sustainability

MDPI

Article

A Machine Learning and Internet of Things-Based Online Fault Diagnosis Method for Photovoltaic Arrays

Adel Mellit [1,*], Omar Herrak [1], Catalina Rus Casas [2] and Alessandro Massi Pavan [3]

[1] Renewable Energy Laboratory, Jijel University, Jijel 18000, Algeria; herrakomar@gmail.com
[2] Departamento de Ingeniería Electrónica y Automática, Universidad de Jaén, 23071 Jaén, Spain; crus@ujaen.es
[3] Department of Engineering and Architecture, Center for Energy, Environment and Transport Giacomo Ciamician, University of Trieste, 34127 Trieste, Italy; apavan@units.it
* Correspondence: adelmellit013@gmail.com

Abstract: In this paper, a novel fault detection and classification method for photovoltaic (PV) arrays is introduced. The method has been developed using a dataset of voltage and current measurements (I–V curves) which were collected from a small-scale PV system at the RELab, the University of Jijel (Algeria). Two different machine learning-based algorithms have been used in order to detect and classify the faults. An Internet of Things-based application has been used in order to send data to the cloud, while the machine learning codes have been run on a Raspberry Pi 4. A webpage which shows the results and informs the user about the state of the PV array has also been developed. The results show the ability and the feasibility of the developed method, which detects and classifies a number of faults and anomalies (e.g., the accumulation of dust on the PV module surface, permanent shading, the disconnection of a PV module, and the presence of a short-circuited bypass diode in a PV module) with a pretty good accuracy (98% for detection and 96% classification).

Keywords: photovoltaic array; machine learning; Internet of Things; fault detection; fault classification

Citation: Mellit, A.; Herrak, O.; Rus Casas, C.; Massi Pavan, A. A Machine Learning and Internet of Things-Based Online Fault Diagnosis Method for Photovoltaic Arrays. *Sustainability* **2021**, *13*, 13203. https://doi.org/10.3390/su132313203

Academic Editor: Manosh C. Paul

Received: 27 October 2021
Accepted: 25 November 2021
Published: 29 November 2021

1. Introduction

The International Energy Agency (IEA) reports that at the end of 2020, global photovoltaic (PV) capacity installations reached 760 GWp [1]. PV monitoring systems are indispensable for the reliable operation and maintenance activities of an impressive number of photovoltaic systems. Recently, with the advances in telecommunication technologies, the Internet of Things (IoT) technology represents a key solution for the design of remote PV monitoring systems [2]. However, a fault diagnosis technique should be embedded in the system in order to prevent and isolate any possible fault, which may compromise the normal operation of a PV installation [3].

With reference to the fault diagnosis methods, a good number of machine learning (ML) methods have been developed and presented in the literature [3,4]. These have demonstrated a good ability in the detection and the classification of both common faults, (e.g., open circuit, short circuit, partial shading, soiling, and degradation) and complex faults (e.g., multiple faults). In general, ML-based models are trained and tested by using measured or simulated data (which can be obtained by using MATLAB/Simulink, (Ver. 2018a, MathWorks: Natick, MA, USA). Only a limited number of works have been verified experimentally using a real-time implementation of the developed algorithms. A fault diagnosis system usually includes of a number of tasks such as the detection, the identification, the classification, and the localization of the faults.

In order to detect a fault occurring in a PV system, the simplest approach consists of a comparison between the measured and the predicted output powers. There are two different approaches which are usually used in order to estimate the PV power:

1. The first uses accurate mathematical models which can estimate the produced power as a function of some parameters, such as the solar irradiance (G), the air temperature

(T), and the cell temperature (Tc). These models can be implicit or explicit and are based on the parameters which can be found in the datasheet of the PV modules;

2. The second is based on a data-driven approach. In this case, a dataset of a number of parameters is used in order to estimate the produced power.

With reference to the identification and classification of the faults, these can be performed by solving a binary classification and multiclass problem using ML techniques [5–7] or conventional methods. These latter are known as conventional methods and are usually used in order to detect simple faults which are not associated with any other fault. Thus, the PV fault classification and identification consists in the solution of a multiclass problem.

In the literature, there are only a few recent works regarding the fault localization in PV systems [8–10], and this topic still represents an important challenge in the field, especially for large-scale PV plants [11].

A low-cost PV monitoring system based on the Internet of Things (IoT) technique has been developed in Ref. [12]. Here, the monitored data were the currents and the voltages of the PV array, as well as the environmental data (T and G). The monitoring system was developed using an Arduino Mega 2560 microcontroller (2005, the Interaction Design Institute Ivrea, Milano, Italy). A simple fault detection and identification procedure has been described in Ref. [13] for the detection and identification of four types of faults occurring in a PV string (permanent shading, soiling/deposit of dust, short-circuited PV modules, and disconnected PV modules). However, the microcontroller used in this study was not suitable for the fault classification based on the ML methods. This was due to the limited resources of the device. In this work, we aim at showing that a Raspberry Pi 4 microprocessor (2012, Raspberry Pi Foundation, Cambridge, UK) can overcome this type of problem.

In the literature, there are already a certain number of papers where the Raspberry Pi has been used for the development of an IoT-based technique for the monitoring of PV systems [14–17]. However, only in a small number of works has this microprocessor been used in the field of automatic PV fault diagnosis. Thus, the main contribution of this paper is the development of a fast and accurate method for the online automatic PV fault detection and classification. The developed method allows the experimental verification of the capability of ML algorithms to detect and classify the faults occurring in PV modules, and to monitor the operation of a PV array, thanks to the use of a webpage which informs the users about the state of their installations. The investigated faults are: the soiling/deposit of dust, permanent shading, the short-circuited bypass diode in PV modules, and disconnected PV modules.

Soiling mainly occurs due to environmental conditions, which depend on the location where the PV system is installed. Permanent shading is caused by the presence of buildings, trees, etc. Short-circuited bypass diodes may be caused by many factors, such as overheating, corrosion, manufacturing problems, bad connections, etc. Open-circuited PV modules are mainly due to the breaking down of panel-panel cables or connections, bypass diode issues, bad connections, etc.

The paper is organized as follows: Section 2 provides the description of the system and of the database used to develop the fault detection method. The proposed fault detection method based on machine leaning algorithms is described in Section 3, while the results and the discussion are reported in Section 4. Finally, the conclusion and perspectives are provided in Section 5.

2. Photovoltaic Array Description and Dataset

The PV array considered in this study consists of three parallel-connected PV modules installed at the University of Jijel (Algeria). Figure 1 shows the PV modules used in this study [12]. The considered faults are: (a) soiling/dust deposit; (b) permanent shading; (c) open-circuit (disconnected of one PV module); and (d) short-circuit (short circuited bypass diode in a PV module). The "Prova I-V tracer" (See Table A1) is used to collect the data (Figure 1e), while the faults are labeled manually. The data (I–V curves) were collected

during different climatic conditions and under normal and abnormal circumstances (faulty PV array). The PV module specifications are listed in Table 1.

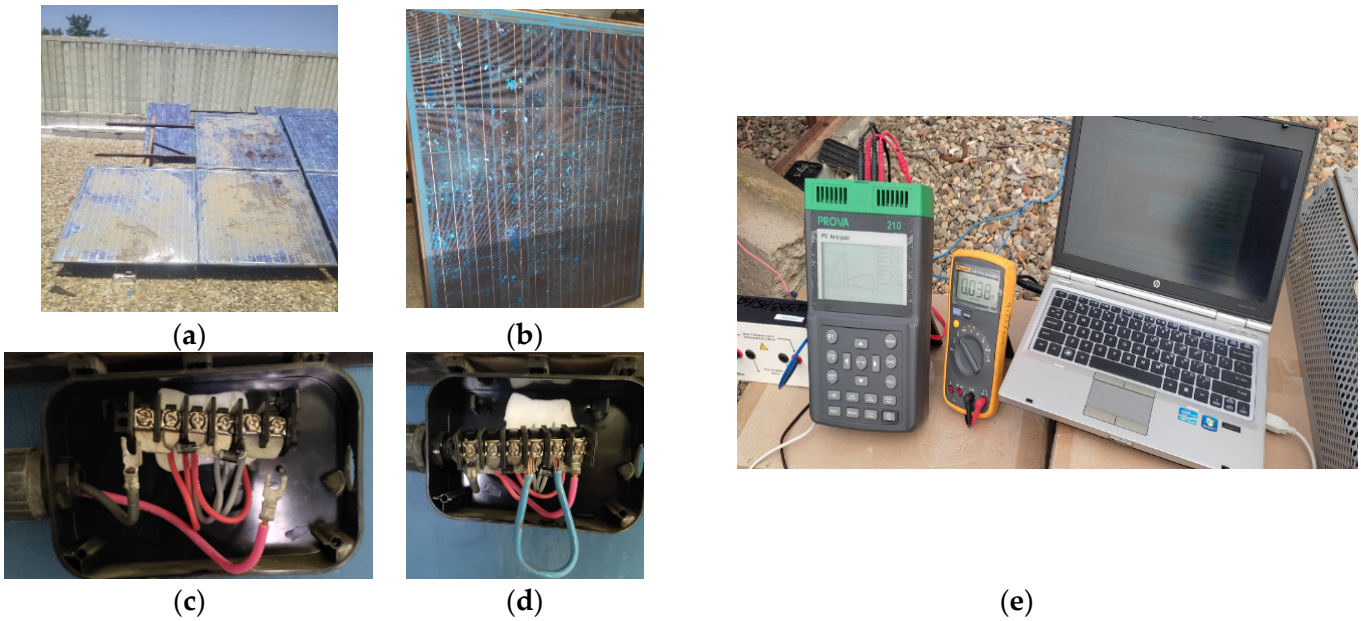


Figure 1. (**a**) Dust deposit on the PV modules surface, (**b**) shading (permeant shadow), (**c**) open circuited PV module (one PV module disconnected), (**d**) short-circuited bypass diode in a PV module, and (**e**) the used measurement instrument ("Prova 210 I–V tracer").

Table 1. PV module specifications.

Power (Pmp)	121.4 W
Maximum voltage (Vmp)	17.1 V
Maximum current (Imp)	7.11 A
Short-circuit current (Isc)	7.98 A
Open circuit voltage (Voc)	21.2 V
Maximum bypass diode current	8.1 A
Current Temperature Coefficient	+0.054%/°C
Voltage Temperature Coefficient %/°C	−0.35%/°C

Figure 2 shows an example of the I–V curves measured during the experiments. A number of measurements have been carried out under different working conditions. In total, we have collected 246 I–V curves from which we could extract the main features of the PV array, such as: the short circuit current (Isc), the open circuit voltage (Voc), the current at maximum power point (Imp), the voltage at the maximum power point (Vmp), the power at the maximum power point (Pmp), and the fill factor (FF). While preparing the dataset, missed data usually occur and they are automatically ignored from the dataset (CSV files), and redundant features have also been removed.

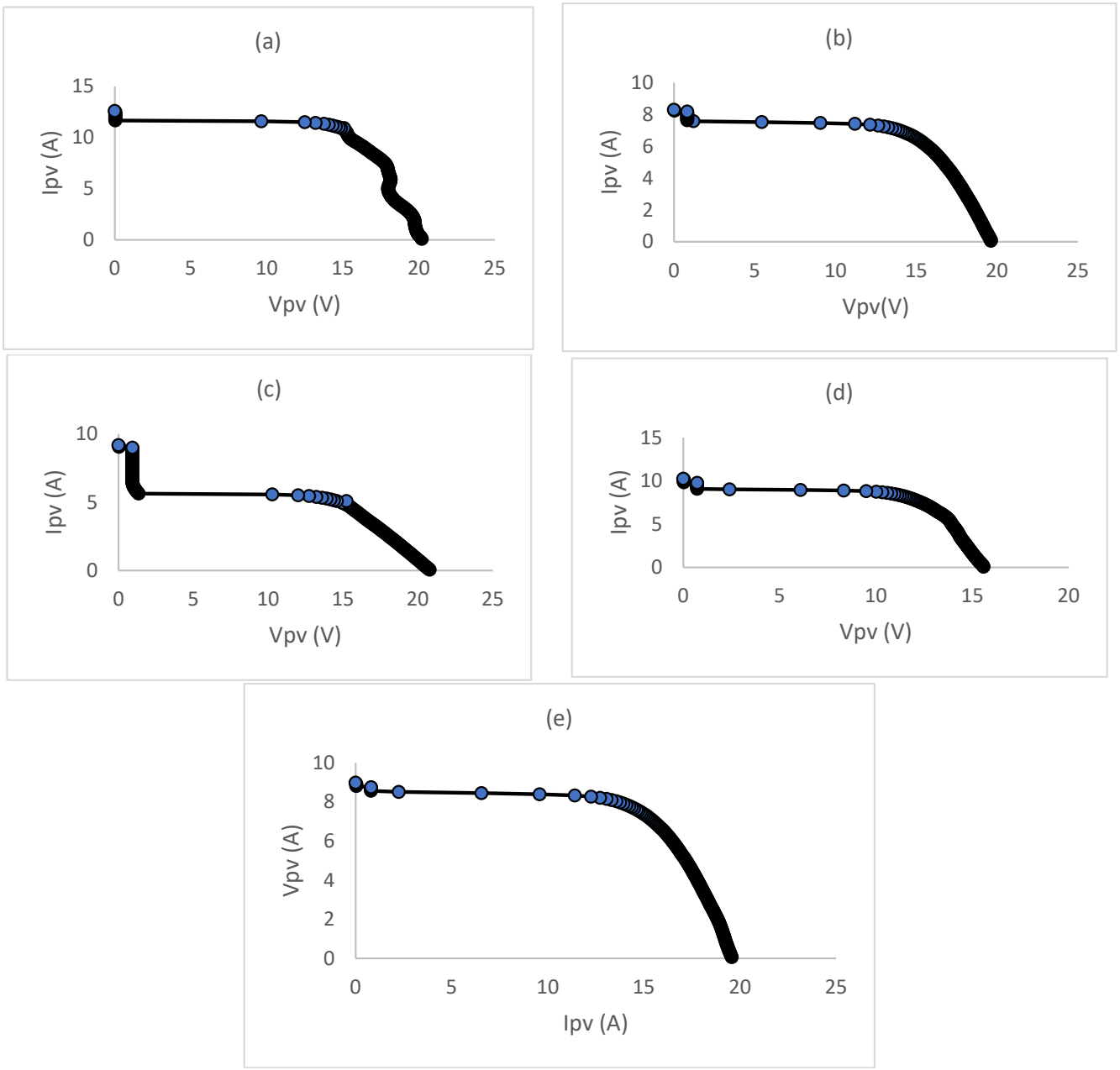


Figure 2. Some of the I-V curves measured for different faults under various working conditions: (**a**) dust deposit on the PV array surface; (**b**) open circuit (one PV module disconnected); (**c**) shading (permanent shadow); (**d**) short circuited (short-circuited bypass diode in one PV module); (**e**) normal operations (no fault).

3. The Proposed Method

The block diagram of the proposed method is shown in Figure 3. There are three main blocks:

Block #1. This block contains the PV array together with the sensors used to measure the PV current, the PV voltage, the solar irradiance, and the cell temperature every thirty minutes.

Block #2. This block comprises the controller, which is based on a Python (version 3.8, Python Software Foundation: Wilmington, DE, USA) code implemented into the Raspberry Pi 4. The code contains the fault detection program, which is based on a ML method (decision tree) [18], an explicit I-V model with features extraction

parameters [19], and a fault classification method based on an ensemble method (random forest) [18].

Block #3. This block includes the webpage application which has been designed for the remote visualization of the stored data, and to notify the users about the status of the PV array.

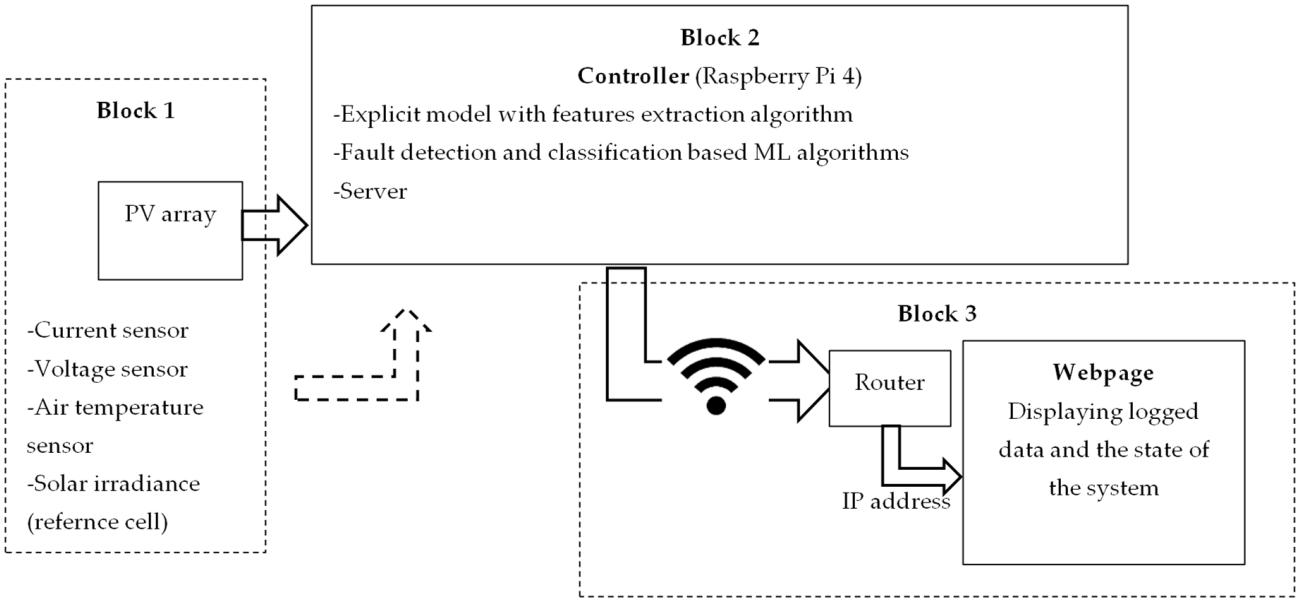


Figure 3. Block diagram of the proposed method.

The code performs four main steps which can be described as follows:

Step 1: import the libraries and the functions from Python.
Step 2: load the dataset and split it into the training and the testing subsets.
Step 3: select the ML algorithm and use the k-fold cross validation technique.
Step 4: fit the ML model and predict the results.

A large number of ML and ensemble techniques (e.g., Naïve Bayes, decision tree, k-nearest neighbors, random forest, neural networks, boosting, bagging, CatBoost, LightGBM, XGboost, etc.) are available in the literature. In this study, a decision tree (DT) algorithm has been used for the detection of faults, while a random forest algorithm (RF) has been used for their classification.

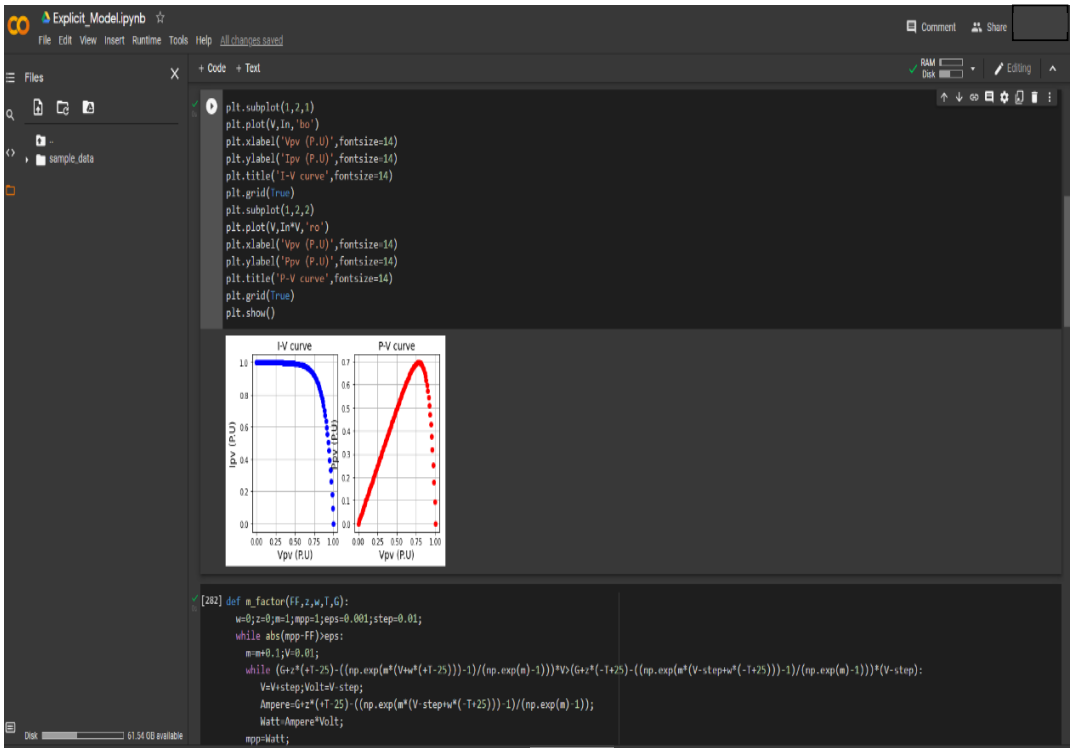
It is assumed that only one fault at time can happen during the measurement process. Multiple faults and faults with similar symptoms are not considered in this study.

With reference to Figure 4, the I-V explicit model described in Ref. [19] is used in order to estimate the I-V curve based on the values of G and T, while a simple algorithm is employed to extract the main features of the I-V curve (Voc, Isc, Vmp, Imp, Pmp, and FF).

The detection of faults in PV arrays can be modeled as a binary classification problem. Among the different available ML methods, the decision tree algorithm has been chosen in order to detect the faulty PV module because of its simplicity. Concerning the PV fault classification, which is a multiclass classification problem, an ensemble learning method has been chosen, the random forest algorithm.

3.1. Programming Language

With reference to Figure 5, the fault detection and classification models are implemented online using Google Colab https://colab.research.google.com (accessed on 23 May 2018), a cloud platform that provides Jupyter netbook https://jupyter.org/ (accessed on 1 February 2015) services. Google Drive https://www.google.com/drive/ (accessed on 1 February 2015) was used in order to read the dataset.



(**a**)

```
print('Extracted parameters:')
print(Vmpp, Impp, FF, Isc, Voc, Vmpp*Impp)

Extracted parameters:
0.7800000000000005 0.8916445206795417 0.7534638142922166 0.99999 0.99999 0.6954827261300429
```

(**b**)

Figure 4. (**a**) An example of the I-V and P-V curves obtained using the explicit model, (**b**) extracted features Vmp, Imp, FF, Isc, Voc, and Pmp.

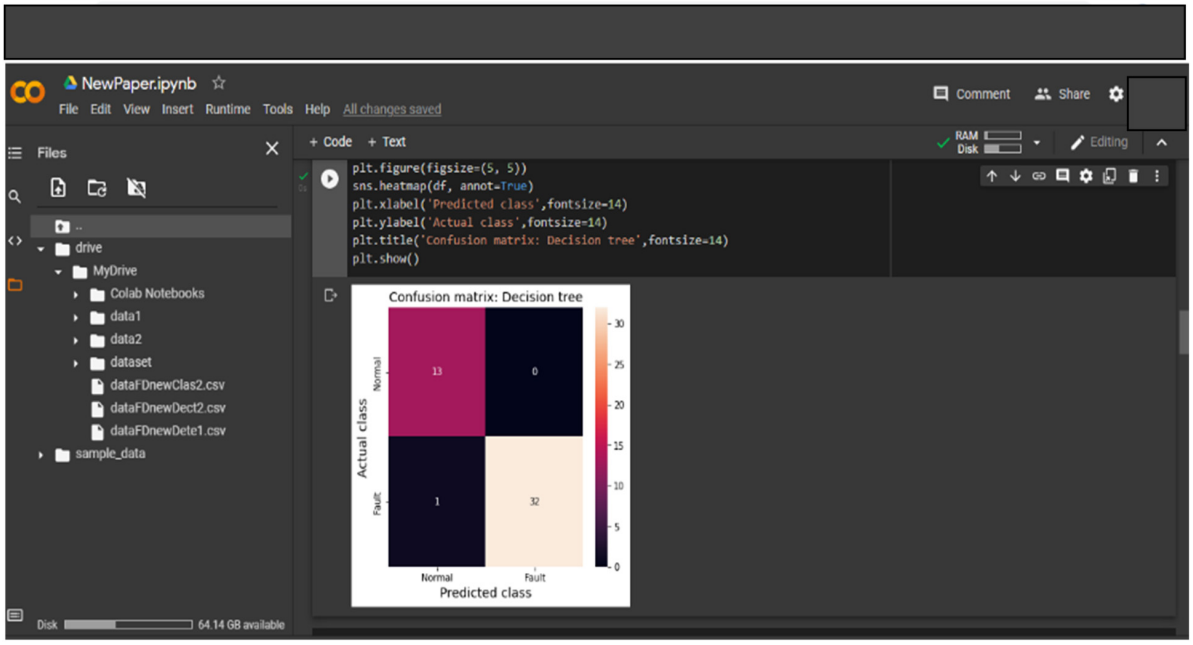


Figure 5. Google Colab interface used for developing and running the proposed method.

As an example, Appendix B shows the main functions used to develop both classifiers.

3.2. Performance Metrics

In order to evaluate the performance of the developed classifiers, the well-known confusion matrix (CM) method has been used in order to calculate the accuracy, the precision, the sensitivity, the F-score, and the misclassification rate, which are defined as follows:

$$\text{Accuracy } (\%) = \frac{\sum_i \text{CM}(i,i)}{\sum_i \sum_j \text{CM}(i,j)} \times 100 \quad (1)$$

$$\text{Precision}_i(\%) = \frac{\text{CM}(i,i)}{\sum_j \text{CM}(j,i)} \times 100 \quad (2)$$

$$\text{Sensitivity}_i(\%) = \frac{\text{CM}(i,i)}{\sum_j \text{CM}(i,j)} \times 100 \quad (3)$$

$$F1 - \text{score } (\%) = 2\frac{(\text{sensitivity} \times \text{Recall})}{(\text{sensitivity} + \text{Precision})} \times 100 \quad (4)$$

$$\text{Misclassified rate } (\%) = 100 - \text{ accuracy} \quad (5)$$

4. Experimental Implementation and Results

A K-fold cross validation technique has been used to resample the dataset without replacement. The advantage of this technique is that each example is used both for the training and for the validation exactly once. This yields a lower variance estimate of the model performance than the holdout method. Each classifier (DT and RF) accepts as input G, T, and the extract I-V features (Isc, Voc, Imp, Vmp, Pmp, and FF).

In order to develop the classifiers (binary classification for the detection and multiclass classification for the fault classification), a number of experiments have been carried out by tuning the hyper-parameters for both the classifiers' decision tree and the random forest algorithms. The adjusted random forest parameters are: max_depth, max_features, n_estimators, and random state. The adjusted decision tree parameters are: criterion, max_depth, and random_state.

A cross validation method has been used to evaluate both classifiers. This is a resampling procedure used to evaluate ML models on a limited data sample [20].

4.1. Fault Detection Performance

The cross-validation accuracy scores for K = 10 are given in Figure 6.

The error metrics are listed in Table 2. The accuracy is 98% and the misclassified rate is 2%. With reference to the confusion matrix shown in Figure 7, in the first row, thirteen normal cases have been classified correctly, while in the second row, thirty-two are correctly classified and only one fault is incorrectly classified as a normal case.

Table 2. Error metrics: precision, recall, F1-score, accuracy, and misclassified rate for the fault detection method using the decision tree learning algorithm (with tuned parameters).

ML Classifier	**Tuned Parameters: Criterion = gini, max_depth = 3 and random_state = 40**				
	Precision (%)	**Sensitivity (%)**	**F1-Core (%)**	**Classification Accuracy (%)**	**Misclassified Rate (%)**
Normal	93	100	95	98	2
Fault	100	97	98		

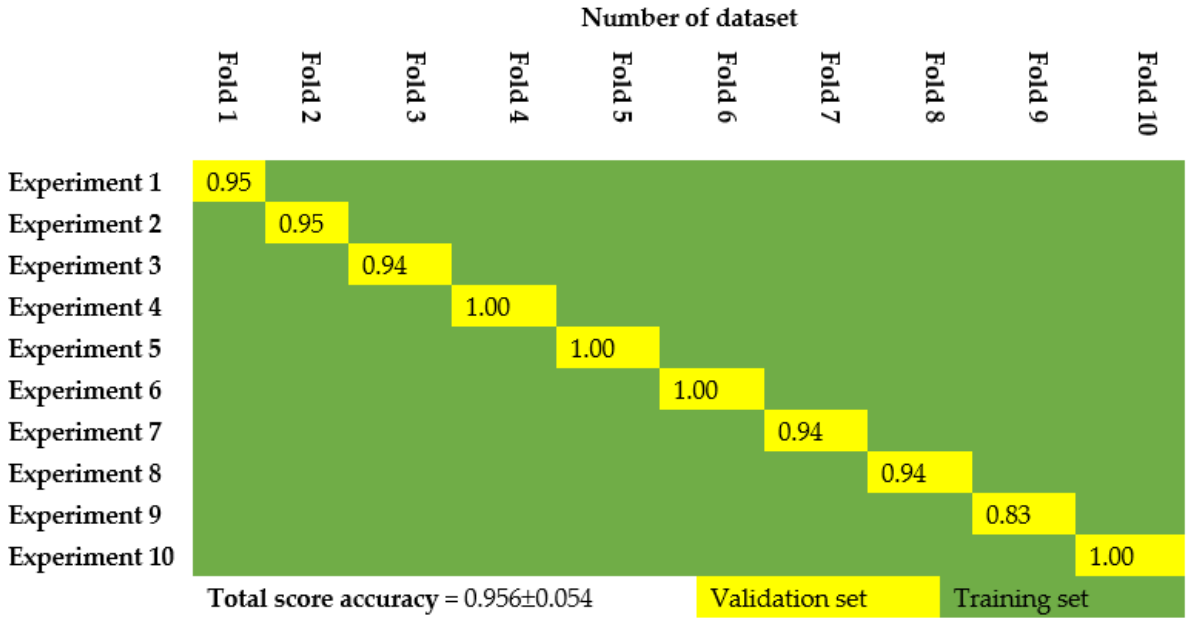


Figure 6. Cross validation accuracy scores for K = 10 (fault detection).

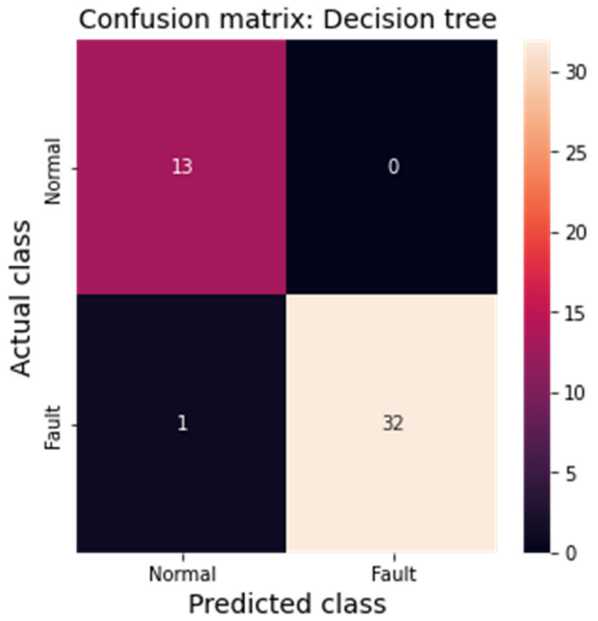


Figure 7. Confusion matrix: fault detection using the decision tree algorithm.

The decision tree results are shown in Figure 8.

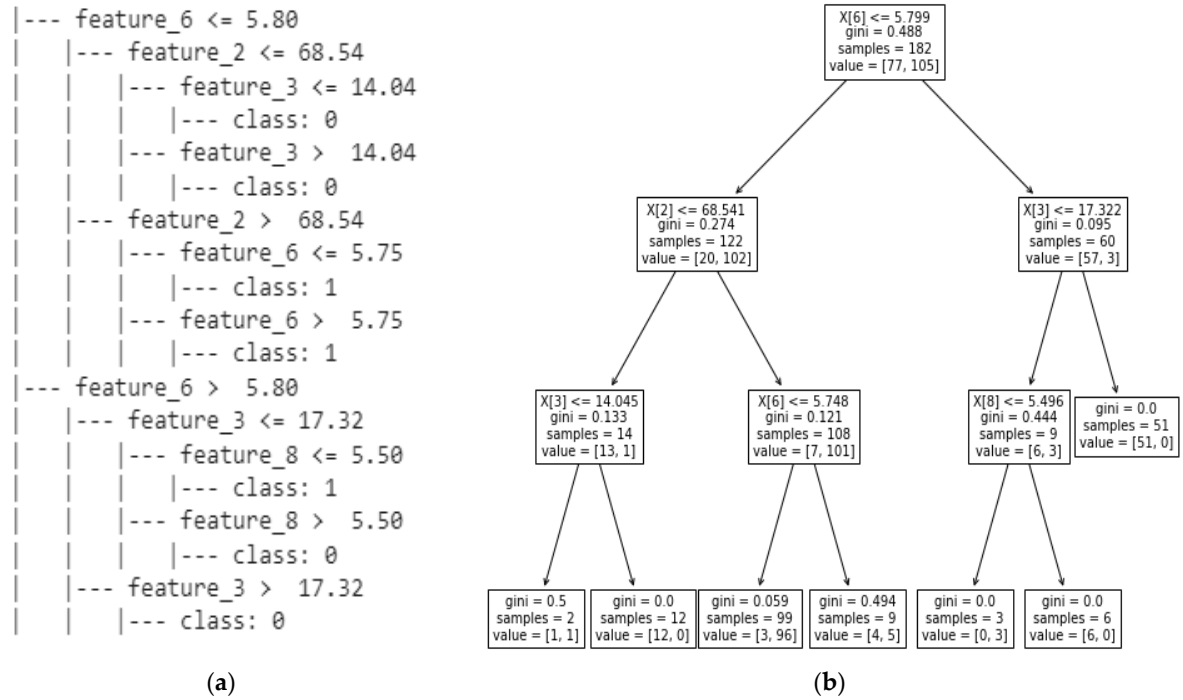


Figure 8. Decision tree model: (**a**) decision tree text and (**b**) decision tree diagram of the model.

4.2. Fault Classification Performance

The cross-validation accuracy scores for K = 10 is given in Figure 9.

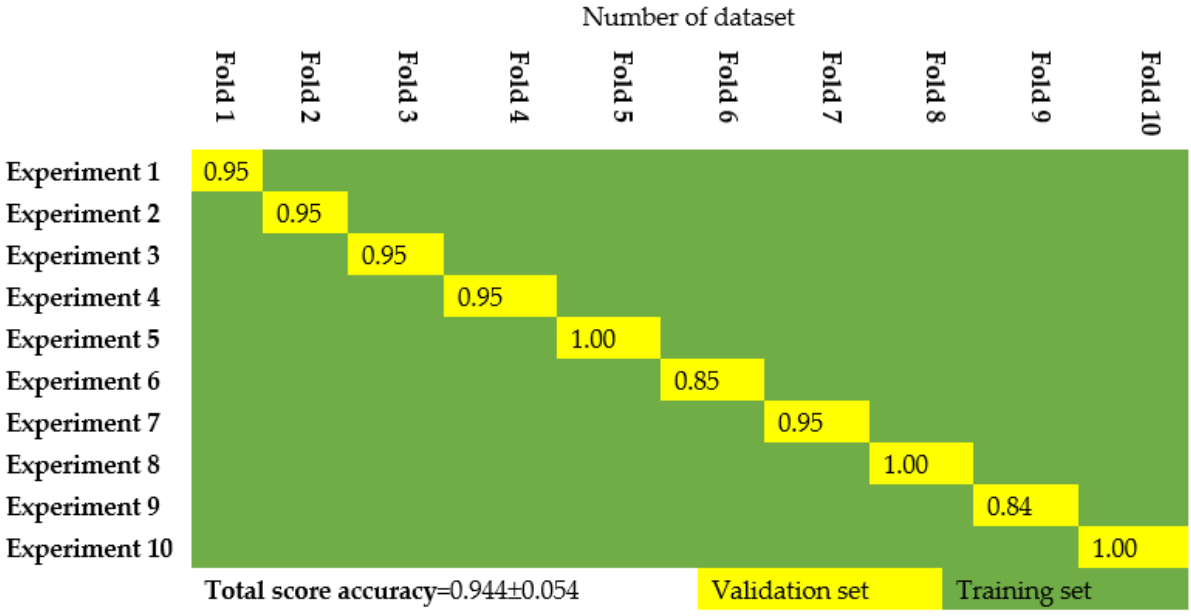


Figure 9. Cross validation accuracy scores for K = 10 (fault classification).

The error metrics are calculated and listed in Table 3. The accuracy is 96% and the misclassified rate is 4%. With reference to the confusion matrix plotted in Figure 10, in the first row, which refers to the fault class 0, nine samples are correctly classified and only one element is not; the sensitivity is 9/10 (90%). In the second row, all the eight samples are correctly classified, and the sensitivity is 8/8 (100%). In the third row, 20 samples are correctly classified, while one is misclassified into class 1; the sensitivity is 20/21 (95%).

With reference to the last row, all eleven samples are correctly classified, and the sensitivity is 11/11 (100%).

Table 3. Error metrics: precision, recall, F1-score, accuracy, and misclassified rate for fault classification using random forest ensemble learning algorithm (with tuned parameters).

ML Classifier Parameters	max_dept h = 5, max_features = sqrt, n_estimators = 300, and random_state = 42				
Fault Classes	**Precision (%)**	**Sensitivity (%)**	**F1-Score (%)**	**Classification Accuracy (%)**	**Misclassified Rate (%)**
Class {0}: dust deposit on the PV array surface	100	90	95	96	4
Class {1}: short-circuited bypass diode in one PV module	80	100	89		
Class {2}: permanent shadow	100	95	98		
Class {3}: disconnected one PV module	100	100	100		

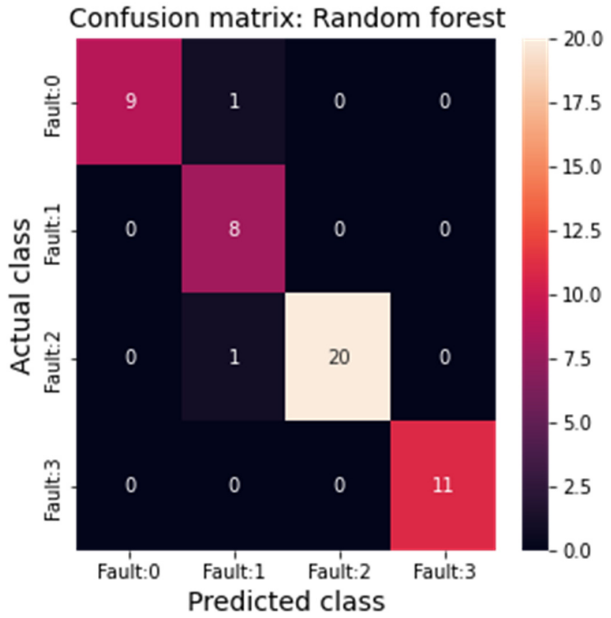


Figure 10. Confusion matrix: fault classification using random forest algorithm.

With reference to the first column of the matrix, all the nine samples are correctly classified, and the precision is 9/9 (100%). In the second column, eight samples are correctly classified, while two are misclassified into class 0 and class 1 respectively; the precision is 8/10 (80%). In the third and the last columns, all the twenty and eleven samples are well classified so that the precisions are 20/20 (100%) and 11/11 (100%), respectively. The F1-score ranges between 89% and 100%. Some of the faults (e.g., class 3, class 2, and class 0) are very well classified, having a precision of 100%. Nevertheless, class 1 faults are characterized by a precision of 80%, and that is acceptable. Globally, the results listed in Table 4 are quite satisfactory.

Table 4. Component specification and costs.

Components	Specification	Cost (Dollars)
Raspberry Pi 4	Cortex-A72 (ARMv8), 4 Go	85
Current sensor ACS712	30 A	6
Voltage sensor	25 V	4
Temperature sensor Type K	Max6675 −20 °C + 80 °C	5
Solar irradiance (Silicon irradiance sensor)	0–1200 W/m^2	50

4.3. Experimental Implementation

Once the detection and classification algorithms have been verified, these have been implemented into the Raspberry Pi 4 for a real time verification. The WiFi module embedded into the Raspberry Pi 4 has been used in order to send data to the cloud.

The following steps represent the main procedure implemented into the microprocessor.

Step 1: read the data (G, T, Ipv, and Vpv) by the Raspberry Pi 4.
Step 2: call the explicit model to estimate the I-V curve.
Step 3: call the features extraction algorithm to calculate Isc, Voc, Imp, Vmp, FF, and Pmp.
Step 4: call the fault detection procedure, and based on the calculated features, display the results on the webpage and go to **Step 1** if the PV system works properly, otherwise go to the next **Step 5**.
Step 5: call the fault classification procedure and identify the nature of the fault.
Step 6: display the results on the webpage and notify the user by indicating the type of fault.

Figure 11 shows the basic experimental setup of the prototype where the Raspberry Pi 4 is used.

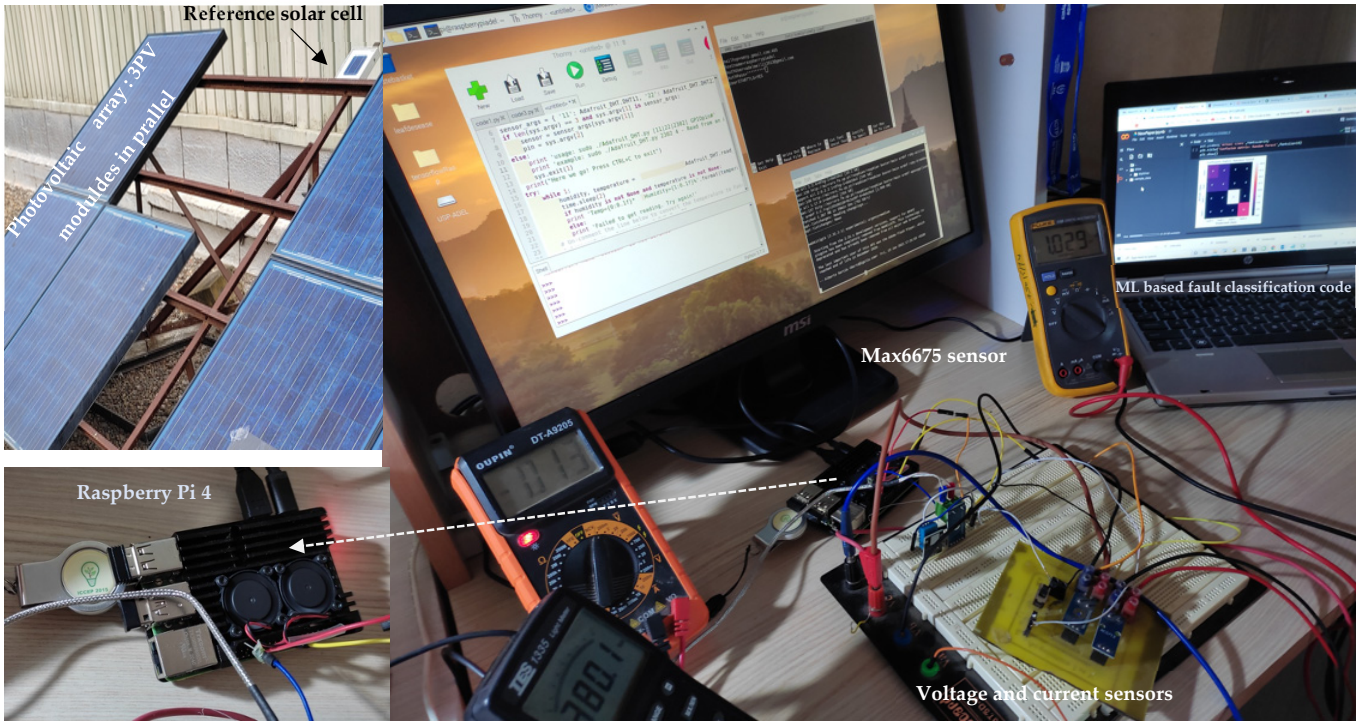


Figure 11. Experimental setup at the laboratory level: Raspberry Pi 4, sensors, PV modules, and data-acquisition system.

The specification and the cost of the main components used in the experiment are listed in Table 4. The total cost is USD 150.

In order to test the method, artificial shading has been created on the field by partially covering one of the PV modules. Figure 12 shows the corresponding I-V curve, the extracted parameters, the current, the voltage, the cell temperature, and the solar irradiance. The state of the system is also displayed in the webpage. The results show clearly that the method is able to detect and correctly classify the fault that occurred on the investigated PV array (type of fault: class #2).

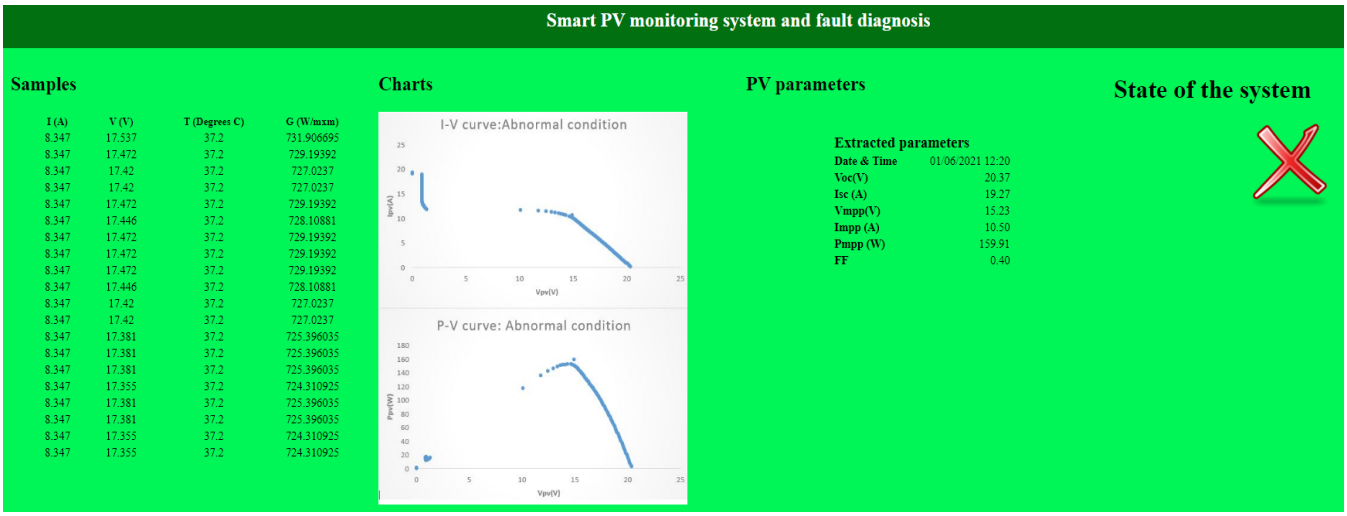


Figure 12. The webpage with the monitored data and the state of the system (https://solar-system.w3spaces.com/ (accessed on 20 May 2021)).

5. Conclusions

In this work, a machine learning-based fault diagnosis method for photovoltaic arrays has been developed and experimentally verified. A Python code, including the decision tree, the random forest, the explicit model of the I–V curve, and the extraction parameters algorithms were written to a Raspberry Pi 4 microprocessor suitable for real-time applications.

The experimental results showed the feasibility of the developed method to detect and classify the common faults occurring in PV arrays. The designed prototype can rapidly detect and classify the examined faults with a good accuracy (98% for the detection and 96% for the classification).

It should be pointed out that the data should be periodically updated in order to keep the classifier working effectively and avoiding false alarms. In addition, multiple PV faults have not been considered in this study, which remain an open challenge.

This work can be further improved and extended for fault detection in photovoltaic systems, including DC-DC converters, batteries, and DC-AC inverters. The webpage designed could be also enhanced by posting more information about the PV installation, as well as to notify users by e-mail or SMS.

Author Contributions: Conceptualization, A.M. and A.M.P.; methodology, A.M.; software, O.H.; validation, O.H., A.M. and C.R.C.; formal analysis, A.M.P.; investigation, O.H.; writing—original draft preparation, A.M.; writing—review and editing, C.R.C.; visualization, C.R.C.; supervision, A.M.P. All authors have read and agreed to the published version of the manuscript.

Funding: A. Massi Pavan acknowledges financial support provided by "DEEP-SEA—Development of energy efficiency planning and services for the mobility of Adriatic Marinas", a project co-financed by the European Regional Development Fund (ERDF) via the cross-border cooperation program Interreg Italy-Croatia. A. Mellit acknowledges financial support provided by the DGRSDT, Algiers, (Algeria) socio-economic project: ''Realization of a smart prototype for fault diagnosis of photovoltaic systems (7/2019)".

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Adel Mellit acknowledges the ICTP for their support throughout the Simons Associate Program.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Prova I-V tracer specifications.

Range (60 V/12A)	Resolution	Accuracy
DC Voltage Measurement		
0–10 V	0.001 V	±1% ± (1% of Vopen ± 0.1 V)
10–60 V	0.01 V	±1% ± (1% of Vopen ± 0.1V)
DC Current Measurement		
0.01–10 A	1 mA	±1% ± (1% of Ishort ± 9 mA)
10–12 A	1 mA	±1% ± (1% of Ishort ± 9 mA)

Appendix B

```
Main code (DT and RF functions)
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.pipeline import make_pipeline
In_train, In_test, Out_train, Out_test = train_test_split(InD, OutD,
test_size=0.20,random_state=42)
    Classifier_RF_Model = make_pipeline(StandardScaler(),
RandomForestClassifier(n_estimators=300, max_depth=5))
    #Classifier_DT_Model = make_pipeline(StandardScaler(),
DecisionTreeClassifier(max_depth=5, random_state=40))
classifier.fit(In_train, Out_train)
Out_pred = classifier.predict(In_test)
```

References

1. Snapshot of Global PV Markets. 2021 Report IEA-PVPS T1-39. 2021. Available online: https://iea-pvps.org/snapshot-reports/snapshot-2021/ (accessed on 25 April 2021).
2. Adhya, S.; Saha, D.; Das, A.; Jana, J.; Saha, H. An IoT Based Smart Solar Photovoltaic Remote Monitoring and Control Unit. In Proceedings of the 2nd International Conference on Control, Instrumentation, Energy & Communication (CIEC), Kolkata, India, 28–30 January 2016; pp. 432–436.
3. Mellit, A.; Kalogirou, S. Artificial intelligence and internet of things to improve efficacy of diagnosis and remote sensing of solar photovoltaic systems: Challenges, recommendations and future directions. *Renew. Sustain. Energy Rev.* **2021**, *143*, 110889. [CrossRef]
4. Tina, G.M.; Ventura, C.; Ferlito, S.; De Vito, S. A State-of-Art-Review on Machine-Learning Based Methods for PV. *Appl. Sci.* **2021**, *11*, 7550. [CrossRef]
5. Tchoketch_Kebir, S.; Cheggaga, N.; Ilinca, A.; Boulouma, S. An Efficient Neural Network-Based Method for Diagnosing Faults of PV Array. *Sustainability* **2021**, *13*, 6194. [CrossRef]
6. Samara, S.; Natsheh, E. Intelligent PV panels fault diagnosis method based on NARX network and linguistic fuzzy rule-based systems. *Sustainability* **2020**, *12*, 2011. [CrossRef]
7. Barker, C.; Cipkar, S.; Lavigne, T.; Watson, C.; Azzouz, M. Real-Time Nuisance Fault Detection in Photovoltaic Generation Systems Using a Fine Tree Classifier. *Sustainability* **2021**, *13*, 2235. [CrossRef]
8. Mellit, A. Recent Applications of Artificial Intelligence in Fault Diagnosis of Photovoltaic Systems. In *A Practical Guide for Advanced Methods in Solar Photovoltaic Systems*; Springer: Cham, Switzerland, 2020; Volume 128, pp. 257–271. [CrossRef]
9. Mehmood, A.; Sher, H.A.; Murtaza, A.F.; Al Haddad, K. Fault detection, Classification and Localization Algorithm for Photovoltaic Array. *IEEE Trans. Energy Convers.* **2021**, *70*, 1–12. [CrossRef]
10. Dhoke, A.; Sharma, R.; Saha, T.K. An approach for fault detection and location in solar PV systems. *Sol. Energy* **2019**, *194*, 197–208. [CrossRef]
11. Mellit, A.; Tina, G.M.; Kalogirou, S.A. Fault detection and diagnosis methods for photovoltaic systems: A review. *Renew. Sustain. Energy Rev.* **2018**, *91*, 1–17. [CrossRef]
12. Hamied, A.; Mellit, A.; Zoulid, M.A.; Birouk, R. IoT-based experimental prototype for monitoring of photovoltaic arrays. In Proceedings of the 2018 International Conference on Applied Smart Systems (ICASS), Medea, Algeria, 24–25 November 2018; pp. 1–5. [CrossRef]
13. Mellit, A.; Hamied, A.; Lughi, V.; Pavan, A.M. A low-cost monitoring and fault detection system for stand-alone photovoltaic systems using IoT technique. In *ELECTRIMACS*; Springer: Cham, Switzerland, 2020; pp. 349–358. [CrossRef]

14. Badave, P.M.; Karthikeyan, B.; Badave, S.M.; Mahajan, S.B.; Sanjeevikuma, P.; Gill, G.S. Health monitoring system of solar photovoltaic panel: An internet of things application. In *Advances in Smart Grid and Renewable Energy*; Springer: Singapore, 2018; pp. 347–355. [CrossRef]
15. Pereira, R.I.; Dupont, I.M.; Carvalho, P.C.; Jucá, S.C. IoT embedded linux system based on Raspberry Pi applied to real-time cloud monitoring of a decentralized photovoltaic plant. *Measurement* **2018**, *114*, 286–297. [CrossRef]
16. Paredes-Parra, J.M.; Mateo-Aroca, A.; Silvente-Niñirola, G.; Bueso, M.C.; Molina-García, Á. PV module monitoring system based on low-cost solutions: Wireless raspberry application and assessment. *Energies* **2018**, *11*, 3051. [CrossRef]
17. Priharti, W.; Rosmawati, A.F.K.; Wibawa, I.P.D. IoT based photovoltaic monitoring system application. *J. Phys. Conf. Ser.* **2019**, *1367*, 012069. [CrossRef]
18. Shalev-Shwartz, S.; Ben-David, S. *Understanding Machine Learning: From Theory to Algorithms*; Cambridge University Press: England, UK, 2014; Available online: https://www.cambridge.org/9781107057135 (accessed on 12 January 2014).
19. Pavan, A.M.; Mellit, A.; Lughi, V. Explicit empirical model for general photovoltaic devices: Experimental validation at maximum power point. *Sol. Energy* **2014**, *101*, 105–116. [CrossRef]
20. Brownlee, J. A Gentle Introduction to k-Fold Cross-Validation. 2018. Available online: https://machinelearningmastery.com/k-fold-cross-validation/ (accessed on 23 May 2018).