figure C.27: Confusion matrix of random forest model on test set for 40% of faulty systems in dataset.

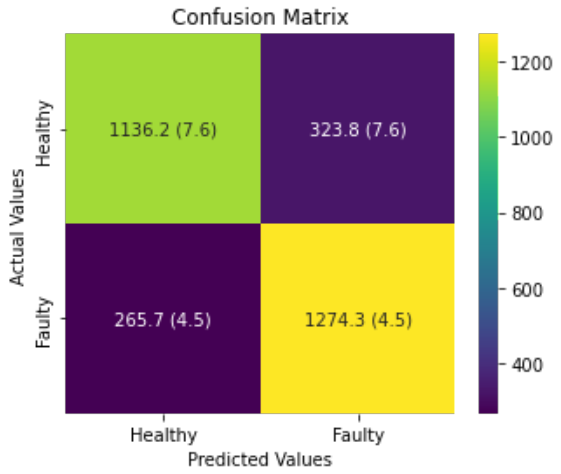


Figure C.28: Confusion matrix of random forest model on test set for 50% of faulty systems in dataset.

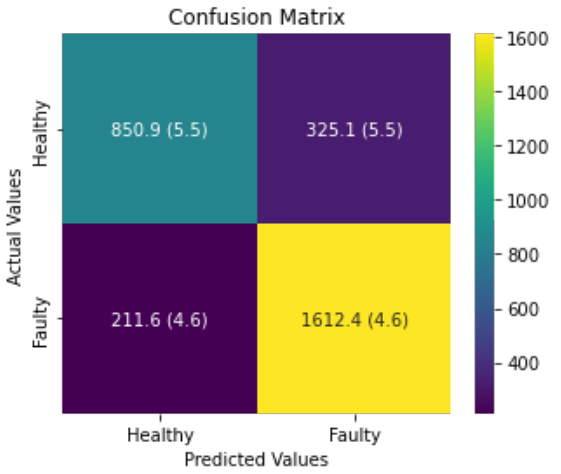


Figure C.29: Confusion matrix of random forest model on test set for 60% of faulty systems in dataset.

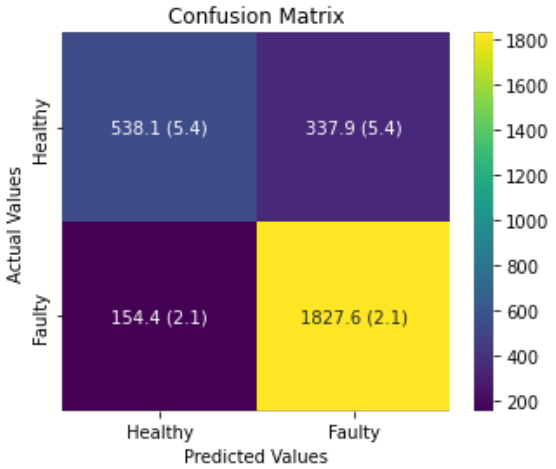


Figure C.30: Confusion matrix of random forest model on test set for 70% of faulty systems in dataset.

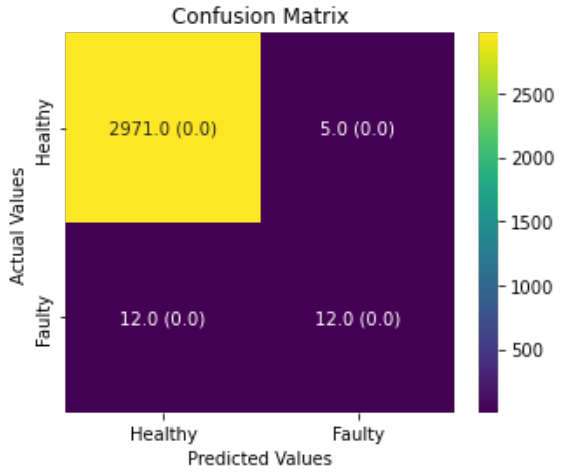


Figure C.31: Confusion matrix of support vector machine model on test set for 1% of faulty systems in dataset.

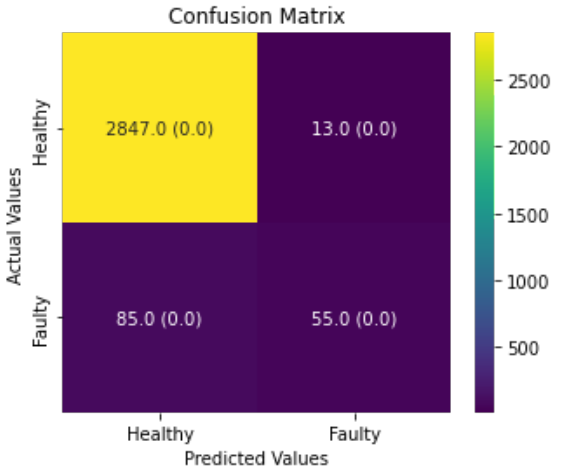


Figure C.32: Confusion matrix of support vector machine model on test set for 5% of faulty systems in dataset.

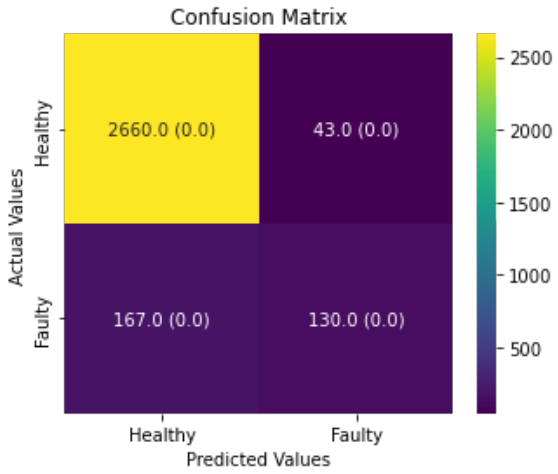


Figure C.33: Confusion matrix of support vector machine model on test set for 10% of faulty systems in dataset.

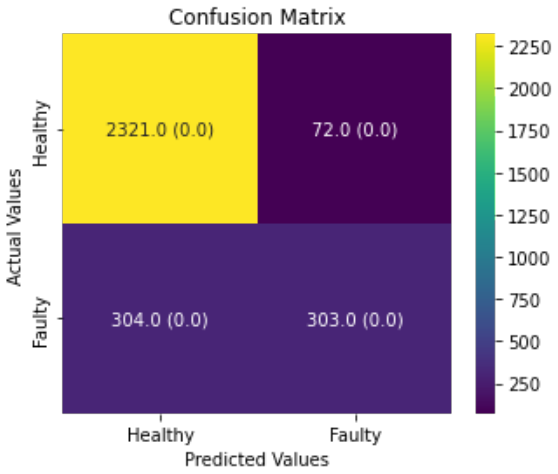


Figure C.34: Confusion matrix of support vector machine model on test set for 20% of faulty systems in dataset.

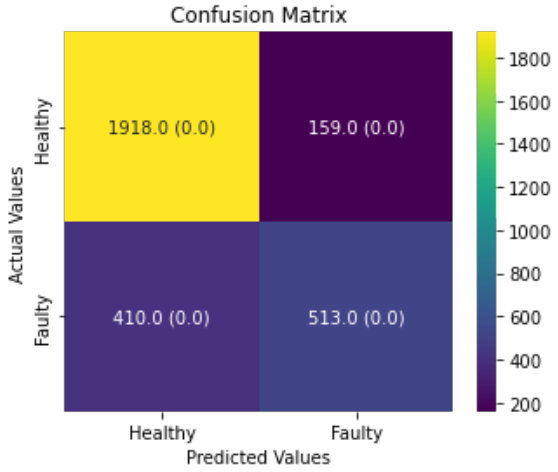


Figure C.35: Confusion matrix of support vector machine model on test set for 30% of faulty systems in dataset.

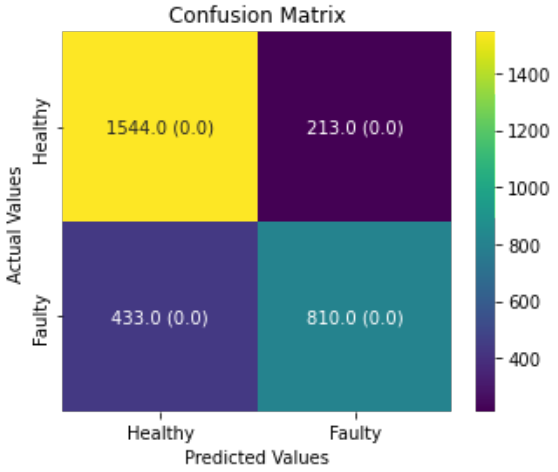


Figure C.36: Confusion matrix of support vector machine model on test set for 40% of faulty systems in dataset.

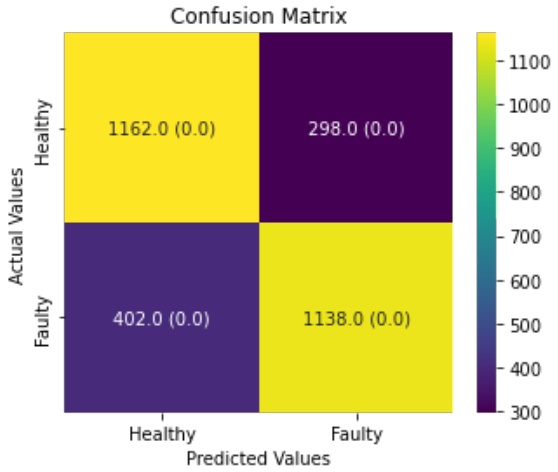


Figure C.37: Confusion matrix of support vector machine model on test set for 50% of faulty systems in dataset.

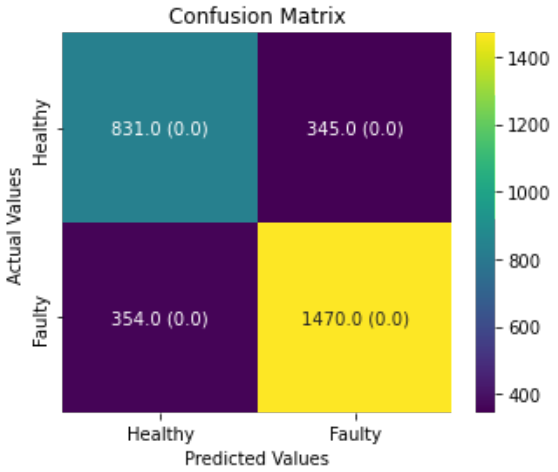


Figure C.38: Confusion matrix of support vector machine model on test set for 60% of faulty systems in dataset.

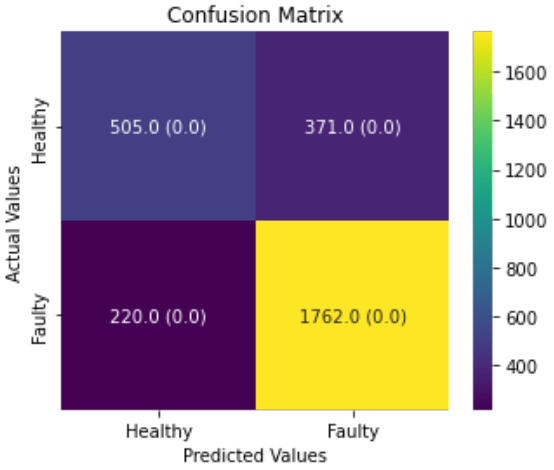


Figure C.39: Confusion matrix of support vector machine model on test set for 70% of faulty systems in dataset.

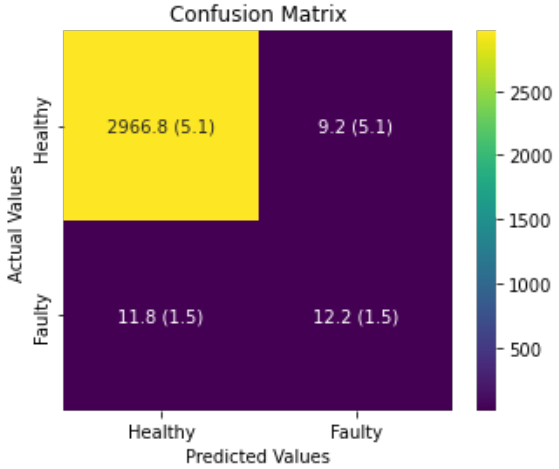


Figure C.40: Confusion matrix of neural network model on test set for 1% of faulty systems in dataset.

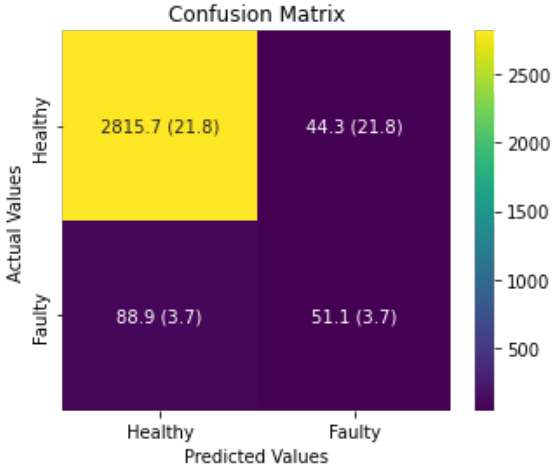


Figure C.41: Confusion matrix of neural network model on test set for 5% of faulty systems in dataset.

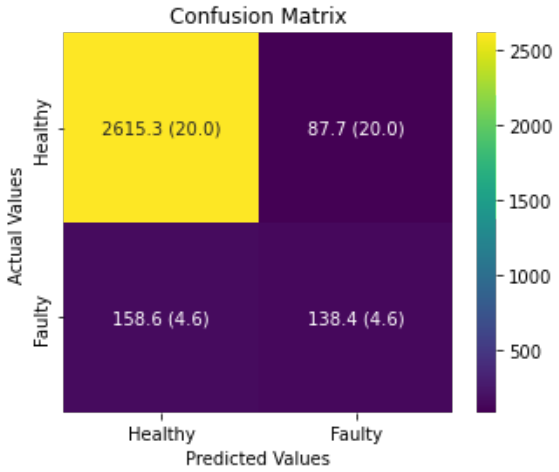


Figure C.42: Confusion matrix of neural network model on test set for 10% of faulty systems in dataset.

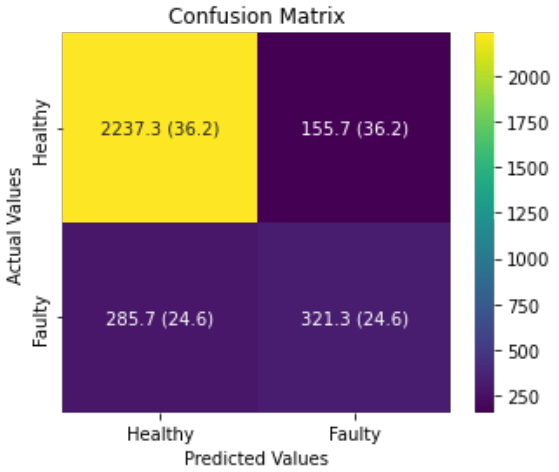


Figure C.43: Confusion matrix of neural network model on test set for 20% of faulty systems in dataset.

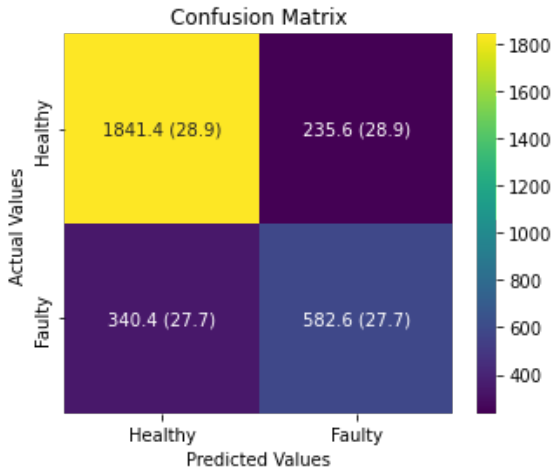


Figure C.44: Confusion matrix of neural network model on test set for 30% of faulty systems in dataset.

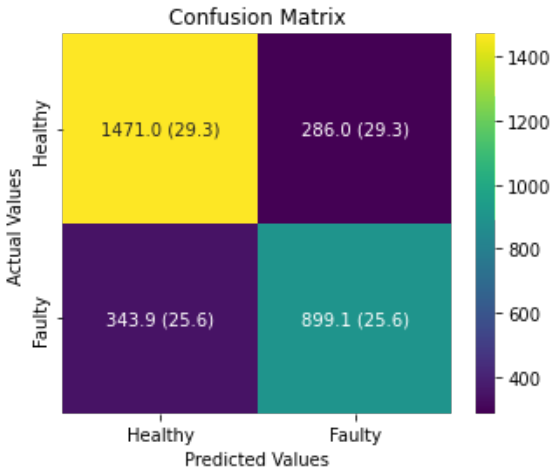


Figure C.45: Confusion matrix of neural network model on test set for 40% of faulty systems in dataset.

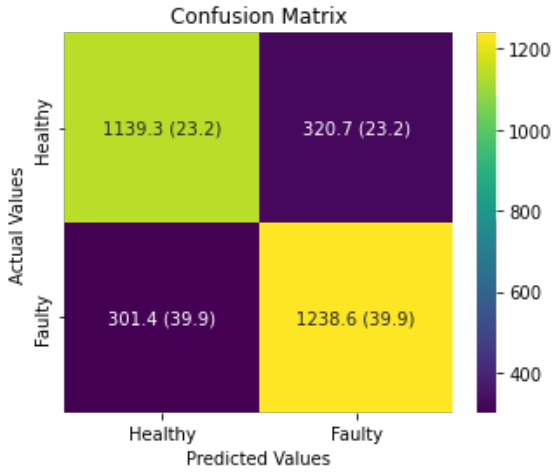


Figure C.46: Confusion matrix of neural network model on test set for 50% of faulty systems in dataset.

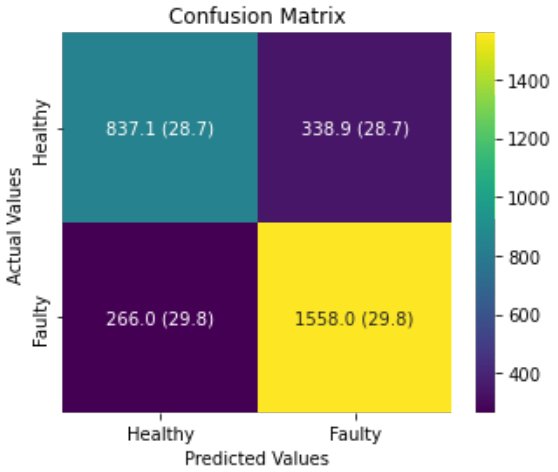


Figure C.47: Confusion matrix of neural network model on test set for 60% of faulty systems in dataset.

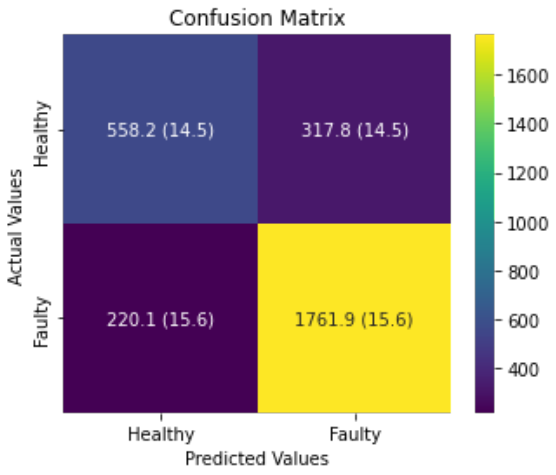


Figure C.48: Confusion matrix of neural network model on test set for 70% of faulty systems in dataset.

C.3. Case 2b

Table C.9: Performance of decision tree model on test set for varying ratio of faulty systems in test set only, maintaining 40% faulty systems in training set.

Metric	1%	2%	5%	10%
Accuracy	0.830 (0.001)	0.831 (0.001)	0.829 (0.001)	0.826 (0.001)
Recall	0.567 (0.000)	0.683 (0.000)	0.724 (0.003)	0.723 (0.003)
Precision	0.033 (0.000)	0.077 (0.001)	0.187 (0.002)	0.331 (0.003)
Specificity	0.833 (0.001)	0.834 (0.001)	0.835 (0.001)	0.838 (0.001)
F1	0.062 (0.000)	0.139 (0.001)	0.297 (0.002)	0.454 (0.003)
AUC	0.700 (0.001)	0.759 (0.001)	0.780 (0.002)	0.780 (0.002)

Table C.10: Performance of random forest model on test set for varying ratio of faulty systems in test set only, maintaining 40% faulty systems in training set.

Metric	1%	2%	5%	10%
Accuracy	0.865 (0.004)	0.863 (0.003)	0.862 (0.004)	0.855 (0.003)
Recall	0.596 (0.033)	0.657 (0.024)	0.667 (0.015)	0.704 (0.012)
Precision	0.043 (0.003)	0.092 (0.003)	0.215 (0.005)	0.379 (0.006)
Specificity	0.867 (0.004)	0.868 (0.003)	0.873 (0.004)	0.872 (0.004)
F1	0.081 (0.006)	0.161 (0.005)	0.326 (0.006)	0.493 (0.005)
AUC	0.732 (0.018)	0.763 (0.011)	0.770 (0.007)	0.788 (0.005)

Table C.11: Performance of support vector machine model on test set for varying ratio of faulty systems in test set only, maintaining 40% faulty systems in training set.

Metric	1%	2%	5%	10%
Accuracy	0.899 (0.000)	0.897 (0.000)	0.887 (0.000)	0.878 (0.000)
Recall	0.633 (0.000)	0.633 (0.000)	0.573 (0.000)	0.638 (0.000)
Precision	0.061 (0.000)	0.117 (0.000)	0.238 (0.000)	0.428 (0.000)
Specificity	0.901 (0.000)	0.902 (0.000)	0.904 (0.000)	0.905 (0.000)
F1	0.111 (0.000)	0.197 (0.000)	0.336 (0.000)	0.512 (0.000)
AUC	0.767 (0.000)	0.768 (0.000)	0.738 (0.000)	0.772 (0.000)

Table C.12: Performance of neural network model on test set for varying ratio of faulty systems in test set only, maintaining 40% faulty systems in training set.

Metric	1%	2%	5%	10%
Accuracy	0.855 (0.000)	0.854 (0.000)	0.853 (0.000)	0.848 (0.000)
Recall	0.667 (0.000)	0.717 (0.000)	0.747 (0.000)	0.754 (0.000)
Precision	0.045 (0.000)	0.092 (0.000)	0.217 (0.000)	0.371 (0.000)
Specificity	0.857 (0.000)	0.857 (0.000)	0.858 (0.000)	0.858 (0.000)
F1	0.084 (0.000)	0.163 (0.000)	0.336 (0.000)	0.497 (0.000)
AUC	0.762 (0.000)	0.787 (0.000)	0.803 (0.000)	0.806 (0.000)

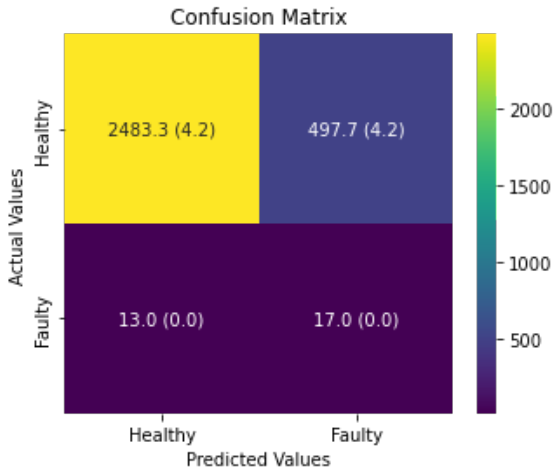


Figure C.49: Confusion matrix of decision tree model on test set for 1% of faulty systems in test set.

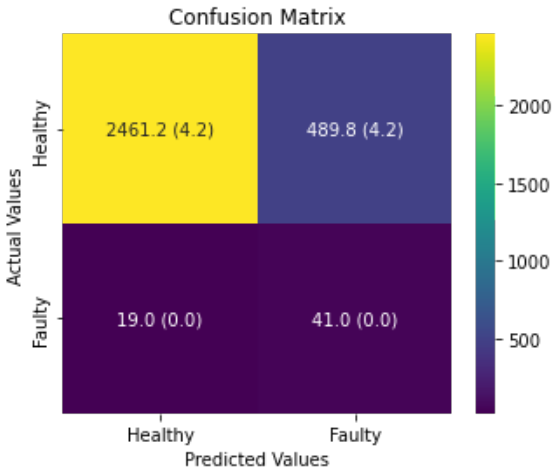


Figure C.50: Confusion matrix of decision tree model on test set for 2% of faulty systems in test set.

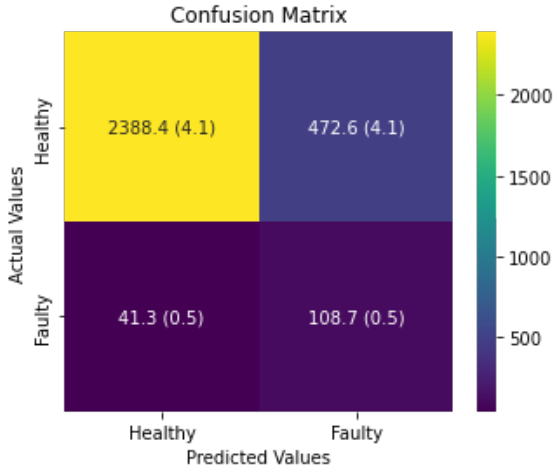


Figure C.51: Confusion matrix of decision tree model on test set for 5% of faulty systems in test set.

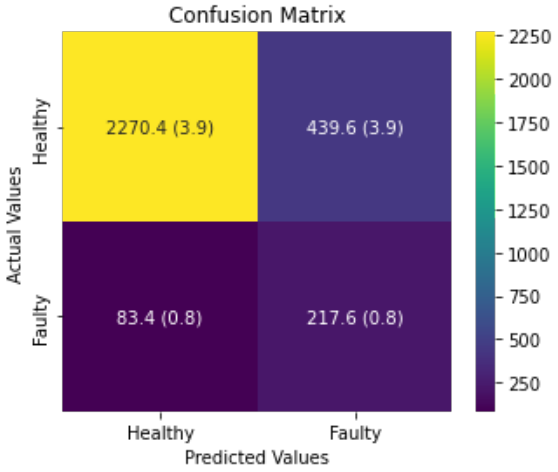


Figure C.52: Confusion matrix of decision tree model on test set for 10% of faulty systems in test set.

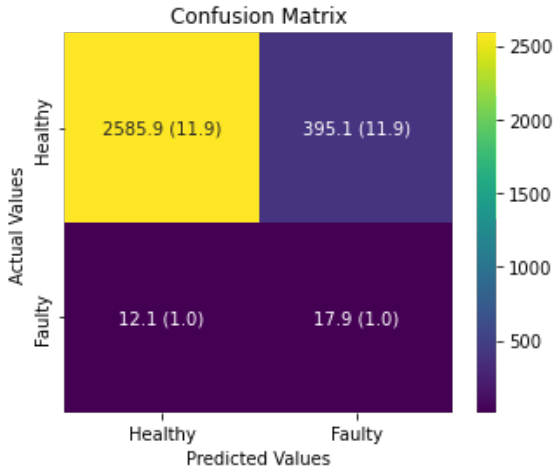


Figure C.53: Confusion matrix of random forest model on test set for 1% of faulty systems in test set.

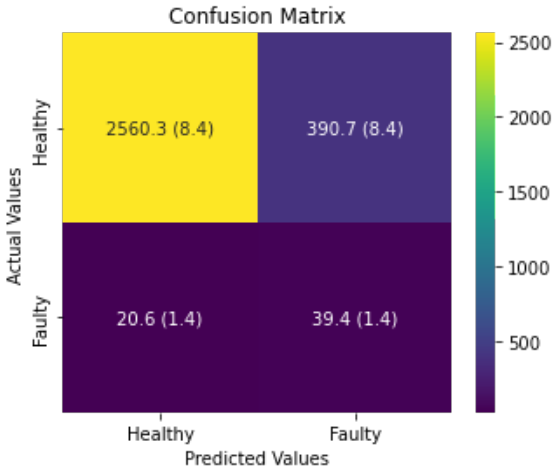


Figure C.54: Confusion matrix of random forest model on test set for 2% of faulty systems in test set.

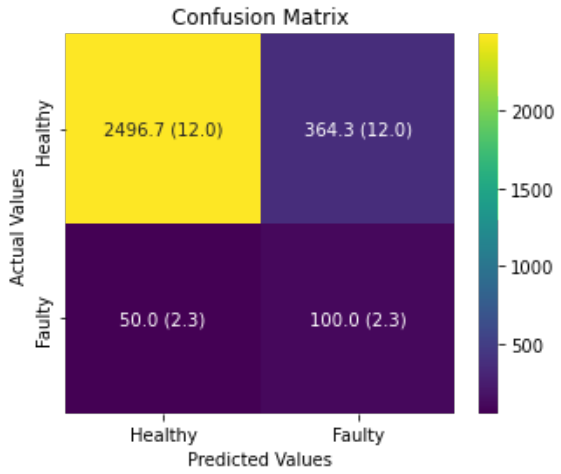


Figure C.55: Confusion matrix of random forest model on test set for 5% of faulty systems in test set.

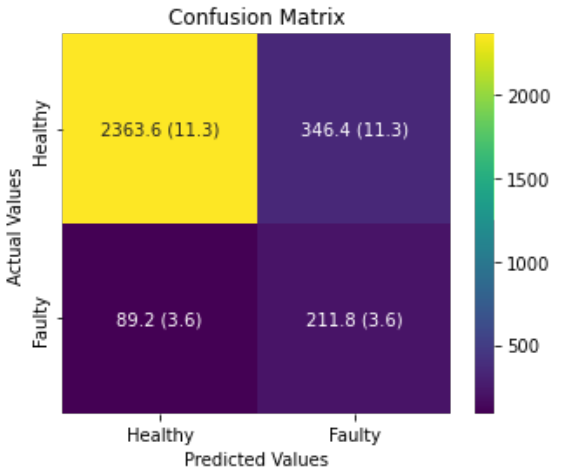


Figure C.56: Confusion matrix of random forest model on test set for 10% of faulty systems in test set.

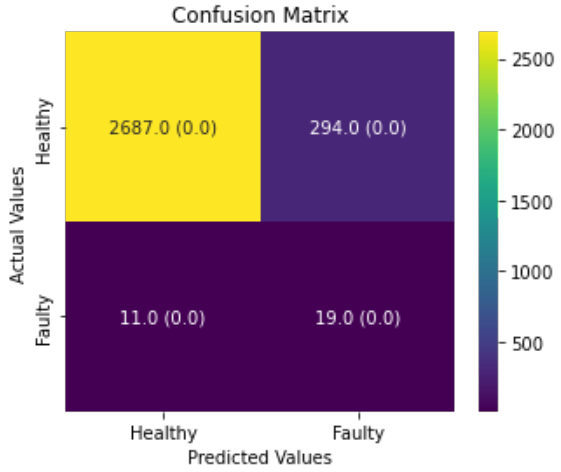


Figure C.57: Confusion matrix of support vector machine model on test set for 1% of faulty systems in test set.

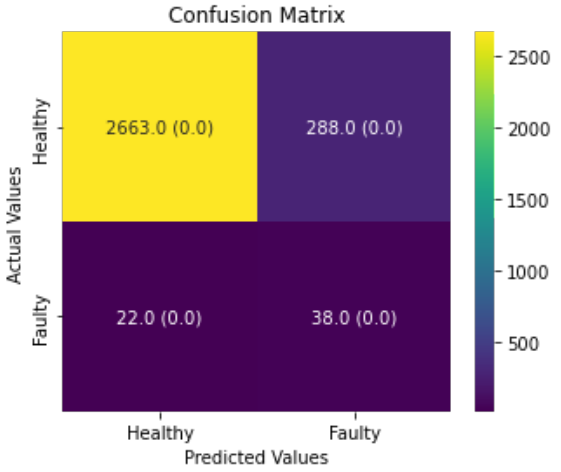


Figure C.58: Confusion matrix of support vector machine model on test set for 2% of faulty systems in test set.

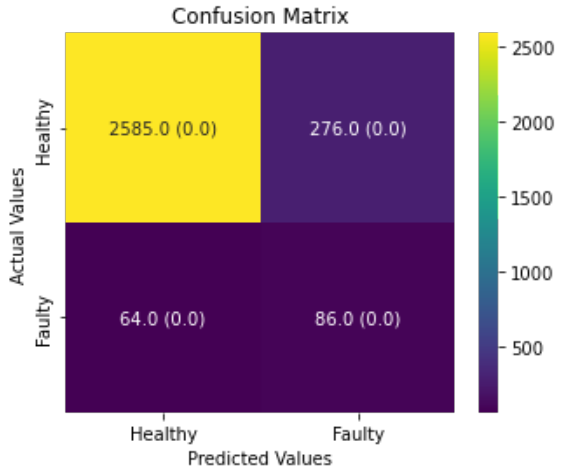


Figure C.59: Confusion matrix of support vector machine model on test set for 5% of faulty systems in test set.

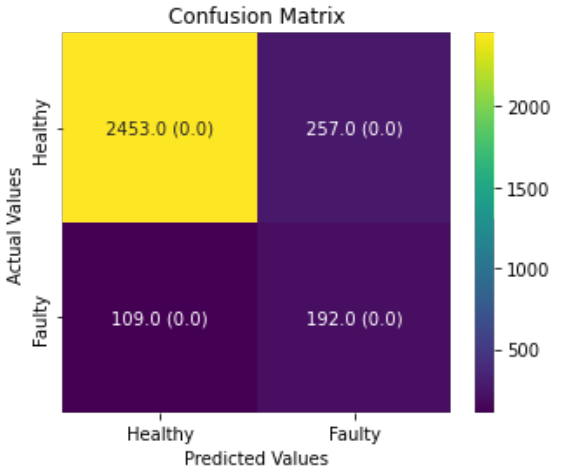


Figure C.60: Confusion matrix of support vector machine model on test set for 10% of faulty systems in test set.

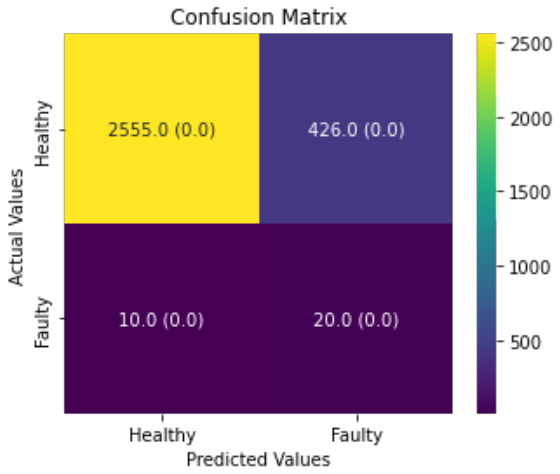


Figure C.61: Confusion matrix of neural network model on test set for 1% of faulty systems in test set.

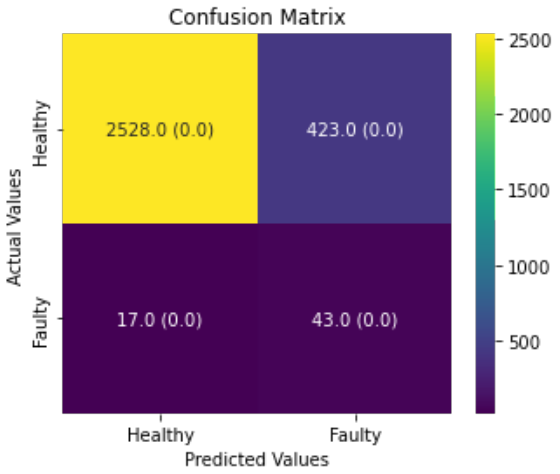


Figure C.62: Confusion matrix of neural network model on test set for 2% of faulty systems in test set.

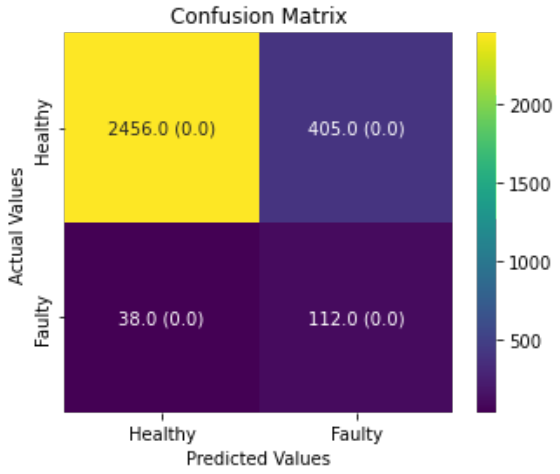


Figure C.63: Confusion matrix of neural network model on test set for 5% of faulty systems in test set.

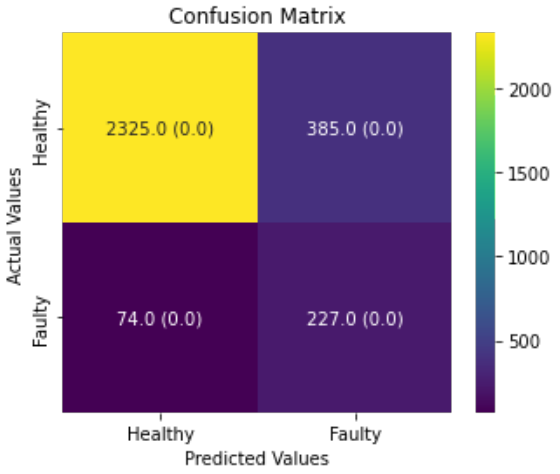


Figure C.64: Confusion matrix of neural network model on test set for 10% of faulty systems in test set.

D

Results Multiclass Classification

D.1. Case 3a

Table D.1: Multiclass classification performance on test set of OVR-classifier using decision tree model.

Class	Precision	Recall	F1
Healthy	0.601 (0.003)	0.508 (0.003)	0.551 (0.002)
Line-Line	0.486 (0.003)	0.583 (0.004)	0.530 (0.003)
Ground	0.423 (0.001)	0.489 (0.001)	0.454 (0.001)
Broken Cell	0.639 (0.005)	0.528 (0.002)	0.578 (0.003)
Broken String	0.915 (0.004)	0.902 (0.003)	0.908 (0.003)
Average	0.613 (0.002)	0.602 (0.001)	0.604 (0.002)

Table D.2: Multiclass classification performance on test set of OVR-classifier using random forest model.

Class	Precision	Recall	F1
Healthy	0.712 (0.008)	0.526 (0.011)	0.605 (0.006)
Line-Line	0.512 (0.011)	0.496 (0.015)	0.504 (0.011)
Ground	0.440 (0.008)	0.654 (0.016)	0.526 (0.010)
Broken Cell	0.716 (0.009)	0.586 (0.008)	0.645 (0.005)
Broken String	0.920 (0.003)	0.952 (0.003)	0.936 (0.002)
Average	0.660 (0.005)	0.643 (0.005)	0.643 (0.005)

Table D.3: Multiclass classification performance on test set of OVR-classifier using support vector machine model.

Class	Precision	Recall	F1
Healthy	0.548 (0.000)	0.493 (0.000)	0.519 (0.000)
Line-Line	0.447 (0.000)	0.440 (0.000)	0.443 (0.000)
Ground	0.400 (0.000)	0.442 (0.000)	0.420 (0.000)
Broken Cell	0.652 (0.000)	0.647 (0.000)	0.650 (0.000)
Broken String	0.940 (0.000)	0.976 (0.000)	0.958 (0.000)
Average	0.598 (0.000)	0.599 (0.000)	0.598 (0.000)

Table D.4: Multiclass classification performance on test set of OVR-classifier using neural network model.

Class	Precision	Recall	F1
Healthy	0.649 (0.026)	0.565 (0.037)	0.603 (0.026)
Line-Line	0.460 (0.019)	0.483 (0.040)	0.470 (0.023)
Ground	0.415 (0.018)	0.503 (0.048)	0.454 (0.023)
Broken Cell	0.723 (0.027)	0.629 (0.025)	0.672 (0.017)
Broken String	0.958 (0.005)	0.970 (0.010)	0.964 (0.004)
Average	0.641 (0.011)	0.630 (0.009)	0.633 (0.010)

Table D.5: Multiclass classification performance on test set of OVO-classifier using decision tree model.

Class	Precision	Recall	F1
Healthy	0.641 (0.003)	0.529 (0.002)	0.580 (0.003)
Line-Line	0.525 (0.007)	0.309 (0.008)	0.389 (0.005)
Ground	0.415 (0.004)	0.742 (0.007)	0.532 (0.003)
Broken Cell	0.622 (0.004)	0.531 (0.002)	0.573 (0.002)
Broken String	0.901 (0.004)	0.915 (0.003)	0.908 (0.002)
Average	0.620 (0.002)	0.605 (0.002)	0.596 (0.002)

Table D.6: Multiclass classification performance on test set of OVO-classifier using random forest model.

Class	Precision	Recall	F1
Healthy	0.710 (0.008)	0.550 (0.008)	0.620 (0.005)
Line-Line	0.516 (0.007)	0.508 (0.013)	0.512 (0.009)
Ground	0.443 (0.008)	0.633 (0.009)	0.521 (0.007)
Broken Cell	0.728 (0.009)	0.605 (0.009)	0.661 (0.008)
Broken String	0.929 (0.004)	0.952 (0.003)	0.940 (0.003)
Average	0.665 (0.004)	0.650 (0.003)	0.651 (0.004)

Table D.7: Multiclass classification performance on test set of OVO-classifier using support vector machine model.

Class	Precision	Recall	F1
Healthy	0.591 (0.000)	0.491 (0.000)	0.536 (0.000)
Line-Line	0.456 (0.000)	0.455 (0.000)	0.455 (0.000)
Ground	0.406 (0.000)	0.496 (0.000)	0.446 (0.000)
Broken Cell	0.688 (0.000)	0.655 (0.000)	0.671 (0.000)
Broken String	0.953 (0.000)	0.978 (0.000)	0.966 (0.000)
Average	0.619 (0.000)	0.615 (0.000)	0.615 (0.000)

Table D.8: Multiclass classification performance on test set of OVO-classifier using neural network model.

Class	Precision	Recall	F1
Healthy	0.688 (0.017)	0.529 (0.023)	0.597 (0.013)
Line-Line	0.474 (0.015)	0.419 (0.029)	0.445 (0.020)
Ground	0.400 (0.009)	0.577 (0.024)	0.472 (0.010)
Broken Cell	0.681 (0.016)	0.642 (0.013)	0.661 (0.007)
Broken String	0.952 (0.004)	0.963 (0.006)	0.958 (0.005)
Average	0.639 (0.005)	0.626 (0.005)	0.627 (0.005)

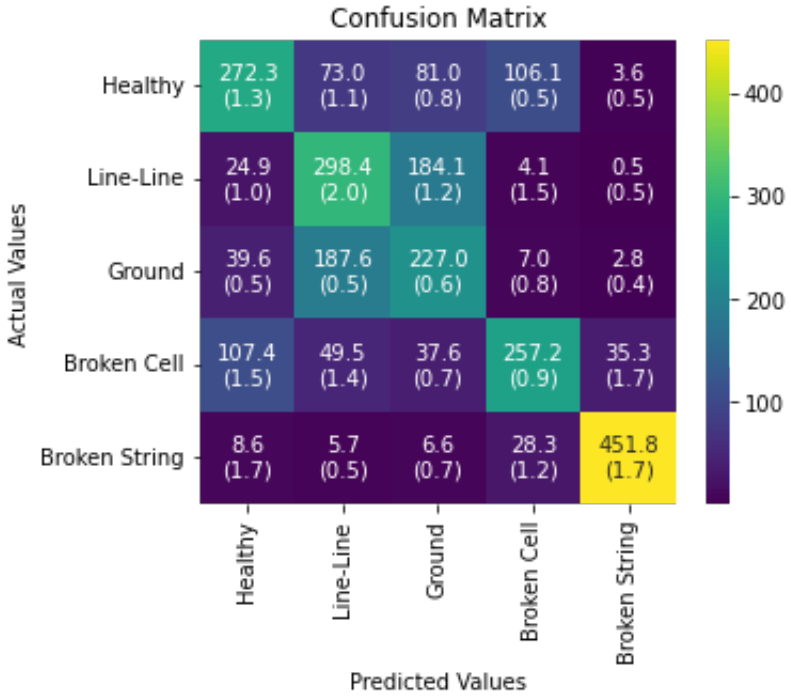


Figure D.1: Confusion matrix of OVR-classifier using decision tree model on test set.

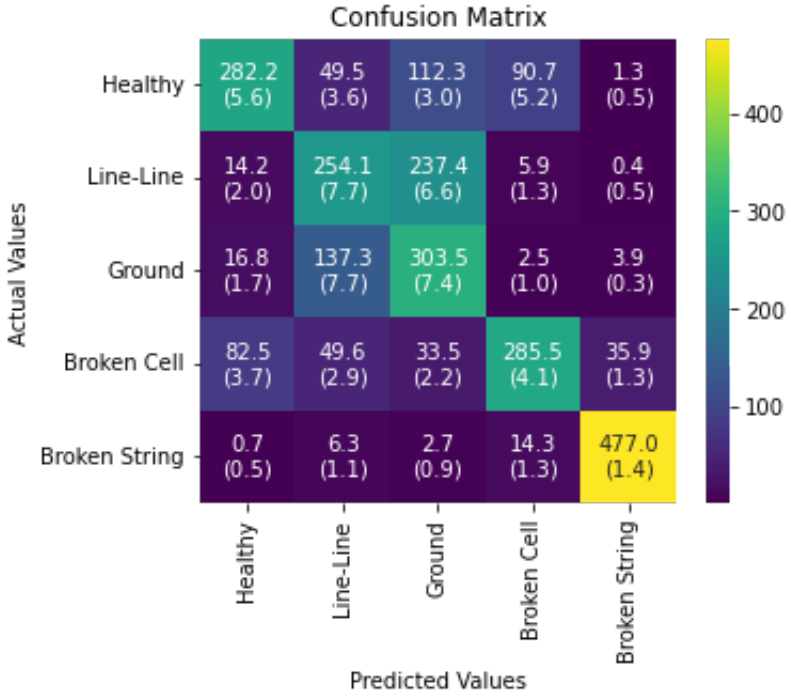


Figure D.2: Confusion matrix of OVR-classifier using random forest model on test set.

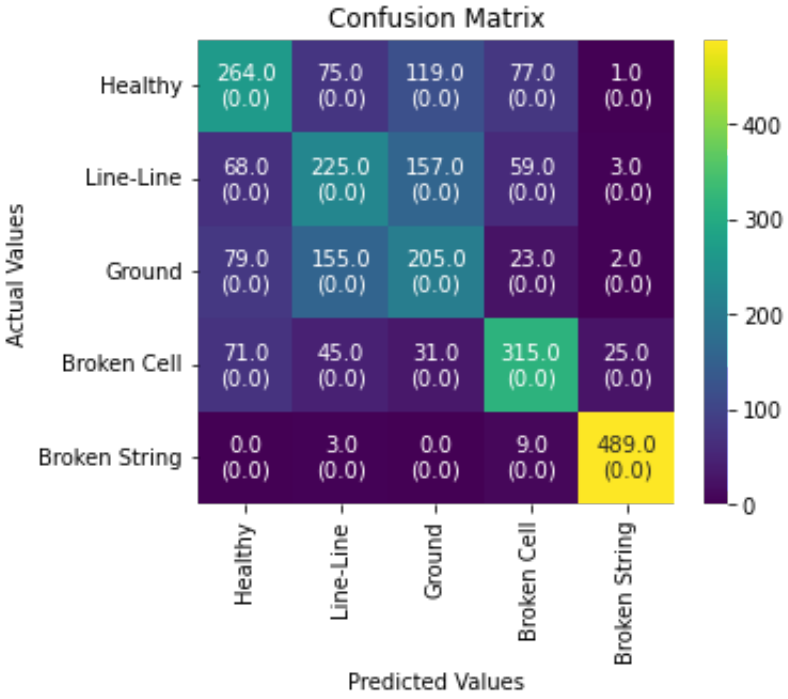


Figure D.3: Confusion matrix of OVR-classifier using support vector machine tree model on test set.

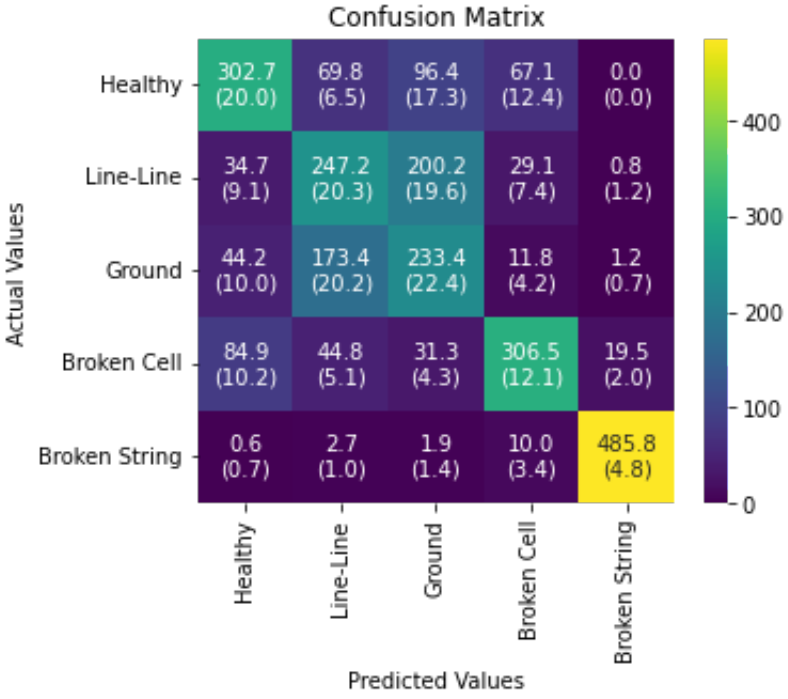


Figure D.4: Confusion matrix of OVR-classifier using neural network model on test set.

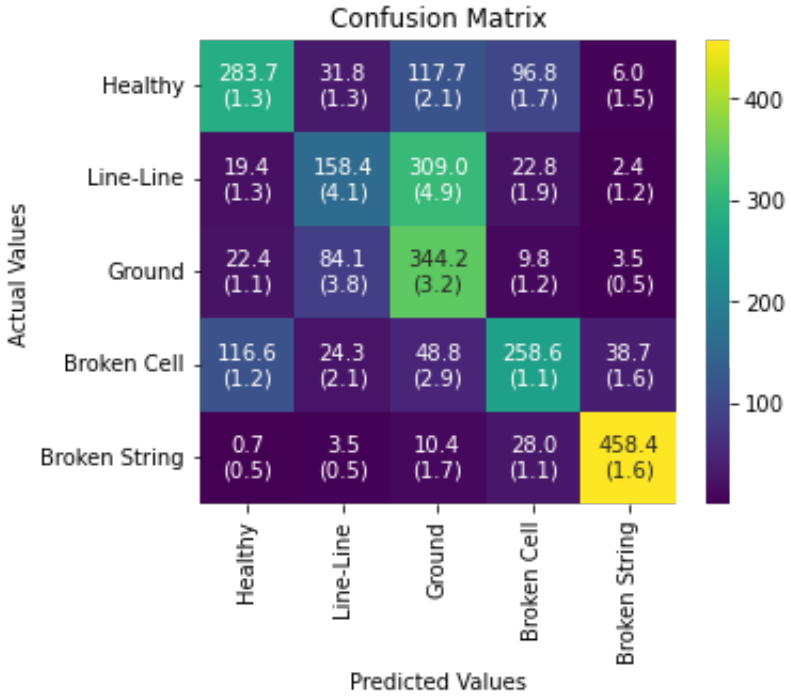


Figure D.5: Confusion matrix of OVO-classifier using decision tree model on test set.

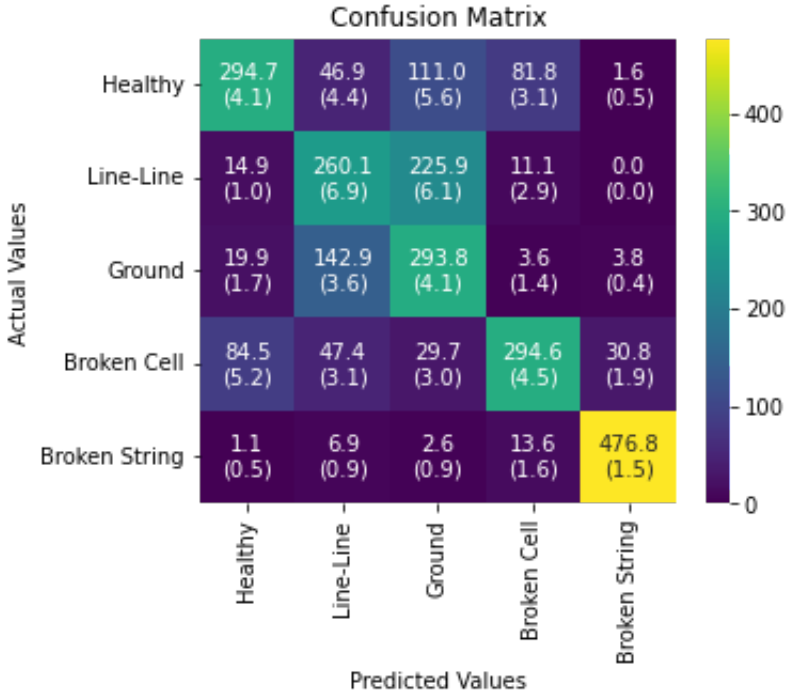


Figure D.6: Confusion matrix of OVO-classifier using random forest model on test set.

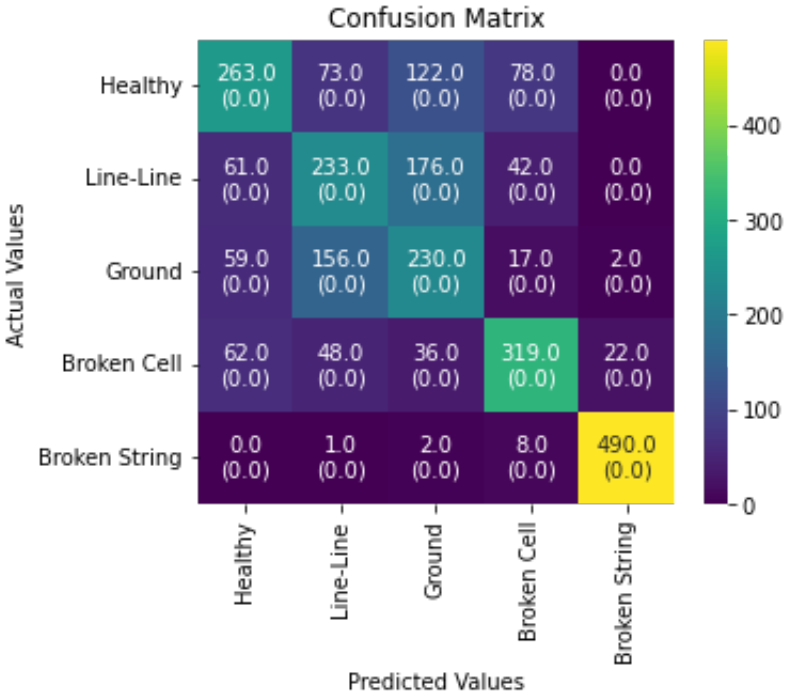


Figure D.7: Confusion matrix of OVO-classifier using support vector machine tree model on test set.

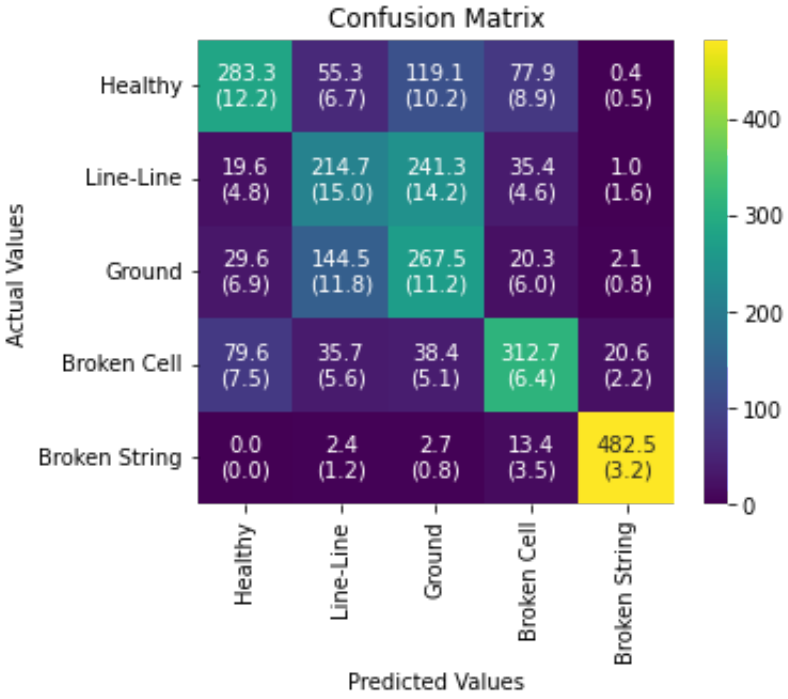


Figure D.8: Confusion matrix of OVO-classifier using neural network model on test set.

D.2. Case 3b

Table D.9: Multiclass classification performance on test set of OVR-classifier using decision tree model with a dataset with 50/50 balance between healthy and faulty systems.

Class	Precision	Recall	F1
Healthy	0.743 (0.000)	0.865 (0.000)	0.800 (0.000)
Line-Line	0.417 (0.002)	0.298 (0.002)	0.347 (0.002)
Ground	0.295 (0.002)	0.322 (0.003)	0.308 (0.002)
Broken Cell	0.689 (0.003)	0.390 (0.004)	0.498 (0.004)
Broken String	0.943 (0.004)	0.908 (0.004)	0.925 (0.003)
Average	0.617 (0.002)	0.557 (0.002)	0.576 (0.002)

Table D.10: Multiclass classification performance on test set of OVR-classifier using random forest model with a dataset with 50/50 balance between healthy and faulty systems.

Class	Precision	Recall	F1
Healthy	0.658 (0.003)	0.966 (0.004)	0.782 (0.002)
Line-Line	0.567 (0.018)	0.234 (0.011)	0.331 (0.010)
Ground	0.287 (0.025)	0.081 (0.014)	0.126 (0.019)
Broken Cell	0.927 (0.011)	0.294 (0.006)	0.447 (0.007)
Broken String	0.932 (0.004)	0.964 (0.003)	0.948 (0.003)
Average	0.674 (0.005)	0.508 (0.004)	0.527 (0.006)

Table D.11: Multiclass classification performance on test set of OVR-classifier using support vector machine model with a dataset with 50/50 balance between healthy and faulty systems.

Class	Precision	Recall	F1
Healthy	0.719 (0.000)	0.896 (0.000)	0.798 (0.000)
Line-Line	0.446 (0.000)	0.268 (0.000)	0.335 (0.000)
Ground	0.282 (0.000)	0.142 (0.000)	0.189 (0.000)
Broken Cell	0.683 (0.000)	0.585 (0.000)	0.630 (0.000)
Broken String	0.960 (0.000)	0.978 (0.000)	0.969 (0.000)
Average	0.618 (0.000)	0.574 (0.000)	0.584 (0.000)

Table D.12: Multiclass classification performance on test set of OVR-classifier using neural network model with a dataset with 50/50 balance between healthy and faulty systems.

Class	Precision	Recall	F1
Healthy	0.763 (0.007)	0.881 (0.006)	0.817 (0.004)
Line-Line	0.437 (0.021)	0.307 (0.027)	0.359 (0.018)
Ground	0.312 (0.007)	0.288 (0.039)	0.298 (0.024)
Broken Cell	0.735 (0.023)	0.539 (0.017)	0.621 (0.012)
Broken String	0.964 (0.006)	0.975 (0.006)	0.969 (0.003)
Average	0.642 (0.009)	0.598 (0.005)	0.613 (0.006)

Table D.13: Multiclass classification performance on test set of OVO-classifier using decision tree model with a dataset with 50/50 balance between healthy and faulty systems.

Class	Precision	Recall	F1
Healthy	0.781 (0.002)	0.819 (0.001)	0.800 (0.001)
Line-Line	0.448 (0.003)	0.344 (0.002)	0.389 (0.003)
Ground	0.306 (0.003)	0.437 (0.004)	0.360 (0.003)
Broken Cell	0.643 (0.004)	0.390 (0.004)	0.486 (0.003)
Broken String	0.916 (0.006)	0.935 (0.002)	0.926 (0.003)
Average	0.619 (0.002)	0.585 (0.001)	0.592 (0.002)

Table D.14: Multiclass classification performance on test set of OVO-classifier using random forest model with a dataset with 50/50 balance between healthy and faulty systems.

Class	Precision	Recall	F1
Healthy	0.699 (0.005)	0.930 (0.006)	0.798 (0.002)
Line-Line	0.535 (0.014)	0.287 (0.010)	0.374 (0.008)
Ground	0.305 (0.013)	0.214 (0.019)	0.252 (0.016)
Broken Cell	0.888 (0.008)	0.341 (0.007)	0.493 (0.007)
Broken String	0.947 (0.003)	0.959 (0.003)	0.953 (0.002)
Average	0.675 (0.005)	0.546 (0.005)	0.574 (0.005)

Table D.15: Multiclass classification performance on test set of OVO-classifier using support vector machine model with a dataset with 50/50 balance between healthy and faulty systems.

Class	Precision	Recall	F1
Healthy	0.722 (0.000)	0.871 (0.000)	0.789 (0.000)
Line-Line	0.466 (0.000)	0.312 (0.000)	0.373 (0.000)
Ground	0.241 (0.000)	0.150 (0.000)	0.185 (0.000)
Broken Cell	0.689 (0.000)	0.587 (0.000)	0.634 (0.000)
Broken String	0.970 (0.000)	0.968 (0.000)	0.969 (0.000)
Average	0.617 (0.000)	0.578 (0.000)	0.590 (0.000)

Table D.16: Multiclass classification performance on test set of OVO-classifier using neural network model with a dataset with 50/50 balance between healthy and faulty systems.

Class	Precision	Recall	F1
Healthy	0.787 (0.011)	0.833 (0.014)	0.809 (0.005)
Line-Line	0.441 (0.023)	0.306 (0.029)	0.360 (0.018)
Ground	0.299 (0.014)	0.380 (0.053)	0.334 (0.030)
Broken Cell	0.679 (0.022)	0.561 (0.018)	0.614 (0.011)
Broken String	0.963 (0.003)	0.962 (0.008)	0.963 (0.004)
Average	0.634 (0.009)	0.608 (0.009)	0.616 (0.007)

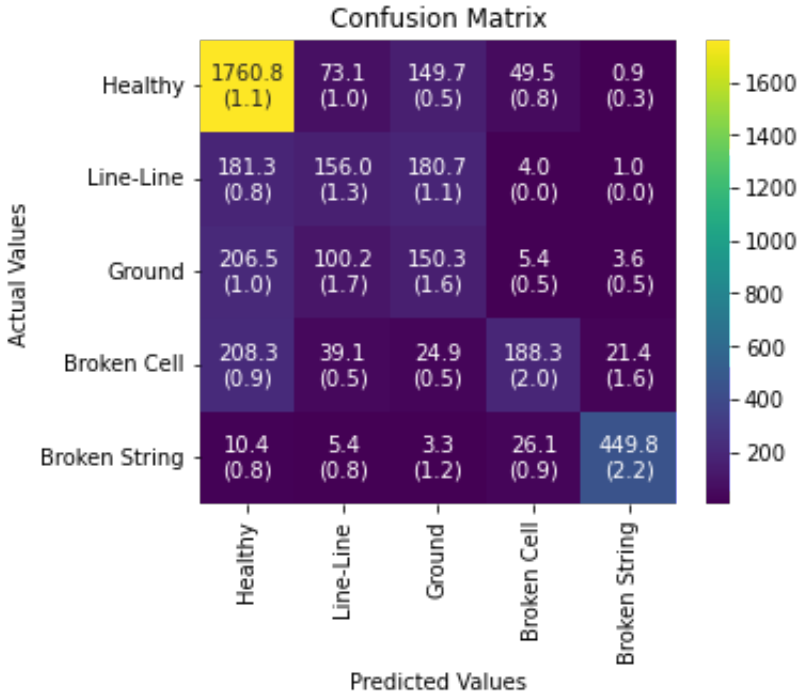


Figure D.9: Confusion matrix of OVR-classifier using decision tree model on test set with a dataset 50/50 balance between healthy and faulty systems.

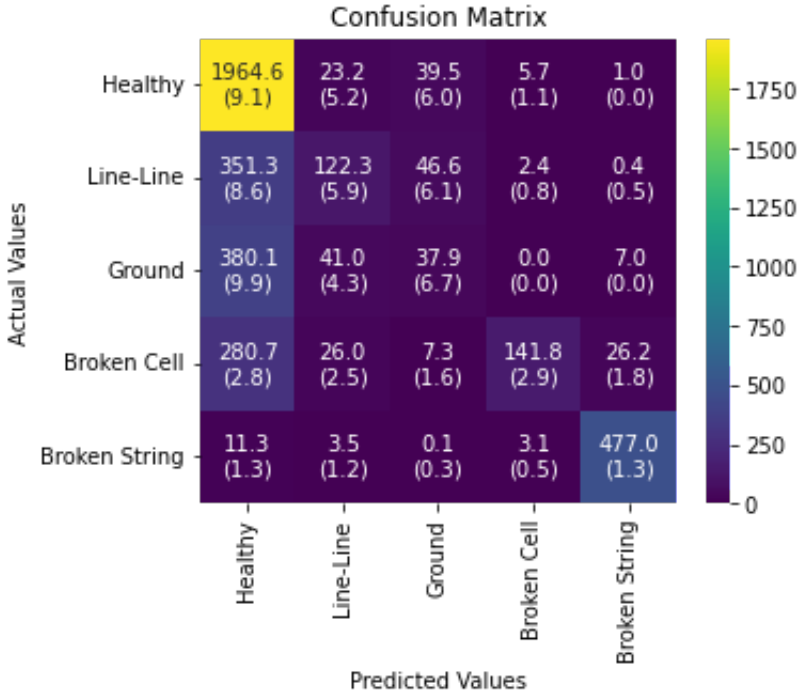


Figure D.10: Confusion matrix of OVR-classifier using random forest model on test set with a dataset with 50/50 balance between healthy and faulty systems.

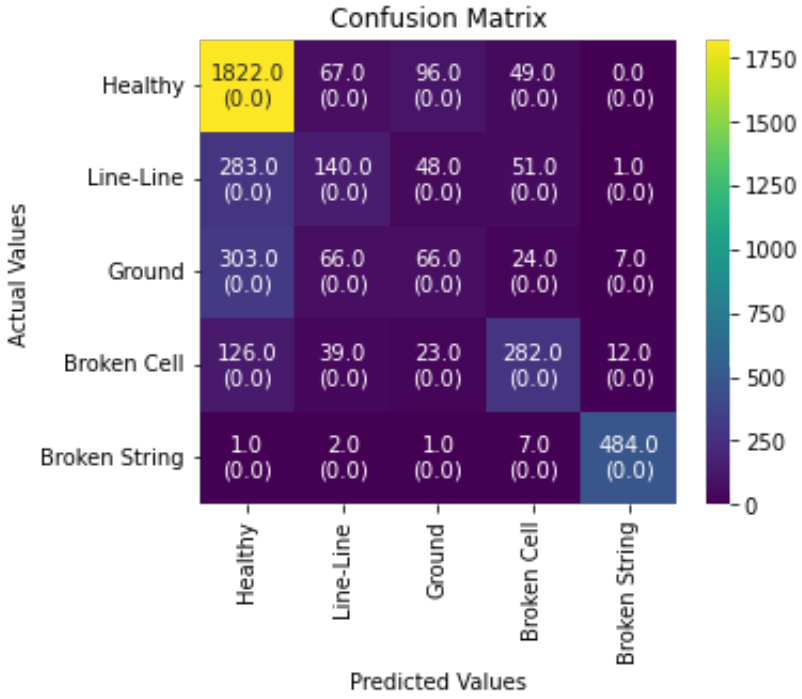


Figure D.11: Confusion matrix of OVR-classifier using support vector machine tree model on test set with a dataset with 50/50 balance between healthy and faulty systems.

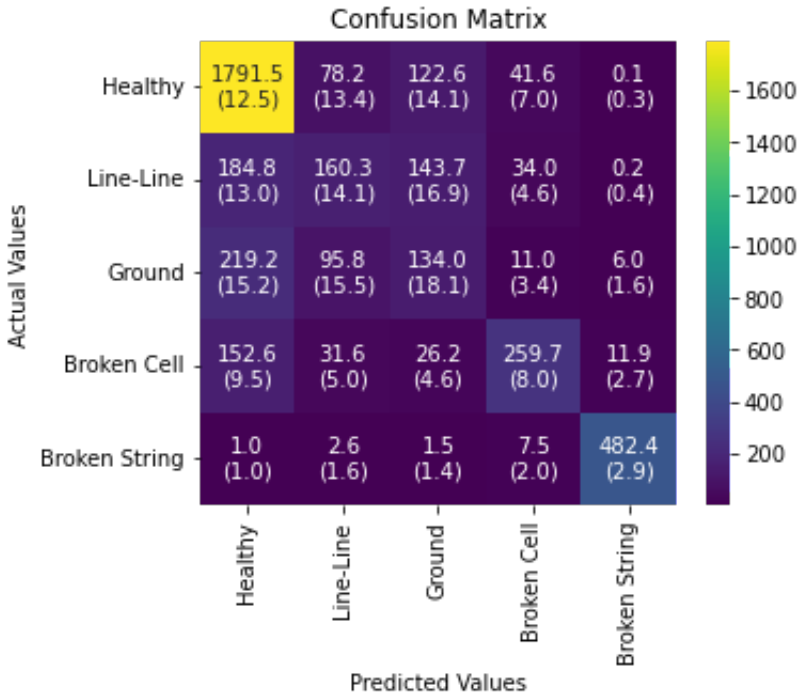


Figure D.12: Confusion matrix of OVR-classifier using neural network model on test set with a dataset with 50/50 balance between healthy and faulty systems.

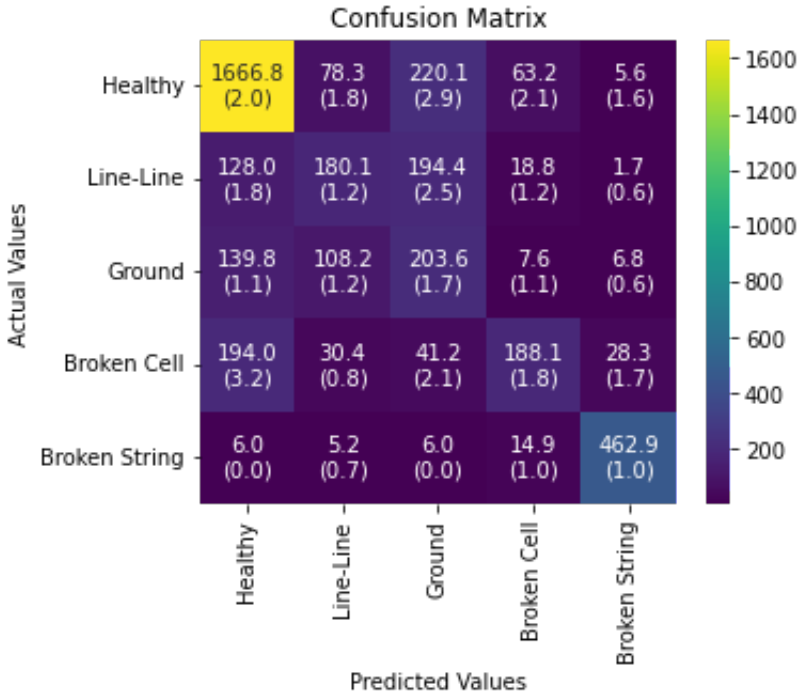


Figure D.13: Confusion matrix of OVO-classifier using decision tree model on test set with a dataset with 50/50 balance between healthy and faulty systems.

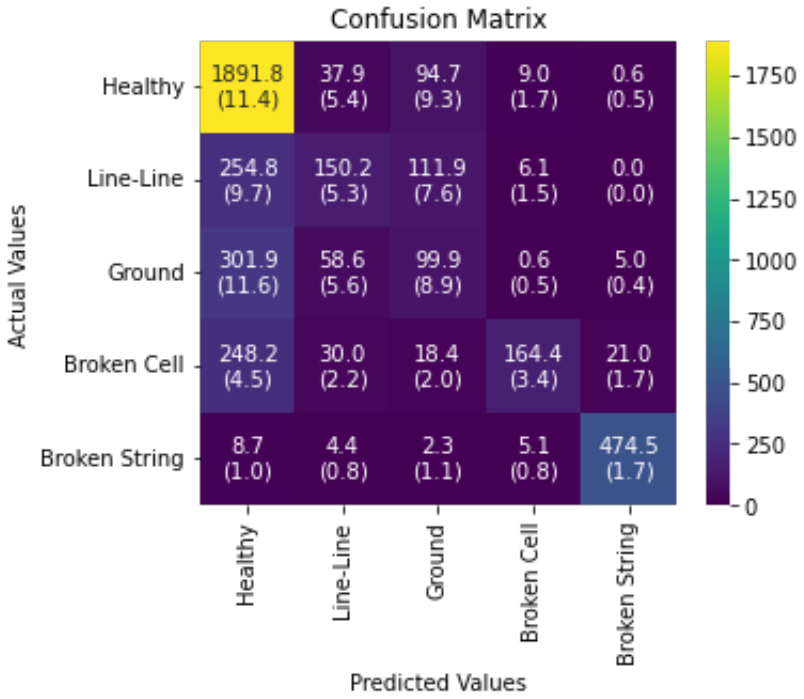


Figure D.14: Confusion matrix of OVO-classifier using random forest model on test set with a dataset with 50/50 balance between healthy and faulty systems.

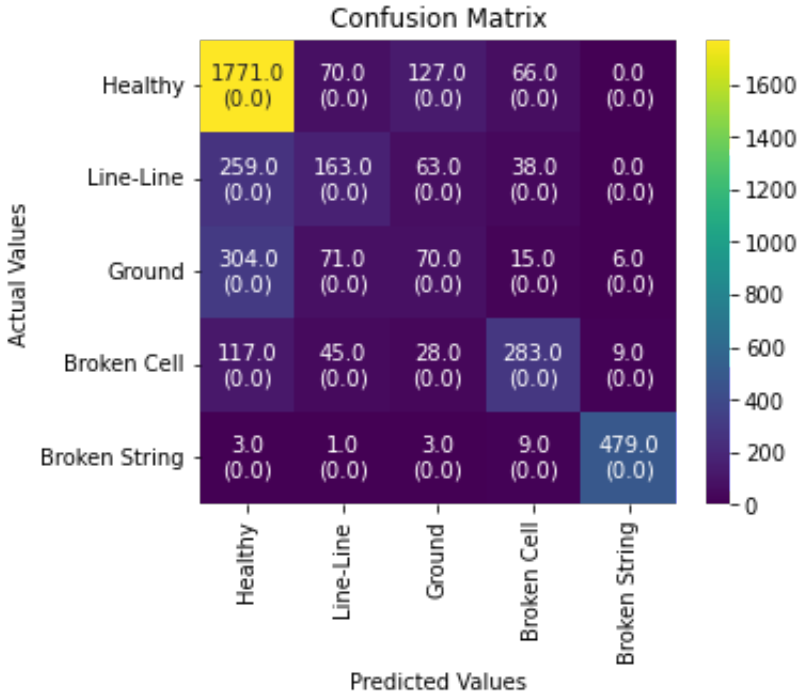


Figure D.15: Confusion matrix of OVO-classifier using support vector machine tree model on test set with a dataset with 50/50 balance between healthy and faulty systems.

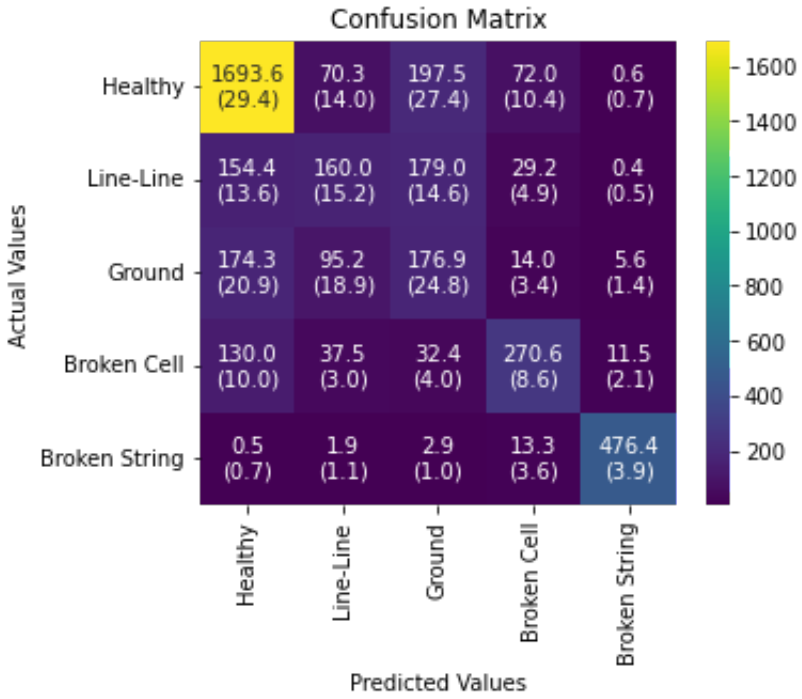


Figure D.16: Confusion matrix of OVO-classifier using neural network model on test set with a dataset with 50/50 balance between healthy and faulty systems.

D.3. Case 3c

Table D.17: Multiclass classification performance on test set of OVR-classifier using decision tree model with a dataset with a combined short-circuit fault class.

Class	Precision	Recall	F1
Healthy	0.642 (0.002)	0.593 (0.003)	0.617 (0.002)
Short-Circuit	0.667 (0.001)	0.837 (0.003)	0.743 (0.001)
Broken Cell	0.691 (0.005)	0.564 (0.003)	0.621 (0.002)
Broken String	0.926 (0.003)	0.931 (0.005)	0.928 (0.003)
Average	0.732 (0.002)	0.731 (0.001)	0.727 (0.001)

Table D.18: Multiclass classification performance on test set of OVR-classifier using random forest model with a dataset with a combined short-circuit fault class.

Class	Precision	Recall	F1
Healthy	0.725 (0.007)	0.563 (0.008)	0.634 (0.007)
Short-Circuit	0.655 (0.002)	0.932 (0.005)	0.769 (0.002)
Broken Cell	0.773 (0.007)	0.574 (0.007)	0.659 (0.007)
Broken String	0.926 (0.003)	0.978 (0.003)	0.951 (0.001)
Average	0.770 (0.003)	0.762 (0.003)	0.753 (0.003)

Table D.19: Multiclass classification performance on test set of OVR-classifier using support vector machine model with a dataset with a combined short-circuit fault class..

Class	Precision	Recall	F1
Healthy	0.637 (0.000)	0.586 (0.000)	0.610 (0.000)
Short-Circuit	0.579 (0.000)	0.664 (0.000)	0.619 (0.000)
Broken Cell	0.739 (0.000)	0.664 (0.000)	0.699 (0.000)
Broken String	0.945 (0.000)	0.976 (0.000)	0.960 (0.000)
Average	0.725 (0.000)	0.723 (0.000)	0.722 (0.000)

Table D.20: Multiclass classification performance on test set of OVR-classifier using neural network model with a dataset with a combined short-circuit fault class.

Class	Precision	Recall	F1
Healthy	0.697 (0.024)	0.601 (0.024)	0.645 (0.016)
Short-Circuit	0.649 (0.009)	0.809 (0.029)	0.720 (0.013)
Broken Cell	0.765 (0.018)	0.663 (0.015)	0.710 (0.013)
Broken String	0.952 (0.005)	0.972 (0.007)	0.962 (0.004)
Average	0.766 (0.010)	0.761 (0.009)	0.759 (0.009)

Table D.21: Multiclass classification performance on test set of OVO-classifier using decision tree model with a dataset with a combined short-circuit fault class.

Class	Precision	Recall	F1
Healthy	0.658 (0.006)	0.537 (0.005)	0.591 (0.003)
Short-Circuit	0.678 (0.003)	0.774 (0.004)	0.723 (0.003)
Broken Cell	0.620 (0.004)	0.603 (0.006)	0.612 (0.004)
Broken String	0.888 (0.003)	0.948 (0.002)	0.917 (0.002)
Average	0.711 (0.002)	0.716 (0.002)	0.711 (0.002)

Table D.22: Multiclass classification performance on test set of OVO-classifier using random forest model with a dataset with a combined short-circuit fault class.

Class	Precision	Recall	F1
Healthy	0.723 (0.007)	0.565 (0.007)	0.635 (0.007)
Short-Circuit	0.653 (0.002)	0.928 (0.003)	0.767 (0.002)
Broken Cell	0.774 (0.011)	0.572 (0.005)	0.658 (0.006)
Broken String	0.925 (0.003)	0.978 (0.001)	0.950 (0.002)
Average	0.769 (0.004)	0.761 (0.003)	0.752 (0.003)

Table D.23: Multiclass classification performance on test set of OVO-classifier using support vector machine model with a dataset with a combined short-circuit fault class.

Class	Precision	Recall	F1
Healthy	0.646 (0.000)	0.596 (0.000)	0.620 (0.000)
Short-Circuit	0.598 (0.000)	0.698 (0.000)	0.644 (0.000)
Broken Cell	0.735 (0.000)	0.658 (0.000)	0.694 (0.000)
Broken String	0.952 (0.000)	0.967 (0.000)	0.959 (0.000)
Average	0.733 (0.000)	0.730 (0.000)	0.730 (0.000)

Table D.24: Multiclass classification performance on test set of OVO-classifier using neural network model with a dataset with a combined short-circuit fault class.

Class	Precision	Recall	F1
Healthy	0.691 (0.023)	0.612 (0.027)	0.649 (0.018)
Short-Circuit	0.659 (0.015)	0.770 (0.026)	0.710 (0.017)
Broken Cell	0.719 (0.022)	0.665 (0.014)	0.691 (0.008)
Broken String	0.952 (0.006)	0.970 (0.008)	0.961 (0.005)
Average	0.755 (0.009)	0.754 (0.009)	0.753 (0.009)

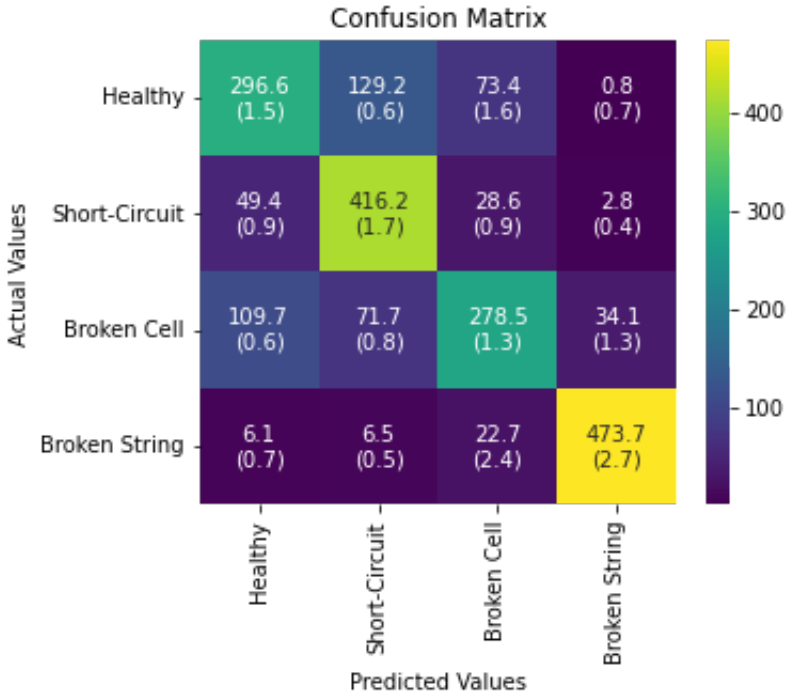


Figure D.17: Confusion matrix of OVR-classifier using decision tree model on test set with a dataset short-circuit fault class.

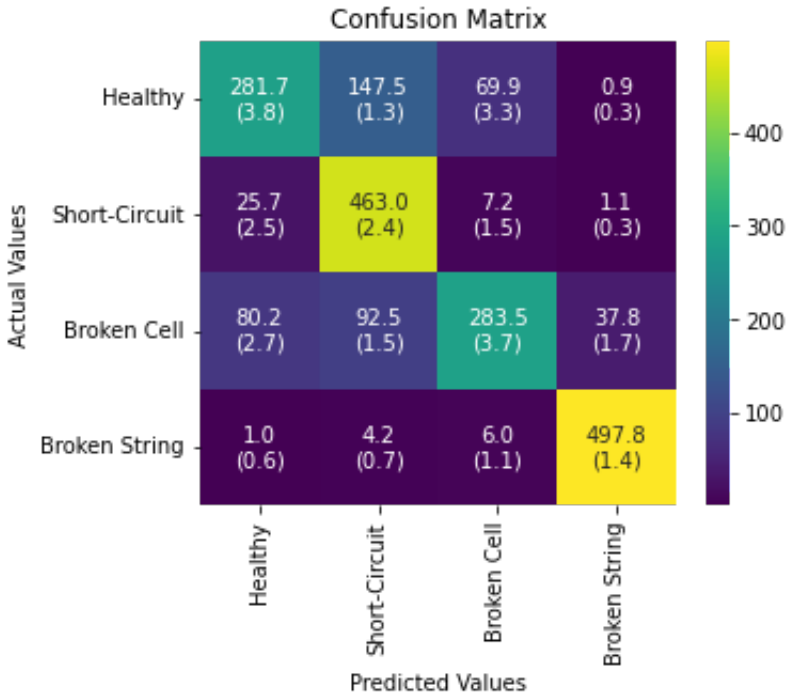


Figure D.18: Confusion matrix of OVR-classifier using random forest model on test set with a dataset with short-circuit fault class.

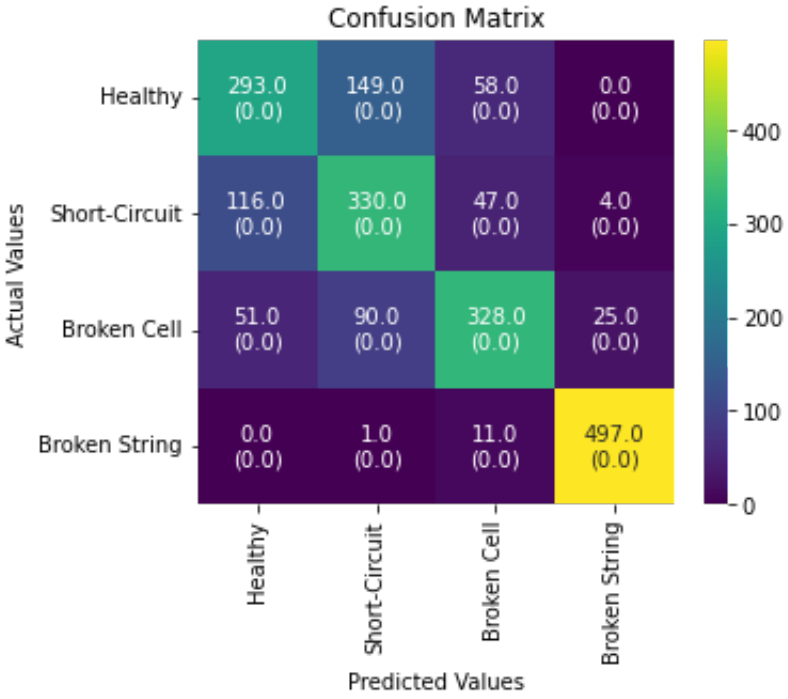


Figure D.19: Confusion matrix of OVR-classifier using support vector machine tree model on test set with short-circuit fault class.

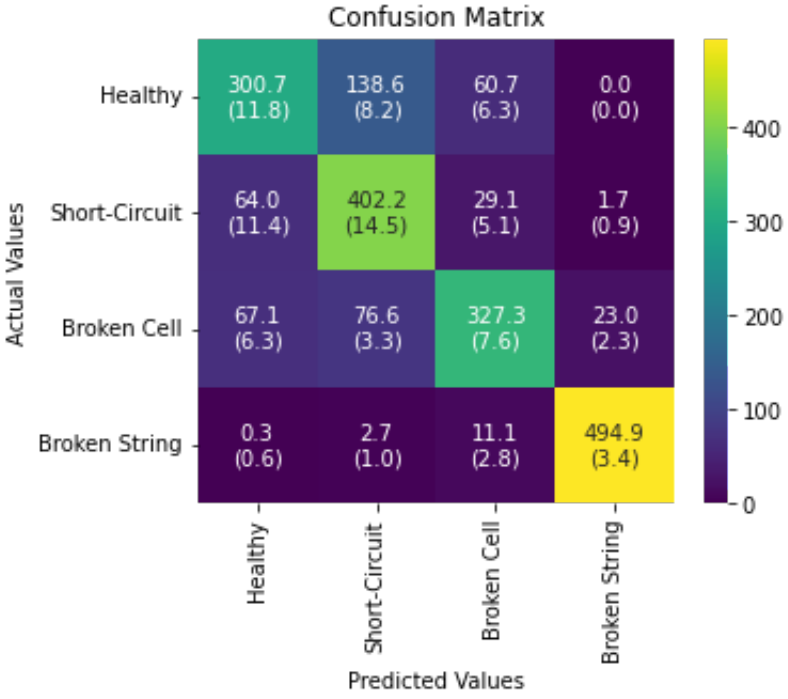


Figure D.20: Confusion matrix of OVR-classifier using neural network model on test set with a dataset with short-circuit fault class.

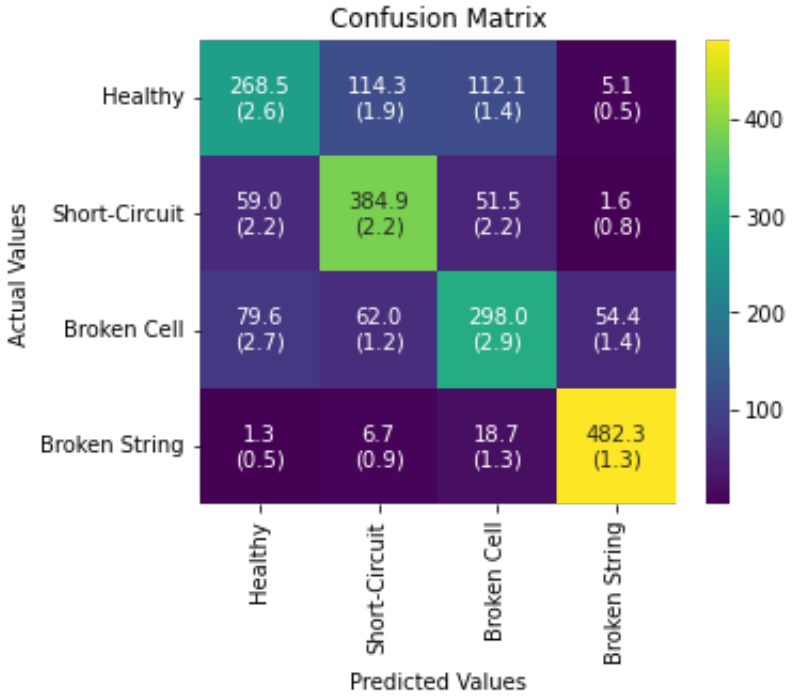


Figure D.21: Confusion matrix of OVO-classifier using decision tree model on test set with a dataset with short-circuit fault class.

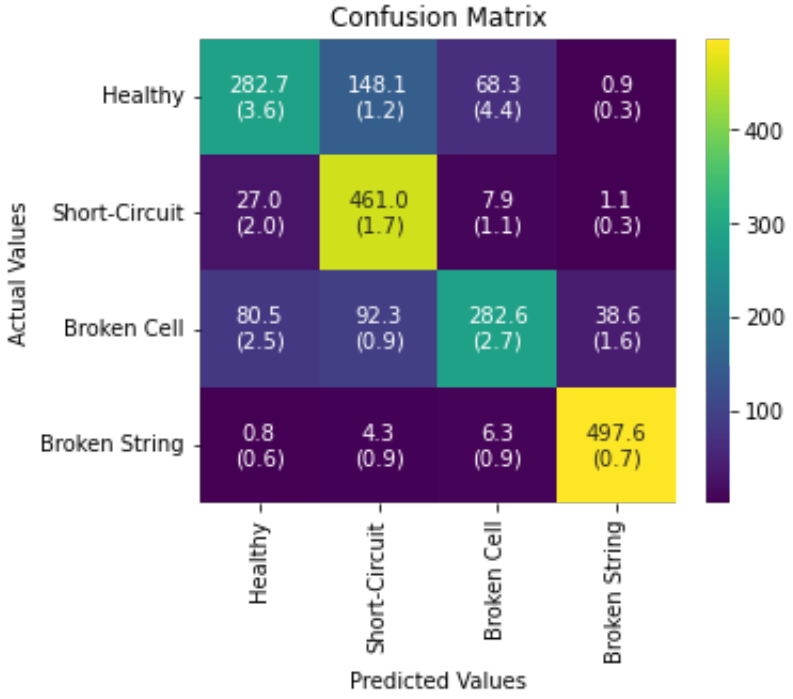


Figure D.22: Confusion matrix of OVO-classifier using random forest model on test set with a dataset with short-circuit fault class.

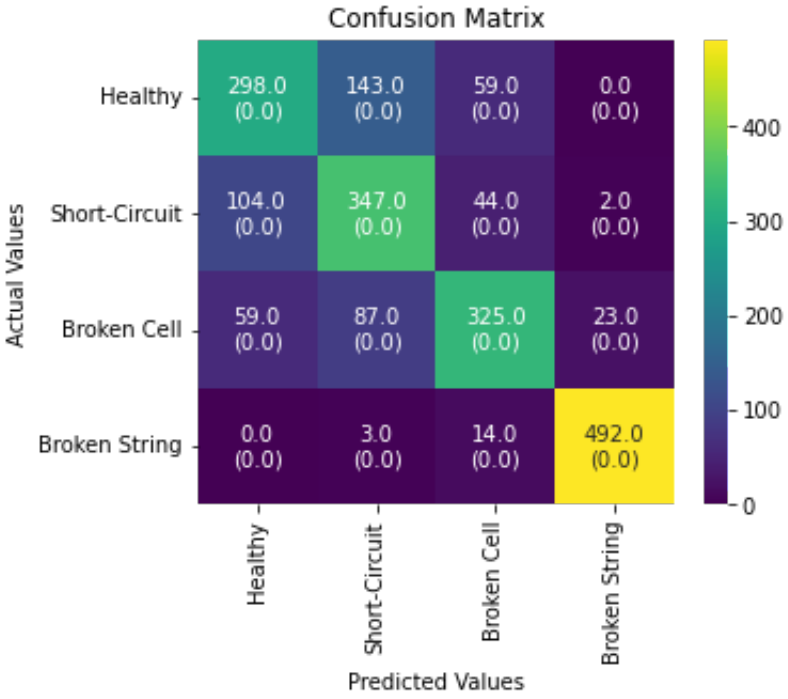


Figure D.23: Confusion matrix of OVO-classifier using support vector machine tree model on test set with short-circuit fault class.

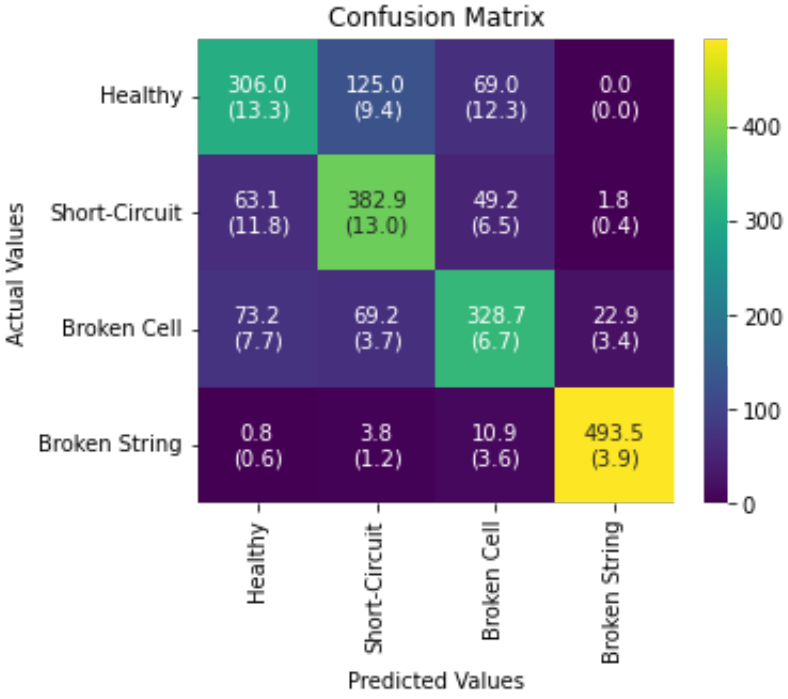


Figure D.24: Confusion matrix of OVO-classifier using neural network model on test set with a dataset with short-circuit fault class.

D.4. Case 3d

Table D.25: Multiclass classification performance on test set of OVR-classifier using decision tree model with a dataset with short-circuit fault class with $R_{fault,max} = 100\,\Omega$.

Class	Precision	Recall	F1
Healthy	0.631 (0.002)	0.613 (0.004)	0.622 (0.003)
Short-Circuit	0.665 (0.003)	0.801 (0.002)	0.727 (0.002)
Broken Cell	0.696 (0.005)	0.556 (0.004)	0.618 (0.004)
Broken String	0.921 (0.003)	0.944 (0.003)	0.932 (0.003)
Average	0.728 (0.002)	0.728 (0.002)	0.725 (0.002)

Table D.26: Multiclass classification performance on test set of OVR-classifier using random forest model with a dataset with short-circuit fault class with $R_{fault,max} = 100\,\Omega$.

Class	Precision	Recall	F1
Healthy	0.720 (0.006)	0.584 (0.009)	0.645 (0.006)
Short-Circuit	0.672 (0.003)	0.910 (0.003)	0.773 (0.003)
Broken Cell	0.760 (0.008)	0.592 (0.009)	0.666 (0.006)
Broken String	0.928 (0.002)	0.976 (0.003)	0.952 (0.002)
Average	0.770 (0.003)	0.766 (0.003)	0.759 (0.003)

Table D.27: Multiclass classification performance on test set of OVR-classifier using support vector machine model with a dataset with short-circuit fault class with $R_{fault,max} = 100\,\Omega$.

Class	Precision	Recall	F1
Healthy	0.657 (0.000)	0.658 (0.000)	0.657 (0.000)
Short-Circuit	0.600 (0.000)	0.668 (0.000)	0.632 (0.000)
Broken Cell	0.737 (0.000)	0.642 (0.000)	0.686 (0.000)
Broken String	0.955 (0.000)	0.969 (0.000)	0.962 (0.000)
Average	0.737 (0.000)	0.734 (0.000)	0.735 (0.000)

Table D.28: Multiclass classification performance on test set of OVR-classifier using neural network model with a dataset with short-circuit fault class with $R_{fault,max} = 100\,\Omega$.

Class	Precision	Recall	F1
Healthy	0.679 (0.012)	0.639 (0.022)	0.658 (0.009)
Short-Circuit	0.655 (0.011)	0.766 (0.023)	0.706 (0.012)
Broken Cell	0.762 (0.017)	0.662 (0.014)	0.708 (0.009)
Broken String	0.955 (0.004)	0.973 (0.006)	0.964 (0.002)
Average	0.763 (0.004)	0.760 (0.004)	0.759 (0.004)

Table D.29: Multiclass classification performance on test set of OVO-classifier using decision tree model with a dataset with short-circuit fault class with $R_{fault,max} = 100\,\Omega$.

Class	Precision	Recall	F1
Healthy	0.661 (0.003)	0.548 (0.003)	0.599 (0.003)
Short-Circuit	0.672 (0.003)	0.754 (0.005)	0.711 (0.004)
Broken Cell	0.596 (0.004)	0.583 (0.007)	0.590 (0.004)
Broken String	0.888 (0.006)	0.948 (0.005)	0.917 (0.004)
Average	0.704 (0.002)	0.709 (0.002)	0.704 (0.002)

Table D.30: Multiclass classification performance on test set of OVO-classifier using random forest model with a dataset with short-circuit fault class with $R_{fault,max} = 100\,\Omega$.

Class	Precision	Recall	F1
Healthy	0.719 (0.006)	0.583 (0.004)	0.644 (0.002)
Short-Circuit	0.671 (0.003)	0.907 (0.004)	0.772 (0.003)
Broken Cell	0.756 (0.006)	0.592 (0.006)	0.664 (0.004)
Broken String	0.929 (0.004)	0.977 (0.003)	0.952 (0.003)
Average	0.769 (0.002)	0.765 (0.002)	0.758 (0.002)

Table D.31: Multiclass classification performance on test set of OVO-classifier using support vector machine model with a dataset with short-circuit fault class with $R_{fault,max} = 100\,\Omega$.

Class	Precision	Recall	F1
Healthy	0.646 (0.000)	0.630 (0.000)	0.641 (0.000)
Short-Circuit	0.620 (0.000)	0.656 (0.000)	0.637 (0.000)
Broken Cell	0.710 (0.000)	0.660 (0.000)	0.688 (0.000)
Broken String	0.954 (0.000)	0.969 (0.000)	0.961 (0.000)
Average	0.733 (0.000)	0.731 (0.000)	0.732 (0.000)

Table D.32: Multiclass classification performance on test set of OVO-classifier using neural network model with a dataset with short-circuit fault class with $R_{fault,max} = 100\,\Omega$.

Class	Precision	Recall	F1
Healthy	0.686 (0.011)	0.625 (0.030)	0.654 (0.018)
Short-Circuit	0.659 (0.018)	0.738 (0.029)	0.696 (0.022)
Broken Cell	0.692 (0.029)	0.657 (0.014)	0.673 (0.012)
Broken String	0.953 (0.007)	0.969 (0.013)	0.961 (0.007)
Average	0.747 (0.013)	0.747 (0.013)	0.746 (0.012)

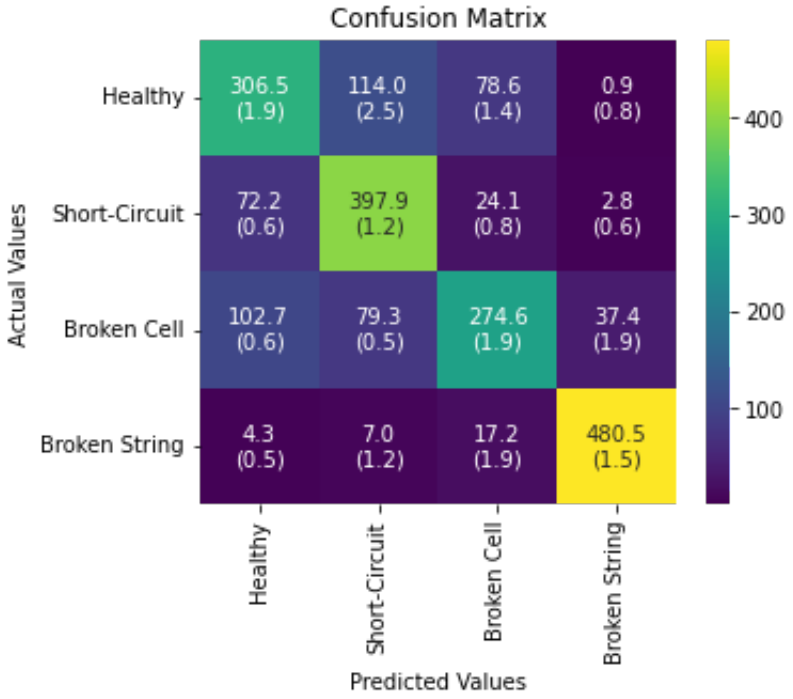


Figure D.25: Confusion matrix of OVR-classifier using decision tree model on test set with a dataset with short-circuit fault class with $R_{fault,max} = 100\,\Omega$.

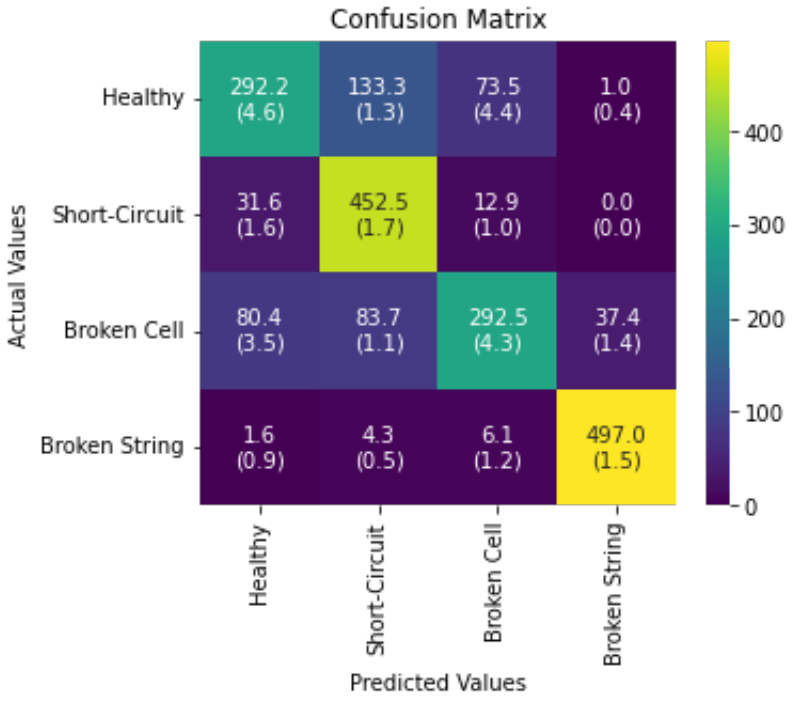


Figure D.26: Confusion matrix of OVR-classifier using random forest model on test set with a dataset with short-circuit fault class with $R_{fault,max} = 100\,\Omega$.

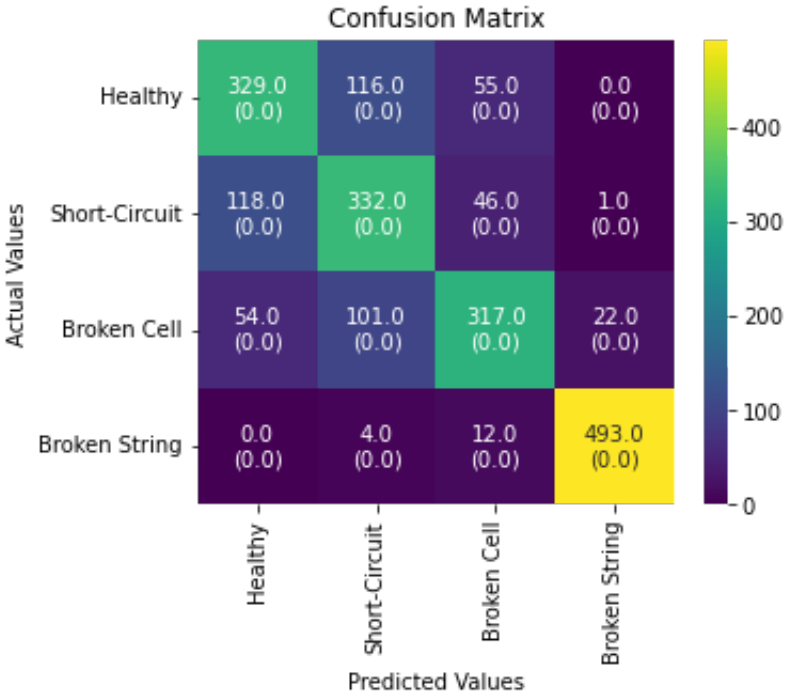


Figure D.27: Confusion matrix of OVR-classifier using support vector machine tree model on test set with a dataset with short-circuit fault class with $R_{fault,max} = 100\,\Omega$.

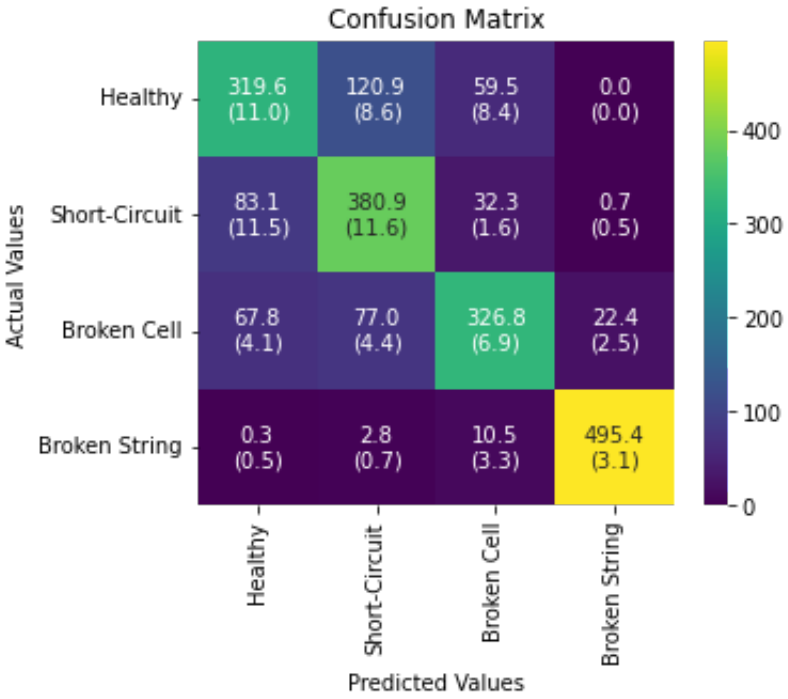


Figure D.28: Confusion matrix of OVR-classifier using neural network model on test set with a dataset with short-circuit fault class with $R_{fault,max} = 100\,\Omega$.

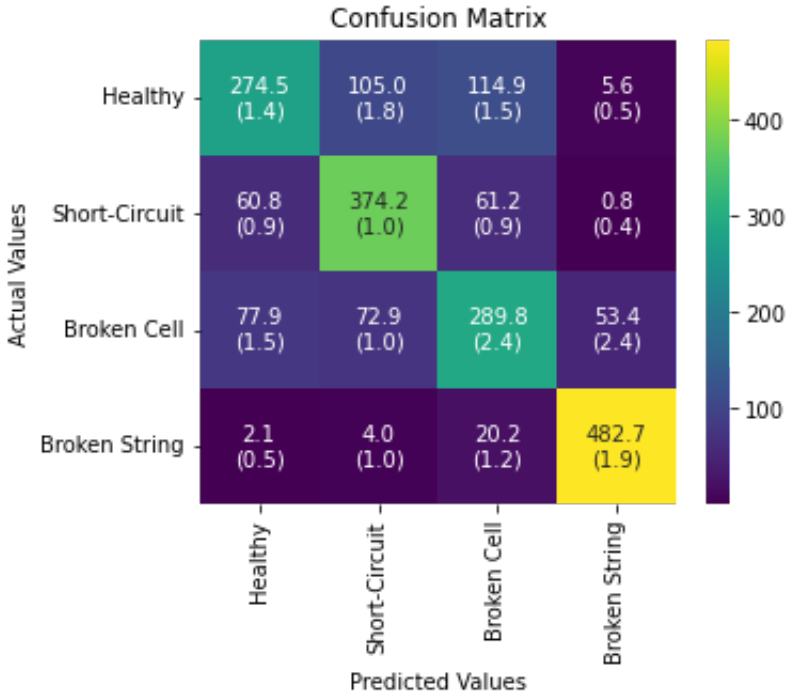


Figure D.29: Confusion matrix of OVO-classifier using decision tree model on test set with a dataset with short-circuit fault class with $R_{fault,max} = 100\,\Omega$.

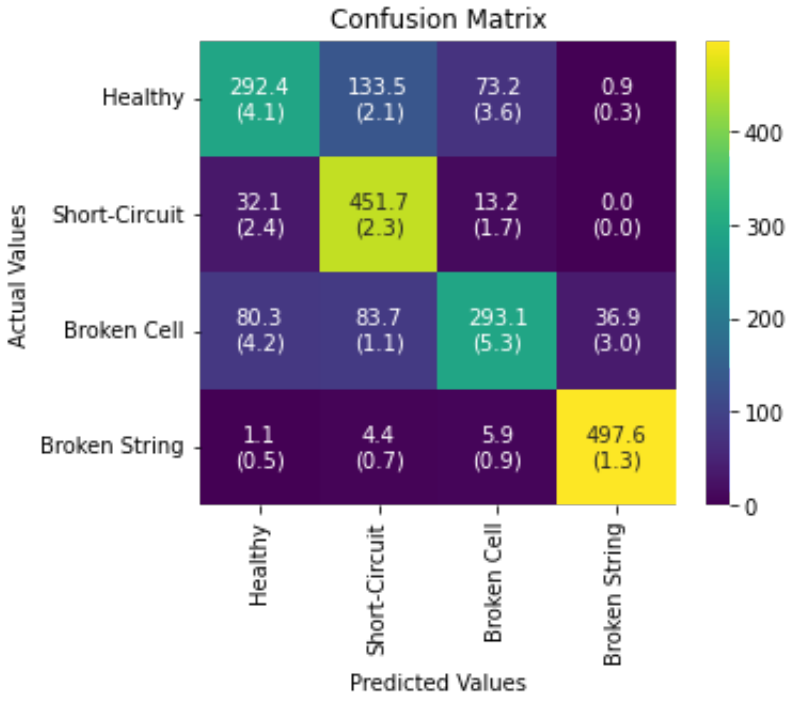


Figure D.30: Confusion matrix of OVO-classifier using random forest model on test set with a dataset with short-circuit fault class with $R_{fault,max} = 100\,\Omega$.

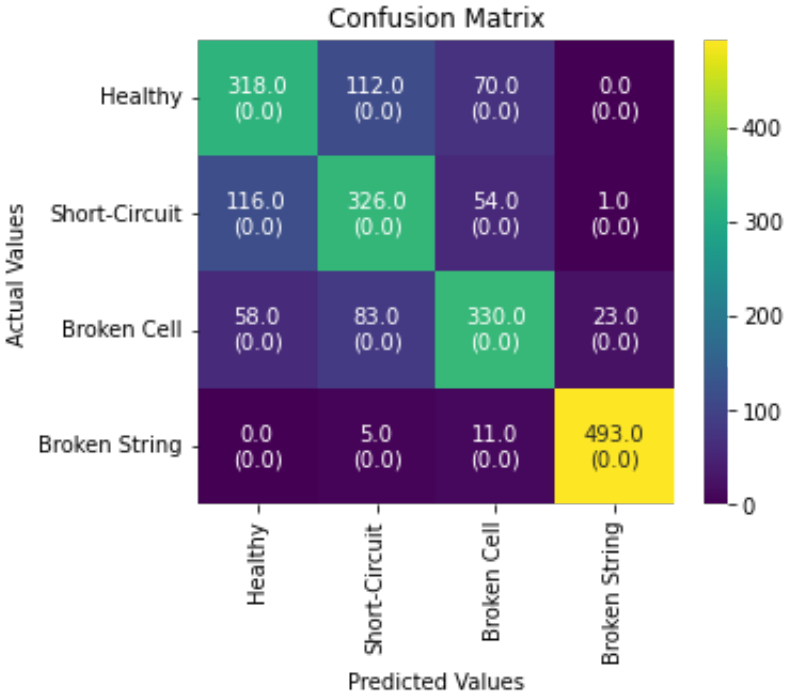


Figure D.31: Confusion matrix of OVO-classifier using support vector machine tree model on test set with a dataset with short-circuit fault class with $R_{fault,max} = 100\,\Omega$.

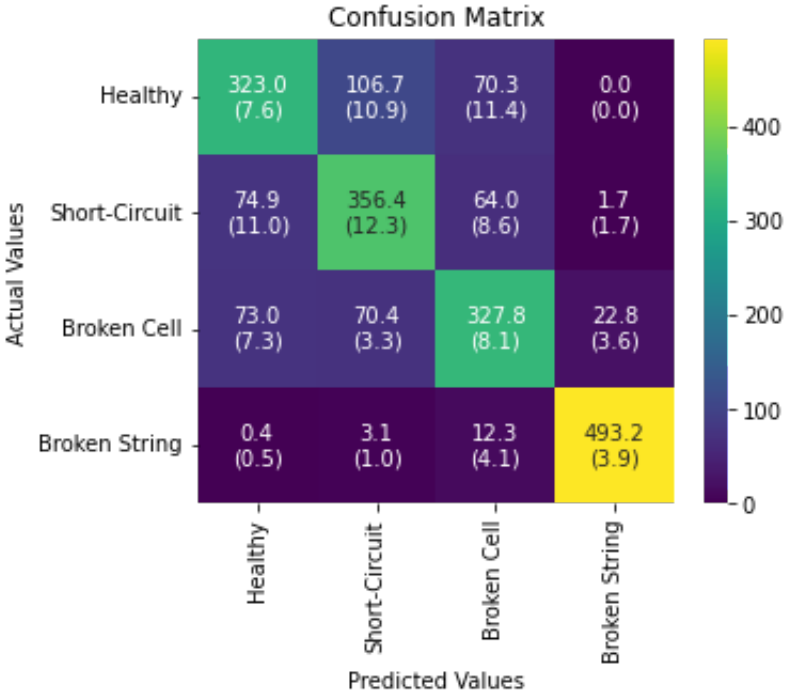


Figure D.32: Confusion matrix of OVO-classifier using neural network model on test set with a dataset with short-circuit fault class with $R_{fault,max} = 100\,\Omega$.

Bibliography

[1] CBS. Productie groene stroom met 40 procent gestegen, 2021. URL https://www.cbs.nl/nl-nl/nieuws/2021/08/productie-groene-stroom-met-40-procent-gestegen.

[2] S. K. Firth, K. J. Lomas, and S. J. Rees. A simple model of PV system performance and its use in fault detection. *Solar Energy*, 84(4):624–635, apr 2010. ISSN 0038092X. doi: 10.1016/j.solener.2009.08.004.

[3] R C Nijman. *Automatically and real-time identifying malfunctioning PV systems using massive on-line PV yield data*. PhD thesis, Delft University of Technology, 2018.

[4] Asma Triki-Lahiani, Afef Bennani-Ben Abdelghani, and Ilhem Slama-Belkhodja. Fault detection and monitoring systems for photovoltaic installations: A review. *Renewable and Sustainable Energy Reviews*, 82:2680–2692, feb 2018. ISSN 18790690. doi: 10.1016/j.rser.2017.09.101.

[5] Santiago Silvestre. Strategies for Fault Detection and Diagnosis of PV Systems. In *Advances in Renewable Energies and Power Technologies*, volume 1, pages 231–255. Elsevier, feb 2018. ISBN 9780128132173. doi: 10.1016/B978-0-12-812959-3.00007-1.

[6] P. Acharya, M. N. Shaikh, S. K. Jha, and A. P. Papadakis. Electrical Modelling Of a Photovoltaic Module. In *Power Systems, Energy Markets and Renewable Energy Sources in South-Eastern Europe*. Trivent Publishing, dec 2016. doi: 10.22618/tp.ei.20163.389014.

[7] Kuei Hsiang Chao, Sheng Han Ho, and Meng Hui Wang. Modeling and fault diagnosis of a photovoltaic system. *Electric Power Systems Research*, 78(1):97–105, jan 2008. ISSN 03787796. doi: 10.1016/j.epsr.2006.12.012.

[8] W. Chine, A. Mellit, V. Lughi, A. Malek, G. Sulligoi, and A. Massi Pavan. A novel fault diagnosis technique for photovoltaic systems based on artificial neural networks. *Renewable Energy*, 90:501–512, may 2016. ISSN 18790682. doi: 10.1016/j.renene.2016.01.036.

[9] A. Chouder and S. Silvestre. Automatic supervision and fault detection of PV systems based on power losses analysis. *Energy Conversion and Management*, 51(10):1929–1937, oct 2010. ISSN 01968904. doi: 10.1016/j.enconman.2010.02.025.

[10] Ye Zhao, Ling Yang, Brad Lehman, Jean François De Palma, Jerry Mosesian, and Robert Lyons. Decision tree-based fault detection and classification in solar photovoltaic arrays. In *Conference Proceedings - IEEE Applied Power Electronics Conference and Exposition - APEC*, pages 93–99, 2012. ISBN 9781457712159. doi: 10.1109/APEC.2012.6165803.

[11] A. Drews, A. C. de Keizer, H. G. Beyer, E. Lorenz, J. Betcke, W. G.J.H.M. van Sark, W. Heydenreich, E. Wiemken, S. Stettler, P. Toggweiler, S. Bofinger, M. Schneider, G. Heilscher, and D. Heinemann. Monitoring and remote failure detection of grid-connected PV systems based on satellite observations. *Solar Energy*, 81(4):548–564, apr 2007. ISSN 0038092X. doi: 10.1016/j.solener.2006.06.019.

[12] Elyes Garoudja, Aissa Chouder, Kamel Kara, and Santiago Silvestre. An enhanced machine learning based approach for failures detection and diagnosis of PV systems. *Energy Conversion and Management*, 151:496–513, nov 2017. ISSN 01968904. doi: 10.1016/j.enconman.2017.09.019.

[13] Sayed A. Zaki, Honglu Zhu, Mohammed Al Fakih, Ahmed Rabee Sayed, and Jianxi Yao. Deep-learning–based method for faults classification of PV system. *IET Renewable Power Generation*, 2021. ISSN 17521424. doi: 10.1049/RPG2.12016. URL https://www.researchgate.net/publication/348518366_Deep-learning-based_method_for_faults_classification_of_PV_system.

[14] H. Mekki, A. Mellit, and H. Salhi. Artificial neural network-based modelling and fault detection of partial shaded photovoltaic modules. *Simulation Modelling Practice and Theory*, 67:1–13, sep 2016. ISSN 1569190X. doi: 10.1016/j.simpat.2016.05.005.

[15] Zhicong Chen, Lijun Wu, Shuying Cheng, Peijie Lin, Yue Wu, and Wencheng Lin. Intelligent fault diagnosis of photovoltaic arrays based on optimized kernel extreme learning machine and I-V characteristics. *Applied Energy*, 204:912–931, oct 2017. ISSN 03062619. doi: 10.1016/j.apenergy.2017.05.034.

[16] Ye Zhao, Roy Ball, Jerry Mosesian, Jean François De Palma, and Brad Lehman. Graph-based semi-supervised learning for fault detection and classification in solar photovoltaic arrays. *IEEE Transactions on Power Electronics*, 30(5):2848–2858, may 2015. ISSN 08858993. doi: 10.1109/TPEL.2014.2364203.

[17] Colombe Warin and Niamh Delaney. Citizen Science and Citizen Engagement. Technical report, European Commission, Brussels, 2020. URL http://europa.eu.

[18] Carlos del Cañizo, Ana Belén Cristóbal, Luisa Barbosa, Gema Revuelta, Sabine Haas, Marta Victoria, and Martin Brocklehurst. Promoting citizen science in the energy sector: Generation Solar, an open database of small-scale solar photovoltaic installations. *Open Research Europe*, 1:21, mar 2021. doi: 10.12688/openreseurope.13069.1.

[19] Dan Stowell, Jack Kelly, Damien Tanner, Jamie Taylor, Ethan Jones, James Geddes, and Ed Chalstrey. A harmonised, high-coverage, open dataset of solar photovoltaic installations in the UK. *Scientific Data*, 7(1):1–15, dec 2020. ISSN 20524463. doi: 10.1038/s41597-020-00739-0. URL www.nature.com/scientificdata.

[20] Navid Haghdadi, Jessie Copper, Anna Bruce, and Iain Macgill. Operational performance analysis of distributed PV systems in Australia. In *Asia-Pacific Solar Research Conference*, 2016.

[21] Nicholas A Engerer and James Hansard. Real-time Simulations of 15,000+ Distributed PV Arrays at Sub-Grid Level using the Regional PV Simulation System (RPSS). In *Solar World Congress*, 2015.

[22] Odysseas Tsafarakis, Panagiotis Moraitis, Bala Bhavya Kausika, Henrik Velde, Saskia 't Hart, Arthur Vries, Peer Rijk, Minne M. Jong, Hans-Peter Leeuwen, and Wilfried Sark. Three years experience in a Dutch public awareness campaign on photovoltaic system performance. *IET Renewable Power Generation*, 11(10):1229–1233, aug 2017. ISSN 1752-1416. doi: 10.1049/iet-rpg.2016.1037. URL https://onlinelibrary.wiley.com/doi/10.1049/iet-rpg.2016.1037.

[23] M. Sabbaghpur Arani and M. A. Hejazi. The Comprehensive Study of Electrical Faults in PV Arrays. *Journal of Electrical and Computer Engineering*, 2016:1–10, 2016. ISSN 2090-0147. doi: 10.1155/2016/8712960. URL https://www.hindawi.com/journals/jece/2016/8712960/.

[24] Mohammed Khorshed Alam, Faisal Khan, Jay Johnson, and Jack Flicker. A Comprehensive Review of Catastrophic Faults in PV Arrays: Types, Detection, and Mitigation Techniques, may 2015. ISSN 21563381.

[25] Ye Zhao, Florent Balboni, Thierry Arnaud, Jerry Mosesian, Roy Ball, and Brad Lehman. Fault experiments in a commercial-scale PV laboratory and fault detection using local outlier factor. In *2014 IEEE 40th Photovoltaic Specialist Conference, PVSC 2014*, pages 3398–3403. Institute of Electrical and Electronics Engineers Inc., oct 2014. ISBN 9781479943982. doi: 10.1109/PVSC.2014.6925661.

[26] A. Charki, P. O. Logerais, D. Bigaud, C. M.F. Kébé, and A. Ndiaye. Lifetime assessment of a photovoltaic system using stochastic Petri nets. *International Journal of Modelling and Simulation*, 37(3):149–155, jul 2017. doi: 10.1080/02286203.2017.1297923.

[27] A. Sayed, M. El-Shimy, M. El-Metwally, and M. Elshahed. Reliability, availability and maintainability analysis for grid-connected solar photovoltaic systems. *Energies*, 12(7):1213, mar 2019. ISSN 19961073. doi: 10.3390/en12071213. URL www.mdpi.com/journal/energies.

[28] Gabriele Zini, Christophe Mangeant, and Jens Merten. Reliability of large-scale grid-connected photovoltaic systems. *Renewable Energy*, 36(9):2334–2340, sep 2011. ISSN 09601481. doi: 10.1016/j.renene.2011.01.036.

[29] Michaël Bressan, Youssef El Basri, Corinne Alonso, and Alonso A Corinne. A new method for fault detection and identification of shadows based on electrical signature of defects. In *EPE'15 ECCE Europe*, 2015. doi: 10.1109/EPE.2015.7309177Ãf. URL https://hal.archives-ouvertes.fr/hal-01208396.

[30] Kelly (Solairgen School of Solar Technology) Provence. Failure Causes in Solar PV Systems, 2019. URL https://solairgen.com/failure-causes-in-solar-pv-systems/.

[31] Geoffrey T Klise, Olga Lavrova, and Renee Gooding. SANDIA REPORT PV System Component Fault and Failure Compilation and Analysis. Technical report, Sandia National Laboratories, 2018. URL https://classic.ntis.gov/help/order-methods/.

[32] Jack D Flicker and Jay Johnson. SANDIA REPORT Photovoltaic Ground Fault and Blind Spot Electrical Simulations. Technical report, Sandia National Laboratories, 2013. URL http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online.

[33] Ye. Zhao and Robert Lyons. Ground-Fault Analysis and Protection in PV Arrays. Technical report, Mersen, 2017.

[34] Ye. Zhao and Robert Lyons. Line-Line Fault Analysis and Protection in PV Arrays. Technical report, Mersen, 2016. URL https://ep-us.mersen.com/sites/mersen_us/files/2018-11/TT-PVPN2-Line-Line-Fault-Analysis-and-Protection-in-PV-Arrays-Tech-Topic.pdf.

[35] Mohammed Khorshed Alam, Faisal H Khan, Jay Johnson, and Jack Flicker. PV Faults: Overview, Modeling, Protection and Detection Techniques. *IEEE 14th Workshop on Control and Modeling for Power Electronics (COMPEL)*, pages 1–7, 2013.

[36] Ye Zhao. *Fault Analysis in Solar Photovoltaic Arrays*. PhD thesis, Northeastern University, 2010.

[37] Siva Ramakrishna Madeti and S. N. Singh. Modeling of PV system based on experimental data for fault detection using kNN method. *Solar Energy*, 173:139–151, oct 2018. ISSN 0038-092X. doi: 10.1016/J.SOLENER.2018.07.038.

[38] Nuri Gokmen, Engin Karatepe, Berk Celik, and Santiago Silvestre. Simple diagnostic approach for determining of faulted PV modules in string based PV arrays. *Solar Energy*, 86(11):3364–3377, nov 2012. ISSN 0038-092X. doi: 10.1016/J.SOLENER.2012.09.007.

[39] SolarEdge Technologies Inc. Technical Note Bypass Diode Effects in Shaded Conditions. Technical report, SolarEdge, 2010. URL https://www.solaredge.com/resource-library#/.

[40] Francisco Serdio Fernández, Miguel Angel Muñoz-García, and Susanne Saminger-Platz. Detecting clipping in photovoltaic solar plants using fuzzy systems on the feature space. *Solar Energy*, 132:345–356, jul 2016. ISSN 0038-092X. doi: 10.1016/J.SOLENER.2016.03.013.

[41] U.S. Energy Information Administation. Solar plants typically install more panel capacity relative to their inverter capacity - Today in Energy - U.S. Energy Information Administration (EIA), mar 2018. URL https://www.eia.gov/todayinenergy/detail.php?id=35372.

[42] Hiren Patel and Vivek Agarwal. MATLAB-based modeling to study the effects of partial shading on PV array characteristics. *IEEE Transactions on Energy Conversion*, 23(1):302–310, mar 2008. ISSN 08858969. doi: 10.1109/TEC.2007.914308.

[43] Hoang. Pham. *System software reliability*. Springer, 1 edition, 2006. ISBN 1-85233-950-0.

[44] Dirk P. Kroese, Thomas Taimre, and Zdravko I. Botev. *Handbook for Monte Carlo methods*. John Wiley & Sons, Inc., Hoboken, 2011. ISBN 978-0-470-17793-8.

[45] CBS. Hernieuwbare Energie in Nederland 2020, 2021. URL https://www.cbs.nl/nl-nl/longread/aanvullende-statistische-diensten/2021/hernieuwbare-energie-in-nederland-2020/5-zonne-energie.

[46] Mahmoud Dhimish and Abdullah Alrashidi. Photovoltaic Degradation Rate Affected by Different Weather Conditions: A Case Study Based on PV Systems in the UK and Australia. *Electronics 2020, Vol. 9, Page 650*, 9(4):650, apr 2020. doi: 10.3390/ELECTRONICS9040650. URL https://www.mdpi.com/2079-9292/9/4/650/htmhttps://www.mdpi.com/2079-9292/9/4/650.

[47] Trevor Hastie, Robert Tibshirani, Gareth James, and Daniela Witten. An introduction to statistical learning (2nd ed.). *Springer texts*, 102:618, 2021. ISSN 01621459.

[48] Diederik P. Kingma and Jimmy Lei Ba. Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, dec 2014. doi: 10.48550/arxiv.1412.6980. URL https://arxiv.org/abs/1412.6980v9.

[49] IBM Cloud Education. What is Underfitting? | IBM, 2021. URL https://www.ibm.com/cloud/learn/underfitting.