SITS/ENTC/Projects/UG/2020-21/GNo 1

A Project Report On

Smart PV system fault detection using AI/ML

By

Nitin Khanka	71833333M
Mohit Jangid	71833290D
Surajpratapsingh Sayar	71833568G
Suhita Vaidya	71833561K

Guide

Mr. Mahesh Deshpande

Sinhgad Institutes

Department of Electronics & Telecommunication

Sinhgad Institute of Technology and Science

Pune 411041

[2020-21]

SINHGAD TECHNICAL EDUCATION SOCIETY'S
SINHGAD INSTITUTE OF TECHNOLOGY & SCIENCE
(Approved by AICTE & Affiliated to savitribai Phule Pune University,(AISHE Code :- C-41214) ID. No.: PU/PN/Engg./323/2008)
Sinhgad Institutes
S. No-49/1, Narhe, Off, Westerly Bypass, Pune-Mumbai Expressway, Pune - 411 041
"Tel. : (020) 6683 1703 / 4 / 5 / 6, Fax : (020) 6683 1710, E-mail : sits@sinhgad.edu Website : www.sinhgad.edu"

Prof. M. N. Navale
M. E.(Elect.), MIE, MBA
FOUNDER PRESIDENT

Dr. (Mrs.)Sunanda M. Navale
B.A. MPM, Ph.D.
FOUNDER SECRETARY

Dr. Rajesh S. Prasad
M.E., MBA, Ph. D.
PRINCIPAL

CERTIFICATE

This is to certify that *Nitin Khanka (Exam Seat No 71833333M), Mohit Jangid (Exam Seat No 71833290D), Surajpratapsingh Sayar (Exam Seat No 71833568G), Suhita Vaidya (Exam Seat No 71833561K)* have successfully completed the Project Stage-I on entitled *Smart PV system fault detection using AI/ML* under my supervision, in the partial fulfilment of Bachelor of Electronics And Telecommunication Engineering of Savitribai Phule Pune University.

Date :
Place :

Mr. Mahesh Deshpande
Guide

Dr. V.M.Rohkale
Head of Department

External Examiner

Dr. S. D. Markande
Principal
Sinhgad Institute of Technology and Science,
Pune 41

ACKNOWLEDGEMENT

We take this opportunity with great pleasure to express our deep sense of gratitude towards our guide *Prof. Mahesh Deshpande* for his valuable guidance, encouragement and cooperation extended to us during this project work.

We are also thankful to *Dr. V. M. Rohkale*, Head, Department of Electronics and Telecommunication for providing departmental facilities for this work.

We would also like to thank *Dr. R. S. Prasad*, Principal, Sinhgad Institute of Technology and Science for their unflinching help, support and cooperation during this project work.

We would also like to thank the Sinhgad Technical Educational Society for providing access to the institutional facilities for our project work.

Nitin Khanka
Mohit Jangid
Surajpratapsingh Sayar
Suhita Vaidya

ABSTRACT

PV systems are subject to various faults and failures, and early fault detection of those faults and failures is very important for the efficiency and safety of the PV systems and to minimize the outage period and maximize the lifetime output. Faults in a PV system can arise from either physical, environmental, or electrical conditions. ML-based fault detection models are trained with data and provide prediction results with very high accuracy. Another study reports the application of ML techniques for fault detection, classification and localization in PV systems. The study claims the development of the algorithm with the prediction accuracy of 100%. However, databased fault detection models for PV systems can sometimes give false predictions, especially when the environmental parameters are not taken into consideration. Our aim is to develop an intelligent fault detection model for PV arrays based on PNN (Probabilistic Neural Network) for accurately classifying the fault in the given module after working on big number of datasets.

CONTENTS

CERTIFICATE . . . . . **I**

ACKNOWLEDGEMENT. . . . . **II**

ABSTRACT. . . . . **III**

1 INTRODUCTION **1**

1.1 Introduction 1

1.2 Relevance 3

1.3 Motivation 3

1.4 Problem Statement 3

1.5 Objective Statements 3

2 LITERATURE REVIEW **4**

2.1 Resume 4

2.2 Review of Paper 5

2.3 Summary 6

3 DESIGN AND DRAWING **7**

3.1 Introduction 7

3.2 Description 9

4 BUDGET OF THE PROJECT **29**

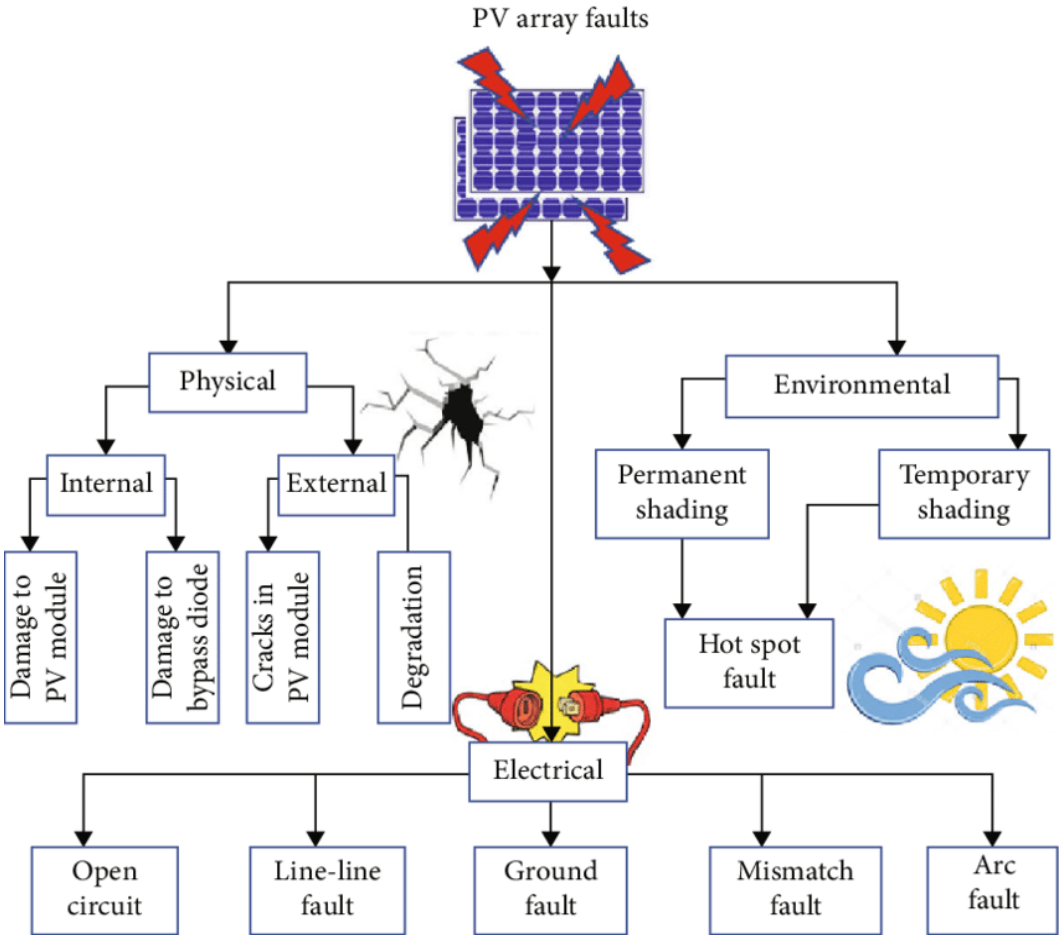
5 WORK PROGRESS. **30**

6 SUMMARY **31**

7 REFERENCE. **32**

1. INTRODUCTION

Fault detection and timely troubleshooting are essential for the optimum performance in any power generation system, including photovoltaic (PV) systems. In particular, the goal for any commercial power-producing house is maximizing power production, minimizing energy loss and maintenance, cost, and the safe operation of the facility. Since PV systems are subject to various faults and failures, early detection of such faults and failures is very crucial for achieving the goal. The US National Electric Code requires the installation of OCPD (Overcurrent Protection Device) and GFDI (Ground Fault Detection Interrupters) in PV installations for protection against certain faults. However, the Bakersfield Fire case, 2009, and Mount Holly, 2011, show the inability of these devices to detect the fault in those particular scenarios. Faults in a PV system can arise from either physical, environmental, or electrical conditions.



Classification of PV array fault

Nowadays, most of the PV systems are built with a monitoring system and have a database constantly backed with huge historical data. Artificial Intelligence (AI) methods are data-based, and with the availability of big data in PV systems, studies in this area seem to be in the momentum. In particular, machine learning (ML) based algorithms and techniques are proposed, where the model is trained with historical data to predict and classify faults. A recent study reports the application of thermography and ML techniques for fault classification in PV modules. The study has adopted a texture feature analysis to study the features of various fault panel thermal images, and the developed algorithm was trained with 93.4% accuracy.

The performance of a trained model for a PV system using ML techniques can greatly vary if new data is fetched from a different environmental condition, especially data from the winter season. The irradiation level in the winter is substantially lower than that in the summer, and studies have shown faults occurring in such lower irradiation levels have higher chances of remaining undetected. Such undetected faults can cause a significant amount of power losses and degradation of the quality of the panel or even lead to deterioration of panels. We propose an intelligent fault diagnosis model for detecting faulty modules and further classifying the fault type that is applicable in all environmental conditions. The model uses the multilayer perceptron (MLP) and follows the supervised learning approach.

1.2 RELEVANCE

PV systems are subject to various faults and failures, and early fault detection of those faults and failures is very important for the efficiency and safety of the PV systems and to minimize the outage period and maximize the lifetime output.

1.3 MOTIVATION

Claimed to be the largest solar power plant in the world, the Bhadla Solar Park is located in Bhadla village, in Rajasthan's Jodhpur district. Spanning 14,000 acres, the fully operational power plant has been installed with a capacity of nearly 2,250 megawatts (MW). Such projects on this large scale require a smart and scientific approach to handle the faulty problems arriving at minute scale that can turn out difficult to detect and identify and hence, the main reason for our topic of project.

1.4 PROBLEM STATEMENT

PV systems are subject to various faults and failures, and early fault detection of those faults and failures is very important for the efficiency and safety of the PV systems.

1.5 OBJECTIVE STATEMENT

The objectives of this work are:

- To analyze data and predict electrical faults.
- To identify and classify fault types.
- To improve maximum efficiency power point tracking as well as power point tracking.

2.LITERATURE REVIEW

2.1 RESUME

Fault detection and timely troubleshooting are essential for the optimum performance in any power generation system, including photovoltaic (PV) systems. In particular, the goal for any commercial power-producing house is maximizing power production, minimizing energy loss and maintenance, cost, and the safe operation of the facility. Since PV systems are subject to various faults and failures, early detection of such faults and failures is very crucial for achieving the goal.

2.2 REVIEW OF PAPERS

S.NO.	Paper Title	Author Name	Relevance
1.	Detection and Prediction of Faults in Photovoltaic Arrays: A Review	Kais AbdulMawjood; Shady S. Refaat; Walid G. Morsi	This paper describes and explains all the types of faults that a PV array can encounter in its functioning. One has to know the faults first in order to detect, predict and rectify them in a proper way, with that point in mind this paper is explains and classifies the faults in an easy way.
2.	Identifying PV module mismatch faults by a thermography-based temperature distribution analysis	Yihua Hu; Wenping Cao; Jien Ma; Stephen J. Finney; David Li	Mismatch faults reduce the power output and cause potential damage to PV cells. This paper defines three fault categories in terms of fault levels, which lead to different terminal characteristics of the PV module.
3.	Solar Array Fault Detection using Neural Networks	Sunil Rao, Andreas Spanias and Cihan Tepedelenlioglu	While using unsupervised machine learning algorithms, a fault could be detected but not identified and unsupervised algorithms could not classify the type of fault. Neural networks allow not only detection but also identification or classification of the fault type with a high accuracy.
4.	Experimental studies of failure detection methods in PV module strings	Takumi Takashima; Junji Yamaguchi; Kenji Otani; Kazuhiko Kato	This paper tells us the application of TDR in a PV Module String.

2.3 SUMMARY

PV systems are subject to various faults and failures, and early fault detection of those faults and failures is very important for the efficiency and safety of the PV systems. ML-based fault detection models are trained with data and provide prediction results with very high accuracy. However, databased fault detection models for PV systems can sometimes give false predictions, especially when the environmental parameters are not taken into consideration. This paper developed an intelligent fault detection model for PV arrays based on PNN for accurately classifying the fault types. The model was trained with a large dataset containing different data values under different environmental conditions in the summer and the winter season. For the experimental verification, various fault state and normal state datasets are collected from 1.8kW (six 300W panels, 2 parallelly connected lines, each with 3 serially connected panels) into the grid-connected PV system. The experimental results demonstrate that the proposed method is superior in accurately predicting the result in cases where fault state and normal state are very hard to distinguish.

3.DESIGN AND DRAWING

3.1 INTRODUCTION

Design is mainly categorized into hardware and Software Design. Hardware is related to the circuit and its components while software design consists of programming of image processing using specific instruction. In this chapter we are going to discuss about block diagram, description of each block ,hardware used and software used.

❖ HARDWARE:

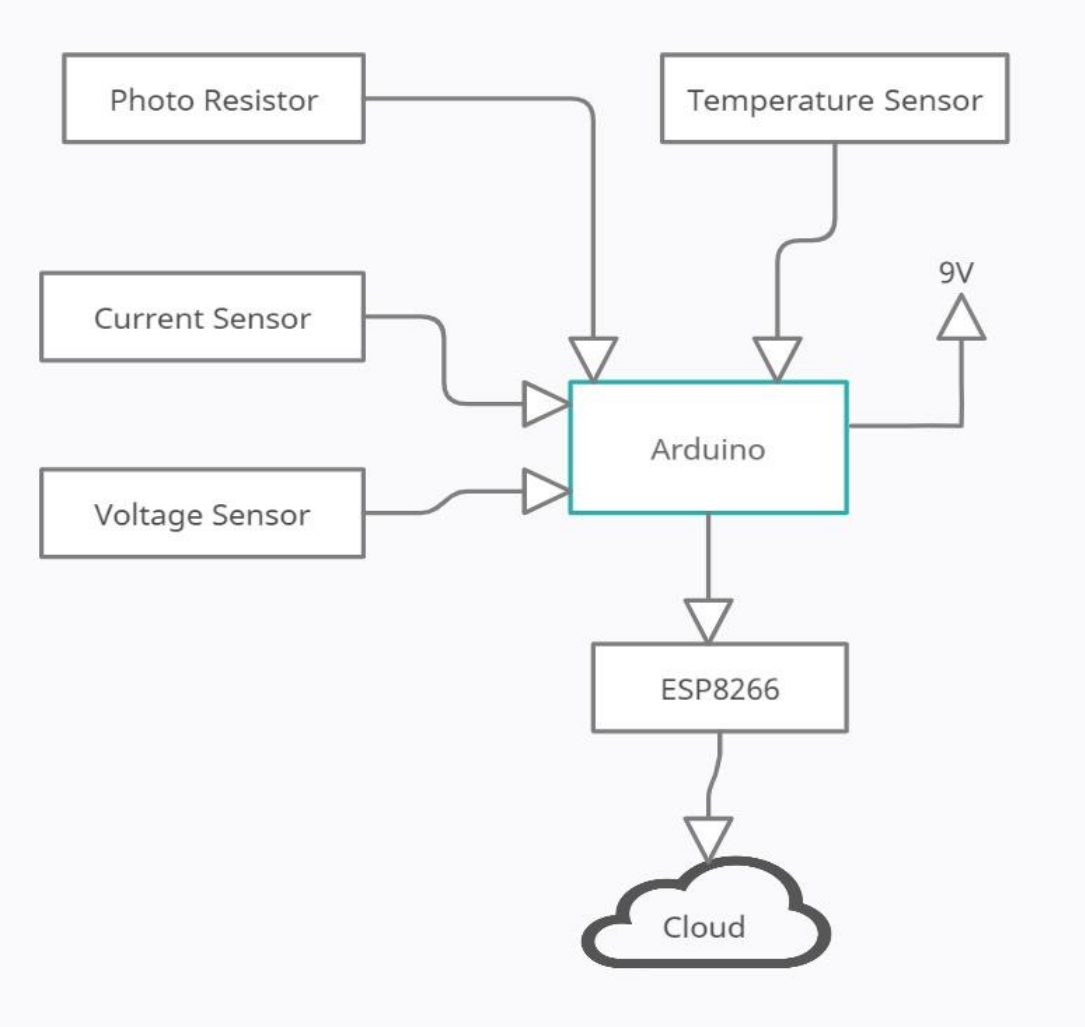


FIGURE 3.1 HARDWARE BLOCK DIAGRAM OF THE SYSTEM

Data Acquisition:

For building the model, we acquired the current, voltage, irradiation level, and temperature data from respective sensors attached to the PV array. The data is acquired at the ground level from various sensors including the current sensor, voltage sensor, temperature sensor and the photoresistor.

- **Current Sensor:** A current sensor is a device that detects electric current in a wire or conductor and generates a signal proportional to that current.

- **Voltage Sensor:** A voltage sensor is a sensor used to calculate and monitor the amount of voltage in an object. Voltage sensors can determine the AC voltage or DC voltage level.

- **Photoresistor:** A photoresistor is a passive component that decreases resistance with respect to receiving luminosity (light) on the component's sensitive surface.

- **Arduino:** The Arduino is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. All the data taken by different sensors is sent to Arduino.

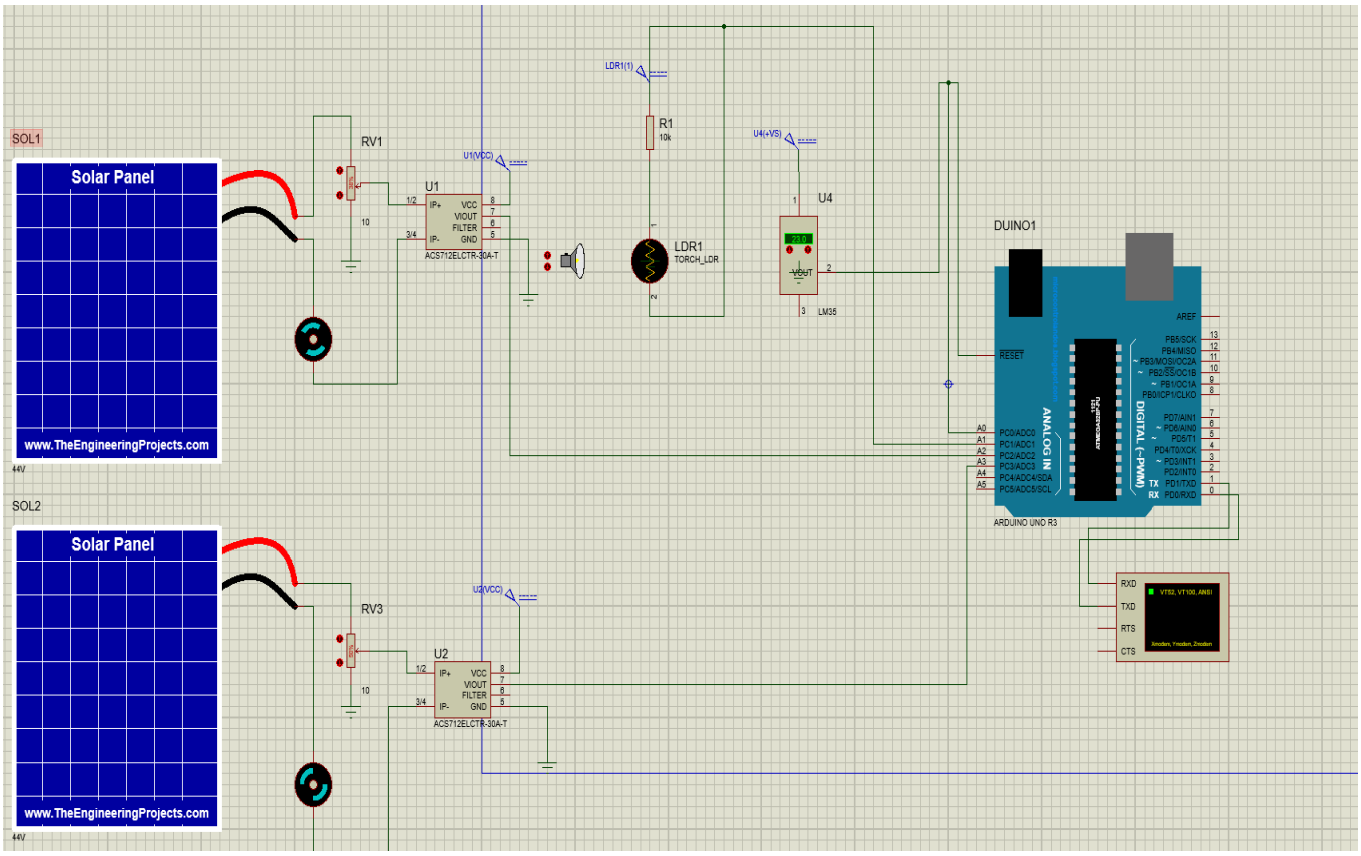
- **ESP8266:** The ESP8266 is a Wi-Fi microchip. Data is received from the Arduino and send to the cloud through ESP8266. The collected data is then essential and plays important role in determining the fault types and its location.

3.2 DESCRIPTION

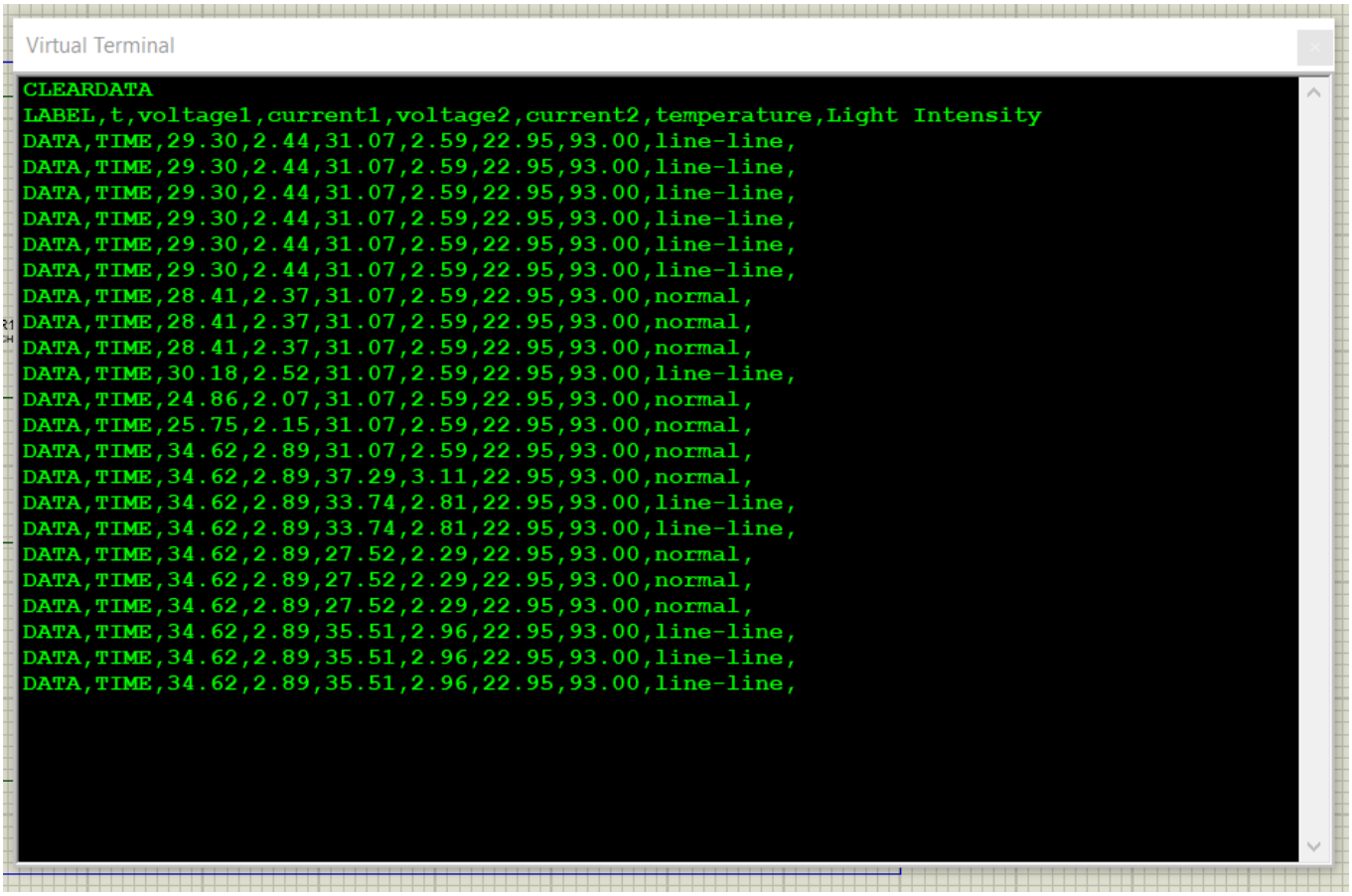
Before going to the actual implementation of the hardware it is easier to simulate and get the results before the actual implementations of the hardware, we have simulated the same circuit on the Proteus 8 Professional from Labcenter Electronics. This software allows use of almost all the circuits elements and simulate the circuit without any much difficulty. So we have used 2 solar panels of 44 V each and the current sensor that we have used is ACS712ELCTR-30A-T. This sensor is capable of measuring currents upto 30 amperes. Next for controlling the light intensity falling on the solar panel we have used potential dividers in series with the panel thus enabling the user to control the light falling on the panel by controlling the voltage across the panel. The same current sensor will also be used to measure the voltage across the panel using the ohm's law since we know the load resistance value. The measured voltage and current values are given to arduino to print them on serial monitor. Next for the temperature we have used the LM35 temperature sensor and a LDR sensor with adjustable light intensity bulb for the irradiance values. Both these sensors give the values to the arduino, which in return prints the acquired values on the serial monitor for display.

Then the programming is done for the arduino to detect the fault and specify the type of fault based on the current and voltage values captured by the sensors.

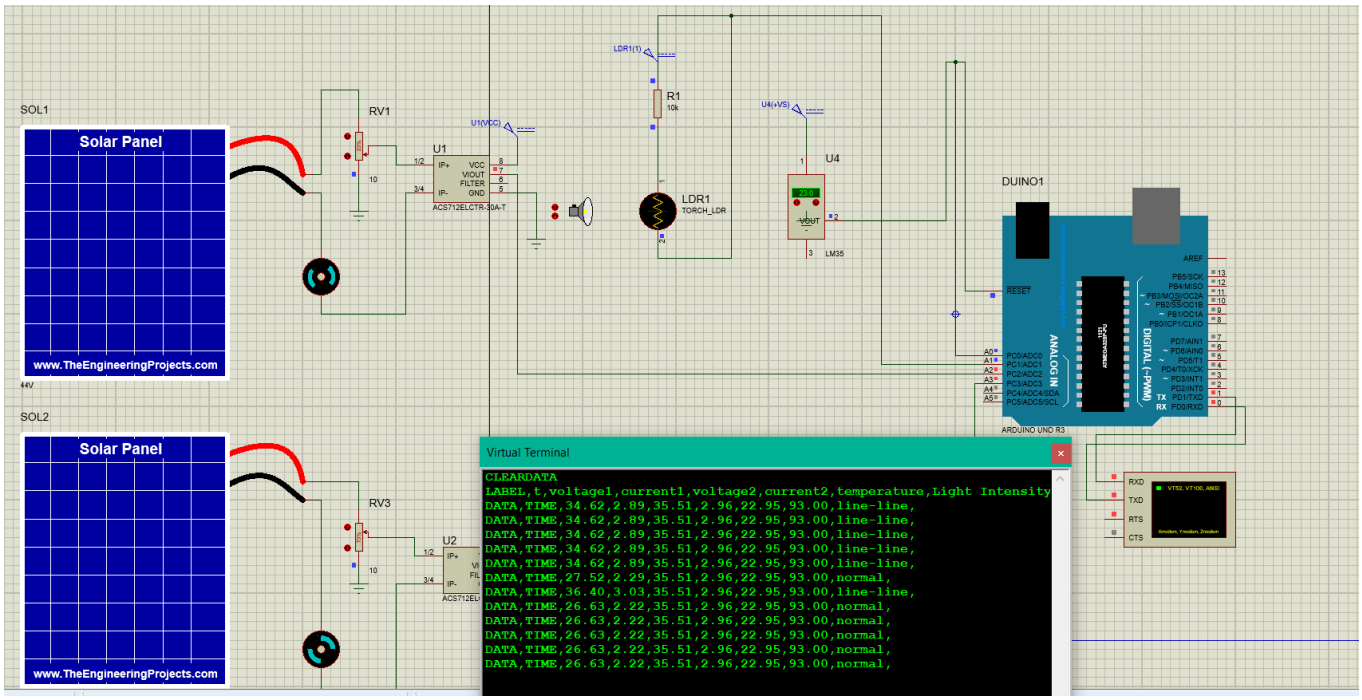
After the programming is done the arduino is able to detect the faults and classify them and hence the data can be generated with different temperature and irradiance values. We then saved copied the data from serial monitor to excel sheet and the data sheet is generated to train the model and test it later.



The above image shows the circuit configuration used in the proteus software for the simulation of the generation of voltage and current values. The solar panel is very clear in the image and the black circle is a motor which is used as the load for the solar panel. The current sensor is connected right infront of the solar panel. The temperature and irradiance sensor are also connected which are present just left side of the arduino. The black circle with zigzag line inside with a left to its left side is the LDR sensor which gathers the irradiance values and a rectangular box to the right of the LDR sensor is the temperature sensor. A black box below the arduino helps to activate and display the values on the virtual terminal of the arduino. When the simulation is run the different values of voltage and current are displayed on the virtual terminal of the arduino.



As it is evident from the image shown above that the virual terminal shows the simulation results in the form of voltage and current values and the state of the panel along with the temperature and irradiance values. Now these values are copied and then put in the excel sheet and then fed to the ML model for the training and testing purposes.



And here is the final image showing the simulation running and virtual terminal shown at the same time. One thing to note here is that we have used a potential divider for changing the irradiance values of the solar panel and only those values can be changed while the simulation is running. This enables us to create different day time irradiance conditions for the panel and generating different values of the voltages and currents and capture the readings.

❖ SOFTWARE:

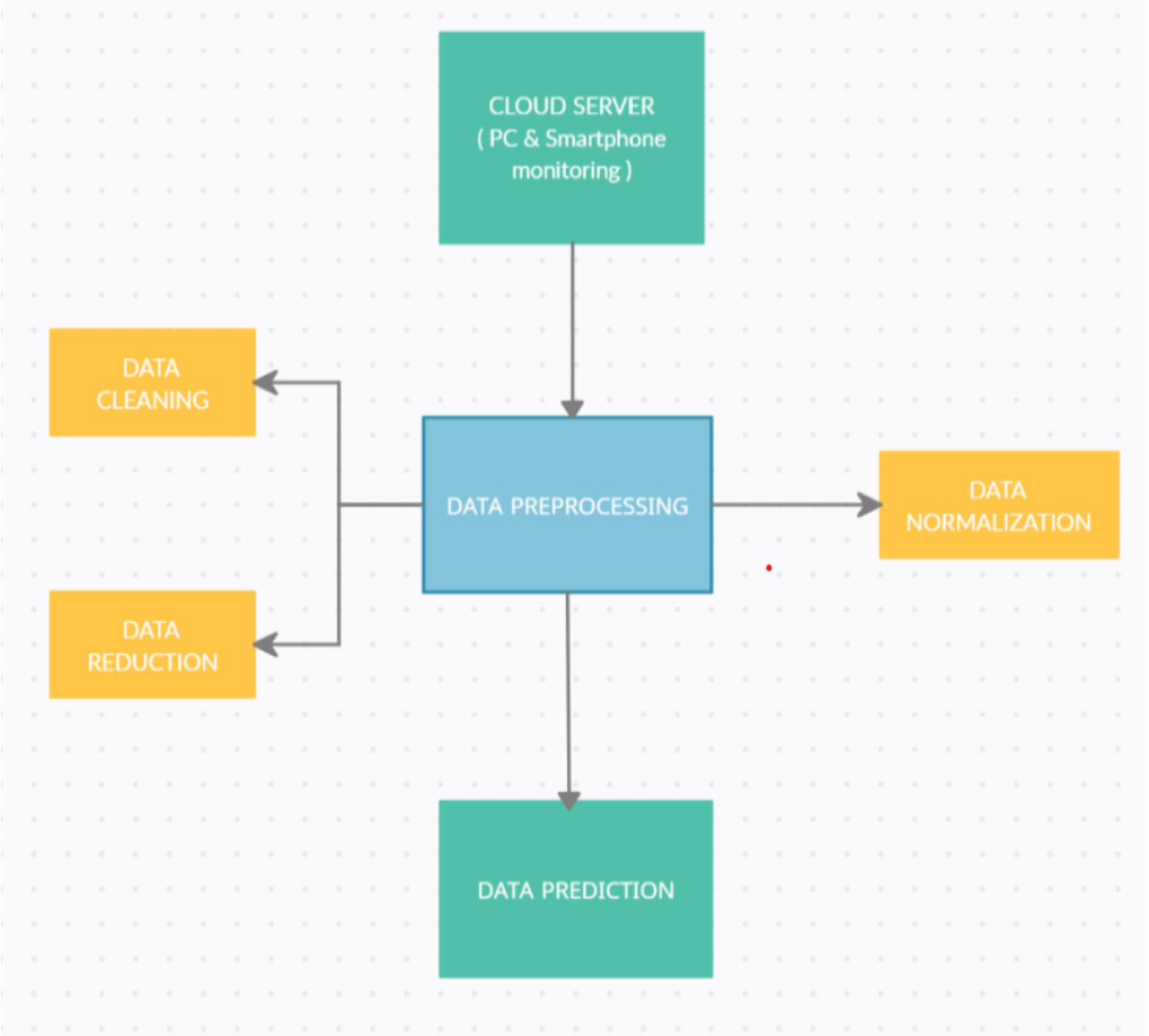


FIGURE 3.2 SOFTWARE FLOWCHART OF THE SYSTEM

Data Pre-processing:

Data preprocessing is the second layer in the proposed system architecture. It consists of all the actions taken before the data inputs are fetched to the model for extracting features.

Components of Data Preprocessing consists of following blocks:

- **Data Cleaning:**

  Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. Therefore, in this step, we will remove all the impurities (i.e., removing duplicate values, handling missing data etc.) from the data. This data is usually not necessary or helpful when it comes to analyzing data because it may hinder the process or provide inaccurate results. There are several methods for cleaning data depending on how it is stored along with the answers being sought. Data cleaning is not simply about erasing information to make space for new data, but rather finding a way to maximize a data set's accuracy without necessarily deleting information.

- **Data Reduction:**

  Data reduction is the transformation of numerical or alphabetical digital information derived empirically or experimentally into a corrected, ordered, and simplified form. In this step, we will reduce our data size, by using technique dimensionality reduction, as it eliminates outdated or redundant features. Data reduction can increase storage efficiency and performance and reduce storage costs. Data reduction reduces the amount of data that is stored on the system using a number of methods. The system supports data reduction pools, which contain thin-provisioned, compressed, and deduplicated volumes.

- **Data Normalization:**

  Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to use a common scale (commonly used scale is 0-1), without distorting differences in the ranges of values or losing information. Normalization avoid problems by creating new values that maintain the general distribution and ratios in the source data, while keeping values within a scale applied across all numeric columns used in the model.

Data Prediction:

- After preprocessing we will feed the data into our classification model for training purpose and will find the accuracy of the model. Firstly we feed data into the SVM classification model. Support Vector Machine (SVM) takes data points and outputs the hyperplane (which in two dimension is simply a line) that best separates the data points. This line is the decision boundary: anything that falls to one side of it will classify as positive/negative data point, and anything that falls on the other side will be classified as negative/positive data points.
- Best hyperplane is that which maximizes the margin from both the data points. In other words, the hyperplane whose distance to the nearest element of each data point is the largest.
- Since SVM works best on text classification problem, and in our project we have to classify three labels which are open, line-line and normal. Therefore, we have used SVM.

SOFTWARE MODELS:

The dataset after being pre-processed was fed to various models such as: Random Forest, DecisionTree, KNeighbors and Gaussian. To avoid overfitting, data was divided in train and test in preprocesssing with train datapoint being 75% and training the model at the remainig 25%.

- **Random Forest Classifier**

```
In [18]: forest_clf = RandomForestClassifier(n_estimators=50)
         forest_clf.fit(X_train, y_train)
         pred_forest = forest_clf.predict(X_test)

In [20]: accuracy_score(y_test,pred_forest)

Out[20]: 0.9988440821080216
```

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

As random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction and we have limited number of features hence a decison tree is suitable here.

- **Decision Tree Classifier**

```
In [22]: DT_clf = DecisionTreeClassifier()
         DT_clf.fit(X_train, y_train)
         pred_DT = DT_clf.predict(X_test)
         print("Accuracy:",accuracy_score(y_test, pred_DT))

         Accuracy: 0.9981336439074101
```

Decision tree learning or induction of decision trees is one of the predictive modelling approaches used in statistics, data mining and machine learning. It uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. Decision trees are among the most popular machine learning algorithms given their intelligibility and simplicity.

Since we have already used random forest which is one of the best decison tree so we have implemented it with this.

- **KNN Classifier**

```
In [23]: KNN_clf = KNeighborsClassifier()
         KNN_clf.fit(X_train, y_train)
         pred_KNN = KNN_clf.predict(X_test)
         print("Accuracy:",accuracy_score(y_test, pred_KNN))

         Accuracy: 0.9970417819187655
```

The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It's easy to implement and understand.

Since its purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point and we have limited number of classes that are 5 this classifier was used.

- **Gaussian naive base Classifier**

```
In [25]: bayes_clf = GaussianNB()
         bayes_clf.fit(X_train, y_train)
         pred_bayes = bayes_clf.predict(X_test)
         print("Accuracy:",accuracy_score(y_test, pred_bayes))

         Accuracy: 0.7670490609986897
```

Naive Bayes are a group of supervised machine learning classification algorithms based on the Bayes theorem. It is a simple classification technique, but has high functionality. Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

Since other functions can be used to estimate the distribution of the data, but the Gaussian (or Normal distribution) is the easiest to work with because you only need to estimate the mean and the standard deviation from your training data.

Using K-fold Cross Validation -

Out of the classifiers used Decision tree, KNN and Random forest has shown promissing accuracies. With Random Forest leading with 99.8%. But the data being unbaised with many classes in the last 25%; hence it was required to shuffle the data randomly and split the data set in groups. Here, k-fold Cross Validation was done with k=10.

- **Decision Tree**

```
In [17]: DTree_clf = DecisionTreeClassifier()
         print(cross_val_score(DTree_clf, X, y, cv = 10, scoring = 'accuracy').mean())

         0.9668022470819995
```

Here, after doing cross validation with cv=10 we have accuracy of 96.6%.

- **KNN**

```
In [18]: K_NN_clf = KNeighborsClassifier()
         print(cross_val_score(K_NN_clf, X, y, cv = 10, scoring = 'accuracy').mean())

         0.9609702116708373
```

After doing cross validation in KNN with cv=10 we have an accuracy of 96%.

- **Random Forest**

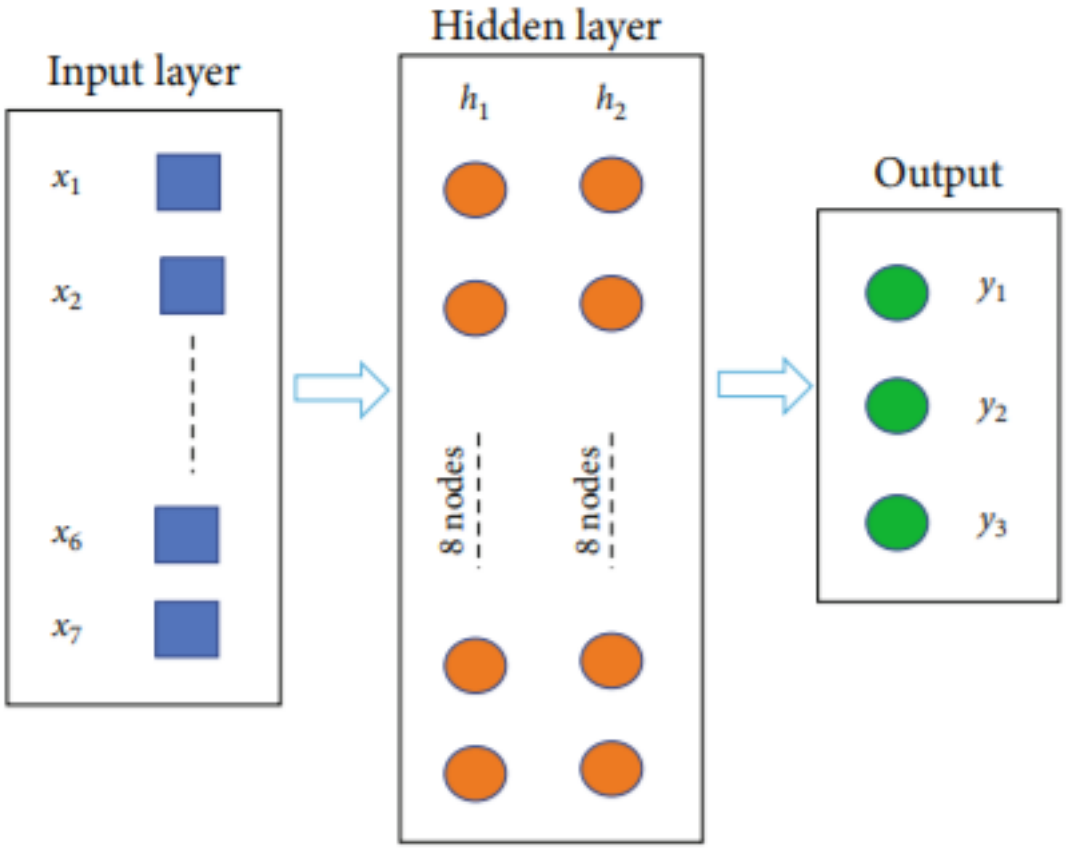
```
In [14]: Randforest_clf = RandomForestClassifier(n_estimators=50)
         print(cross_val_score(Randforest_clf, X, y, cv = 10, scoring = 'accuracy').mean())

         0.9732930058190219
```

After applying cross validation to this model we have an average accuracy of 97.4%.

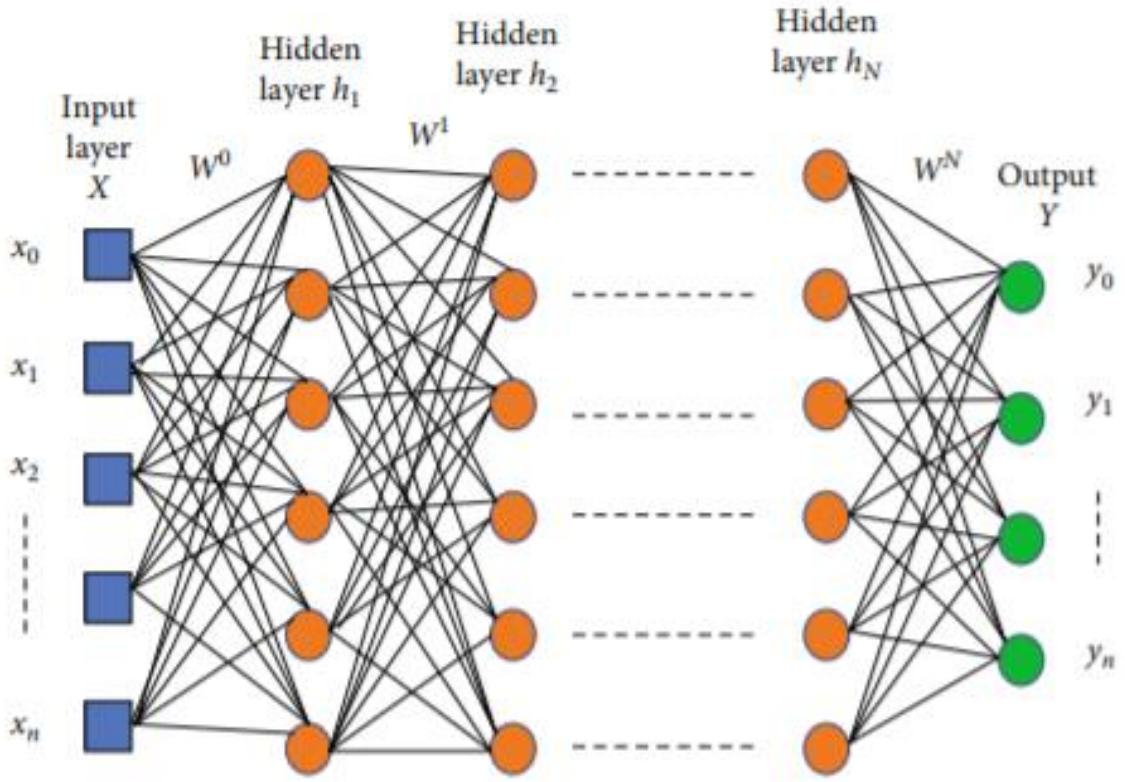
Upon training the Decision tree, KNN and Random forest with cross validation it was observed that the accuracy percentage was reduced by 2-3%. This was due to the pressence of unbaised dataset hence the accuracy result after cross validation is the accurate one. The highest accuracy observed was of Random Forest: with an average of 97.45%.

NEURAL NETWORKS

The performance of a trained model for a PV system using ML techniques can greatly vary if new data is fetched from a different environmental condition, especially data from the winter season. The irradiation level in the winter is substantially lower than that in the summer, and studies have shown faults occurring in such lower irradiation levels have higher chances of remaining undetected.



Such undetected faults can cause a significant amount of power losses and degradation of the quality of the panel or even lead to deterioration of panels. We propose an intelligent fault diagnosis model for detecting faulty modules and further classifying the fault type that is applicable in all environmental conditions. The model uses the multilayer perceptron (MLP) and follows the supervised learning approach. It is robustly trained with historical data of different faulty and normal states in different environmental conditions especially focusing on winter. The data was collected from gird-connected PV system located in Portugal.



DNN / MLP

```
Epoch 25/30
1099037/1099037 [==============================] - 176s 160us/step - loss: 0.0669 - accuracy: 0.9831 - val_loss: 0.0503 - val
_accuracy: 0.9872
Epoch 26/30
1099037/1099037 [==============================] - 1381s 1ms/step - loss: 0.0664 - accuracy: 0.9827 - val_loss: 0.0492 - val_
accuracy: 0.9868
Epoch 27/30
1099037/1099037 [==============================] - 488s 444us/step - loss: 0.0621 - accuracy: 0.9830 - val_loss: 0.0379 - val
_accuracy: 0.9909
Epoch 28/30
1099037/1099037 [==============================] - 490s 445us/step - loss: 0.0576 - accuracy: 0.9836 - val_loss: 0.0407 - val
_accuracy: 0.9881
Epoch 29/30
1099037/1099037 [==============================] - 485s 441us/step - loss: 0.0545 - accuracy: 0.9840 - val_loss: 0.0561 - val
_accuracy: 0.9824
Epoch 30/30
1099037/1099037 [==============================] - 486s 442us/step - loss: 0.0576 - accuracy: 0.9832 - val_loss: 0.0513 - val
_accuracy: 0.9849
dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
```

Fig: Result of Multi-layer Perceptron of 30 Epochs.

Confusion Matrix

```
Out[21]: Text(0.5, 1.0, 'Confusion Matrix')
```

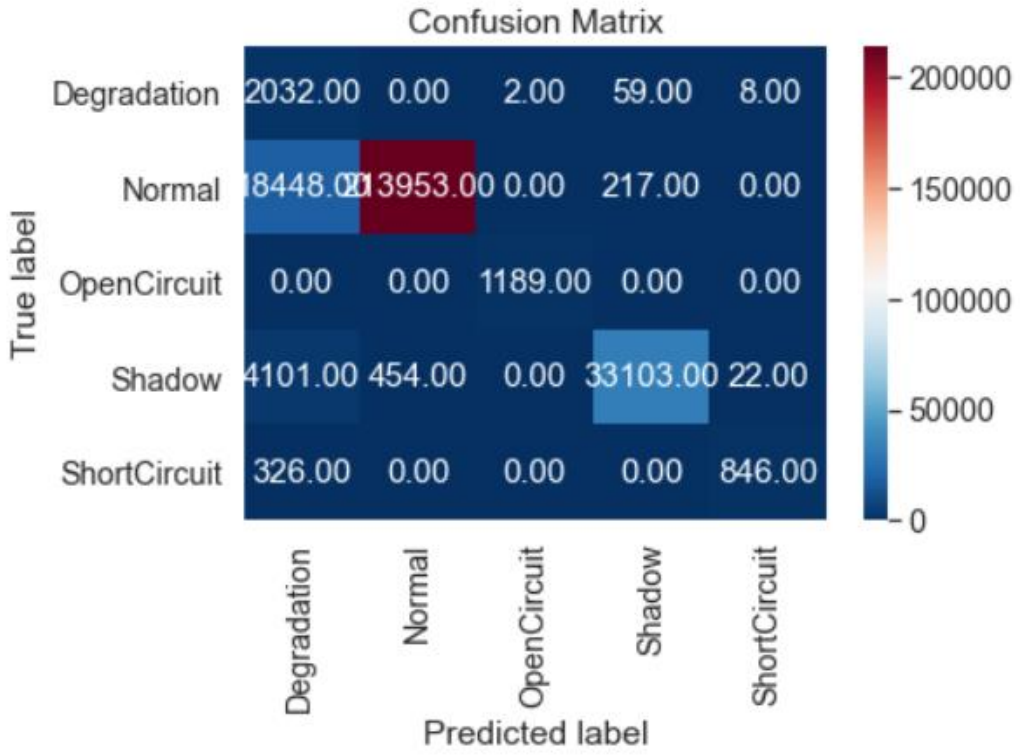


Fig: depiction of correlation between predicted and True labels of various states.

Multilayer Perceptron and Feature Extraction:

A multilayer perceptron (MLP) or probabilistic neural network (PNN) is a nonlinear learning algorithm in ML and is widely applied in both supervised and unsupervised learning. However, most of its application is found in the classification problem of supervised learning.

$$\Phi_{ij}(y) = \frac{1}{(2\pi)^{1/2}\omega^d}\frac{1}{d}\sum_{1}^{d} e^{-\frac{(y-y_{ij})(y-y_{ij})^T}{\omega^2}}$$

Here, Φij(y) is the probability density function of input vector y, d is the total category number of training samples, y(ij) is the jth training center of the ith type of samples, and ω is the smoothing factor. Assuming that we used an input layer with n0 neurons, input layer X can be given as:

$$X = (x_0, x_1, \cdots, x_n)$$

For the feature extraction, the hidden layer is designed with two layers: h1 being the first hidden layer and h2 being the second hidden layer. Each of the input dimensions (x1 to x6) is fed to h1, and then, the output from h1 goes to h2. The outputs hji of neurons in the hidden layers are computed as:

$$h_i^j = f\left(\sum_{k=1}^{n_{i-1}} W_{k,j}^{i-1} h_{i-1}^k\right), \quad i = 2, \cdots, N, \boxtimes j = 1, \cdots, n_i$$

where Wi−1(k,j) is the weight between the neuron k in the hidden layer I and neuron j in the hidden layer +1, n(i) is the number of the neurons in the ith hidden layer. Both of the hidden layers use uniformdistribution as the kernel initializer for initializing the weights in the network. Also, we chose ReLU (Rectified Linear Unit) as the activation function because of its several advantages in nonlinear datasets with multiple dimensions.

$$y = \max(0, x)$$

The output layer consists of three layers: y1, y2, y3, y4 and y5. The network outputs are computed as:

$$y_i = f\left(\sum_{k=1}^{n_N} W_{k,j}^N h_N^k\right)$$

Finally, the model is fitted to train with a batch size of 5 with 30 epochs. The table shows the different parameters used to construct the MLP as the fault classifier.

where wN(k,j) is the weight between the neuron k in the N(th) hidden layer and the neuron j in the output layer and n(N) is the number of the neurons in the Nth hidden layer. The output layer also uses uniform distribution as the kernel initializer, but unlike hidden layers, it uses Softmax as the activation function to represent the logits into probabilities.

The Softmax function is given as:

$$F(X_i) = \frac{\exp\ (X_i)\ i = 0, 1, 2, \cdots k}{\sum_{j=0}^{k} \exp(x_j)}.$$

Because of the nature of classification, we have used categorical crossentropy as the loss function given in equation where ŷ is the predicted output.

$$L(y, \hat{y}) = -\sum_{j=0}^{M}\sum_{i=0}^{N}\left(y_{ij} * \log\ \left(\hat{y}_{ij}\right)\right)$$

Categorical crossentropy will compare the distribution of the predictions (the activations in the output layer, one for each class) with the true distribution, where the probability of the true class is set to 1 and 0 for the other classes. Among many other optimizers, we used Adam (Adaptive Moment Estimation) for optimizing the proposed model. Adam uses adaptive learning for each of the parameters, and the weight of a learning rate is divided by a running average of recent gradients.

Parameters	Values
Algorithm	Stochastic gradient descent
Activation Function	ReLu(input, hidden layers) Softmax(output layer)
Layers	5 hidden layers with 8 units each, 5 units at the output
Loss Function	Categorical Crossentry
Optimizer	Adam
Data Split	Train: 80%, Test: 20%
Batch size	5 (with 30 epochs)
Tuning	k-fold crossvalidation, dropout

DNN / MLP:

```
Epoch 25/30
1099037/1099037 [==============================] - 176s 160us/step - loss: 0.0669 - accuracy: 0.9831 - val_loss: 0.0503 - val
_accuracy: 0.9872
Epoch 26/30
1099037/1099037 [==============================] - 1381s 1ms/step - loss: 0.0664 - accuracy: 0.9827 - val_loss: 0.0492 - val_
accuracy: 0.9868
Epoch 27/30
1099037/1099037 [==============================] - 488s 444us/step - loss: 0.0621 - accuracy: 0.9830 - val_loss: 0.0379 - val
_accuracy: 0.9909
Epoch 28/30
1099037/1099037 [==============================] - 490s 445us/step - loss: 0.0576 - accuracy: 0.9836 - val_loss: 0.0407 - val
_accuracy: 0.9881
Epoch 29/30
1099037/1099037 [==============================] - 485s 441us/step - loss: 0.0545 - accuracy: 0.9840 - val_loss: 0.0561 - val
_accuracy: 0.9824
Epoch 30/30
1099037/1099037 [==============================] - 486s 442us/step - loss: 0.0576 - accuracy: 0.9832 - val_loss: 0.0513 - val
_accuracy: 0.9849
dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
```

Fig: Result of Multi-layer Perceptron of 30 Epochs.

Confusion Matrix:

```
Out[21]: Text(0.5, 1.0, 'Confusion Matrix')
```

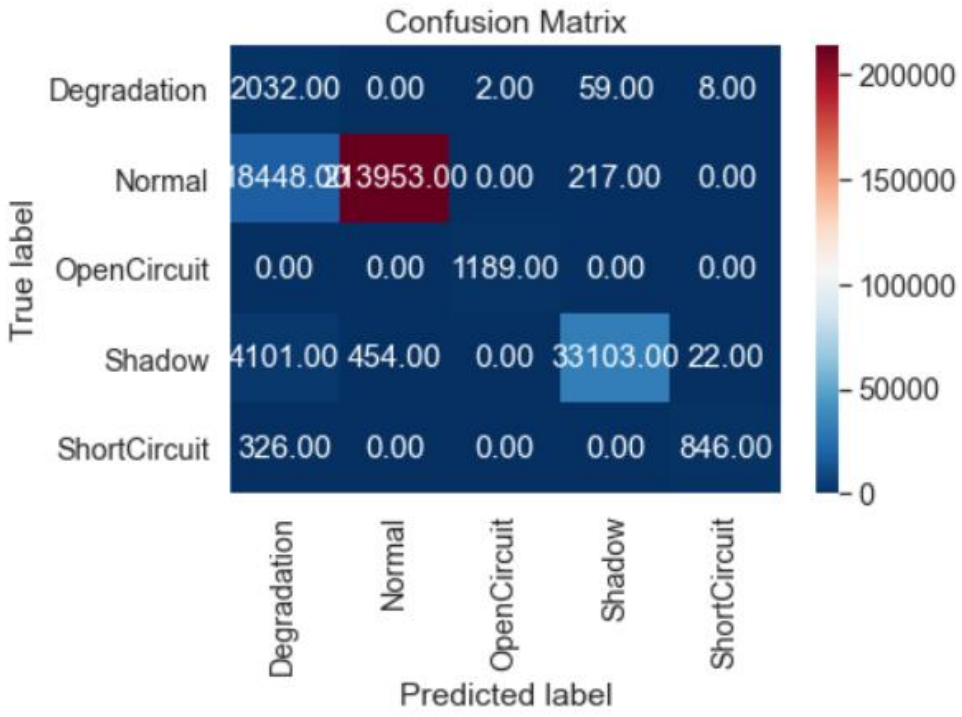


Fig: depiction of correlation between predicted and True labels of various states.

To check the bias-variance tradeoff, a k-fold crossvalidation test is performed with the 5 validations split into the training data. Also, for improving the model and reducing overfitting, we implemented the dropout regularization technique. The dropout rates of 0.1 and 0.2 were selected for the first and second hidden layers, respectively. The result of the evaluation, improvement, and tuning of the model is provided:

```
In [17]: def build_classifier():
             model = Sequential()
             model.add(Dense(input_dim = 6, units = 8, kernel_initializer = 'uniform', activation = 'relu'))
             model.add(Dense(units = 8, kernel_initializer = 'uniform', activation = 'relu'))
             model.add(Dense(units = 5, kernel_initializer = 'uniform', activation = 'softmax'))
             model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
             return model

         model = KerasClassifier(build_fn = build_classifier, batch_size = 5, epochs = 20)
         accuracies = cross_val_score(estimator = model, X = X_train, y = y_train, cv = 5) #CV = K-fold cross val. splits
         mean = accuracies.mean()
         variance = accuracies.std()
         print("Average accuracy:", mean, "Variance:", variance)
         Epoch 12/20
         879230/879230 [==============================] - 117s 133us/step - loss: 0.0588 - accuracy: 0.9822
         Epoch 13/20
         879230/879230 [==============================] - 117s 133us/step - loss: 0.0540 - accuracy: 0.9828
         Epoch 14/20
         879230/879230 [==============================] - 117s 133us/step - loss: 0.0552 - accuracy: 0.9835
         Epoch 15/20
         879230/879230 [==============================] - 117s 133us/step - loss: 0.0493 - accuracy: 0.9840
         Epoch 16/20
         879230/879230 [==============================] - 117s 133us/step - loss: 0.0538 - accuracy: 0.9839
         Epoch 17/20
         879230/879230 [==============================] - 117s 133us/step - loss: 0.0502 - accuracy: 0.9842
         Epoch 18/20
         879230/879230 [==============================] - 117s 133us/step - loss: 0.0455 - accuracy: 0.9849
         Epoch 19/20
         879230/879230 [==============================] - 117s 133us/step - loss: 0.0444 - accuracy: 0.9852
         Epoch 20/20
         879230/879230 [==============================] - 117s 133us/step - loss: 0.0462 - accuracy: 0.9853
         219807/219807 [==============================] - 12s 56us/step
         Average accuracy: 0.9780726313591004 Variance: 0.005480874747660622
```

Fig: Result of MLP of 20 Epochs with cv = 5

4. BUDGET OF THE PROJECT

Sr.No	Name	Description	Quantity	Price per item	Total price (in Rs.)
1.	Arduino	Uno	1	400	400
2.	Temperature sensor	DS18B20	5	35	157
3.	Wires	Jumper MTM,FTM,FTF		1.4	175
4.	ESP8266		1	280	280
5.	Digital Light Intensity sensor	BH1750L	1	260	260
6.	Solar Panel	50W 12V	2	2400	4800
7.	Voltage Detection sensor module	Robodo VOLTSENS	1	200	200
8	Current Detection sensor module	WCS1500 200A	1	849	849
	Total				**7121**

5. WORK PROGRESS

List of activities	August	Septemb-er	October-Novemb-er	Decembe r-January	February - March	April-May
Working on the idea						
Literature survey						
Brainstor-ming the project Develop-ment						
Developi-ng the model						
Training of the model						
Modelling and evaluation						

6. SUMMARY

PV systems are subject to various faults and failures, and early fault detection of those faults and failures is very important for the efficiency and safety of the PV systems. ML-based fault detection models are trained with data and provide prediction results with very high accuracy. However, databased fault detection models for PV systems can sometimes give false predictions, especially when the environmental parameters are not taken into consideration. This paper developed an intelligent fault detection model for PV arrays based on PNN for accurately classifying the fault types. The model was trained with a large dataset containing different data values under different environmental conditions in the summer and the winter season. For the experimental verification, various fault state and normal state datasets are collected from 1.8kW (six 300W panels, 2 parallelly connected lines, each with 3 serially connected panels) into the grid-connected PV system. The experimental results demonstrate that the proposed method is superior in accurately predicting the result in cases where fault state and normal state are very hard to distinguish.

7. REFERENCES

1 D.A.Shaikh,GhoraleAkshayG,Chaudhari Prashant , Kale Parmeshwar L"India Intelligent Autonomous Farming Robot with Plant Disease Detection usingImage Processing", International Journal of Advanced Research in Computer andCommunicationEngineeringVol.5,Issue4,April2016.

2 S.LagadandS.Karmore,"Designanddevelopmentofagrobotforpesticidespray-ingusinggradingsystem",InternationalconferenceofElectronics,CommunicationandAerospaceTechnology(ICECA),Coimbatore,pp.279-283,2017.

3 Arjun Prakash R, Bharathi G B,Manasa V and Gayathri S "Pesticide Spray-ing Agricultural Robot", International Research Journal of Power and EngineeringVol.3(2),pp.056-060,November,2017.

4 JayaprakashSethupathy1,VeniS"OpenCVBasedDiseaseIdentificationofMangoLeaves", International Journal of Engineering and Technology (IJET) Vol 8 No 5Oct-Nov2016

5 S.K.Pilli,B.Nallathambi,S.J.GeorgeandV.Diwanji,"eAGROBOT-Arobotfor early crop disease detection using image processing", International ConferenceonElectronicsandCommunicationSystems(ICECS),Coimbatore,2014