SOLAR POWER PREDICTION USING ARTIFICIAL INTELLIGENCE

A PROJECT REPORT
submitted by

CB.EN.U4EEE16145 RAHUL ASHWIN M

in partial fulfillment for the award of the degree
of
BACHELOR OF TECHNOLOGY
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

AMRITA SCHOOL OF ENGINEERING, COIMBATORE

AMRITA VISHWA VIDYAPEETHAM
COIMBATORE- 641112
May 2020

AMRITA VISHWA VIDYAPEETHAM
AMRITA SCHOOL OF ENGINEERING, COIMBATORE, 641112

BONAFIDE CERTIFICATE

This is to certify that the project report entitled, **SOLAR POWER PREDICTION USING ARTIFICIAL INTELLIGENCE** submitted by

NAMES OF THE STUDENTS	**REG NOS**
Rahul Ashwin M	CB.EN.U4EEE16145

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in ELECTRICAL & ELECTRONICS ENGINEERING** is a bonafide record of the work carried out under my guidance and supervision at Amrita School of Engineering, Coimbatore.

SUPERVISOR:
DR RESMI R
Assistant Professor (Sr.Gr),Electrical and Electronics Engineering Department, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore, Tamil Nadu, India.

SUPERVISOR:
DR R JAYABARATHI
Associate Professor, Electrical and Electronics Engineering Department, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore, Tamil Nadu, India.

CHAIRPERSON
DR S BALAMURUGAN
Head of department, Electrical and Electronics Engineering Department, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore, Tamil Nadu, India.

ABSTRACT

The electricity demand in the world is always increasing. However, the traditional source of fossil fuel is limited and it leaves a significant carbon foot print. The generation and load balance are required in the economic scheduling of generating unit in the smart grid. With the increasing penetration of photovoltaic power system into the utility network, predicting the power output of a PV installation is important from resource planning perspective. The forecast of power output, is a challenging task for PV power system as the power output varies largely with the external conditions like sunshine, temperature, etc. In present Indian scenario solar energy is directly fed to the grid without checking the status of the grid since the grid is always deprived of power. There are various machine learning techniques available for time series analysis and forecasting. This project focuses on dealing with solar forecast system with a prototype of small scale which will then be used for the forecasting in real time. The weather parameters include temperature, humidity, pressure and irradiance. The weather and solar data are stored in a real time database and the machine leaning model is run on Google Colab with GPU support. A scalable model of Random Forest is run in the cloud with data acquired from the installed sensors. The dependent and independent variables are trained and tested. The forecasted power output acquired from the trained model is displayed graphically in an app created using Wireframe. The real-time data and the forecasted data stored in the user database will be displayed in the app.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

S.No	SYMBOL	DEFINITION
1	X	Input training vector
2	T	Target output vector
3	A	Learning rate parameter
4	x_i	Input unit i
5	v_{0j}	Bias on jth hidden unit
6	w_{0k}	Bias on kth output unit
7	z_j	Hidden unit j
8	$F(x)$	Sum of squares of nonlinear functions
9	$J_i(x)$	Jacobian of $f_i(x)$
10	λ_k	Non negative scalars
11	I	Identity matrix
12	p_k	Solution of the constrained sub problem
13	$\mathbf{c}_t$	Cell state
14	$\mathbf{f}_t$	Forget Gate
15	$\mathbf{i}_t$	Input Gate
16	$\mathbf{o}_t$	Output Gate
17	$\mathbf{c}_{t-1}$	Initial Hidden State
18	$\mathbf{h}_{t-1}$	Initial Input State
19	$\mathbf{x}_t$	Present input state

20	$\mathbf{h}_t$	New output state

LIST OF ABBREVIATIONS

S.No	ABBREVIATION	EXPANSION
1	IRENA	International Renewable Energy Agency
2	PV	Photovoltaic
3	ANN	Artificial Neural Network
4	NWP	Numerical Weather Prediction
5	ARIMA	Autoregressive Integrated Moving Average
6	NNs	Neural Networks
7	SVR	Support Vector Regression
8	k-NN	k-nearest neighbor
9	MSE	Mean square error
10	RNN	Recurrent neural network
11	LSTMs	Long short-term memory networks
12	CTC	Connectionist Temporal Classification
13	HMM	Hidden Markov models
14	MQTT	Message Queuing Telemetry Transport
15	IoT	Internet of Things
16	Json	JavaScript Object Notation
17	Csv	Comma separated values

Chapter 1
INTRODUCTION

1.1 INTRODUCTION

Energy production and its usage play a major role for the development of a country. In this project work we are shifting our focus from non-renewable resources like coal and gas to clean energy sources like wind energy and solar energy. According to International Renewable Energy Agency (IRENA) India is the lowest cost producer of solar power globally. There is need for solar power prediction for better planning and utilization in the generation side. In present Indian scenario, solar energy is directly fed to the grid without checking the status of the grid since the grid is always deprived of power. No communication exists between the consumers and utility other than smart meter. Also like any other renewable resources, solar energy is inherently uncertain as it is heavily dependent on solar irradiance and other environmental factors like humidity, temperature and geographic location. As a result, forecasting plays a significant role in PV based systems for operation and planning purposes. Accurate solar energy generation forecast can help with reducing the uncertainty and result in better demand side management.

Solar energy is becoming increasingly popular since it is environmentally friendly, easily available and cost-effective. Solar power generated by Photovoltaic (PV) panels is growing especially due to the recent advances in PV technologies and the decreasing prices of PV rooftop panels. Solar energy obtained from solar power module is fluctuating in nature. This fluctuating nature has a great impact on the PV system planning, operation, and its economic analysis. This variability creates challenges for the large-scale integration of solar power in the electricity grid and motivates the development of methods for accurate prediction of the solar power output.

One of the technical problems in the management of PV systems is the reverse power flow from PV systems to bulk power systems, which can lead malfunctions of the systems and economic losses. Therefore, it is important to precisely predict the amount of electrical power generation from PV systems. Thus, it is advisable to have a prediction method for photovoltaic power system. Currently, a number of methods are being applied for photovoltaic power generation prediction. In terms of theory and methodology, they are classified into three categories; neural network (NN) based model, time series model, and Machine Learning model. Among these models random forest has high prediction accuracy for our test data.

The existing approaches for PV power forecasting can be classified based on different criteria. Depending on the way the prediction is done, there are two groups: indirect and direct. The indirect approaches rely on meteorological data from weather Models, Numerical Weather Prediction (NWP) and satellite images showing the movement of clouds to predict the solar irradiance and other meteorological variables. These predictions are then converted into PV power predictions, using the characteristics of the PV plant. The main disadvantage of these methods is that the required meteorological information is not always available for the PV site, and not at the required resolution level, which limits their applicability for highly accurate forecasts. In contrast, the direct approaches does not convert solar irradiance into solar power instead it directly predict the PV power output from previous PV power and weather data, and

weather forecasts for the future days. Based on the prediction method used, there are also two main groups of approaches: using statistical methods such as Autoregressive Integrated Moving Average (ARIMA) and using machine learning methods such as Neural Networks (NNs), Support Vector Regression (SVR) and k-nearest neighbor (k-NN). The machine learning methods usually show better performance than the statistical methods.

Chapter 2
LITERATURE REVIEW

The literatures considered to be relevant for the completion of this project work are included in this section.
Chow et al. [1] applied NNs to directly predict the solar power output for 10-20 min forecasting horizon. To train the NNs, it is used as inputs both celestial data (solar elevation and azimuth angles) and meteorological data (dry bulb temperature and solar radiation).

Liu et al. [2] also developed a NN based approach but they used as inputs the power output, aerosol index, temperature, humidity and wind speed from previous similar day, and the weather predictions for the targeted day. Both approaches achieved promising accuracy.

Malik et al. [3] A well-known physical approach for solar power forecasting is analytical PV power forecasting method which is based on appropriate theoretical models The paper presents the results in two stages, firstly theoretical predicted ground level irradiance is calculated without any attenuation factor and in second stage the irradiance is adjusted using the cloud attenuation model ($C.I_{GH}$) proposed in this paper.

De Giorgi et al. [4] statistical methods based on multi-regression analysis and the Elmann articial neural network (ANN) are developed in order to predict power of a PV plant. This model was trained using three different input vectors. First input vector only includes solar power data, second vector consists of both solar power data and solar irradiance. Third vector contains solar power data and temperature data along with solar irradiance. Best predication was done when the third vector is used for training the model.

Kazem et al. [5] In general it has been found that the PV efficiency increases with decrease of humidity. Relative humidity affects efficiency of photovoltaic as it affects the current, voltage and power. Result shows that when we have decreasing in relative humidity the voltage, current and efficiency increased. Also, it is found that Monocrystalline panel has the highest efficiency when relative humidity is decreased with respect to other technologies.

Gökmen et al. [6] This study reveals that how significant wind speed can affect a design parameter and operating performance of a PV system. From the tilt angle analysis, it can be concluded that especially summer period angles are affected much more compared to the winter period angles. It is observed that when wind speed is not taken into account, yearly energy is being underestimated. This difference is emerging from overestimating PV module temperature by not considering cooling effect of wind speed as in traditional method. Here, it is important to emphasize that wind would exist whether it is to be considered or not. The useful conclusion would be that wind speed should be considered when the energy production is assessed especially in the planning stage. Especially, at high altitudes such as hills and mountains, or very windy locations, the contribution from wind speed should not be ignored.

Khan et al. [7]After the experiments were conducted, it was observed that the power accession of 7-12% is observed when the Solar Panel is installed at a particular altitude ahead of the ground which indeed can be identified as the most probable and easy solution in order to utilize the less resources in getting maximum output.

Chapter 3

FACTORS AFFECTING PERFORMANCE OF A PV CELL

In this chapter the relevance of the main factors which affects in the performance of the Photo Voltaic cell are included in detail. Collection of solar data and its analysis will lead to the forecasting of the same.

3.1 TEMPERATURE

Solar cells perform better in cold rather than in hot climate. While an increase in heat doesn't affect the amount of solar energy which a solar panel receives, it does affect how much energy they output. High temperatures cause a drop-in voltage and, in turn, a drop-in power. Power is the product of voltage and current **(P = V x I)**, so although current increases slightly with temperature, the overall power is reduced by a significant drop in the potential difference between the electrons (voltage).

Sunlight is made up of charged particles called photons that originate from the collision of hydrogen atoms in the Sun's core. As these photons impact the semiconductors of the panels (the solar cells), they impart their energies into the electrons in the cell. Once these electrons are excited to a higher energy state, they move around and are ultimately collected and channeled into a stream of electrons. When the temperature is increased, the electron's rest temperature is effectively increased. Since it takes less energy to excite the electron, less energy is initially transferred from the photon which results in less production of power. The efficiency is based on the maximum voltage and current and the input power generated by the sun.

Efficiency = (Max current *Max voltage) / input power from the sun

For each degree rise in temperature above 25˚C (rated panel temperature) the panel output decays by about 0.25% for amorphous cells and about 0.4-0.5% for crystalline cells. Thus, the panels will produce 25% less power compared to what they are rated for at 25˚C. Thus, a 100W panel will produce only 75W in May/June in most parts of India where temperatures reach 45˚C and beyond in summer and electricity demand is high. Solar panels are tested under the NOCT (Nominal Operating Cell Temperature) which is the temperature reached by open circuit cells in a module under the following conditions

1) Irradiance (light) falling on the solar panel at 800W/m^2
2) Air temperature of 20ºC
3) Wind speed at 1m/s
4) Panel is mounted with an open back (air can circulate behind panel).

Most good quality panels available today in India have NOCT values of 47±2˚C. Lower the NOCT the better it is expected to perform in hotter climates.

3.2 HUMIDITY AND DEW-POINT

Humidity can bring down the efficiency of solar panel in two ways:

1. Tiny water droplets, or water vapor, can collect on solar panels and reflect or refract sunlight away from solar cells. This reduces the amount of sunlight hitting them and producing electricity.
2. Consistent hot and humid weather can degrade the solar panels themselves over their lifetime. This is true for both crystalline silicon cells and thin film modules, but cadmium telluride (thin film) solar cells perform about 5 percent better in tropical climates.

These effects plunge the reception level of the direct component of solar radiation. Humidity alters the irradiance non-linearly and irradiance itself causes little variations in voltage in a non-linear manner and large variations in the current linearly.
Greater scattered angles occur with smaller water vapor particles. More diffraction is also the result of more water vapor particles in the atmosphere. It is crystal clear that with much higher relative humidity of tropical countries like Malaysia we will have a disappointingly sharper drop in irradiance level. Wind speed has a reverse effect on relative humidity which in turn affects the received irradiance.

3.3 ALBEDO

Albedo is the measure of the diffuse reflection of solar radiation out of the total solar radiation received by an astronomical body. The proportion reflected is not only determined by properties of the surface itself, but also by the spectral and angular distribution of solar radiation reaching the Earth's surface. The average albedo of the Earth from the upper atmosphere, its planetary albedo, is 30–35% because of cloud cover, but widely varies locally across the surface because of different geological and environmental features. Although the spectral effects of direct and diffuse radiation on solar photovoltaic (PV) performance are relatively well understood, recent investigations have shown that there can be a spectral bias introduced due to albedo from common ground surfaces that can impact the optimal selection of PV materials for a known location. Silicon solar cells convert about 1/6 of incident sunlight into electricity and dissipate most of the remaining 5/6 as heat. So, in terms of their direct climate effect, they have an albedo, or reflectivity of 1/6. This is comparable to the albedo of standard asphalt shingles, so for most people, installing solar panels doesn’t have a net heating effect.

3.4 SHADING ON SOLAR PANELS

Ideally solar panels should be located such that there will never be shadows on them because a shadow on even a small part of the panel can have a surprisingly large effect on the output. The cells within a panel are normally all wired in series and the shaded cells affect the current flow of the whole panel. But there can be situations where it cannot be avoided, and thus the effects of partial shading should be considered while planning. If the affected panel is wired in

series (in a string) with other panels, then the output of all those panels will be affected by the partial shading of one panel. In such a situation, an obvious solution is to avoid wiring panels in series if possible.

A DC optimizer adjusts its output voltage and current to maintain maximum power without compromising the performance of other modules. For instance, when a shaded module produces electricity with a lower current, the DC optimizer will boost the current at its output to match the current flowing through the unshaded modules; to compensate, the optimizer reduces its output voltage by the same amount it boosts the current. This allows the shaded module to produce the same amount of electrical power without impeding the output of other modules. A system utilizing DC optimizers still needs an inverter to convert electricity from DC to AC.

3.5 CLOUD COVER

Light equals power, so the more direct light the panels receive, the more power will be the power produced. Which means less direct light will produce less amounts of power. Bright, sunny days will contribute to your system working at peak capacity. But on a day with thick cloud cover, power production will be much lower than average. Besides direct light, solar panels will also absorb diffused light and albedo, or reflective, light. As an example of diffused light: a cloud may briefly cover the sun, but we still receive light from the whole sky. That light, though not directed in a beam towards the panels, will still be absorbed. Power production will be reduced by about half, but will not come to a complete standstill. Panels can also absorb reflective light from shiny or light-coloured surfaces, such as we see with bodies of water or snow. An unusual phenomenon also exists when there are patches of cumulus clouds drifting through the sun's beam which is called the edge of cloud effect, as the sun peeks out of the spaces in between the clouds, the direct light combined with the reflective light will briefly boost your panels' power production. The increase is relatively small and short-lived but interesting to note. Most inverters allow for this brief surge of power so there is little danger to the solar panel system.

3.6 ALTITUDE

Earth's top of the atmosphere sunlight's intensity is about 30% more intense than the actual received on the land. In the Solar panels what we use today, actually we make use of the 70% energy coming from the Sun and utilize the working of our panels to fulfill our energy needs. As the Solar Panel is placed at a probable altitude of 27.432 meters/90 foot from the ground level, it is observed that the gases and the humidity factor along with factors affecting from the presence of population, which consists of emission of different gases from the masses, the usage of fossil fuels and much more are actually playing their role in stopping or limiting certain amount of proper intensities to reach to the Solar Panel and hence making the Solar panel less effective.

Solar energy can be most efficiently harnessed in areas near the equator or those which otherwise receive large amounts of insolation. Weather consideration shows that an altitude of

12.4 miles (20 kilometers) would be optimal for solar power collection within the earth's biosphere. The effects on solar panel output at lower altitudes, while not nearly as noticeable, are not well-known and could prove to be useful and a further power increase of 7-12% is possible to be observed due to placement of solar panels at a particular height of 90 feet (27.4 m) above ground level.

3.7 WINDSPEED

When a solar panel is too hot, it reduces efficiency due to the science behind a solar panel generating electricity. On the other hand, cooler solar panel temperatures improve efficiency. Cooler panels allow more energy to get through as an electric current than hot panels do. Wind reduces the temperature of the panel. Solar panels cooled by 1 degree Celsius are 0.05 percent more efficient. This percentage adds up over time.

Solar panels are always exposed to the elements, so it is natural to be concerned about damage from extreme weather. Heavy wind is a particularly common concern, whether it comes from hurricanes, tornadoes, or simple gusts. Solar panels are built to last for decades, and the builders that install them have got very good at making sure they can hold up to some serious abuse from nature. Most solar panel systems will hold up to heavy winds without any problems. A normal system can tolerate winds of at least 140 miles per hour and keeping working. That is near the middle of the range for category 4 hurricanes, which are far worse than anything that most homeowners will have to deal with unless they live in a particularly stormy area. The owner may need to plug the inverter back in or clean some debris off of the panels, but they will rarely need to do more to get their system working again after a serious storm. Many areas have local laws that set requirements for the durability of solar panels. The places that are prone to high winds tend to set minimum wind speeds that all panels must be able to survive. It can be a good idea to look those laws up to compare the minimum requirements with the average wind speeds in the area before investing in a system.

Chapter 4

METHODOLOGY

The various algorithms used for solar data analysis for the purpose of forecasting and prediction are included in this chapter. The block diagram of solar PV forecasting system used for real time data collection and the analysis is given in the Figure.1.

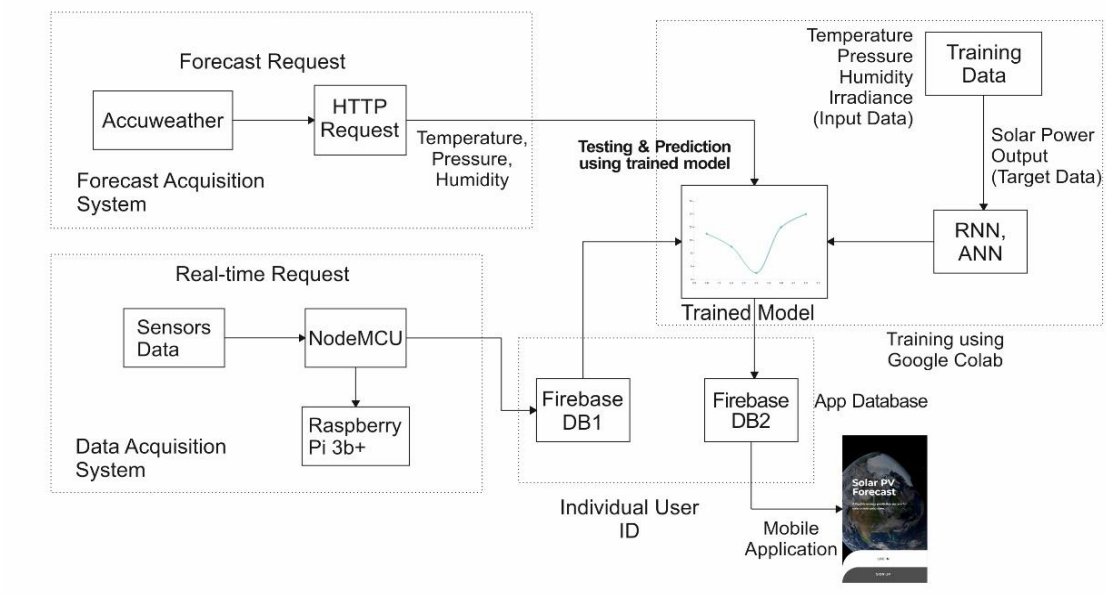

Fig.1 Block diagram of solar PV forecasting system

The data is obtained from the Global Energy Forecasting Competition 2014 (GEFCOM2014) is given in Figure.2, which also includes forecasting in the domains of electric load, wind power, solar power, and electricity prices. Hence our objective is to determine the solar forecast on an hourly basis for a future period of 5 days. The machine is trained to create a model on to determine our target variable “Solar Power”. The train data collected is been converted into .csv file. The files are then pre-processed into solar missing data software to fill the blank spaces of the attributes which are missing.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	ZONE ID	TIMESTAM	VAR78	VAR79	VAR134	VAR157	VAR164	VAR165	VAR166	VAR167	VAR169	VAR175	VAR178	VAR228	POWER
2	1	20120401	0.001967	0.003609	94843.63	60.22191	0.244601	1.039334	-2.50304	294.4485	2577830	1202532	2861797	0	0.754103
3	1	20120401	0.005524	0.033575	94757.94	54.6786	0.457138	2.482865	-2.99333	295.6514	5356093	2446757	5949378	0	0.555
4	1	20120401	0.030113	0.132009	94732.81	61.29489	0.771429	3.339867	-1.98254	294.4546	7921788	3681336	8939176	0.001341	0.438397
5	1	20120401	0.057167	0.110645	94704.06	67.77528	0.965866	3.106102	-1.44605	293.2615	9860520	4921504	11331679	0.002501	0.145449
6	1	20120401	0.051027	0.18956	94675	70.17299	0.944669	2.601146	-1.90449	292.7329	11143097	6254380	13105558	0.003331	0.111987
7	1	20120401	0.036996	0.099045	94676.94	72.37404	0.641353	1.333368	-1.72843	292.0771	11815767	7558415	14198503	0.00396	0.057244
8	1	20120401	0.080911	0.121323	94708.06	81.79874	0.753142	1.457923	-1.03462	291.0693	12274591	8798617	14925342	0.00497	0.088718
9	1	20120401	0.036159	0.139069	94748.81	87.85406	0.788338	2.374826	-1.08904	289.0735	12351290	10041167	15112951	0.006477	0.030064
10	1	20120401	0.036372	0.072609	94785.81	88.79349	0.502275	1.985531	-0.96301	288.0313	12351290	11257316	15112951	0.006725	0.000128
11	1	20120401	0.014353	0.035797	94817.75	90.45067	0.501918	1.999518	-0.93032	287.4058	12351290	12460132	15112951	0.006745	0
12	1	20120401	0.005387	0.004501	94847.38	90.67618	0.156774	1.809739	-0.09818	286.3208	12351290	13599879	15112567	0.00675	0
13	1	20120401	0.00087	0	94852.06	92.92891	0.014343	1.354908	-0.92226	285.0354	12351290	14724565	15112567	0.00675	0
14	1	20120401	0	0	94810.63	94.98952	0.004364	1.256864	-1.15641	284.7332	12351290	15820126	15112567	0.00675	0
15	1	20120401	0	0	94758.13	94.81033	0	0.921034	-0.80404	284.4626	12351290	16911888	15112567	0.00675	0
16	1	20120401	0.00148	0	94730.38	95.55997	0.039764	0.62243	0.318149	284.1331	12351290	17989312	15112567	0.00675	0
17	1	20120401	0.005981	0	94721.56	99.39195	0.386597	-0.4045	1.343286	282.8701	12351290	19080544	15112567	0.00675	0
18	1	20120401	0.020294	0	94707.75	99.67973	0.937317	-0.49365	1.279655	282.8855	12351290	20227568	15112567	0.00675	0
19	1	20120401	0.043617	0	94699.69	100.2071	0.970123	-0.29845	1.204877	283.1572	12351290	21420224	15112567	0.006755	0
20	1	20120401	0.047783	0	94733.31	99.53146	0.893005	-0.12777	1.207767	283.3142	12351290	22618864	15112567	0.006763	0
21	1	20120401	0.043663	0	94749.81	99.42308	0.891571	-0.23109	1.276311	283.1843	12351290	23790336	15112567	0.006772	0
22	1	20120401	0.044991	0	94769	98.57301	0.623688	-0.05542	1.248788	283.6531	12417521	24904848	15240341	0.006779	0.006346
23	1	20120401	0.03886	0	94787.94	92.12311	0.832794	-0.85174	0.972498	286.2585	12947510	26133136	15980090	0.006783	0.071538
24	1	20120401	0.035019	0	94775.25	82.74525	0.953293	-1.10296	0.310311	288.4182	13965544	27398656	17317616	0.006791	0.229167
25	1	20120402	0.07444	0	94736.13	71.7513	0.932465	-0.19305	-0.44757	290.2673	15464841	28768880	19196208	0.006801	0.346474

Fig. 2 Raw GEFCOM 2014 data

From the training data (original data with 12 attributes) 4 attributes namely temperature, humidity, pressure and irradiance have been taken for input to the neural network (feed-forward backpropagation network).

The neural network is then tested for Levenberg-Marquardt backpropagation

4.1 ACCUMULATION OF HISTORICAL WEATHER DATA AND SOLAR PANEL DATA OF GEFCOM 2014

The parameters in the data are as follows:

1. Date and time
2. Temperature
3. Relative humidity
4. Pressure
5. Irradiance (W/m^2)
6. DC Output (W)

4.2 CLEANING OF ACCUMULATED DATA

As the data obtained has missing values and with the current sensors data can be used with only a few parameters namely date and time temperature, pressure, humidity, irradiance, voltage, current and power. The rest of the data can be neglected focusing mainly on the above mentioned parameters.

4.3 DETECTION OF ANOMALY OUTLIERS

Anomaly detection is the process of identifying unexpected items or events in datasets, which differ from the norm. Anomaly detection is often applied on unlabeled data, taking only the internal structure of the dataset into account.

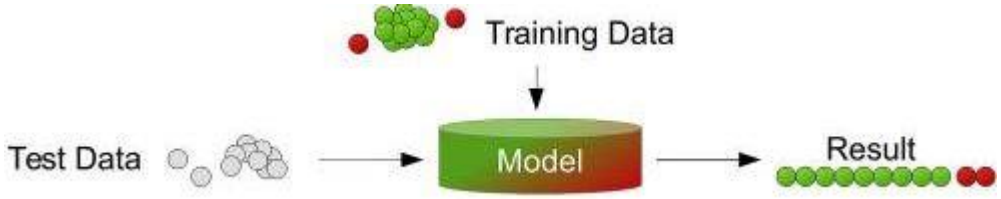

Fig.3 Working of Anomaly detection

This describes the setup where the data comprises of fully labeled training and test data sets. An ordinary classifier can be trained first and applied afterwards. This scenario is very similar to traditional pattern recognition with the exception that classes are typically strongly unbalanced. Few algorithms than can detect the anomaly are Isolation forest, CLBOF and KNN.

4.3.1 ISOLATION FOREST

Isolation Forest is built on the basis of decision trees. In these trees, partitions are created by first randomly selecting a feature and then selecting a random split value between the minimum and maximum value of the selected feature. Outliers are less frequent than regular observations and are different from them in terms of values (they lie further away from the regular observations in the feature space). That is why by using such random partitioning they should be identified closer to the root of the tree (shorter average path length, i.e., the number of edges an observation must pass in the tree going from the root to the terminal node), with fewer splits necessary.

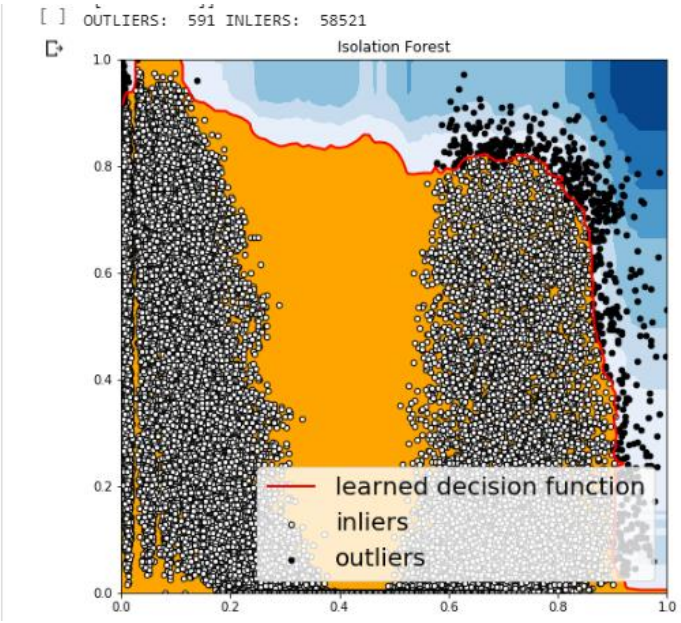

Fig.4 Outliners in Isolation Forest

Isolation forest is defined as $S(n,x) = 2^{-E(h(x))/c(n)}$ where *h(x)* is the path length of observation *x*, *c(n)* is the average path length of unsuccessful search in a Binary Search Tree and *n* is the number of external nodes. A score close to 1 indicates anomalies. Score much smaller than 0.5 indicates normal observations. If all scores are close to 0.5 then the entire sample does not seem to have clearly distinct anomalies.

4.3.2 K-NEAREST NEIGHBOR

KNN classifies a data point based on how its neighbors are classified. Besides classification into groups, *k*-NN can also be used to estimate continuous values. To approximate a data point's value, *k* -NN takes the aggregated value of its most similar neighbors. The *k* in *k* -NN is a parameter that refers to the number of nearest neighbors to include in the majority voting process. Choosing the right value of *k* is a process called parameter tuning, and is critical to prediction accuracy. If *k* is too small, data points would match immediate neighbours only, amplifying errors due to random noise. If *k* is too large, data points would try to match far flung neighbors, diluting underlying patterns. But when *k* is just right, data points would reference a suitable number of neighbors such that errors cancel out to reveal subtle trends in the data. To achieve the best fit and lowest error, the parameter *k* can be tuned by cross-validation. In our binary (i.e. two class) classification problem, we can also avoid tied votes by choosing *k* to be an odd number.

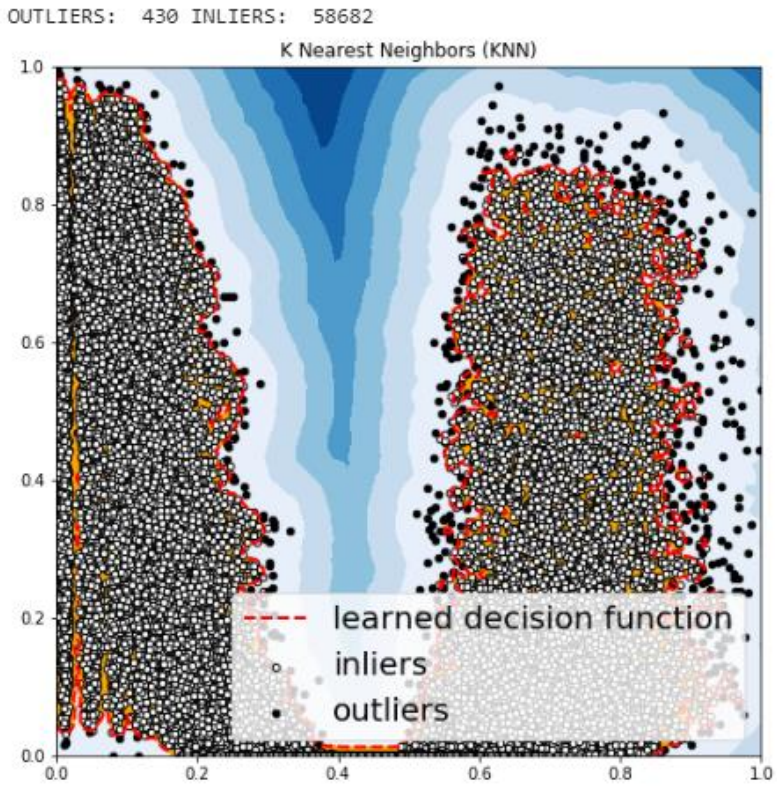

Fig.5 Outliners in KNN

The simplest approach to detecting anomalies is by visualizing the data in a plot and separating the outliners as shown in figure 4.

4.3.3 CLUSTER BASED LOCAL OUTLINER FACTOR (CBLOF)

CBLOF uses clustering in order to determine dense areas in the data and performs density estimation for each cluster afterwards. In theory, every clustering algorithm can be used to cluster the data in a first step. After clustering, CBLOF uses a heuristic to classify the resulting clusters into large and small clusters. Finally, an anomaly score is computed by the distance of each instance to its cluster centre multiplied by the instances belonging to its cluster. For small clusters, the distance to the closest large cluster is used. The procedure of using the amount of cluster members as a scaling factor should estimate the local density of the clusters.

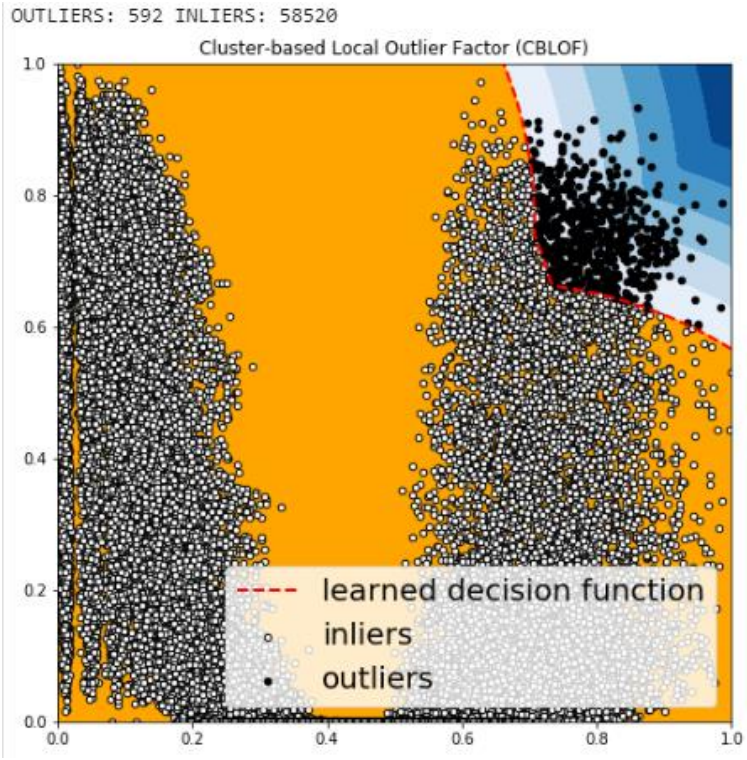

Fig.6 Outliners in CBLOF

The results of CBLOF using a simple two-dimensional dataset is visualized as shown in figure 4.Clustering-based anomaly detection algorithms are very sensitive to the parameter *k*, since adding just a single additional centroid might lead to a very different outcome. Cluster Based Local Outliner Factor is one of the most commonly used algorithm for the detection of Anomaly.

4.4 MATLAB

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-**use** environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include: Math and computation. The main purpose of using matlab is to identify the machine learning method with best accuracy. It requires the user to input the training data to learn from and the target data to achieve results. Once the simulation has been done the next step is to operate it on a bigger and more robust platform for flexibility and ease in updating the process.

4.5 GOOGLE COLAB

Google colab is a free cloud service that allows the users to take the load off the local machines and run their code on a better GPU. The main focus here is to allow the code to run free without any hindrance from lack of memory. Since the python code used involves various machine learning methods for comparison and the colab notebook is easy to operate the results can be achieved with less complications and the availability of libraries makes it easier to visualize the results and convey the observations to the audience.

4.6 FEED FORWARD BACKPROPAGATION NETWORK

Learning factors of Backpropagation Network

a. INITIAL WEIGHTS

The final result may be affected by the initial weights of the multilayer feed-forward network. They are initialized with random values .The weights are set to small values in the range $[\frac{-3}{\sqrt{oi}}, \frac{3}{\sqrt{oi}}]$ where oi is the number of processing elements *j* that feed-forward to process element *i*.

b. LEARNING RATE AND MOMENTUM FACTOR

For a slower learning rate and better output the α is set to 0.01 and momentum factor to 0.9 by default using *learngdm* in neural network programming using *nntool* in MATLAB.

4.7 LEVENBERG – MARQUARDTS ALGORITHM

Levenberg -Marquardt is a popular alternative to the Gauss-Newton method of finding the minimum of a function $F(x)$ that is a sum of squares of nonlinear functions,

$$F(x) = \frac{1}{2}\sum_{i=1}^{m}[f_i(x)]^2.$$

Let the Jacobian of $f_i(x)$ be denoted $J_i(x)$, then the Levenberg-Marquardt method searches in the direction given by the solution $\mathbf{p}$ to the equations

$$\left(J_k{}^{\mathrm{T}} J_k + \lambda_k I\right) p_k = -J_k{}^{\mathrm{T}} f_k,$$

Where λ_k are nonnegative scalars and I is the identity matrix. The method has the nice property that, for some scalar Δ related to λ_k, the vector p_k is the solution of the constrained sub problem of minimizing $\|J_k\, p + f_k\|_2^2 / 2$ subject to $\|p\|_2 \le \Delta$ (Gill *et al.* 1981, p. 136).

In function optimization problems with neural networks, the training algorithm is designed to determine the best network parameters in order to minimize network error. Various function optimization methods can be applied to ANN model training. One of these methods is the Levenberg-Marquardt algorithm. The Levenberg-Marquardt algorithm provides a numerical solution to the problem of minimizing a nonlinear function, over a space of parameters for the function. It is a popular alternative to the Gauss–Newton method of finding the minimum of a function.

4.8 RNN (RECURRENT NEURAL NETWORK)

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behaviour. Unlike feed forward neural networks, RNNs can use their

internal state (memory) to process sequences of inputs. The term "recurrent neural network" is used to refer to two broad classes of networks with a similar general structure, where one is finite impulse and the other is infinite impulse. Both classes of networks exhibit temporal dynamic behaviour. A finite impulse recurrent network is a directed acyclic graph that can be unrolled and replaced with a strictly feed forward neural network, while an infinite impulse recurrent network is a directed cyclic graph that cannot be unrolled. Both finite impulse and infinite impulse recurrent networks can have additional stored state, and the storage can be under direct control by the neural network. The storage can also be replaced by another network or graph, if that incorporates time delays or has feedback loops. Such controlled states are referred to as gated state or gated memory, and are part of long short-term memory networks (LSTMs).

4.8.1 LSTM (LONG SHORT-TERM MEMORY)

Long short-term memory (LSTM) is a deep learning system that avoids the vanishing gradient problem. LSTM is normally augmented by recurrent gates called "forget" gates. LSTM prevents back propagated errors from vanishing or exploding. Instead, errors can flow backwards through unlimited numbers of virtual layers unfolded in space. That is, LSTM can learn tasks that require memories of events that happened thousands or even millions of discrete time steps earlier. Problem-specific LSTM-like topologies can be evolved. LSTM works even given long delays between significant events and can handle signals that mix low and high frequency components.

Many applications use stacks of LSTM RNNs and train them by Connectionist Temporal Classification (CTC) to find an RNN weight matrix that maximizes the probability of the label sequences in a training set, given the corresponding input sequences. CTC achieves both alignment and recognition. LSTM can learn to recognize context-sensitive languages unlike previous models based on hidden Markov models (HMM) and similar concepts. LSTM RNN utilises Bidirectional RNN which uses a finite sequence to predict or label each element of the sequence based on the element's past and future contexts. This is done by concatenating the outputs of two RNNs, one processing the sequence from left to right, the other one from right to left. The combined outputs are the predictions of the teacher-given target signals.

4.9 DIFFERENCE BETWEEN RNN AND LSTM

The structure of RNN is very similar to Long Term Short Term. However, the main difference is with how parameters are calculated and constructed. One of the advantages with LSTM is **insensitivity to gap length**. RNN and HMM rely on the hidden state before emission / sequence. If we want to predict the sequence after 1,000 intervals instead of 10, the model would forget the starting point where Long Short Term Memory remembers the starting point.

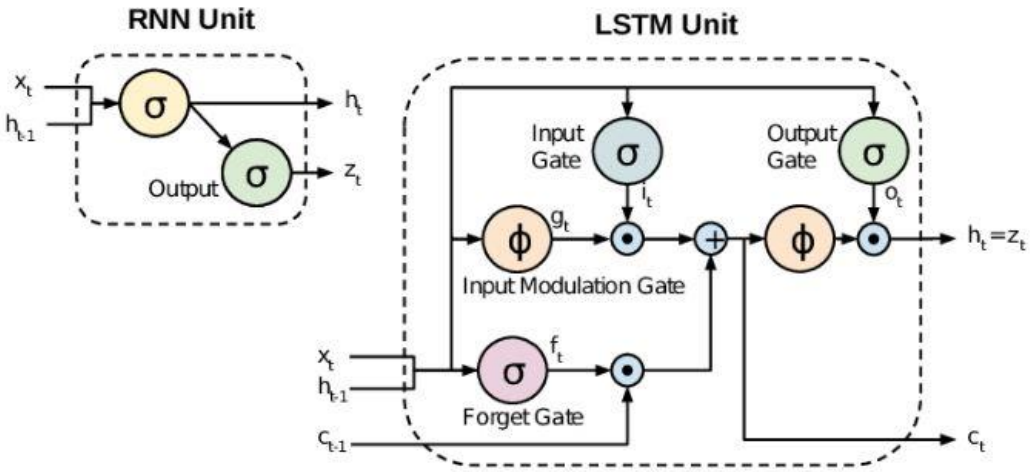

Fig. 7 RNN and LSTM

4.10 LSTM WORKING

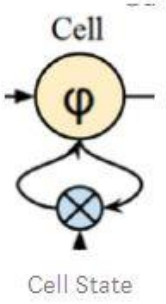

Fig. 8 Cell State

The equation of a cell state is given as:

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tilde{\mathbf{c}}_t$$

The **long-term memory** is usually called the **cell state**. The looping arrows indicate recursive nature of the cell. This allows information from previous intervals to be stored within the LSTM cell. Cell state is modified by the forget gate placed below the cell state and also adjust by the input modulation gate. From equation, the previous cell state forgets by multiply with the forget gate and adds new information through the output of the input gates. The **remember vector** is usually called the **forget gate**. The output of the forget gate tells the cell state which information to forget by multiplying 0 to a position in the matrix. If the output of the forget gate is 1, the information is kept in the cell state.

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_t\right)$$

From equation, sigmoid function is applied to the weighted input/observation and previous hidden state.

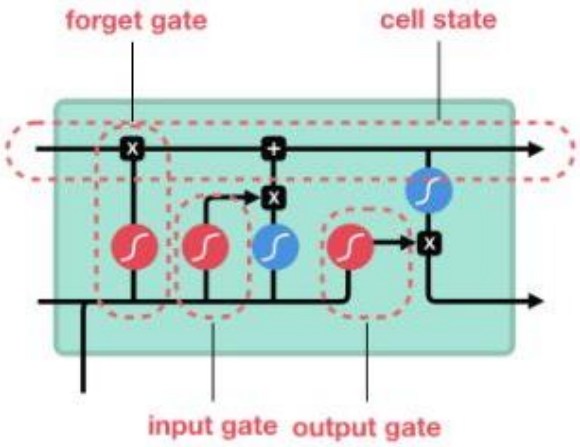

Fig. 9 LTSM

The equation of input gate is given as:

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i\right)$$

The **save vector** is usually called the **input gate**. These gates determine which information should enter the cell state / long-term memory. The important parts are the activation functions for each gate. The input gate is a **sigmoid** function and has a range of [0, 1]. Because the equation of the cell state is a summation between the previous cell states, sigmoid function alone will only add memory and not be able to remove/forget memory.

4.10.1 LSTM CELL

The equation for the output gate is given as:

$$\mathbf{o}_t = \sigma\left(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o\right)$$

The **focus vector** is usually called the **output gate** and the working memory is called as the hidden state. The first sigmoid activation function is the **forget gate**. Which information should be forgotten from the previous cell state (Ct-1). The second sigmoid and first tanh activation function is our **input gate**. Which information should be saved to the cell state or should be forgotten? The last sigmoid is the **output gate** and highlights which information should be going to the next **hidden state.**

4.11 TRAINING AND TESTING IN GOOGLE COLAB

The cleaned data is then imported in the Google colab and using 25% of the data the machine is trained for making predictions. Then once this is achieved the rest 75% is taken as target and compared to the predictions and accordingly MSE and accuracy is found.

Once the machine has been trained the goal resides in improving the accuracy using various available machine leaning methods and anomaly detection.

Google colab is also responsible for the coordination between Firebase, android application and Blynk.

4.12 MQTT

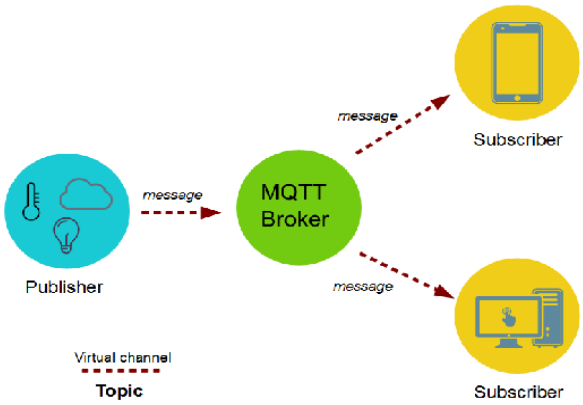

Fig. 10 MQTT

MQTT is one of the most commonly used protocols in IoT projects. It stands for Message Queuing Telemetry Transport. MQTT is based around the idea that devices can **publish** or **subscribe** to **topics**. If Device #1 has recorded the temperature from one of its sensors, it can **publish** a message which contains the temperature value it recorded, to a topic (e.g. *"Temperature"*). This message is sent to an MQTT Broker, which you can think of as a switch/router on a local area network. Once the MQTT Broker has received the message, it will send it to any devices (in this case, Device #2) which are **subscribed** to the same topic. In this project, we will be publishing to a topic using an ESP8266, and creating a Python script that will subscribe to this same topic, via a Raspberry Pi which will act as the MQTT Broker.

The broker is primarily responsible for receiving all messages, filtering the messages, decide who **is** interested in them and then publishing the message to all subscribed clients.

The broker can store the data in the form of retained messages (need to subscribe with database client) so that new subscribers to the topic can get the last value straight away.

It also keeps track of all the session's information as the devices goes on and off called "persistent sessions".

The main advantages of MQTT broker are:

1. Eliminates vulnerable and insecure client connections
2. Can easily scale from a single device to thousands
3. Manages and tracks all client connection states, including security credentials and certificates
4. Reduced network strain without compromising the security (cellular or satellite network)

Chapter 5

ANALYSIS OF SOLAR DATA FOR PREDICTION

A detailed study of different sets of solar data is carried out in this chapter for understanding the method of usage of it in different algorithms for the purpose of prediction.

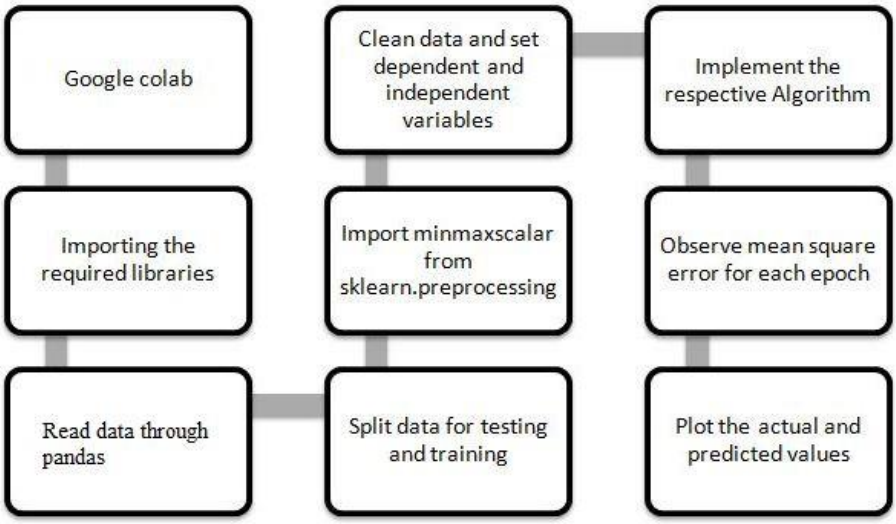

Fig.11 Flowchart for implementing any algorithm

Figure.11 shows the flowchart for implementing an algorithm. Accumulation of data is done from GEFCOM2014 .Here the climatic Zone ID 1 represents the countries that have a hot-humid climate and more than receives 20 inches of annual precipitation.

The target variable is solar power. There are 12 independent variables from the ECMWF NWP output to be used as below.

- 078.128 Total column liquid - Vertical integral of cloud liquid water content
- 079.128 Total column ice - Vertical integral of cloud ice water content
- 134.128 surface pressure
- 157.128 Relative humidity at 1000 mbar - Relative humidity is defined with respect to saturation of the mixed phase, i.e. with respect to saturation over ice below -23C and with respect to saturation over water above 0C. In the regime in between a quadratic interpolation is applied.
- 164.128 total cloud cover - Total cloud cover derived from model levels using the model's overlap assumption
- 165.128 U wind component
- 166.128 V wind component
- 167.128 temperature
- 169.128 surface solar rad down - Accumulated field
- 175.128 surface thermal rad down - Accumulated field
- 178.128 top net solar rad - Net solar radiation at the top of the atmosphere. Accumulated field
- 228.128 total precipitation - Convective precipitation + stratiform precipitation (CP +LSP). Accumulated field.

a. GEFCOM 2014 data

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	ZONE ID	TIMESTAM	VAR78	VAR79	VAR134	VAR157	VAR164	VAR165	VAR166	VAR167	VAR169	VAR175	VAR178	VAR228	POWER
2	1	20120401	0.001967	0.003609	94843.63	60.22191	0.244601	1.039334	-2.50304	294.4485	2577830	1202532	2861797	0	0.754103
3	1	20120401	0.005524	0.033575	94757.94	54.6786	0.457138	2.482865	-2.99333	295.6514	5356093	2446757	5949378	0	0.555
4	1	20120401	0.030113	0.132009	94732.81	61.29489	0.771429	3.339867	-1.98254	294.4546	7921788	3681336	8939176	0.001341	0.438397
5	1	20120401	0.057167	0.110645	94704.06	67.77528	0.965866	3.106102	-1.44605	293.2615	9860520	4921504	11331679	0.002501	0.145449
6	1	20120401	0.051027	0.18956	94675	70.17299	0.944669	2.601146	-1.90449	292.7329	11143097	6254380	13105558	0.003331	0.111987
7	1	20120401	0.036996	0.099045	94676.94	72.37404	0.641353	1.333368	-1.72843	292.0771	11815767	7558415	14198503	0.00396	0.057244
8	1	20120401	0.080911	0.121323	94708.06	81.79874	0.753142	1.457923	-1.03462	291.0693	12274591	8798617	14925342	0.00497	0.088718
9	1	20120401	0.036159	0.139069	94748.81	87.85406	0.788338	2.374826	-1.08904	289.0735	12351290	10041167	15112951	0.006477	0.030064
10	1	20120401	0.036372	0.072609	94785.81	88.79349	0.502275	1.985531	-0.96301	288.0313	12351290	11257316	15112951	0.006725	0.000128
11	1	20120401	0.014353	0.035797	94817.75	90.45067	0.501918	1.999518	-0.93032	287.4058	12351290	12460132	15112951	0.006745	0
12	1	20120401	0.005387	0.004501	94847.38	90.67618	0.156774	1.809739	-0.09818	286.3208	12351290	13599879	15112567	0.00675	0
13	1	20120401	0.00087	0	94852.06	92.92891	0.014343	1.354908	-0.92226	285.0354	12351290	14724565	15112567	0.00675	0
14	1	20120401	0	0	94810.63	94.98952	0.004364	1.256864	-1.15641	284.7332	12351290	15820126	15112567	0.00675	0
15	1	20120401	0	0	94758.13	94.81033	0	0.921034	-0.80404	284.4626	12351290	16911888	15112567	0.00675	0
16	1	20120401	0.00148	0	94730.38	95.55997	0.039764	0.62243	0.318149	284.1331	12351290	17989312	15112567	0.00675	0
17	1	20120401	0.005981	0	94721.56	99.39195	0.386597	-0.4045	1.343286	282.8701	12351290	19080544	15112567	0.00675	0
18	1	20120401	0.020294	0	94707.75	99.67973	0.937317	-0.49365	1.279655	282.8855	12351290	20227568	15112567	0.00675	0
19	1	20120401	0.043617	0	94699.69	100.2071	0.970123	-0.29845	1.204877	283.1572	12351290	21420224	15112567	0.006755	0
20	1	20120401	0.047783	0	94733.31	99.53146	0.893005	-0.12777	1.207767	283.3142	12351290	22618864	15112567	0.006763	0
21	1	20120401	0.043663	0	94749.81	99.42308	0.891571	-0.23109	1.276311	283.1843	12351290	23790336	15112567	0.006772	0
22	1	20120401	0.044991	0	94769	98.57301	0.623688	-0.05542	1.248788	283.6531	12417521	24904848	15240341	0.006779	0.006346
23	1	20120401	0.03886	0	94787.94	92.12311	0.832794	-0.85174	0.972498	286.2585	12947510	26133136	15980090	0.006783	0.071538
24	1	20120401	0.035019	0	94775.25	82.74525	0.953293	-1.10296	0.310311	288.4182	13965544	27398656	17317616	0.006791	0.229167
25	1	20120402	0.07444	0	94736.13	71.7513	0.932465	-0.19305	-0.44757	290.2673	15464841	28768880	19196208	0.006801	0.346474

Fig. 12 GEFCOM2014 data

The GEFCOM data is downloaded from the GEFCOM contest website for learning purpose and understanding how the data is processed in the neural network for prediction purposes. The data belongs to the region Denmark (Zone ID 1) and this data is to be processed for further machine learning purposes. The data is cleaned for erroneous values and then used for training the machine learning model.

b. Ettimadai data

temp	pressure	humidity	lux	power
30	1014.6	61.3	3698.666	13605
30.2	1015	60.7	12903.75	14404
30.8	1015	59	15046.67	15151
31.09091	1015	56.63636	16304.47	15892
31.36364	1015	57.45455	11277.35	9596
32.9	1015	53.7	19558.5	13121
33	1015	52	22715	18908
34	1015	50.45455	22488.94	19779
34.1	1015	49.8	23442.92	20539
36	1015	45.3	25064	21229
36.27273	1015	44.63636	26476.59	21999
35.90909	1015	44.27273	28230.83	22762
37	1015	41	29538.83	23379
37.36364	1015	40.72727	31366.92	22980
36.81818	1015	41	31605.45	22249
37.27273	1015	40.18182	33271.36	22778
37.8	1015	39	35646.16	23671
36.72727	1015	41.18182	35525.45	24160
37.09091	1015	40.27273	33072.35	25091
38.27273	1015	38.45455	33632.88	25842
38.5	1015	38.7	33951.16	25868

Fig. 13 Ettimadai data collected using Hardware setup

The collected data had to be cleaned since the tie stamp for each document (PV cell and atmospheric data) was different. The data was then merged together and the missing values were obtained from solar forecast websites.

5.2 CLEANING OF ACCUMULATED DATA:

Cleaned data of GEFCOM 2014 with selected 4 attributed as dependent variables.

Temperature (°C)	Surface Pressure (hPa)	Relative Humidity %	Irradiance (W/m^2)
21.4484863	948.43625	60.22190857	7717.397222
22.6513672	947.579375	54.67860413	7126.930556
21.4545898	947.328125	61.29489136	5385.366667
20.2614746	947.040625	67.77528381	3562.713889
19.7329102	946.75	70.17298889	1868.527778
19.0771484	946.769375	72.3740387	1274.511111
18.0693359	947.080625	81.79873657	213.0527778
16.0734863	947.488125	87.85406494	0
15.03125	947.858125	88.79348755	0
14.4057617	948.1775	90.45066833	0
13.3208008	948.47375	90.67617798	0
12.0354004	948.520625	92.9289093	0

Fig.14 Data with 4 major attributes
(Surface pressure, Relative Humidity, Temperature, Surface Irradiance)

After the data has been cleaned the required parameters are selected and copied in a separate csv file is it is imported in MATLAB for training the Neural Network.

5.3 MATLAB SIMULATION

To have a better understanding of the prediction techniques machine learning models are used. And accordingly the method with least MSE in implemented in Google colab.

The training data (input) and target data are imported in the workspace. The imported data is fed to NN Tool and used for creating a Feed Forward Network using nntool command in MATLAB. Once the Network has been trained the output is saved in the Network1_output and the MSE is saved in the network1_errors which is later used for comparing the data with the target data.

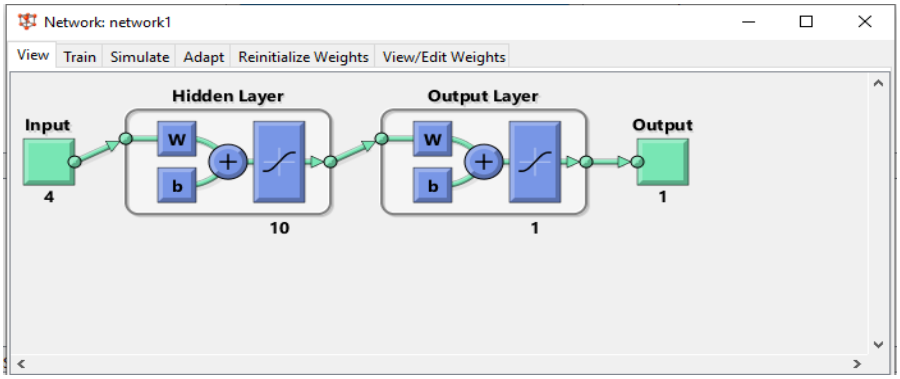

Fig. 15 Sample NN layer diagram

The structure of the neural network is shown above the input data is fed to the hidden layer and weights are calculated for each iteration and adjusted. The same process is repeated in the output layer as well to find its weights.

1. Training the neural network with Levenberg-Marquardt and Bayesian Regularization and finding the MSE.
2. Creating Network with Feed-forward backpropagation.

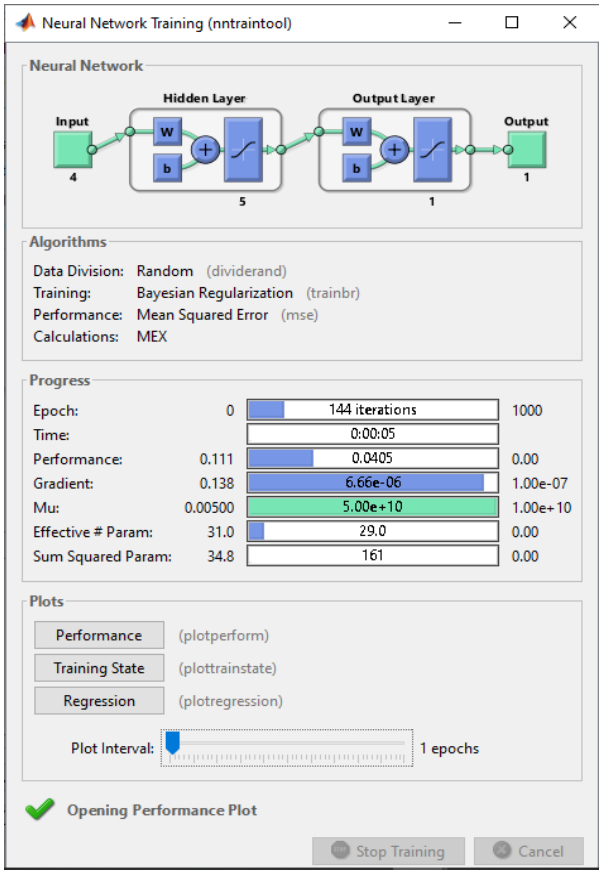

Fig. 16 NN Tool Operation

The figure above shows the MATLAB NN Tool in progress for 1000 epochs (iterations)

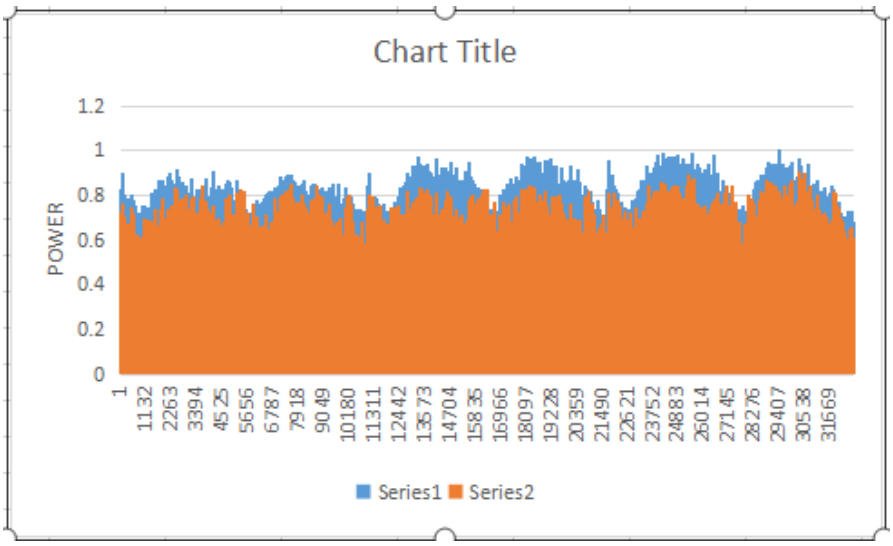

Fig. 17 Output power Plot

The Predicted Output from the MATLAB and the actual output are compared and plotted in MATLAB.

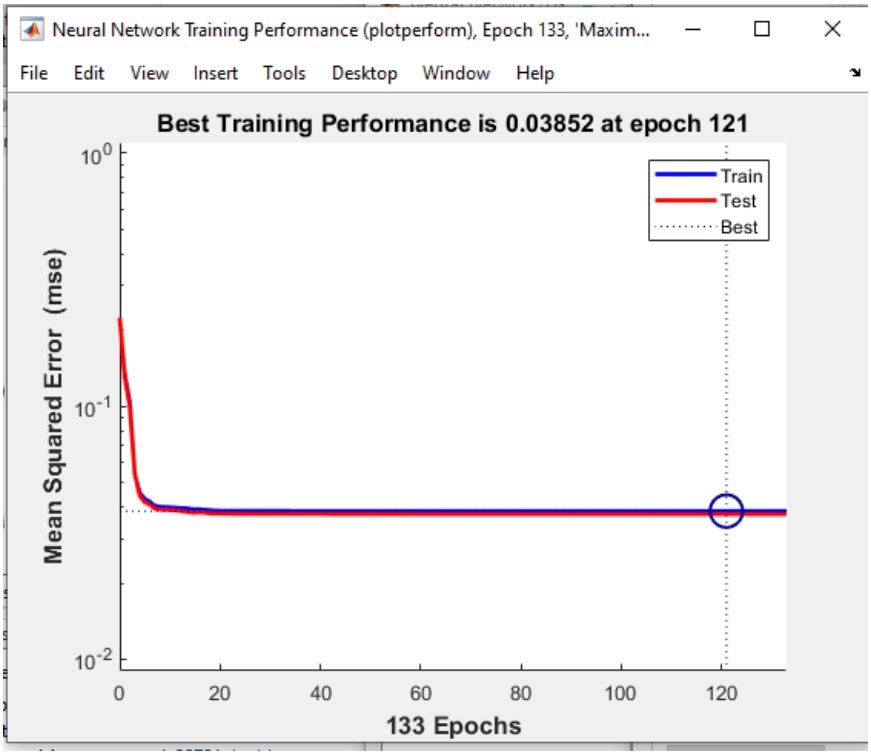

Fig. 18 MSE Plot

The MSE of each iteration is plotted and it is found that the minimum MSE is 0.03852 and the prediction is best at epoch 121.

5.4 PREDICTIONS USING NEURAL NETWORKS
5.4.1A ARTIFICIAL NEURAL NETWORK

ANNs are considered as nonlinear statistical data modelling tools where the complex relationships between inputs and outputs are modeled and patterns are found. ANN operates on Hidden State. These hidden states are similar to neurons. Each of this hidden state is a transient form which has a probabilistic behavior.

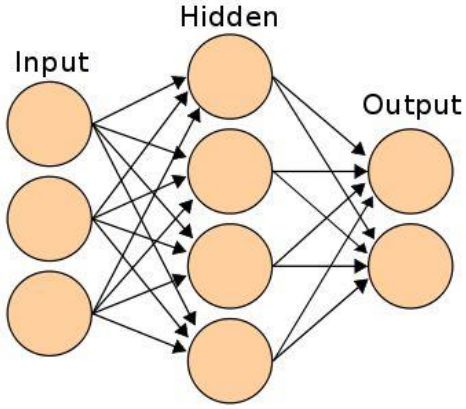

Fig.19 Three stages of ANN

Every linkage calculation in an Artificial Neural Network is similar.
Logit (H1) = W(I1H1) * I1 + W(I2H1) * I2 + W(I3H1) * I3 + Constant
Where I1, I2 and I3 are the inputs and H1, H2.H3 and H4 are the respective hidden states. Outputs are O1 and O2.
The activation of H1 is given as:
P (H1) = 1/(1+e^(-f))
Re-calibration of weights on the linkage between hidden node and output node is a function of this error rate on output nodes. Using these errors we can re-calibrate the weights of linkage between hidden nodes and the input nodes in a similar way. ANN does multiple re-calibrations for each linkage weights. Hence, the time taken by the algorithm rises much faster than other traditional algorithm for the same increase in data volume.

5.4.1B MEAN SQUARE ERROR

The loss parameter here is the Mean Squared Error, which is the metric used here.
The Accuracy is lower and MSE is higher for ANN since the previous state weights and initial weights are not taken into account unlike Recurrent Neural Networks. Thus to improve the model accuracy RNN is used.

```
Epoch 1/5
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:190:

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:197:

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:207:

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:216:

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:223:

44334/44334 [==============================] - 2s 48us/step - loss: 0.1183 - acc: 0.4379
Epoch 2/5
44334/44334 [==============================] - 1s 33us/step - loss: 0.0769 - acc: 0.4445
Epoch 3/5
44334/44334 [==============================] - 2s 35us/step - loss: 0.0745 - acc: 0.4445
Epoch 4/5
44334/44334 [==============================] - 2s 34us/step - loss: 0.0744 - acc: 0.4445
Epoch 5/5
44334/44334 [==============================] - 2s 35us/step - loss: 0.0744 - acc: 0.4445
```

Fig.20 Mean square error of ANN

5.4.2A RNN (RECURRENT NEURAL NETWORK)

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behaviour. Unlike feed forward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. The term "recurrent neural network" is used to refer to two broad classes of networks with a similar general structure, where one is finite impulse and the other is infinite impulse. Both classes of networks exhibit temporal behaviour. A finite impulse recurrent network is a directed acyclic graph that can be unrolled and replaced with a strictly feed forward neural network, while an infinite impulse recurrent network is a directed cyclic graph that cannot be unrolled. Both finite impulse and infinite impulse recurrent networks can have additional stored state, and the storage can be under direct control by the neural network. The storage can also be replaced by another network or graph, if that incorporates time delays or has feedback loops. Such controlled states are referred to as gated state or gated memory, and are part of long short-term memory networks (LSTMs).

5.4.2B LSTM (LONG SHORT-TERM MEMORY)

Long short-term memory (LSTM) is a deep learning system that avoids the vanishing gradient problem. LSTM is normally augmented by recurrent gates called "forget" gates. LSTM prevents back propagated errors from vanishing or exploding. Instead, errors can flow backwards through unlimited numbers of virtual layers unfolded in space. That is, LSTM can learn tasks that require memories of events that happened thousands or even millions of discrete time steps earlier. Problem-specific LSTM-like topologies can be evolved. LSTM works even given long delays between significant events and can handle signals that mix low and high frequency components.

Many applications use stacks of LSTM RNNs and train them by Connectionist Temporal Classification (CTC) to find an RNN weight matrix that maximizes the probability of the label sequences in a training set, given the corresponding input sequences. CTC achieves both alignment and recognition. LSTM can learn to recognize context-sensitive languages unlike previous models based on hidden Markov models (HMM) and similar concepts. LSTM RNN utilises Bidirectional RNN which uses a finite sequence to predict or label each element of the sequence based on the element's past and future contexts. This is done by concatenating the outputs of two RNNs, one processing the sequence from left to right, the other one from right to left. The combined outputs are the predictions of the teacher-given target signals.

5.4.2C MEAN SQUARE ERROR

The Training time taken is nearly 48 minutes, which causes delay while forecasting in the app. The MSE is improved significantly when compared to ANN since the initial weights are taken into account.

```
Epoch 1/1
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:190: T

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:197: T

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:207: T

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:216: T

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:223: T

30000/30000 [==============================] - 2923s 97ms/step - loss: 0.0257
```

Fig.21 Mean Square error of LSTM

5.5 MACHINE LEARNING

5.5.1A GRADIENT BOOSTING (GBM)

Gradient boosting works on the basis of involving these

- A loss function to be optimized.
- A weak learner to make predictions.
- An additive model to add weak learners to minimize the loss function.

The loss function used depends on the type of problem being solved. It must be differentiable, but many standard loss functions are supported. Regression may use a squared error and classification may use logarithmic loss. A benefit of the gradient boosting framework is that a new boosting algorithm does not have to be derived for each loss function that may want to be used, instead, it is a generic enough framework that any differentiable loss function can be used. Decision trees are used as the weak learner in gradient boosting. Trees are constructed in a greedy manner, choosing the best split points based on purity scores like Gini or to minimize

the loss. It is common to constrain the weak learners in specific ways, such as a maximum number of layers, nodes, splits or leaf nodes. Trees are added one at a time, and existing trees in the model are not changed. A gradient descent procedure is used to minimize the loss when adding trees. The output for the new tree is then added to the output of the existing sequence of trees in an effort to correct or improve the final output of the model. A fixed number of trees are added or training stops once loss reaches an acceptable level or no longer improves on an external validation dataset. Traditionally, gradient descent is used to minimize a set of parameters, such as the coefficients in a regression equation or weights in a neural network. After calculating error or loss, the weights are updated to minimize that error. The output for the new tree is then added to the output of the existing sequence of trees in an effort to correct or improve the final output of the model. A fixed number of trees are added or training stops once loss reaches an acceptable level or no longer improves on an external validation dataset.

5.5.1B MEAN SQUARE ERROR

```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import classification_report
from sklearn.model_selection import GridSearchCV

baseline = GradientBoostingRegressor(learning_rate=0.1, n_estimators=80,max_depth=3
baseline.fit(x_train,y_train)
y_pred = baseline.predict(x_test)
```

```
Mean Absolute Error: 0.07813228774423883
Mean Squared Error: 0.015716207191123707
Root Mean Squared Error: 0.12536429791261827
Accuracy: 78.75805046721074
```

Fig.22 Mean square error of GBM

5.5.1C OUTPUT

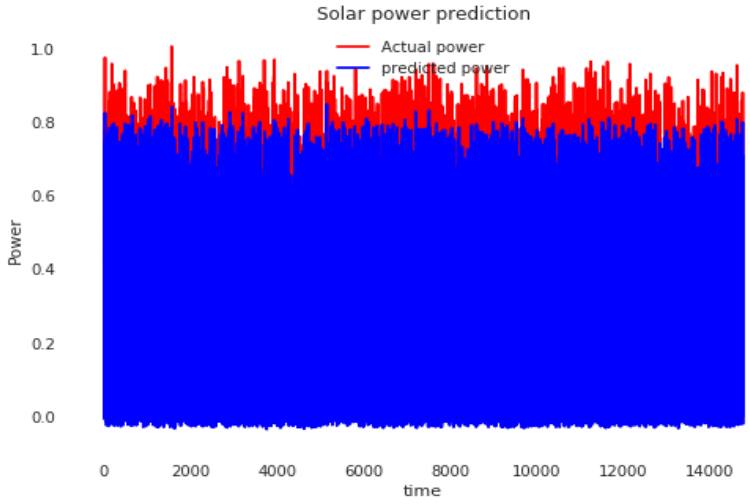

Fig.23 Output graph of GBM

5.5.2A XG BOOST ALGORITHM

XG Boost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data, artificial neural networks tend to outperform all other algorithms or frameworks. All the possible solutions to a decision based on certain conditions are represented graphically. Bootstrap aggregating is an ensemble meta algorithm which combines the predictions from multiple decision trees through a majority voting mechanisms. Random forest is a Bagging based (Bootstrap aggregation) algorithm where only a subset of features are selected at random to build a forest which is a collection of trees.

Then the models are built sequentially by minimising the errors from the previous models while boosting the influence of high performing models. Now, the gradient boosting employs the gradient descent algorithm to minimise the errors in sequential models. The optimised gradient boosting algorithm can now handle the missing values and regularization to over fitting by parallel processing and tree pruning.

XG Boost approaches the process of sequential tree building using parallelized implementation. This is possible due to the interchangeable nature of loops used for building base learners; the outer loop that enumerates the leaf nodes of a tree, and the second inner loop that calculates the features. This nesting of loops limits parallelization because without completing the inner loop (more computationally demanding of the two), the outer loop cannot be started. Therefore, to improve run time, the order of loops is interchanged using initialization through a global scan of all instances and sorting using parallel threads. This switch improves algorithmic performance by offsetting any parallelization overheads in computation

5.5.2B MEAN SQUARE ERROR

```
import xgboost as xgb
#data_dmatrix = xgb.DMatrix(data=x,label=y)
xg_reg = xgb.XGBRegressor(objective ='reg:linear', colsample_bytree = 0.8,
                max_depth = 15, alpha = 1, n_estimators = 180, nround=10000
  seed= 19,
  silent= True,
  skip_drop = 0.0,
  subsample= 0.6,max_bin= 256)
xg_reg.fit(x_train,y_train)
y_pred = xg_reg.predict(x_test)
```

```
Mean Absolute Error: 0.06938665808482576
Mean Squared Error: 0.014415659596921782
Root Mean Squared Error: 0.12006523059121563
Accuracy: 80.51586429754956
```

Fig.24 Mean square error of XG Boost

5.5.2C OUTPUT

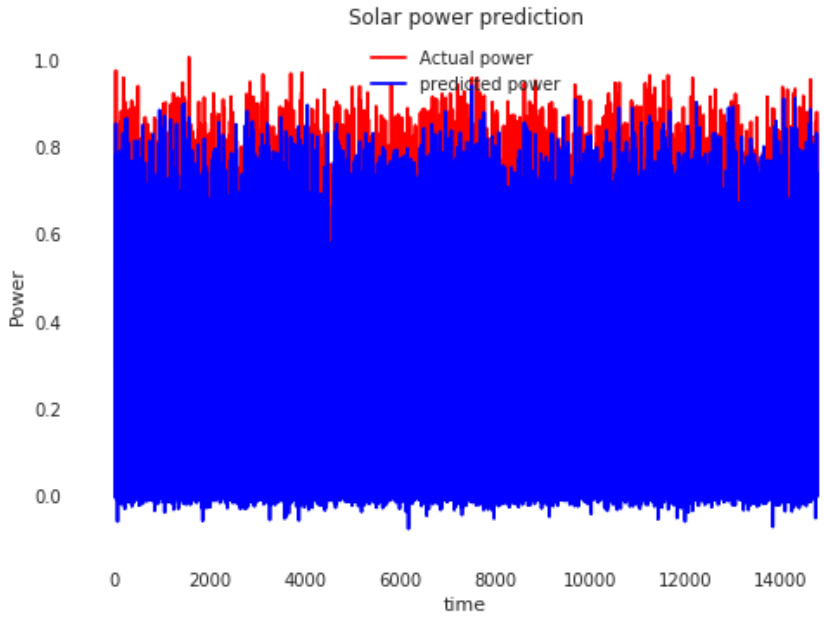

Fig.25 Output graph of XG Boost

5.5.3A RANDOM FOREST

The random forest combines hundreds of decision trees, trains each one on a slightly different set of the observations, splitting nodes in each tree considering a limited number of the features. The final predictions of the random forest are made by averaging the predictions of each individual tree. Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all

trees in the forest.When training, each tree in a random forest learns from a random sample of the data points.

5.5.3B MEAN SQUARE ERROR (MSE)

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import accuracy_score
from sklearn import metrics
regressor = RandomForestRegressor(n_estimators= 100, random_state=0)
regressor = regressor.fit(x_train, y_train)
y_pred = regressor.predict(x_test)
```

```
Mean Absolute Error: 0.06390994571215754
Mean Squared Error: 0.01382559414183301
Root Mean Squared Error: 0.11758228668397724
accuracy: 0.8131339390921802
```

Fig.26 Mean square error of Random Forest

5.5.3C OUTPUT

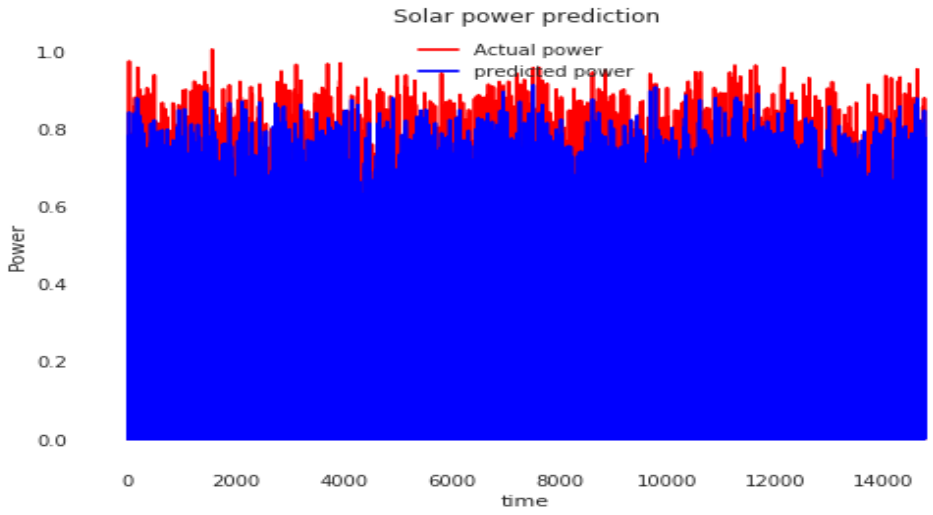

Fig.27 Output graph of Random Forest

ALGORITHM NAME	MEAN SQUARED ERROR	ACCURACY (%)
Linear Regression	0.2090907	40.9%
ANN	0.0775	44.5%
Support Vector Machine	0.0299	59.55%
Gradient Boosting Machine	0.015	78.75%
XG Boost	0.0144	80.51%
Random Forest	0.0138	81.31%

Fig.28 MSE chart for machine learning algorithms

5.6 CONCLUSION

Based on the analysis of different machine learning algorithms for analysis of solar data it is observed that Random forest is providing highest accuracy of 81.31 with less mean square error of 0.0138.So for the real time data analysis this algorithm can be used to get the accurate and effective prediction.

Chapter 6
HARDWARE IMPLEMENTATION FOR SOLAR DATA PREDICTION

A hardware setup is necessary to predict the solar data based on a real time system, which is shown in Figure.29.

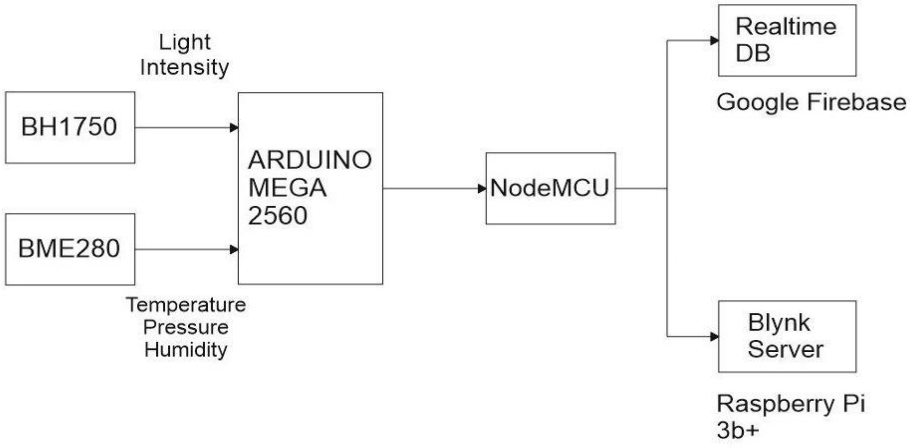

Fig.29 Data acquisition System

6.1 GETTING THE DATA FROM THE SENSORS (BH1750 AND BME280)

The data is received from the BME280 and BH1750 is connected to Arduino-mega where it is interfaced with NodeMCU to transfer the data to Blynk.

The real time data is sent to the Blynk and stored as CSV file for each sensor as different names.

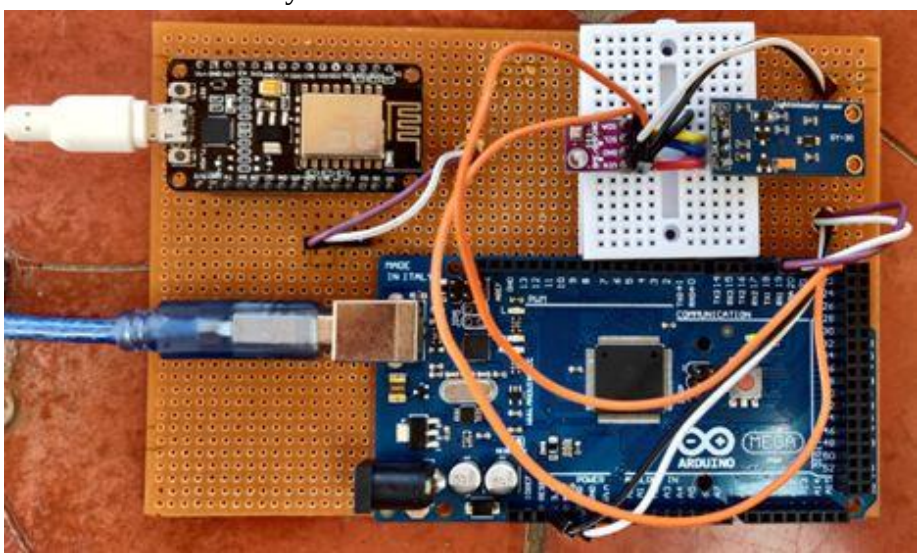

Fig.30 The sensors interfaced with NodeMCU

6.2 OPENWEATHERMAP

OpenWeatherMap is an online service that provides weather data. It is owned by OpenWeather Ltd, headquartered in London, United Kingdom. It provides current weather data, forecasts and historical data. OpenWeatherMap uses VANE platform for processing and distributing data to

millions of user at any instant. The data is collected from NOAA GFS Model and Environmental Canada.
OWM collects data from weather stations and forecasts of meteorological services and research laboratories, combining long-term and short-term forecasts with real-time data from weather stations, processing them and immediately updating current weather and forecasts in their API. Data is stored in the VANE database and processing with their unique algorithms to create interpolated data of current weather conditions anywhere in the world, as well as into a variety of weather maps. Finally, OWM provides the API which allows access to all the weather data including a variety of maps.
The sources are listed below:

- NOAA
- Environment Canada
- European Centre for Medium-Range Weather Forecasts (ECMWF)
- Japan Meteorological Agency
- METAR data from airports
- APRS network

The API used from OWM can be used to access 5 day forecast as shown below.
http://api.openweathermap.org/data/2.5/forecast?q=Ettimadai&appid=f85f698defeb58ebd90e9157445c24d6

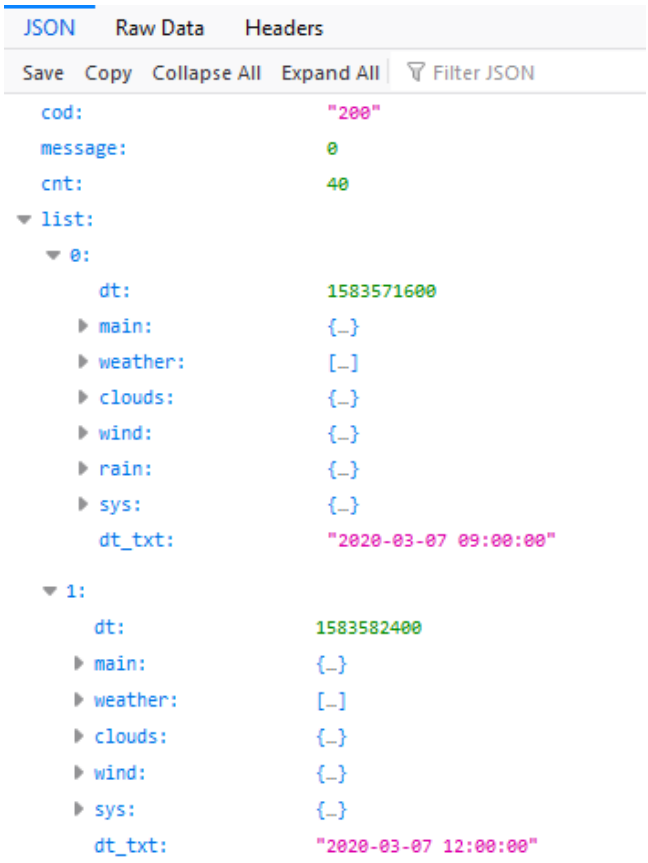

Fig.31 Data retrieved from OpenWeatherMap API

Chapter 7
USER INTERFACE FOR SOLAR POWER PREDICTION

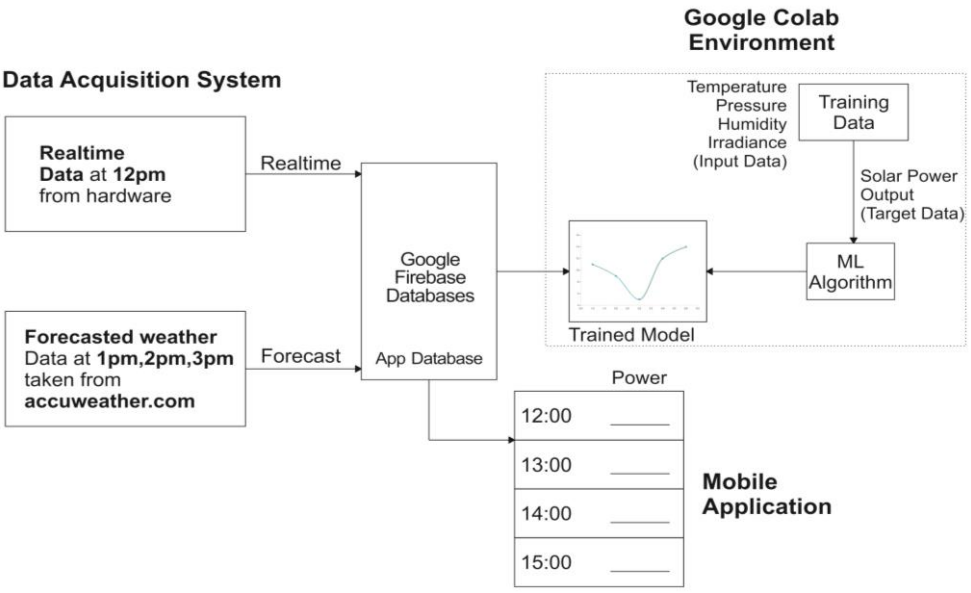

Fig. 32 Data exchange flow diagram

7.1 BLYNK APP

Blynk is an online platform to control Arduino, Raspberry Pi and the likes over the Internet. It's a digital dashboard where users can build a graphic interface for their project. Blynk is used here for the purpose of controlling as well as monitoring the real time data from any respected hardware setups.

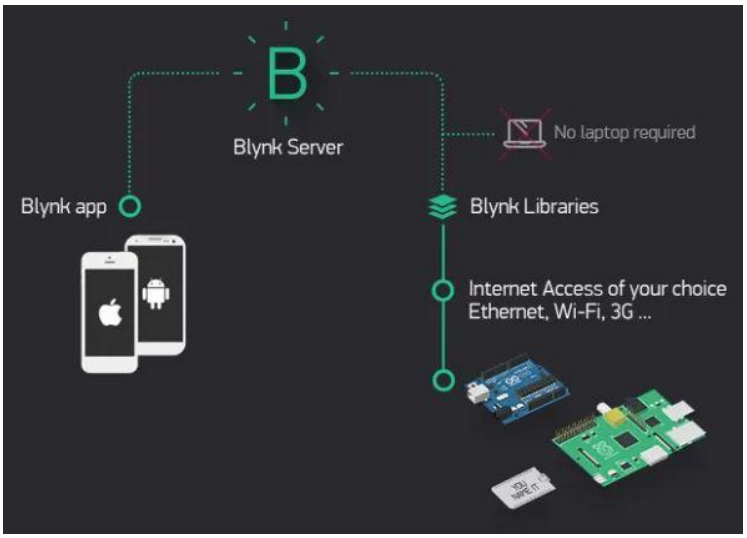

Fig. 33 Blynk app

It mostly supports Arduino boards, Raspberry Pi models, the ESP8266, Particle Core, and a handful of other common microcontrollers and single-board computers.
For our case the Blynk is used as a local server. The Raspberry pi 3, NodeMCU and a mobile phone with Blynk installed are connected to a local network.
The sensor data is collected in Blynk and stored as a CSV file for each parameter which can be separately downloaded. A new project is created by selecting NodeMCU from hardware which is present in the tools. The esp8266 library has been installed for monitoring the real time data from the respective sensors. Then a graphical interface is created by just dragging and dropping the widgets such as Temperature, pressure, humidity, irradiance. The connection type is given as Wi-Fi.
After creating the interface, an authorization token is sent to the user by the Blynk. All these widgets which were used are free of cost. History graph widget is selected from the widget box so that the data can be monitored and represented graphically.

	A	B	C
1	30	1.58E+12	0
2	30	1.58E+12	0
3	30	1.58E+12	0
4	30.27273	1.58E+12	0
5	30.2	1.58E+12	0
6	30	1.58E+12	0
7	30	1.58E+12	0
8	30	1.58E+12	0
9	30.63636	1.58E+12	0
10	30.8	1.58E+12	0
11	31	1.58E+12	0
12	31	1.58E+12	0
13	31	1.58E+12	0
14	31.45455	1.58E+12	0
15	31.09091	1.58E+12	0
16	31.6	1.58E+12	0
17	31.18182	1.58E+12	0
18	31	1.58E+12	0
19	31.2	1.58E+12	0
20	31.36364	1.58E+12	0
21	31.27273	1.58E+12	0
22	32	1.58E+12	0
23	32	1.58E+12	0
24	32.18182	1.58E+12	0

Fig.34 Real time data plotted in Blynk and data downloaded from Blynk (Temperature data)

6.2 FIREBASE:

The firebase is an online web application development platform used majorly for uploading the data and using API keys for accessing the data via an android application.

The API key from Openweather.com is used for getting the next 5 days forecast including that day's.

The API key used is

http://api.openweathermap.org/data/2.5/forecast?q=Ettimadai&appid=f85f698defeb58ebd90e9157445c24d6

Once the data has been requested, data with timestamp from 9 am to 6 pm is separated using pandas library and uploaded to firebase.

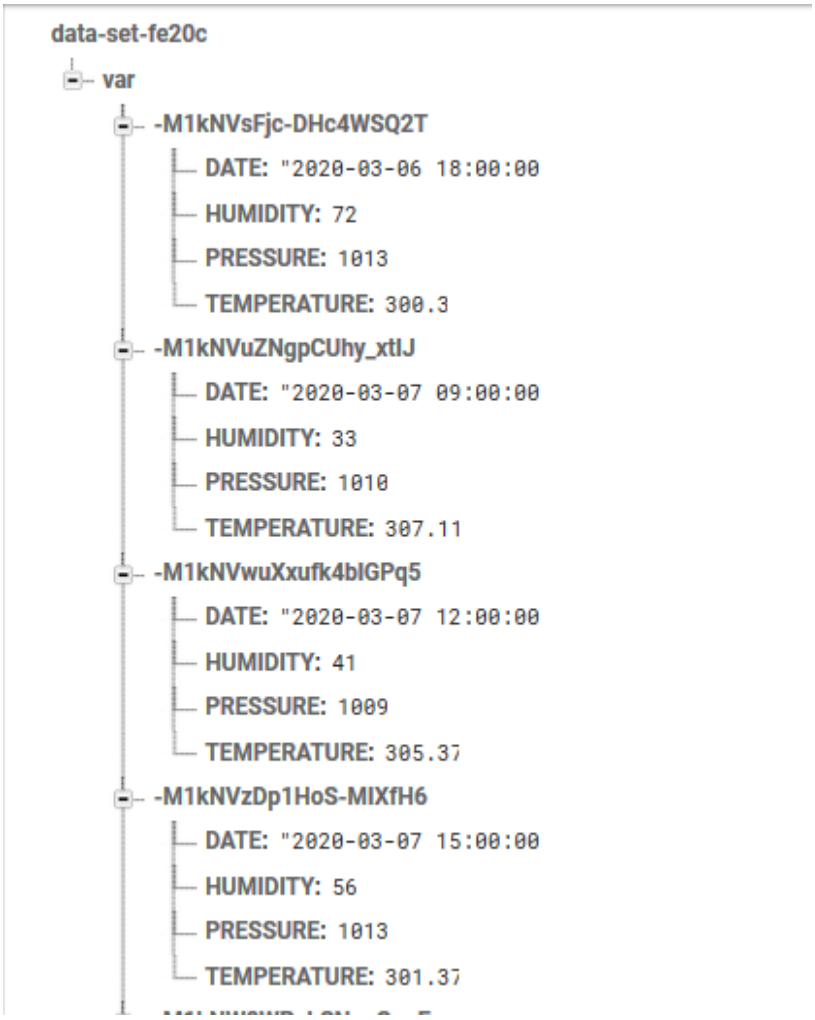

Fig. 35 Forecasted data stored in firebase

6.3 FORECASTING IN REAL TIME

The meteorological data and PV output data of academic block 2 (Amrita University Coimbatore) is collected and then Random Forest Regression is implemented on this data. The accuracy of the prediction varies with the quantity of data available. The current accuracy as measured is 77.52% and will rise if the duration of collection is more than 1 month. The graphical representation is given in Figure.38 along with the location in Figure 36 and the collected real data in Figure.37.

LOCATION FOR OUR HARDWARE DATA COLLECTION

```
[4] import folium
    my_map1 = folium.Map(location = [10.9027, 76.9006], zoom_start = 19 )
    tooltip = 'Click me!'
    folium.Marker([10.903968 , 76.898265], popup='<i>ROOF TOP OF AB2 (50KW)</i>', tooltip=tooltip).add_to(my_map1)
    my_map1
```

Fig. 36 Hardware Location

Time	Temperaure (°C)	Pressure (hPa)	Humidity (%)	Lux (lx)	Power (KW)	Power (MW)
8:30:00 AM	30	1014.6	61.3	3698.666	13605	0.13605
8:35:00 AM	30.2	1015	60.7	12903.749	14404	0.14404
8:40:00 AM	30.8	1015	59	15046.668	15151	0.15151
8:45:00 AM	31.09090909	1015	56.636364	16304.471	15892	0.15892
8:50:00 AM	31.36363636	1015	57.454545	11277.348	9596	0.09596
9:00:00 AM	32.9	1015	53.7	19558.501	13121	0.13121
9:05:00 AM	33	1015	52	22714.999	18908	0.18908
9:10:00 AM	34	1015	50.454545	22488.939	19779	0.19779
9:15:00 AM	34.1	1015	49.8	23442.916	20539	0.20539
9:20:00 AM	36	1015	45.3	25063.999	21229	0.21229
9:25:00 AM	36.27272727	1015	44.636364	26476.592	21999	0.21999
9:30:00 AM	35.90909091	1015	44.272727	28230.833	22762	0.22762
9:35:00 AM	37	1015	41	29538.833	23379	0.23379
9:40:00 AM	37.36363636	1015	40.727273	31366.917	22980	0.2298
9:45:00 AM	36.81818182	1015	41	31605.454	22249	0.22249
9:50:00 AM	37.27272727	1015	40.181818	33271.359	22778	0.22778
9:55:00 AM	37.8	1015	39	35646.163	23671	0.23671
10:00:00 AM	36.72727273	1015	41.181818	35525.449	24160	0.2416
10:05:00 AM	37.09090909	1015	40.272727	33072.345	25091	0.25091
10:10:00 AM	38.27272727	1015	38.454545	33632.876	25842	0.25842
10:15:00 AM	38.5	1015	38.7	33951.164	25868	0.25868
10:20:00 AM	38.81818182	1014.272727	37.181818	37142.797	26132	0.26132
10:25:00 AM	38	1014.363636	38.181818	37526.511	25978	0.25978
10:30:00 AM	38.27272727	1014	37.818182	38254.088	26123	0.26123

Fig. 37 Real time data collection

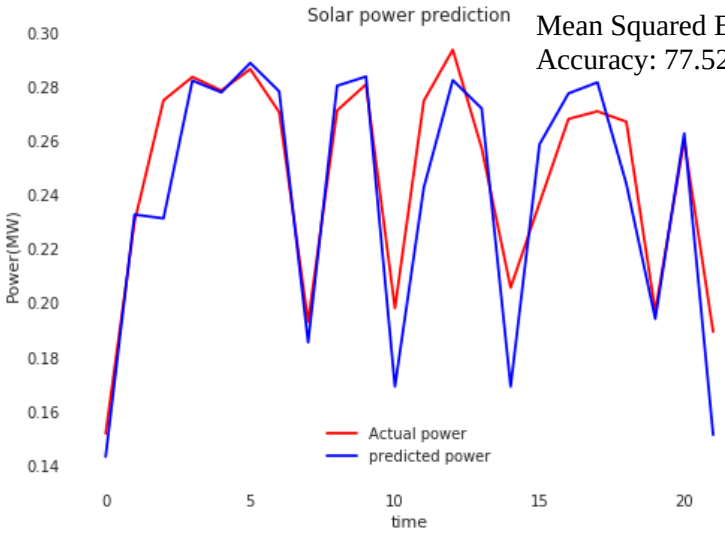

Fig. 38 Actual value vs. prediction

6.4 SOLAR FORECASTING APPLICATION

6.3.1 DEVELOPMENT

a)Adobe XD

Adobe XD is a vector-based user experience design tool for web apps and mobile apps developed and published by Adobe Inc. It is available for macOS and Windows, although there are versions for iOS and Android to help preview the result of work directly on mobile devices.

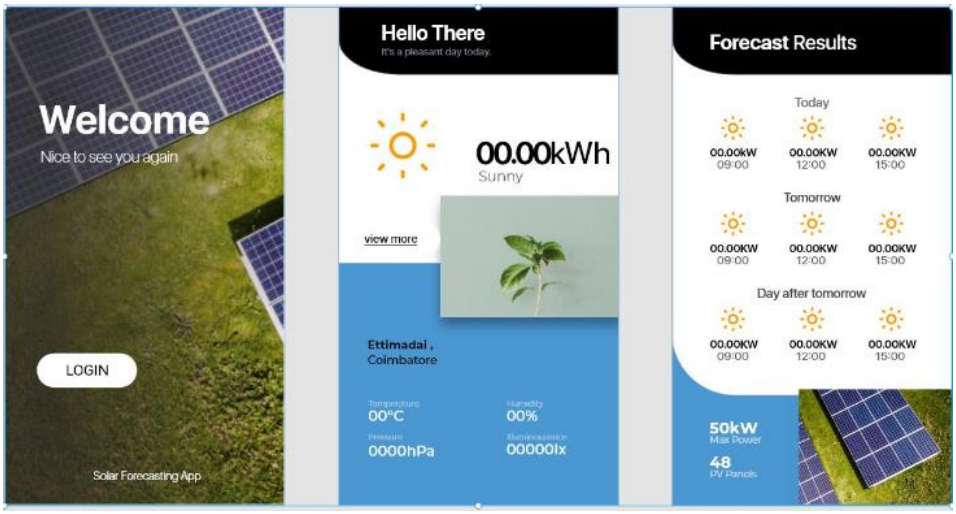

Fig.39 Adobe XD application design

b) Android Studio

Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development.

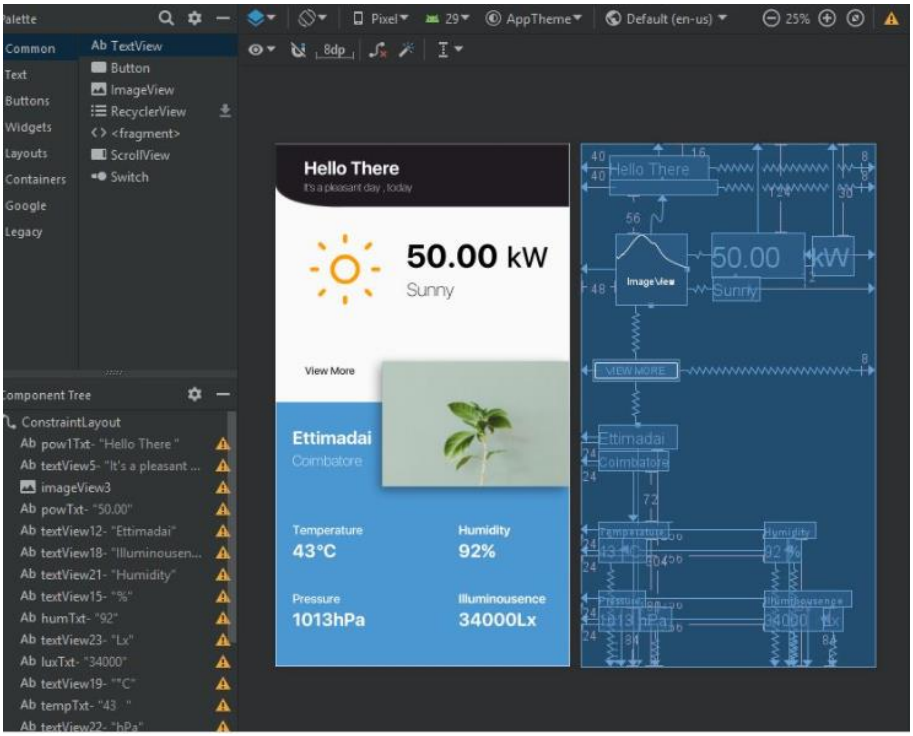

Fig.40 User interface design

6.4.2 INTERFACE AND WORKING

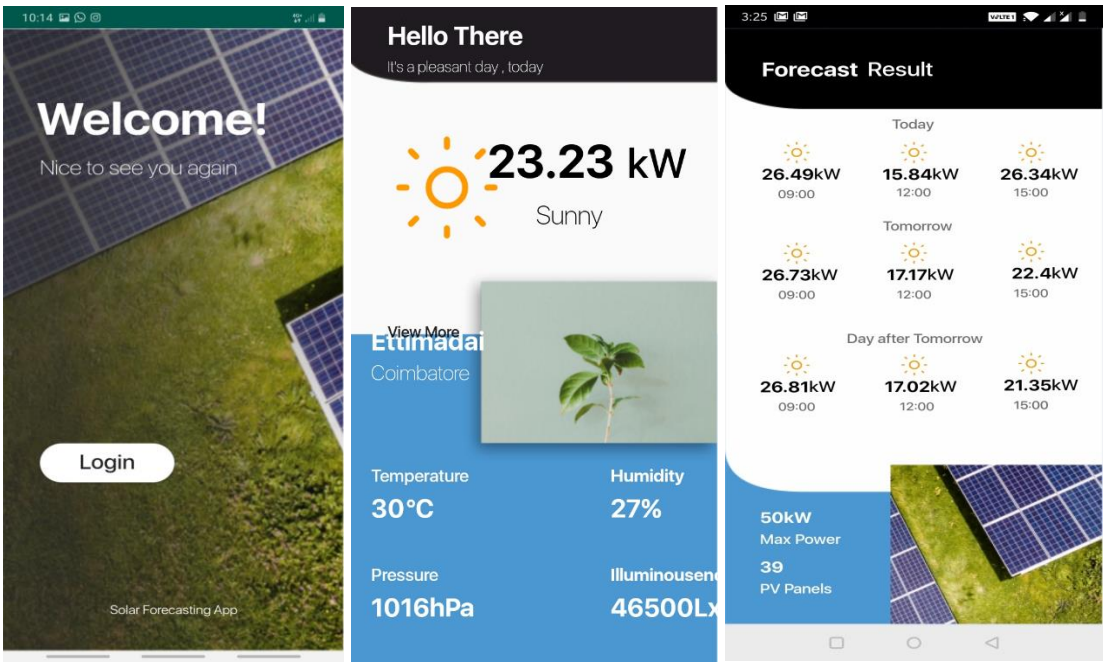

Fig.41 Android application for user

The application developed for user interface consists of three pages:

- User login page
- Home page/Real time data display
- Forecast display

The Firebase database consists of two separate parents for real time data storage and forecast data storage. As shown in the figure above the real time data consists of temperature, humidity, pressure, lux and PV output. Whereas the forecast data consists of PV output for the next three days.

Chapter 8
CONCLUSION

8.1 CONCLUSION

A real time Data acquisition system to log weather parameters and solar data from installed 50kW Photovoltaic system which is located at Academic Block II, Amrita Vishwa Vidyapeetham, Coimbatore has been developed.

The hardware for data-logging as well as extraction of the data of weather parameters such as temperature, humidity, pressure and Lux has been created. The data for the prediction of weather is collected from OpenWeatherMap using the API provided.

Various machine learning algorithms such as Linear regression, Random forest and Neural Network algorithms such as Artificial Neural Network, LSTM have been implemented to find the best algorithm with least Mean square error, high accuracy and short execution time.

An Android Application has been created to display the real time and forecasted power outputs.

A Real-time database Google Firebase is updated to store the real time data and the predicted data.

REFERENCES

[1] S. K. Chow, E. W. Lee, and D. H. Li, "Short-term prediction of photovoltaic energy generation by intelligent approach," Energy and Buildings, vol. 55, 2012, pp. 660-667.

[2] J. Liu, W. Fang, X. Zhang, and C. Yang, "An improved photovoltaic power forecasting model with the assistance of aerosol index data," IEEE Transactions on Sustainable Energy, vol. 6, 2015, pp. 434-442.

[3] K. Malik, B. A. Bhatti and F. Kamran, "An approach to predict output of PV panels using weather corrected global irradiance," International Conference on Intelligent Systems Engineering (ICISE), Islamabad, 2016, pp. 111-117.

[4] M. G. De Giorgi, P. M. Congedo and M. Malvoni, "Photovoltaic power forecasting using statistical methods: impact of weather data," IET Science, Measurement & Technology, vol. 8, no. 3, May 2014, pp. 90-97.

[5] Kazem, Hussein A and Chaichan, Miqdam and Al-Shezawi, I.M. and Al-Saidi, H.S. and Al-Rubkhi, H.S. and Al-Sinani, Jamila and Al-Waeli, Ali.” Effect of Humidity on the PV Performance in Oman.”, Asian Transactions on Engineering, 2012.

[6] Gökmen, Nuri& Hu, Weihao&Hou, Peng& Chen, Z & Sera, Dezso&Spataru, Sergiu.” Investigation of wind speed cooling effect on PV panels in windy locations”, Renewable Energy, 2016.

[7] I. Khan, H. Zhu, J. Yao, and D. Khan, “Photovoltaic power forecasting based on artificial neural network software engineering method,” in 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), Nov 2017, pp. 747–750.

[8] X. Sun and T. Zhang, "Solar Power Prediction in Smart Grid Based on NWP Data and an Improved Boosting Method," IEEE International Conference on Energy Internet (ICEI), Beijing, 2017, pp. 89-94.

[9] J. Wu and C. K. Chan, “The prediction of monthly average solar radiation with ann and arima,” in 11th International Conference on Machine Learning and Applications, vol. 2, Dec 2012, pp. 469–474.

[10] R. H. Inman, H. T. C. Pedro, and C. F. M. Coimbra, “Solar forecasting methods for renewable energy integration,” Prog. Energy Combust. Sci., vol. 39, no. 6, Dec. 2013, pp. 535–576.

[11] Cyril Voyant, Gilles Notton, SoterisKalogirou , Marie-Laure Nivet ,Christophe Paoli , FabriceMotte , Alexis Fouilloy,“ Machine learning methods for solar radiation forecasting: A review,” Renewable Energy, 2017.

[12] W. Lee, K. Kim, J. Park, J. Kim and Y. Kim, "Forecasting Solar Power Using Long-Short Term Memory and Convolutional Neural Networks," in *IEEE Access*, vol. 6, pp. 73068-73080, 2018.

[13] I. S. Isa, S. Omar, Z. Saad, N. M. Noor and M. K. Osman, "Weather Forecasting Using Photovoltaic System and Neural Network," *2010 2nd International Conference on Computational Intelligence, Communication Systems and Networks*, Liverpool, 2010, pp. 96-100.

[14] Shubham Billus, Shivam Billus, Rishab Behl, "Weather Prediction through Sliding Window Algorithm and Deep Learning," *International Journal of Scientific Research in Computer Science and Engineering*, Vol.6, Issue.5, pp.20-24, 2018.