TUS

THE TECHNOLOGICAL UNIVERSITY OF THE SHANNON: MIDLANDS MIDWEST

Predictive Maintenance of Solar Panels Using Digital Twins – Statistical vs Deep learning Time Series Forecasting

Parimal Ganesh Sawant

This research project is submitted in fulfilment of the degree of Masters of Science in Data Analytics at the Technological University of the Shannon: Midlands Midwest, Athlone

Faculty of Business and Hospitality

Department of Accounting and Business Computing

Prof. Terri Hoare

24th August 2025

DECLARATION

I hereby certify that the material, which is submitted in this thesis towards the award of MSc. in Data Analytics, is entirely my own work and has not been submitted for any academic assessment other than part fulfilment of the above-named award.

Future students may use the material contained in this thesis provided that the source is acknowledged in full.

Signed: Parimal Ganesh Sawant Date: 24/08/2025

ACKNOWLEDGEMENTS

I would like to thank my supervisor Terri Hoare for her guidance and sharing of expertise throughout this research project.

I would also like to show appreciation to all of the lecturers at TUS, who helped me gain the knowledge to be able to complete this dissertation.

Finally, I wish to express my heartfelt gratitude to my friends and family for their unwavering support throughout the course of this research project. Their belief in my abilities has been a constant source of motivation.

A special thanks to my friend, Rushikesh Badgujar, for his invaluable collaborative efforts in the work related to digital twins. His insights and teamwork have greatly contributed to the success of this project.

The journey of self-improvement is ongoing, and I am truly grateful to everyone who has been a part of this process.

LIST OF ABBREVIATIONS

ARIMA - Autoregressive Integrated Moving Average

SARIMA - Seasonal Autoregressive Integrated Moving Average

LSTM - Long Short-Term Memory

PATCHTST - Patch Time Series Transformer

MAE - Mean Absolute Error

MSE - Mean Squared Error

MAPE - Mean Absolute Percentage Error

MASE - Mean Absolute Scaled Error

RNN - Recurrent Neural Network

PV – Photovoltaic

AWS - Amazon Web Services

LIST OF FIGURES

Figure 3.1 The CRISP-DM process

Figure 3.2 Line Chart for Power Generated per Kilowatt in Ireland

Figure 3.3: Summary statistics of dataset

Figure 3.4: Boxplot for outlier detection

Figure 3.5: Correlation Matrix of features

Figure 3.6: Geographic Plotting of Solar Panel Output European Countries

Figure 3.7: Flow Chart for Data Preparation

Figure 3.8: Heatmap for Null Value Detection

Figure 3.9: Train-Test Split

Figure 3.10: LSTM RNN architecture

Figure 3.11: PatchTST Architecture Diagram

Figure 3.12: Architectural Overview

Figure 4.1: ADF Test Result 1

Figure 4.2: ADF Test Result 2

Figure 4.3: Autocorrelation for ARIMA

Figure 4.4: Model 1 ARIMA

Figure 4.5: Model 2 ARIMA

Figure 4.6: Model 3 ARIMA

Figure 4.7: ACF Test SARIMA Model

Figure 4.8: SARIMA Model with 120 days seasonality

Figure 4.9: SARIMA Model with 30 days seasonality

Figure 4.10: Model 1: Baseline LSTM yearly predictions

Figure 4.12: Model 2: Enhanced LSTM yearly predictions

Figure 4.13: Model 2: Enhanced LSTM January month hourly predictions

Figure 4.14: PatchTST daily predictions for 2024

Figure 4.15: PatchTST hourly predictions for January

LIST OF TABLES

Table 3.1: Data Types and values

Table 3.2: Expected behaviour of features

Table 4.1: ARIMA Model Evaluation

Table 4.2: SARIMA Model Evaluation

Table 4.3: LSTM Model Evaluation

Table 4.4: PatchTST Model Evaluation

ABSTRACT

The research explores predictive maintenance approaches for solar photovoltaic (PV) systems through the combination of digital twin technology and advanced forecasting techniques. The study uses six years of hourly data from Renewable Ninjas to compare the use of traditional statistical approaches such as ARIMA and SARIMA with deep learning (LSTM) and transformer-based architecture (PatchTST). The results shows that ARIMA and SARIMA were unable to operate effectively hourly dataset, yielding baseline performance that failed to capture the non-linear dynamics and high-frequency variability when applied to daily resampled data. In comparison, LSTM enhances accuracy significantly by capturing sequential relationships, and PatchTST provides computationally effective long-term performance but underperforming during extreme weather conditions. By integrating the model with digital twin architecture, the research demonstrates improved operational reliability, reduced downtime and fault detection. This work highlights the potential of combining digital twin technology with modern forecasting approaches for scalable, low cost and environmentally friendly solar plant maintenance.

Keywords: Predictive Maintenance, Solar Photovoltaic (PV), Digital Twin, ARIMA, SARIMA, LSTM, PatchTST, Time Series Forecasting, Renewable Energy

TABLE OF CONTENT

DECLARATION ..... **i**

Acknowledgements ..... **iii**

List Of Abbreviations ..... **iv**

List Of Figures ..... **v**

List Of Tables ..... **vi**

Abstract ..... **vii**

Chapter 1 Introduction **1**
1.1 Business Problem 1
1.2 Research Problem 1
1.3 Aim 2
1.4 Objective 2
1.5 Hypothesis 3
1.6 Scope 3
1.7 Limitations 4

Chapter 2 Literature Review **5**
2.1 Introduction 5
2.2 Solar Photovoltaic Systems: Overview and Basic Principles 6
2.3 Failure Patterns and Maintenance Challenges 7
2.4 Power Grid Operation and Imbalance Issues 8
2.5 Digital Twin Technology 9
2.6 Forecasting Approaches: Traditional to Advanced Methods 9
2.7 Integrating Forecasting into Grid and Maintenance Planning 10
2.8 Research Gap and Future Opportunities 11

Chapter 3 Research Methodology **14**
3.1 Introduction 14
3.2 Business Understanding 15
3.3 Data Understanding 16
3.4 Data Preparation 23
3.5 Modelling 27
3.6 Evaluation 31
3.7 Deployment 34

Chapter 4 Results And Analysis 36
4.1 ARIMA Model 36
4.2 SARIMA Model 40
4.3 RNN Model 44
4.4 PatchTST Model 48
4.4 Comparison Of Models 51

Chapter 5. Conclusion And Future Work 53
5.1 Conclusion 53
5.2 Future Work 54

References 55

CHAPTER 1 INTRODUCTION

1.1 BUSINESS PROBLEM

The global trend towards renewable energy has highlighted the need for more reliable, efficient, and cost-effective solutions for maintaining solar photovoltaic (PV) systems. Solar power plays an important role in decarbonisation strategies and energy transition globally, yet solar PV systems face a significant operational challenge in maintaining their efficiency and performance over the long term. Traditional solar PV system practices are based on scheduled or regular inspections, which often result in downtime, inefficiency, and unrecognised system degradation. The main business issue resolved by this research is the failure of traditional maintenance practices to reduce downtime and detect the problem leading to performance degradation.

As solar PV plants increase in size and complexity, it becomes difficult to manage large sets of solar panels and components spread over remote geographical locations. Maintenance becomes extremely costly, especially when the plant administrators are dependent on manual inspections or schedule-based maintenance that fail to incorporate real-time system performance or external environmental factors. Traditional maintenance that relies on periodic inspection and fixing on the basis of estimated or reported defects, lacks the flexibility or accuracy to guarantee maximum performance in changing operating conditions.

The inefficiency of traditional maintenance results in increased operational cost, reduced energy output, and increased risk of equipment breakdown, which affects the profitability and long-term sustainability of solar PV systems. The failure to forecast breakdowns in real-time also disturbs the smooth power grid operations, as unexpected outages or energy production dips from PV systems create an imbalance in energy supply.

In order to solve this business issue, there is a need for an increased level of proactive data-driven maintenance that applies time-series forecasting, monitors real-time data, and uses digital twin technology. This study introduces a solution that aligns these components together to streamline the process, lower cost, enhance PV solar reliability, and lifespan. Through the prediction of maintenance requirements using data models, the study seeks to improve operational effectiveness and achieve substantial cost reductions in the renewable energy industry.

1.2 RESEARCH PROBLEM

The research problem addressed by the study lies in existing limitations in time-series forecasting approaches of solar photovoltaic (PV) systems, more specifically, the difficulty in precisely forecasting future problems and system degradations. Solar PV systems need to be always kept maintained for maximum energy output, traditional statistical approaches such as ARIMA and SARIMA, which provides a report notice are not adequate for detecting and addressing potential failures beforehand. This method is ineffective in capturing the non-linear pattern of the solar PV system performance as it is moderated by multiple factors, including the weather conditions, environmental factors, and system-specific parameters such as panel age and component wear.

Current methods of forecasting, i.e., ARIMA (Autoregressive Integrated Moving Average) and SARIMA (Seasonal ARIMA), have been applied in some industries for time-series forecasting.

These models are unable to capture the complexity of renewable energy systems, especially solar PV systems, whose environmental conditions and energy output have complex, non-linear relationships. These models are also incapable of dealing with high-frequency data (e.g., hourly observations) and have limitations in capturing seasonality and variability associated with solar energy production. Although they are able to record general trends, they cannot deliver the accuracy for real-time decision-making, particularly for the detection of early indications of failure or degradation of performance.

The research aims to fill the gap by creating a time-series forecasting model that utilises digital twin technology in combination with traditional forecasting models, deep learning models, and transformer models, to offer a more precise and reliable way to forecast the health of solar PV systems. In response to addressing the limitations of traditional models, this study is intended to provide a solution that can identify problems in the early stage, enhance system reliability, and minimise operational expenses while maintaining optimal performance of solar PV systems.

Research Question:

How do traditional time-series forecasting models (ARIMA, SARIMA), deep learning models (LSTM), and transformer-based models (PatchTST) compare in terms of accuracy and performance when applied to solar photovoltaic (PV) system data for predictive maintenance?

1.3 AIM

The goal of this study is to create and test a time-series forecasting model for solar photovoltaic (PV) systems by comparing time-series models, deep learning models (LSTM), and transformer models (PatchTST) based on accuracy, to improve the reliability of the system, minimise downtime, and maximise operational efficiency.

1.4 OBJECTIVE

The objective of this research is to implement a time-series forecasting model for solar photovoltaic (PV) systems that combines traditional forecasting models such as ARIMA and SARIMA, deep learning models (LSTM), and a transformer-based model (PatchTST). This framework seeks to enhance the reliability, efficiency, cost savings of solar PV systems through failure prediction and degradation anticipation before they occur. The objectives of this research are as follows:

- Analyse Traditional Models: Test the effectiveness of traditional models like ARIMA and SARIMA in solar PV performance forecasting, especially whether they can model non-linear relationships and high-frequency data. This provides a baseline to compare the efficiency of more modern models.
- Implement Deep Learning Model (LSTM): Explore the use of Long Short-Term Memory (LSTM) networks in forecasting solar PV output, highlighting their ability to capture intricate, non-linear relationships and identify long-term trends in sequential data. The objective is to show how deep learning models can surpass the drawbacks of traditional approaches.
- Integrate Transformer-Based Models (PatchTST): Investigate the capabilities of Patch Time Series Transformer (PatchTST), an advanced transformer model, in learning both short and long-term dependencies in solar PV data. The objective is to determine the

strengths of transformers in forecasting output and how effective they are in handling nonlinear, high-dimensional datasets.

- Compare Model Performance and Accuracy: Do a comparative study of the accuracy, efficiency in computation, and scalability of ARIMA, SARIMA, LSTM, and PatchTST models on solar PV data. Metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and Mean Absolute Percentage Error (MAPE) will be utilised to compare each model's performance.
- Evaluate Feasibility for Real-World Deployment: Determine the practical feasibility of using a time-series forecasting model in actual solar PV plants based on issues such as integration with current systems, computational needs, and scalability.

1.5 HYPOTHESIS

NULL HYPOTHESIS (H_0)

There is no difference in the forecasting accuracy of traditional forecasting models such as ARIMA and SARIMA and advanced forecasting models LSTM and PatchTST when used on solar photovoltaic data. Both models are equally accurate, reliable and efficient in making predictions regarding system performance and breakdowns.

ALTERNATIVE HYPOTHESIS (H_1)

Advanced forecasting models such as Long Short-Term Memory (LSTM) and Patch Time Series Transformer (PatchTST), outperformed traditional forecasting models such as ARIMA and SARIMA in terms of forecasting the performance and failures of solar photovoltaic systems. The forecast is more reliable, accurate, and efficient resulting in optimal maintenance schedule and lower operational cost.

1.6 SCOPE

The scope of this research is on the development and evaluation of a forecasting output for a solar photovoltaic (PV) system using time-series forecasting models. The research will investigate and compare the accuracy of traditional statistical approaches like ARIMA and SARIMA, deep learning models (LSTM), and transformer-based models (PatchTST) in predicting solar PV system performance as well as possible failures or degradation.

The research will be undertaken using a six-year hourly solar PV performance dataset from Renewable Ninjas, which comprises environmental data like irradiance, temperature, and humidity in addition to solar panel output. The dataset will serve as a foundation on which different forecasting models will be developed and tested. The dataset will cover hourly, daily, and seasonal variation in solar power generation, and it will provide a broad overview of the factors affecting solar PV performance.

While the main geographical focus of this research is solar PV systems functioning in Ireland, the framework can be used and modified to apply to solar PV systems functioning in other places with comparable environmental conditions. The study will determine how each of the models captures the non-linearity and seasonal attributes of solar energy generation, particularly the ability of the models to deal with the complexity and variability associated with renewable energy systems.

In summary, the focus of this research is on the development of time-series forecasting for solar energy systems towards optimal, efficient, and sustainable renewable resources.

1.7 LIMITATIONS

Although the work seeks to create a time-series forecasting platform for solar photovoltaic (PV) systems, the following limitations should be considered.

- Model complexity: Deep learning techniques such as Long Short-Term Memory (LSTM) and Patch Time Series Transformer (PatchTST) are computationally intensive for training, particularly with high-frequency data (hourly). Such models also tend to suffer from overfitting in case of improper tuning, and thus, real-world performance might suffer with less accurate predictions.
- Data preprocessing and Quality: Data cleansing and preprocessing might have been done, but there are some outliers, missing values, or errors in the data that might introduce some amount of noise in terms of accuracy. The structure of the dataset is good, yet it is a problem to handle noisy or incomplete data for the models.
- Generalisability: Although the models shall be tested with data from one region (Ireland), the scope of the research does not cover various solar PV technologies, configurations, or sites with various environmental conditions. The performance and applicability of the models in other environments might need additional testing.
- Economic and Practical Feasibility: The economic feasibility of undertaking the proposed time-series forecasting architecture, with the incorporation of advanced forecasting models and digital twin technology, can be a constraint for small or medium-scale solar PV operations. The study will concentrate on the theoretical developments and comparisons between the models, with minimal focus on the extensive cost-benefit analysis for execution in different solar plant designs.
- Model Interpretability: Deep learning and transformer-based models tend to be "black-box" models, implying that they could be non-transparent in making predictions. This tends to restrict the interpretability of the model's decision-making process, which could be problematic for situations where explainability is important for plant operators or stakeholders.

CHAPTER 2 LITERATURE REVIEW

2.1 INTRODUCTION

This literature review explores how researchers apply time-series forecasting methodologies that are integrated with digital twin technology in solar photovoltaic (PV) systems. Solar PV technology emerges as a critical component toward achieving decarbonisation targets as researched by (Kenu E. Sarah, 2020), given the global urgency to transition toward sustainable energy sources. This review comprehensively surveys literature so as to examine the principles, current practices, challenges, and innovations within maintenance for solar PV plants. It does critically engage with key parts of the literature. It also presents perceptive analyses and notes points of accord and conflict.

This is elaborated in Section 2.1 Solar Photovoltaic Systems: (Guerra *et al.*, 2018) highlights overview and Basic Principles, which discusses the basic operational principles of solar photovoltaic systems, plus it stresses the photovoltaic effect wherein semiconductor materials like silicon produce electricity when they are exposed to sunlight. About solar panel efficiency, (Markvart and Castaner, 20062003) suggested key seminal texts provide foundational knowledge. They do specifically detail all of the distinctions between monocrystalline and polycrystalline silicon cells. Study by (Huld *et al.*, 2010) determines panel performance is influenced in so many ways, as is shown collectively in these works, including irradiance, temperature, and also panel orientation.

Following this, the review gives a careful study of trends within problems and difficulties influencing the upkeep of solar PV systems. Seminal reports as well as recent studies by (Aghaei *et al.*, 2022) and (Marc Köntges *et al.*, 2014) delineate issues that include Potential Induced Degradation (PID), microcracks, hotspots, and soiling. These are examined in Section 2.2 Failure Patterns and Maintenance Challenges, which insights into advanced maintenance strategies are critically needed in order to prevent impactful yet gradual degradation of solar systems. Furthermore, the challenges are critically examined regarding intrinsic issues in maintaining large utility-scale PV installations. As the examination by (Marc Köntges *et al.*, 2014), manual inspections and periodic maintenance practices have important limitations.

A key aspect of this review is that it focuses on transitioning from customary reactive maintenance to predictive maintenance. Recent literature, like the work by (Said *et al.*, 2024) and (Aghaei *et al.*, 2022), shows how data-driven approaches use time-series analysis, anomaly detection, as well as traditional statistical approaches, so failures are anticipated proactively. Section 2.2.3 Need for Predictive Maintenance explores that these approaches are identified as being important in reduction of operational costs. They also improve reliability in order to optimize system performance, particularly within challenging or remote environmental conditions.

Furthermore, this review gives a critical evaluation of power grids' operational complexities, namely imbalances from renewable energy sources, as (Soto *et al.*, 2022) and (Ulbig, Borsche, and Andersson, 2014), discussed. Section 2.3 Power Grid Operation and Imbalance Issues evaluates these challenges. Inverter-based energy sources create low rotational inertia along with variable solar generation, posing research challenges. Research by (Mirzapour, Rui and Sahraei-Ardakani, 2024) suggests these challenges do require advanced mitigation strategies

such as energy storage systems, as well as demand response programs plus improved transmission technologies.

An integral part of the literature review explores digital twin technology, extensively detailed by (Jiang *et al.*, 2021), (Tao *et al.*, 2019), and (Fuller *et al.*, 2020), throughout a relatively novel concept. Section 2.4 Digital Twin Technology discusses how digital twins can be critically evaluated for all of their potential to then provide virtual real-time replicas for physical systems. Due to this, time-series forecasting capabilities are improved to a large degree. However, the review identifies a clear research gap in frameworks for time-series forecasting models integrating digital twins within solar PV.

Then, Section 2.5 Forecasting Approaches: Traditional to Advanced Methods compares traditional statistical models such as ARIMA and SARIMA with advanced techniques such as Recurrent Neural Networks (RNN) and the newest technique, Patch Time Series Transformer (PatchTST). Research by (Nie *et al.*, 2022) discussed Forecasting: Principles and Practices in terms of their literature on forecasting accuracy and applicability within renewable energy resources.

Finally, Section 2.6 Forecasting in Grid and Maintenance Planning highlights the comprehensive use of time-series forecasting in grid operations and maintenance planning, explored by (Rokhforoz *et al.*, 2021). The final part of the review critically addresses issues related to model accuracy, data quality, and computational complexity, and advises for further research on a forecasting framework using real-time data.

2.2 SOLAR PHOTOVOLTAIC SYSTEMS: OVERVIEW AND BASIC PRINCIPLES

Research by (Kenu E. Sarah, 2020) highlights Solar Photovoltaic (PV) systems are one of the most popular renewable energy sources in the world due to their ability to directly convert sunlight into electricity. Solar PV systems are seen as a key technology in the global shift to a decarbonised and sustainable energy sector. A solar PV system consists of solar panels, inverters, and a balance of system components (BOS), all working in tandem to generate, convert, and distribute electricity.

- **Working Principle of Solar Photovoltaic Systems**

  The solar cell, which operates based on the photovoltaic effect, is the primary component of a solar PV system. An electric current is produced when sunlight strikes the semiconductor material (typically silicon), exciting electrons. Research by (Guerra *et al.*, 2018) proves current is then captured by a metal contact on the surface of the cell. The energy conversion efficiency of the solar cells depends on a variety of parameters, such as the material of the cell and its design. (Augustin McEvoy and Castaner, 2012) says the majority of today's commercial cells are primarily made from monocrystalline and polycrystalline silicon. Polycrystalline cells are less efficient but more affordable than monocrystalline cells, which usually have higher efficiency (~18 – 22 %).

- **Solar Energy Conversion Efficiency**

  Previous research by (Huld *et al.*, 2010) proves that solar panels consist of solar cells that are connected in series or parallel to boost the output of voltage and current. Solar cells are

made from silicon-based semiconductors. Solar irradiance and the angle of installation are the two factors that affect the panel efficiency. Solar energy conversion efficiency is one of the important metrics for evaluating a solar panel's performance. Polycrystalline panels often give efficiency between 15% and 18%, monocrystalline panels have the highest efficiency up to 22%. (Augustin McEvoy and Castaner, 2012) highlights number of factors, like temperature, irradiance, and panel orientation, can affect the overall efficiency. For example, high temperatures can reduce the efficiency of solar panels, as silicon-based materials have lower conductivity at high temperatures. Soiling (dirt building up on the panel's surface) can also reduce performance by obstructing sunlight and decreasing energy consumption.

2.3 FAILURE PATTERNS AND MAINTENANCE CHALLENGES

Solar photovoltaic (PV) systems are generally robust, but they are also prone to different failure patterns and degradation over time, much like any other technical system. To ensure longevity and efficiency, particularly in large-scale commercial or utility applications, it is essential to understand these failure patterns and related maintenance problems.

- **Common Failure Issues**

The (Marc Köntges *et al.*, 2014) report identified several key failure factors that cause solar panel efficiency to decline. Delamination, junction box faults, cell cracks, Potential Induced Degradation (PID), and bypass diode failure are some of the common failure patterns. Among these, PID stands out since it can reduce panel efficiency by 30% in high voltage, high temperature, and humid environments. Since these errors typically occur gradually, it can be challenging to identify and address them before they cause major loss in performance. The report by (Augustin McEvoy and Castaner, 2012) suggests performance degradation can also be caused by other factors such as hot spots, microcracks, and soiling.

Recent research by (Said *et al.*, 2024) suggests hotspots are frequently caused by shading or misaligned solar cells, leading to thermal stress and creating localised heating. Microcracks may not be visible to the human eye, especially in high-stress situations. Soiling, or the buildup of dust, dirt, or other particles on the panel surface, lowers solar panel efficiency by 5% – 10%, especially in dry or arid areas. These issues, if left unchecked, might result in more serious breakdowns and low power production.

- **Challenges in Maintenance**

Recent research by (Marc Köntges *et al.*, 2014) faced numerous challenges arise with the maintenance of PV systems, especially in large utility-scale facilities. A utility-scale facility has thousands or even millions of panels, making manual inspection ineffective and expensive. Automated diagnosis and monitoring systems are essential for the detection of failure in large systems.

The study by (Aghaei *et al.*, 2022) proved that the frequency of maintenance is another major challenge. Traditional periodic maintenance could miss catching sudden failures like PID or microcracks before they cause significant damage. This problem is even worse in remote areas, where regular maintenance checks and access to the plant for maintenance can be difficult and costly. Environmental conditions in which the PV systems are operated

also significantly affect the degradation process, and temperature, humidity, and solar irradiance are some of the factors contributing to wear and tear in different regions.

- **Need for time-series forecasting**

Study by (Said *et al.*, 2024) predicted the necessity of shift from reactive maintenance (fixing failures) to time-series forecasting (anticipating and preventing failures is crucial to rendering PV systems more dependable and affordable. Using real-time data and past performance trends, time-series forecasting uses data driven approaches to forecast when certain components will fail. This can be handled by using time-series analysis, anomaly detection, and advanced machine learning algorithms to identify abnormal performance trends and predict potential failures before they occur.

Sensor data built into the panels and other components of the system can be continuously monitored to detect deviations from normal performance, allowing for targeted maintenance instead of generic, scheduled checks. (Aghaei *et al.*, 2022) proves this saves operational cost and minimises system downtime, improving overall system reliability and efficiency.

2.4 POWER GRID OPERATION AND IMBALANCE ISSUES

The operation of power grid involves carefully balancing the supply and demand of electricity. Maintaining this balance is crucial for ensuring the grid's stability and reliability. However, imbalances occur due to several factors, posing challenges to grid operations.

- **Causes of Power Grid Imbalances**

The unpredictable nature of renewable energy sources, such as solar and wind power, is one of the main reasons for system imbalances. Solar and wind power are weather dependent sources and can lead to sudden fluctuations in the amount of power generated. Research by (Soto *et al.*, 2022) found that the integration of large-scale solar power systems can cause significant grid disturbances, particularly in regions with high solar penetration. These disruptions are frequently caused due to mismatch of solar power generation and electricity demand, especially during peak hours when solar generation declines due to unpredictable weather, as demand increases.

Another factor is that the grids with a high share of inverter-based generation have low rotational inertia. Inertia from traditional synchronous generators helps to maintain grid frequencies and stabilise unexpected imbalances. Renewable energy sources like wind and solar, which are connected via inverters, lack inherent inertia. (Ulbig, Borsche and Andersson, 2014) researched about how using renewable energy sources instead of traditional generators can lead to faster frequency dynamics, which makes it more challenging to maintain grid stability.

- **Mitigation Strategies**

A number of strategies have been used to address these imbalances. Energy storage systems, such as batteries, can be used to store excess energy produced during high renewable output released during times of high demand. To help balance the supply and demand, demand response programs encourage customers to adjust their electricity usage during peak hours. Enhancements such as high-voltage direct current (HVDC) and flexible

AC transmission systems (FACTS) are being implemented to increase grid stability (Mirzapour, Rui and Sahraei-Ardakani, 2024).

2.5 DIGITAL TWIN TECHNOLOGY

The concept of Digital Twin originated in manufacturing but was soon adopted across other industries. (Jiang *et al.*, 2021) defined a Digital Twin as a virtual representation of a physical asset, continuously updating real-time data to stimulate and optimise the performance. They emphasised that the twin enables visualisation, monitoring, and forecasting how it would behave under different circumstances.

(Tao *et al.*, 2019) expanded on the use of digital twins in complex engineering and smart manufacturing, presenting a framework that integrates simulation models, real-time sensor data, and machine-learning algorithms. According to their research, digital twins could enhance the decision-making process by providing actionable insights on maintenance and asset management.

(Fuller *et al.*, 2020) discusses the broader applications of digital twins, including issues with cybersecurity, model integrity, and data integration. They concluded that the deployment of digital twins depends heavily on the quality of real-time data and the robustness of time-series forecasting algorithms embedded in the system.

- **Integration of Digital Twins and Time Series Forecasting Models**

  The concept of Digital Twin originated in manufacturing but was soon adopted across other industries. (Jiang *et al.*, 2021) defined a Digital Twin as a virtual representation of a physical asset, continuously updating real-time data to stimulate and optimise the performance. They emphasised that the twin enables visualisation, monitoring, and forecast how it would behave under different circumstances.

  (Schlechtingen and Ferreira Santos, 2011) used machine learning techniques to estimate maintenance on wind turbines SCADA data in the renewable energy sector. Even if they did not fully create a digital twin, their work set the foundation for implementing a time-series forecasting model with real-time forecasts.

  Despite these advancements, there is still a research gap in the application of time-series forecasting models based on digital twins to renewable energy systems, particularly those that combine renewable (wind, solar, etc) technologies.

2.6 FORECASTING APPROACHES: TRADITIONAL TO ADVANCED METHODS

Accurate forecasting for domains such as energy systems, finance, and climate modelling is crucial. The forecasting models have evolved from traditional statistical methods to advanced techniques, each offering its advantages and challenges.

- **Time Series Forecasting: Traditional Method**

  Time series forecasting methods such as ARIMA (Autoregressive Integrated Moving Average) and SARIMA (Seasonality Autoregressive Integrated Moving Average) have been foundational in statistical learning. These models are effective with linear trends and

seasonality. But they often struggle with non-linear patterns in modern datasets (*Forecasting: Principles and Practice (3rd ed*.)

- **Recurrent Neural Networks**

To identify temporal dependencies in sequential data, the recurrent neural networks (RNNs) were developed. RNNs, unlike traditional methods can model complex, non-linear relationships by maintaining a hidden state that captures information about previous steps. These features can be used for time series forecasting and speech recognition. RNNs face challenges called as vanishing gradient problem, which creates a problem learning over long sequences (Bengio, Simard and Frasconi, 1994).

- **Patch Time Series Transformer (PatchTST): Advanced Method**

Transformer models that were initially developed for natural language processing have significantly impacted time series forecasting. (Nie *et al.*, 2022) determined, Patch Time Series Transformer PatchTST, a transformer-based model, divides the time series data into small patches, enabling the model to capture local patterns while maintaining the overall efficiency. The difficulties of modelling long range dependencies and complex interactions in time series data are addressed by this method.

PatchTST has outperformed traditional transformer models and deep learning models in long term time forecasting tasks. Its ability to handle large datasets with multiple variables makes it suitable for smart grid management and forecasting of renewable energy.

2.7 INTEGRATING FORECASTING INTO GRID AND MAINTENANCE PLANNING

Integrating Forecasting into the power grid is necessary to increase the reliability and efficiency of modern electrical grids. Advanced forecasting approaches are being supplemented with traditional planning methods, considering the increasing use of renewable energy sources and the complexity of grid operations.

- **Role of Forecasting in Grid Planning**

(Kaur *et al.*, 2020) researched for effective grid planning, accurate forecasting of demand and the generation of electricity is crucial. Decisions on various entities like resource allocation, load balancing, and infrastructure development are influenced by forecasts. For example, long-term load forecasting (LTLF) helps in strategic planning of resource allocation and infrastructure growth, and short-term load forecasting (STLF) helps with real-time scheduling and unit commitment.

The integration of renewable energy sources results in variability and uncertainty in grid operations. Stabilising the grid depends on forecasting models that account for maintaining grid stability. (Zhang, Wang and Giannakis, 2018) suggested that improve grid management and increase the accuracy of energy generation, advanced deep learning techniques have been used, thereby aiding in better grid management.

- **Forecasting in Maintenance Planning**

According to (Rokhforoz *et al.*, 2021) forecasting of grid systems has a major impact on maintenance planning, as it allows early detection of possible problems and aids in the optimisation of maintenance schedules. Condition based forecasting and maintenance strategies use forecasting data to foresee equipment failures and plan maintenance tasks accordingly.

A time-series forecasting model estimates the remaining useful life of the components by performing analysis over the sensor data and past performance. (Cennet *et al.*, 2022) studied this approach enables the power grid to schedule maintenance only when necessary, reducing the downtime and maintenance cost.

- **Integrated Forecasting Frameworks**

Integrated forecasting frameworks combining load, generation, and maintenance prediction are essential for comprehensive grid management. Such a framework improves coordinated decision making across different operational levels. To optimise maintenance schedules while considering the operational difficulties, multi-agent systems have been proposed to model the integration between central operations and decentralised procedures (Rokhforoz *et al.*, 2021).

The scheduling of operations and time-series forecasting have been integrated using stochastic programming techniques. These models are responsible for uncertainties in system operations, system downtime, and maintenance needs, which allows decision makers to plan strategies to maintain grid reliability (Cennet *et al.*, 2022).

- **Challenges in Grid Forecasting**

Despite the advancements in technology, integration of forecasting into grids and maintenance planning challenges. Significant challenges include model accuracy, computational complexity, data quality, and data availability. Additionally, forecasting models need to be modified and updated regularly due to the dynamic nature of modern grids.

Further research should be needed on improving the scalability and adaptability of forecasting models, implementing real-time data streams, and developing standardised frameworks for the integration of forecasting into grid operations and maintenance planning. Overcoming the challenges and advances in this sector requires collaboration between utilities, researchers, and technology developers.

2.8 RESEARCH GAP AND FUTURE OPPORTUNITIES

The literature review has highlighted significant enhancements and practices in forecasting of grids and comparison of time-series forecasting models within renewable energy systems. However, there are still several research gaps that provide opportunities for further research and innovation, particularly when it comes to combining digital twin technology with time-series forecasting approaches.

- **Research Gap**

Time-series forecasting and digital twins are primarily discussed individually in the literature that is currently in publications, with a little integration of time-series forecasting

specifically in the field of solar photovoltaic (PV) systems. Time-series forecasting models in solar photovoltaic systems or renewable energy systems have proven valuable in anticipating failures and optimising systems performance, their full potential is yet to be realised within renewable energy contexts through integration with digital twins. Digital twins, which provide a virtual representation of a physical system, promise significant outcomes in forecasting capabilities by simulating real-time operating conditions and potential scenarios (Jiang *et al.*, 2021). However, existing research limits the use of digital twins to simpler task simulations, leaving a gap to implement these complex, advanced virtual models comprehensively in solar photovoltaic (PV) systems.

Additionally, current practices of traditional statistical approaches typically make use of standard analytics or traditional statistical approaches like time series forecasting models (ARIMA and SARIMA), which may not always be useful to address the complexity and unpredictability present in solar energy systems. There is a significant gap in research examining the traditional statistical approaches with modern advanced methods, particularly in solar photovoltaic (PV) systems, varied by diverse operational circumstances and fluctuating outputs.

(Nie *et al.*, 2022) determined a crucial area for additional research, which is highlighted by the lack of comparison analysis between traditional statistical models (e.g., SARIMA and ARIMA) and advanced deep learning models such as Recurrent Neural Network (RNN) and Patch Time Series Transformer (PatchTST). Advanced forecasting models, such as RNNs, have proven in capturing complex patterns and nonlinear temporal correlations that traditional models fail to represent effectively.

- **Future Opportunities**

Addressing these gaps opens up several paths for further research. The development and evaluation of a fully integrated digital twin system that is designed specifically for time-series forecasting in solar energy plants is one significant opportunity. Future research could investigate how a digital twin, combined with an advanced forecasting model, could improve the accuracy and effectiveness of time-series forecasting strategies. This research would enhance the real-time decision making for the renewable energy sector, reducing downtime, operational cost, improving system reliability, and improving the overall lifespan of a solar panel.

Further, extending comparative analysis between the traditional and advanced forecasting techniques provides another significant opportunity to expand. To elucidate the benefits and limitations of each approach, a comparison should be done between traditional statistical methods (e.g., SARIMA) and advanced methods (e.g., PatchTST and RNN) with solar photovoltaic (PV) plants datasets. Evaluating the performance of the newly released PatchTST model, which has shown significant potential in handling complex time-series datasets with enhanced accuracy and efficiency (Nie *et al.*, 2022), would provide valuable comparison insights on traditional models and the newly released model for renewable energy forecasting and maintenance planning.

Additionally, this research can also be extended to include other renewable energy technologies such as hydroelectric, geothermal, and biomass power plants. Applying time-series forecasting approaches to these sources may further accelerate the transition towards

a completely robust and sustainable energy framework and a broader and more impactful contribution to global energy goals.

By filling the existing gaps in the literature and using a time-series forecasting approach in a digital twin network, this study will contribute to actionable insights toward proactive asset management for renewable energy. It will also support global sustainability efforts by providing dependable and scalable energy solutions.

CHAPTER 3 RESEARCH METHODOLOGY

3.1 INTRODUCTION

The research methodology chapter provides a crucial role in demonstrates the systematic approach adopted to carry out research work in an organised way. It provides transparency, validity, and reliability of the findings by explaining the methods and procedures adopted. The research is significantly targeted towards comparison of time-series forecasting approaches and digital twin technology application for the efficiency and reliability optimisation of solar photovoltaic (PV) systems. Solar PV systems, being part of sustainable energy solutions, must operate at their best potential to ensure continuous energy output and cost-efficiency.

The rapid growth and reliance on renewable sources of energy point to the necessity of time-series forecasting strategies that minimise downtime and maintenance costs. Time-series forecasting through data driven approaches identifies early signs of potential failure before they escalate into major issues. To enhance this capability, integration with digital twins, a digital representation of a physical machine in real-time, provides advantages by continuous monitoring, generating real-time analysis, and scenario predictions for real-time comparison and forecasting data output.

The quantitative methodology is guided by the Cross-Industry Standard Process for Data Mining (CRISP-DM) framework. The process consists of data understanding, preparation, modelling, evaluation, and deployment phases. A variety of forecasting models, traditional statistical approaches SARIMA, and recent deep learning methods like LSTM and transformer-based approaches like PatchTST are comparatively analyzed to determine the most accurate and timely system failure forecasting model.

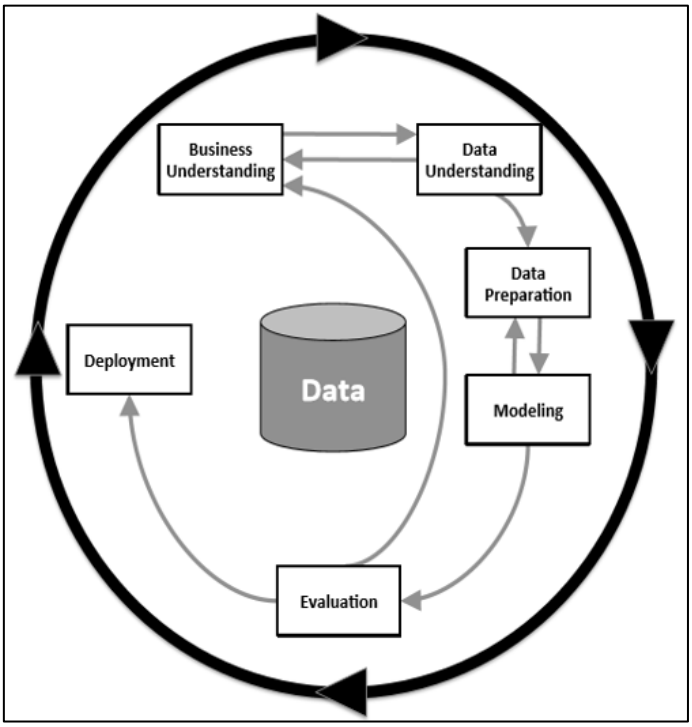

Figure 3.1 The CRISP-DM process, including the six key phases and the important relationships between them (Wirth and Hipp, 2000, reproduced in Kelleher et al., 2020, p.14)

As visualised in the figure above, the CRISP-DM process offers a cyclical and iterative strategy that ensures smooth execution of data driven projects. The comparison of time-series forecasting models in this research is managed throughout its lifecycle using these six phases: Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation, and Deployment. Since each phase builds upon the one before it, this allows flexibility and iterative refinements based on findings and insights. This structured procedure provides the simulation of digital twin, time-series forecasting, and real-time sensor data to maximise the system performance.

3.2 BUSINESS UNDERSTANDING

The objective of research is to use digital twin technology to improve the operational efficiency and dependability of solar photovoltaic (PV) systems. The main goal is to detect early failures, reduce downtime, and make decisions in real-time. Research by (Said *et al.*, 2024) suggests that this is aligned with current industry demands highlighted in renewable energy literature.

Traditional maintenance methods, such as reactive (failure-based) and predictive (planned) maintenance, have been proven to be ineffective for new solar PV installations. As highlighted by (Marc Köntges *et al.*, 2014) these practices fail to detect major issues in solar panels like Potential Induced Degradation (PID), microcracking, and soiling before significant performance degradation is realised. In addition, PV systems are exposed to environmental interactions such as solar irradiance, temperature, precipitation, and shading that affect the solar panel performance unpredictably.

The proposed system of solar panel maintenance relies on the past performance data to predict the energy generation using a time-series forecasting model, allowing prompt interventions. As (Aghaei *et al.*, 2022) described, this proactive approach can reduce maintenance costs and optimise system availability. This could be particularly implemented in a remote or utility-scale solar plant or in a field where the inspection of this site can be costly and time-consuming, real-time analysis of solar panels plays a critical role. Research by (Jiang *et al.*, 2021) suggests that real-world applications have shown improved maintenance and cost savings up to 30% – 40%.

A digital twin is a virtual copy of a physical system with real-time data updates. In the case of solar PV systems, digital twins enable constant monitoring, simulation of fault detection, and forward maintenance planning. Digital twin bridges the gap between action and data acquisition through enabling advanced analysis, simulation, root cause determination, and scenario analysis, with promising results and early trials in the renewable energy sector.

This study is set to:

- Comparison of time-series forecasting approaches using digital twin system in renewable energy environment.
- Integrate time-series forecasting models such as SARIMA, LSTM, and PatchTST and compare their accuracy and response time.
- Evaluate results using standard error metrics (MAE, MSE, MAPE, MASE) as proposed by (Cennet *et al.*, 2022).
- Propose deployment strategies using real-time data and outlining considerations for implementation in a live solar plant environment.

3.3 DATA UNDERSTANDING

In this section, we focus on understanding the dataset used for time-series forecasting of solar panel output. Data collection is done from Renewable Ninjas platform with specific targeting Solar PV performance in Ireland. The data span is for six years ranging from 2019 - 2024, and five years are utilised for time-series forecasting and one year for testing. This strategic division ensures solid modelling and an extensive validation model. Additionally, real-time data incorporation is enabled through an API connection provided by Renewable Ninjas, where direct access to real-time data performance through 20000 panels and 4000 grids is simulated.

The real-time data captured through API is crucially managed and stored using digital-twin technology, which enables dynamic updating and ongoing analytics. The historical and real-time data blend improves the prediction accuracy and reliability, offering valuable insights into the performance of solar PV systems under different operating conditions.

The primary objective here is to analyse the structure, quality, and relevance of data to ensure it can be used for the purpose of time series forecasting. We also plan to ensure any trend, anomaly, or pattern that may emerge from the data, and fix any missing values, outliers, or data format issues.

The dataset includes key features like:

- PV Output: The amount of energy generated by each solar panel.
- Temperature: Ambient temperature data that can influence panel efficiency.
- Irradiance: Measurement of sunlight intensity, crucial for energy generation predictions.
- Wind Speed: Wind speed can affect cooling, which can influence the temperature and efficiency of solar panels.
- Humidity: Humidity levels can impact the solar panel performance by influencing the electrical properties of the semiconductor material.
- Precipitation: Rainfall or snow may clean the panel surface, affecting the amount of light absorbed and impacting efficiency.

The interaction between these features will be analysed for their impact on the performance and efficiency of the solar panel. High temperature under low irradiance and high humidity can be an indicator of poor conditions, which may bring about higher degradation. Wind speed and precipitation will also be considered as potential predictors for panel performance and the need for maintenance.

TIME SERIES AND GRANULARITY

The time column indicates that the data is recorded in hourly intervals, offering a high possibility for solar PV output forecasting. This resolution is ideal for understanding the daily patterns in solar output and environmental variables.

- Time coverage: 6 full years (2019 – 2024)
- Number of records: 52,584 rows
- Use case benefit: High granularity of data allows daily pattern recognition (e.g., sunrise / sunset shifts, seasonal trends, monthly, weekly, and daily trends)

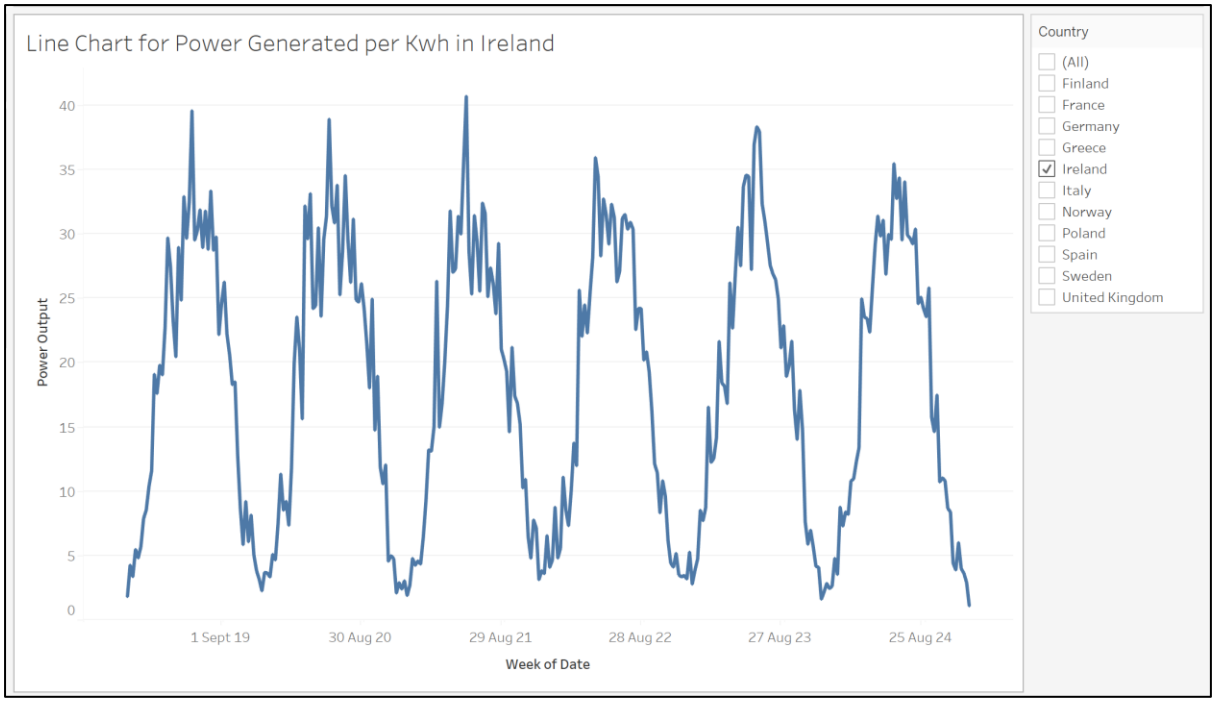

Figure 3.2 Line Chart for Power Generated per Kilowatt in Ireland (Source – Self)

This chart is visualised in Tableau Desktop for better understanding the time patterns of solar energy generation in Ireland, following is a line of weekly power production per kwh over 6 years from 2019 to 2024. The graph shows both short term variation and long-term seasonal performance of the solar panels under real deployment.

The x-axis has a smooth week-by-week trend for 6 years. The y-axis is the output in kilowatt at consistent intervals, with a high-resolution time series granularity. This granularity allows us to get insights into intra-week and seasonal patterns, ideal for the creation of time-series forecasting models that can detect performance degradation and optimise maintenance schedules.

The power output shows an annual fluctuation, with a peak value in summer (typically between June and August) and a minimum value in winter (between December and February). Seasonal change is expected through solar irradiance, daylight, and other variables such as snowfall or cloud cover. The recurring waveform over the years suggests reliable seasonality, which is well suited for time series forecasting techniques like SARIMA, RNN, and PatchTST.

The clear seasonal behaviour in the data makes it suitable for seasonal forecasting and trend modelling, which are key components of time-series forecasting. The ability to forecast these cycles with precision allows operators to:

- Plan maintenance during low output weeks.
- Adjust expectations and performance benchmarks of solar PV output over year by year.
- Detect deviations from usual curve as early indications of inefficiency.

FORMAT AND DATA TYPES

The dataset has been structured in both time and environmental performance metrics, and the formatting of these features plays an important role in time-series forecasting models. The time

stamp currently consists of time stamp values as string object types (e.g., "01-01-2019 00:00"). As human-readable, this is not best suited for time-based indexing of time series analysis. For time series forecasting, timestamps must be converted into a structured format (i.e., Python datetime objects).

Time conversion enables pandas datetime indexing, which is essential for time-aware operations like rolling mean, resampling, window-based statistics, and model fitting. This also facilitates extraction and addition of new features based on time such as hour, day, month, season or week to be treated as additional predictors. It supports time-based filtering of data, including daytime records only or separating winter months for seasonal studies.

All other columns, such as Humidity, Temperature_C, Cloud_Cover, Snowfall, PV_Output, are of float 64 data type, which is ideal for statistical operations and time-series forecasting algorithms. This enables:

- Efficient computation of statistical summaries (mean, median, standard deviation, etc.)
- Smooth integration of regression-based models such as SARIMA, ARIMA, PatchTST, and deep learning models like LSTM.
- Easy application of data scaling techniques (StandardScaler, MinMaxScaler), which are generally required for model convergence and accuracy.

Table 3.1: Data Types and values

Column Name	Data Type	Example Value
time	date	01-01-2019 00:00
Humidity	continuous	0.005
Irradiance	continuous	16.603
Temperature_C	continuous	5.068
Cloud_Cover	continuous	0.568
Snowfall	continuous	0.946
PV_Output	continuous	0.723

The table above shows the column names, data types, and example values from the dataset. Through ensuring proper data type and conversion, the dataset becomes fully compatible to be used in advanced time series forecasting and digital twin simulation platforms. The preprocessing stage is the foundation to build a robust model that can benefit from temporal dynamics, detect anomalies, and trigger a timely maintenance schedule.

OUTLIERS AND DISTRIBUTION ANALYSIS

Outlier detection is an important step in understanding the quality of data, patterns, and consistency of time-series data. Outliers in solar panels may occur due to equipment failure, environmental abnormalities, or irregular events like snowstorms or sudden cloud bursts. Identifying of outliers are extremely essential before feeding the data into time series forecasting models like LSTM, SARIMA and PatchTST.

Table 3.2: Expected behaviour of features

Features	Expected Behaviour
PV_Output	Should have many zeros (nighttime), mid-range values during cloudy days, and peaks on sunny days. Outliers may include extremely high outputs (>95th percentile).
Temperature_C	Should range between -5°C and 30°C for Ireland. Outliers may indicate sensor errors or heatwaves.
Irradiance	Should be zero at night and rise during the day; outliers may show unusually high irradiance.
Cloud_Cover	Typically, between 0 and 1. Values outside this range are erroneous.
Snowfall	Usually zero in most months; any non-zero value is important to highlight.
Humidity	Should range between 0 and 1. Outliers > 1 or < 0 indicate possible data corruption.

The table above represents general expectations of data for solar panel in Ireland.

[4]:

	time	Humidity	Irradiance	Temperature_C	Cloud_Cover	Snowfall	PV_Output
count	52584	52584.000000	52584.000000	52584.000000	52584.000000	52584.000000	52584.000000
unique	52584	NaN	NaN	NaN	NaN	NaN	NaN
top	01-01-2019 00:00	NaN	NaN	NaN	NaN	NaN	NaN
freq	1	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	0.006832	129.006354	9.750528	0.653308	0.001259	0.108908
std	NaN	0.001856	198.138742	4.518115	0.267590	0.019088	0.170009
min	NaN	0.002000	0.000000	-3.392000	0.000000	0.000000	0.000000
25%	NaN	0.005000	0.000000	6.295750	0.442000	0.000000	0.000000
50%	NaN	0.007000	5.342000	9.557000	0.692000	0.000000	0.001000
75%	NaN	0.008000	198.318000	13.067250	0.903000	0.000000	0.168000
max	NaN	0.014000	902.727000	25.099000	0.998000	1.254000	0.735000

Figure 3.3: Summary statistics of dataset (Source – Self)

To understand variability within the dataset, descriptive statistics of the dataset were computed using the describe function in Jupyter Notebook. The summary provides a clear overview of the distribution of each variable within key statistics like count, mean, standard deviation, minimum, maximum, and percentile values (25th, 50th, and 75th).

From the output, the dataset represents a wide range of environmental and performance variable values. Solar irradiance and PV Output show a wide range of spread, indicating high variability of solar power in the availability of sunlight, cloud cover, and temperature. Due to nighttime, which results in a high frequency of zero or near-zero values, certain features represent skewness in their distribution.

Overall, the descriptive analysis validates the quality and variability within the dataset and provides a foundation for the selection of appropriate data transformation, normalisation, and outlier detection techniques prior to forecasting.

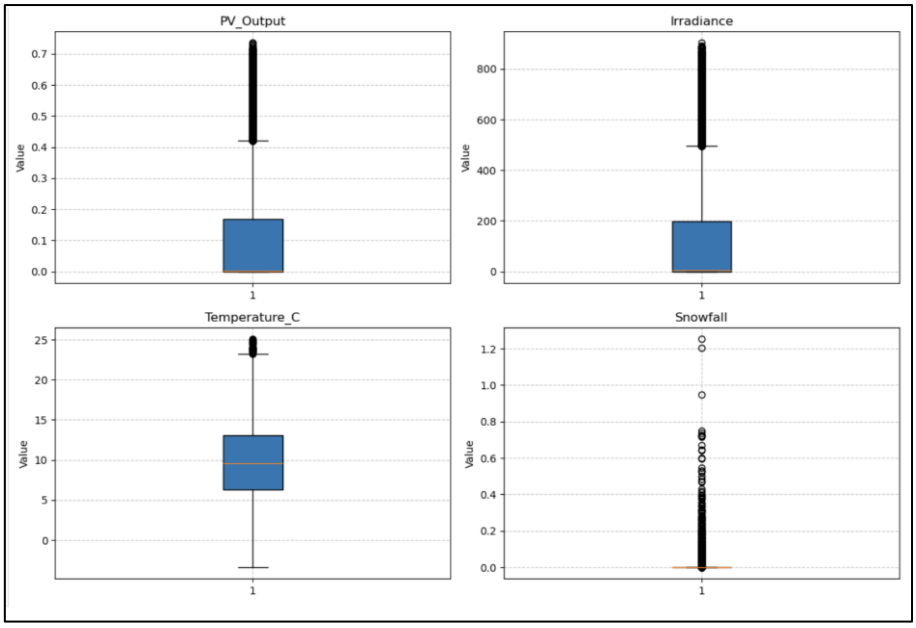

Figure 3.4: Boxplot for outlier detection (Source – Self)

The boxplots in the above figure for PV_Output, Irradiance, Temperature_C, and Snowfall are visualised in Jupyter Notebook, which provides a clear understanding of the data and whether there are any outliers present. There are many extreme upper outliers for all four features, which is what we see in actual environmental and energy data. The interquartile range in both PV output and solar irradiance is tightly clustered near the lower values with an extended upper tail. This is the exact behaviour of a solar energy system, where most of the values are tiny or zero during cloudy weather and a few observations represent peak performance under full solar exposure.

The temperature in Celsius has a balanced distribution with a box-shaped symmetry, centre median, and a few outliers. This implies stable seasonal and daily temperature variation within the expected operation ranges. Snowfall exhibits extremely meagre distribution, with the majority of values being zero and a few unique non-zero entries being present as outliers. This box plot shows that snowfall events are very uncommon in the dataset.

In summary, the boxplots ensure that the dataset contains realistic variability and outliers to be retained and not removed as noise. The outliers denote crucial operating conditions such as extreme weather and peak performance that strongly affect maintenance timings and necessity. The patterns in the boxplot support the case that advanced forecasting models that are capable of handling non-linearity and outliers' sensitivity, such as LSTM and PatchTST, should be used.

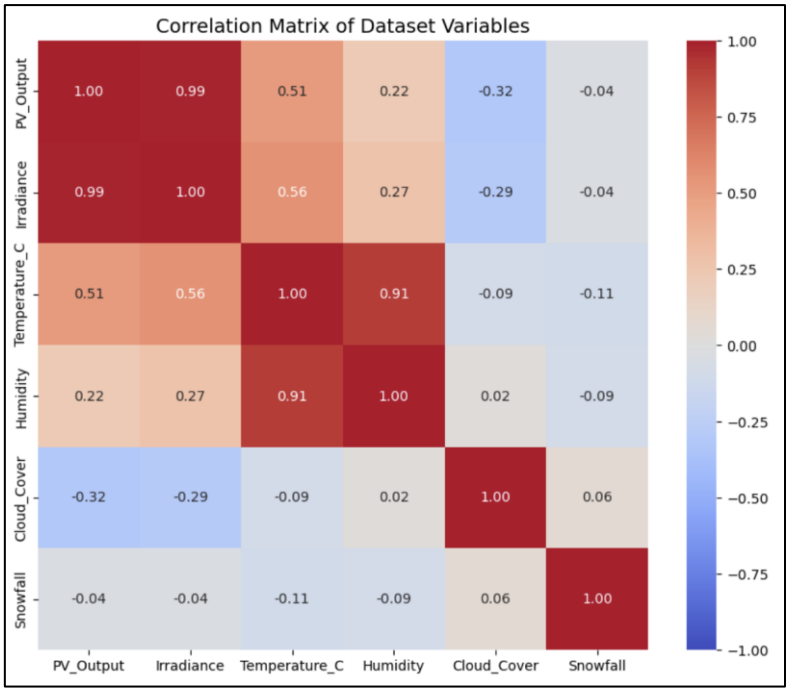

Figure 3.5: Correlation Matrix of features (Source – Self)

The correlation matrix above in Figure 3.5 is visualised in Jupyter Notebook using the matplotlib library, which shows the representation of how environmental variables and performance variables in the dataset are interrelated. Their interrelations are essential in identifying how all the factors influence the solar PV performance. It can be seen from the matrix that PV Output is correlated with Irradiance (0.99), meaning the power produced by the solar panel is very sensitive when the solar panels are exposed to sunlight. This positive correlation is supported by the physical principles of solar energy production, where irradiance tends to yield an increase in energy production (Daut *et al.*, 2012). Due to this strong correlation, Irradiance is one of the top options to be used as a solar energy production predictor.

PV Output also shows a moderately positive correlation with temperature (0.51). Though higher temperatures can result in better efficiency of solar panels, extremely high temperatures can also cause a decline in performance due to heat losses (Radziemska, 2003). Hence, the moderate positive correlation between PV Output and temperature suggests that temperature plays a crucial role in energy production. This correlation might also vary with season, with lower temperatures resulting in better panel performance in winter.

Another observation is the strong positive correlation between temperature and humidity (0.91). This indicates that within the geographic scope of the dataset, higher temperature is associated with higher humidity. These actions would affect the solar panel performance through moisture and heat, which impacts the material's electrical properties within the solar panels. The impact of humidity on solar panel should be further investigated for long-term system degradation.

While cloud cover is negatively correlated with the PV output (-0.32), which means that an increase in cloud cover reduces the solar energy output, cloud cover is also weakly correlated with temperature and humidity, which suggests that cloud cover does not have a direct

relationship with other weather conditions. Snowfall exhibits very poor correlations with every other variable, including PV output. Snowfall can be considered a very rare and random event and does not exhibit a consistent trend with temperature, humidity, irradiance, or cloud cover.

Overall, the correlation matrix shows that while Irradiance and temperature are the key factors of PV output, there are supporting roles for other variables like cloud cover and humidity to play in the analysis of solar panel activity. The findings in the matrix suggest what features need to be included and will have an impact on time series forecasting for predicting power output.

GEOGRAPHICAL SOLAR POWER OUTPUT ANALYSIS

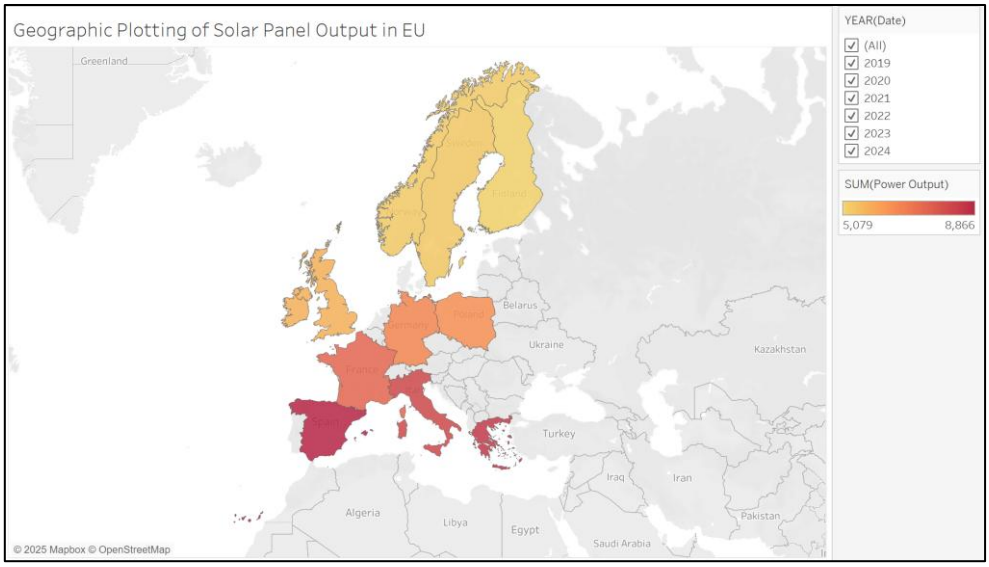

Figure 3.6: Geographic Plotting of Solar Panel Output European Countries (Source – Self)

The map is visualised using Tableau, which illustrates solar panel performance in EU countries from 2019 to 2024, highlighting significant regional differences in the production of solar energy, which changes based on environmental factors such as irradiance, weather, and infrastructure availability. The map is a gradient colour with darkest red showing maximum solar energy generation and yellow for lesser energy generation, based on the production of energy from a single solar panel in kWh. The region plays a crucial role in how conditional influence solar energy production and how variation influences solar energy policies.

Southern Europe is the most dominant with the highest level of solar production. Spain, Italy, and France are strongly shaded in dark red, illustrating continuous high-power generation. These southern European regions are defined by more extended sun hours, more transparent skies, and an overall improved climate, which maximises the solar energy generation. Spain has high solar irradiance, with maximum investments in solar energy. The leadership of solar energy production is due to their geographical location and highly developed infrastructure in solar energy.

Central European countries have a moderate level of solar energy. Poland and Germany both fall under the orange category, which means that although they receive good sunlight, the unpredictability, cloud cover, and reduced daylight hours restrict their solar potential. Northern

Europe receives the lowest solar output, which is represented in yellow on the map. Countries like Sweden and Finland are characterised by their long winter and short daylight, which generally limit their solar energy production. This geographical variability highlights the significance of regional time-series forecasting models, taking into account local environmental conditions to maximise solar panel performance and reduce downtime.

CONCLUSION

The data understanding phase has provided a thorough analysis of Renewable Ninjas 6-year solar panel performance data from 2019 to 2024. The data is available with no missing values, and it is appropriately formatted for time series forecasting, offering hourly data that is capable of capturing daily fluctuations as well as seasonal changes in solar energy production. The key attributes, such as PV output, irradiance, humidity, and wind speed, influence the solar panel's performance over time. Descriptive statistics and box plots revealed some unique trends of solar energy production based on seasons, with summer yielding high outputs and winter yielding low outputs. The correlation matrix indicated high correlation between irradiance and PV output (0.99), and temperature with humidity (0.91), validating their significance in models. The data is well structured for time series forecasting models like SARIMA, LSTM, and PatchTST, with features configured for scaling, transforming, and output handling. In summary, the dataset is clean, robust, and very suitable for time-series forecasting and prediction, a good foundation for which to develop models that can optimise output cycles depending on environmental factors.

3.4 DATA PREPARATION

Data preparation is a process of building robust forecasting models. The process involves completely transforming the raw data into a clean, structured, and usable format. In the prediction of solar panel output, data preparation involves some significant activities such as handling missing data, transforming data into appropriate data types, feature engineering, and ensuring that the dataset is ready for the modelling algorithm.

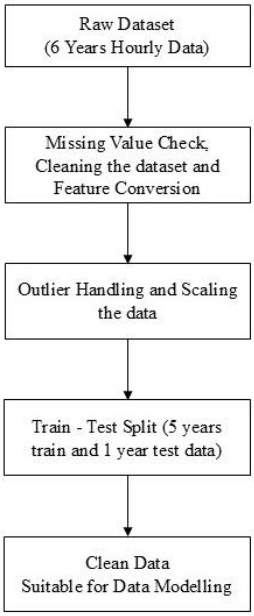

Figure 3.7: Flow Chart for Data Preparation (Source – Self)

The flow chart above for data preparation is visualised in Microsoft PowerPoint, which summarises the step-by-step process of preparing the solar PV dataset for predictions. The process starts from a raw six-years hourly dataset, followed by handling and checking of missing values. Running the imputation process if it is required, otherwise going ahead with cleaning and feature conversion. Outliers handling and scaling is done so that feature magnitudes are standardised. Further, the data is separated into a training and test split for 5 years of train and 1 year testing data, ensuring the dataset is ready for accurate forecasting.

MISSING VALUES AND DATA COMPLETENESS

A critical component of data preparation, especially for time series forecasting, is ensuring that the dataset doesn't consist of missing or null values. Missing values can lead to the training of forecasting models and may lead to biased decisions, particularly when the data is structured temporally.

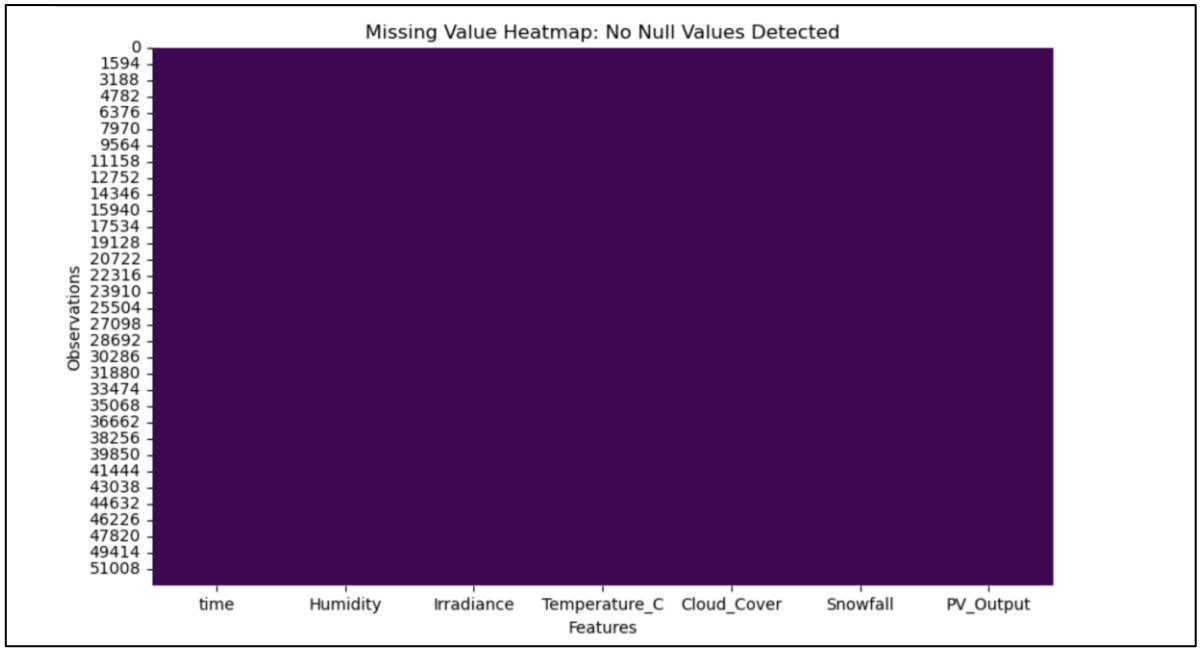

Figure 3.8: Heatmap for Null Value Detection (Source – Self)

The above figure is visualised by seaborn in Jupyter Notebook, which gives insights through a blank (dark) heatmap, indicating there are no missing (NaN) values present in any of the columns: time, Humidity, Irradiance, Temperature in Celsius, Cloud Cover, Snowfall, and PV Output. Each of the 52,584 hourly observations contains a set of values that are complete and consistent. This eliminates the need for imputations in data such as forward filling, backward filling, or interpolation, which are typically employed if missing values occur in time series data. The dataset can therefore be applied in modelling immediately without preprocessing overhead data and ensuring reliability of historical trends.

DATA CLEANING AND FEATURE CONVERSION

The dataset that will be used for forecasting of solar panel output consists of 6 years of hourly data. The initial step is to make sure that the dataset is well formatted, especially the time column. From the data understanding section, the time is originally in string format. When performing time-series analysis, it is important to convert the column into a datetime format using pandas. This conversion allows us to perform time-based indexing, resampling, and extraction of additional time-related features such as hour, day, month, and season.

Additionally, it is important to ensure that continuous variables like humidity, irradiance, temperature, etc, are kept in numeric data types (e.g., float 64) for time-series forecasting and statistical analysis. In this case, the dataset follows the same format, so minimum adjustments are required in terms of data types for these features.

INVALID DATA REMOVAL (29TH FEBRUARY 2021)

An incorrect timestamp of 29th February 2021 was discovered in the dataset, as 2021 is not a leap year. This record was removed from the dataset as it was an invalid date, and its presence could lead to incorrect analysis or forecasting output. This maintains the integrity of time-based analysis, ensuring that the dataset consists of only calendar dates that are valid.

HANDLING OUTLIERS AND SCALING

Outliers are identified in the previous section through box plots and descriptive statistics. Outliers must be handled with care as they point towards significant events such as peak solar generation or extreme weather conditions. Outliers could be capped, removed, or retained depending on the nature of the outliers.

Scaling is the second most important step in data preparation, particularly for scale-sensitive deep learning algorithms (e.g., neural networks). Applying normalisation and standardisation to features like temperature, PV output, irradiance, and humidity ensures that the features have similar magnitudes, which will improve the accuracy of the model.

DATA PREPARATION FOR MODELLING

Model training is the process of feeding historical data to the selected models and optimising their parameters to reduce forecasting errors. Training is a repetitive process with the aim of ensuring proper model performance evaluation. Performance for each model is evaluated by making use of a range of evaluation metrics to select the most suitable model for solar PV system failure prediction.

- Training Data

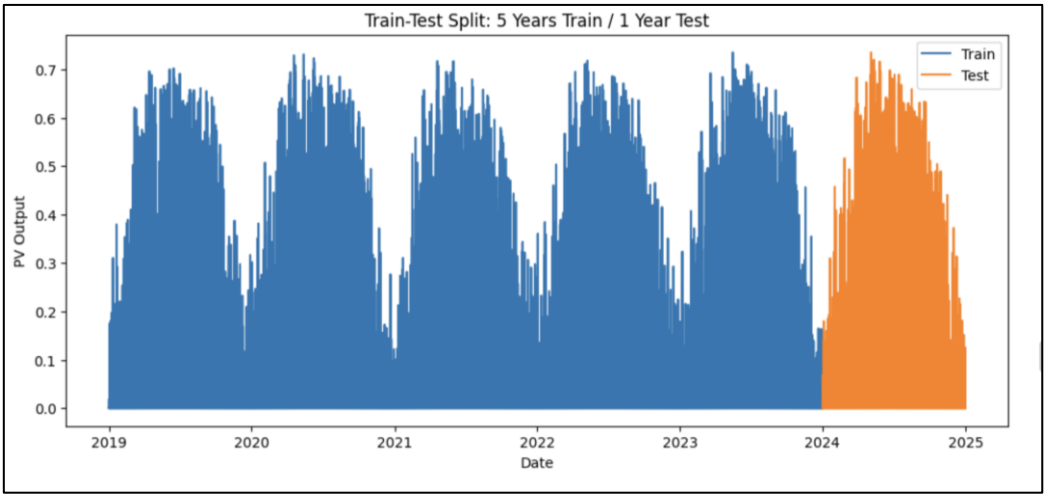

Figure 3.9: Train-Test Split (Source – Self)

The training dataset consists of five years of historical solar PV data, which is pre-processed and divided into training and validation sets. In Figure 3.9 above, the blue highlighted line chart displays the train data, and the orange highlight shows the validation data. The dataset is normalised so order that the input variables will be on the same scale, which is important to the performance of models like RNN and PatchTST, which are sensitive to the distribution of input data.

FINAL CHECKS AND DATA QUALITY ASSURANCE

Before training the model, it is necessary to ensure that the data meets the following requirements:

- No missing values: Since the dataset is free from missing values, no imputation of data is required.

- Correct formatting and consistency: There must be columns with the respective data types (e.g., date and float 64).
- Stationarity: Stationarity of data must be checked in case of SARIMA and ARIMA models. If required, data must be transformed (i.e., differenced) such that the data is stationary.

CONCLUSION

The data preparation phase transforms the raw data into a form that is compatible with the forecasting models so that we can capture the dynamics of solar panel performance and determine the perfect timing for maintenance or degradation of the solar panel. Converting the timestamps, creating new time features, handling outliers, scaling the data, and resampling are the steps that we follow to make sure that the data is ready for time-series forecasting. This prepares the data for accurate solar output prediction and the schedule of operation and maintenance.

3.5 MODELLING

The modelling section provides the development of time-series forecasting models that are used to forecast solar photovoltaic (PV) system faults and maintenance needs. The integration of time-series forecasting and digital twin technology opens a new category in monitoring and optimisation of solar PV power plants. The objective of modelling in this research is to predict the operational performance of solar PV systems, as well as their modes of failure, based on historical data and real-time data obtained through digital twin technology. The models selected are designed to make solar PV output more efficient, reliable, and less expensive through advanced forecasting.

SELECTION OF TIME-SERIES FORECASTING MODELS

The accuracy of time-series forecasting models is used to ensure that the system's performance is accurately forecasted. In this research, we are using a range of models from traditional time-series forecasting models like ARIMA and SARIMA to deep learning algorithms like LSTM and transformer-based models (PatchTST). These models are selected based on their effectiveness in detecting time-dependent relationships as well as their ability to process large datasets, as in our case for solar PV system data.

- ARIMA (AutoRegressive Integrated Moving Average)

ARIMA is a traditional, linear-time series model that forecasts a variable based on its own past and past forecast errors after low frequency drift has been removed through differencing. As suggested by (Vijay Kotu and Bala Deshpande, 2019) it can be written as ARIMA (p, d, q) where p is autoregressive order (the number of previous values included), d is differencing order (the number of times the series needs to be differenced to render it stationary) and q (the number of past errors included) is the moving average order. ARIMA provides a strong horizon, it detects local autocorrelation patterns of generation without extra inputs, and it is fast, interpretable, and easy to debug.

ARIMA (p, d, q) is defined in equation 1 below:

$$\phi(B)\,(1-B)^{d}y_t = c + \theta(B)\,\varepsilon_t \text{ ----------(1)}$$

A defensible ARIMA workflow is: (1) stabilise variance (optional log/Box-Cox); (2) achieve stationarity by setting d by plots and tests (e.g., ADF/KPSS) and inspection of ACF decay; (3) infer p and q from ACF/PACF patterns, then compare candidates by AIC /AICc /BIC; (4) estimate parameters with maximum likelihood; (5) inspect residual they must resemble white noise (Ljung–Box over a few lags, residual ACF/PACF flat). If residual seasonality remains (e.g., diurnal, annual), ARIMA is extended to SARIMA or is supplemented by the inclusion of Fourier terms. For multi-step horizons, make recursive direct forecasts of output prediction intervals depending on the variation of innovation.

ARIMA suits best for 1-6 hours forecasting for solar PV systems where recent output dynamics dominate and exogenous drivers are few. If there are some weather changes, such as cloud cover, temperature, and irradiance, it can be improved by an ARIMAX specification, which conditions on such predictors but retains AR/MA form in errors.

- SARIMA (Seasonal ARIMA)

SARIMA is an extension of ARIMA that incorporates seasonal factors within the forecasting models. This is relevant to solar PV production that exhibits strong hourly data m = 24 or often annual seasonality m = 24. As said by (Vijay Kotu and Bala Deshpande, 2019) the SARIMA model handles (i) non-seasonal dynamics (short-term autocorrelation and shocks) and (ii) seasonal recurrence (periodic peaks /troughs) simultaneously, yielding more stable forecasts when periodic structure exists.

SARIMA model (p, d, q) (P, D, Q) m is written in equation 2 below:

$$\phi(B)\,\Phi(B^m)\,(1-B)^d\,(1-B^m)^D\,y_t \;=\; c \;+\; \theta(B)\,\Theta(B^m)\,\varepsilon_t, \text{ ----------(2)}$$

Identification and estimation of a practical workflow are:

Season detection: Look for strong spikes in the ACF at multiples of m (e.g., 24, 48, …). Choose m accordingly (hourly PV at Athlone: m=24).

Differencing: Apply seasonal differencing D (and non-seasonal d) until the series appears stationary (flattened seasonal ACF, ADF/KPSS results acceptable).

Order selection. Use seasonal ACF/PACF features to indicate (P, Q) and non-seasonal ACF/PACF for (p, q); confirm with AIC /AICc /BIC on a small grid (e.g., p, q, P, Q $\in$ {0,1,2,3}; D $\in$ {0,1}).

Estimation and diagnostics: Estimate parameters by maximum likelihood; residuals should be uncorrelated at ordinary and seasonal lags (check Ljung–Box at lags like 24, 48). Any residual seasonal structure signals under-differencing or too low seasonal orders.

For hourly forecasting, SARIMA (p, d, q) (P, D, Q) 24 captures the daily cycle of irradiance while preserving the short-term dynamics. It tends to outperform the non-seasonal ARIMA when seasonal patterns are strong and there are no important exogenous features. For multi-step horizons, forecasts are generally recursive, with increasing prediction intervals as the horizon increases based on the estimated innovation variance.

SARIMA is interpretable, computationally efficient, and well-suited to periodic PV time series. Weaknesses include linearity (non-linear effects are not modelled),

sensitivity to structural breaks (e.g., sudden soiling/cleaning), and potential degradation in performance at very long horizons, motivating comparison with non-linear learners (RNNs, Transformer-like models) in subsequent sections.

- Recurrent Neural Networks (RNNs)

RNNs are a part of deep learning algorithms that are designed to handle sequential data, which makes them well-suited for time-series forecasting. RNNs have the ability to maintain a hidden state, which allows them to capture long-term dependencies and non-linear relationships in sequential data. RNNs are able to handle the complexity in renewable energy data that is hard to model using traditional models like ARIMA.

Standard RNN struggles with long-term dependencies due to the vanishing gradient problem. Long Short-Term Memory (LSTM) network addresses this by the introduction of a cell gate and controls information flow.

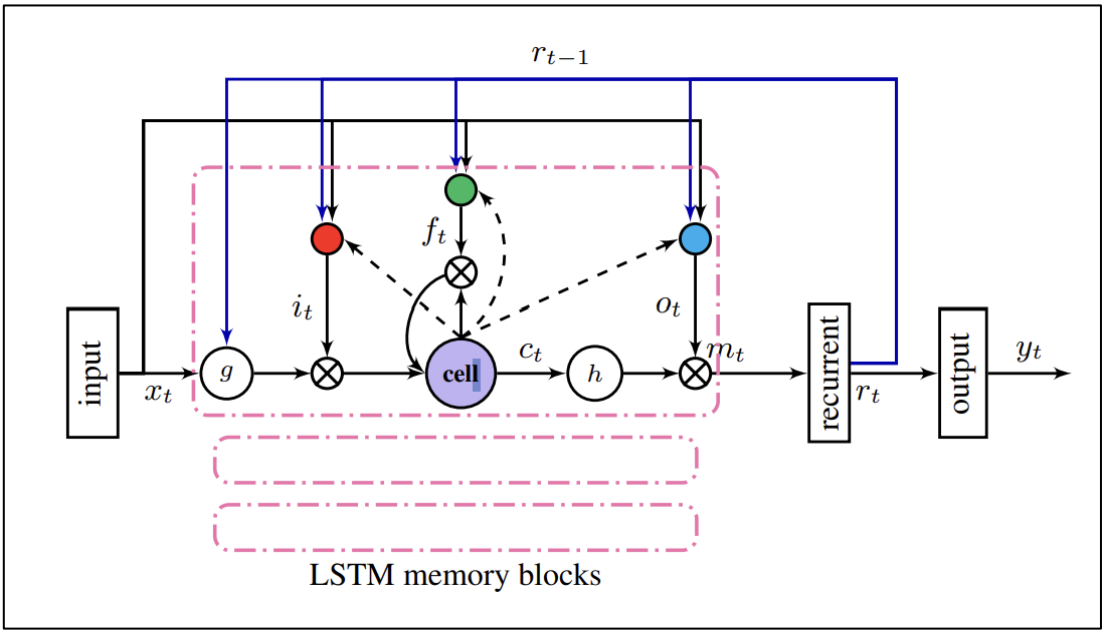

Figure 3.10: LSTM RNN architecture. A single memory block is shown for clarity. Source - (Sak, Senior and Google)

The figure 3.10 above shows the internal computational flow of an LSTM memory block, which is designed to capture both long-term and short-term dependencies in sequential data.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \text{ ----------(3.1)}$$

Input Gate in equation 3.1 above controls new information from current input x_t and previous hidden state r_{t-1} is allowed into the cell state.

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + b_f\right) \text{ ----------(3.2)}$$

Forget Gate in equation 3.2 above determines the proportion of previous state c_{t-1} that is retained or discarded, enabling the network to reset outdated information

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \text{ ----------(3.3)}$$

Output Gate Regulates in equation 3.3 above represents the exposure of the updated cell state to its hidden state h_t and ultimately to the output y_t.

$$\widetilde{c_t} = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \text{ ----------(3.4)}$$

Candidate Cell Gate in equation 3.4 above represents in equation 3.4 above represents the potential that new content could be added to the cell state.

$$c_t = f_t \odot c_{t-1} + i_t \odot \widetilde{c_t} \text{ ----------(3.5)}$$

$$h_t = o_t \odot \tanh(c_t) \text{ ----------(3.6)}$$

Here, $\odot$ in equation 3.5 and 3,6 denotes element wise multiplication. The gate learns when to store, forget or expose information enabling much longer memory spans.

- Patch Time Series Transformer (PatchTST)

Patch Series transformer (PatchTST) is an adaptation of the transformer model initially proposed by (Vaswani *et al.*, 2017) for natural language processing to the time series forecasting domain. Unlike other transformers, which process each timestamp independently, Patch TST clusters consecutive time steps together into patches before embedding them. The method pays attention to local patterns within patches while capturing global dependencies within the entire sequence.

In recent times, (Nie *et al.*, 2022) presented PatchTST, a transformer model that highly enhances time series forecasting performance by addressing short-term and long-term dependencies. PatchTST partitions the time-series data into patches so that the model can run local patterns while keeping all the overall context. This transformer is particularly beneficial for processing large-scale data with multiple variables, including data coming from solar PV systems on a real-time basis.

The diagram below, in figure 3.11 by (Tang *et al.*, 2025) presents the processing pipeline for PatchTST for time-series forecasting. The input series is first normalised and then patched, in which consecutive time moments are aggregated in an attempt to highlight local patterns. These patches are then passed through a projection layer and augmented with positional embeddings in an attempt to store temporary information. These patches are embedded and fed into a transformer encoder, in which multi-head self-attention is employed to extract long-range and long dependencies before being fed into feed-forward layers with residual connection and normalised. The encoder output is flattened and fed into a linear head to output the predicted series of outputs.

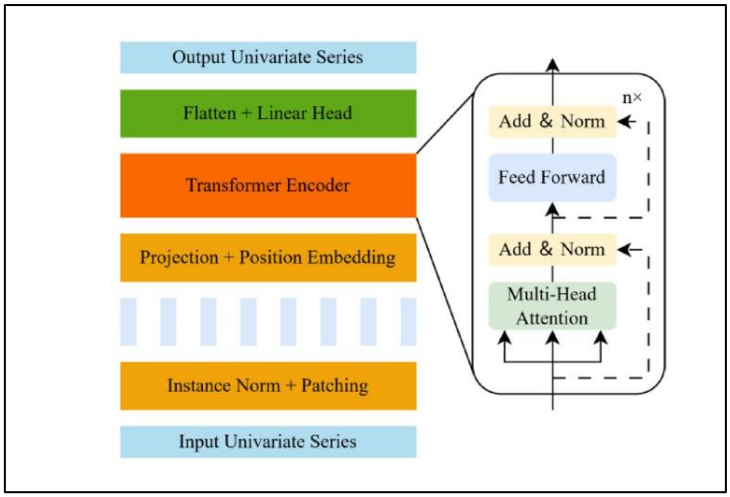

Figure 3.11: PatchTST Architecture Diagram Source - (Tang *et al.*, 2025)

IMPLEMENTATION OF DIGITAL TWIN FRAMEWORK

Once the best-performing time-series forecasting model is established, it is going to be integrated into a digital twin framework. The digital twin is the real-time simulation of the solar PV system, which will be continuously updated. This integration enables a proactive maintenance schedule based on the forecasting insights generated by the model. The real-time data, along with the predictions from the trained model, ensures the possibility of detecting anomalies, failure points, and down-time before they turn into severe problems.

- Proactive maintenance: The model identifies periods of underperformance or degradation and schedules maintenance-based activities based on forecasting insights, ensuring that maintenance is done at the optimal time.
- Cost reduction: By forecasting the downtime and failures even before they arise, the system minimizes downtime and reduces unwanted maintenance activities.

CONCLUSION

The modelling section of this research outlines the process of selecting various time-series forecasting models for solar PV data. Traditional models like ARIMA and SARIMA provide a good foundation but fail when dealing with complex and non-linear data patterns. RNNs and PatchTST are advanced models that provide high accuracy and scalability, making them suitable for real-time time-series forecasting.

By integration of these models with digital twin technology, the research aims to provide actionable insights that can significantly improve the efficiency, reliability, and cost effectiveness of the solar PV maintenance. The outcome of this research is to develop more scalable and robust time-series forecasting models that can be used globally.

3.6 EVALUATION

The evaluation of time-series forecasting models for solar photovoltaic (PV) systems maintenance is essential to ensure the models provide reliable and actionable predictions for performance degradation and failure prediction. In this section, we will discuss the main evaluation metrics to be utilised to assess the accuracy, robustness, computational efficiency, stability, and real-time applicability of the models. The four performance metrics used are as

follows: Mean Absolute Error (MAE), Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE), and Mean Absolute Scaled Error (MASE).

Forecast accuracy is the most important measure to compare different time-series forecasting models. Accuracy plays a crucial role in effective time-series forecasting because the predictions of the model will determine when and how maintenance interventions are scheduled.

- Mean Absolute Error (MAE)

  MAE is one of the simplest and most widely used for error metrics, which provides average error size in a set of predictions without considering the direction. MAE is defined in equation 4 below:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \text{ ----------(4)}$$

- Mean Squared Error (MSE)

  MSE is widely used to evaluate time-series forecasting models. It predicts the average of the squared errors, which penalise large errors more then MAE. MSE is defined in equation 5 below:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \text{ ----------(5)}$$

- Mean Absolute Percentage Error (MAPE)

  MAPE estimates the predictions accuracy in percentage terms, which is a simple measure to read. MAPE is defined in equation 6 below:

$$\text{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \text{ ----------(6)}$$

- Mean Absolute Scaled Error (MASE)

  MASE is a more advanced measure that standardises the error against the errors of naïve forecasting method, typically the naïve forecast or the average values of previous ones. MASE is defined in equation 7 below:

$$\text{MASE} = \frac{\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|}{\frac{1}{n-1}\sum_{i=2}^{n}|y_i - y_{i-1}|} \text{ ----------(7)}$$

COMPUTATIONAL EFFICIENCY

Computational efficiency plays a crucial role in the successful deployment of time-series forecasting models, particularly in real-time applications where quick decision-making is essential. The efficiency of the model is measured not only by its efficiency, but also by the efficiency with which it can process the information to make predictions. In solar photovoltaic, where a large amount of real-time data is collected in a continuous manner, computational efficiency determines whether using a model to make real-time decisions is possible or not.

Some key factors for computational efficiency:

- Training Time: This refers to the time taken by the model to learn from the available data. Training time becomes significant when dealing with big data. The best algorithm and model structure can significantly reduce time, allowing the system to be updated more frequently as new data is obtained.
- Prediction Time: After a model is trained, it must predict quickly when new data is entered into the system. During real-time applications, the prediction time must be as low as possible so that fast maintenance decisions can be carried out.
- Resource Utilisation: Time-series forecasting models particularly, deep learning models, are likely to be computationally resource-intensive and will require more memory usage and computational resources.

Improving the computational efficiency is essential to make sure the time-series forecasting model can run smoothly on available hardware resources, especially when multiple solar panels generate huge data.

PRACTICAL APPLICABILITY

Practical applicability depends on the simplicity of the time-series forecasting model, which can be directly put in the current operational setup of a solar PV system. For the model to be useful, it must ensure that the predictions are correct, but also send these insights to the maintenance person. Practical applicability of the model is based on the following:

- Ease of integration: The model must be compatible with existing infrastructure, including data storage systems and real-time monitoring systems. Seamless integration ensures that the model can be deployed without significant modifications to the architecture of the system to save time and expense for deployment.
- User-Friendly interface: User-friendly time-series forecasting models are required, or a user-friendly interface for results is required, the technical operators may not be technically skilled in machine learning or data science. The outcome produced by the algorithm should be visualised and made interpretable.
- Flexibility to real conditions: Solar PV systems are dynamic, with fluctuating environmental conditions, panel configurations, and operational procedures. A time-series forecasting model must remain flexible to changing conditions and continue to make accurate predictions for a time span. Such flexibility ensures the long-term survival and usability of the model in constantly changing conditions.

Practicality is key because the goal of time-series forecasting is to provide actionable insights that maximizes operations, minimise downtime, and improve the overall reliability of solar PV systems. A model that is not simple or practical may struggle to function in real-time scenarios.

CONCLUSION

In this section, we discussed the evaluation metrics that will be used to assess the performance of time-series forecasting models for solar PV maintenance. We will be using the four metrics for evaluation of performance: MAE, MSE, MAPE, and MASE, each with its advantages and applications in assessing the reliability of the prediction model. Additionally, combined with strength, computational efficiency, and real-time implementation, these measures ensure that the models used are fit for real-world applications.

The next step in this research will be finding the best model that balances the accuracy, efficiency, and scalability in real-time time-series forecasting, thereby guaranteeing improved operational performance for solar PV systems.

3.7 DEPLOYMENT

ARCHITECTURAL OVERVIEW

The deployment phase is about releasing the time-series forecasting model in such a way that real-time analysis can be set up, visualised, and acted upon. The phase completes the loop between model development and actual usage in a way that stakeholders can interact with the predictions and performance metrics in an easily consumable and decision-enabling manner.

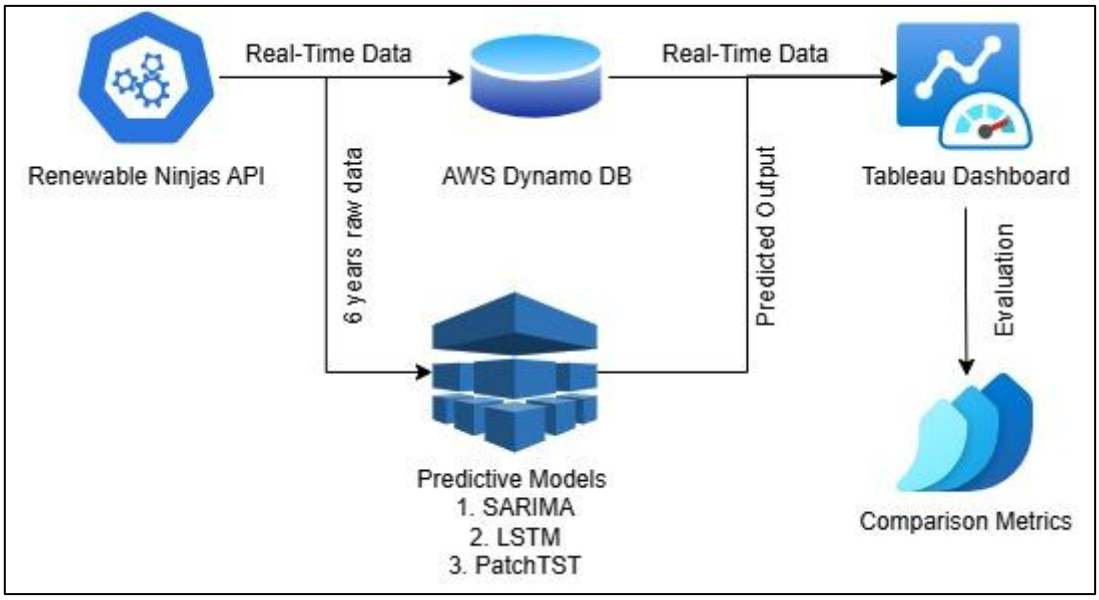

Figure 3.12: Architectural Overview (Source – Self)

The figure above is visualised in draw.io, which shows the deployment workflow for this research, which is integrated with AWS DynamoDB for real-time data storage, a time-series forecasting model, and Tableau for visualisation. The architecture allows for continuously ingestion, storage, and processing of the arrival solar PV performance data from a solar plant, as well as real-time model performance.

- Data storage

Real-time Data Collection: The system integrates with the Renewable Ninjas API to stream real-time solar PV data (e.g., irradiance, panel temperature, humidity, PV output) hourly.

AWS DynamoDB: The real-time storage with DynamoDB was chosen, which was due to its low latency, high availability, and scalability to handle the constant and high-value data stream from several panel clusters.

- Model Deployment and Integration

Batch and real-time predictions: Historical predictions are stored in the baseline performance assessment. Real-time data is stored in DynamoDB, is fetched, and predictions are compared with real-time data for immediate visualisation.

Model comparison: Predictions of all models are logged alongside actual outputs and are visualised in a line chart. Evaluation metrics, e.g., MAE, MSE, MAPE, and MASE, are used to select the perfect model.

- Visualisation in Tableau

Live Data Connection: Tableau will be connected directly to AWS DynamoDB to enable real-time dashboards. Line chart outputs predicting PV output against actual output for the current day/week. Real-time tracking of irradiance, humidity, and temperature alongside prediction accuracy.

CONCLUSION

The deployment framework integrates AWS DynamoDB, Tableau, and advanced forecasting models to develop an end-to-end real-time time-series forecasting for solar PV systems. Using DynamoDB scalability and low-latency capabilities, the system enables seamless ingestion and storage of real-time performance data from one solar plant. Three forecasting models, SARIMA, RNN, and PatchTST, are applied to the real-time data to check the accuracy of forecasting models. Dashboards with Tableau provide easy to understand visualisation of model performance, weather, and forecast vs actual plots to enable informed, ahead of time decisions by maintenance crews.

The deployment approach not only allows for real-time monitoring and immediate anomaly detection but also facilitates model improvement through interactive updating and tracking performance. Closing the loop from data collection, prediction, and visualisation, the system makes forecasting an actionable, scalable, and operator friendly solution.

CHAPTER 4 RESULTS AND ANALYSIS

This chapter provides the findings that are obtained from research, with an emphasis on how various models of forecasting are performed when used to forecast solar photovoltaic (PV) output. The main purpose of this section is to analyse how the chosen models ARIMA, SARIMA, Long Short-Term Memory (LSTM), and PatchTST performed when predicting solar hourly output. By comparing the results, the study gives insights into the strengths and weaknesses of the different methods used in the context of renewable energy forecasting.

The models are compared based on the accuracy measures such as Mean Squared Error (MSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Mean Absolute Squared Error (MASE). Besides numerical performance, consideration is also placed on the capability of models with seasonality, non-linear trends, and huge data size granularity.

This chapter is organised with each model appearing distinctly, starting with traditional statistical approaches such as ARIMA and SARIMA, followed by deep learning and transformer models, LSTM and PatchTST. Each of the subsections describes the implementation issues, the result of performance, and the implications of findings. This chapter ends with a comparative analysis, study limitations, and ethical implications around data use and modelling.

4.1 ARIMA MODEL

The Autoregressive Integrated Moving Average (ARIMA) model was the initial method used to predict the photovoltaic (PV) generation in this research. ARIMA is a traditional time series forecasting model that represents the data in terms of three primary components: autoregression (AR), differencing to attain stationarity (I), and moving average (MA). Although it is commonly used for forecasting in many fields, including economics, energy demand, and climate prediction, the ARIMA model faces a lot of challenges when it is applied to the project's dataset.

The dataset included 6 years of hourly PV data and environmental parameters. ARIMA holds a linear form and works optimally with stationery and time-series, working with such a large dataset that contains seasonality and non-linear patterns, which proved to be difficult to predict the output with multiple 0's. Hourly granularity provided high-frequency fluctuations that ARIMA was not able to capture well. Although differencing minimised some trends, it also caused data loss and increased the complexity of the model, resulting in higher computation and longer training time.

STATIONARITY AND DATA PREPARATION

The first step in applying ARIMA was to ensure the stationarity of the PV output data. Stationarity is important as ARIMA is based on the premise that statistical properties like mean and variance remain constant over time. The Argumental Dickey Fuller (ADF) test was conducted on raw data to check the data consistency over time. The test produced a statistic of –2.57 and a p-value of 0.098, which provided weak evidence against the null hypothesis of a unit root. This revealed data is initial dataset was not stationary.

```
ADF Test Statistic : -2.574127946794259
p-value : 0.09846325477047069
#Lags Used : 14
Number of Observations Used : 2177
weak evidence against null hypothesis, time series has a unit root, indicating it is non-stationary
```

Figure 4.1: ADF Test Result 1 (Source – Self)

```
ADF Test Statistic : -12.988046035186882
p-value : 2.8508157694138538e-24
#Lags Used : 7
Number of Observations Used : 1819
strong evidence against the null hypothesis(Ho), reject the null hypothesis. Data has no unit root and is stationary
```

Figure 4.2: ADF Test Result 2 (Source – Self)

To compensate for this, seasonal differencing was used, after taking the seasonal difference for 1 year, that is, 365 days. The ADF test statistically generated greatly improved to -12.99, with an effective p being < 0.005. This gave firm evidence against the null hypothesis and verified that the data is stationary now.

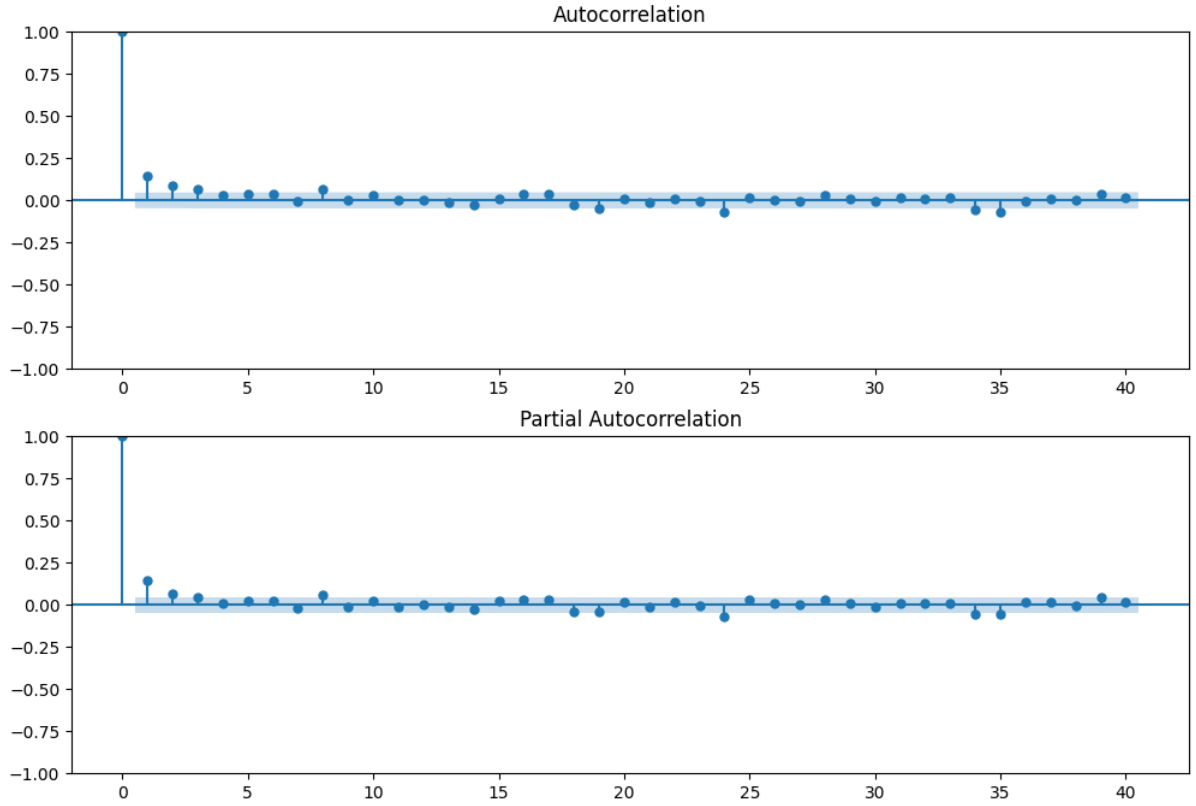

Figure 4.3: Autocorrelation for ARIMA (Source – Self)

After establishing stationarity, the ACF and PACF plots were checked to determine the possible MA and AR terms. The ACF plots have a big lag at small spikes at small lags, whereas the PACF had pointy cutoffs at small lags, which was in accordance with having autoregressive components. Yet both plots revealed residual seasonality and uneven volatility, which implies the full dynamics of the series could not be captured by a straightforward ARIMA model.

The autocorrelation structure also captured the nonlinearity and highly stochastic nature of solar generation, in which environmental factors like irradiance and temperature have a nonlinear effect on output. While ACF and PCAF were helpful, they highlighted the

fundamental challenge of modelling a purely statistical ARIMA to renewable energy data that are both short-term noisiness and long-term seasonality.

HOURLY FORECASTING ATTEMPTS

The model was initially attempted on hourly data. But ARIMA did not yield significant results. The output either forecasted reduced to flat lines or became computationally too hard to fit because of the dataset's complexity. This results in one of ARIMA's key shortcomings: while it can operate well on smaller, aggregated datasets, it is not scalable on high-frequency, highly volatile time-series data like hourly PV generation.

Because hourly forecasting posed difficulties, the series was reduced to daily values to minimise noise and make the data more manageable for ARIMA. Successfully running at this level, the model was capable of generating forecasts. Even for the daily data, the results showed considerable failings. The ARIMA generated forecasts were generally linear with linear fluctuations against daily PV output. This can be seen in figures 4.4, 4.5, and 4.6 below, where the daily output showed peaks and trends, while the ARIMA forecasts smoothened these fluctuations entirely.

This highlights the structural limitations of ARIMA. Its use of linear assumptions overcomplicates nonlinear, weather-dependent dynamics that are important for solar forecasting. Consequently, even though the model did converge on the daily data technically, the resulting forecasts were not practically useful for accurately capturing the real variability of PV generation.

MODEL PERFORMANCE

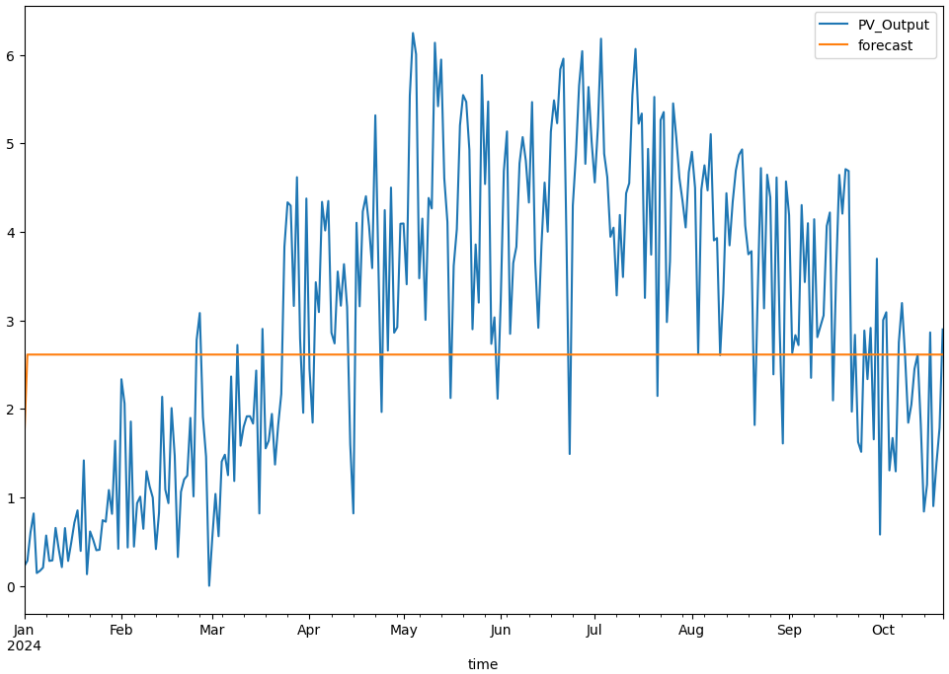

Figure 4.4: Model 1 ARIMA (Source – Self)

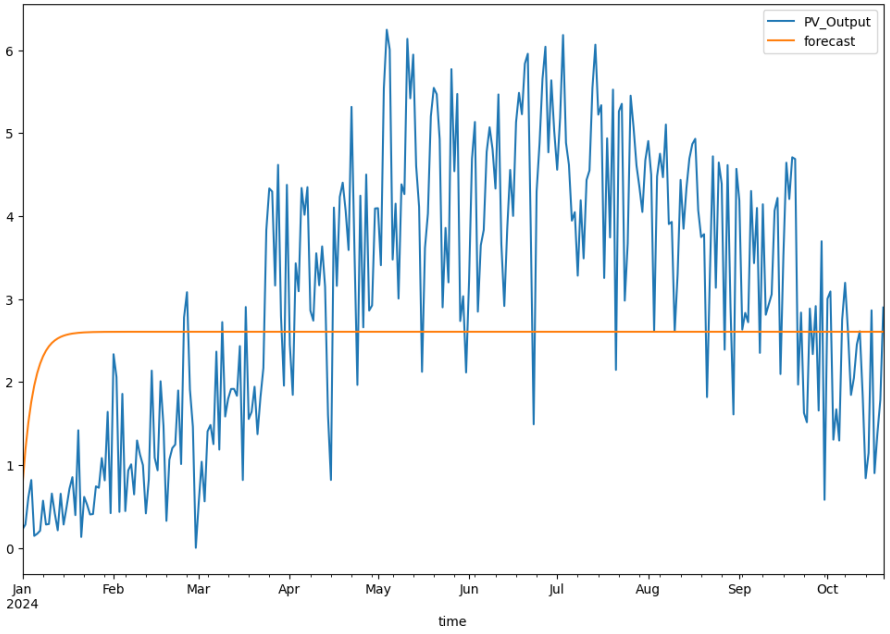

Figure 4.5: Model 2 ARIMA (Source – Self)

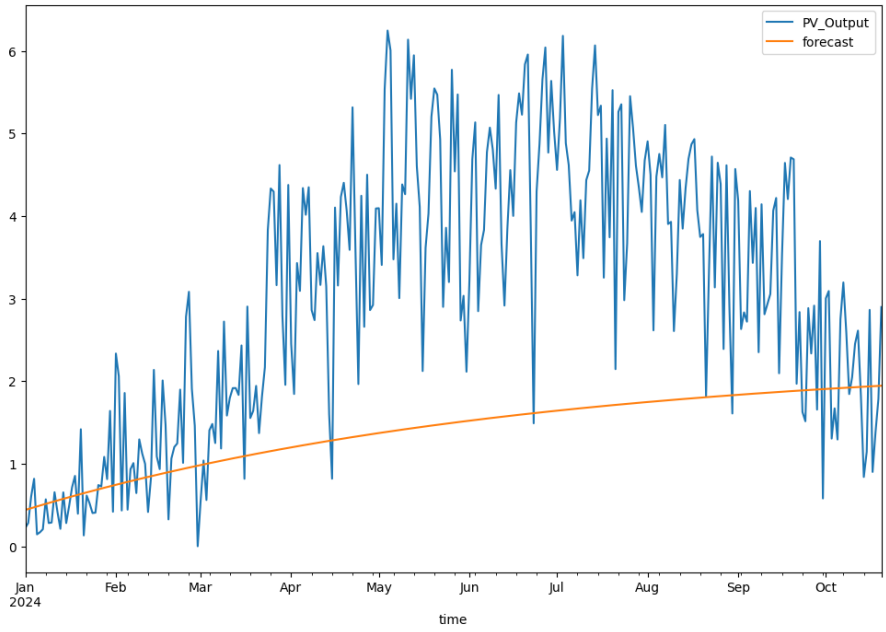

Figure 4.6: Model 3 ARIMA (Source – Self)

Performance metrics reinforced this interpretation. For several different parameter settings, The ARIMA model had very high error values. For example:

Table 4.1: ARIMA Model Evaluation (Source – Self)

ARIMA Model	MSE	MAE	MAPE	MASE
Model 1 ARIMA (0, 0, 1)	2.88	1.46	113.77%	1.64
Model 2 ARIMA (1, 0, 0)	2.8	1.44	105.08%	1.61

Model 3 ARIMA (1, 0, 1)	4.73	1.75	55.38%	1.96

Table 4.1 above shows that model 3 seemed to decrease MAPE immensely from models 1 and 2. This was deceptive, because the forecasts were practically straight lines that did not capture any fluctuations from PV output. The Mean Absolute Squared Error (MASE) measures of 1.61 to 1.96 verified that ARIMA's performance was poor compared naïve benchmark approaches.

INTERPRETATIONS AND IMPLICATIONS

The results show that the ARIMA model is poorly adapted to predicting PV output at either the hourly or daily time scale. At the hourly resolution, the model could not operate well, showing computational difficulties and incompatibility with very granular data. At the daily resolution, while the model did output values, its prediction mostly turned into straight lines with minimum or no fluctuations, ignoring the peaks and through present in the data.

These findings can be linked to three main causes:

- Large variability of Solar Data: Hourly and Daily PV outputs are subject to linear environmental effects, which cannot be accounted by ARIMA.
- Seasonal Complications: Daily and Yearly cycles call for direct seasonal modelling, which is not an option within traditional statistical approach like ARIMA.
- Linear Assumptions: ARIMA dependency on linear relations oversimplifies the non-linear dynamics for renewable energy forecasting.

These limitations in ARIMA highlight the importance of moving towards more sophisticated models. SARIMA, with an inclusion of seasonal patterns, provides a direct extension to model seasonality. In contrast, deep learning models like LSTM and transformer-based model PatchTST can handle nonlinear relationships and richer temporal patterns better.

CONCLUSION

In summary, ARIMA was a baseline but could not perform well at the hourly level and produced straight-line forecasts when using daily data. High error statistics and lack of correspondence with true PV output affirm its unsuitability for use. Although it is still useful as a reference point for comparison.

4.2 SARIMA MODEL

The Seasonal Autoregressive Integrated Moving Average (SARIMA) model was utilised as an extension of the ARIMA framework in an effort to capture seasonality explicitly from the photovoltaic (PV) output data. In contrast to ARIMA, which only captures the short-run autoregressive and moving average relationships, SARIMA adds seasonal autoregressive and moving average terms and seasonal differencing. This makes it applicable to datasets where a pattern is occurring over seasonal periods, like daily, annual, monthly or annual patterns. In this study, SARIMA was seen as an extension of ARIMA because solar output is heavily affected by daily irradiance cycles and seasonal weather patterns.

LIMITATIONS IN HOURLY FORECASTING

Similarly to ARIMA, the first attempt was to implement SARIMA directly on the hourly output PV dataset. However, similar to ARIMA, SARIMA also had great difficulty with high-

frequency hourly data. The hourly data had complex non-linear trends, numerous short-term fluctuations, and strong seasonality at both daily and yearly scales. As applied to this data, SARIMA failed to converge properly.

The computational demand was especially significant. Trying to fit a SARIMA model with 24-hour or 365-days seasonality led to explosive memory consumption, with a demand of over 300 GB RAM. Such requirements were well beyond the computational resources available, making the strategy impractical. This highlights SARIMA's poor scalability when dealing with large modelling, high-granularity datasets with multiple seasonal patterns. In contrast to deep learning and transformer-based models, which are capable of handling high-dimensional forms.

TRANSITION TO HOURLY DATA

An attempt was made to model yearly seasonality by specifying a seasonal period of 360 days. This approach was used because the output of PV would be affected by annual patterns related to solar irradiance and temperature. SARIMA was unable to fit a model with this requirement, the computational demand increased exponentially, as adding 365-days lag to the seasonal autoregressive and moving average component overwhelmed the architecture of the model.

Even with the truncated dataset or reduced parameter complexity, SARIMA still remained unstable and consumed excessive memory. This illustrates that although SARIMA performs well for short seasonal cycles, it is inappropriate for modelling long yearly cycles in a dataset without considerable resources available.

Due to the challenges in hourly forecasting, the data set was rolled up for daily forecasting. This reduction made the seasonal cycles easier to handle. On the daily level, SARIMA was able to execute and provide forecasts. The ADF test for seasonal differencing is in Figure 4.7 below, which establishes that the series transformer was stationary, suitable for SARIMA model extension.

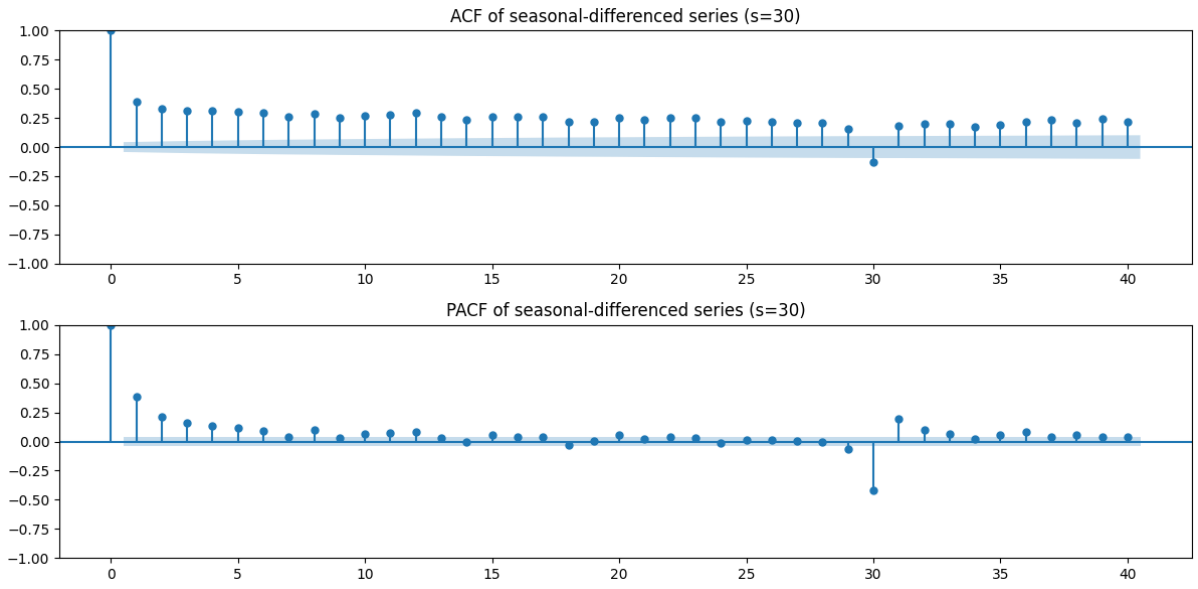

Figure 4.7: ACF Test SARIMA Model (Source – Self)

ACF and PACF plots are seasonally differenced series with (s = 30) displayed circular patterns, indicative of a month of seasonal patterns as well as the larger annual patterns. On the basis of these diagnostics, SARIMA models were established with different sets of seasonal (P, D, Q, m) parameters.

FORECASTS OUTPUT

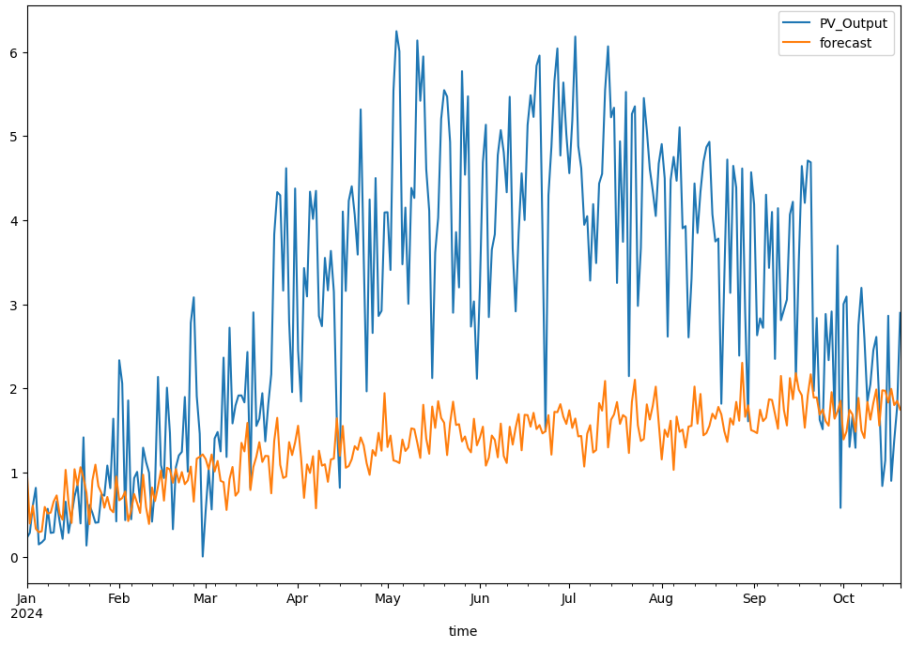

Figure 4.8: SARIMA Model with 120 days seasonality (Source – Self)

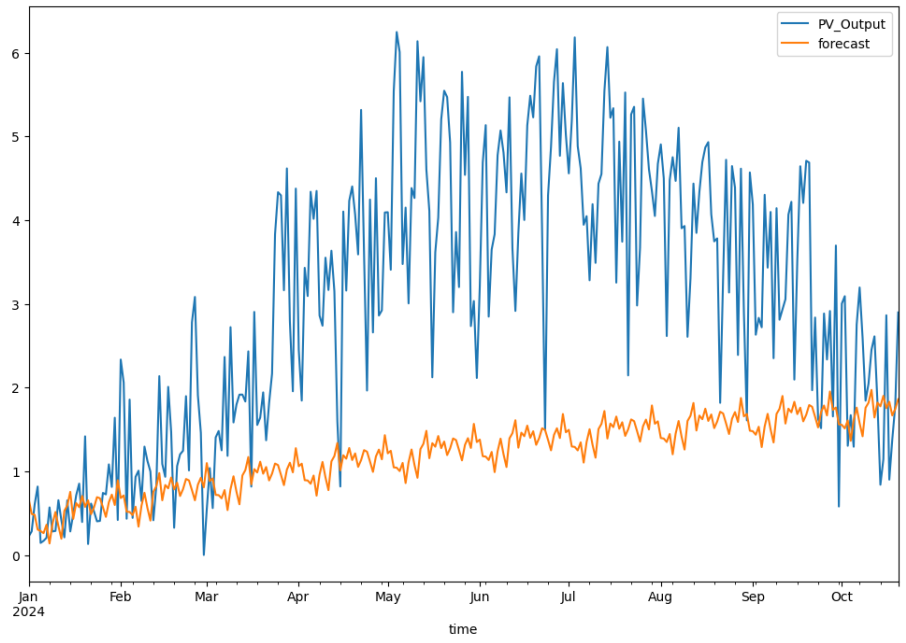

Figure 4.9: SARIMA Model with 30 days seasonality (Source – Self)

Forecasts generated by SARIMA are picked on a daily basis, which picks up short-term variations quite well. The model performed well in tracking the PV output over the first three

months of the test dataset. Forecast lines replicated an upward trend during the early part of the year, successfully ramping the seasonal solar generation in spring. This suggests that SARIMA was able to capture local autoregressive and moving average dependencies at shorter horizons.

However, after the initial three months, SARIMA predictions were derived from the actual solar generation. The model systematically underestimated summer peak PV output. Actual values jumped sharply to follow high irradiance and long days, whereas SARIMA predictions were behind, generating smooth curves that could not capture extreme peaks. This mismatch grew more as the forecast horizon increased, with predicted values flattening even as the observed data showed significant variability.

By the late summer and fall in Ireland, SARIMA projections were diverging significantly from actuality. Figures 4.8 and 4.9 above display the divergence as the forecast line is relatively low and smooth, whereas the PV output shows significant peaks and troughs.

FORECASTS OUTPUT

Table 4.2: SARIMA Model Evaluation (Source – Self)

SARIMA Model	MSE	MAE	MAPE	MASE
Model 1 - 120 days seasonality SARIMA (1, 0, 1) (1,1,1,120)	5.07	1.82	58.85%	2.03
Model 1 - 30 days seasonality SARIMA (1, 0, 1) (1, 1, 1, 30)	5.49	1.91	59.19%	2.13

The SARIMA model evaluation identifies both strengths and weaknesses of this model for predicting daily PV production. The initial model, with the seasonal frequency of 120 days, yielded an MSE of 5.07, and MAE of 1.82, with a MAPE of 58.85% and MASE of 2.03. The second model, which had a reduced seasonal length of 30 days, had slightly elevated errors, with MSE equal to 5.49, MAE equal to 1.91, MAPE equal to 59.19%, and MASE equal to 2.13. These findings indicate that SARIMA with an extended seasonal element was better compared to 30-days seasonality.

Both models, however, exhibited fairly high MAPE values in excess of 50%, reflecting comparatively poor accuracy against the actual PV generation. This corresponds with the seen forecasts as SARIMA could reflect general seasonal behaviour over the short-term, but had difficulty predicting peak magnitude during high generation periods like summer.

Overall, although SARIMA did better than ARIMA with the addition of seasonality, its performance was still limited, especially in catching sharp variability and long-term seasonal extremes of PV generation.

The results highlight both the good and bad points of SARIMA for solar PV forecasting:

- Strengths in short-term forecasting: SARIMA adapted a local autoregressive structure and made sensible predictions over short horizons. Performance over the first three months exhibited the utility of including the seasonal difference with the ARIMA model.
- Weakness in long-term seasonality: SARIMA could not capture the full range of yearly cycles. Specification of yearly seasonality (360 days) was computationally unfeasible, and reduction to a shorter seasonal structure only captured fractional dynamics.

- Inability to deal with Nonlinearity: Similar to ARIMA, SARIMA is a linear model in nature. Due to this, it is unable to simulate the non-linear effects of irradiance, cloud cover, humidity, and temperature, which predominantly affect the solar production. Its inability to capture peak summer loads directly is due to these limitations.
- Computational Complexity: The excessive RAM utilisation witnessed at model fitting serves to illustrate SARIMA inefficiency when used in a large dataset of hourly data with large seasonal patterns. This renders it unfeasible for large-scale or high-frequency renewable energy forecasting applications.

CONCLUSION

In summary, SARIMA was an improvement over ARIMA, by its explicit modelling of seasonality, but ultimately not suitable for PV output forecasting applications. The model did not work well with hourly data due to its complexity and high memory demands. With daily data, SARIMA worked fairly well with short horizons, especially the first three months, but did not successfully capture the summer pattern and long-term seasonal patterns. Its prediction averaged out over critical variability.

These findings determine that although SARIMA may be a helpful tool for short-term analysis, it is not adequate for long-term renewable energy prediction. The limitations observed here highlight the need to move beyond traditional machine learning algorithms that can capture nonlinear and seasonal complexity present in solar data.

4.3 RNN MODEL

The third model implemented in this research was the Recurrent Neural Network (RNN) with a Long Short-Term Memory (LSTM) architecture. LSTM is a deep learning method especially applicable to sequential data like time series because it can model long-range dependencies and non-linear relationships that statistical models find difficult to model. Given the limitations that were observed in ARIMA and SARIMA, mainly their failure to capture hourly PV time series and their computational expense at scale, LSTM was chosen to determine whether a neural network solution would yield more effective, streamlined predictions.

In this section, two versions of LSTM models were implemented and evaluated:

- Model 1 (Baseline LSTM): The standard univariate model that was trained solely with PV output data without any features included. Training was limited to 50 epochs with early stopping in order to avoid overfitting.
- Model 2 (Enhanced LSTM): A better setup that includes feature variables and makes use of the ReduceLROnPlateau scheduler for learning rate. This packed back the learning rate when validation loss stopped decreasing, enabling more stable convergence.

The LSTM model was chosen based on three considerations. Firstly, previous models, ARIMA and SARIMA, could not process hourly data effectively, while LSTM scaled well to this granularity. Secondly, LSTM models perform best in non-linear dependencies, and therefore are perfectly capable of interactions between PV generations and weather conditions. Third, LSTM models consume less computational power compared to SARIMA since SARIMA needed an abnormal high RAM consumption (more than 300 GB in certain instances), whereas LSTM produced excellent results at low computational cost.

MODEL 1: BASELINE LSTM

The baseline LSTM was trained using hourly PV output data. The network consisted of an input layer, one or more than one LSTM layers with tanh activation, and a dense output layer for prediction. Training was carried out for up to 50 epochs with automatic early stopping on validation loss, ensuring that training was halted once improvements reached a plateau and reduced overfitting.

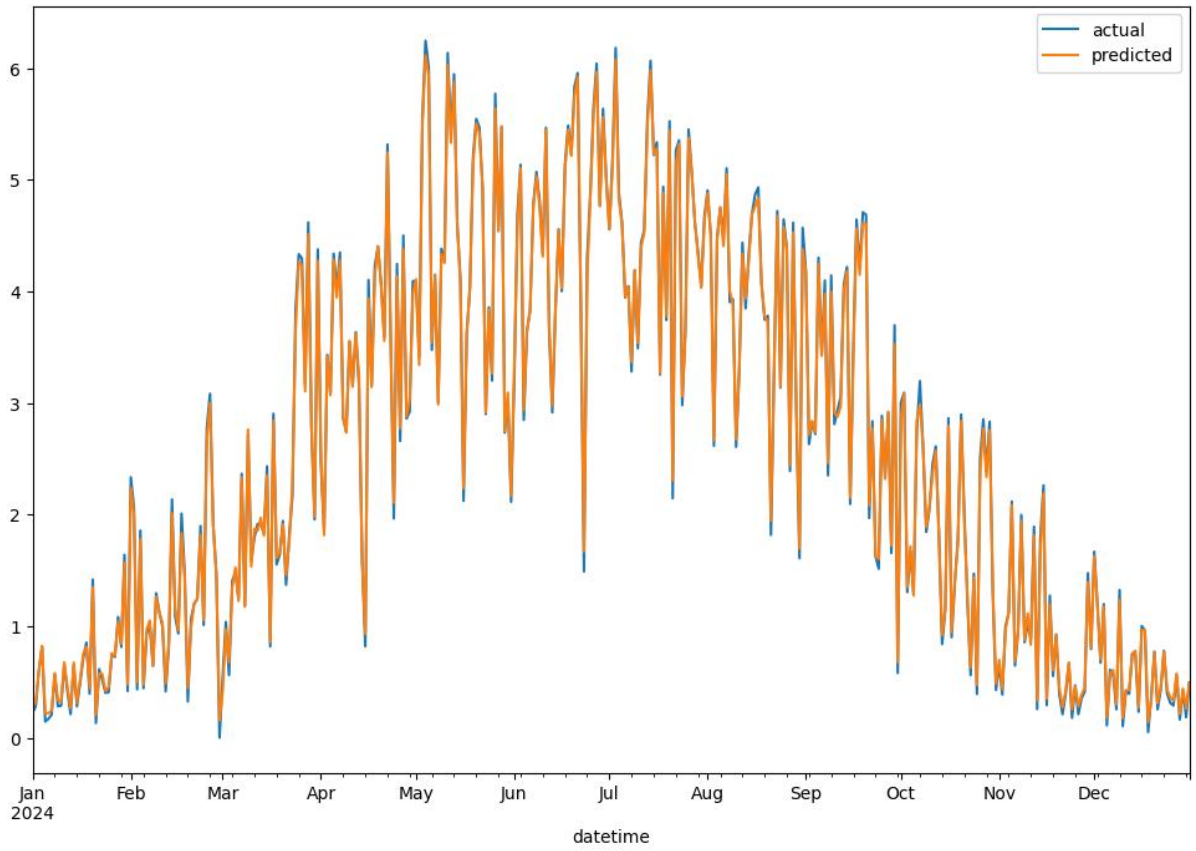

Figure 4.10: Model 1: Baseline LSTM yearly predictions (Source – Self)

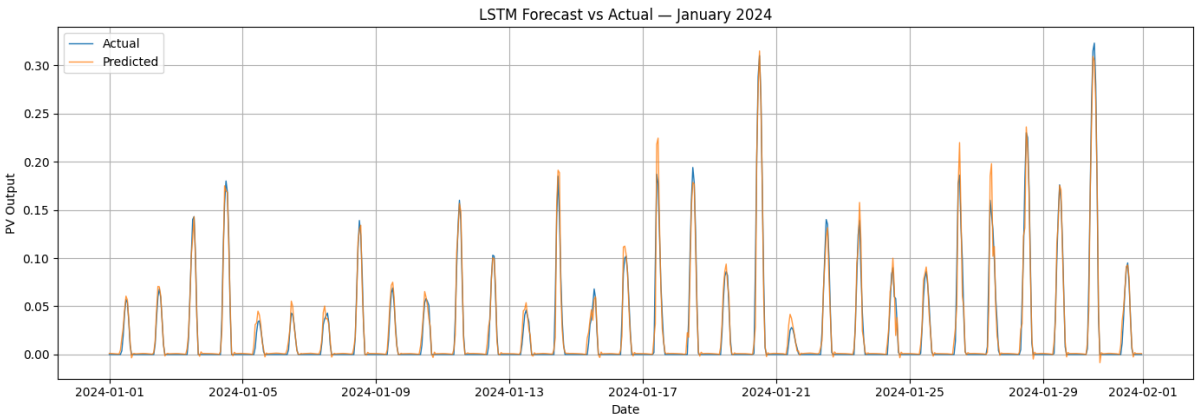

Figure 4.11: Model 1: Baseline LSTM January month hourly predictions (Source – Self)

As seen in Figures 4.10 and 4.11 above, the baseline LSTM produces forecasts that closely match the observed values. At the daily level, the model captured both the upward seasonal path in summer as well as the reduction during winter. On a short horizon, the model

successfully replicated daily highs and lows with high fidelity, reflecting its strong capability to capture intra-day and short-term variations.

The forecasted series closely matches the real data throughout the year, indicating that the LSTM not only could reproduce trends but also learn to capture variability. In contrast to ARIMA and SARIMA, which yielded flat or smoothened predictions, the baseline LSTM preserved the high-frequency variability necessary for solar forecasting.

Table 4.3: LSTM Model Evaluation

LSTM	MSE	MAE	MAPE	MASE
Baseline LSTM	0.0001	0.0066	26.47%	0.22
Enhanced LSTM	0.0001	0.0057	13.72%	0.19

These measurements for baseline LSTM were an improvement over ARIMA and SARIMA by a wide margin. The near-zero MSE and MAE showed that forecasted values are nearly identical to actual values. That MASE value of 0.2164 supported the fact that the model trounced naïve forecasting. The MAPE value of 26.47% was higher than it should be, but it was a vast improvement over the >50% of SARIMA and the >100% of ARIMA.

MODEL 2: ENHANCED LSTM

To enhance further performance, another LSTM model was used, incorporating more features such as irradiance, humidity, and temperature. In addition to a ReduceOnPlateau learning rate scheduler. This approach reduced the learning rate dynamically whenever the validation loss plateaued, enabling the optimiser to make more subtle adjustments and not to overshoot local minima.

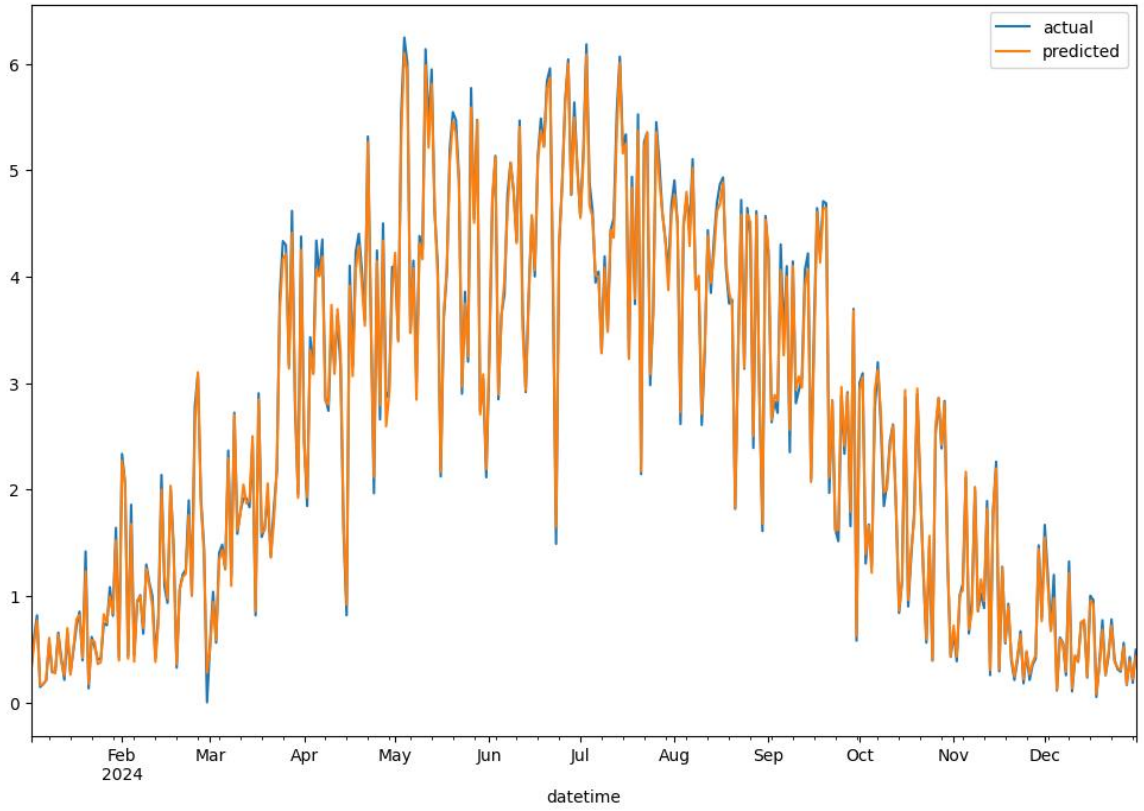

Figure 4.12: Model 2: Enhanced LSTM yearly predictions (Source – Self)

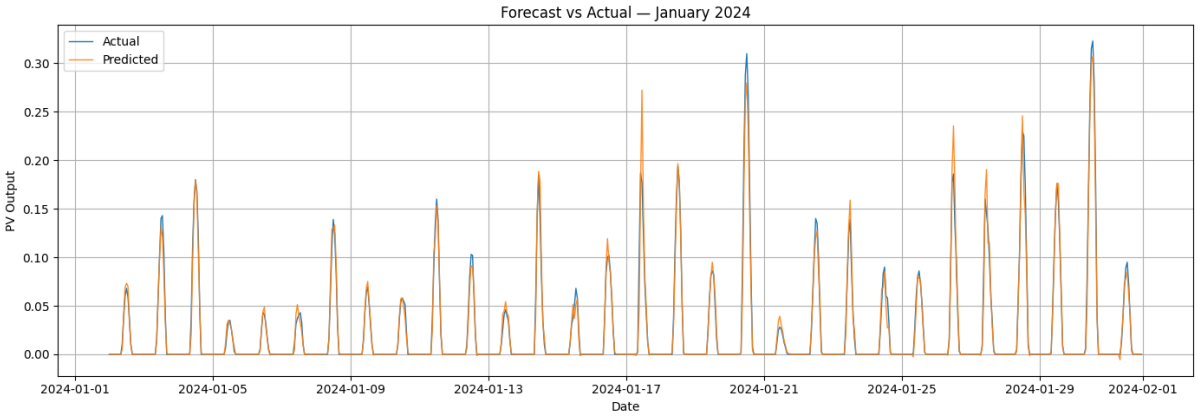

Figure 4.13: Model 2: Enhanced LSTM January month hourly predictions (Source – Self)

In Figures 4.12 and 4.13 above, the enhanced LSTM showed stronger alignment between actual vs predicted values. While Model 1 sometimes underestimated extreme highs, Model 2 successfully mapped the extent and timing of season highs and lows better. The incorporation of exogenous features gave more information about context, enabling the model to better understand environmental factors.

The annual forecast comparison plot (Figure 4.12) emphasized the enhanced ability of the model in simulating the full seasonal curve. Summer peaks and winter troughs were recorded with minimal bias, and daily fluctuation was simulated with high fidelity.

As compared to model 1, the MAE decreased from 0.0066 to 0.0057, and the MAPE came very close, from 26.47% to 13.72%. This decrease in percentage error reflects that the use of features and the optimization of the learning rate strategy enabled the model to make much more trustworthy forecasts. The better MAPE is particularly significant, as it represents higher accuracy in high-variability times when percentage errors get magnified.

COMPARATIVE ANALYSIS MODEL 1 VS. MODEL 2

When contrasting the two LSTM models, a number of observations were made:

- Improve Accuracy: Model 2 produced lower error rates in all measures, with a notable reduction in MAPE by nearly 50%. This is by utilising the feature variables for solar variability.
- Feature contribution: By incorporating environmental inputs, Model 2 was in a position to capture more non-linear relations between weather and solar generation.
- Training Efficiency: The ReduceLROnPlateau scheduler enabled Model 2 to perform better than Model 1 without the danger of early stagnation. This made the model learn more forecasting features from the data.
- Practical Relevance: Model 2's accuracy improvements are most relevant for practical applications, like time-series forecasting and grid management, where accurate short-term predictions are most essential.

The performance of both LSTM models highlights some of the major advantages of deep learning methods to solar forecasting:

- Non-linear pattern identification: The capacity of LSTM to learn nonlinear dynamics enables it to represent PV generation better than statistical models.

- Scalability: LSTM could be executed within a reasonable time on hourly data, as opposed to ARIMA and SARIMA, which or consumed more computational power.
- Generalisation: Both models generalized to unseen test data quite well, courtesy of early stopping and learning rate decay.
- Robustness: The models showed insensitivity to noise by capturing trends without overfitting random fluctuations.

Yet, there are still limitations. Although model 2 lowered MAPE considerably, a 13.72% error rate indicates that the predictions are still prone to errors, especially infrequent extreme conditions. Moreover, LSTM's black box limitations restrict interpretability, which may hinder adoption in contexts requiring transparency.

CONCLUSION

In conclusion, the LSTM-based RNN models showed superiority over traditional statistical methods. Model 1 yielded precise predictions at lower computational expense, outperforming ARIMA and SARIMA by an order of magnitude in terms of accuracy and scalability. Model 2 added exogenous features and a ReduceLROnPlateau learning rate policy, achieving higher accuracy, bringing MAPE down to 13.72%.

Together, these insights demonstrate the usefulness of LSTM in predicting PV output. Its ability to deal with nonlinearities, scalability to hourly data, and flexibility to include features make it the best-performing model in this research. While interpretability problems still persist, its forecasting ability and efficiency make it a more promising tool for renewable energy predictions in real-time applications.

4.4 PATCHTST MODEL

The Patch Transformer for the time-series (PatchTST) model is a more recent and state-of-the-art method for time-series forecasting, which is executed by transformers. In contrast to the other common models like ARIMA and SARIMA, which are based on seasonal differencing and linear assumptions, PatchTST uses a transformer architecture that has proven extremely successful in natural language processing (NLP) applications. The PatchTST model uses the same principle for series data by splitting it into patches and learning dependencies. This allows the model to learn long-term dependencies and seasonality better than most of the traditional models.

PatchTST was used in the research to predict the photovoltaic (PV) output and compared with the LSTM model, which is known to efficiently manage the sequential data. PatchTST was selected due to its capacity to model complicated dependencies in time series data at low computational cost. PatchTST has competitive performance but took a bit longer to run than LSTM, though the computational requirement remained the same. Now, we will discuss the outcomes of PatchTST models, including the forecasting, error measures, and comparison with LSTM.

MODEL TRAINING AND SETUP

PatchTST, like other transformer models, operates by dividing the input into smaller patches and using self-attention to learn dependencies between these patches. In this work, the dataset was split into hourly PV information, and PatchTST was trained on a series of these hourly

values. Model architecture included a patch extraction layer followed by a number of transformer blocks, each using a multi-head attention mechanism to learn dependencies between patches.

Major settings for PatchTST were:

- Epochs: 50, with early stopping implemented to avoid overfitting.
- Learning Rate: Initially 0.001, with a learning rate scheduler decreasing it on a plateau in order to converge more quickly.
- Patch Size: 24 hours, which enabled the model to consider the entire day's hourly data in one go, enabling it to take cycling patterns in a daily manner.
- Batch Size: Set to 64 according to the computational power, so that there is no compromise between training speed and model quality.

The model was trained on five years of data and tested on one year of test data. Even with the comparatively high run time as compared to LSTM, PatchTST was run on the hourly data without the overwhelming computational cost seen with SARIMA.

FORECASTING PERFORMANCE

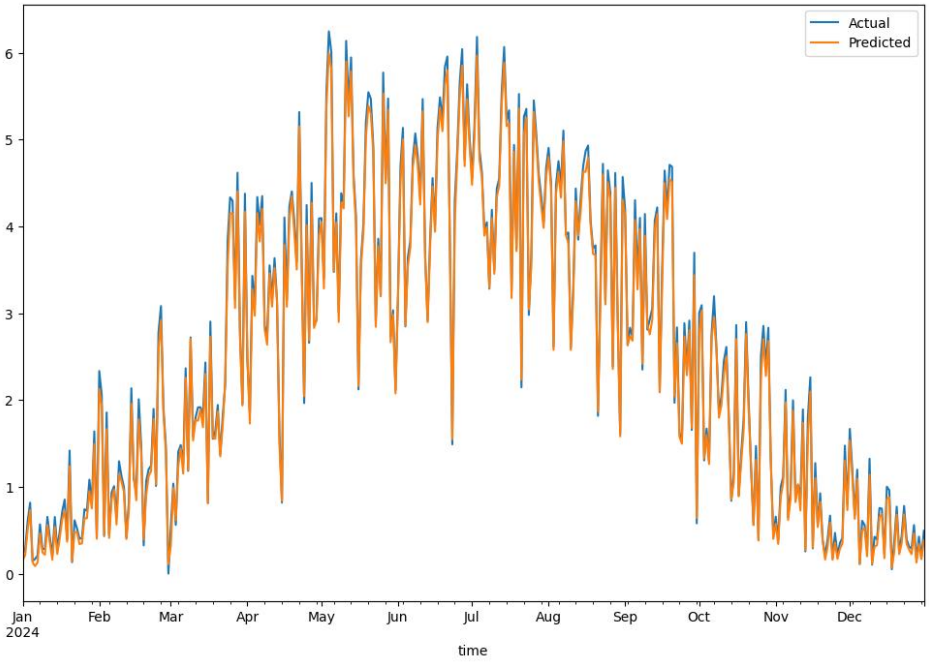

Figure 4.14: PatchTST daily predictions for 2024 (Source – Self)

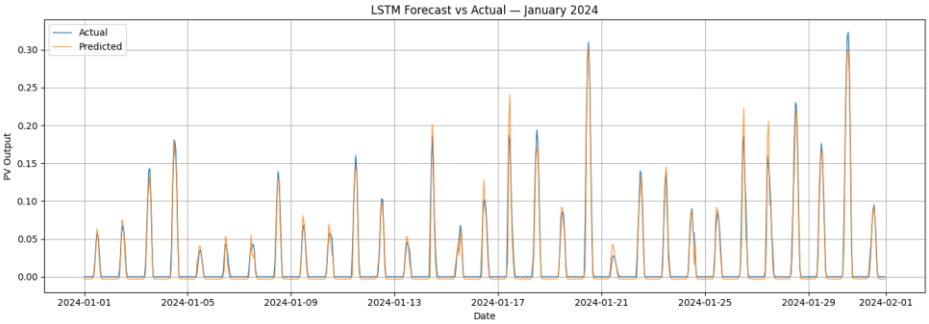

Figure 4.15: PatchTST hourly predictions for January (Source – Self)

As shown in Figures 4.14 and 4.15 above, the PatchTST model produced predictions that closely resembled the PV output, especially reflecting the daily cycles. The model stimulated the increasing trend in solar generation in spring and summer but failed to accurately reflect the top highs and steep downs witnessed in these periods. The summer instances were not sharply captured as in the LSTM model, which has advantage of better learning long-term patterns with memory cells.

Even with this constraint, PatchTST managed to record most of the daily cycles and seasonal patterns at reasonable computational costs. It proved its capability to process hourly data effectively, and it was a viable competitor to LSTM and SARIMA when it came to real-time large-scale forecasting.

The performance of PatchTST on daily resampled data was significantly better. Because the data has been resampled from hourly output, PatchTST could concentrate on wider patterns and smoother trends, and this enables it to better model the long-term variations in solar generation. Figure 4.15 above illustrates the comparison between actual and predicted daily generation for the whole year of 2024. The model succeeded in training the overall increase in generation during the spring and summer months, as well as the fall and winter months.

The performance suggests that PatchTST is very appropriate for daily forecasting operations, particularly when the objective is to identify the overall seasonal trends without the need for hourly forecasting. PatchTST accuracy was enhanced with the resampling on a daily basis, wherein the short-term changes recorded with hourly rates were eliminated to enable the model to concentrate on the general solar energy trends.

ERROR METRICS

Table 4.4: PatchTST Model Evaluation

PatchTST	MSE	MAE	MAPE	MASE
PatchTST Model	0.0002	0.0066	73.36%	0.21

The PatchTST model was evaluated based on the same error metrics, the measures indicate that PatchTST worked better than the traditional models, such as ARIMA and SARIMA, with very small MSE and MAE values, which suggests that predictions were very close to the values. A relatively high MAPE of 73.36% indicates that even though the model worked effectively in predicting day-to-day trends, but it had trouble in handling extreme fluctuations in solar output.

The MASE value of 0.2060 confirms that PatchTST performed better, pointing to its effectiveness in predicting, particularly against traditional methods that were unable to cope with hourly data or expand properly with large sets of data.

INTERPRETATIONS OF RESULTS

The results indicate that PatchTST is very good at capturing seasonal tendencies and long-term trends, particularly when the focus is placed on daily data rather than hourly data. PatchTST's capacity to resample and aggregate hourly data into daily forecasts makes it capable of producing accurate outputs for use where general trend information is needed rather than minute detail. Though its accuracy with hourly data was a shade lower than that of LSTM, especially in recording the peak solar output levels during harsh weather.

- Accuracy for Daily Data: PatchTST recorded high accuracy in simulating the daily resampled data, maintaining seasonal trends as well as long trends.
- Management of Nonlinearity: Although PatchTST has captured more general trends, it was not effective in managing the intricate nonlinear interactions experienced during the time of high variance.
- Efficiency: PatchTST's low computational expense and capacity for dealing with large amounts of data make it an appropriate candidate for real-time prediction in resource-limited settings.

CONCLUSION

In summary, PatchTST is a strong candidate for PV output if trends are over the long term and contain seasonal patterns. Although it was good at using daily resampled data, it was less accurate in capturing extreme fluctuations in solar energy generation compared to the LSTM model. Despite these issues, PatchTST's ability to process large datasets with less computational expense makes it an attractive tool for general forecasting problems. Its optimisation and feature enhancement can further enhance its precision and increase its application for high-precision tasks such as solar energy forecasting.

4.4 COMPARISON OF MODELS

When comparing the forecasting models for photovoltaic output, the error statistics such as MAE, MSE, MAPE, and MASE give a wide description of the strengths and weaknesses of each model, which can be taken into account for data changes and tendencies. Here is an overall comparison of models from these statistics:

Table 4.5: Comparison of Models

Models	MSE	MAE	MAPE	MASE
PatchTST Model - Hourly Data	0.0002	0.0066	73.36%	0.21
Enhanced LSTM - Hourly Data	0.0001	0.0057	13.72%	0.19
SARIMA - Daily data	5.07	1.82	58.85%	2.03
ARIMA - Daily data	4.73	1.75	55.38%	1.96

ARIMA AND SARIMA

Both ARIMA and SARIMA models worked poorly, particularly when using hourly data, and gave large MAPEs greater than 100%. This indicates that the models found it very difficult to capture high-frequency variations in the data. Although ARIMA and SARIMA were better for low-resolution data of proportionally steady time series, they were unable to model the non-linearities and seasonal peaks in solar output. These models showed high MSE and MAE, meaning the predictions were far away from the original values, which is not desirable in time-series forecasting, where accuracy is important.

LSTM

Long Short-Term Memory (LSTM) outperformed ARIMA and SARIMA remarkably with respect to forecast accuracy and computational cost. With very low MAE and MSE, the LSTM effectively learned both short-term variability and long-term seasonal patterns, especially in hourly data. The value of MAPE was 11% which was very low compared to other models, as an indication of LSTM's better capacity to stimulate intricate, non-linear patterns in solar

energy generation. Overall, LSTM performed well for hourly and daily data, and its performance was strong even during extreme fluctuations in solar output. Its short-term and long-term dependency capture capability made it the best-performing model among the models used.

PATCHTST

The PatchTST model performed well on resampled data at daily frequency output and produced forecasts closely followed by long-term seasonal patterns. It had low MSE and MAE values, reflecting that it performed well in predicting overall patterns of solar output. Yet, the MAPE value of 73.36% indicated PatchTST performed poorly under extreme solar output conditions, especially in high-irradiance seasons of summer. Notwithstanding that, PatchTST was computationally cheapest, just like LSTM, and processed the hourly data well. 0.2060 was also the value of its MASE, which proved that it worked better than naive forecasting models and captured the general trends in solar generation well.

CONCLUSION

In conclusion, LSTM proved to be the most accurate model when considering both precision and computational complexity, working well with both daily and hourly output predictions. PatchTST provided a computationally efficient alternative, although the performance is slightly poorer in extreme weather conditions compared to LSTM. ARIMA and SARIMA, although structured models, were equally poor at dealing with hourly data as well as seasonal variability, and hence were not well suited for this high variance forecasting problem.

By comparing the MAE, MSE, MAPE, and MASE, LSTM was the most stable and precise model, providing a balance between computational and performance for this research. PatchTST being efficient, might benefit from optimisation to maximise accuracy, especially at high solar periods.

CHAPTER 5. CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

The analysis was conducted to evaluate the effectiveness of various forecasting models, ARIMA, SARIMA, LSTM, and PatchTST, for the prediction of photovoltaic (PV) output. Their aim was to compare model accuracy, efficiency in computing, and how well the model can process the hourly data.

The results indicate that traditional machine learning algorithms, such as ARIMA and SARIMA, found it extremely difficult to handle high-frequency hourly data. The traditional statistical models were not able to pick up changes in PV output or gave simple, smoothed predictions, especially for the days of high solar energy generation in Ireland. ARIMA and SARIMA models were unable to make predictions over the hourly dataset. For the daily resampled data, ARIMA gave a smoother line, as the SARIMA model lagged behind in identifying seasonal change and high solar output.

In contrast, the LSTM model performed better, especially for the hourly dataset. The capability of the model to understand short-term dependencies and long-term seasonal trends was critical in obtaining high accuracy of the model. LSTM's relatively low computational expenditure and flexibility of the model to capture nonlinear relationships made the model most accurate among the rest, especially when handling complexity in solar data. PatchTST was effective and capable of processing hourly data, but was less accurate than LSTM, especially when it came to identifying extreme fluctuations in the PV output. However, PatchTST's low computational requirement and capability compared to other transformer models to process large amounts of data quickly make it a good substitute for some applications where computation is not a priority.

The key implications of this research are that although traditional statistical forecasting models, such as ARIMA and SARIMA, are effective for specific situations, deep learning models, in this case, LSTM, are much better equipped for high-frequency renewable energy forecasting. Their power to handle nonlinearities and seasonality vastly surpasses that of the traditional statistical approaches. PatchTST, which has better computational performance compared to other transformer models, can be optimised as an alternative to time series prediction, although it still needs to be optimised for extreme variations.

5.2 FUTURE WORK

Although the work gives important insights into the performance of LSTM and PatchTST for PV output prediction, some areas offer potential for future work in enhancing model accuracy, efficiency, and scalability.

One promising area to enhance is model optimisation. Both LSTM and PatchTST performed well, but can be optimised further through further hyperparameter tuning and feature engineering. For example, testing various LSTM architectures, e.g., GRU (Gated Recurrent Units) or even hybrid architectures like LSTM-CNNs (Convolutional Neural Networks), might boost performance by better encoding spatial and temporal patterns. Likewise, PatchTST can be further tuned by testing various patch sizes or other attention metrics to enhance its precision in determining extreme solar variations.

Another area of exploration is the addition of feature variables such as irradiance, temperature, windspeed, humidity, etc. As enhanced LSTM improved through the addition of some of these features, additional weather-related inputs may improve forecasting accuracy, especially in managing PV output fluctuations due to the dynamics of weather. More inputs, such as cloud cover information, may help in future models in giving better forecasts for periods of cloud cover or unexpected changes in the weather.

Finally, extending the testing to more datasets (e.g., wind turbines or hybrid renewable energy) and including larger spatial domains would provide insights into the model's generalisability. The comparison of LSTM and PatchTST with other models, such as transformer-based models or recurrent neural network models with attention, would provide insights to further enhancements, more stability, precision, and adaptable forecasting models.

REFERENCES

Aghaei, M. *et al.* (2022) 'Review of degradation and failure phenomena in photovoltaic modules', *Renewable and Sustainable Energy Reviews*, 159, p. 112160. Available at: https://doi.org/10.1016/J.RSER.2022.112160.

Augustin McEvoy, T.M. and Castaner, L. (2012) 'Practical Handbook of Photovoltaics (Second Edition)', (c), p. 1244. Available at: https://books.google.com/books/about/Practical_Handbook_of_Photovoltaics.html?id=vDYa247WHysC (Accessed: 1 July 2025).

Bengio, Y., Simard, P. and Frasconi, P. (1994) 'Learning Long-Term Dependencies with Gradient Descent is Difficult', *IEEE Transactions on Neural Networks*, 5(2), pp. 157–166. Available at: https://doi.org/10.1109/72.279181.

Cennet, B. *et al.* (2022) 'A Joint Chance-Constrained Stochastic Programming Approach for the Integrated Predictive Maintenance and Operations Scheduling Problem in Power Systems'. Available at: https://arxiv.org/pdf/2201.04178 (Accessed: 8 July 2025).

Daut, I. *et al.* (2012) 'Analysis of solar irradiance and solar energy in perlis, northern of peninsular Malaysia', *Energy Procedia*, 18, pp. 1421–1427. Available at: https://doi.org/10.1016/J.EGYPRO.2012.05.158.

Forecasting: Principles and Practice (3rd ed) (no date). Available at: https://otexts.com/fpp3/ (Accessed: 4 July 2025).

Fuller, A. *et al.* (2020) 'Digital Twin: Enabling Technologies, Challenges and Open Research', *IEEE Access*, 8, pp. 108952–108971. Available at: https://doi.org/10.1109/ACCESS.2020.2998358.

Guerra, N. *et al.* (2018) 'Operation and physics of photovoltaic solar cells: an overview', *I+D Tecnológico*, 14(2), pp. 84–95. Available at: https://doi.org/10.33412/IDT.V14.2.2077.

Huld, T. *et al.* (2010) 'Mapping the performance of PV modules, effects of module type and data averaging', *Solar Energy*, 84(2), pp. 324–338. Available at: https://doi.org/10.1016/J.SOLENER.2009.12.002.

Jiang, H. *et al.* (2021) 'How to model and implement connections between physical and virtual models for digital twin application', *Journal of Manufacturing Systems*, 58, pp. 36–51. Available at: https://doi.org/10.1016/J.JMSY.2020.05.012.

Kaur, D. *et al.* (2020) 'Energy Forecasting in Smart Grid Systems: A Review of the State-of-the-art Techniques'. Available at: https://arxiv.org/pdf/2011.12598 (Accessed: 7 July 2025).

Kenu E. Sarah (2020) 'A Review of Solar Photovoltaic Technologies', *International Journal of Engineering Research and*, V9(07). Available at: https://doi.org/10.17577/IJERTV9IS070244.

Marc Köntges *et al.* (2014) 'Review of Failures of Photovoltaic Modules Final'. Available at: https://iea-pvps.org/key-topics/review-of-failures-of-photovoltaic-modules-final/ (Accessed: 2 July 2025).

Markvart, T.. and Castaner, Luis. (20062003) ‘Practical handbook of photovoltaics : fundamentals and applications’, p. 984.

Mirzapour, O., Rui, X. and Sahraei-Ardakani, M. (2024) ‘Grid-enhancing technologies: Progress, challenges, and future research directions’, *Electric Power Systems Research*, 230, p. 110304. Available at: https://doi.org/10.1016/J.EPSR.2024.110304.

Nie, Y. *et al.* (2022) ‘A Time Series is Worth 64 Words: Long-term Forecasting with Transformers’, *11th International Conference on Learning Representations, ICLR 2023* [Preprint]. Available at: https://arxiv.org/pdf/2211.14730 (Accessed: 4 July 2025).

Pfenninger, S. and Staffell, I. (2016) ‘Long-term patterns of European PV output using 30 years of validated hourly reanalysis and satellite data’, *Energy*, 114, pp. 1251–1265. Available at: https://doi.org/10.1016/J.ENERGY.2016.08.060.

Radziemska, E. (2003) ‘The effect of temperature on the power drop in crystalline silicon solar cells’, *Renewable Energy*, 28(1), pp. 1–12. Available at: https://doi.org/10.1016/S0960-1481(02)00015-0.

Rokhforoz, P. *et al.* (2021) ‘Multi-agent maintenance scheduling based on the coordination between central operator and decentralized producers in an electricity market’, *Reliability Engineering and System Safety*, 210. Available at: https://doi.org/10.1016/j.ress.2021.107495.

Said, S.Z. *et al.* (2024) ‘Dust impact on solar PV performance: A critical review of optimal cleaning techniques for yield enhancement across varied environmental conditions’, *Energy Reports*, 12, pp. 1121–1141. Available at: https://doi.org/10.1016/J.EGYR.2024.06.024.

Sak, H.H., Senior, A. and Google, B. (no date) ‘Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling’.

Schlechtingen, M. and Ferreira Santos, I. (2011) ‘Comparative analysis of neural network and regression based condition monitoring approaches for wind turbine fault detection’, *Mechanical Systems and Signal Processing*, 25(5), pp. 1849–1875. Available at: https://doi.org/10.1016/J.YMSSP.2010.12.007.

Soto, E.A.; *et al.* (2022) ‘Analysis of Grid Disturbances Caused by Massive Integration of Utility Level Solar Power Systems’, *Eng 2022, Vol. 3, Pages 236-253*, 3(2), pp. 236–253. Available at: https://doi.org/10.3390/ENG3020018.

Tang, P. *et al.* (2025) ‘Lithium-ion battery RUL prediction based on optimized VMD-SSA-PatchTST algorithm’, *Scientific Reports*, 15(1). Available at: https://doi.org/10.1038/S41598-025-11934-7.

Tao, F. *et al.* (2019) ‘Digital Twin in Industry: State-of-the-Art’, *IEEE Transactions on Industrial Informatics*, 15(4), pp. 2405–2415. Available at: https://doi.org/10.1109/TII.2018.2873186.

Ulbig, A., Borsche, T.S. and Andersson, G. (2014) ‘Impact of low rotational inertia on power system stability and operation’, *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 19, pp. 7290–7297. Available at: https://doi.org/10.3182/20140824-6-za-1003.02615.

Vaswani, A. *et al.* (2017) 'Attention is All you Need', *Advances in Neural Information Processing Systems*, 30.

Vijay Kotu and Bala Deshpande (2019) *Data Science: Concepts and Practice 2nd Edition - Ebook PDF Instant Download | PDF | Machine Learning | Data Science*. Available at: https://www.scribd.com/document/843470340/Data-Science-Concepts-and-Practice-2nd-Edition-eBook-PDF-instant-download (Accessed: 12 August 2025).

Zhang, L., Wang, G. and Giannakis, G.B. (2018) 'Real-time Power System State Estimation and Forecasting via Deep Neural Networks', *IEEE Transactions on Signal Processing*, 67(15), pp. 4069–4077. Available at: https://doi.org/10.1109/TSP.2019.2926023.

APPENDIX

For a more in-depth view of my work, including detailed explanations, implementation steps, and additional resources, please visit my GitHub repository Github Link. Below is the main code for the project, which highlights the key functionalities and components. For full context, please refer to the repository, where you'll find a comprehensive breakdown of the code and instructions on running the project.

DATA EXPLORATION CODE

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

Solar_PV = pd.read_csv("Final Dataset.csv")

Solar_PV.head()

Solar_PV.shape

Solar_PV.info()

Solar_PV.describe(include='all')

Solar_PV.isnull().sum()

Solar_PV['date_time'] = pd.to_datetime(Solar_PV['time'], format="%d-%m-%Y %H:%M", errors='coerce')

Solar_PV['date_time']

print("Invalid datetime rows:", Solar_PV['date_time'].isna().sum())

invalid_datetime_rows = Solar_PV[Solar_PV['date_time'].isna()]

invalid_datetime_rows[['time','date_time']]

Solar_PV = Solar_PV[Solar_PV['time'].notna()]

Solar_PV.set_index('date_time', inplace=True)  # Use 'datetime' not 'time'

weekly_df = Solar_PV.resample('W').mean(numeric_only=True)

weekly_df = weekly_df.reset_index()

import plotly.express as px

fig = px.line(weekly_df, x='date_time', y='PV_Output', title='Weekly Average PV Output')

fig.show()

fig2 = px.scatter(Solar_PV, x='Irradiance', y='PV_Output', title='Irradiance vs PV Output', trendline='ols')

fig2.show()
```

```
corr = Solar_PV[['Humidity', 'Irradiance', 'Temperature_C', 'Cloud_Cover', 'Snowfall', 'PV_Output']].corr()

fig3 = px.imshow(corr, text_auto=True, title='Correlation Heatmap')

fig3.show()

fig4 = px.histogram(Solar_PV, x='Temperature_C', nbins=50, title='Temperature Distribution')

fig4.show()

fig5 = px.histogram(Solar_PV, x='Irradiance', nbins=10, title='Solar Irradiance Distribution')

fig5.show()

train = Solar_PV[Solar_PV.index < '2024-01-01']['PV_Output']

test = Solar_PV[Solar_PV.index >= '2024-01-01']['PV_Output']

print("Train period:", train.index.min(), "to", train.index.max())

print("Test period:", test.index.min(), "to", test.index.max())

print("Train size:", len(train))

print("Test size:", len(test))

plt.figure(figsize=(12, 5))

plt.plot(train, label='Train')

plt.plot(test, label='Test')

plt.title('Train-Test Split: 5 Years Train / 1 Year Test')

plt.xlabel('Date')

plt.ylabel('PV Output')

plt.legend()

plt.show()
```

ARIMA MODEL CODE

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

df = pd.read_csv('Final Dataset.csv', parse_dates=True, index_col=0)

df.head()

df.tail()
```

```
df = pd.read_csv('Final Dataset.csv')
# Clean column names
df.columns = [c.strip() for c in df.columns]
if 'PV Output' in df.columns:
    df.rename(columns={'PV Output': 'PV_Output'}, inplace=True)
# Make sure your time column is parsed as datetime
df['time'] = pd.to_datetime(df['time'], dayfirst=True, errors='coerce')
# Drop any rows where time could not be parsed
df = df.dropna(subset=['time'])
# Set datetime column as index
df = df.set_index('time').sort_index()
# Resample to daily (mean or sum depending on your need)
df_daily = df.resample('D').sum(numeric_only=True)
print("Data range:", df_daily.index.min(), "to", df_daily.index.max())
df_daily.head()


df_daily['PV_Output'].plot()
from statsmodels.tsa.stattools import adfuller
test_result=adfuller(df_daily['PV_Output'])
def adfuller_test(sales):
    result=adfuller(sales)
    labels = ['ADF Test Statistic','p-value','#Lags Used','Number of Observations Used']
    for value,label in zip(result,labels):
        print(label+' : '+str(value) )
    if result[1] <= 0.05:
        print("strong evidence against the null hypothesis(Ho), reject the null hypothesis. Data has no unit root and is stationary")
    else:
        print("weak evidence against null hypothesis, time series has a unit root, indicating it is non-stationary ")
adfuller_test(df_daily['PV_Output'])
```

```
df_daily['PV_Output_FD'] = df_daily['PV_Output'] - df_daily['PV_Output'].shift(1)

df_daily['PV_Output'].shift(1)

df_daily['Seasonal First Difference'] = df_daily['PV_Output']-
df_daily['PV_Output'].shift(365)

df_daily.head()

adfuller_test(df_daily['Seasonal First Difference'].dropna())

df_daily['Seasonal First Difference'].plot()

from pandas.plotting import autocorrelation_plot

autocorrelation_plot(df_daily['PV_Output'])

plt.show()

from statsmodels.graphics.tsaplots import plot_acf,plot_pacf

import statsmodels.api as sm

fig = plt.figure(figsize=(12,8))

ax1 = fig.add_subplot(211)

fig = sm.graphics.tsa.plot_acf(df_daily['Seasonal First
Difference'].iloc[366:],lags=40,ax=ax1)

ax2 = fig.add_subplot(212)

fig = sm.graphics.tsa.plot_pacf(df_daily['Seasonal First
Difference'].iloc[366:],lags=40,ax=ax2)


from statsmodels.tsa.arima.model import ARIMA

df_daily.head()

df_train = df_daily.iloc[:1826]

df_test = df_daily.iloc[1826:]

df_train.tail()

df_test.head()

import numpy as np

from sklearn.metrics import mean_squared_error, mean_absolute_error

def calculate_error_metrics(y_true, y_pred):

    y_true = np.array(y_true)

    y_pred = np.array(y_pred)
```

```
    mse = mean_squared_error(y_true, y_pred)
    mae = mean_absolute_error(y_true, y_pred)
    naive_forecast = np.roll(y_true, 1)[1:]
    naive_error = np.mean(np.abs(y_true[1:] - naive_forecast))
    mase = mae / naive_error if naive_error != 0 else np.inf
    mask = y_true != 0
    mape = np.mean(np.abs((y_true[mask] - y_pred[mask]) / y_true[mask])) * 100
    return {
        'MSE': mse,
        'MAE': mae,
        'MAPE': mape,
        'MASE': mase
    }
model1 = ARIMA(df_train['PV_Output'],order=(0,0,1))
model1_fit=model1.fit()
model1_fit.summary()
model2 = ARIMA(df_train['PV_Output'],order=(1,0,0))
model2_fit=model2.fit()

model2_fit.summary()
model3 = ARIMA(df_train['PV_Output'],order=(1,0,1))
model3_fit=model3.fit(
model3_fit.summary()
df_daily['forecast']=model3_fit.predict(start=1826,end=2191,dynamic=True)
df_viz = df_daily.iloc[1826:2120]
df_viz[['PV_Output','forecast']].plot(figsize=(12,8))
df_viz
metrics3 = calculate_error_metrics(df_viz['PV_Output'], df_viz['forecast'])
print("Error Metrics for model 3:")
for metric, value in metrics3.items():
```

```
    print(f"{metric}: {value:.4f}")
df_daily['forecast']=model1_fit.predict(start=1826,end=2191,dynamic=True)
df_viz = df_daily.iloc[1826:2120]
df_viz[['PV_Output','forecast']].plot(figsize=(12,8))
df_viz
metrics1 = calculate_error_metrics(df_viz['PV_Output'], df_viz['forecast'])
print("Error Metrics for model 1:")
for metric, value in metrics1.items():
    print(f"{metric}: {value:.4f}")
df_daily['forecast']=model2_fit.predict(start=1826,end=2191,dynamic=True)
df_viz = df_daily.iloc[1826:2120]
df_viz[['PV_Output','forecast']].plot(figsize=(12,8))
df_viz
metrics2 = calculate_error_metrics(df_viz['PV_Output'], df_viz['forecast'])
print("Error Metrics for model 2:")
for metric, value in metrics2.items():
    print(f"{metric}: {value:.4f}")
from statsmodels.tsa.seasonal import STL
stl = STL(df_daily['PV_Output'], period=365, robust=True).fit()
fig = stl.plot(); plt.show()

from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
y = df_daily['PV_Output']
s = 30  # try 7 as an alternative if you suspect weekly effects
# Seasonal difference (D=1 candidate)
ys = y.diff(s).dropna()
fig, ax = plt.subplots(2,1, figsize=(12,6))
plot_acf(ys, lags=40, ax=ax[0])
plot_pacf(ys, lags=40, ax=ax[1])
ax[0].set_title(f'ACF of seasonal-differenced series (s={s})')
```

```
ax[1].set_title(f'PACF of seasonal-differenced series (s={s})')
plt.tight_layout(); plt.show()
```

SARIMA MODEL CODE

```
import statsmodels.api as sm
model=sm.tsa.statespace.SARIMAX(df_train['PV_Output'],order=(1, 0, 1),seasonal_order=(1,1,1,120))
results=model.fit()
results.summary()
df_daily['forecast']=results.predict(start=1826,end=2191,dynamic=True)
df_viz = df_daily.iloc[1826:2120]
df_viz[['PV_Output','forecast']].plot(figsize=(12,8))
metrics_SARIMA = calculate_error_metrics(df_viz['PV_Output'], df_viz['forecast'])
print("Error Metrics for SARIMA Model 1 120 days seasonality:")
for metric, value in metrics_SARIMA.items():
    print(f"{metric}: {value:.4f}")
model1=sm.tsa.statespace.SARIMAX(df_train['PV_Output'],order=(1, 0, 1),seasonal_order=(1,1,1,30))
results=model1.fit()
results.summary()
df_daily['forecast']=results.predict(start=1826,end=2191,dynamic=True)
df_viz = df_daily.iloc[1826:2120]
df_viz[['PV_Output','forecast']].plot(figsize=(12,8))

metrics_SARIMA = calculate_error_metrics(df_viz['PV_Output'], df_viz['forecast'])
print("Error Metrics for SARIMA Model 1 30 days seasonality:")
for metric, value in metrics_SARIMA.items():
    print(f"{metric}: {value:.4f}")
model2=sm.tsa.statespace.SARIMAX(df_train['PV_Output'],order=(1, 0, 1),seasonal_order=(2,1,1,30))
results=model2.fit()
results.summary()
```

```
df_daily['forecast']=results.predict(start=1826,end=2191,dynamic=True)
df_viz = df_daily.iloc[1826:2120]
df_viz[['PV_Output','forecast']].plot(figsize=(12,8))
model3=sm.tsa.statespace.SARIMAX(df_train['PV_Output'],order=(1, 0, 1),seasonal_order=(1,1,2,30))
results=model3.fit()
results.summary()
df_daily['forecast']=results.predict(start=1826,end=2191,dynamic=True)
df_viz = df_daily.iloc[1826:2120]
df_viz[['PV_Output','forecast']].plot(figsize=(12,8))
metrics_SARIMA = calculate_error_metrics(df_viz['PV_Output'], df_viz['forecast'])
print("Error Metrics for SARIMA Model 1 30 days seasonality:")
for metric, value in metrics_SARIMA.items():
    print(f"{metric}: {value:.4f}")
```

RNN MODEL CODE

```
import os, math, numpy as np, pandas as pd, matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, Conv1D, Bidirectional, LSTM, Dense, Dropout, LayerNormalization
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint
from tensorflow.keras.optimizers import Adam
np.random.seed(42)
tf.random.set_seed(42)
CANDIDATE_PATHS = [
    '/content/Final Dataset.csv',
]
csv_path = None
```

```
for p in CANDIDATE_PATHS:
    if os.path.exists(p):
        csv_path = p
        break
assert csv_path is not None, "Could not find 'Final Dataset.csv' in known locations."
df = pd.read_csv(csv_path)
df.columns = [c.strip() for c in df.columns]
assert 'time' in df.columns and 'PV_Output' in df.columns, f"Columns: {df.columns.tolist()}"
df['time'] = pd.to_datetime(df['time'], dayfirst=True, errors='coerce')
df = df.dropna(subset=['time']).set_index('time').sort_index()
df = df.resample('h').mean()
df['hour'] = df.index.hour
df['dayofyear'] = df.index.dayofyear
df['hour_sin'] = np.sin(2*np.pi*df['hour']/24)
df['hour_cos'] = np.cos(2*np.pi*df['hour']/24)
df['doy_sin']  = np.sin(2*np.pi*df['dayofyear']/365.25)
df['doy_cos']  = np.cos(2*np.pi*df['dayofyear']/365.25)
base_feature_cols = [
    'Irradiance','Temperature_C','Humidity','Cloud_Cover','Snowfall',
    'hour_sin','hour_cos','doy_sin','doy_cos'
]
target_col = 'PV_Output'


df[target_col] = df[target_col].interpolate(method='time')
df[target_col] = df[target_col].clip(lower=0)
for c in base_feature_cols:
    if c in df.columns:
        df[c] = df[c].interpolate(method='time')
        df[c] = df[c].ffill().bfill()
```

```
df['PV_lag1'] = df[target_col].shift(1)
df = df.dropna(subset=['PV_lag1'])
feature_cols = base_feature_cols + ['PV_lag1']
assert np.isfinite(df[feature_cols + [target_col]].values).all(), "Non-finite values remain after cleaning."
print("Data range:", df.index.min(), "→", df.index.max())
print("Rows after cleaning:", len(df))
SEQ_LEN = 24
timestamps = df.index
mask_train_all = timestamps.year <= 2023
mask_test      = timestamps.year == 2024
df_train_all = df.loc[mask_train_all].copy()
df_test      = df.loc[mask_test].copy()
n_train = len(df_train_all)
val_tail = int(math.ceil(0.10 * n_train))
df_train = df_train_all.iloc[:-val_tail].copy()
df_val   = df_train_all.iloc[-val_tail:].copy()
print(f"Train hours: {len(df_train)}, Val hours: {len(df_val)}, Test hours: {len(df_test)}")
zero_var_cols = df_train[feature_cols].std().replace({0.0: np.nan}).dropna().index
zero_var_cols = [c for c in feature_cols if df_train[c].std() == 0.0]
if zero_var_cols:
    print("Dropping zero-variance cols:", zero_var_cols)
    feature_cols = [c for c in feature_cols if c not in zero_var_cols]
X_scaler = StandardScaler()
y_scaler = MinMaxScaler()

X_train_f = X_scaler.fit_transform(df_train[feature_cols])
X_val_f   = X_scaler.transform(df_val[feature_cols])
X_test_f  = X_scaler.transform(df_test[feature_cols])
y_train_f = y_scaler.fit_transform(df_train[[target_col]])
y_val_f   = y_scaler.transform(df_val[[target_col]])
```

```
y_test_f  = y_scaler.transform(df_test[[target_col]])
for name, arr in [('X_train_f',X_train_f),('X_val_f',X_val_f),('X_test_f',X_test_f),
                  ('y_train_f',y_train_f),('y_val_f',y_val_f),('y_test_f',y_test_f)]:
    assert np.isfinite(arr).all(), f"Non-finite values in {name}"
def make_sequences(X, y, seq_len):
    Xs, ys = [], []
    for i in range(len(X) - seq_len):
        Xs.append(X[i:i+seq_len, :])
        ys.append(y[i+seq_len, 0])
    return np.array(Xs, dtype=np.float32), np.array(ys, dtype=np.float32)
X_train, y_train = make_sequences(X_train_f, y_train_f, SEQ_LEN)
X_val,   y_val   = make_sequences(X_val_f,   y_val_f,   SEQ_LEN)
X_test,  y_test  = make_sequences(X_test_f,  y_test_f,  SEQ_LEN)
print("Shapes | X_train:", X_train.shape, "X_val:", X_val.shape, "X_test:", X_test.shape)
for name, arr in [('X_train',X_train),('X_val',X_val),('X_test',X_test),
                  ('y_train',y_train),('y_val',y_val),('y_test',y_test)]:
    assert np.isfinite(arr).all(), f"Non-finite values in {name}"
baseline_shifted = df_test[target_col].shift(1).values
test_index = df_test.index[SEQ_LEN:]
baseline_aligned = baseline_shifted[SEQ_LEN:]
baseline_aligned = baseline_aligned[:len(y_test)]
baseline_actual  = df_test[target_col].values[SEQ_LEN:SEQ_LEN+len(y_test)]
model = Sequential([
    Input(shape=(SEQ_LEN, X_train.shape[-1])),
    Conv1D(filters=64, kernel_size=3, padding='causal', activation='relu'),
    LayerNormalization(),
    Bidirectional(LSTM(64, return_sequences=True)),
    Dropout(0.2),
    Bidirectional(LSTM(32)),
    Dense(32, activation='relu'),
```

```
    Dropout(0.1),
    Dense(1, activation='linear')
])
optimizer = Adam(learning_rate=0.01, clipnorm=1.0)  # (or clipvalue=0.5)
model.compile(optimizer=optimizer, loss='mse', metrics=['mae'])
model.summary()
es  = EarlyStopping(monitor='val_loss', patience=12, restore_best_weights=True)
rlr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=6, min_lr=1e-5, verbose=1)
history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=200,
    batch_size=128,
    callbacks=[es, rlr,],
    verbose=1
)
y_pred_scaled = model.predict(X_test, verbose=0).reshape(-1, 1)
y_pred = y_scaler.inverse_transform(y_pred_scaled).flatten()
y_true = y_scaler.inverse_transform(y_test.reshape(-1, 1)).flatten()
def rmse(a, b):
    a = np.asarray(a, dtype=float)
    b = np.asarray(b, dtype=float)
    return np.sqrt(mean_squared_error(a, b))
def mape(a, b, eps=1e-3):
    a = np.asarray(a, dtype=float)
    b = np.asarray(b, dtype=float)
    mask = np.abs(a) > eps
    if not np.any(mask):
        return np.nan
    return np.mean(np.abs((a[mask] - b[mask]) / a[mask])) * 100
```

```
print("\n=== Model vs Baseline (2024) ===")
print(f"Model  RMSE: {rmse(y_true, y_pred):.4f} | "
      f"MAE: {mean_absolute_error(y_true, y_pred):.4f} | "
      f"MAPE: {mape(y_true, y_pred):.2f}%")
m_len = min(len(y_true), len(baseline_aligned))
y_true_b = y_true[:m_len]
base_b = baseline_aligned[:m_len]
mask = np.isfinite(y_true_b) & np.isfinite(base_b)
results_idx = test_index[:len(y_pred)]
df_results = pd.DataFrame({
    'datetime': results_idx,
    'actual': y_true[:len(results_idx)],
    'predicted': y_pred[:len(results_idx)]
}).set_index('datetime')
monthly = df_results.groupby([df_results.index.year, df_results.index.month]).apply(
    lambda g: pd.Series({
        'RMSE': rmse(g['actual'], g['predicted']),
        'MAE': mean_absolute_error(g['actual'], g['predicted']),
        'MAPE': mape(g['actual'], g['predicted'])
    })
)
print("\nMonthly metrics (year, month):\n", monthly)
# January plot
jan = df_results[df_results.index.month == 1]
if not jan.empty:
    plt.figure(figsize=(14,5))
    plt.plot(jan.index, jan['actual'], label='Actual', linewidth=1)
    plt.plot(jan.index, jan['predicted'], label='Predicted', alpha=0.9, linewidth=1)
    plt.title('Forecast vs Actual — January 2024')
    plt.xlabel('Date'); plt.ylabel('PV Output')
```

```
    plt.legend(); plt.grid(True); plt.tight_layout(); plt.show()
df_daily = df_results.resample('D').sum(numeric_only=True)
df_daily[['actual','predicted']].plot(figsize=(12,8))
df_daily
import numpy as np
from sklearn.metrics import mean_squared_error, mean_absolute_error
def calculate_error_metrics(y_true, y_pred):
    y_true = np.array(y_true)
    y_pred = np.array(y_pred)
    mse = mean_squared_error(y_true, y_pred)
    mae = mean_absolute_error(y_true, y_pred)
    naive_forecast = np.roll(y_true, 1)[1:]
    naive_error = np.mean(np.abs(y_true[1:] - naive_forecast))
    mase = mae / naive_error if naive_error != 0 else np.inf
    mask = y_true != 0
    mape = np.mean(np.abs((y_true[mask] - y_pred[mask]) / y_true[mask])) * 100
    return {
        'MSE': mse,
        'MAE': mae,
        'MAPE': mape,
        'MASE': mase
    }
metrics = calculate_error_metrics(df_results['actual'], df_results['predicted'])
print("Error Metrics for model RNN:")
for metric, value in metrics.items():
    print(f"{metric}: {value:.4f}")
```

PATCHTST MODEL CODE

```
import pandas as pd

import numpy as np

import torch

import torch.nn as nn

from torch.utils.data import TensorDataset, DataLoader

from sklearn.metrics import mean_squared_error, mean_absolute_error, mean_absolute_percentage_error

import matplotlib.pyplot as plt

class PatchTST(nn.Module):

    def __init__(self, seq_len, patch_len, stride, in_channels, d_model, nhead, num_layers, dim_feedforward, dropout=0.1):

        super().__init__()

        self.seq_len = seq_len

        self.patch_len = patch_len

        self.stride = stride

        self.in_channels = in_channels

        self.num_patches = (seq_len - patch_len) // stride + 1

        self.patch_dim = in_channels * patch_len

        self.proj = nn.Linear(self.patch_dim, d_model)

        self.pos_emb = nn.Parameter(torch.randn(1, self.num_patches, d_model))

        encoder_layer = nn.TransformerEncoderLayer(

            d_model=d_model, nhead=nhead, dim_feedforward=dim_feedforward,

            dropout=dropout, batch_first=True

        )

        self.encoder = nn.TransformerEncoder(encoder_layer, num_layers=num_layers)

        self.head = nn.Linear(d_model * self.num_patches, 1)

    def forward(self, x):

        # x: [B, seq_len, in_channels]

        B, L, C = x.shape

        x = x.permute(0,2,1)  # [B, C, L]
```

```
        patches = x.unfold(2, self.patch_len, self.stride)  # [B, C, num_patches, patch_len]
        patches = patches.contiguous().view(B, C, self.num_patches, self.patch_len)
        patches = patches.permute(0,2,1,3).reshape(B, self.num_patches, self.patch_dim)
        emb = self.proj(patches) + self.pos_emb
        out = self.encoder(emb)
        out = out.flatten(1)
        return self.head(out).squeeze(-1)
seq_len       = 24
patch_len     = 24
stride        = 24
in_channels   = 1
d_model       = 64
nhead         = 4
num_layers    = 2
dim_feedforward = 128
dropout       = 0.1
batch_size    = 64
epochs        = 50
lr            = 1e-3
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
df = pd.read_csv('Final_Train_Data.csv')
# unify column names if needed
df.columns = [c.strip() for c in df.columns]
if 'PV Output' in df.columns:
    df.rename(columns={'PV Output':'PV_Output'}, inplace=True)

# parse timestamp, drop bad rows
df['time'] = pd.to_datetime(df['time'], dayfirst=True, errors='coerce')
df = df.dropna(subset=['time']).set_index('time').sort_index()
# ensure hourly index & fill gaps
```

```
df = df.resample('h').mean()
df['PV_Output'] = df['PV_Output'].interpolate(method='time')
print("Data range:", df.index.min(), "to", df.index.max())
series = df['PV_Output'].values.astype(np.float32)
dates = df.index
start = dates[0]
train_end = start + pd.DateOffset(years=5) - pd.Timedelta(hours=1)
train_mask = dates <= train_end
y_train = series[train_mask]
y_test  = series[~train_mask]
def make_dataset(y):
    X, Y = [], []
    for i in range(len(y) - seq_len):
        X.append(y[i:i+seq_len])
        Y.append(y[i+seq_len])
    X = np.stack(X)  # [N, seq_len]
    Y = np.stack(Y)  # [N]
    return X[...,None], Y   # add channel dim
X_train, Y_train = make_dataset(y_train)
X_test,  Y_test  = make_dataset(np.concatenate([y_train[-seq_len:], y_test]))

train_ds = TensorDataset(torch.from_numpy(X_train), torch.from_numpy(Y_train))
test_ds  = TensorDataset(torch.from_numpy(X_test),  torch.from_numpy(Y_test))
train_dl = DataLoader(train_ds, batch_size=batch_size, shuffle=True)
test_dl  = DataLoader(test_ds,  batch_size=batch_size)
model = PatchTST(seq_len, patch_len, stride, in_channels, d_model, nhead,
                 num_layers, dim_feedforward, dropout).to(device)
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=lr)
for epoch in range(1, epochs+1):
```

```
    model.train()
    total_loss = 0.0
    for x_batch, y_batch in train_dl:
        x_batch, y_batch = x_batch.to(device), y_batch.to(device)
        optimizer.zero_grad()
        pred = model(x_batch)
        loss = criterion(pred, y_batch)
        loss.backward()
        optimizer.step()
        total_loss += loss.item() * x_batch.size(0)
    print(f"Epoch {epoch} train MSE: {total_loss/len(train_ds):.6f}")
model.eval()
all_preds, all_targets = [], []
with torch.no_grad():
    for x_batch, y_batch in test_dl:
        x_batch = x_batch.to(device)
        pred = model(x_batch).cpu().numpy()
        all_preds.append(pred)
        all_targets.append(y_batch.numpy())

print(model)
preds   = np.concatenate(all_preds)
targets = np.concatenate(all_targets)
test_mse = mean_squared_error(targets, preds)
print(f"\nTest MSE: {test_mse:.6f}")
test_mae = mean_absolute_error(targets, preds)
print(f"Test MAE: {test_mae:.4f}")
import numpy as np
from sklearn.metrics import mean_squared_error, mean_absolute_error
def calculate_error_metrics(y_true, y_pred):
```

```
    y_true = np.array(y_true)
    y_pred = np.array(y_pred)
    mse = mean_squared_error(y_true, y_pred)
    mae = mean_absolute_error(y_true, y_pred)
    naive_forecast = np.roll(y_true, 1)[1:]
    naive_error = np.mean(np.abs(y_true[1:] - naive_forecast))
    mase = mae / naive_error if naive_error != 0 else np.inf
    mask = y_true != 0
    mape = np.mean(np.abs((y_true[mask] - y_pred[mask]) / y_true[mask])) * 100
    return {
        'MSE': mse,
        'MAE': mae,
        'MAPE': mape,
        'MASE': mase
    }
metrics = calculate_error_metrics(preds, targets)
print("Error Metrics for model PatchTST:")
for metric, value in metrics.items():
    print(f"{metric}: {value:.4f}")
test_index = df.index[df.index > train_end]
preds_series   = pd.Series(preds,  index=test_index[:len(preds)],  name='Predicted')
actual_series  = pd.Series(targets, index=test_index[:len(targets)], name='Actual')
cmp = pd.concat([actual_series, preds_series], axis=1)
cmp.head()
import calendar
jan = cmp[cmp.index.month == 1]
plt.figure(figsize=(14, 5))
plt.plot(jan.index, jan['Actual'], label='Actual', linewidth=1)
plt.plot(jan.index, jan['Predicted'], label='Predicted', alpha=0.8, linewidth=1)
plt.title('LSTM Forecast vs Actual — January 2024')
```

```
plt.xlabel('Date')
plt.ylabel('PV Output')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
df_daily = cmp.resample('D').sum(numeric_only=True)
df_daily
df_daily[['Actual','Predicted']].plot(figsize=(12,8))
df_daily
def rmse(a, b):
    a = np.asarray(a, dtype=float)
    b = np.asarray(b, dtype=float)
    return np.sqrt(mean_squared_error(a, b))
# Robust MAPE: ignore near-zero actuals to avoid blow-ups at night
def mape(a, b, eps=1e-3):
    a = np.asarray(a, dtype=float)
    b = np.asarray(b, dtype=float)
    mask = np.abs(a) > eps
    if not np.any(mask):
        return np.nan
    return np.mean(np.abs((a[mask] - b[mask]) / a[mask])) * 100
from sklearn.metrics import mean_squared_error, mean_absolute_error
monthly = cmp.groupby([cmp.index.year, cmp.index.month]).apply(
    lambda g: pd.Series({
        'RMSE': rmse(g['Actual'], g['Predicted']),
        'MAE': mean_absolute_error(g['Actual'], g['Predicted']),
        'MAPE': mape(g['Actual'], g['Predicted'])
    })
)
```

```
print("\nMonthly metrics (year, month):\n", monthly)
```