ABSTRACT

The rise of renewable energy presence in power grids globally necessitates the need for accurate solar power prediction as solar power is unpredictable by nature. Grid balancing is the matching of supply with demand; the failure to properly balance load distribution can paralyze the grid.
Standard methods such as using weather data are generalized for an entire location and aren't accurate. Forecasting using onsite data is more accurate.

In this project we've trained prediction models with onsite data germane to forecasting.
The power generated from a 5W solar panel was measured using a voltage divider collected using an Arduino. The collected time series dataset was converted into a csv file using PLX-DAQ (Parallax Data Acquisition)

The PV power generated was seasonal with no trend implying non stationarity. The data was mathematically transformed to make it stationary and denoised using digital signal processing techniques. The time taken to train the SARIMA model was 13.942 seconds and it produced a forecast with a mean squared error score of 0.3453, implying a high accuracy.

Our analysis shows that the prediction is fairly accurate and coupled with a low training time, it implies feasibility in onsite deployment. The forecast values can be coupled with non-renewable energy production to ensure proper grid balancing.

LIST OF FIGURES

Contents

1 INTRODUCTION

1.1 Project Pipeline

Predicting solar power generation is crucial for power grid balancing; therefore, the archetype of solar power forecasting is a model that is accurate with minimal negative feedback. Standard methods such as weather data are antithetical to this as they are generalized for an entire location and they do not possess minimal negative feedback as weather data is generally available hourly as shown by Majidpour *et al.* [1]

Training prediction models with onsite data germane to forecasting is more accurate and possesses minimal negative feedback if computed frequently. The goal of this project is to develop a short-term power forecasting model and the project pipeline is illustrated in figure 1.1.

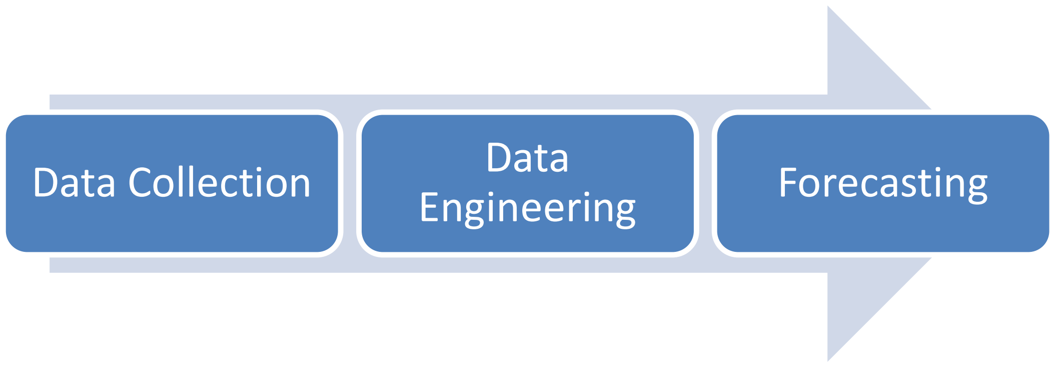


Figure 1.1: Project Pipeline

The project pipeline consists of three stages: data collection, data engineering, and forecasting.
In data collection, real-time measurement devices collect data. Data Engineering and forecasting compose the machine learning component where the collected data is transformed to train machine learning algorithms for forecasting future solar power generation as shown by Park and Ahn.[2]

1.2 Data Collection

This stage involves measuring the power generated from the solar panel and storing it in a format that is suitable for analysis. To acquire and collect Real-time data of PV system we need an appropriate method which is cost effective as well as easy to maintain. Using a microcontroller board such as an Arduino Uno and Excel to store data is a cheap and easy-to-maintain method as shown by El Hammoumi *et al.*[3]

1.3 Data Engineering

Machine learning is the study of computer algorithms that improve automatically through experience and by the use of data. It's a part of artificial intelligence. In the data engineering phase, data is engineered to make it compatible with machine learning models.
The collected data is transformed using a suitable mathematical transformation, and digital signal processing methods are deployed to remove any prevailing noise in the data.

1.4 Forecasting

In this stage, the data is bifurcated into two parts: the training set and the testing set. The training set is to train the model to forecast the data in the testing set. The mean squared error score is computed to assess the accuracy of our forecasting model as shown by Sorkun, Incel and Paoli.[4]

2 BACKGROUND THEORY

2.1 Solar Power

A solar cell is a thin semiconductor wafer made from two layers of silicon. One layer is positively charged, and the other negatively charged, forming an electric field. When light energy from the sun strikes a photovoltaic solar cell, it energizes the cell and causes electrons to 'come loose' from atoms within the semiconductor wafer. Those loose electrons are set into motion by the electric field surrounding the wafer, and this motion creates an electrical current.[5]

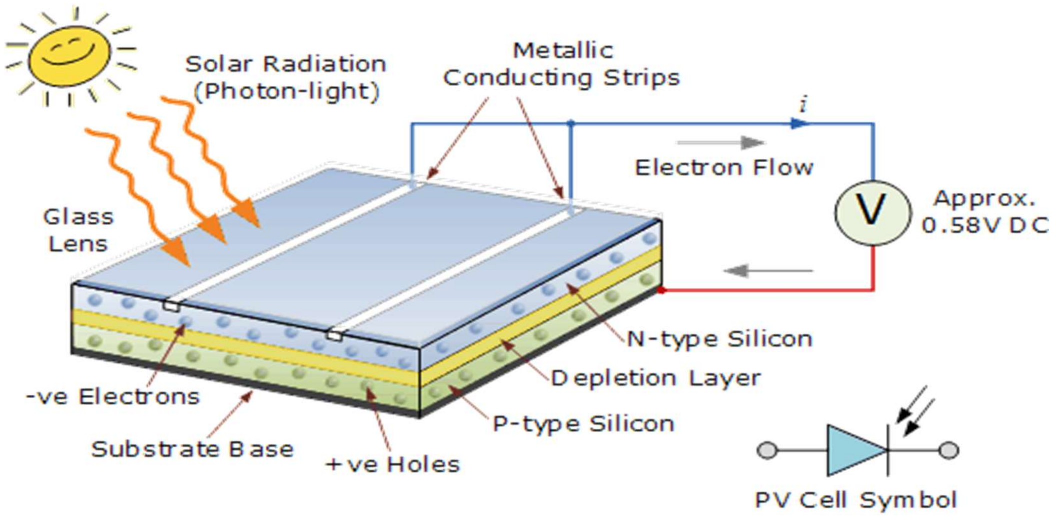


Figure 2.1: Solar Panel Working

2.2 Time Series Data

Data collected at adjacent points in time is referred to as time series data. A time series is actually a realization of a stochastic process going on in the background.

Stochastic processes are a sequence/collection of random variables which are functions of time. Instead of considering fixed random variables X, Y, or even sequences of independent and identically distributed random variables, we consider sequences X_0, X_1, X_2 …. Where X_t represents some random quantity at time t.

The value X_t depends on the quantity X_{t-1} at time t-1, or even the value X_s for other times $s<t$.

A stationary time series is what we're going to be building our models on. A stationary time series has the following properties:

- No systematic change in mean

- No systematic change in variation
- No periodic fluctuations

In short, one portion of the time series has properties that are similar to the other parts.

2.3 Real-time measurement system

2.3.1 Arduino UNO

The Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 Analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button.[6]
Microcontrollers like Arduino Uno are capable of understanding Binary – 0's and 1's. So, if we have 0 Volt it will correspond to 0 and similarly 5 Volt will correspond to 1. Microcontrollers like Arduino Uno have an Analog to Digital Converter (ADC) built into them that allows us to convert these Analog values into Digital values which the microcontroller can understand. For our project we will be working with Analog 0 (A0) pin. Arduino boards contain a multichannel, 10-bit Analog to digital converter.[7]

2.3.2 Voltage divider

A voltage divider is a simple circuit constructed using two resistors. This reduces the voltage and has the added benefit of increasing the input impedance, which means that the measurement device will not load down the circuit it is trying to measure and distort the reading.[8]

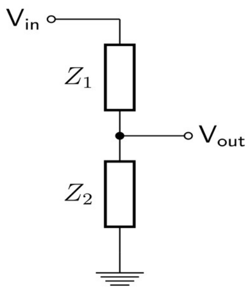


Figure 2.2: Voltage Divider

In our circuit $V_{Reference}$ is 5 Volt and V_{Source} is 17 Volt maximum as mentioned in datasheet provided for our PV panel, so we need to design a voltage divider with resistor values keeping this in mind. We need to have V_{out} in between 4.0 Volt to 5 Volt to keep our Arduino safe from adverse effect of overheating. Using the Voltage divider formula-

$$V_{out} = \frac{V_{Source} \times R_2}{R_2 + R_1}$$

If we select $R_1 = 1000\ \Omega$ and $R_2 = 330\ \Omega$ we get $V_{out} = 4.22$ Volt which is in between 4.0 Volt to 5 Volt. Hence, we will proceed with these values for our resistor. Now the measurement device in our case Arduino Uno will not load down the circuit it is trying to measure and distort the reading. Here the Voltage divider circuit is used as a "step-down" to convert large voltage into smaller values. These smaller values are then converted by ADC from digital value to an Analog value.[9] then we calculate the actual voltage using this equation-

$$V_{in} = ADC\ Value\ \div \frac{R_2}{R_2 + R_1}$$

2.3.3 Arduino IDE

The Arduino IDE software makes it possible to write, modify a program and convert it into a series of instructions understood by the microcontroller of an Arduino board. The IDE can run on Windows, Linux, or Mac. The Arduino board in this project is programmed by the IDE that serves as a code editor and compiler and can transfer the program code to the microcontroller through a USB cable.

2.3.4 PLX-DAQ (Parallax Data Acquisition)

The data logging of the measurement data was done using an add-on called PLX-DAQ (Parallax Data Acquisition). It is an open-access program used to establish communication between Excel on a Windows Computer and any device that supports serial port protocol. We need to connect the software to the same serial port through which our Arduino IDE is connected to Arduino Uno.[10]

2.4 Data Analysis

2.4.1 Pandas

Pandas is a Python library of data structures and tools for working with structured data sets common to statistics, finance, social sciences, and many other fields. The library provides integrated, intuitive routines for performing common data manipulations and analysis on such data sets. It serves as a strong complement to the existing scientific Python stack while implementing and improving upon the kinds of data manipulation tools found in other statistical programming languages such as R.

2.4.2 Jupyter Notebook

The Jupyter notebook is an open-source, browser-based tool functioning as a virtual lab notebook to support workflows, code, data, and visualizations detailing the research process. It is machine and human-readable, which facilitates interoperability and scholarly communication. These
notebooks can live in online repositories and provide connections to research objects such as datasets, code, methods documents, workflows, and publications that reside elsewhere

2.4.3 Matplotlib

Matplotlib is a portable 2D plotting and imaging package aimed primarily at visualization of scientific, engineering, and financial data. matplotlib can be used interactively from the Python shell, called from python scripts, or embedded in a GUI application (GTK, Wx, Tk, Windows)..

2.4.4 Statsmodels

Statsmodels is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration. The results are tested against existing statistical packages to ensure that they are correct. The package is released under the open source Modified BSD (3-clause) license.[11]

2.5 Discrete Wavelet Transform

The discrete wavelet transform (DWT) is any wavelet transform where the wavelets are sampled discretely. DWT's advantage over Fourier transforms is temporal resolution: it captures both location information and frequency. The DWT leads to a sparse representation for many real-world signals and images. It concentrates signal and image features in a few large-magnitude wavelet coefficients. Wavelet coefficients which are small in value are typically noise and can be removed without affecting the signal quality.

2.6 ARIMA (Auto Regressive Integrated Moving Average)

An ARIMA model is a class of statistical models for forecasting time series data. All ARIMA models are built on stationary time series data. A stationary time series data has the following properties:

1) No systematic change in mean
2) No systematic change in variation
3) No periodic fluctuations

If the time series data in question is seasonal, we use SARIMA (Seasonal Auto Regressive Integrated Moving Average) models. A Moving Average model is built from a finite set of innovations. An Auto Regressive model is built from a current innovation Z_t together with knowledge of a finite set of prior states (the X's) as shown by Atique *et al.*[12]

Let's take the Zt's to be white noise $Zt{\sim}iid(0, \sigma^2)$; *iid* refers to independent and identically distributed random variables.

$$Moving\ Average(q)\ process\colon\ X_t = \theta_0 Z_t + \theta_1 Z_{t-1} + \cdots + \theta_q Z_{t-q}$$

$$X_t = Z_t + history$$

$$Auto\ Regressive(p)\ process\colon X_t = Z_t + \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p}$$

The SARIMA model is represented by

$$(p, d, q)\ \ (P, D, Q)_m$$

The first bracket corresponds to the non-seasonal part of the model and the second corresponds to the seasonal part of the model.

p = Non seasonal auto regressive order
d = Non seasonal differencing
q = Non seasonal moving average order
P = Seasonal auto regressive order
D = Seasonal differencing
Q = Seasonal moving average order
m = Time span of repeating seasonal pattern

3 METHODOLOGY

3.1 Data Collection

Now to make a real-time monitoring system using an Arduino Uno and voltage divider. We will use a breadboard and some male-to-male jumper wires.

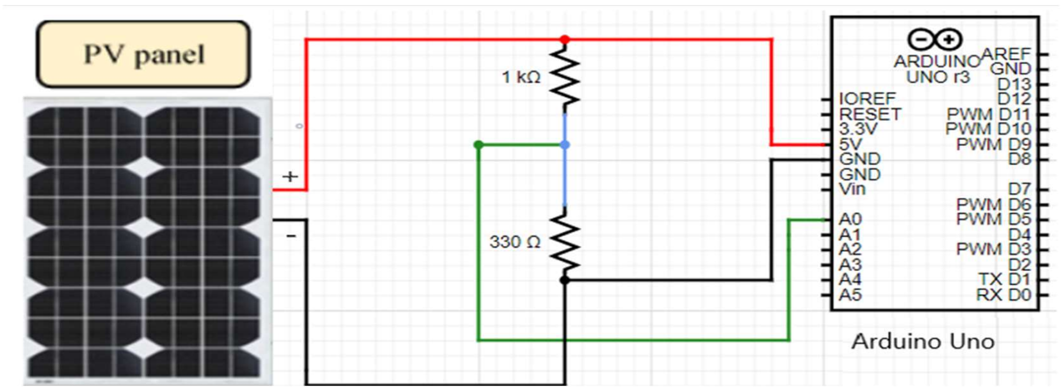


Figure 3.1: Circuit Diagram

We will connect the positive wire from the PV panel to our first R_1 and then connect $V_{Reference}$ (5 Volt) to the same column. Then we will connect the Ground terminal of the Arduino and PV Panel to R_2. After that, we will connect the Analog 0 (A0) pin of the Arduino in between both the resistors. Now our circuit is complete

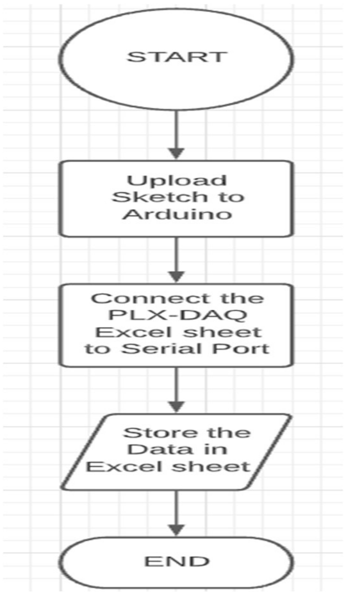
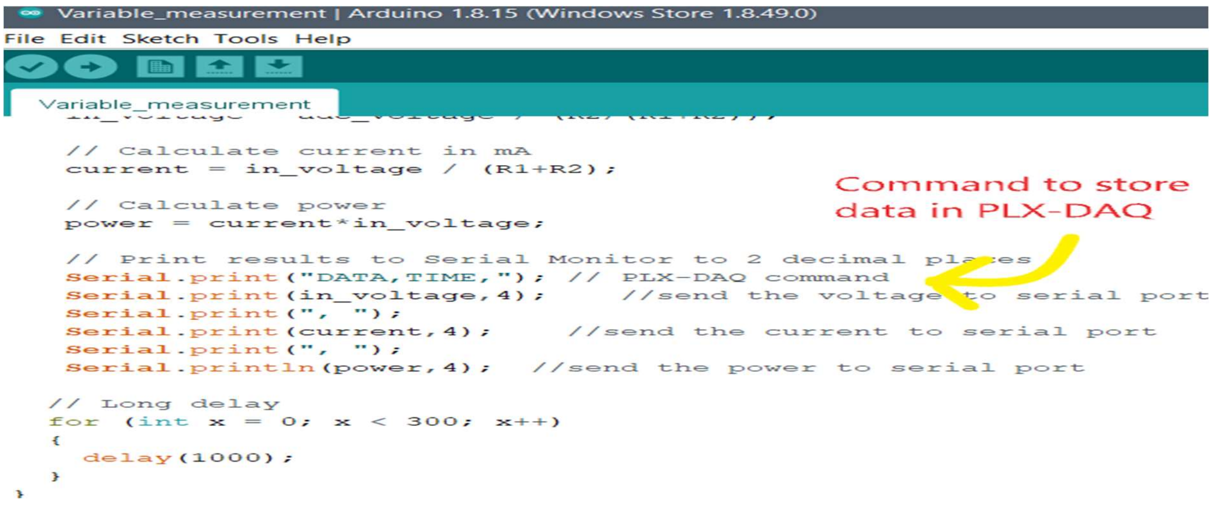

START
Upload Sketch to Arduino
Connect the PLX-DAQ Excel sheet to Serial Port
Store the Data in Excel sheet
END

Figure 3.2: Data Collection Flowchart

To collect the data from the Real-time monitoring system. We will upload the written sketch (Code) from Arduino IDE to Arduino Uno.[13]. Once the sketch is successfully uploaded error-free then we open the PLX-DAQ Excel Sheet. After that, we will connect it to the same serial port through which we uploaded our Sketch. Once the connection is completed, PLX-DAQ will start collecting and storing data in an Excel file in the same way we specified in the code using the PLX-DAQ command.


Variable_measurement | Arduino 1.8.15 (Windows Store 1.8.49.0)
File Edit Sketch Tools Help
Variable_measurement
Command to store data in PLX-DAQ

```
    // Calculate current in mA
    current = in_voltage / (R1+R2);

    // Calculate power
    power = current*in_voltage;

    // Print results to Serial Monitor to 2 decimal places
    Serial.print("DATA,TIME,"); // PLX-DAQ command
    Serial.print(in_voltage,4);     //send the voltage to serial port
    Serial.print(", ");
    Serial.print(current,4);     //send the current to serial port
    Serial.print(", ");
    Serial.println(power,4);   //send the power to serial port

  // Long delay
  for (int x = 0; x < 300; x++)
  {
    delay(1000);
  }
}
```

Figure 3.3: Arduino IDE interface

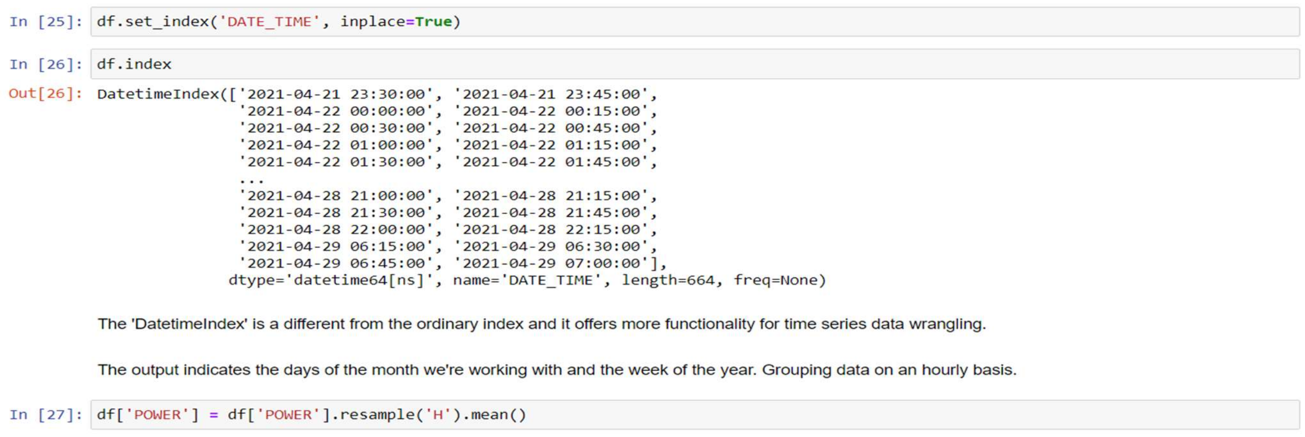
3.2 Data Transformation & Denoising

The time series data is further processed in Python using Anaconda's Jupyter Notebook. The collected CSV file from the previous stage is converted into a Pandas Dataframe data structure for further analysis as shown by McKinney[14].

The following Python libraries were used to analyse data in Python-

- Pandas
- Numpy
- Matplotlib
- Seaborn

The column containing the time at which the data was collected was converted into Pandas' 'DatetimeIndex' format. The 'DatetimeIndex' is a different from the ordinary index and it offers more functionality for time series data wrangling. The data was then upscaled from 15 minute intervals to 1 hour intervals by taking the mean.



```
In [25]: df.set_index('DATE_TIME', inplace=True)

In [26]: df.index

Out[26]: DatetimeIndex(['2021-04-21 23:30:00', '2021-04-21 23:45:00',
                        '2021-04-22 00:00:00', '2021-04-22 00:15:00',
                        '2021-04-22 00:30:00', '2021-04-22 00:45:00',
                        '2021-04-22 01:00:00', '2021-04-22 01:15:00',
                        '2021-04-22 01:30:00', '2021-04-22 01:45:00',
                        ...
                        '2021-04-28 21:00:00', '2021-04-28 21:15:00',
                        '2021-04-28 21:30:00', '2021-04-28 21:45:00',
                        '2021-04-28 22:00:00', '2021-04-28 22:15:00',
                        '2021-04-29 06:15:00', '2021-04-29 06:30:00',
                        '2021-04-29 06:45:00', '2021-04-29 07:00:00'],
                       dtype='datetime64[ns]', name='DATE_TIME', length=664, freq=None)
```

The 'DatetimeIndex' is a different from the ordinary index and it offers more functionality for time series data wrangling.

The output indicates the days of the month we're working with and the week of the year. Grouping data on an hourly basis.

```
In [27]: df['POWER'] = df['POWER'].resample('H').mean()
```

Figure 3.4: Time series data indexing

This data was then plotted using matplotlib and the ***seasonal_decompose()*** function from the ***statsmodels.tsa.seasonal*** python package to inspect trend, seasonality and residuals. The dataset must be stationary in order to train our forecasting models. A difference transformation was applied to the non-stationary time series to make it stationary. The differenced time series' stationarity was verified using the *Augmented Dickey-Fuller* statistical test.

Any data collection process is subject to noise and it can obscure useful patterns. Smoothing is used to extract those patterns. The time series data was denoised using the ***denoise_wavelet()*** function from the ***skimage.restoration*** Python package as shown by Randles *et al.*[15]

3.3 Forecasting

Our time series data is seasonal; therefore, we'll use the SARIMA model to forecast our solar power generation. SARIMA has 7 parameters: p, d, q, P, D, Q, and m. In this case m = 24. The values of p, d, q, P, D, and Q have to be chosen.[16] [17]

We enlisted the help of ***sm.tsa.graphics.plot_acf()*** and ***sm.tsa.graphics.plot_pacf()*** functions from the ***statsmodels.api*** Python library to plot the ACF (Autocorrelation Function) and PACF(Partial Autocorrelation function) plots to narrow down the range of parameters.

All combinations of SARIMA were computed using the ***pm.auto_arima()*** method from the ***pmdarima*** Python package for the range of parameters obtained from the ACF and PACF plots. The AIC (Akaike Information Criteria) metric was used to choose the SARIMA Model.

The last 5 days of the time series were used to train and test our model. The first 4 days were used to train the model (fitting the SARIMA Model) . The ACF and PACF plots are very objective measures of quality. The AIC metric is a more numerical measure of quality. Following parsimonious principles the model with the lowest AIC was chosen. This model was used to make a prediction for the 5th day and the prediction was compared with the actual data.

4 RESULT ANALYSIS

4.1 Data Wrangling

The collected time series data was converted into a Pandas Dataframe and the index was changed to Pandas' 'DateTime Index'. The resulting dataset had 2 columns: Time and the solar power generated.

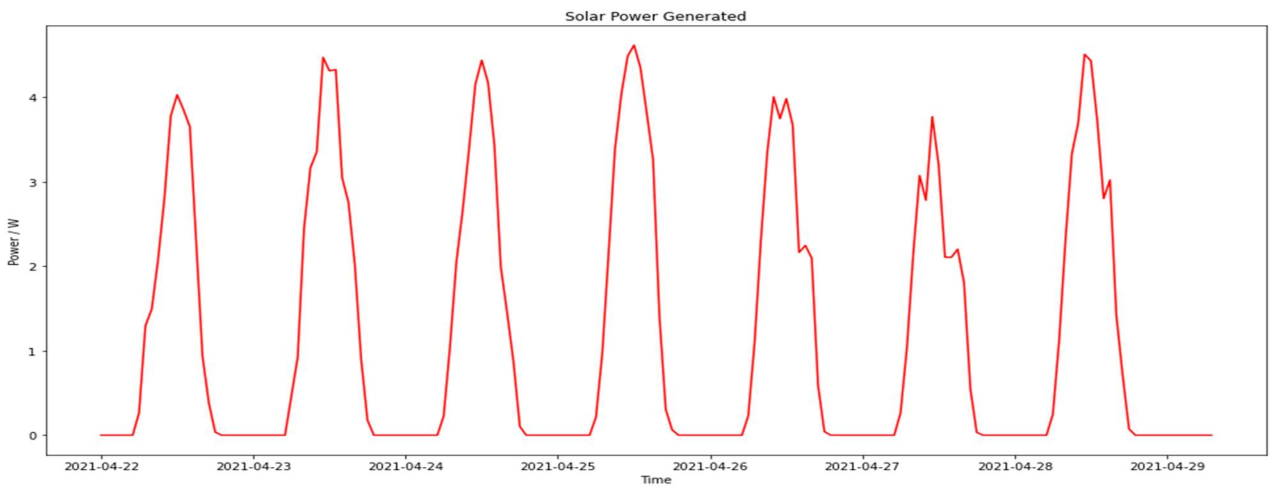


Figure 4.1: Graph of Data Collected

Using time as the independent variable and power as the dependent variable, the time series and the distribution of power values were plotted using matplotlib as illustrated in figure 4.2.

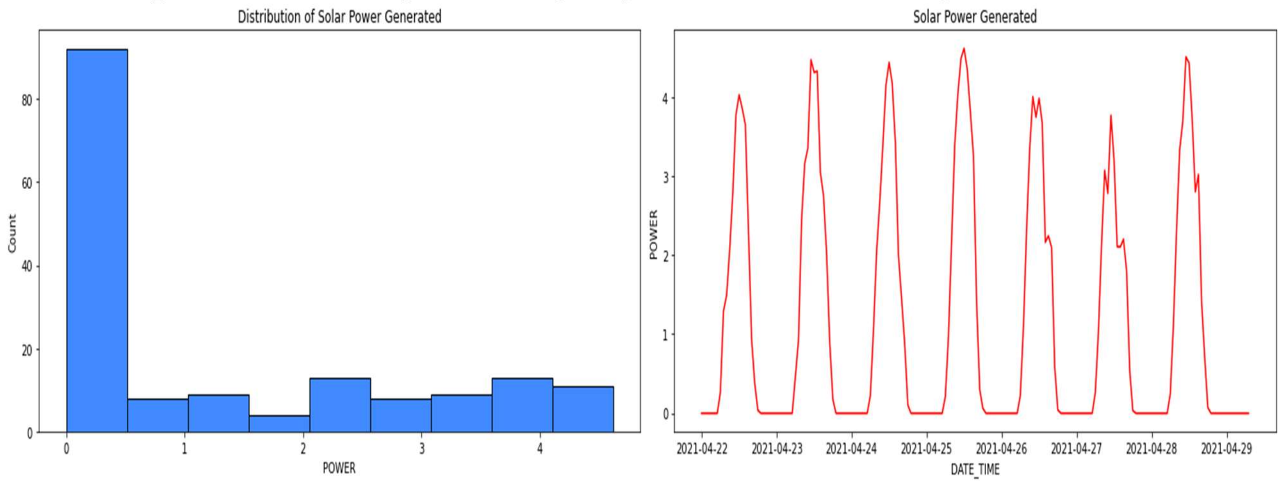


Figure 4.2: Distribution of Power

Visually, there's no significant trend in our data but it has strong seasonality as the solar panel generates power during the day and peaks at midday when the intensity of the sun is the strongest.

Decomposing the time series data into its 3 components: trend, seasonality, and residuals using ***seasonal_decompose()*** from ***statsmodels.tsa.seasonal*** yielded the following result.

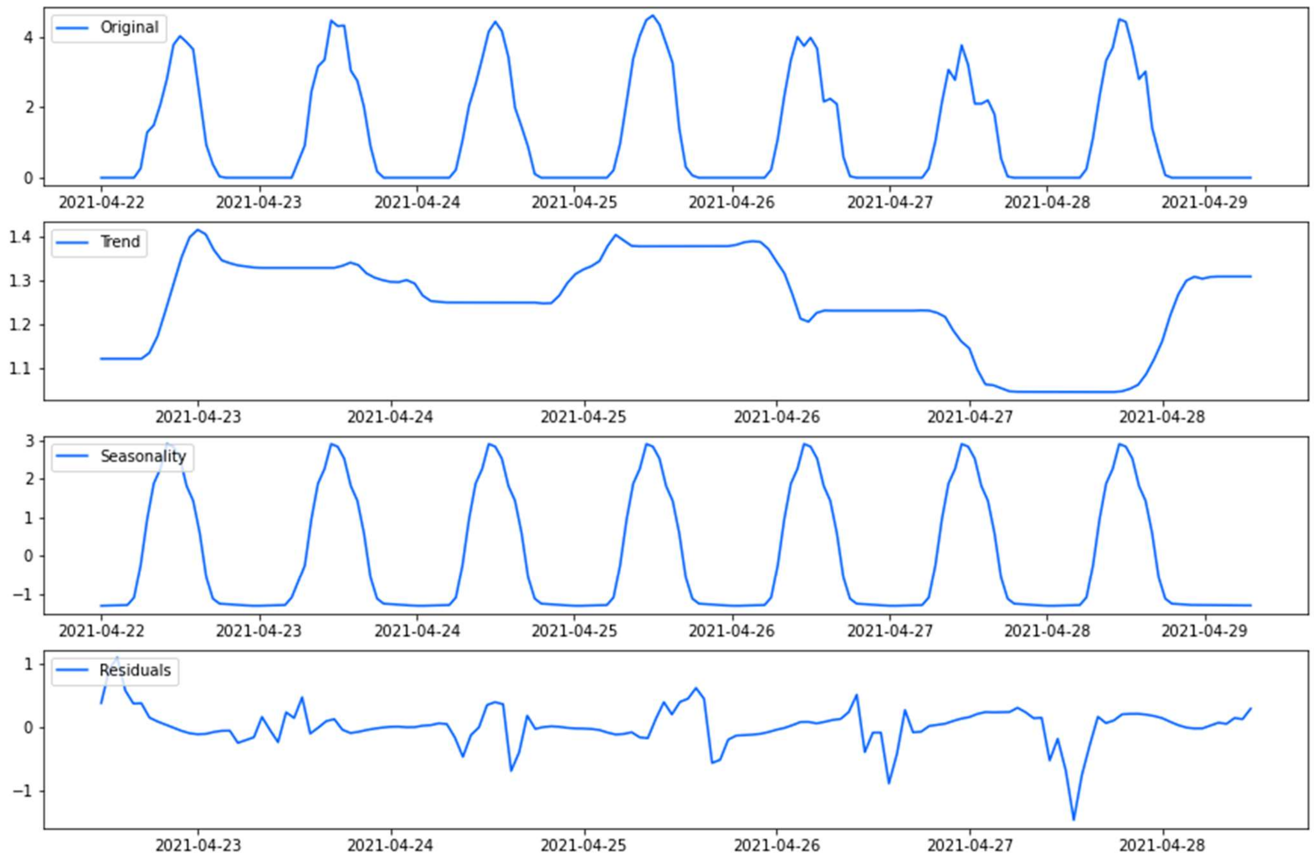


Figure 4.3: Decomposition of Data Collected

Judging from the seasonal decomposition; a period of 24 captures the seasonality.

In order to make the time series stationary, the seasonality has to be removed and this was done by seasonal differencing with a period of 24. The results of this transformation are given in figure 4.4.

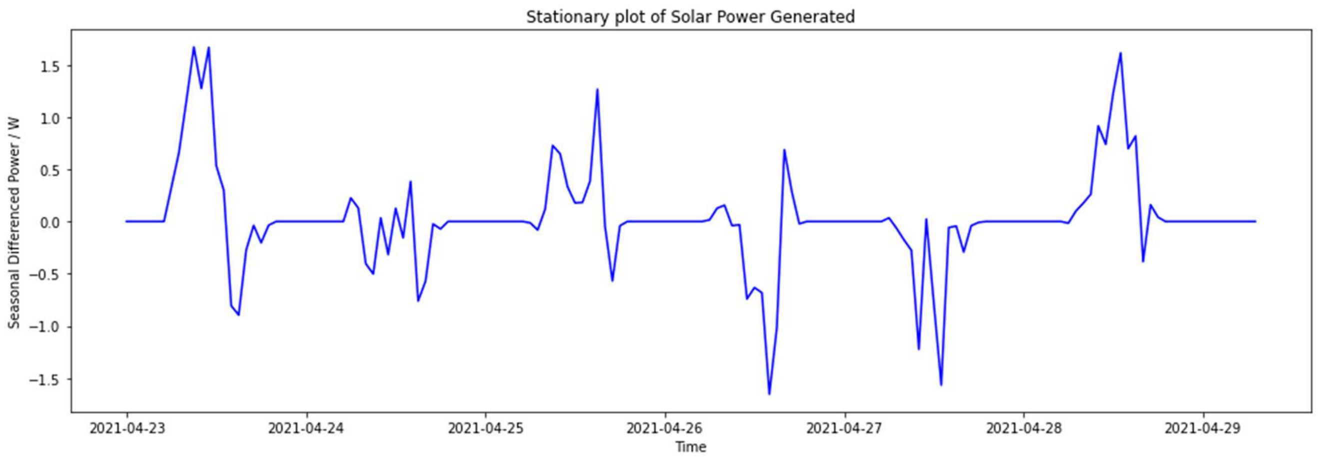
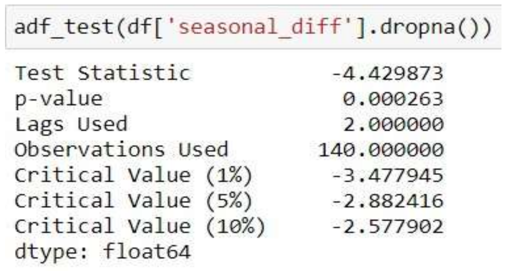


Figure 4.4: Transformed time series

The stationarity of the seasonal differenced time series was verified using the Augmented Dickey-Fuller Test. The results of the ADF test are given in Figure 4.5.



```
adf_test(df['seasonal_diff'].dropna())
```

```
Test Statistic             -4.429873
p-value                     0.000263
Lags Used                   2.000000
Observations Used         140.000000
Critical Value (1%)        -3.477945
Critical Value (5%)        -2.882416
Critical Value (10%)       -2.577902
dtype: float64
```

Figure 4.5: Augmented Dickey-Fuller test results

The more negative the Test Statistic, the more likely that the time series is stationary. The p-value tells us how confident we can be about our Test Statistic. In this case, our Test Statistic is fairly negative and our p-value is close to zero(well below our 0.05 threshold). Therefore, we can infer that our seasonal differenced time series data is stationary.

The time series was denoised using wavelet transform, the results of denoising are given in figure 4.6

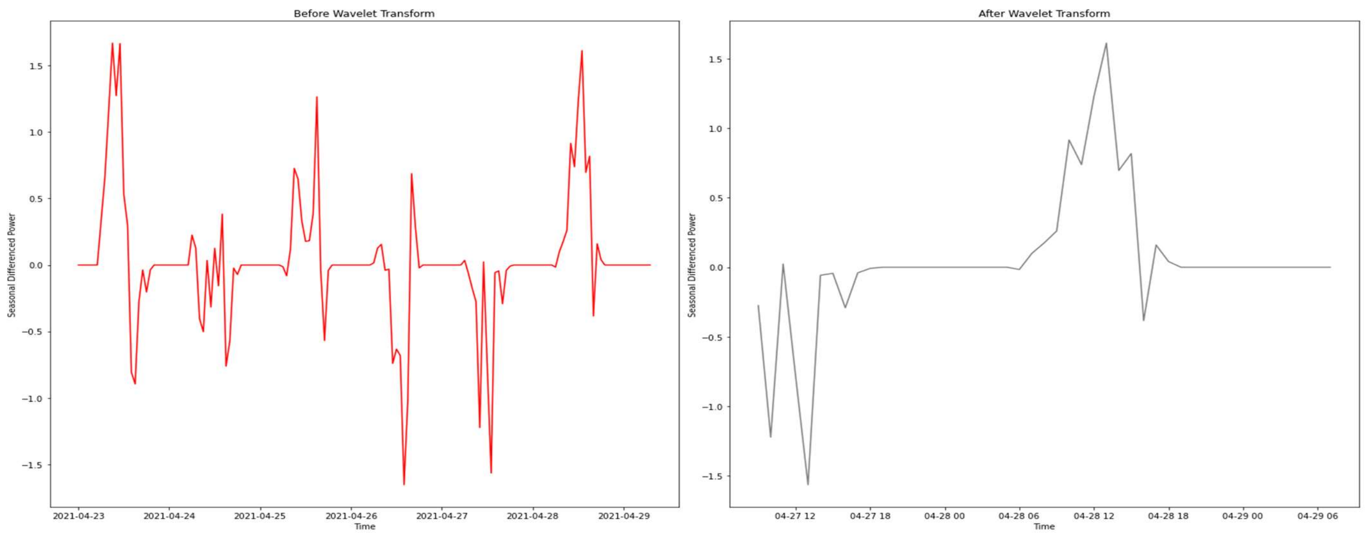


Figure 4.6: Denoised time series

4.2 Predictions

In order to train and test our model, we split our transformed and denoised data into 2 parts as illustrated in figure 4.7. Out of the last 5 days, the first 4 days will be used for training and the 5th day will be used for testing.

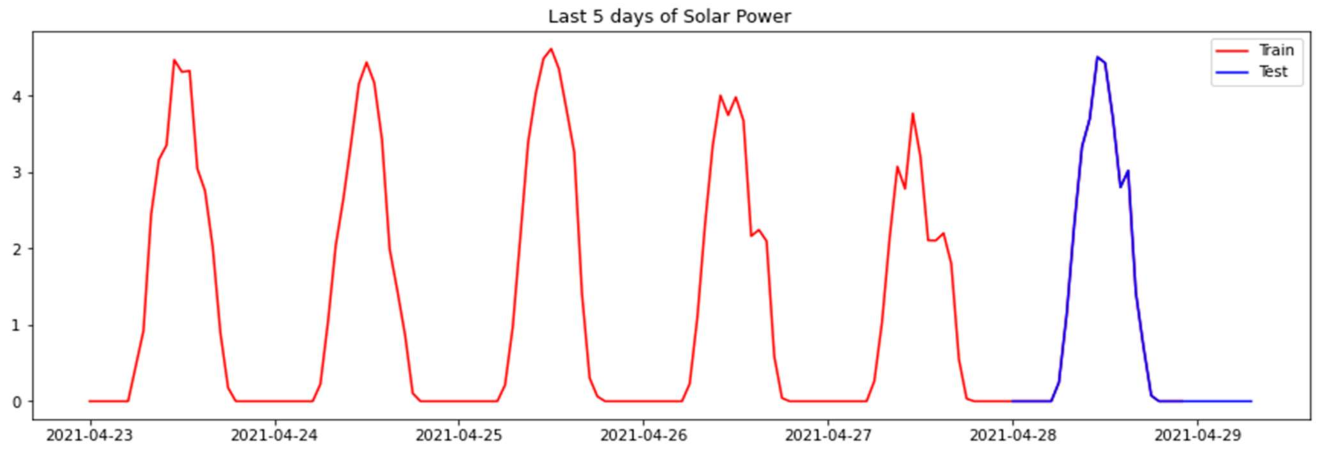


Figure 4.7: Train and test set split

The 'Train' dataset was used to plot the ACF and PACF plots given in figure 4.8.

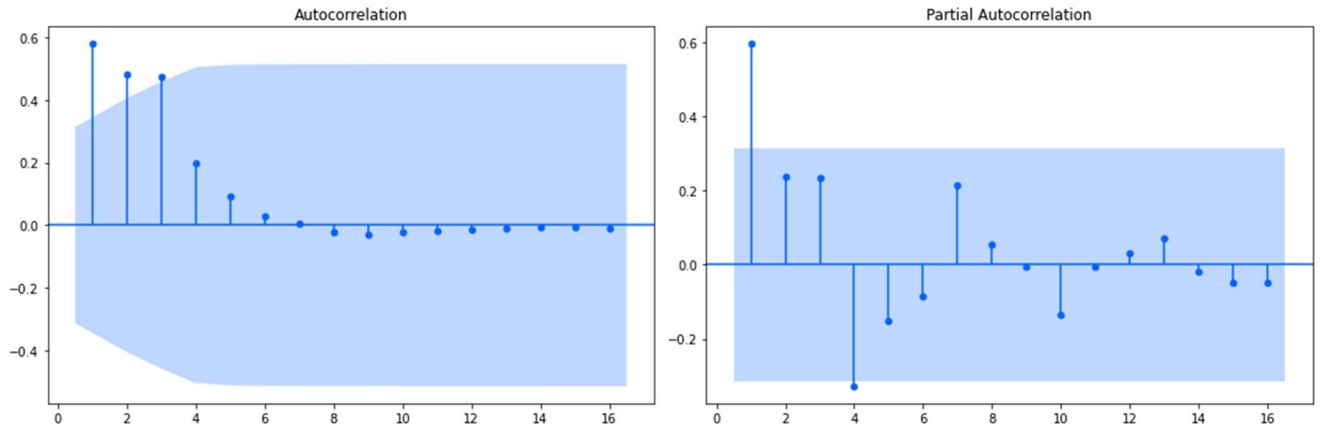
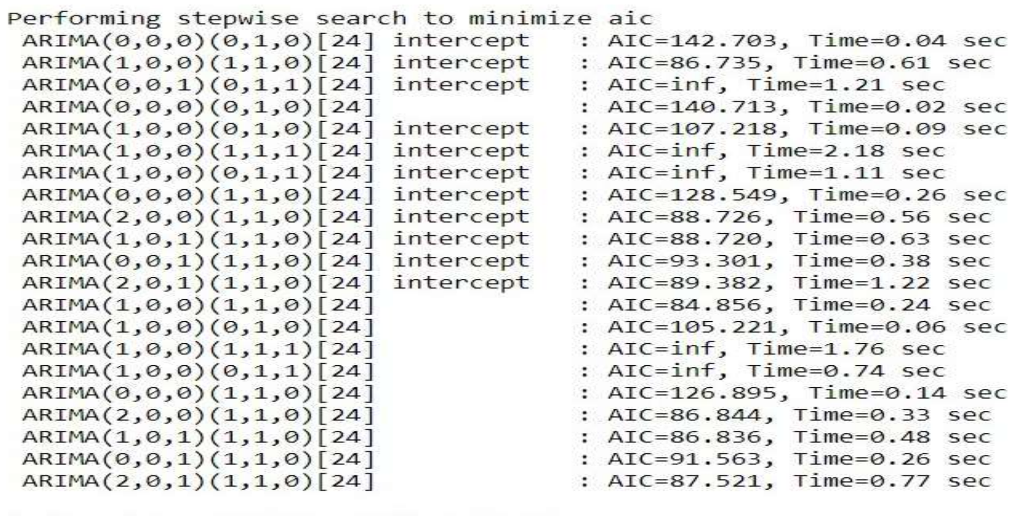


Figure 4.8: ACF and PACF plots

The SARIMA model has 7 parameters and the plots in figure 4.8 can be used to find them.
The ACF and PACF plots can be used to infer the values of q and p respectively.
We can make the following inferences on parametric ranges:
p: Range from 0 to 4
q: Range from 0 to 3
d: Range from 0 to 1
P & Q: Range from 0 to 1
We have seasonal differenced our dataset once as well, so we'll set D = 1.
m = 24
After iterating through the range of expected values of p, d, q, P, D, and Q , the model with the minimum AIC was chosen.



```
Performing stepwise search to minimize aic
 ARIMA(0,0,0)(0,1,0)[24] intercept   : AIC=142.703, Time=0.04 sec
 ARIMA(1,0,0)(1,1,0)[24] intercept   : AIC=86.735, Time=0.61 sec
 ARIMA(0,0,1)(0,1,1)[24] intercept   : AIC=inf, Time=1.21 sec
 ARIMA(0,0,0)(0,1,0)[24]             : AIC=140.713, Time=0.02 sec
 ARIMA(1,0,0)(0,1,0)[24] intercept   : AIC=107.218, Time=0.09 sec
 ARIMA(1,0,0)(1,1,1)[24] intercept   : AIC=inf, Time=2.18 sec
 ARIMA(1,0,0)(0,1,1)[24] intercept   : AIC=inf, Time=1.11 sec
 ARIMA(0,0,0)(1,1,0)[24] intercept   : AIC=128.549, Time=0.26 sec
 ARIMA(2,0,0)(1,1,0)[24] intercept   : AIC=88.726, Time=0.56 sec
 ARIMA(1,0,1)(1,1,0)[24] intercept   : AIC=88.720, Time=0.63 sec
 ARIMA(0,0,1)(1,1,0)[24] intercept   : AIC=93.301, Time=0.38 sec
 ARIMA(2,0,1)(1,1,0)[24] intercept   : AIC=89.382, Time=1.22 sec
 ARIMA(1,0,0)(1,1,0)[24]             : AIC=84.856, Time=0.24 sec
 ARIMA(1,0,0)(0,1,0)[24]             : AIC=105.221, Time=0.06 sec
 ARIMA(1,0,0)(1,1,1)[24]             : AIC=inf, Time=1.76 sec
 ARIMA(1,0,0)(0,1,1)[24]             : AIC=inf, Time=0.74 sec
 ARIMA(0,0,0)(1,1,0)[24]             : AIC=126.895, Time=0.14 sec
 ARIMA(2,0,0)(1,1,0)[24]             : AIC=86.844, Time=0.33 sec
 ARIMA(1,0,1)(1,1,0)[24]             : AIC=86.836, Time=0.48 sec
 ARIMA(0,0,1)(1,1,0)[24]             : AIC=91.563, Time=0.26 sec
 ARIMA(2,0,1)(1,1,0)[24]             : AIC=87.521, Time=0.77 sec
```

Figure 4.9: SARIMA model training

The lowest AIC score of our computation is 84.856 and the corresponding model is SARIMA(1,0,0)(1,1,0) with a period of 24; therefore, we'll use this model for forecasting.
The forecast for the test set was made using the selected model and the forecast is given in figure 4.10.

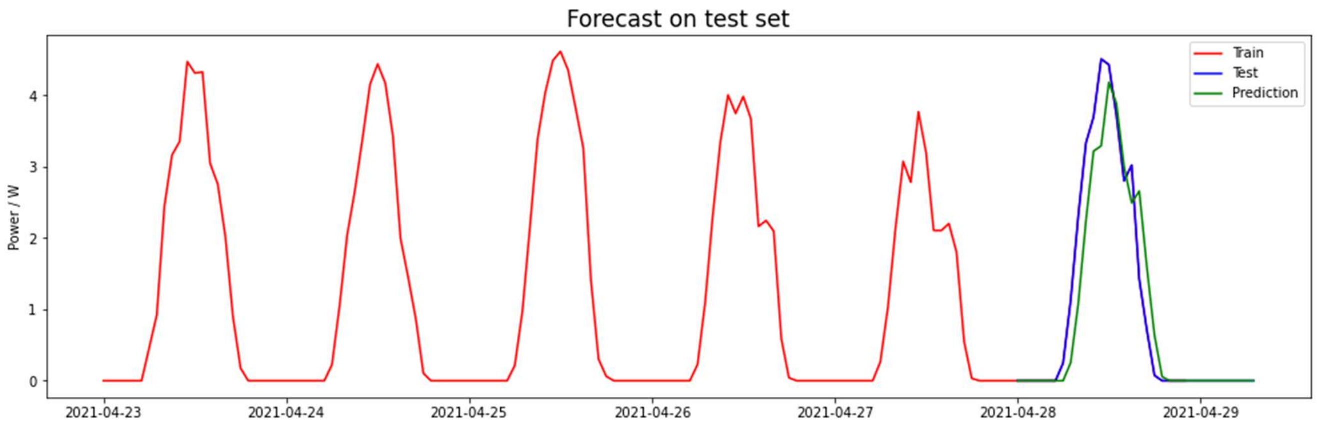


Figure 4.10: Forecasting on testing set

A closer look at the prediction is given in figure 4.11

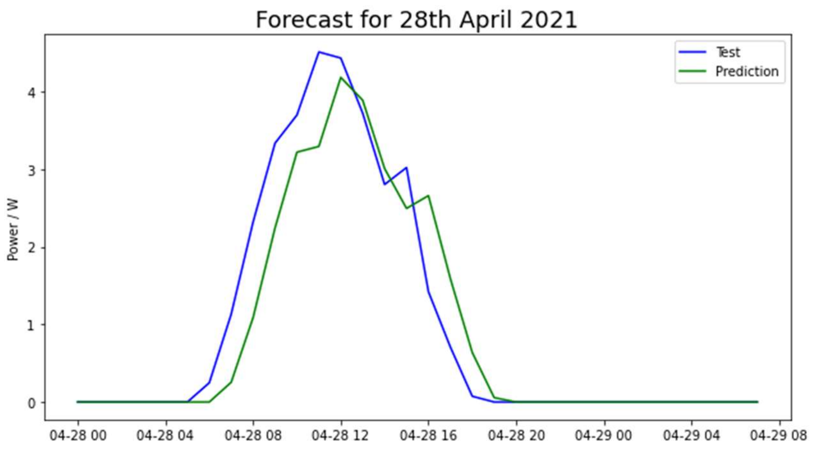


Figure 4.11: Prediction

The time taken to train the SARIMA model was 13.942 seconds and it produced a forecast with a mean squared error score of 0.3453, implying a high accuracy.

5 CONCLUSIONS & FUTURE SCOPE

5.1 Work Conclusions

Our analysis shows that the PV power generated is seasonal with no trend as the solar panel generates power during the day and peaks at midday when the intensity of solar irradiation is the strongest, implying non stationarity.
The idea behind using a Voltage divider arose out of necessity and a will to think outside the box. Due to government-imposed restrictions amidst the pandemic, sensors couldn't be procured, and we had to develop ingenious ways to measure voltage. Using the voltage divider method not only makes this monitoring system more readily accessible but also user-friendly.

The dataset had 15 minute intervals which were upscaled to 1 hour intervals. The time series data was then made stationary using seasonal differencing and tested using the Augmented Dickey-Fuller test. Additionally, we smoothed the time series data using a wavelet transform to remove high frequency noise components which do not contain any useful information. The ACF and PACF plots were used to objectively determine the range of SARIMA parameters. The last 5 days of the dataset were used to train the SARIMA model to forecast the power generated.

5.2 Future Scope of Work

Our analysis shows that the prediction is fairly accurate and coupled with a low training time, implies feasibility in onsite deployment. The forecast values can be coupled with non-renewable energy production to ensure proper grid balancing. Current and voltage sensors can be used to improve accuracy of data collected.
Furthermore, deep learning algorithms like RNN & LSTM can be used to improve accuracy with significant increases in model fit time.

REFERENCES

1. Mostafa Majidpour, Hamidreza Nazaripouya, Peter Chu, Hemanshu R. Pota & Rajit Gadh, "Fast Univariate Time Series Prediction of Solar Power for Real-Time Control of Energy Storage System.", MDPI, UCLA, 107-120. 10.3390/forecast1010008, September 2018.
2. Neungsoo Park & Hyung Keun Ahn, "Multi-Layer RNN-based Short-term Photovoltaic Power Forecasting using IoT Dataset.", AEIT International Annual Conference, Konkuk University, 1-5 10.23919/AEIT.2019.8893348, September 2019.
3. Aboubakr Al Hammoumi, Saad Motahhir, Abdelilah Chalh, Abdelaziz El Ghzizal & Aziz Derouich, "Low-Cost Virtual Instrumentation of PV panel characteristics using Excel and Arduino in comparison with traditional instrumentation", Renewables: 5 Vol. (Springer Open), SMBA University, https://doi.org/10.1186/s40807-018-0049-0 [Online], March 2018.
4. Murat C. Sorkun, Özlem D. Incel & Christophe Paoli, "Time Series Forecasting on Multivariate Radiation using Deep Learning (LSTM)", Turkish Journal of Electrical Engineering and Computer Sciences, Dutch Institute of Fundamental Energy Research, 28. 211-223. 10.3906/elk-1907-218, January 2020.
5. Solar panel, https://www.certainteed.com/solar/solar-101-abcs-solar-power/
6. Arduino Uno, https://en.wikipedia.org/wiki/Arduino_Uno
7. Analog to Digital Converter, https://learn.sparkfun.com/tutorials/analog-to-digital-conversion/all
8. Measuring DC Voltage using Arduino, https://startingelectronics.org/articles/arduino/measuring-voltage-with-arduino/
9. Voltage divider, https://dronebotworkshop.com/dc-volt-current/
10. PLX-DAQ, https://www.parallax.com/package/plx-daq/
11. Statsmodels, https://www.statsmodels.org/stable/index.html
12. Sharif Atique, Subrina Noureen, Vishwajit Roy & Stephen Bayne, "Forecasting of total daily solar energy generation using ARIMA: A case study", IEEE 9th Annual Computing and Comunication Workshop and Conference (CCWC), Las Vegas, NV, USA, 10.1109/CCWC.2019.8666481, March 2019.
13. Code for Measuring Voltage and other variables using Arduino and PLX-DAQ Excel, https://github.com/Mukunds0507333/Time-Series_PV-System-Output/blob/main/IoT/Variable_measurement.ino
14. Wes McKinney, "pandas: A Foundational Python Library for Data Analysis and Statistics.", Python High Performance Science Computer, 2011.
15. Bernadette M. Randles, Irene V. Pasquetto, Milena S. Golshan and Christine L. Borgman, "Using the Jupyter Notebook as a Tool for Open Science: An Empirical Study", ACM/IEEE Joint Conference on Digital Libraries (JCDL), UCLA, June 2017.
16. IoT Dataset, https://github.com/Mukunds0507333/Time-Series_PV-System-Output/tree/main/files
17. Code for Forecasting Photovoltaic System Power Generation, https://github.com/Mukunds0507333/Time-Series_PV-System-Output/blob/main/Forecasting_PV_Power_Generation.ipynb

PROJECT DETAILS

Student Details			
Student Name	**Mukund Sureshkumar**		
Register Number	189202085	Section / Roll No	C 2
Email Address	mukunds0507333@gmail.com	Phone No (M)	+91 8122065644
Student Name	Mishkat Neyazi		
Register Number	189402164	Section / Roll No	C 3
Email Address	mishkat.neyazi@gmail.com	Phone No (M)	+91 9319126512
Project Details			
Project Title	**Forecasting Solar Power Generation**		
Project Duration	3 Months and 3 Weeks	Date of reporting	01/05/2021
Organization Details			
Organization Name	**Manipal University Jaipur**		
Full postal address with pin code	Jaipur-Ajmer Express Highway, Dehmi Kalan, Near GVK Toll Plaza, Jaipur, Rajasthan 303007		
Website address	https://jaipur.manipal.edu/		
Supervisor Details			
Supervisor Name	**Dr Himanshu Chaudhary**		
Designation	Associate Professor (Senior Scale), Department of ECE		
Full contact address with pin code			
Email address	himanshu.chaudhary@jaipur.manipal.edu	Phone No (M)	
Internal Guide Details			
Faculty Name	**Dr Himanshu Chaudhary**		
Full contact address with pin code			
Email address	himanshu.chaudhary@jaipur.manipal.edu	Phone No (M)	