AN IOT AND SUPERVISED LEARNING BASED SENSORLESS TECHNIQUE FOR PANEL LEVEL SOLAR PV ARRAY FAULT DIAGNOSIS

PROJECT PHASE-1 REPORT

submitted by

LIYON APUM
Reg. No : **MAC20EE062**

MUHAMMED SAFAD P
Reg. No : **MAC20EE070**

SAIYAD ALI RAPHEEQUE
Reg. No : **MAC20EE094**

ARJUN T
Reg. No : **LMAC20EE116**

to

the APJ Abdul Kalam Technological University
in partial fulfillment of the requirements for the award of the Degree

of

Bachelor of Technology
in
Electrical and Electronics Engineering

Department of Electrical and Electronics Engineering

Mar Athanasius College of Engineering
Kothamangalam, Kerala, India 686 666

DECEMBER 2023

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
MAR ATHANASIUS COLLEGE OF ENGINEERING
KOTHAMANGALAM

CERTIFICATE

This is to certify that the report entitled **An Iot and Supervised Learning Based Sensor-Less Technique for Panel Level SPV Array Fault Diagnosis** submitted by **Mr.Saiyad Ali Rapheeque** (**Reg.No.MAC20EE094**) to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Electrical and Electronics is a bonafide record of the project carried out by him under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Prof. Leela Salim	Prof. Leela Salim	Smt. Beena M Varghese
Project Guide	Project Coordinator	Head of the Dept.

Dept. Seal

06 - 12 - 2023

ACKNOWLEDGEMENT

It is a great pleasure to acknowledge all those who have assisted and supported me for successfully completing our project.

First of all, I thank God Almighty for his blessings as it is only through his grace that I was able to complete my project successfully.

I take this opportunity to extend my sincere thanks to our project Guide Prof. Leela Salim, Assistant Professor, Department of Electrical & Electronics Engineering for her constant support and immense contribution for the success of our project.

I also extend my sincere thanks to our faculty advisor and Project Coordinator Prof. Leela Salim, Assistant Professor, Electrical & Electronics Engineering Department and all other members of the Department of Electrical & Electronics Engineering for sharing their valuable comments during the preparation of our project.

I am also grateful to Prof. Beena M Varghese, Head of Electrical & Electronics Engineering Department for the valuable guidance as well as timely advice which helped us a lot during the preparation of the project.

I am deeply indebted to Dr. Bos Mathew Jos, Principal, Mar Athanasius College of Engineering for his encouragement and support.

I whole heartedly thank all my classmates, for their valuable suggestions and for the spirit of healthy competition that existed between us.

ABSTRACT

An IoT and Supervised Learning-Based Sensor-less Technique for Panel Level Solar Photovoltaic Array Fault Diagnosis introduces an innovative fault diagnosis methodology for Solar Photovoltaic (SPV) arrays, featuring a sensor-less electronic circuit with a bipolar junction transistor (BJT) and Zener diode for fault detection. Integrated with an IoT-based web application for precise fault localization at the panel level, the methodology relies on supervised learning techniques, such as Support Vector Machine (SVM) and Logistic Regression (LR), for fault classification. Demonstrating potential cost-effectiveness and data efficiency, the methodology's simulated accuracies in fault detection and classification appear promising. Primary simulations encompassing LTspice-based FDEC implementation, MATLAB/SIMULINK data collection, fault characteristic analysis, Python-based Fault Monitoring and Data Acquisition System (FMDAS) deployment, and machine learning (ML) algorithms for fault classification yield encouraging outcomes. While pending laboratory-scale experiments, these preliminary simulated results align with envisioned objectives, offering a promising path for SPV array fault diagnosis. The approach holds potential in addressing critical challenges within SPV arrays, presenting a prospective stride in solar energy technology. Further validation via physical experiments is necessary to substantiate these findings and adapt the methodology for practical application in real-world scenarios, aligning with the dynamic needs of modern microgrids and operating environments.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The performance of solar photovoltaic (SPV) systems is critical for meeting escalating energy demands sustainably. Yet, SPV arrays are vulnerable to faults that detrimentally affect their efficiency and lifespan. Conventional fault detection methods struggle due to intricate environmental conditions and the nonlinear behavior of solar cells. Moreover, prevalent learning-based approaches demand copious data and costly sensors, limiting practicality for many SPV systems.

This article introduces an innovative fault diagnosis methodology revolutionizing SPV arrays. It not only detects but also precisely localizes and categorizes faults. What sets this approach apart is its fusion of a sensor-less electronic circuit, an IoT-driven monitoring system, and a sophisticated supervised learning model. This unique amalgamation enables advanced fault diagnosis and management within solar photovoltaic systems.

At the core of this methodology lies the Fault Detection Electronic Circuit (FDEC), utilizing components like a bipolar junction transistor (BJT) and Zener diode. This circuit adeptly identifies various faults, including Line to Line (LL), Line to Ground (LG), Open Circuit (OC), and Partial Shading Conditions (PSCs).

To circumvent heavy reliance on labeled data in traditional supervised learning, a novel supervised learning approach is introduced. This method efficiently classifies faults even with limited labeled data availability. By leveraging strategically designed datasets, it optimizes fault classification accuracy.

The machine learning methodology hinges on the Support Vector Machine and Logistic Regression (SVM-LR) model. This fusion excels in fault classification, providing precision in identifying and categorizing faults within SPV arrays. The SVM and logistic regression collaboration significantly contributes to fault diagnosis, reducing dependence on labeled datasets.

Critical to this methodology is its independence from excessive data and costly sensors. The IoT-based Fault Monitoring and Data Acquisition System (FMDAS) integral to this setup oversees individual solar panels for fault localization while minimizing redundant data transmission. Selective data relay to the server upon anomaly detection streamlines operations, conserving resources.

1.1 Research Objectives

The objective is to develop an innovative fault diagnosis methodology for Solar PV systems, integrating sensor-less fault detection, IoT-based monitoring, and machine learning models for precise fault localization and classification. The aim is to overcome limitations in existing fault diagnosis approaches, optimizing accuracy, cost-effectiveness, and sustainability in SPV array fault detection.

1.2 Overview of the Thesis

Chapter 1 sets the stage, outlining the scope and significance of fault diagnosis in Solar PV systems.

Chapter 2 delves into existing research and methodologies related to Solar PV system fault diagnosis.

Chapter 3 introduces Solar PV systems, emphasizing the need for fault monitoring and detailing various fault types.

Chapter 4 explores fault detection methods and details the design of the Fault Detection Electronic Circuit (FDEC).

Chapter 5 presents outcomes from LTspice implementation, IoT-based FMDAS, dataset creation, and machine learning model comparisons.

Chapter 6 summarizes key findings, implications, and future directions for fault diagnosis in Solar PV systems.

CHAPTER 2

LITERATURE REVIEW

Mohammed Khorshed et.al[1] discuss faults in photovoltaic (PV) arrays, such as ground faults, line-to-line faults, and arc faults, can lead to catastrophic failures. While these incidents have been relatively infrequent, events like the fires in Bakersfield, CA, USA, on April 5, 2009, and in Mount Holly, NC, USA, on April 16, 2011, underscore the necessity for enhancements in fault detection, mitigation techniques, and updates to existing codes and standards. This examination delves into how faults impact the functioning of PV arrays and pinpoints the constraints of current detection and mitigation approaches.

Hariharan et.al[2] discuss anomalies like faults and partial shading can diminish the maximum power a photovoltaic (PV) array can produce. Detecting these issues is crucial for enhancing the efficiency and reliability of the system. Traditional protection mechanisms often fall short in identifying faults during cloudy or low irradiance situations, potentially leading to safety concerns and fire risks in PV setups. This study introduces a method that utilizes measurements of array voltage, array current, and irradiance to detect faults and partial shading across all irradiation scenarios. The proposed method categorizes the PV array's status into three scenarios: normal operation, partial shading, and faults.

Hu Yihua et.al[3] explores that PV stations worldwide harness solar energy through photovoltaic arrays, but minimizing costs and ensuring efficient operation is crucial. Early fault detection has become increasingly pivotal, prompting the development of a fault diagnosis technique analyzing terminal characteristics of faulty PV strings and arrays. The technique divides the terminal current-voltage curve of a faulty PV array into high-voltage and low-voltage sections for diagnosis. It involves analyzing working points of healthy string modules and both healthy and faulty modules within an unhealthy string in each section, enabling precise identification of faulty PV modules.

This fault information is vital for tasks like maximum power point tracking and array reconfiguration. Additionally, optimizing voltage sensor placement can reduce the need for string current sensors and decrease the number of voltage sensors required.

K. Yurtseven et.al[4] presents an innovative approach to fault detection in photovoltaic (PV) systems. The proposed method relies on mapping the inherent characteristics of the PV plant site, offering a simple and practical sensorless solution. By leveraging the unique features of the PV system, the authors aim to detect faults efficiently. The paper likely discusses the methodology, results, and practical implications of this approach, providing valuable insights for enhancing the reliability and performance of PV systems.

Yi Zhehan et.al[5] introduces a method to detect DC side short-circuit faults in photovoltaic (PV) arrays comprising multiple PV panels interconnected in a series/parallel setup. These faults are challenging to identify, especially under low irradiance conditions and when a maximum power point tracking algorithm is active. Left undetected, such faults can significantly reduce solar system output, harm the panels, and pose fire risks. The proposed detection scheme relies on a pattern recognition strategy utilizing multiresolution signal decomposition to extract relevant features. These features are then used by a fuzzy inference system to determine fault occurrences. Through case studies involving simulations and experiments, the effectiveness and reliability of this approach in detecting faults in PV arrays are demonstrated.

Li Kui et.al[6] discusses a novel approach to tackle the challenge of detecting and isolating arc faults in DC microgrids and photovoltaic systems. By utilizing a planar location method requiring only two detection points, the system forms a horizontal triangle between an antenna array and the fault source, capitalizing on the relatively smaller height compared to the horizontal dimension of these systems. Signal pulses are isolated using cross-correlation techniques, and a combination of neural networks (NN) and received signal strength indicators (RSSIs) helps estimate the arc fault distance. Recognizing the limitation of data from dual detection points for NN training, a data-augmented NN (DANN) technique is introduced to bolster accuracy and robustness in distance estimation

CHAPTER 3

SOLAR PV SYSTEM

3.1 Introduction

A solar PV (photovoltaic) system is a renewable energy technology that converts sunlight into electricity. It consists of solar panels made up of numerous solar cells, which capture photons from the sun and generate direct current (DC) electricity. An inverter then converts this DC power into alternating current (AC) electricity, suitable for use in homes or businesses. The generated electricity can either be consumed on-site, stored in batteries for later use, or fed back into the grid, often through net metering arrangements, allowing the system owner to earn credits or financial incentives. Solar PV systems are environmentally friendly, reducing greenhouse gas emissions and reliance on fossil fuels while contributing to a more sustainable and resilient energy future.

3.2 Need for Fault Monitoring

Fault monitoring in solar photovoltaic (PV) systems is essential for ensuring the reliable and efficient operation of solar power installations. Solar PV systems are exposed to various environmental stressors, including temperature fluctuations, dust and debris, and potential electrical and mechanical faults, which can significantly impact their performance. Detecting and addressing these faults in a timely manner is crucial to maximize energy production, reduce downtime, and extend the lifespan of the equipment. Fault monitoring systems employ advanced technologies such as remote sensing, data analytics, and automated alerts to continuously assess the health of the solar PV system. They can identify issues like panel degradation, inverter malfunctions, shading, and electrical wiring problems. By proactively identifying and mitigating these faults, solar PV owners and operators can minimize energy losses, lower maintenance costs, and ensure a stable power supply, contributing to a more sustainable and reliable energy source. Additionally, fault monitoring systems play a critical role in enhancing the safety of the system by identifying potential hazards and enabling prompt corrective actions, thereby reducing the risk of electrical fires and other safety concerns associated with solar PV installations.

3.3 Types of Faults in PV Systems

Within the domain of Solar Photovoltaic (SPV) modules, a spectrum of faults can arise, causing disruptions in the smooth functioning of these systems. These faults present a variety of abnormalities that impact the operational efficiency and effectiveness of both individual solar panels and the overall array. Common manifestations of faults in SPV modules are diverse, encompassing:

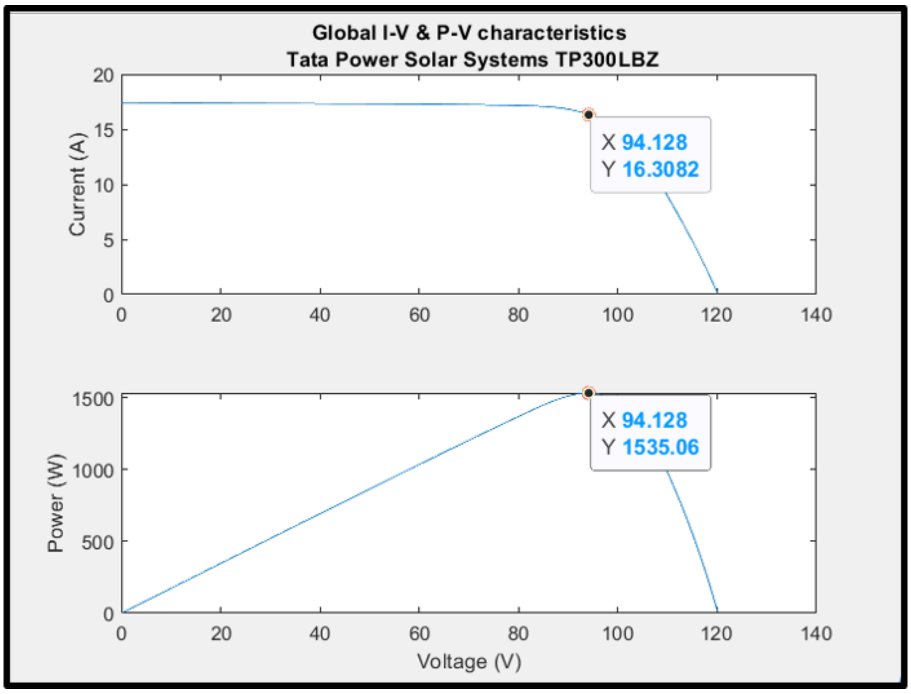


Figure 3.1: SPV Characteristics of Healthy Panel.

3.3.1 Line-to-Line Fault

A line-to-line fault in a solar PV system occurs when two conductors (wires) carrying electrical current come into direct contact with each other. This type of fault can lead to a short circuit, resulting in a surge of current that can damage components such as inverters and pose safety risks. It can be caused by physical damage to wiring or faulty connections.

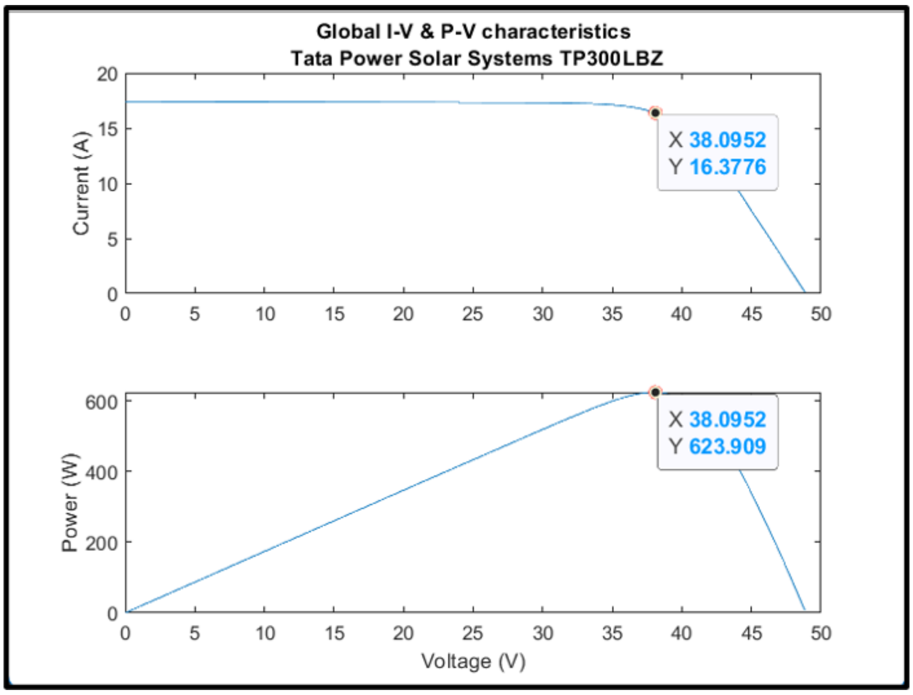


Figure 3.2: SPV Different Associated Faults: Line-to-Line (LL) fault.

The provided graph (Fig. 3.2) illustrates the behavior of a solar panel system affected by a line-to-line (LL) fault. In this scenario, two wires within the solar panel system come into contact, resulting in a short circuit that diminishes both the voltage and power output. The graph (Fig. 3.2) consists of two distinct graphs portraying the relationship between current and voltage, as well as power and voltage.

The graph (Fig. 3.2) depicting current and voltage displays how the current (measured in A) varies in response to changes in the voltage (measured in V) across the solar panel system experiencing an LL fault. It showcases an I-V curve represented by a blue line. This curve begins at a point of high current and low voltage, eventually trailing off to a point of low current and high voltage. This particular curve shape signifies that the solar panel system encounters reduced resistance and diminished output voltage due to the LL fault. Notably, two red circles mark intersections on this I-V curve, denoting the short circuit current (ISC) and the open circuit voltage (VOC). ISC stands as the maximum current achievable when the voltage hits zero, while VOC represents the maximum voltage when the current is at zero. Both ISC and VOC are notably lower than their usual values due to the presence of the LL fault.

Meanwhile, the power and voltage graph showcase the relationship between power (measured in MW) and voltage (measured in V) in the solar panel system with an LL fault. This graph (Fig. 3.2) illustrates a P-V curve delineated by a blue line that begins at zero, ascends to reach a maximum point, and subsequently descends back to zero. The apex of this curve identifies the maximum power point (MPP), indicating the optimal operational point for the solar panel system. Additionally, a red circle marks the intersection on this P-V curve, representing the MPP of the solar panel system affected by the LL fault. Notably, this MPP registers lower than in normal operating conditions due to the impact of the LL fault.

3.3.2 Line-to-Ground Fault

A line-to-ground fault, also known as a ground fault, happens when one of the conductors in a PV system comes into contact with the ground or a conductive surface. This type of fault can lead to electrical leakage, posing a risk of electric shock and system under performance. Ground faults are typically detected and mitigated using ground fault protection devices and proper grounding systems.

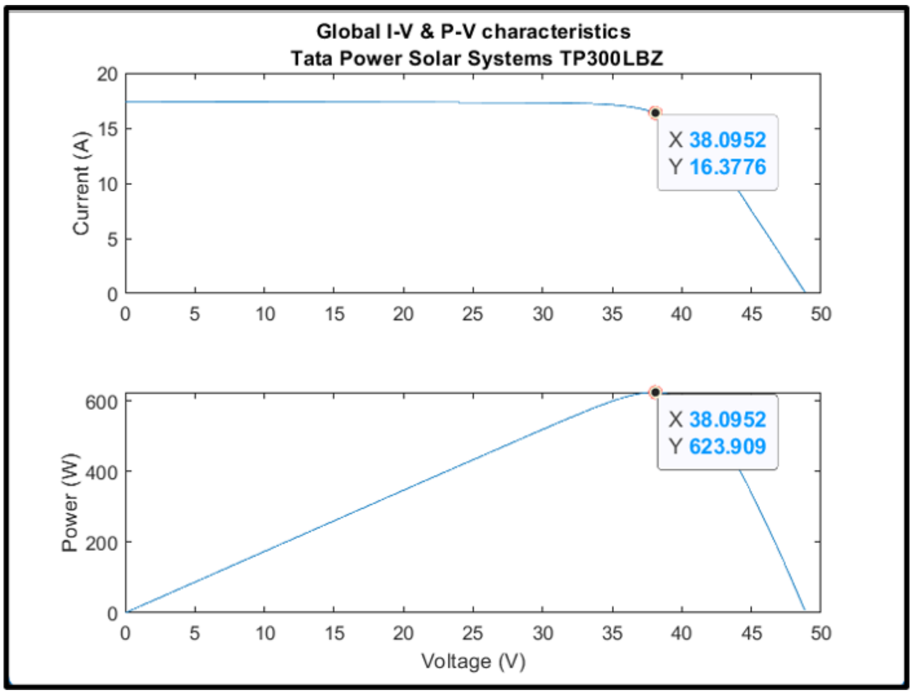


Figure 3.3: SPV Different Associated Faults: Line-to-Ground (LG) fault.

The graph (Fig. 3.3) depicts the behavior of a solar panel system experiencing a line-to-ground (LG) fault, a situation where one of the wires in the solar panel setup comes into contact with the ground. This contact causes a leakage of current and subsequently results in a reduction in voltage within the system. The visual includes two distinct graphs showcasing the characteristics of this fault.

The first graph illustrates the relationship between the current (measured in Amperes) and voltage (measured in Volts) of the solar panel system. It displays a blue line known as the I-V curve, tracing a path from high current and low voltage to low current and high voltage. This curve's shape signifies that the solar panel system is experiencing both low resistance and low output voltage, directly attributed to the line-to-ground fault. Additionally, a marked data point, located at coordinates (38.9052, 16.3776) on the I-V curve, signifies the operational point of the solar panel system under this fault condition.

The second graph portrays the relationship between the power output (measured in Megawatts) and the voltage of the solar panel system. It exhibits a red line, referred to as the P-V curve, which rises from zero to a maximum point before dropping back to zero. The highest point on this curve represents the maximum power point (MPP), indicating the optimal operating point of the solar panel system. Similarly, a marked data point, aligning with the coordinates (38.9052, 23.0909) on the P-V curve, mirrors the previous data point from the I-V curve. This data point reflects the power output of the solar panel system during the line-to-ground fault, showcasing a lower power output compared to the standard operation due to this fault.

3.3.3 Open Circuit Fault

An open circuit fault in a solar PV system occurs when there is a break or disconnection in the electrical circuit, preventing the flow of current. This fault can result from damaged or disconnected wiring, loose connections, or faulty components, leading to a significant decrease in energy production. Open circuit faults need to be identified and repaired to restore the system's functionality.

The graph (Fig. 3.4) illustrates the effects of an open circuit fault in a solar photovoltaic (SPV) system. Such a fault occurs when there's a break or gap in the wiring of the SPV system, leading to a loss of current and power. The image consists of two graphs: one depicting the relationship between current and voltage, and the other showcasing power against voltage.

In the current and voltage graph, there's a blue line representing the I-V curve. It starts at a point with high current and low voltage, ending at zero current and high voltage. This curve shape indicates that the SPV system exhibits high resistance and produces a high output voltage due to the open circuit fault. The graph marks a point labeled "Y 16.26" on the I-V curve, representing the system's operating point under the

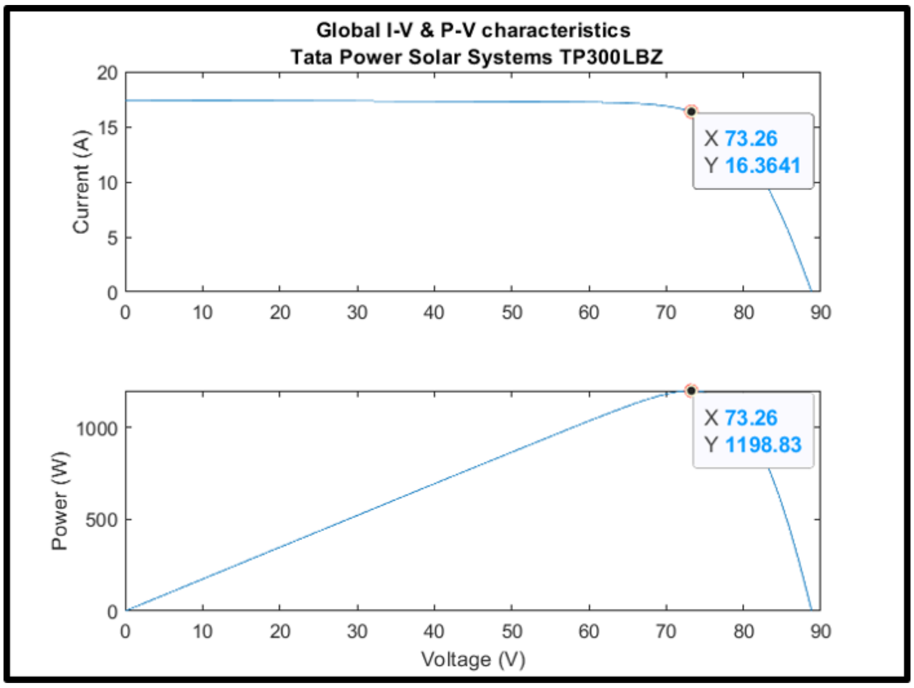


Figure 3.4: SPV Different Associated Faults: Open Circuit (OC) fault.

fault condition, known as the open circuit voltage (VOC). VOC denotes the maximum voltage the SPV system can generate when the current is zero. It's notably higher than the normal case due to the open circuit fault.

In the power and voltage graph, a green line represents the P-V curve. It starts at zero, reaches a maximum point, then drops back to zero. The peak of this curve marks the maximum power point (MPP), which signifies the optimal operating point of the SPV system. The graph also indicates a point labeled "Y 1198.23" on the P-V curve, representing the power output of the SPV system under the fault condition, which is the MPP. However, the MPP under the open circuit fault condition is lower compared to the normal case, reflecting the impact of the fault on the SPV system's power output.

3.3.4 Short Circuit Fault

A short circuit fault in a solar PV system happens when there's an unintended connection between two points in the electrical circuit with low resistance. This fault can occur due to damaged insulation, loose wiring, or faulty components, causing an excessive flow of current. Short circuit faults pose risks like overheating, equipment damage, and potential fire hazards. Identifying and fixing these faults promptly is crucial to ensure system safety and efficiency.

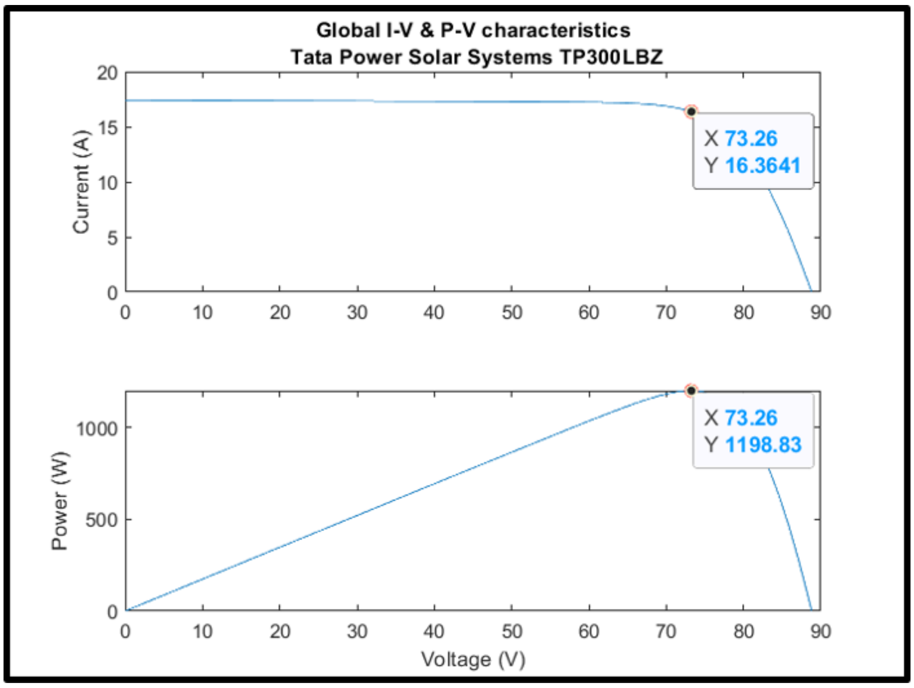


Figure 3.5: SPV Different Associated Faults: Open Circuit (OC) fault.

The graph (Fig. 3.5) presents the characteristics of a solar panel system affected by a short circuit (SC) fault. A short circuit fault occurs when current travels through an unintended path, resulting in a decrease in both voltage and power output. The image contains two graphs illustrating these effects.

The first graph, depicting current and voltage, showcases the relationship between the current (A) and voltage (V) within the solar panel system. This graph features a linear line, known as the I-V curve, starting from a high current and low voltage point and ending at a low current and high voltage point. This specific curve shape signifies the system's low resistance and reduced output voltage due to the short circuit fault. Additionally, marked data points on the I-V curve, such as (73.26, 16.3641) and P-V Curve (73.26, 1198.83), represent the system's operational points under varying conditions.

The second graph, demonstrating power and voltage, displays the correlation between power (MW) and voltage (V) in the solar panel system. Calculated by multiplying current and voltage, the power is illustrated via a parabolic curve called the P-V curve. This curve initiates from zero, reaches a peak indicating the maximum power point (MPP), and then descends back to zero. Similar to the I-V curve, the P-V curve also showcases the same two marked data points, denoting the power output of the system under different conditions. Notably, due to the short circuit fault, the power output is lower compared to the normal operating condition.

3.3.5 Partial Shading Condition

Partial shading in a solar PV system occurs when some of the solar panels are shaded while others are exposed to sunlight. This condition can significantly reduce the overall system's energy output due to the mismatch in power production between shaded and unshaded panels. In such scenarios, bypass diodes and maximum power point tracking (MPPT) algorithms in inverters are used to mitigate the effects of shading and optimize energy generation by managing the voltage and current levels in the system.

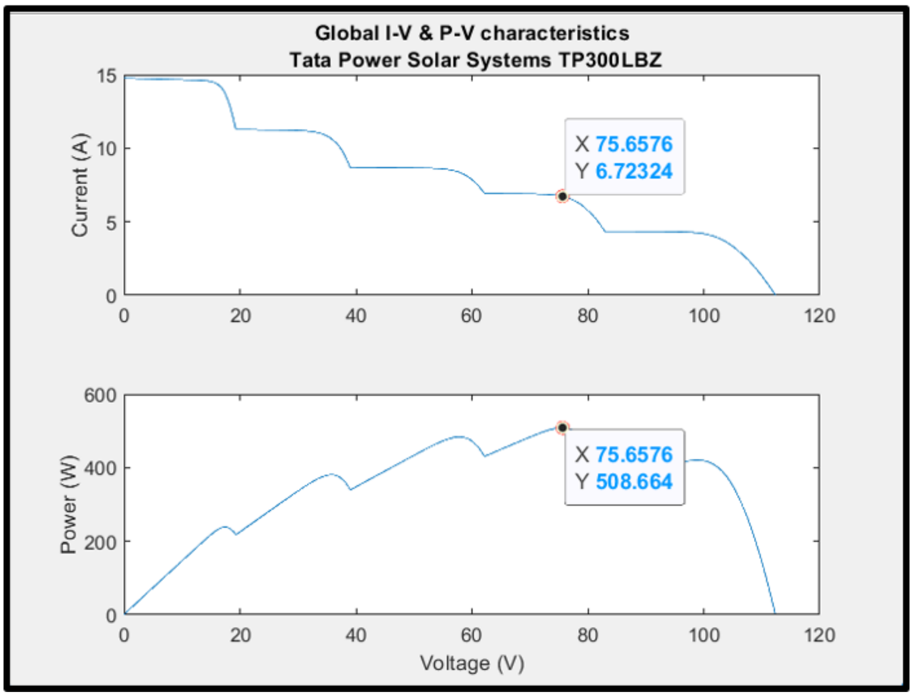


Figure 3.6: SPV Different Associated Faults: Partial Shading Conditions.

This graph (Fig. 3.6) depicts the behavior of a solar panel when subjected to partial shading conditions, where certain areas of the panel are obstructed from receiving sunlight. This obstruction results in decreased power output and efficiency of the solar panel. The graph comprises two graphs illustrating the relationship between current and voltage, as well as power and voltage.

The current and voltage graph portrays fluctuations in current (A) concerning the voltage (V) of the solar panel amidst partial shading conditions. The graph exhibits a blue curve representing thebI-V curve. Unlike the standard smooth and steep I-V curve, the partial shading scenario introduces multiple peaks and valleys, known as local maxima and minima. These irregularities are consequences of current and voltage mismatches among the interconnected solar cells. The highest point on the I-V curve, termed the global maximum, signifies the optimal operational point of the solar panel under partial shading. An orange marker at (75.6576, 6.7376) highlights the operating point of the solar panel within these conditions.

The power and voltage graph showcases the relationship between power (W) and voltage (V) of the solar panel during partial shading. The blue curve, referred to as the P-V curve, starts from zero, reaches a peak, and then diminishes back to zero. Unlike the typical parabolic and symmetrical P-V curve, partial shading causes multiple peaks and valleys, aligning with the local maxima and minima on the I-V curve. The highest point on the P-V curve, known as the global maximum power point (GMPP), signifies the optimal power output of the solar panel under partial shading. An orange marker at (75.6576, 508.664) represents the power output of the solar panel within this condition, notably lower than the standard output due to partial shading.

The Faults discussed in this section is summorized by the Fig. 3.7. The faults and corresponding indications from the image are given: PSC (Partial Shading Conditions) (F5); OC (Open Circuit Fault) (F3); SC (Short Circuit Fault) (F2); L-L Fault (Line to Line) (F1); L-G Fault (Line to Ground) (F4);

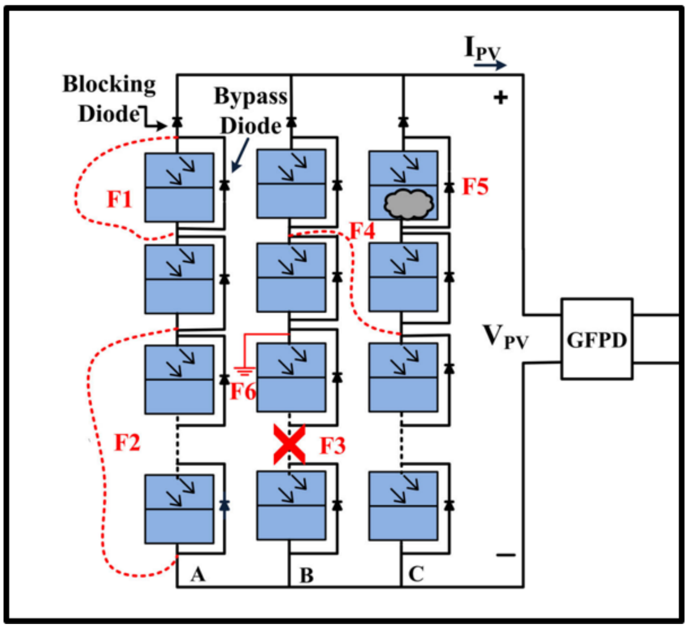


Figure 3.7: SPV Different Associated Faults.

3.4 Schematic Diagram of Proposed System

Solar panels serve as a crucial mechanism for converting sunlight into electricity, offering a clean and renewable energy source. However, these panels are susceptible to various faults that can significantly diminish their effectiveness. Therefore, it becomes imperative to promptly and accurately identify, localize, and categorize these faults to ensure optimal performance.

One viable method to accomplish this involves the implementation of an IoT-driven Fault Monitoring and Detection System (FMDAS). This system works by gathering data from the solar panel array and transmitting it to a cloud server. Here, a machine learning algorithm, such as Support Vector Machine-Logistic Regression (SVM-LR), processes the data to pinpoint the fault's type and location. Additionally, the SVM-LR classifier can generate a confusion matrix, providing insights into the precision and accuracy of fault diagnosis.

The Flowchart (Fig. 3.8) visually represents the process of fault detection, localization, and classification within a solar panel system. The flowchart outlines three distinct steps: fault detection, localization, and classification. The illustration also delineates the system's components, including the solar panel array, the IoT-based FMDAS, the SVM-LR classifier, and the confusion matrix. Altogether, the image serves as a demonstration of how an automated and intelligent system can significantly enhance the reliability and performance of solar panels.

Moreover, in the realm of fault detection in electronic circuits, specialized circuits

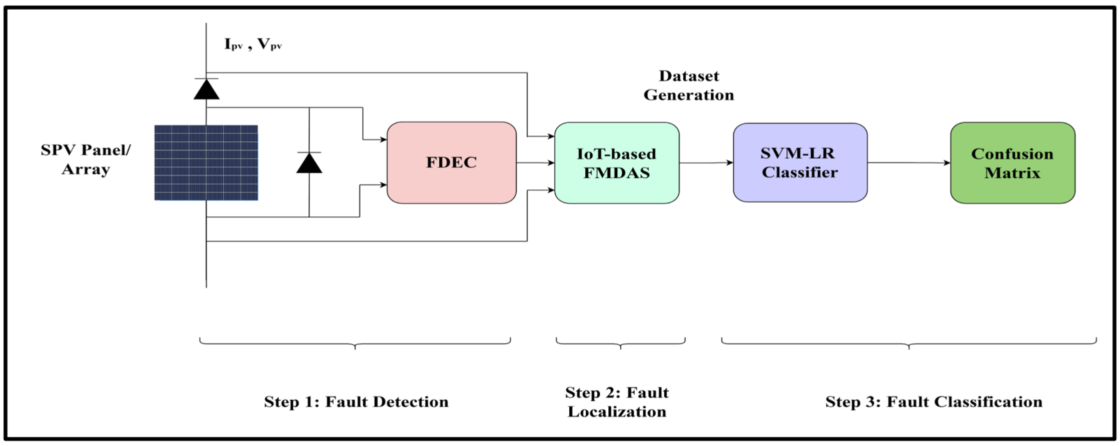


Figure 3.8: Schematic of the proposed SPV array fault detection, localization, classification methodology.

exist to identify issues like short circuits, open circuits, or component failures. These circuits are designed to gauge and indicate the fault's type and location by modulating the voltage output. This technology significantly contributes to the identification of faults within electronic systems.

3.5 Conclusion

In conclusion, the incorporation of fault monitoring within solar PV systems emerges as a critical necessity owing to the wide array of potential faults inherent to these systems. Ranging from line-to-line and line-to-ground faults to open and short circuits, as well as the complex issues stemming from partial shading conditions, a comprehensive approach to monitoring and detection becomes indispensable. The outlined schematic diagram of the proposed system serves as a foundational guide, delineating a robust monitoring framework meticulously designed to address these diverse fault scenarios. By acknowledging the paramount importance of fault monitoring and comprehending the spectrum of potential issues, this chapter lays the groundwork for the implementation of effective fault detection and mitigation strategies. Such strategies are pivotal in safeguarding the optimal functionality and safety standards upheld by solar PV systems.

CHAPTER 4

FAULT DETECTION

4.1 Introduction

Fault detection in solar PV (photovoltaic) arrays is a critical aspect of ensuring efficient and reliable renewable energy generation. These systems are susceptible to various issues, including module degradation, shading, soiling, electrical faults, and environmental factors, all of which can significantly impact their performance and overall energy output. Detecting and diagnosing faults in a timely manner is essential to minimize downtime and maintenance costs while maximizing energy yield.

4.2 Electronic Circuit Design

A BJT and Zener diode-based sensorless electronic circuit is designed to detect faulty or bypassed SPV panels. Zener diode is used in a reverse-biased mode in the breakdown region to maintain the voltage across it. In addition, it works as a shielding device that prevents large reverse currents and sudden voltage spikes during sudden faulty events. In the proposed detection circuit, an n-p-n-BJT is operated in saturation and cut-off region. BJT with Zener diode acts as a constant current regulator.

Fig.4.1 shows the integration of the fault detection circuit with SPV panels. The input of the BJT (base terminal of BJT) is connected to the bypass diode of the SPV panel through the Zener diode. Two resistors R1, R2 and the output of the transistor is connected to the biasing supply of 5 V through collector resistance R3. R1 and R2 are arranged in a voltage divider path, whose midpoint is used as output for fault detection. R2 is selected to limit the current flowing through the Zener diode. With the power (P_Z) rated 1 W and voltage (V_Z) rated 5.1 V, the maximum Zener current can be calculated as:

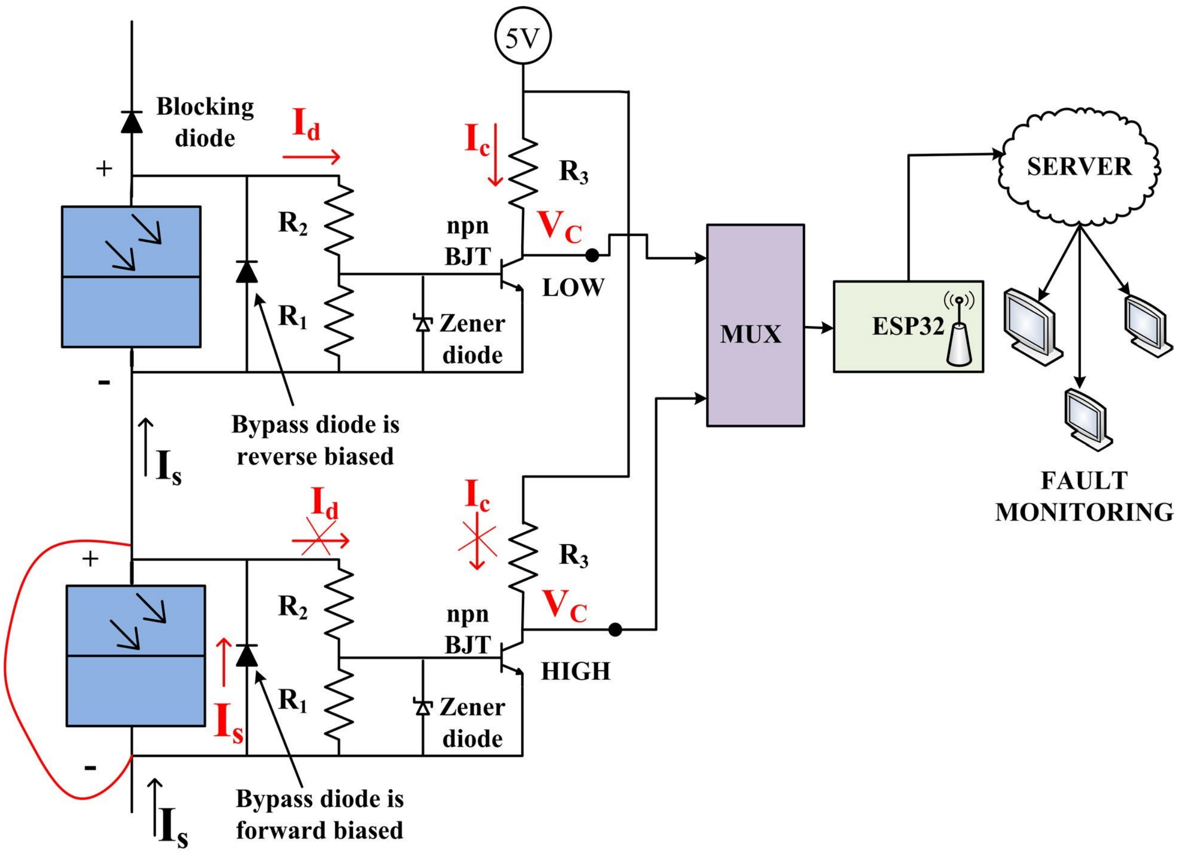


Figure 4.1: Fault Detection Electronic Circuit.

$$I_Z = \frac{P_Z}{V_Z}. \tag{1}$$

A small value of R2 results in a large diode current, which increases the power consumption of the FDEC. R2 value is chosen to be 39 kΩ based on various experimental iterations. Taking diode forward voltage (V_F) as 1.2 V and $V_{pv,\min}$ as 5 V, the diode current is deduced as

$$I_d = \frac{V_{\text{pv, min}} - V_F}{R_2} \tag{2}$$

$V_{pv,\min}$ is the minimum panel voltage under MPPT condition, and its value is assumed to be 5 V. R1 is taken as 1.5 kΩ for better Zener voltage regulation. The collector current of BJT is given as

$$I_C = \beta \times I_B \tag{3}$$

where, β is the current gain of BJT and I_B is the base current. The value of I_d is derived to be 100 µA, which is small enough to avoid any loading effect of the sensing circuit. For a given value of I_d, the β is estimated to be 0.1. With saturation voltage between collector and emitter ($V_{CE,\text{sat}}$) taken as 0.3 V and V_{bias} as 5 V, the collector resistance R3 can be calculated as

$$R_3 = \frac{V_{\text{bias}} - V_{CE,\text{sat}}}{I_C}. \tag{4}$$

4.3 Working of the Interfaced Circuit

From Fig. 4.1, it can be observed that under normal conditions (NF), the bypass diode of the associated SPV panel remains reverse biased, and the panel carries string current IS through it. In this condition, BJT operates under saturation mode, and collector voltage VC remains 'LOW'. During any faulty event (LL, LG, OC, or PSC), the bypass diode becomes forward biased, and the string current IS starts flowing through it, making zero forward current Id through the resistor R2 and thus forcing BJT to work under the cut-off region. Due to this, collector current IC becomes zero, and the collector voltage VC settles to biasing voltage and becomes 'HIGH'.

4.4 Advantages of the Fault Detection Method

The advantages of the proposed fault detection method are summarized as follows.

- The FDEC, including all components, costs as low as INR 10 (0.15). Given the compactness of the proposed circuit, a little space is required to interface it with the SPV panel near the junction box.
- The proposed FDEC consumes only 0.01% of panel-rated power. As observed from the experiments, it consumes 0.221 mA when connected to a $Vmpp = 18.1V$ and $Impp = 2.2A$ SPV panel.
- Zener diode works as a shielding device which can withstand high reverse current and avoid voltage spikes, thus providing isolation to the sensing circuit.
- It is notable that the proposed FDEC works optimally in the temperature range of $-55°$C to $+175°$C.

4.5 Conclusion

In wrapping up this chapter, the fault detection system represents a notable stride in fortifying the reliability and safety of electronic circuit-dependent systems. The intricate exploration of electronic circuit design, along with a thorough understanding of how the interfaced circuit operates, has illuminated a sturdy approach to fault detection. The highlighted advantages underscore the practicality and effectiveness of this method, emphasizing its potential to significantly reduce risks stemming from faults in various systems. Through this comprehensive coverage, the complexities of fault detection and the importance of proactive measures in fortifying system resilience is studied. Altogether, the outlined fault detection method stands as a promising avenue for reinforcing the trustworthiness and operational robustness of electronic systems.

CHAPTER 5

SIMULATION RESULTS

5.1 Introduction

The simulation results encapsulates a comprehensive exploration of fault detection, IoT replication, and ML modeling within the context of system simulations. This delves into the intricacies of replicating real-world scenarios in simulated environments, showcasing how fault detection electronic circuits (FDEC) function in detecting anomalies within Solar PV panels. Furthermore, this elucidates the challenges of simulating IoT-based systems using software platforms, outlining the process of emulating hardware functionalities through Python simulations. Additionally, this section navigates through the intricacies of ML modeling for fault classification, emphasizing the significance of dataset acquisition, preprocessing, and the evaluation of various ML algorithms to ascertain their efficiency in categorizing distinct fault types.

5.2 Implementation of FDEC Circuit in LTspice

The FDEC, or Fault Detection Electronic Circuit, serves as a crucial component connected to each Solar PV panel within an array. Its primary function is to generate a distinct output voltage based on the status of the panel. In the event of a faulty circuit, the FDEC yields an output voltage of 5 volts, whereas a healthy panel produces an output voltage of 0.1 volt.

The simulation process for the FDEC commenced by initially modeling a 300W Solar PV module within the LTspice environment. This module was then linked to the FDEC for comprehensive analysis of the circuit's behavior. Deliberate faults such as open circuit (OC), short circuit (SC), Line-to-Line (LL), and Line-to-Ground (LG) were intentionally induced into the Solar PV module. Subsequently, the terminal win-

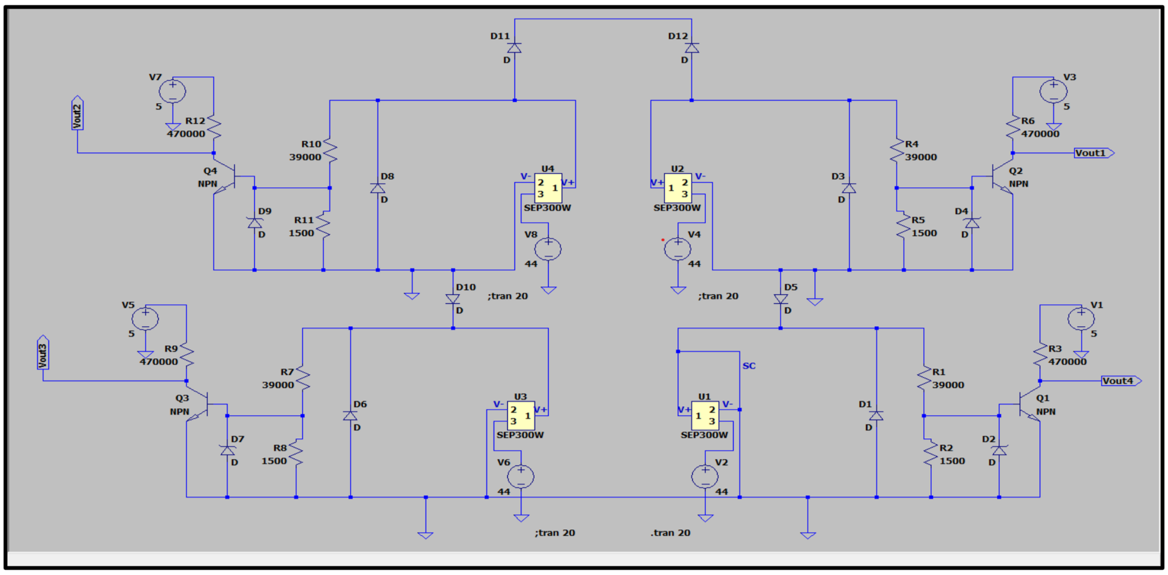


Figure 5.1: FDEC implemented on a series-parallel configuration of an SPV Array.

dow of LTspice displayed the expected output of 5 volts for the faulty conditions.

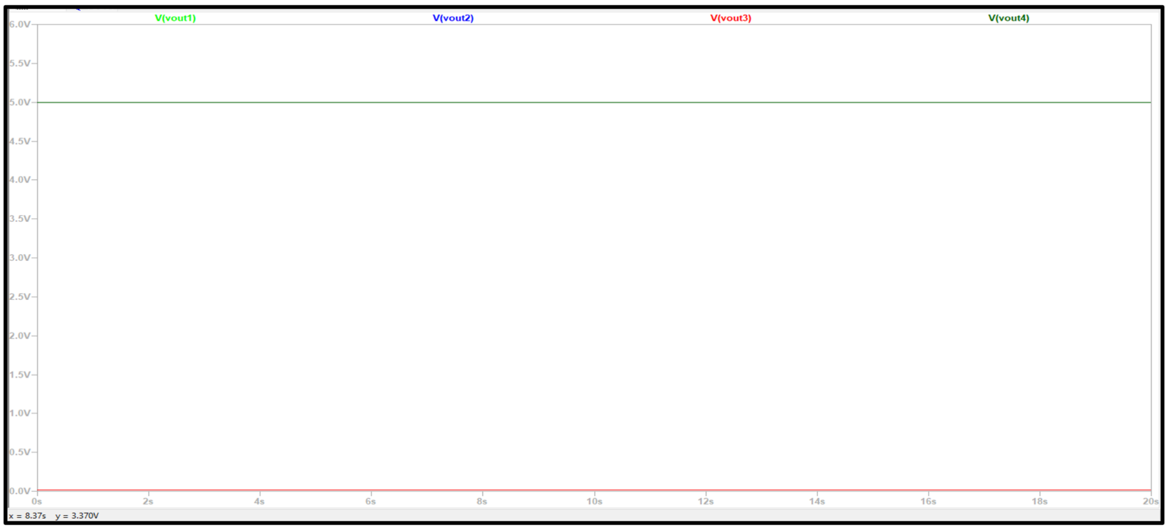
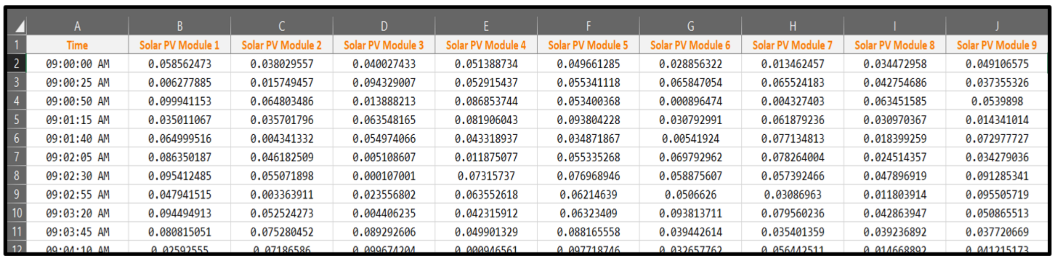


Figure 5.2: FDEC Simulation Result for Healthy, Faulty Panel.

The simulation effectively showcased how a series-parallel setup responds to faults, as illustrated in the accompanying figure. This method elucidated the manner in which the FDEC was simulated and its response to introduced faults within the system.

5.3 Implementation of IoT based FMDAS Technique

The implementation of an IoT-based Fault Monitoring and Data Acquisition System (FMDAS) involves replicating the behavior of hardware using a simulated platform,



	A	B	C	D	E	F	G	H	I	J
1	Time	Solar PV Module 1	Solar PV Module 2	Solar PV Module 3	Solar PV Module 4	Solar PV Module 5	Solar PV Module 6	Solar PV Module 7	Solar PV Module 8	Solar PV Module 9
2	09:00:00 AM	0.058562473	0.038029557	0.040027433	0.051388734	0.049661285	0.028856322	0.013462457	0.034472958	0.049106575
3	09:00:25 AM	0.006277885	0.015749457	0.094329007	0.052915437	0.055341118	0.065847054	0.065524183	0.042754686	0.037355326
4	09:00:50 AM	0.099941153	0.064803486	0.013888213	0.086853744	0.053400368	0.000896474	0.004327403	0.063451585	0.0539898
5	09:01:15 AM	0.035011067	0.035701796	0.063548165	0.081906043	0.093804228	0.030792991	0.061879236	0.030970367	0.014341014
6	09:01:40 AM	0.064999516	0.004341332	0.054974066	0.043318937	0.034871867	0.00541924	0.077134813	0.018399259	0.072977727
7	09:02:05 AM	0.086350187	0.046182509	0.005108607	0.011875077	0.055335268	0.069792962	0.078264004	0.024514357	0.034279036
8	09:02:30 AM	0.095412485	0.055071898	0.000107001	0.07315737	0.076968946	0.058875607	0.057392466	0.047896919	0.091285341
9	09:02:55 AM	0.047941515	0.003363911	0.023556802	0.063552618	0.06214639	0.0506626	0.03086963	0.011803914	0.095505719
10	09:03:20 AM	0.094494913	0.052524273	0.004406235	0.042315912	0.06323409	0.093813711	0.079560236	0.042863947	0.050865513
11	09:03:45 AM	0.080815051	0.075280452	0.089292606	0.049901329	0.088165558	0.039442614	0.035401359	0.039236892	0.037720669
12	09:04:10 AM	0.02592555	0.07186586	0.099674204	0.000946561	0.097718746	0.032657762	0.056442511	0.014668892	0.041215173

Figure 5.3: Data Collected against the Time from each FDEC of the SPV module.

primarily due to the limitations in simulating IoT in open-source software. To emulate the functionalities of IoT hardware, Python serves as the chosen platform for this simulation. The hardware aspect involves the utilization of an ESP-32 for data transmission, where real-time data is compared against a predetermined threshold temperature value. If the real-time data surpasses this threshold, indicating a faulty value, a message is generated either for display or transmission to a server.

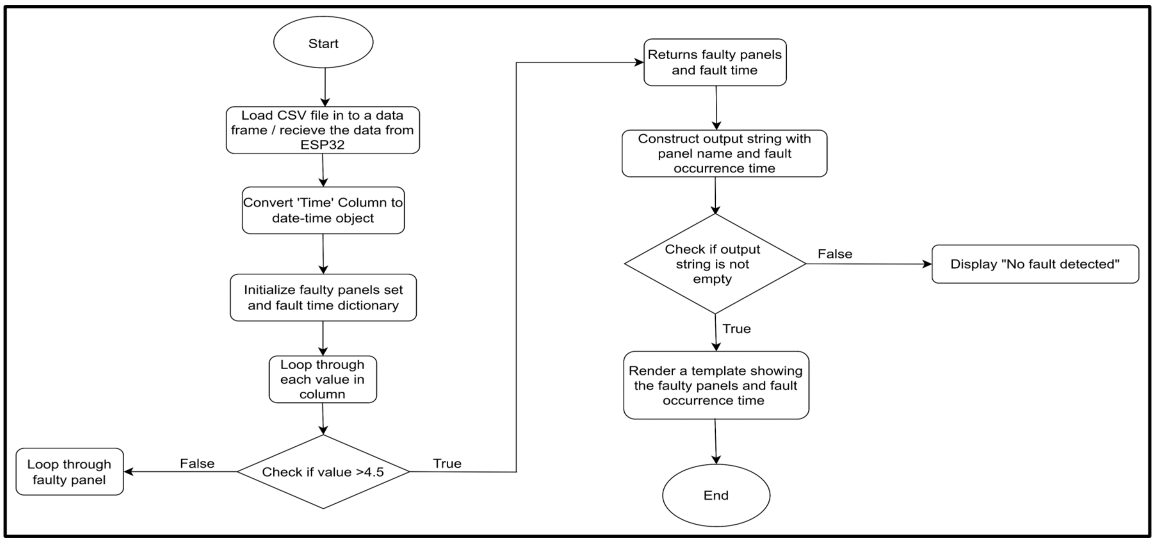
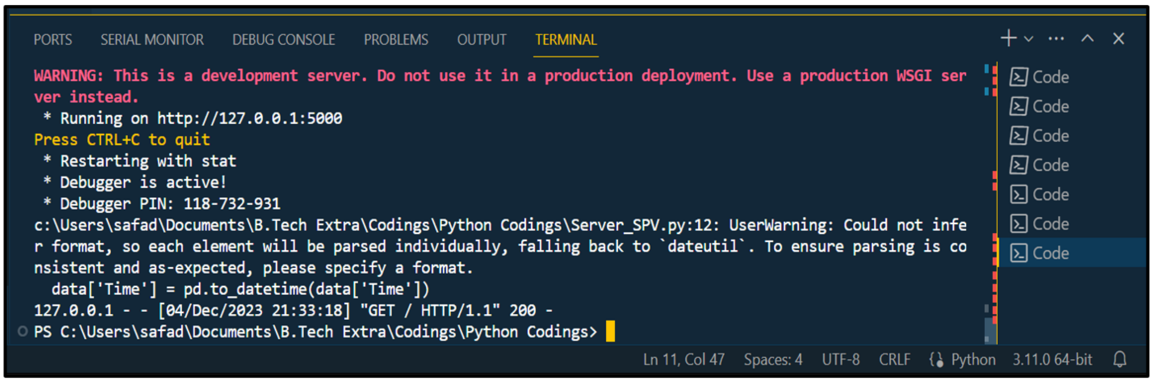

Start
Load CSV file in to a data frame / recieve the data from ESP32
Convert 'Time' Column to date-time object
Initialize faulty panels set and fault time dictionary
Loop through each value in column
Check if value >4.5
False
Loop through faulty panel
True
Returns faulty panels and fault time
Construct output string with panel name and fault occurrence time
Check if output string is not empty
False
Display "No fault detected"
True
Render a template showing the faulty panels and fault occurrence time
End

Figure 5.4: Python Program Flowchart.

However, as simulating IoT directly in open-source software is unfeasible, an alternative approach is adopted. Initially, data collection is executed by running a Solar PV (SPV) model within the MATLAB environment. The SPV model, simulated in SIMULINK, generates data crucial for replicating IoT behavior using Python. This data collection involves noting down information from nine SPV panels at intervals of 25 seconds, resulting in the acquisition of 1000 data points starting from 9:00 AM. The output data obtained represents the status of each SPV module as generated by the Fault Detection Electronic Circuit (FDEC).

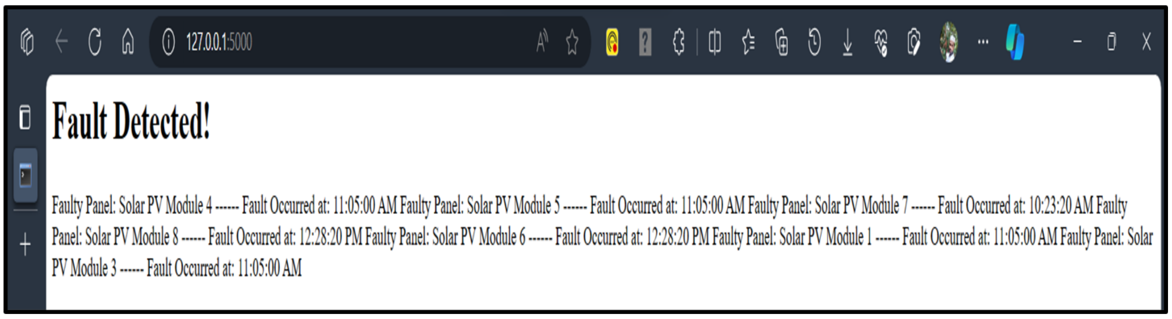
Subsequently, a Python code is employed for the analysis of the acquired data. This



```
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI ser
ver instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 118-732-931
c:\Users\safad\Documents\B.Tech Extra\Codings\Python Codings\Server_SPV.py:12: UserWarning: Could not infe
r format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is co
nsistent and as-expected, please specify a format.
  data['Time'] = pd.to_datetime(data['Time'])
127.0.0.1 - - [04/Dec/2023 21:33:18] "GET / HTTP/1.1" 200 -
PS C:\Users\safad\Documents\B.Tech Extra\Codings\Python Codings>
```

Figure 5.5: Terminal window upon running the python code.

code is programmed to detect faults within the SPV panels based on the collected data. When a fault is detected within a particular SPV panel, the Python code triggers a notification on a local server, indicating the occurrence of a fault at the specific time. This process mirrors the functionality of an IoT-based system, where the hardware would communicate fault occurrences to a central server or display.



Fault Detected!

Faulty Panel: Solar PV Module 4 ------ Fault Occurred at: 11:05:00 AM Faulty Panel: Solar PV Module 5 ------ Fault Occurred at: 11:05:00 AM Faulty Panel: Solar PV Module 7 ------ Fault Occurred at: 10:23:20 AM Faulty Panel: Solar PV Module 8 ------ Fault Occurred at: 12:28:20 PM Faulty Panel: Solar PV Module 6 ------ Fault Occurred at: 12:28:20 PM Faulty Panel: Solar PV Module 1 ------ Fault Occurred at: 11:05:00 AM Faulty Panel: Solar PV Module 3 ------ Fault Occurred at: 11:05:00 AM

Figure 5.6: Server window showing the status of each SPV module.

Through this simulated approach, the FMDAS, responsible for monitoring faults and acquiring data, successfully replicates the behaviors of an IoT hardware platform using software simulation. The Python-based system mimics the threshold-based comparison and fault detection akin to an actual IoT hardware setup, demonstrating fault monitoring and data acquisition functionalities in a simulated environment.

5.4 Data-Set Creation For ML Modelling

In the domain of ML modeling for fault classification, the initial steps involve detecting and categorizing faults, setting the groundwork for the subsequent task: fault classification. Before training a classification model, a dataset is essential. To acquire this dataset, researchers often explore platforms like Kaggle, IEEE, or other repositories containing relevant data. In cases where data isn't readily available, an alternative involves generating data using tools like the SIMULINK model. Running simulations within SIMULINK systematically generates data, with the data quantity increasing with longer model runs. These simulations deliberately encompass various fault scenarios such as short circuit (SC), open circuit (OC), Line to Line (LL), Line to Ground (LG), Partial Shading Condition (PSC), and more. Each occurrence of a fault generates corresponding data points, eventually stored in a CSV file for further analysis.

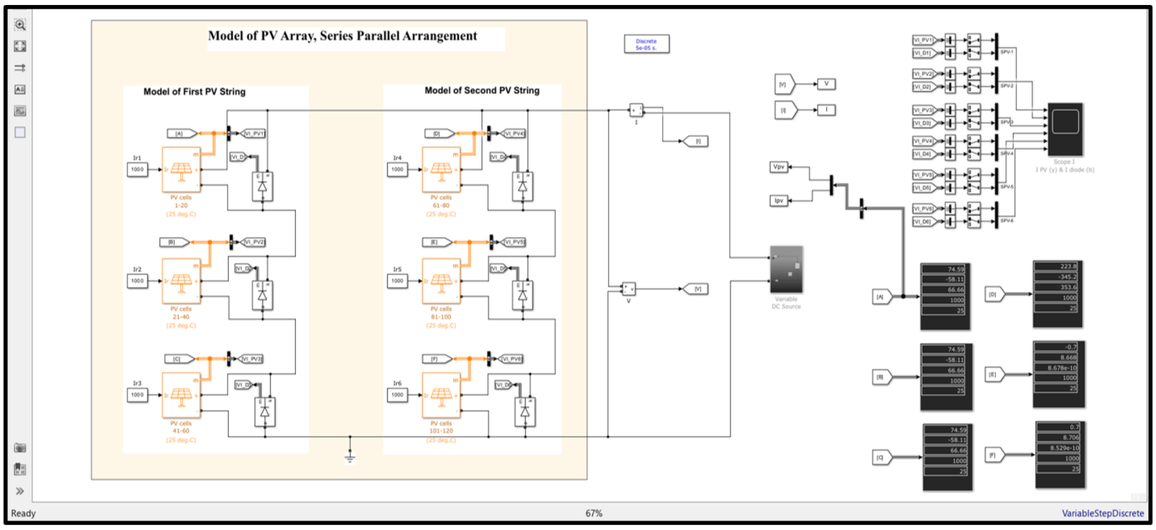
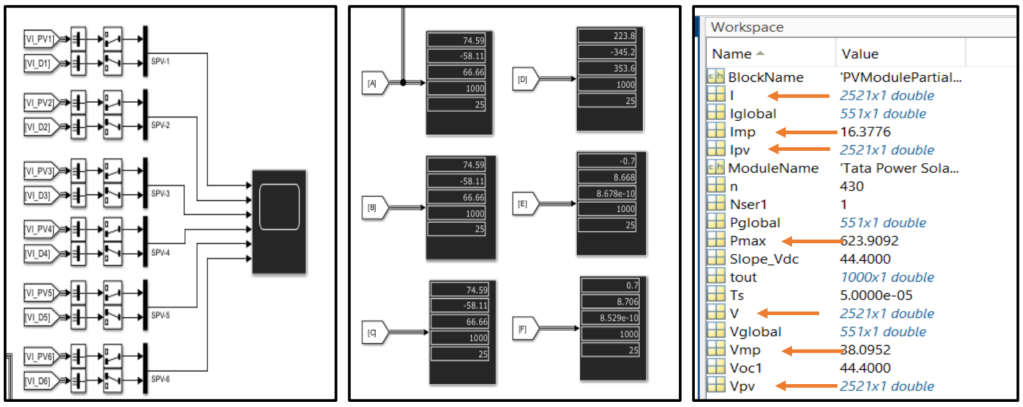


Figure 5.7: MATLAB SPV Array Simulation.

Preprocessing is a critical phase in preparing collected data for effective utilization in machine learning models. Cleaning the dataset involves identifying and rectifying any inconsistencies, errors, or outliers present in the data. This step ensures that the dataset is free from irregularities that could potentially distort the model's learning process. For instance, it might involve dealing with duplicate entries, correcting formatting issues, or resolving discrepancies in data entries that could mislead the model during training.

Handling missing values is another vital aspect of preprocessing. Datasets often contain missing or incomplete information, which can hinder the model's performance. Strategies like imputation (filling missing values with reasonable approximations) or removing rows or columns with excessive missing data are commonly employed to address this issue. Ensuring that the dataset is complete and consistent is crucial for training accurate and reliable models.

Scaling the data for normalization purposes involves bringing different features to



Workspace

Name	Value
BlockName	'PVModulePartial...
I	2521x1 double
Iglobal	551x1 double
Imp	16.3776
Ipv	2521x1 double
ModuleName	'Tata Power Sola...
n	430
Nser1	1
Pglobal	551x1 double
Pmax	623.9092
Slope_Vdc	44.4000
tout	1000x1 double
Ts	5.0000e-05
V	2521x1 double
Vglobal	551x1 double
Vmp	38.0952
Voc1	44.4000
Vpv	2521x1 double

Figure 5.8: Data Collection from Simulink platform.

a similar scale. In many machine learning algorithms, features with larger numerical scales might dominate the learning process, leading to biased model outcomes. Normalization techniques like scaling features to a similar range (e.g., between 0 and 1) or standardizing them (transforming to have a mean of 0 and a standard deviation of 1) help prevent such biases. This step ensures that each feature contributes proportionally to the model's learning without being influenced by its original scale. Ultimately, the goal of preprocessing is to refine the dataset, making it more conducive for machine learning algorithms to learn patterns and relationships within the data accurately.

The ML training process is integral and results in a confusion matrix, acting as a metric for the model's performance. Initially, the dataset is partitioned into segments, typically divided into training and testing sets, by the ML algorithms themselves. This partitioning is crucial for assessing the model's efficiency during the training process, allowing validation and evaluation of its performance on unseen data.

The classification model's efficacy is determined through this process, where the dataset undergoes training, validation, and testing phases. It's a critical stage in ensuring the model's ability to accurately classify faults based on the patterns and features extracted from the dataset.

5.5 Machine Learning and Model Comparison

Detecting faulty panels involves using machine learning (ML) and deep learning (DL) models. Initially, it's crucial to gather data on panel characteristics and potential faults. After collecting the data, these models are utilized to classify functional and faulty panels by analyzing patterns and features within the acquired information. Metrics like performance measures and confusion matrices evaluate how accurately these models identify defective units. The choice between ML and DL models depends on factors such as data quality and quantity. Models can range from traditional ML algorithms like Random Forests or Support Vector Machines to more complex DL architectures like Convolutional Neural Networks or Recurrent Neural Networks. The effectiveness of these models depends on data intricacies and the desired accuracy for fault detection.

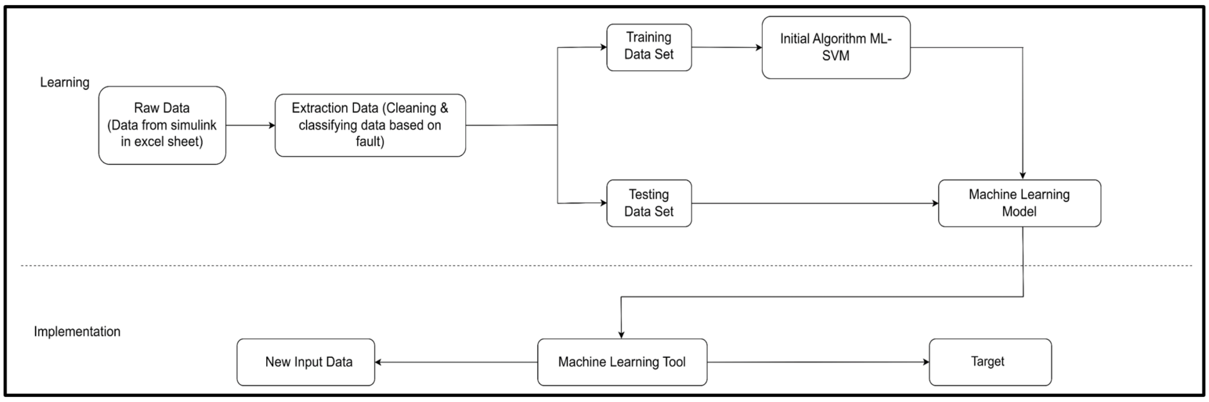


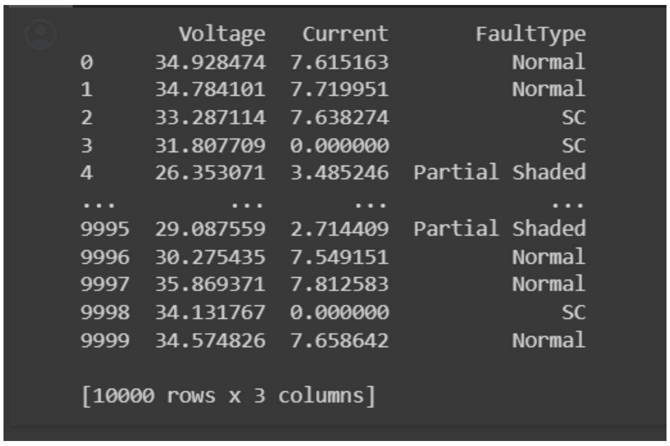
Figure 5.9: Flowchart of the ML modelling.

Algorithm 1: Importing Data from Excel using Pandas

```
1: Input: Excel file 'pvdataset.xlsx'
2: Output: DataFrame 'imported_data'
3: function ImportExcelData
4:     Import the Pandas library as 'pd'
5:     Define excel_file_path as the path to the Excel file 'pvdataset.xlsx'
6:     Read the Excel file specified by excel_file_path into a DataFrame named
       'imported_data' using pd.read_excel()
7:     Print the first few rows of 'imported_data' using the print() function
8: end function
```

When plotting histograms of voltage and current data from a DataFrame, it's essential to understand their respective distributions. Histograms are effective visualizations that showcase the frequency distribution of values within a dataset.

The histogram of voltage provides insight into the distribution of voltage values captured in the dataset. It illustrates the spread and frequency of different voltage levels, showcasing how often each voltage value occurs within the dataset. Understanding the voltage distribution is crucial in various fields, such as electrical engineering or physics, as it helps identify common voltage ranges or outliers.



```
        Voltage   Current       FaultType
0     34.928474  7.615163          Normal
1     34.784101  7.719951          Normal
2     33.287114  7.638274              SC
3     31.807709  0.000000              SC
4     26.353071  3.485246  Partial Shaded
...         ...       ...             ...
9995  29.087559  2.714409  Partial Shaded
9996  30.275435  7.549151          Normal
9997  35.869371  7.812583          Normal
9998  34.131767  0.000000              SC
9999  34.574826  7.658642          Normal

[10000 rows x 3 columns]
```

Figure 5.10: Data Set importing.

Similarly, the histogram of current displays the frequency distribution of current values present in the dataset. It reveals patterns in current levels, highlighting the occurrence and variability of different current values. Analyzing the current distribution aids in comprehending the typical current behavior or identifying unusual fluctuations within the dataset.

In combination, these histograms offer a comprehensive view of the voltage and current characteristics within the dataset, enabling insights into their respective distributions, central tendencies, and potential outliers or irregularities. These visualizations serve as foundational tools for understanding the underlying patterns and behaviors of voltage and current data.

The utilization of scatter plots holds significant relevance within the context of visualizing relationships between voltage and current concerning different fault types in a power system dataset. Scatter plots provide a powerful graphical representation, enabling the simultaneous display of voltage and current values for distinct fault types. By visualizing these relationships, patterns or trends in how voltage and current interact in various fault scenarios become discernible. This graphical insight aids engineers and analysts in identifying potential correlations or distinctions between fault types based on the observed voltage-current distributions, thereby facilitating a deeper understanding of the system behavior under different fault conditions.

Support Vector Machine - Logistic Regression (SVM-LR) blends the strengths of SVM's robustness in handling complex data distributions with Logistic Regression's simplicity and interpretability. It combines the margin-maximizing nature of SVMs with LR's probabilistic framework, yielding a hybrid model that balances non-linear decision boundaries while providing probabilistic predictions. SVM-LR efficiently han-

Algorithm 2: Plotting Histograms of Voltage and Current

```
Require: imported_data - DataFrame containing voltage and current data
 1: Import the Matplotlib library as 'plt'
 2: Plot a histogram of the 'Voltage' column from the imported data using
    plt.hist()
 3: Set the number of bins to 20
 4: Set the color of the bars to blue with an alpha value of 0.7 for transparency
 5: Set the label for the x-axis as 'Voltage' using plt.xlabel()
 6: Set the label for the y-axis as 'Frequency' using plt.ylabel()
 7: Set the title of the histogram as 'Histogram of Voltage' using plt.title()
 8: Display the histogram using plt.show()
 9: Plot a histogram of the 'Current' column from the imported data using
    plt.hist()
10: Set the number of bins to 20
11: Set the color of the bars to blue with an alpha value of 0.7 for transparency
12: Set the label for the x-axis as 'Current' using plt.xlabel()
13: Set the label for the y-axis as 'Frequency' using plt.ylabel()
14: Set the title of the histogram as 'Histogram of Current' using plt.title()
15: Display the histogram using plt.show()
```

Algorithm 3: Scatter Plot of Voltage vs. Current by Fault Type

```
Require: imported_data - DataFrame containing voltage, current, and fault type
    data
Ensure: Scatter plot showing voltage vs. current, colored by fault type
 1: Import the Seaborn library as 'sns'
 2: Import the Matplotlib library as 'plt'
 3: Set the style for Seaborn as 'whitegrid' using sns.set()
 4: Create a figure with a size of 10x10 using plt.figure(figsize=(10, 10))
 5: Create a scatter plot of 'Voltage' vs. 'Current' with different colors for each
    'FaultType'
 6: Use sns.scatterplot() and specify x='Voltage', y='Current',
    hue='FaultType', data=imported_data, palette='Set2', alpha=0.9
 7: Set the label for the x-axis as 'Voltage' using plt.xlabel()
 8: Set the label for the y-axis as 'Current' using plt.ylabel()
 9: Set the title of the plot as 'Voltage vs. Current by Fault Type' using plt.title()
10: Display the legend with the title 'Fault Type' at the upper center using
    plt.legend(title='Fault Type', loc='upper center')
11: Display the plot using plt.show()
```

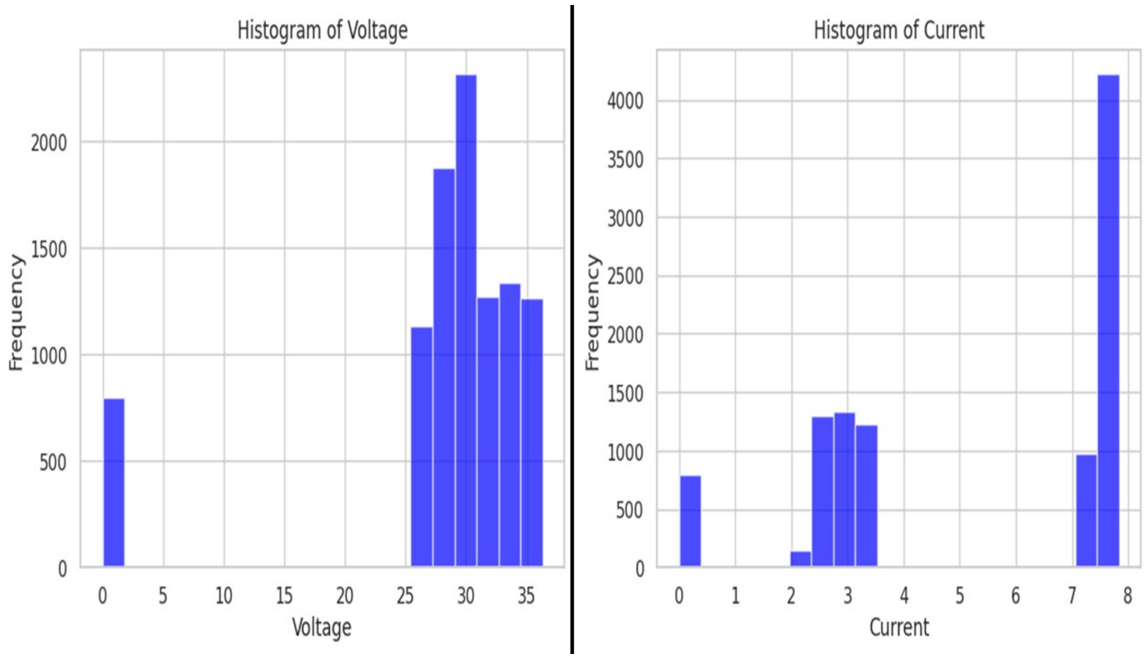


Figure 5.11: Histogram of the Voltage and Current.

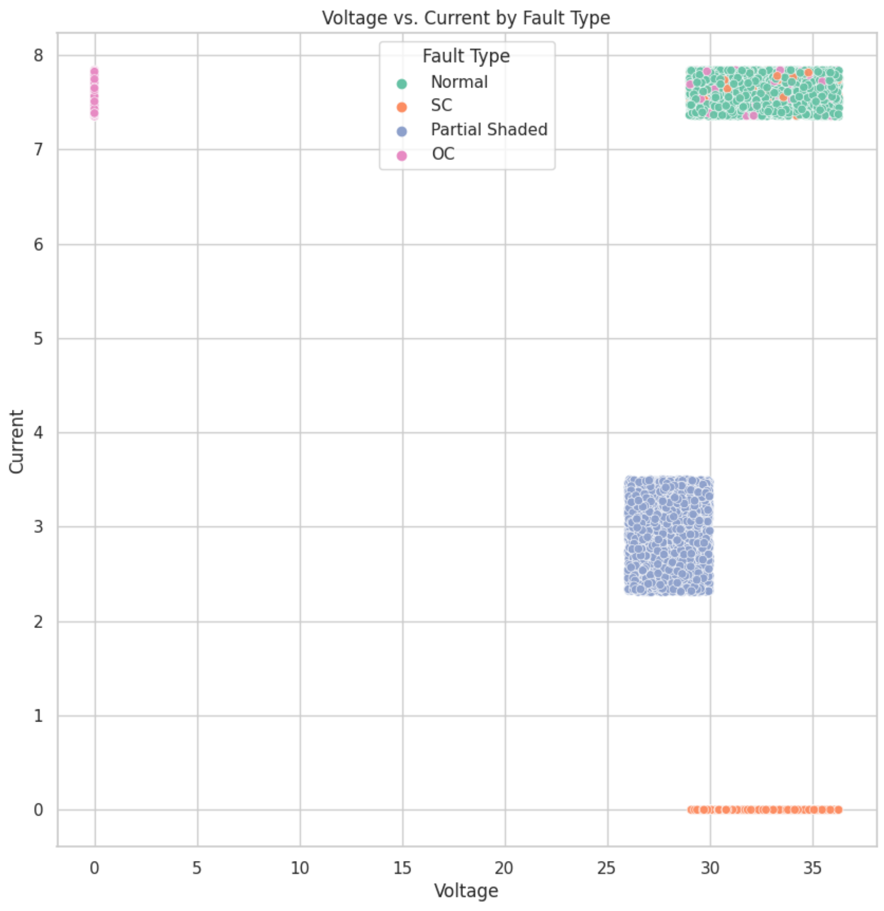


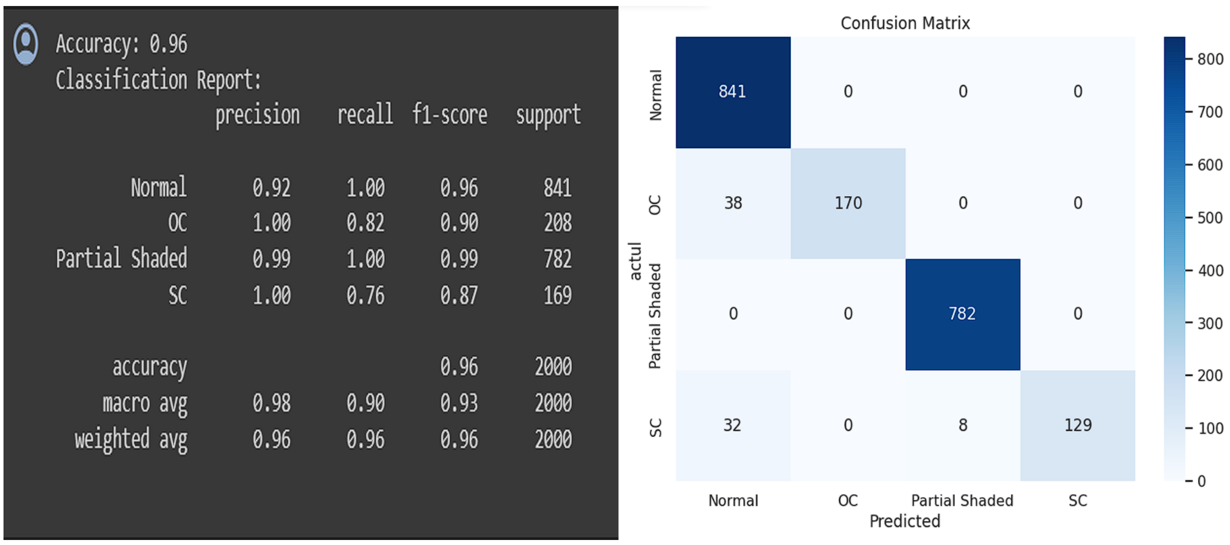
Figure 5.12: Voltage vs Current by plot type

dles both linearly separable and non-linearly separable data by incorporating the kernel trick for non-linear transformations, ensuring adaptability to diverse datasets. This fusion empowers practitioners with a versatile tool that maintains SVM's efficacy in handling complex data while leveraging LR's interpretability, making it valuable in various classification scenarios.

Algorithm 4: Training SVM Model for FaultType Prediction and Confusion Matrix Visualization

```
Require: Dataset 'pvdataset.xlsx'
Ensure: Trained SVM model, Evaluation metrics (Accuracy, Classification Report),
    Heatmap visualization of the confusion matrix
 1: Import necessary libraries: pandas, train_test_split, SVC, accuracy_score,
    classification_report, StandardScaler
 2: Load the dataset 'pvdataset.xlsx' into a DataFrame 'data' using pd.read_excel()
 3: Separate the features (X) and the target variable (y)
 4:     X contains 'Voltage' and 'Current' columns
 5:     y contains the 'FaultType' column
 6: Split the dataset into training and testing sets using train_test_split()
 7:     Set aside 20
 8: Standardize the features using StandardScaler()
 9:     Scale the training and testing features separately
10: Train an SVM model using SVC()
11:     Use a linear kernel and set the regularization parameter C=1 (adjustable)
12: Make predictions on the test set using svm_model.predict()
13: Evaluate the model's performance
14:     Calculate accuracy using accuracy_score() between y_test and y_pred
15:     Generate a classification report using classification_report()
16: Display the accuracy and classification report
17:     Print the accuracy score rounded to two decimal places
18:     Print the classification report showing precision, recall, and F1-score for each
    class
19: Create a new figure with a size of 8x6 using plt.figure(figsize=(8, 6))
20: Generate a heatmap of the confusion matrix using sns.heatmap()
21:     Use the 'conf_matrix' as input data
22:     Enable annotations on the heatmap by setting annot=True
23:     Set the format of the annotations to 'd' (decimal)
24:     Use the 'Blues' colormap to represent the data
25:     Label the x-axis with 'Predicted' using plt.xlabel()
26:     Label the y-axis with 'Actual' using plt.ylabel()
27:     Set the title of the heatmap as 'Confusion Matrix' using plt.title()
28: Display the heatmap using plt.show()
```

The decision function scores in Support Vector Machines (SVM) are numerical values that indicate the distance of data points from the decision boundary established by the SVM algorithm. In binary classification, these scores help classify data points: positive scores suggest one class, while negative scores suggest the other. Larger abso-



```
Accuracy: 0.96
Classification Report:
                precision    recall  f1-score   support

        Normal       0.92      1.00      0.96       841
            OC       1.00      0.82      0.90       208
Partial Shaded       0.99      1.00      0.99       782
            SC       1.00      0.76      0.87       169

      accuracy                           0.96      2000
     macro avg       0.98      0.90      0.93      2000
  weighted avg       0.96      0.96      0.96      2000
```

Figure 5.13: Report of Support Vector Machine-Logistic Regression.

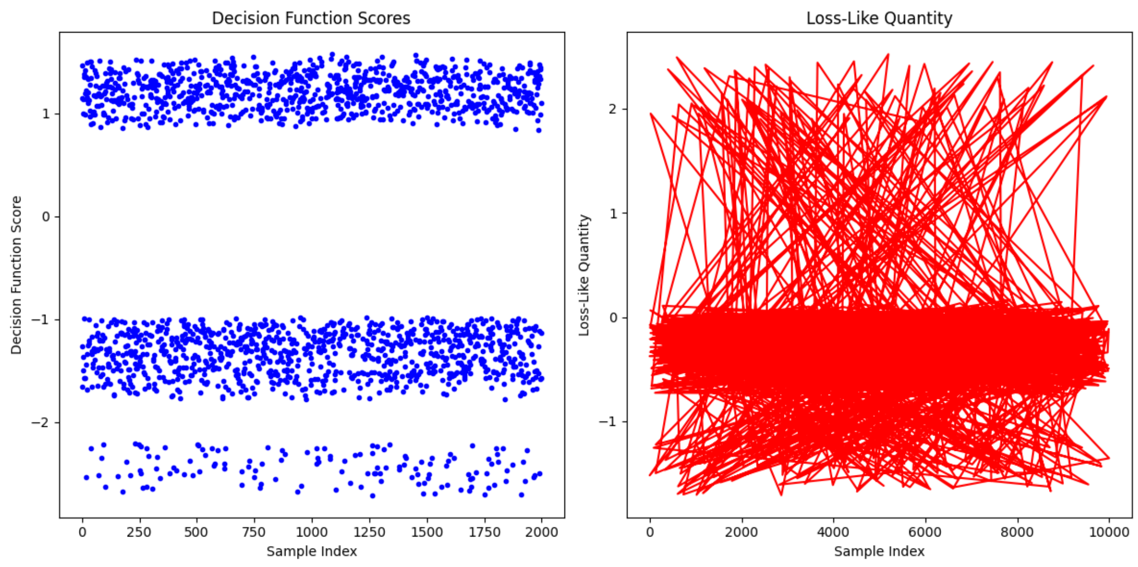


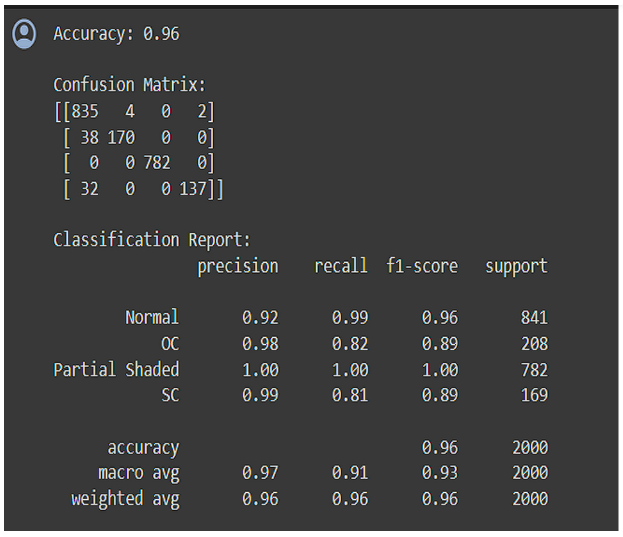
Figure 5.14: Decision Function Scores and Loss-Like Quantity of SVM-LR.

lute values of these scores typically denote greater confidence in the classification.

The ”loss-like” quantity, calculated using hinge-loss formula 1−(decision function scores× true labels), serves as a measure of how well the SVM model is performing. It signifies the margin or distance of a data point from the decision boundary. Lower values of this loss metric indicate that the data points are correctly classified or are closer to the decision boundary, implying a more confident and accurate model. The hinge loss penalizes misclassifications, contributing to the optimization process of the SVM algorithm by aiming to minimize this loss to achieve better separability and classification accuracy.

Algorithm 5: Visualizing Decision Function Scores and Loss-Like Quantity

```
Require: Dataset 'pvdataset.xlsx'
Ensure: Visualization of decision function scores and loss-like quantity
 1: Import necessary libraries: pandas, train_test_split, SVC, StandardScaler, numpy,
    matplotlib
 2: Load the dataset 'pvdataset.xlsx' into a DataFrame 'data' using pd.read_excel()
 3: Separate the features (X) and the target variable (y)
 4:     X contains 'Voltage' and 'Current' columns
 5:     y is mapped to suitable labels for SVM (-1 for fault types other than
    'Normal', 1 for 'Normal')
 6: Split the dataset into training and testing sets using train_test_split()
 7:     Set aside 20
 8: Standardize the features using StandardScaler()
 9:     Scale the training and testing features separately
10: Train an SVM model using SVC()
11:     Use a linear kernel, set the regularization parameter C=1, enable probability
    estimates
12: Obtain decision function scores on the test set using
    svm_model.decision_function()
13: Calculate loss-like quantity using hinge loss:
    $loss\_values = 1 - decision\_function\_scores \times y\_test$
14: Plot the decision function scores and loss-like quantity
15:     Create a figure with a size of 12x6 using plt.figure(figsize=(12, 6))
16:     Subplot 1: Scatter plot of decision function scores
17:     Subplot 2: Line plot of loss-like quantity
18:     Set appropriate labels and titles for both plots
19:     Display the plots using plt.tight_layout() and plt.show()
```



```
Accuracy: 0.96

Confusion Matrix:
[[835   4   0   2]
 [ 38 170   0   0]
 [  0   0 782   0]
 [ 32   0   0 137]]

Classification Report:
                precision    recall  f1-score   support

       Normal       0.92      0.99      0.96       841
           OC       0.98      0.82      0.89       208
Partial Shaded      1.00      1.00      1.00       782
           SC       0.99      0.81      0.89       169

     accuracy                           0.96      2000
    macro avg       0.97      0.91      0.93      2000
 weighted avg       0.96      0.96      0.96      2000
```

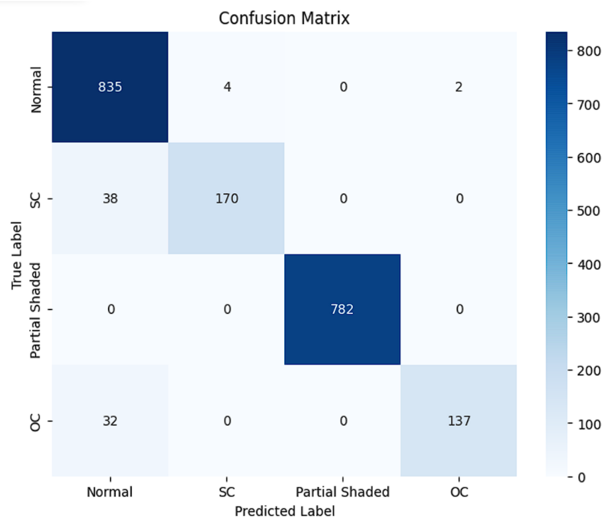


Figure 5.15: Report of Random Forest Classifier.

The Random Forest Classifier is a machine learning method designed for classification tasks, relying on an ensemble of decision trees. During training, it builds multiple trees using different subsets of both features and the training data. This diversity among trees helps enhance the model's reliability.

When it comes to making predictions, the Random Forest combines the outputs of each individual tree to determine the final prediction. For classification, it selects the class that gathers the most votes from the trees, while for regression tasks, it averages the outputs. By leveraging this collective decision-making approach, Random Forests often yield better accuracy and resilience against overfitting, allowing them to generalize well to new, unseen data. Another advantage lies in their ability to assess feature importance, making them widely applicable across various domains, including finance, healthcare, natural language processing, and image recognition.

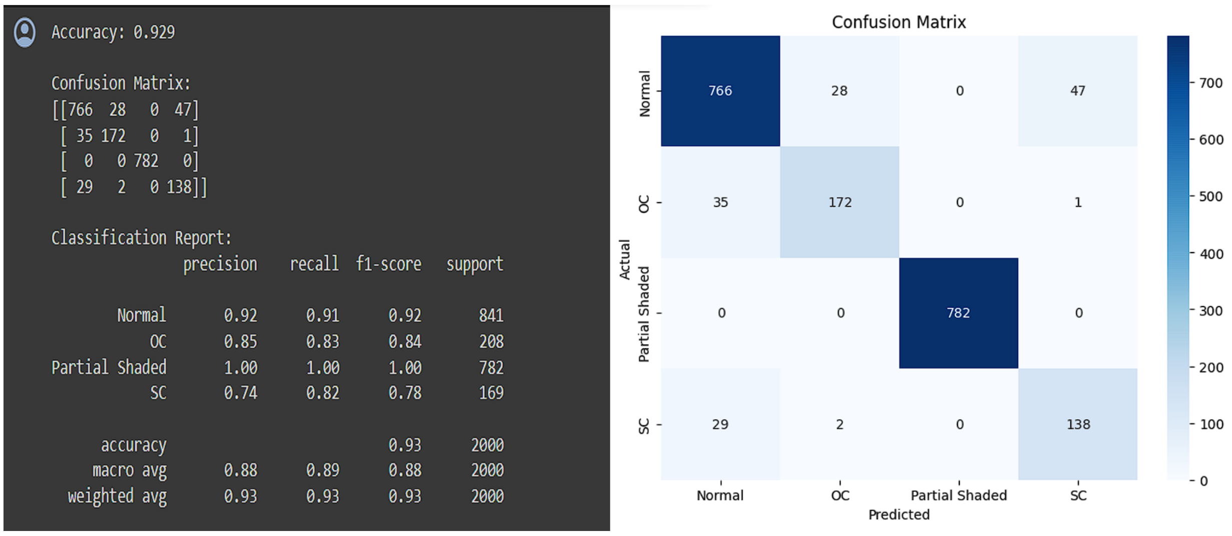
The Decision Tree Classifier is a machine learning method used for classification and regression tasks. It works by segmenting the feature space into distinct sections, making sequential decisions based on feature values to ultimately make a prediction. Each node in the tree signifies a decision based on a particular feature, and each leaf node represents the final prediction or class label.

This classifier functions by repeatedly dividing the dataset into subsets based on features that best separate the data according to specific criteria (such as Gini impurity or information gain). It's recognized for its tree-like structure, which is easy to interpret, enabling a clear understanding of how decisions are made. However, if not properly controlled, Decision Trees can overfit the data, particularly when creating deep trees

Algorithm 6: Training and Evaluating Random Forest Classifier

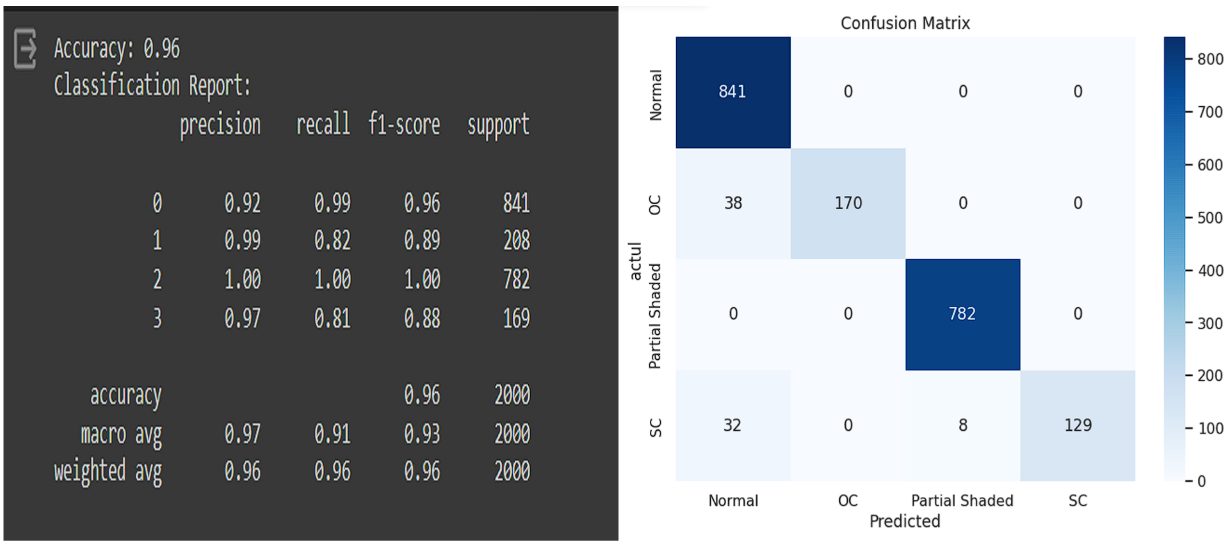
```
Require: Dataset 'pvdataset.xlsx'
Ensure: Classifier performance metrics (Accuracy, Confusion Matrix, Classification
    Report), Visualization of Confusion Matrix
 1: Import necessary libraries: pandas, matplotlib.pyplot, seaborn, train_test_split,
    RandomForestClassifier, accuracy_score, classification_report, confusion_matrix
 2: Load the synthetic dataset 'pvdataset.xlsx' into a DataFrame 'synthetic_data'
    using pd.read_excel()
 3: Separate the features (X) and the target variable (y)
 4:     X contains 'Voltage' and 'Current' columns
 5:     y contains the 'FaultType' column
 6: Split the dataset into training and testing sets using train_test_split()
 7:     Set aside 20
 8: Initialize the Random Forest classifier with n_estimators=100 and
    random_state=42
 9: Train the classifier using rf_classifier.fit()
10: Make predictions on the test set using rf_classifier.predict()
11: Evaluate the classifier's performance
12:     Calculate accuracy using accuracy_score() between y_test and y_pred
13:     Generate a confusion matrix using confusion_matrix()
14:     Generate a classification report using classification_report()
15: Display the results
16:     Print the accuracy score rounded to two decimal places
17:     Print the confusion matrix
18:     Print the classification report showing precision, recall, and F1-score for each
    class
19: Visualize the confusion matrix using seaborn
20:     Create a figure with a size of 8x6 using plt.figure(figsize=(8, 6))
21:     Generate a heatmap of the confusion matrix using sns.heatmap()
22:     Show the true and predicted labels on x and y axes, respectively
23:     Display the heatmap using plt.show()
```

that might capture noise in the dataset. Methods like pruning or employing ensemble techniques such as Random Forests help address these concerns, enhancing the classifier's ability to generalize and handle diverse datasets.



```
Accuracy: 0.929

Confusion Matrix:
[[766  28   0  47]
 [ 35 172   0   1]
 [  0   0 782   0]
 [ 29   2   0 138]]

Classification Report:
               precision    recall  f1-score   support

       Normal       0.92      0.91      0.92       841
           OC       0.85      0.83      0.84       208
Partial Shaded       1.00      1.00      1.00       782
           SC       0.74      0.82      0.78       169

     accuracy                           0.93      2000
    macro avg       0.88      0.89      0.88      2000
 weighted avg       0.93      0.93      0.93      2000
```

Figure 5.16: Report of Decision Tree Classifier.



```
Accuracy: 0.96
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.99      0.96       841
           1       0.99      0.82      0.89       208
           2       1.00      1.00      1.00       782
           3       0.97      0.81      0.88       169

    accuracy                           0.96      2000
   macro avg       0.97      0.91      0.93      2000
weighted avg       0.96      0.96      0.96      2000
```

Figure 5.17: Report of Voting Classifier.

The Voting Classifier is an ensemble learning technique that combines multiple individual machine learning models to generate predictions. It aggregates predictions from various base models, which can be classifiers or regressors, and outputs the most common prediction for classification tasks or the average prediction for regression tasks. This approach is founded on the idea that combining diverse models often yields more accurate and robust predictions compared to relying on a single model.

Algorithm 7: Training and Evaluating Decision Tree Classifier

```
Require: Dataset 'pvdataset.xlsx'
Ensure: Classifier performance metrics (Accuracy, Confusion Matrix, Classification
    Report), Visualization of Confusion Matrix
 1: Import necessary libraries: pandas, seaborn, matplotlib.pyplot, train_test_split,
    DecisionTreeClassifier, accuracy_score, classification_report, confusion_matrix
 2: Load the synthetic dataset 'pvdataset.xlsx' into a DataFrame 'dataset' using
    pd.read_excel()
 3: Separate the features (X) and the target variable (y)
 4:     X contains 'Voltage' and 'Current' columns
 5:     y contains the 'FaultType' column
 6: Split the dataset into training and testing sets using train_test_split()
 7:     Set aside 20
 8: Initialize the Decision Tree classifier with default parameters and
    random_state=42
 9: Train the classifier on the training set using clf.fit()
10: Make predictions on the test set using clf.predict()
11: Evaluate the classifier's performance
12:     Calculate accuracy using accuracy_score() between y_test and y_pred
13:     Generate a confusion matrix using confusion_matrix()
14:     Generate a classification report using classification_report()
15: Display the results
16:     Print the accuracy score
17:     Print the confusion matrix
18:     Print the classification report showing precision, recall, and F1-score for each
    class
19: Visualize the confusion matrix using seaborn and matplotlib
20:     Create a figure with a size of 8x6 using plt.figure(figsize=(8, 6))
21:     Generate a heatmap of the confusion matrix using sns.heatmap()
22:     Show the true and predicted labels on x and y axes, respectively
23:     Display the heatmap using plt.show()
```

There exist two primary types of Voting Classifiers: hard and soft voting. In hard voting, the final prediction is determined by a majority vote among the individual models. In contrast, soft voting considers the confidence or probability estimates from each model, averaging them to derive the final prediction. The Voting Classifier is effective when the base models exhibit different strengths and weaknesses or when they are trained on different subsets of data. This diversity allows for a more comprehensive analysis of the dataset, potentially leading to improved generalization when making predictions on new, unseen data.

Algorithm 8: Training and Evaluating Voting Classifier

```
Require: Dataset 'pvdataset.xlsx'
Ensure: Classifier performance metrics (Accuracy, Classification Report),
    Visualization of Confusion Matrix
 1: Import necessary libraries: pandas, sklearn.model_selection, sklearn.ensemble,
    sklearn.tree, sklearn.svm, sklearn.metrics, seaborn, matplotlib.pyplot
 2: Load the synthetic dataset 'pvdataset.xlsx' into a DataFrame 'synthetic_data'
    using pd.read_excel()
 3: Convert 'FaultType' to numerical labels using LabelEncoder()
 4: Separate the features (X) and the target variable (y)
 5:     X contains 'Voltage' and 'Current' columns
 6:     y contains the numerical labels for 'FaultType'
 7: Split the data into training and testing sets using train_test_split()
 8:     Set aside 20
 9: Define the classifiers: RandomForestClassifier, DecisionTreeClassifier, SVC
10: Create a VotingClassifier using RandomForestClassifier, DecisionTreeClassifier,
    SVC, with soft voting
11: Train the VotingClassifier using voting_classifier.fit()
12: Make predictions on the test set using voting_classifier.predict()
13: Evaluate the classifier's performance
14:     Calculate accuracy using accuracy_score() between y_test and y_pred
15:     Generate a classification report using classification_report()
16: Display the results
17:     Print the accuracy score rounded to two decimal places
18:     Print the classification report showing precision, recall, and F1-score for each
    class
19: Visualize the confusion matrix using seaborn and matplotlib
20:     Create a figure with a size of 8x6 using plt.figure(figsize=(8, 6))
21:     Generate a heatmap of the confusion matrix using sns.heatmap()
22:     Show the true and predicted labels on x and y axes, respectively
23:     Display the heatmap using plt.show()
```

Presently, several ML algorithms, such as Random Forest Classifier, Decision Tree, and SVM-LR, are being implemented and evaluated. The objective is to compare their efficiencies in handling different fault scenarios. Each fault type may exhibit varied results when subjected to different ML algorithms. Therefore, the ongoing assessment focuses on determining the most suitable ML model for each specific fault category, identifying the algorithm that yields the most efficient and accurate fault classification results. This comparative analysis aids in selecting the optimal ML approach tailored to the distinctive characteristics of each fault type.

The assessment of ML algorithms for fault classification heavily relies on performance metrics to pinpoint the best-suited model for each fault category. Evaluating precision, recall, accuracy, and the F1 score for each ML algorithm used on the dataset provides critical insights. These metrics shed light on how effectively the model identifies faults while minimizing the occurrence of false positives and negatives.

Additionally, the selection of an ML algorithm factors in computational complexity and scalability essential for real-time fault classification. While some algorithms may boast higher accuracy rates, they could also be computationally intensive, making them less practical for immediate deployment in real-world scenarios. Striking a balance between accuracy and computational efficiency is pivotal in choosing the most viable model for integration into fault monitoring systems. Hence, beyond comparing efficiency across fault types, the assessment considers the computational demands and scalability of ML algorithms, especially concerning real-time applications within fault detection frameworks.

5.6 Conclusion

Concluding this segment on simulation outcomes, the FDEC (Fault Detection), IoT-based FMDAS (Fault Monitoring and Detection System), and ML (Machine Learning) methodologies showcased substantial progress in fault analysis within electronic systems. Their implementation signifies a leap forward in enabling robust fault detection, localization, and classification. The integration of these methods fosters a holistic approach to fault management, leveraging data-driven insights for predictive analysis and model comparison. The creation of dedicated datasets for machine learning modeling serves as a pivotal foundation for enhancing fault prediction accuracy and refining classification models. The combined capabilities of these techniques highlight a promising trajectory toward bolstering fault management strategies, thereby fortifying system reliability through proactive fault detection and preemptive mitigation measures.

CHAPTER 6

CONCLUSION

In the exploration of fault detection and system analysis, a spectrum of tools and platforms has been strategically employed for distinct purposes. The Fault Detection Electronic Circuit (FDEC) was meticulously crafted and executed within LTspice, showcasing its proficiency in identifying faults within Solar PV panels. MATLAB/SIMULINK emerged as a cornerstone tool, serving a dual role: initially for data collection from the Solar PV modules and subsequently for an in-depth study of fault characteristics. The insights gleaned from MATLAB/SIMULINK fueled the understanding of fault behaviors, crucial for subsequent analysis. Further advancing the analytical landscape, the Fault Monitoring and Data Acquisition System (FMDAS) was crafted using Python, leveraging the collected data to emulate IoT behaviors. This technique facilitated the real-time relay of Solar PV module status to a dedicated server, enabling continuous and remote monitoring. The realm of fault classification delved into the realm of machine learning, employing ML algorithms, including Support Vector Machines (SVM). These algorithms were instrumental in classifying faults, with a detailed comparative study performed to ascertain their respective efficiencies. This integration of ML algorithms into fault classification underscored their relevance and efficacy within this multifaceted exploration of fault detection and system analysis within Solar PV systems. Each tool and technique, from fault identification and data collection to fault classification and comparative algorithmic analysis, played an integral role in establishing a comprehensive framework for understanding, detecting, and managing faults in Solar PV systems.

REFERENCES

[1] M. K. Alam, F. Khan, J. Johnson and J. Flicker, ”A Comprehensive Review of Catastrophic Faults in PV Arrays: Types, Detection, and Mitigation Techniques,” in IEEE Journal of Photovoltaics, vol. 5, no. 3, pp. 982-997, May 2015, doi: 10.1109/JPHOTOV.2015.2397599.

[2] R. Hariharan, M. Chakkarapani, G. Saravana Ilango and C. Nagamani, ”A Method to Detect Photovoltaic Array Faults and Partial Shading in PV Systems,” in IEEE Journal of Photovoltaics, vol. 6, no. 5, pp. 1278-1285, Sept. 2016, doi: 10.1109/JPHOTOV.2016.2581478.

[3] Y. Hu et al., ”Online Two-Section PV Array Fault Diagnosis With Optimized Voltage Sensor Locations,” in IEEE Transactions on Industrial Electronics, vol. 62, no. 11, pp. 7237-7246, Nov. 2015, doi: 10.1109/TIE.2015.2448066.

[4]] Z. Yi and A. H. Etemadi, ”Fault Detection for Photovoltaic Systems Based on Multi-Resolution Signal Decomposition and Fuzzy Inference Systems,” in IEEE Transactions on Smart Grid, vol. 8, no. 3, pp. 1274-1283, May 2017, doi: 10.1109/TSG.2016.2587244.Z.

[5] Yi and A. H. Etemadi, ”Line-to-Line Fault Detection for Photovoltaic Arrays Based on Multiresolution Signal Decomposition and Two-Stage Support Vector Machine,” in IEEE Transactions on Industrial Electronics, vol. 64, no. 11, pp. 8546-8556, Nov. 2017, doi: 10.1109/TIE.2017.2703681.

[6] K. Li, S. Zhao and Y. Wang, ”A Planar Location Method for DC Arc Faults Using Dual Radiation Detection Points and DANN,” in IEEE Transactions on Instrumentation and Measurement, vol. 69, no. 8, pp. 5478-5487, Aug. 2020, doi: 10.1109/TIM.2020.2966311.

[7] A. Eskandari, J. Milimonfared and M. Aghaei, ”Fault Detection and Classification for Photovoltaic Systems Based on Hierarchical Classification and Machine Learning Technique,” in IEEE Transactions on Industrial Electronics, vol. 68, no. 12, pp. 12750-12759, Dec. 2021, doi: 10.1109/TIE.2020.3047066.

[8] S. Madichetty, Y. V. S. Manoj, S. A. Kareem and S. Mishra, ”A Novel High-Speed Sensorless Faulty Panel Detection Technique for an SPV String/Array: An accurate and cost-effective approach for SPV industry,” in IEEE Power Electronics Magazine, vol. 9, no. 1, pp. 33-39, March 2022, doi: 10.1109/MPEL.2022.3140985.

[9] J. -M. Huang, R. -J. Wai and W. Gao, ”Newly-Designed Fault Diagnostic Method for Solar Photovoltaic Generation System Based on IV-Curve Measurement,” in IEEE Access, vol. 7, pp. 70919-70932, 2019, doi: 10.1109/ACCESS.2019.2919337.

[10] A. F. Murtaza, H. A. Sher, K. Al-Haddad and F. Spertino, ”Module Level Electronic Circuit Based PV Array for Identification and Reconfiguration of Bypass Modules,” in IEEE Transactions on Energy Conversion, vol. 36, no. 1, pp. 380-389, March 2021, doi: 10.1109/TEC.2020.3002953.