CLOUD BASED EFFICIENT ENERGY AUDIT MANAGEMENT SYSTEM USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

KEERTHANA. R	**810021105039**
PREETHI. V. R	**810021105063**
RAGHAVARTHINII. J. K	**810021105067**
SAKTHIPRIYA. S. L	**810021105073**

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRICAL AND ELECTRONICS ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING BIT - CAMPUS, TIRUCHIRAPPALLI - 620 024

ANNA UNIVERSITY : CHENNAI 600025

MAY 2025

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **" CLOUD BASED EFFICIENT ENERGY AUDIT MANAGEMENT SYSTEM USING MACHINE LEARNING "** is the bonafide work of " **KEERTHANA. R, PREETHI. V. R, RAGHAVARTHINII. J. K, SAKTHIPRIYA. S. L "** who carried out the project work under my supervision.

SIGNATURE

Dr. P. ANBALAGAN

SUPERVISOR,

Department of EEE,

University College of

Engineering , BIT Campus,

Tiruchirapalli – 620 024

SIGNATURE

Dr. R. KANIMOZHI

HEAD OF THE DEPARTMENT,

Department of EEE,

University College of

Engineering , BIT Campus,

Tiruchirapalli – 620 024

Submitted for the project held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

In recent years, the fusion of cloud computing and machine learning (ML) paradigm has significantly transformed multiple industries,especially energy auditing is a crucial factor for optimizing the economy of power system , failing to do which will cause significant financial loss, and operational and environmental risks. . Though, conventional techniques are effective but cannot handle huge datasets, affecting the speed of processing. This challenge has driven the integration of ML algorithms and cloud computing, making energy auditing more efficient, scalable, and precise.

This project presents a cloud-based energy audit management system that leverages historical power consumption data, solar irradiance metrics, and wind speed records collected between 2020 and 2024. Data preprocessing inconsistencies are addressed through ML-driven imputation and anomaly detection, while feature engineering enhance predictive accuracy. Machine learning models—including Random Forest, Artificial Neural Networks (ANN), and Support Vector Machines (SVM)—were employed for power consumption forecasting, renewable energy prediction, and anomaly identification. A serverless cloud architecture facilitated scalable storage, real-time analytics, and hybrid accessibility, enabling stakeholders to visualize trends and optimize energy usage through interactive dashboards. The system emphasizes renewable energy integration, utilizing solar irradiance and wind speed data to forecast energy power consumption potential for 2025 to 2030. Solar and wind energy predictions guide infrastructure planning, ensuring efficient utilization of renewable resources. A cost-saving tool, aligned with regional energy tariffs, automates operational expense analysis, demonstrating the financial viability of transitioning to hybrid energy systems.

By bridging ML-driven insights with cloud-based accessibility, the platform reduces reliance on manual audits, minimizes human error, and promotes data-driven sustainability strategies.

ACKNOWLEDGEMENT

Any piece of work that has proved its way remains incomplete if the sense of gratitude and the respect is not being deemed to those who have proved to be supportive during its development period. Though these words are not enough. Our sincere thanks to our college Dean **Dr. T. SENTHIL KUMAR**.University college of Engineering, BIT Campus, for the supports in doing this project.

We take this opportunity to express our deep sense of gratitude and indebtedness to our Head of the department **Dr. R KANIMOZHI**, Department of Electrical and Electronics Engineering and project coordinator,University college of Engineering, BIT Campus, for the valuable ideas and encouragement given to us in bringing this work successfully.

We articulate my heartfelt thanks and to our guide **Dr. P. ANBALAGAN**, Department of Electrical and Electronics Engineering,University college of Engineering, BIT Campus, for the valuable guidance, sincere encouragement and support throughout this project. We convey our hearty thanks to all our teaching and non-teaching staffs and to our parents for helping us in this project to make this project a complete success.

We would especially like to express our extreme gratitude and sincere thanks to our class-coordinators **Dr. P.VIJAYARAJAN** and **Dr.R.VIJAYARAGAVAN** Assistant professor,Department of Electrical and Electronics Engineering, University college of Engineering, BIT campus, Tiruchirappalli for his enthusiastic and innovative guidance during the entire period of our project.

We are obliged to thank the God, our parents, our family and friends for their constant encouragement and support.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.

CHAPTER NO.	TITLE	PAGE NO.

CHAPTER NO.	TITLE	PAGE NO.

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

ANN	ARTIFICIAL NEURAL NETWORKS
API	APPLICATION PROGRAMMING INTERFACE
CART	CLASSIFICATION AND REGRESSION TREES
CI/CD	CONTINUOUS INTEGRATION AND CONTINUOS DEPLOYMENT
IDLE	INTEGRATED DEVELOPMENT AND LEARNING ENVIRONMENT
KDE	KERNEL DENSITY ESTIMATE
MAE	MEAN ABSOLUTE ERROR
MLP	MULTILAYER PERCEPTRON
MSE	MEAN SQUARE ERROR
R^2	R - SQUARED
RFR	RANDOM FOREST REGRESSION
SVM	SUPPORT VECTOR MACHINE
SVR	SUPPORT VECTOR REGRESSION

CHAPTER 1
INTRODUCTION

1.1 GENERAL

India- the global leader in energy consumption in 2023, with its consumption of about 1,407 terawatt-hours (TWh) ranking third globally. Tamil Nadu being among the industrial and economic giants of India, is one of the key contributors in this regard, with its consumption of 103.679 TWh indicating the increasing demand for efficient energy management which is shown in Fig 1.1. With industries growing and urbanization increasing,the necessary for energy savings by incorporating environmental energy saving solutions has never been greater.

Even with the continued efforts globally to improve energy efficiency, most energy consumed is lost through antiquated auditing processes, inefficient monitoring, and lack of real-time predictive analytics. Conventional, energy auditing processes employ manual data gathering and back-end analysis, which provide delayed corrective actions, inaccurate estimates, and poor scalability. As the world moves towards smart energy management, integrating Machine Learning (ML) and Cloud Computing (CC) into auditing systems is like transforming a classical process to advanced, automated, and data-centric process.

Energy auditing is one of the primary tools employed to quantify consumption patterns, identify anomalies and draw energy saving plans to enhance efficiency. Historically, audits have been conducted by on-site testing, analysis of past data, and expert opinion, which, although effective in restricted scenarios, are unable to handle large-scale industrial demands and fluctuating energy consumption. As infrastructure complexity and energy price volatility increased, it was essential to shift towards real-time monitoring, automated forecasting, and predictive cost-saving practices.

Cloud computing revolutionized data handling with remote access, scalable storage facilities, and high-speed processing capabilities for the exclusive utilization of energy audit systems. Cloud integration for this function enables auditors to efficiently

manage large energy data sets from distant locations with reduced dependence on site surveys and increased accuracy of audit output.

In the meanwhile, Machine Learning algorithms enable intelligent automation that enables audits to:

- Detect inefficiencies and power loss.
- Forecast future energy consumption patterns
- Optimization of dynamic resource allocation These innovations mark the departure from static energy audits to adaptive, self-improving AI-based audit systems, which improve decision-making and sustainability.

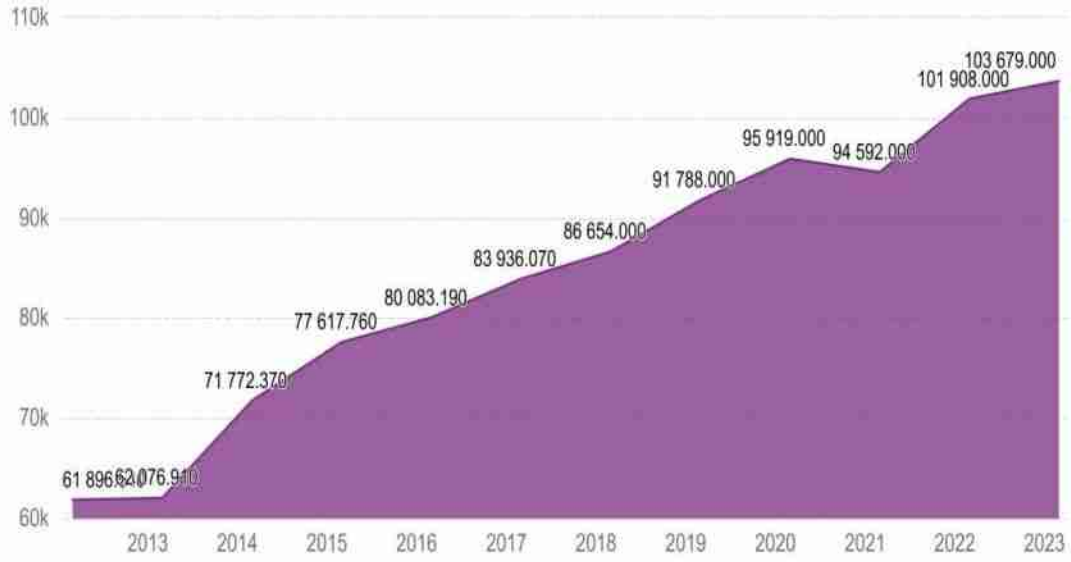


Fig 1.1 Energy consumption of Tamilnadu from (2013 to 2023)

Source:WWW.CEICDATA.COM

1.2 SIGNIFICANCE AND MOTIVATION

Industries have a responsibility to smoothly integrate renewable energy sources, such as wind and solar, into the infrastructure and grids that are already in place. Forecasting solar radiation, determining wind energy usage, and optimizing peak generation hours are responsibilities that fall to machine learning-based analytics. Reducing storage loss and enhancing grid stability are impossible without intelligent auditing technologies. The need for cloud and machine learning-based energy audit

platforms is further highlight the increasing dependence for intelligent renewable energy forecasting.

Conventional auditing techniques continue to employ reactive analysis rather than proactive action despite advancements in energy monitoring and control. The majority of industries still rely on manual inspections conducted at regular intervals, which typically fail to identify small inefficiencies or changes in real time that greatly elevate operating costs. Furthermore, even though cloud computing makes data more accessible, most cloud-based energy audit systems currently in use lack artificial intelligence (AI), which results in poor prediction capabilities and lagging optimization procedures. The implementation of economically viable energy-saving measures is hindered by the lack of machine learning-based anomaly detection and dynamic forecasting capabilities.

Similarly, renewable energy auditing is lacking in forecasting energy generation, maximizing grid integration, and dynamic load balancing. By developing a cloud-driven machine learning-based energy audit system, this study aims to close these gaps by offering:

- Real-time analytics to increase automation and audit accuracy.
- Automated cost-saving recommendations based on algorithm monitoring.
- Optimization of renewable energy, improving integration and storage efficiency Scalable solutions for commercial, governmental, and industrial applications.

Energy auditing will be transformed from a passive to an active, self-improving system with cloud-based infrastructure and AI-powered analytics, leading to increased efficiency, reduced operating costs, and the use of green energy.

1.3 RESEARCH OBJECTIVE AND SCOPE

The main goal of this research is:

- ✓ To create an effective energy auditing system that uses cloud computing(cc) and machine learning to data – driven, intelligent and scalable energy management.

✓ Motivated for the increasing demand for optimal use of renewable resources and economic energy balancing, this study audits patterns of energy consumption, identifies inefficiencies and investigates predictive modelling methods to improve power generation sustainability.

✓ In order to do this, a number of machine learning algorithms have been investigated and evaluated using weather and power consumption data for Tamil Nadu from 2020 to 2024.

✓ Tamil Nadu's energy consumption over the next five years (2025–2030) can be predicted thanks to these datasets, which offer insightful information about past trends and environmental influences.

✓ In order to evaluate the possibility to decreases the grid dependency and encourage the use of solar and wind energy, a renewable energy utilization calculator has also been created. Advanced machine learning ensemble models like Support Vector Machine (SVM), Random Forest Regressor (RFR), and Artificial Neural Networks (ANN) are used in energy forecasting to increase prediction accuracy and efficiency.

✓ Additionally, a cost-saving calculator has been included to examine how much the use of renewable energy reduces reliance on non-renewable resources, lowering overall consumption costs and encouraging the use of sustainable energy.

✓ This research creates a next-generation energy auditing framework that can perform proactive optimization, intelligent forecasting, and dynamic monitoring by fusing cloud computing scalability with machine learning-driven predictive analytics.

✓ This gives the commercial, governmental, and industrial sector a powerful tool for achieving energy efficiency and sustainability.

1.4 CONCLUSION

This research culminates in the development of a next-generation energy auditing framework. By effectively combining the scalability of cloud computing with

the predictive power of machine learning-driven analytics, this framework enables proactive optimization, intelligent forecasting, and dynamic monitoring. Ultimately, this provides a robust tool for the commercial, governmental, and industrial sectors to achieve enhanced energy efficiency and sustainability.

CHAPTER 2
LITERATURE REVIEW

2.1 INTRODUCTION

The incorporation of machine learning further enhances the overall efficiency in predicting anomaly detection, forecasting, optimized and faster solutions than that of traditional way, this uncover usage patterns, consumption trends, renewable energy analysis and predict future demands with higher accuracy. These insights empower organizations to identify inefficiencies, reduce energy wastage, and adopt renewable energy sources strategically which streamlines energy audit process and also aligns global sustainability goals. This literature review explores the architecture, machine learning techniques, and practical applications of cloud-based energy audit management systems, emphasizing their potential to revolutionize energy management practice globally.

2.2 STATE OF ART

This paper [1] by González-Briones et al., provides a review of different machine learning models that have been applied to the task of forecasting electricity consumption in the year 2019. The review made allowed to observe that for the data set using the Linear Regression and Support Vector Regression has obtained a success of 85.7% being the best results provided. The models of machine learning that the literature shows that better results produce (K-Nearest Neighbors, Linear Regression, Random Forests, Support Vector Regression and Decision Tree) have been evaluated.This work [2] by Nagesh, focuses on improving energy consumption prediction models using machine learning and sampling strategies. This work include further investigation on gradient-based sampling strategies. The machine learning model can also be extended to deep learning field to handle high- dimensional input scenario of the year 2021 .This system aims to 7 analyze energy consumption patterns in homes to identify areas for potential energy savings.

This paper by Chang et al., [5] presents a comparative analysis of various machine learning algorithms to predict energy consumption in buildings. It evaluates

the performance of different models to identify the most effective approaches for building energy prediction. Here , the random forest classifier achieved the highest classification accuracy at 92.28%, followed closely by the support vector classifier at 91.38%, and the artificial neural network at 89.89%.For five energy levels, the random forest again performed best with 83.40% accuracy, the support vector classifier reached 81.07%, and the artificial neural network had 78.07%.With seven energy levels, classification accuracy dropped across all models, with random forest at 76.39%, support vector classifier at 72.57%, and artificial neural network at 66.51%.

This paper by Agil Yolchuyev [22] is a research effort to develop an integrated platform that allows for the intelligent monitoring, control, and planning of energy consumption and generation in urban contexts. This project contributes to the design of devices and strategies to optimize the use of energy resources, with a focus on sustainability and citizen participation. Improvements in communication infrastructure and forecasting models are also proposed to more effectively manage renewable and non-renewable sources in the field of smart power grids.

The proposed [9] Energy Cloud system by H.Sequeira et al., offers innovative tools for real-time monitoring and analysis, providing industries with actionable insights and improved efficiency. These developments pave the way for sustainable and intelligent energy management practices, addressing the complexities and scale of modern industrial operations. This paper [14] by Naveen explores the role and advantages of leveraging cloud computing infrastructure for the continuous monitoring of energy consumption data, how enhances smart grid capabilities by improving scalability, cost- efficiency, and real-time data processing.A promising future lies in further integrating cloud technologies for optimized power management, monitoring, and economic dispatching. [24] by Y. Jiang et al., Concludes that model optimization using RS can effectively predict heating loads, aiding in energy efficiency improvements.

2.3 EXISTING RESEARCH GAPS

Both Machine Learning (ML) and Cloud Computing face significant hurdles in fully realizing their potential for energy auditing. For ML, key limitations include a lack of real-time adaptability in models, current scalability issues preventing widespread industrial application, insufficient integration with renewable energy optimization, and a need for more robust comparative analysis of ML models across diverse datasets. In Cloud Computing, the primary challenges are the scarcity of AI-driven solutions for predictive analytics, unaddressed security and data privacy concerns, minimal exploration of edge computing for localized optimization, and a lack of comprehensive cloud-based forecasting models for large-scale renewable energy utilization. Overcoming these challenges will require developing more sophisticated, adaptable, secure, and scalable technological advancements in both domains to truly revolutionize energy auditing and management.

2.3.1 Energy Auditing in Machine Learning

- ✓ **Limited Real-Time Adaptability:** Most studies focus on historical data analysis for energy forecasting, but there is a lack of real-time adaptive models that continuously learn and adjust based on live consumption patterns.
- ✓ **Scalability Issues:** While ML models have been tested for household and building energy audits, there is limited research on large-scale industrial applications where energy consumption is more complex.
- ✓ **Integration with Renewable Energy:** Existing studies primarily focus on predicting electricity demand, but fewer works explore how ML can optimize renewable energy integration within smart grids.
- ✓ **Comparative Analysis of ML Models:** While some papers compare ML algorithms, there is a need for benchmarking models across diverse datasets to determine the most effective approach for different energy auditing scenarios.

2.3.2 Cloud Computing in Energy Auditing

- ✓ **Lack of AI-Driven Cloud Solutions:** Many cloud-based energy audit systems focus on data storage and accessibility, but fewer studies integrate AI-driven automation for predictive analytics and anomaly detection.
- ✓ **Security & Data Privacy Concerns:** Cloud computing introduces data security risks, yet existing research lacks robust frameworks for secure energy auditing in cloud environments.
- ✓ **Limited Research on Edge Computing:** While cloud platforms enable remote monitoring, there is minimal exploration of edge computing for localized, real-time energy optimization without relying on centralized cloud servers.
- ✓ **Renewable Energy Forecasting in Cloud Systems:** Studies on home energy management using cloud services exist, but there is a gap in cloud-based forecasting models for large-scale renewable energy utilization.

2.3.3 Potential contributions of our research

- ✓ Develop real-time adaptive ML models for energy auditing.
- ✓ Create scalable frameworks specifically for industrial energy audits.
- ✓ Integrate renewable energy forecasting capabilities into ML-driven audits.
- ✓ Enhance cloud-based energy audit systems with AI-driven analytics and automation.
- ✓ Establish robust security and data privacy frameworks for cloud-based energy auditing.
- ✓ Conduct comprehensive benchmarking of ML models across diverse datasets for various energy auditing scenarios.

2.4 CONCLUSION

The findings of the review will be used to design a real – time energy audit management system based on artificial intelligence and automated methods. The review

suggests suitable methods for the further development of the project and limitation so the traditional methods. Not all reviewed papers provided the same evaluation metrics, therefore, it is impossible to evaluate and compare the results of each similar study. The higher metric system was implemented to perform classification tasks in some of the research papers.

CHAPTER 3

CONCEPT OF DATA PROCESSING METHODOLOGY AND TECHNIQUES

3.1 DATA SELECTION AND PREPROCESSING

In order to improve energy analysis, anomaly detection, predictive forecasting, and renewable energy evaluation, the system presents a cloud-based, machine learning-driven platform. By offering real-time insights into consumption trends and possible savings from solar and wind energy sources, it solves the lack of integrated intelligence in conventional energy auditing. The platform combines Random Forest-based anomaly detection to find consumption inefficiencies with cloud infrastructure for scalable energy data processing. Furthermore, to increase prediction accuracy , different forecasting models such as Random Forest, Support Vector Machines (SVM), and Artificial Neural Networks (ANN) are compared. Additionally, the system provides a basic energy-saving suggestions. Using historical Tamil Nadu energy data, a feasibility study in solar and wind energy is carried out to encourage the adoption of renewable energy, allowing for more effective planning for sustainable integration. The system produces five-year energy forecasts (2025–2030) based on data from 2020–2024, giving consumers and businesses useful information.

3.2 SOURCES OF DATA

This multi-source data collection aims to build a comprehensive picture for advanced energy forecasting and auditing. The power consumption data provides the foundational understanding of actual demand, enabling the identification of historical consumption patterns and informing future load predictions for grid stability and efficient resource allocation. Solar irradiance data is crucial for accurately forecasting renewable energy generation. Meteorological data, including wind speed and other weather history from Chennai, serves to capture the environmental factors directly influencing both energy demand and renewable generation. Together, these datasets

enable the development of robust predictive models that can account for diverse variables, leading to more precise energy auditing, optimized power distribution, and improved renewable energy integration within the region.

3.2.1 Power consumption data collection

The power consumption data from the Southern Region Load Dispatch Center (SRLDC). The SRLDC, being the operational hub for power distribution in Southern India, provides a crucial and authentic record of energy demand across the entire region. This comprehensive data set, visually represented in "Fig 3.1", is invaluable for gaining a deep understanding of how energy is consumed in this specific geographical area. This detailed temporal understanding of power consumption is essential for developing accurate predictive models and effective energy management strategies.

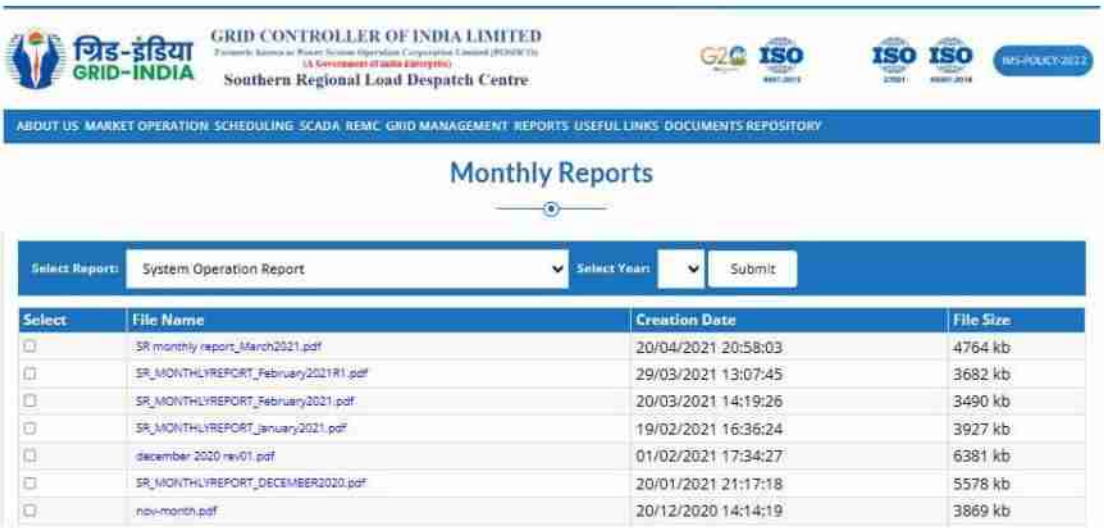


Fig 3.1 A sample of monthly reports from SRLDC website

The actual withdrawal data of Tamil Nadu is taken establishing the baseline for forecasting future energy needs and identifying key consumption behaviors. It serves as the fundamental target variable for predictive models.

The SRLDC provides a detailed report of southern region power schedule which is shown in Fig 3.2 form this the actual withdrawal data of Tamil Nadu is taken for our demonstration purpose from that a sample of data is displayed in Table 3.1

NET DRAWAL SCHEDULE AND ACTUAL DRAWAL FOR THE MONTH OF March-2021

	ANDHRA PRADESH		TELANGANA		KARNATAKA		KERALA		TAMILNADU		PONDICHERRY	
DATE	Net Schd	Act. Drawal	Net Schd	Act. Drawal	Net Schd	Act. Drawal	Net Schd	Act. Drawal	Net Schd	Act. Drawal	Net Schd	Act. Drawal
01-Mar-21	69.17	70.28	143.26	145.01	87.53	88.90	54.07	54.86	190.84	190.63	7.55	7.53
02-Mar-21	71.70	72.38	144.95	145.95	97.12	100.62	54.48	54.96	200.66	198.99	7.72	7.59
03-Mar-21	72.34	72.18	145.62	145.91	92.99	91.80	56.31	56.52	196.98	193.85	7.86	7.62
04-Mar-21	68.37	68.12	144.80	144.74	90.14	88.58	57.37	57.43	193.40	192.27	7.83	7.75
05-Mar-21	71.37	71.50	144.61	144.15	93.12	92.95	55.19	55.06	191.39	190.90	7.85	7.83
06-Mar-21	74.93	75.11	143.86	144.78	92.84	93.90	51.17	51.57	190.58	195.45	8.01	7.92
07-Mar-21	72.79	73.92	145.82	146.91	81.99	83.56	50.28	51.26	179.83	177.98	7.37	7.27
08-Mar-21	66.60	66.90	146.97	147.01	77.96	77.21	55.07	54.93	195.28	194.88	7.97	7.77
09-Mar-21	76.93	78.79	147.06	147.54	78.43	79.06	55.96	55.68	203.31	200.91	8.33	8.29
10-Mar-21	85.84	86.42	149.49	150.54	80.73	80.75	55.67	55.56	198.14	196.56	8.38	8.44
11-Mar-21	85.65	87.67	153.66	156.08	90.31	92.62	54.14	54.83	199.75	199.36	8.49	8.28
12-Mar-21	85.78	86.32	153.28	152.75	106.05	108.41	55.92	56.16	202.50	200.03	8.58	8.28
13-Mar-21	84.94	85.68	147.58	145.90	114.48	114.30	57.02	57.07	209.44	207.83	8.54	8.21
14-Mar-21	86.10	86.64	146.44	146.25	103.08	103.30	58.42	58.45	196.84	195.10	7.90	7.62
15-Mar-21	85.16	86.81	143.63	145.06	104.88	106.60	58.81	59.23	206.86	205.45	8.44	8.21
16-Mar-21	82.08	85.12	138.06	139.66	101.33	106.06	55.94	56.02	213.04	212.18	8.41	8.25
17-Mar-21	79.02	79.03	149.70	151.82	96.91	105.01	54.74	54.28	214.56	213.29	8.55	8.51
18-Mar-21	77.16	77.09	147.64	148.59	107.66	112.59	56.38	56.62	216.87	215.93	8.54	8.24
19-Mar-21	82.71	84.56	150.00	151.81	100.24	107.22	57.55	58.09	220.10	221.08	8.60	8.47
20-Mar-21	89.74	90.39	153.13	154.30	100.15	105.53	58.11	58.71	220.19	221.96	8.86	8.49
21-Mar-21	89.55	89.45	135.77	136.01	119.63	118.67	58.09	58.14	204.64	204.50	7.95	7.76

Fig 3.2 A Sample of PDF from monthly reports for southern regions

Table 3.1 Extracted sample data of Power consumption for Tamilnadu

DATE	Power_Consumption(MU)
2020-01-01 00:00:00	159.68
2020-01-02 00:00:00	159.68
2020-01-03 00:00:00	159.68
2020-01-04 00:00:00	159.68
2020-01-05 00:00:00	159.68
2020-01-06 00:00:00	159.68
2020-01-07 00:00:00	159.68
2020-01-08 00:00:00	159.68
2020-01-09 00:00:00	159.68

3.2.2 Weather data collection

The five years of meteorological information of 13.12°N, 80.17°E Chennai, Tamil Nadu, India weather history collected from **Wunderground** website whose home page is shown in Fig 3.3.

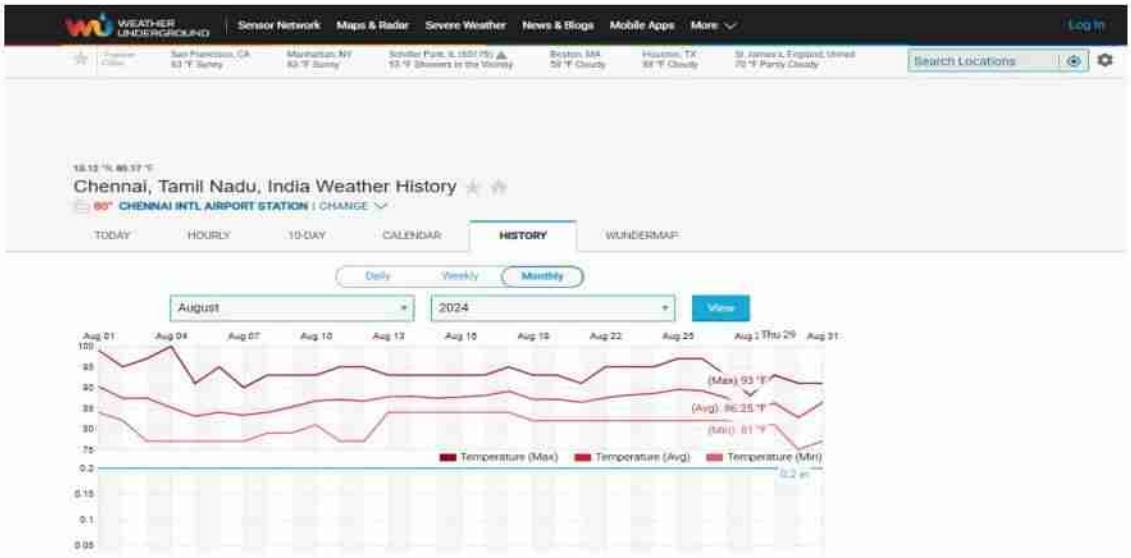


Fig 3.3 Monthly weather data from wunderground

Daily Observations

Time	Temperature (°F)			Dew Point (°F)			Humidity (%)			Wind Speed (mph)			Pressure (in)			Precipitatio
Aug	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Total
1	99	90.3	84	79	76.5	75	75	64.8	50	15	10.6	6	29.7	29.6	29.6	0.00
2	95	87.4	82	82	77.2	75	84	72.1	56	15	8.3	2	29.7	29.6	29.6	0.00
3	97	87.4	77	81	76.9	73	94	72.7	53	15	10.3	5	29.8	29.7	29.6	0.00
4	100	85.2	77	82	78.5	73	100	81.7	53	15	7.9	2	29.7	29.7	29.6	0.00
5	91	83.0	77	82	77.4	73	94	83.9	66	18	6.8	2	29.7	29.7	29.6	0.00
6	95	84.0	77	82	77.5	72	94	81.7	63	23	9.5	5	29.7	29.6	29.6	0.00
7	90	83.3	77	81	77.8	75	94	83.7	70	14	5.7	1	29.7	29.7	29.6	0.00
8	93	84.0	79	79	77.7	75	94	81.8	63	17	8.6	2	29.7	29.7	29.6	0.00
9	93	85.3	79	81	78.5	75	89	80.3	67	15	9.2	5	29.7	29.7	29.6	0.00
10	93	86.8	81	82	79.7	75	89	79.3	66	15	7.6	2	29.7	29.7	29.6	0.00
11	95	87.0	77	82	78.3	73	89	75.8	66	17	8.8	2	29.7	29.6	29.5	0.00

Fig 3.4 Daily weather data option from Wunderground

Fig 3.4 shows the datas that are available in the wundeground website and it includes variables like temperature(°F), dew point(°F), average wind speed(mps), maximum wind speed(mps), atmospheric pressure(hPa), humidity(g/m^3) are taken in account because, these variables significantly influence power consumption, particularly for heating and cooling purposes. Understanding the relationship between weather patterns and energy demand is essential for accurate forecasting. The dataset offers a comprehensive view of the environmental conditions.

Table 3.2 Extracted sample weather data

DATE	Temperature (F)	Dew Point (F)	Max Wind Speed (mps)	Avg Wind Speed (mps)	Atm Pressure (hPa)	Humidity (g/m^3)
2020-01-01 00:00:00	82	79	0.002222222	0.000944444	30	88.99106775
2020-01-02 00:00:00	88	79	0.002777778	0.001166667	30	70.08280141
2020-01-03 00:00:00	88	81	0.003333333	0.001416667	30	76.08803973
2020-01-04 00:00:00	88	79	0.002777778	0.001583333	29.9	70.08280141
2020-01-05 00:00:00	86	79	0.002777778	0.001916667	29.9	75.81749318
2020-01-06 00:00:00	88	77	0.004166667	0.00225	30	64.48523034
2020-01-07 00:00:00	86	75	0.004166667	0.002583333	30	64.12278109
2020-01-08 00:00:00	84	75	0.003333333	0.001611111	29.9	69.43654461
2020-01-09 00:00:00	88	72	0.0025	0.001111111	29.9	52.12836646

Table 3.2 presents a daily meteorological dataset, likely for Chennai, spanning early January 2020. It includes key weather parameters such as Temperature (F), Dew Point (F), Maximum and Average Wind Speed (mps), Atmospheric Pressure (hPa), and Humidity (g/m^3), offering insights into the local environmental conditions during this period.

3.2.3 Solar irradiance data collection

Solar Irradiance Data is Sourced from **National Aeronautics And Space Administration (NASA)** utilizing a five-year historical dataset, which quantifies "All Sky Surface Shortwave Downward Irradiance" in kilowatt-hours per square meter per day (kW-hr/m²/day) is shown in Fig 3.5 and a sample of data given is presented in Table 3.3 .

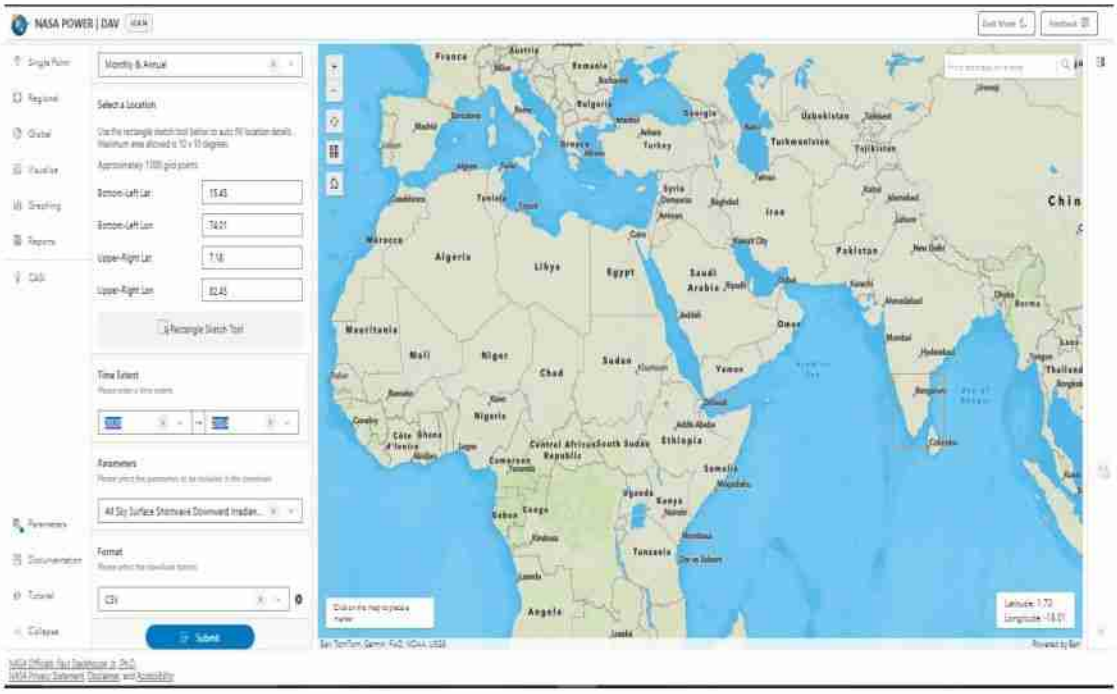


Fig 3.5 Monthly solar data from NASA

By meticulously analyzing the fluctuations and patterns within this comprehensive dataset, the study gains invaluable insights into the daily, seasonal, and annual variations in solar intensity within the region. Such insights are not only fundamental for accurately evaluating the potential for solar power generation but also

crucial for forecasting how solar intensity affects overall energy demand, particularly during periods of high solar output. Ultimately, understanding these variations is indispensable for robust renewable energy planning, optimizing grid integration of solar power, and enhancing the overall sustainability of energy systems.

Table 3.3 Extracted sample solar data

DATE	solar irradiance (kg-hr/m^2/day)
2020-01-01 00:00:00	5.69
2020-01-02 00:00:00	5.69
2020-01-03 00:00:00	5.33
2020-01-04 00:00:00	5.4
2020-01-05 00:00:00	5.32
2020-01-06 00:00:00	5.72
2020-01-07 00:00:00	5.61
2020-01-08 00:00:00	5.42

3.2.4 Wind speed data collection

The five years of windspeed information of Tamil Nadu, India weather history collected from **Wunderground** website .

Variable maximum wind speed (mps) is taken in account because, this variable significantly helps in finding energy that can be utilized through wind. Understanding the relationship between wind speed patterns and energy demand is essential for forecasting wind energy utilization. The dataset offers a comprehensive view of the maximum wind speed conditions shown in Table 3.4.

Table 3.4 Extracted sample wind speed data

DATE	Max Wind Speed (mps)
2020-01-01 00:00:00	0.002222222
2020-01-02 00:00:00	0.002777778
2020-01-03 00:00:00	0.003333333
2020-01-04 00:00:00	0.002777778
2020-01-05 00:00:00	0.002777778
2020-01-06 00:00:00	0.004166667
2020-01-07 00:00:00	0.004166667
2020-01-08 00:00:00	0.003333333

3.3 DATA CLEANING AND FEATURE SELECTION

Data cleaning is a crucial step in preparing energy consumption and environmental datasets for machine learning models. Since real-world data often contains inconsistencies, missing values, and outliers, ensuring high-quality data is essential for accurate forecasting. Handling missing data is a crucial work before doing any process . ML-powered anomaly detection help in eliminating these inconsistencies. Standardizing units is another essential process, ensuring uniform measurement formats for variables such as temperature, wind speed, and humidity across different sources. Additionally, timestamp formatting is critical for time-series data, ensuring correct chronological order for trend identification and accurate forecasting.

Normalization and scaling are fundamental feature engineering techniques that improve machine learning model performance by ensuring consistency in variable magnitude. Since energy consumption values and environmental factors vary widely, scaling helps enhance stability and efficiency in computational processes. Min-Max Scaling rescales features to a fixed range, making them comparable and reducing bias

in ML algorithms. By implementing effective data cleaning and proper feature scaling, ML models can enhance predictive accuracy, improve pattern recognition, and drive optimized energy management.

The flowchart Fig 3.6 illustrates a comprehensive data-driven forecasting process. It begins with the crucial step of "Power, Weather, Monthly Data Collection," followed by "Data Concatenation" to unify diverse datasets. "Missing Value Processing" then cleans the data, which is subsequently refined through "Feature Selection" to identify key variables. An initial "Forecasting" step leads into the "Model Building" phase, where sophisticated machine learning algorithms like SVM, RFR, and ANN are trained. Finally, these models are used for "Prediction," with their performance rigorously assessed using metrics such as MSE, R^2, and MAE, ensuring robust and accurate forecasts.

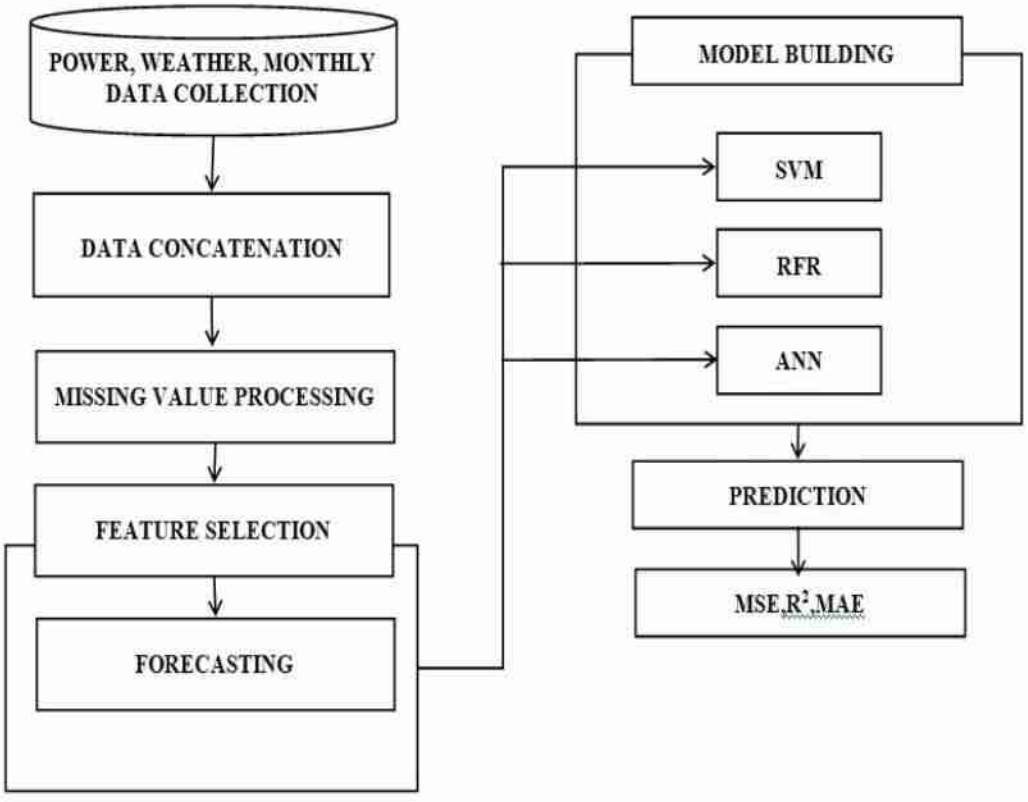


Fig 3.6 Feature selection and forecasting

3.4 MACHINE LEARNING MODELS

3.4.1 SVR

SVM can be divided into SVC (Support Vector Classification) for classification and SVR for forecasting. Support Vector Regression (SVR) arises to perform classifications in datasets whose output variable is continuous and oscillates within a subset of IR. SVR's have been used regularly in the prediction of electrical consumption.There is one major advantage within the SVR optimization formulation: there is a unique solution which minimizes a convex function. SVR does have a potential disadvantage: scalability.

The convex criterion function is optimized using quadratic programming optimization algorithms.

$$K(x_i, x) = \exp(-\gamma * ||x_i - x||^2)$$

Here, $K(x_i, x)$is the kernel function,γ (gamma) is a tunablehyper parameter,
$||x_i - x||^2$ is the Euclidean distance squared between the two points.

$$\text{Scaled Prediction} = \left(\sum_{i=1}^{n} (\alpha_i - \alpha_i^*) * K_i \right) + b$$

Here,$(\alpha_i - \alpha_i^*)$ is the Lagrange Multiplier.K_i is the kernel function, b is the bias term.

$$\text{Predicted Power Consumption} = \{ (\text{Scaled Prediction} * (\text{Power}_{max} - \text{Power}_{min})) + \text{Power}_{min} \}$$

It works by finding an optimal hyper plane that separates different data points and can handle non-linear relationships using kernel functions.

3.4.2 RFR

Random Forests (RFs) are a powerful and widely used ensemble learning technique that can handle both classification and regression tasks effectively. The core

idea behind RFs is to build a multitude of decision trees during training time and aggregate their predictions to improve overall accuracy and robustness. Each individual tree in the forest is constructed by sampling, with replacement, from the original dataset—a method known as bootstrapping. Moreover, at each node of a tree, a random subset of features is selected, and the best split is chosen only from this subset, introducing diversity among the trees and reducing correlation between them.

Unlike traditional decision tree methods such as CART (Classification and Regression Trees), which grow a single tree based on the full dataset and all features, RFs intentionally introduce randomness in both the data (via bootstrapping) and the feature selection process. This helps in mitigating the risk of over fitting, which is a common issue in decision trees when they are allowed to grow too deep. In RFs, the trees are grown to their maximum possible size without pruning, allowing each tree to capture complex patterns in the data, while the ensemble effect ensures that the final model remains generalizable.

The ensemble generates a collection of diverse classifiers (trees), each of which may have high variance, but when combined through averaging (for regression) or majority voting (for classification), the variance is significantly reduced. As a result, the random forest model offers high accuracy, robustness to noise, and better generalization to unseen data. Additionally, because decision trees are relatively fast to construct and evaluate, RFs can scale well with large datasets and high-dimensional feature spaces, making them suitable for a wide range of practical applications.

3.4.3 ANN

Artificial Neural networks are a computational approach for loosely modeling the way a biological brain solves problems with large clusters of biological neurons connected by axons. It is based on a large collection of neural units known as artificial neurons. Each neuron is connected with many others to form neural network, and links

can be enforcing or inhibitory in their effect on the activation state of connected neural units. Each individual neural unit may have a summation function which combines the values of all its inputs together.

Multi-layer perceptron (MLP) is a feed forward artificial neural network that uses back propagation for training. It has at least three layers of nodes in which, except the first layer, each node uses a nonlinear activation function. MLP minimizes an error function by learning proper weights for each pattern in training set.
Each neuron's weighted sum before activation is given by:

$$Z_1 = (Input_1 * w_{i}1) + (Input_2 * w_{i}2) + b_i$$

Where:

$w_{i}1, w_{i}2$ are the weights for the respective inputs.

b_i is the bias for neuron ii.

After applying the ReLU activation function:

$$a_1 = \max(0, Zi)$$

Final Output Calculation:

The final neuron output (before scaling to power consumption) is:

$$Z_{out} = (a_1 \times w_{a1}) + (a_2 \times w_{a2}) + (a_3 \times w_{a3}) + b_{out}$$

$$a_{out} = Z_{out}$$

Predicted Power Consumption Equation:

$$\text{Predicted Power Consumption} = a_{out} \times (Power_{max} - Power_{min}) + Power_{min}$$

3.5 CLOUD – BASED ENERGY AUDITING

The architecture in Fig 3.7 integrates data collection, machine learning models, cloud storage, and interactive dashboards, providing an end-to-end auditing system. Energy

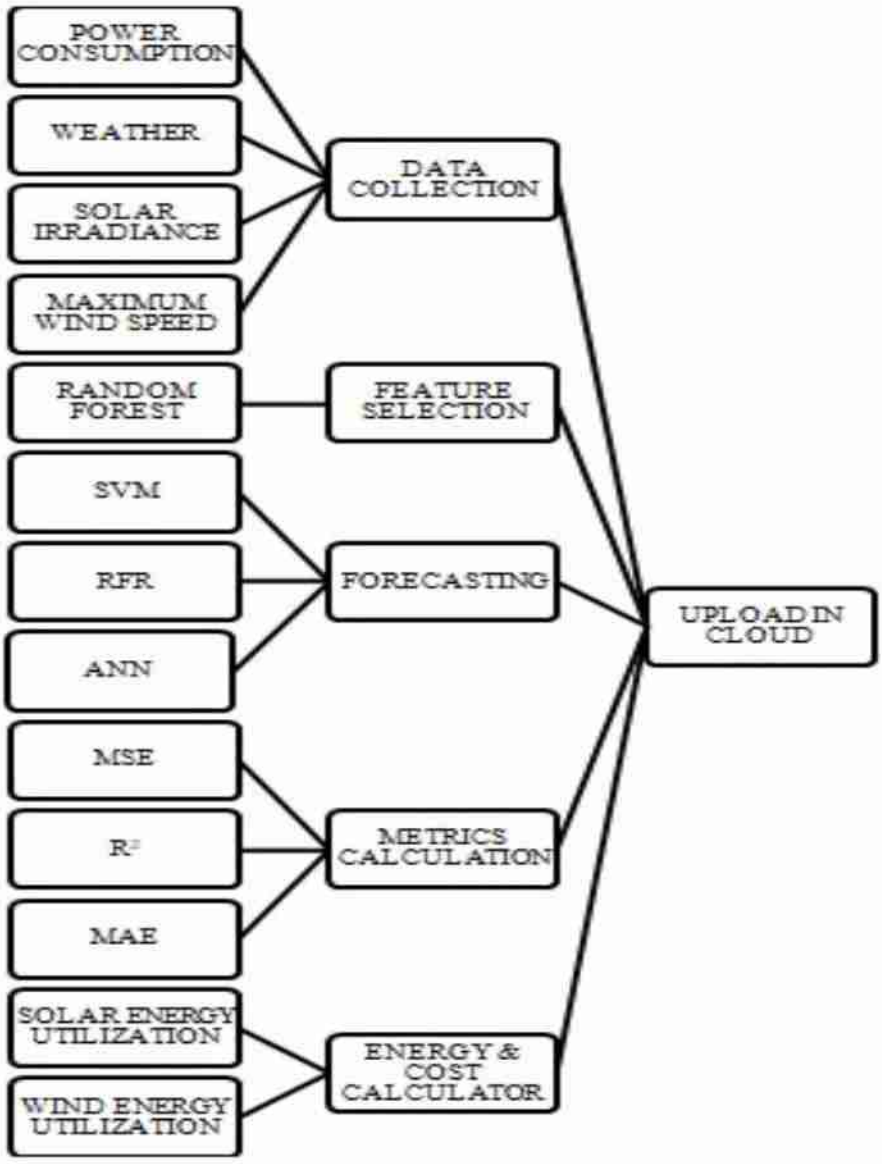


Fig 3.7 Overall Architecture

auditing is a vital component in optimizing renewable energy utilization and ensuring efficient power management. Leveraging cloud computing and machine learning (ML) significantly enhances data processing scalability, automation, and real-time

forecasting. Cloud computing plays a pivotal role in scalable data processing, allowing seamlessly management and analyze of vast datasets with enhanced flexibility and efficiency. In the context of energy forecasting and renewable energy utilization, cloud integration provides on-demand computing resources, real-time accessibility, and interactive auditing features that empower decision-makers with actionable insights. This architecture integrates data collection, machine learning models, cloud storage, and interactive dashboards, providing an end-to-end auditing system.

3.5.1 Data collection & ingestion layer

This foundational Data Ingestion Layer (or Data Acquisition Layer) is the crucial starting point for any energy data system. Its core responsibility is to systematically gather and store raw energy-related data from a wide array of sources. These sources are incredibly diverse, encompassing publicly available websites that might publish energy reports or real-time consumption figures, APIs (Application Programming Interfaces) offered by energy providers or smart device manufacturers for programmatic data access, and IoT sensors deployed in homes, businesses, or industrial settings that constantly stream usage data. Furthermore, it integrates with large-scale energy grids to capture operational telemetry and incorporates vast historical datasets to provide crucial context and identify long-term trends. The collection process itself involves various techniques: web scraping to extract data from web pages, and API requests for more structured data retrieval. Once acquired, this raw data is meticulously parsed into structured formats such as readily analyzable Excel sheets, ubiquitous CSV files for tabular data, and flexible JSON datasets for hierarchical information.

A critical aspect of this layer is the immediate and direct upload of all collected data to GitHub. This isn't just for storage; it transforms GitHub into a robust version control system. This practice allows for comprehensive tracking of every change made to the data, fosters seamless collaboration among data engineers and analysts, and provides the invaluable ability to revert to previous data versions if errors or

inconsistencies are discovered. This meticulous approach to version control is paramount for ensuring data integrity and establishing a clear, auditable trail of all data modifications. Ultimately, the overarching goal of this layer is to guarantee the continuous and structured collection of data, forming the reliable bedrock upon which all subsequent energy data analysis, modeling, and insight generation will be built. Without a robust and well-maintained data ingestion layer, the downstream analytical processes would lack the necessary high-quality, consistent input.

3.5.2 Github storage and management layer

Unlike conventional cloud-based storage solutions, this architecture innovatively leverages GitHub as its centralized data storage and version control system, effectively transforming it into a structured data lake specifically for audit datasets. This strategic choice allows for the direct storage of raw datasets within the repository. Crucially, these data files are stored alongside the scripts used for their collection and processing, simplifying data retrieval and ensuring that the context of data generation is always maintained. The inherent version control capabilities of GitHub are fully exploited to meticulously track all dataset updates, enabling complete reproducibility of analyses and providing a robust historical record of every data change, which is vital for auditing purposes.

This architecture further distinguishes itself through its seamless integration with Streamlit Cloud. This integration facilitates the real-time retrieval of audit data, allowing for dynamic dashboards and interactive applications that can access and display the latest information directly from the GitHub repository. The combined approach of using GitHub as a secure, trackable, and accessible data storage medium, coupled with its real-time integration with Streamlit Cloud, achieves the overarching goal of maintaining a highly efficient and transparent data environment. This ensures that audit datasets are not only securely stored and versioned but also readily available

for immediate use and analysis, fostering a high degree of confidence in the data's integrity and accessibility.

3.5.3 Machine learning and forecasting layer

This pivotal layer is dedicated to data preprocessing and energy forecasting, leveraging the power of machine learning (ML) models to extract meaningful insights and predict future trends. Before any modeling can occur, the raw energy data undergoes rigorous preprocessing. This involves essential steps such as data cleaning to handle missing values, inconsistencies, or outliers; normalization to scale features to a common range, preventing certain features from dominating the model; and transformation to convert data into a suitable format for ML algorithms. These preprocessing tasks are efficiently carried out using robust Python libraries like Pandas for data manipulation, NumPy for numerical operations, and SciPy for scientific computing.

For energy forecasting, a careful model selection process is undertaken, primarily focusing on supervised learning algorithms. These include Random Forest, an ensemble method known for its robustness and ability to handle complex non-linear relationships; Neural Networks, capable of learning intricate patterns in large datasets, particularly useful for time-series forecasting; and Support Vector Machines (SVM), effective for both classification and regression tasks, even with limited data. Beyond individual models, the architecture also explores hybrid techniques, which involve combining multiple ML algorithms to potentially achieve higher accuracy and improve the predictive power of energy consumption forecasts. All these ML models are trained and deployed within a cloud execution environment, allowing for scalable computing resources and seamless interaction with the datasets securely hosted on GitHub, ensuring a continuous and efficient workflow from data acquisition to model deployment. The ultimate goal of this sophisticated layer is to forecast renewable

energy utilization trends with high precision, providing critical information for energy management and planning.

3.5.4 Cloud deployment and interactive auditing layer

Upon the completion of data processing and energy forecasting, the system transitions to the crucial phase of data visualization and auditing, primarily facilitated through Streamlit Cloud, which ensures real-time accessibility for users. The entire deployment framework is built around a hosted application on Streamlit Cloud that seamlessly interacts with the processed and forecasted data securely stored on GitHub. This integration allows the application to pull the latest information directly, ensuring that users are always working with up-to-the-minute insights.

For clear and intuitive presentation, a variety of powerful visualization tools are employed, including Matplotlib for static and animated graphs, Seaborn for aesthetically pleasing statistical graphics, and advanced features of MatPlotlib (likely referring to more complex plots or interactive elements within Matplotlib) to create dynamic charts and heatmaps. Beyond mere display, the platform offers robust audit features. Users can access real-time forecasting directly derived from the GitHub-hosted datasets, providing immediate insights into future energy trends. Furthermore, the system generates actionable recommendations for energy optimization, leveraging the intelligence gleaned from the machine learning insights. A critical auditing feature is the ability to perform comparative analysis between historical and predicted trends, allowing users to assess model accuracy and understand deviations. The overarching goal of this layer is to provide a seamless, interactive auditing experience for users, empowering them with comprehensive, easy-to-understand data to make informed decisions regarding energy utilization.

3.5.5 Scalability and optimization layer

This strategic approach to deployment and automation is critical for ensuring the entire architecture remains inherently scalable, efficient, and autonomous. A key

enabler of this is serverless execution, which leverages cloud-native functions for automated auditing processes. This means that compute resources are provisioned on demand, scaling automatically with workload fluctuations without requiring constant server management, thus optimizing efficiency and cost.

Furthermore, the deep integration with GitHub is central to its autonomy; Streamlit applications directly fetch and update audit data from the repository, creating a live link between the data source and the user interface. This direct connection ensures that any changes or new data in GitHub are immediately reflected in the auditing dashboards. To maintain a continuously improving system, CI/CD (Continuous Integration/Continuous Deployment) pipelines are implemented, specifically utilizing GitHub Actions. These pipelines automate the deployment of new machine learning models and application updates, ensuring that as soon as a new model is trained or code is refined, it can be seamlessly integrated into the live auditing system without manual intervention. The overarching goal of these interconnected elements is to achieve a self-sustaining, adaptable, and continuously improving auditing system that can evolve with data and model advancements while minimizing operational overhead.

3.6 CONCLUSION

The workflow begins with Data Collection, where renewable energy audit data is gathered from various sources and then uploaded to GitHub. This collected data is managed under Storage, maintaining structured datasets within GitHub repositories. Subsequently, the system moves to ML Processing, applying advanced machine learning algorithms to these datasets for accurate forecasting. Finally, for user interaction and insights, Deployment involves hosting interactive dashboards on Streamlit Cloud, ensuring Real-Time Accessibility for users to seamlessly interact with the energy forecasts.

CHAPTER 4

DEVELOPMENT OF HYBRID ENERGY FORECASTING TOOL USING MACHINE LEARNING

4.1 INTEGRATION OF RENEWABLE ENERGIES

The integration of solar and wind energy into modern power grids is crucial for sustainable energy management. However, renewable sources are inherently variable, influenced by weather patterns, geographical conditions, and demand fluctuations. This is where Machine Learning (ML) plays a transformative role helping optimize energy utilization, predict generation levels, and improve efficiency. The energy cost savings calculator is based on the principles of renewable energy utilization, electrical consumption, and economic feasibility. It combines solar and wind energy forecasting to determine potential electricity savings and CO_2 emission reductions.

4.2 SOLAR ENERGY UTILIZATION CALCULATIONS

4.2.1 Solar Energy Prediction & Electrical Characteristics

- The tool uploads solar irradiance data and forecasts future energy generation using historical data trend analysis.
- Electrical parameters such as voltage, current, and panel connections are computed using standard PV system principles.
- Machine learning concepts (random forest) are applied to simulate future irradiance.

4.2.2 Solar Cost Analysis Theory

- A financial model is integrated to estimate installation costs, including panel, inverter, battery, wiring, and labor expenses.
- Government subsidies are considered, helping users evaluate affordability for solar adoption.

- Cost estimations are based on scalability metrics used in industry financial modeling.

The solar system size is computed using the equation:

$$S = \frac{E}{H * \eta}$$

Where:

S = Recommended solar system size (in kW)

E = Daily energy consumption (kWh)

H = Peak sunlight hours per day (hours)

η = System efficiency factor (percentage converted to decimal)

The predicted solar energy output depends on the irradiance levels and panel efficiency:

$$E_{Solar} = \frac{I * \eta * a}{1000}$$

Where:

E_{solar} = Solar energy output per day (kWh)

I = Solar irradiance (W/m^2)

η = Solar panel efficiency (percentage converted to decimal)

A = Total panel area (m^2)

1000 = Conversion factor (Wh to kWh)

The total voltage and current depend on series and parallel panel arrangements:

$$V_{total} = V_{panel} \times N_{series}$$

$$I_{total} = I_{panel} \times N_{parallel}$$

Where:

V_{total} = Total voltage (Volts)

I_{total} = Total current (Amps)

V_{panel} = Voltage per panel (typically ~30V)

I_{panel} = Current per panel (typically ~8A)

N_{series} = Number of panels in series

$N_{parallel}$ = Number of panels in parallel

The total cost of solar installation includes panel costs, inverter, battery storage, and miscellaneous expenses:

$$C_{total} = (C_{panel} + C_{inverter} + C_{battery} + C_{structure} + C_{wiring} + C_{installation})$$

Where:

C_{total} = Total installation cost ($ or ₹)

C_{panel} = Cost per watt × System size × 1000 (conversion kW to W)

$C_{inverter}$ = Inverter cost

$C_{battery}$ = Battery capacity (kWh) × Cost per kWh

$C_{structure}$ = Structural mounting cost

C_{wiring} = Wiring and accessories cost

$C_{installation}$ = Labor installation cost

4.3 WIND ENERGY UTILIZATION CALCULATIONS

4.3.1 Future Wind Speed Forecasting

- The tool uploads real wind speed data and generates predictions for future wind energy availability.
- Using historical trends and statistical models, the system estimates wind power generation from 2025–2030, helping users optimize turbine usage.

4.3.2 Electrical Characteristics Analysis

- The code computes turbine electrical output, including total wind power and kWh/day estimation.
- Multiple turbines can be simulated to assess large-scale energy generation capacity.
- Efficiency factors are applied to improve realistic wind power calculations

The fundamental equation for wind power output is:

$$P=\frac{1}{2}\times\rho\times A\times V^3\times\eta$$

Where:

P = Wind power output (Watts)

ρ = Air density (kg/m³), typically 1.225 kg/m³

A = Swept rotor area per turbine (m²)

V = Wind speed (m/s)

η = Efficiency factor (percentage converted to decimal)

To convert power output into usable energy, the formula is:

$$E_{Turbine} = \frac{P * T}{1000}$$

Where:

$E_{Turbine}$= Energy output per turbine per day (kWh)

P = Wind power (Watts)

T = Total time in hours (24 hours/day)

1000 = Conversion factor from Watt-hours to kWh

To determine total energy generation for multiple turbines:

$$E_{total} = E_{turbine} \times N$$

Where:

E_{total} = Total wind energy output (kWh/day)

$E_{turbine}$ = Energy produced by a single turbine

N = Number of turbines in operation

4.4 ENERGY COST SAVINGS TOOL

4.4.1 Monthly solar energy generation

Solar energy production relies on panel size and peak sunlight hours**:**

$$E_{solar} = S \times H \times D$$

Where:

E_{solar} = Monthly solar energy generation (kWh)

S = Solar system size (kW)

H = Peak sunlight hours per day (assumed 4.5 hours)

D = Number of days per month (30)

4.4.2 Monthly wind energy generation

Wind energy output depends on turbine capacity, air density, wind speed, and efficiency:

$$E_{wind} = 0.5 \times \rho \times V \times \frac{T}{1000} \times \eta \times H \times D$$

Where:

E_{wind} = Monthly wind energy generation (kWh)

ρ = Air density (kg/m³), assumed 1.225 kg/m³

V = Average wind speed (m/s)

T = Wind turbine capacity (kW)

η = Efficiency factor (percentage converted to decimal)

H = Hours per day (24)

D = Number of days per month (30)

4.4.3 Total energy generated by solar & wind

$$E_{total} = E_{solar} + E_{wind}$$

Where:

E_{total} = Combined energy output (kWh)

4.4.4 Energy cost savings calculation

- ✓ **TNERC Tariff Rates (July 2024 Order)** are encoded for various **consumer categories**, such as:
- ✓ **Industries, Institutions, Domestic Consumers, Agriculture, and Construction**.
- ✓ **Different pricing structures** are defined based on **energy consumption slabs**, ensuring real-world applicability.

The savings depend on energy consumption Vs renewable generation**:**

$$\text{Savings} = (E_{total} - E_{used}) \times C$$

Where:

E_{total}= Total energy generated from solar & wind

E_{used}= User's monthly electricity consumption (kW)

C = Cost per kWh based on consumer category

CO_2 Emission Reduction Calculation

Each unit of renewable-generated electricity replaces fossil fuel consumption, leading to a reduction in carbon emissions. The CO_2 savings are estimated by multiplying the total renewable energy generated by 0.82 kg/kWh, representing the average emission offset per unit of clean energy.

This calculator provides a data-driven approach to assess energy efficiency, financial benefits, and environmental impact, enabling informed decision-making for

sustainable energy adoption. The CO_2 saved is estimated based on the assumption that each kWh offsets 0.82 kg of CO_2 emissions:

$$CO_2 = E_{total} \times 0.82$$

Where:

CO_2 = Carbon dioxide emission reduction (kg)

4.5 EVALUATION METRICS FOR ENERGY AUDITING MODELS

To ensure accurate forecasting, efficient renewable energy utilization, and financial impact assessment, we use specific evaluation metrics tailored for each aspect of energy auditing. Model Accuracy Metrics (For Solar & Wind Predictions) These metrics evaluate the precision and reliability of forecasting models.

4.5.1 Mean Absolute Error (MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \overline{y_i}|$$

Where,

n = Total number of data points (predictions)

y_i= Actual value (true value) for the i-th observation

$\overline{y_i}$ = Predicted value for the i-th observation

$|y_i - \overline{y_i}|$ = Absolute error for the i-th prediction

Measures the average absolute difference between actual and predicted energy output. Lower MAE values indicate better forecasting accuracy, reducing deviations in solar/wind predictions.

4.5.2 Mean Squared Error (MSE)

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

Where,

n = Total number of data points(predicted)

y_i= Actual value for the i-th observation

$\hat{y}_i$ = Predicted value for the i-th observation

Penalizes larger errors more heavily, ensuring that significant deviations in energy forecasting are minimized. Helps in identifying potential outliers affecting predictive efficiency.

4.5.3 R^2 Score (Coefficient of Determination)

$$SS_{res} = \sum (y_i - \hat{y}_i)\,2$$

$$SS_{tot} = \sum (y_i - \bar{y}_i)2$$

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

Where,

y_i = Actual value

$\hat{y}_i$ = Predicted value

$\bar{y}_i$= mean of actual vale

SS_{res}= sum of squared residuals

SS_{tot} = total sum of squares

Determines how well predictions align with actual energy consumption trends and indicates a stronger model fit, meaning the forecasted energy generation closely matches real-world data.

4.6 SOFTWARE MODEL

4.6.1 Python IDLE-Pycharm

Python IDLE (Integrated Development and Learning Environment) is a lightweight and user-friendly IDE that comes bundled with Python, making it accessible for beginners and experienced developers alike. It offers an interactive shell for executing code snippets, along with a simple editor that features syntax highlighting, automatic indentation, and basic debugging capabilities. IDLE is particularly useful for writing small scripts, testing functions, and learning Python programming due to its straightforward interface. While it lacks the advanced features of professional-grade IDEs like PyCharm or VS Code, its simplicity and direct integration with Python make it a convenient choice for quick development and experimentation.

PyCharm which is shown in Fig 4.1 is a robust Integrated Development Environment (IDE) designed for Python programming, offering a seamless platform for development, execution, and debugging. It streamlines the coding process by providing intelligent code completion, built-in debugging tools, and an intuitive interface that aids in efficient error analysis. With its powerful features like integrated testing frameworks, version control support, and database connectivity, PyCharm enables developers to write, test, and refine their code with precision. Whether handling complex applications or optimizing algorithms, PyCharm ensures a smooth workflow, making it an ideal choice for software development and research projects.

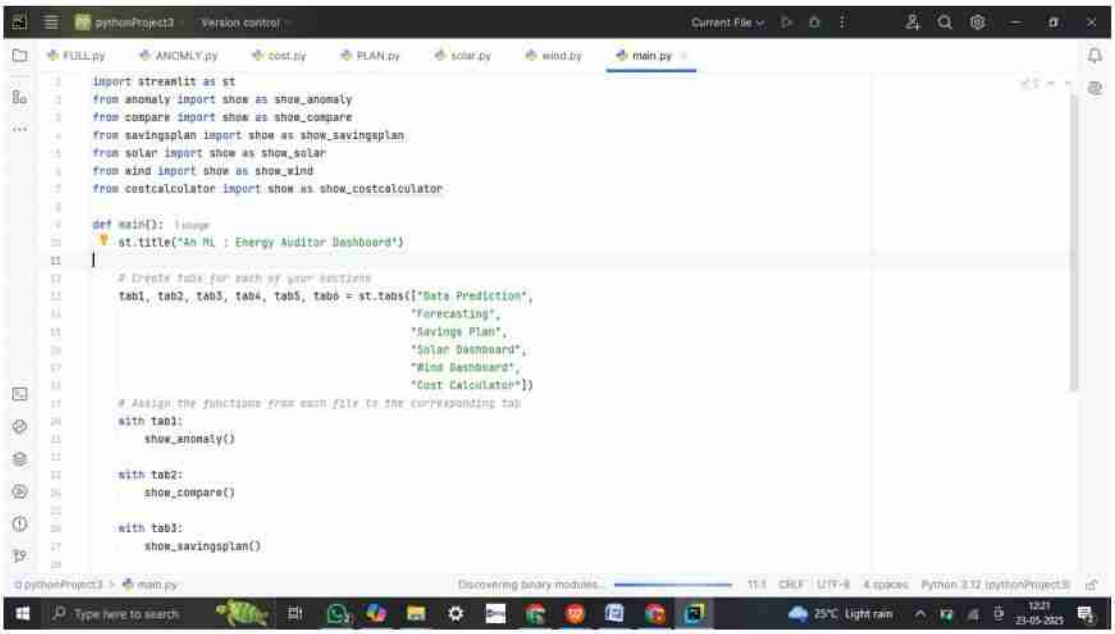


Fig 4.1 Pycharm community edition

4.6.2 Github

GitHub serves as an efficient hosting platform for managing code, enabling seamless integration with Streamlit Cloud for deployment. By using GitHub, developers can store, update, and track changes to their code, ensuring version control and collaboration. When connected to Streamlit Cloud, GitHub allows automatic syncing, meaning any modifications pushed to the repository are instantly reflected in the deployed application. This setup ensures smooth workflow management, making it easier to maintain and update projects without manual intervention. Additionally, GitHub offers both public and private repositories, providing flexibility in code access and security. Through this integration, developers can efficiently handle application development, debugging, and deployment, ensuring their projects remain organized and accessible.Fig 4.2 shows the main code page of the project

To set up GitHub for Streamlit Cloud

1. Create a GitHub repository and upload your Streamlit project files.
2. Connect GitHub to Streamlit Cloud by authorizing access.
3. Deploy your app , ensuring it updates whenever you push new code.

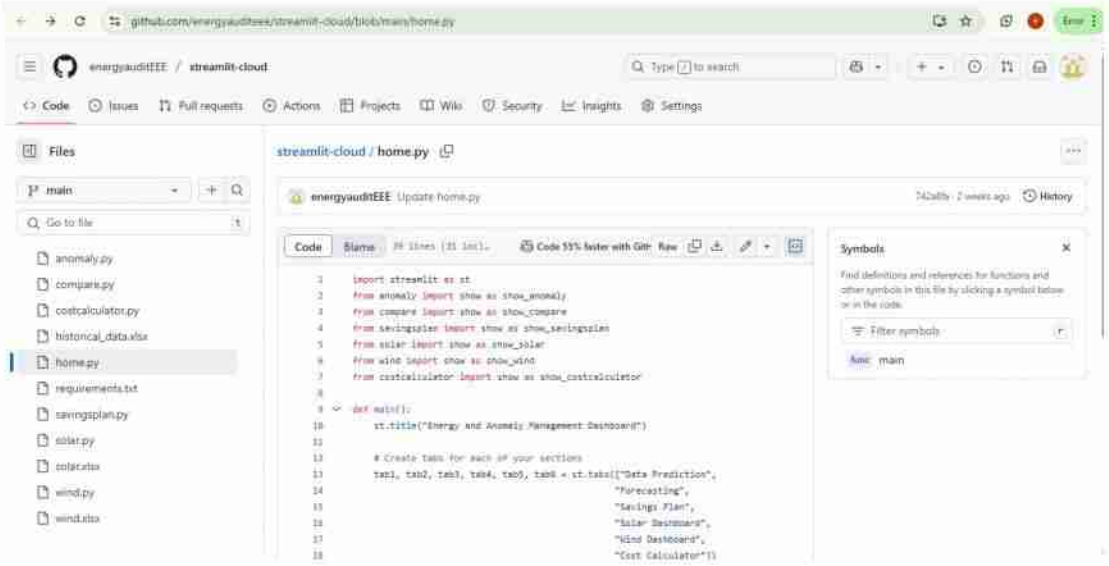


Fig 4.2 GITHUB repository

4.6.3 Streamlit

Streamlit which is shown in Fig 4.3 is an open-source Python framework designed for building interactive and data-driven web applications with minimal effort. It allows developers to create highly dynamic and visually appealing apps using only Python, eliminating the need for complex front-end development. Streamlit provides an intuitive way to integrate data visualization, machine learning models, and interactive widgets such as sliders and buttons, making it ideal for rapid prototyping. With its real-time updates and seamless integration with popular Python libraries like Matplotlib, Plotly, and Pandas, Streamlit is widely used in fields such as data science, artificial intelligence, and research. Additionally, it offers convenient deployment options, including Streamlit Cloud, enabling users to share applications effortlessly. Its simplicity and powerful features make it a preferred choice for developers seeking to create interactive apps quickly and efficiently.

Fig 4.3 Cloud deployment Streamlit page

4.6.4 Libraries used

Streamlit – A framework for building interactive web applications using Python. It simplifies the process of creating data-driven apps without requiring front-end development.

Pandas – A powerful library for data manipulation and analysis, offering flexible data structures like Data-frames for handling structured data efficiently.

Matplotlib – A widely used plotting library for creating static, animated, and interactive visualizations in Python.

Seaborn – Built on top of Matplotlib, Seaborn specializes in statistical data visualization, making it easier to create aesthetically pleasing and informative plots.

Scikit-learn – A machine learning library that provides simple and efficient tools for data mining, analysis, and predictive modeling.

OpenPyxl – A library for reading, writing, and modifying Excel files, allowing Python users to work with spreadsheets programmatically.

4.7 CONCLUSION

This chapter discuss about step -by- step approach of the project starting from data collection to cloud deployment .It discuss on the various forecasting methods used and also machine learning algorithms enhance this process by identifying anomalies, predicting consumption trends, and recommending optimization strategies. This integration not only improves energy efficiency but also reduces operational costs and environmental impact.

CHAPTER 5

RESULTS

5.1 INTRODUCTION

The Energy & Anomaly Management Dashboard shown in Fig 5.1 is an intuitive Streamlit application designed to provide a comprehensive, integrated solution for advanced energy data analysis and strategic management. Accessible via a unified platform, this dashboard empowers users with granular control and insights across various energy facets.

Key Features Include:

- ✓ **Data Prediction & Anomaly Detection:** Identify and analyze unusual patterns or critical deviations within energy datasets, crucial for maintaining system health and detecting potential issues.
- ✓ **Energy Forecasting:** Leverage predictive analytics to forecast future energy consumption trends, aiding in proactive resource planning and demand management.
- ✓ **Energy Savings Plan:** Develop, track, and optimize energy-saving initiatives, enabling users to identify opportunities for efficiency and cost reduction.
- ✓ **Solar Energy Dashboard:** Gain in-depth insights into solar energy system performance, production, and potential, supporting renewable energy optimization.
- ✓ **Wind Energy Dashboard:** Explore data-driven insights into wind energy potential and operational performance, facilitating informed decisions on wind power integration.
- ✓ **Energy Cost Calculator:** Estimate electricity costs and quantify potential savings from renewable energy adoption or efficiency measures, supporting financial planning.

This integrated dashboard offers a user-friendly interface, providing a holistic and actionable approach to energy management, from anomaly detection to strategic planning and renewable energy integration.

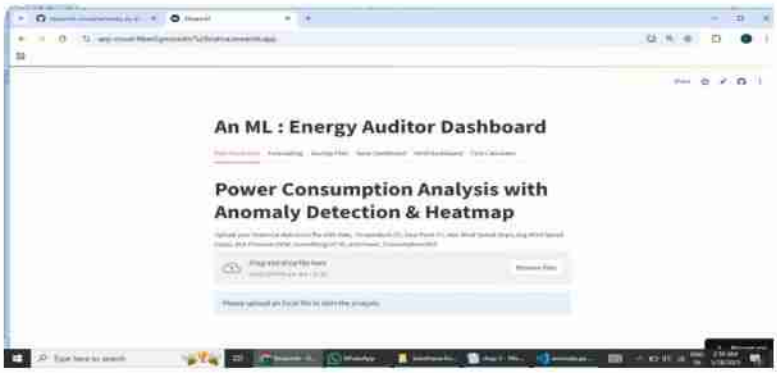


Fig 5.1 The Streamlit Cloud's Energy Audit Dashboard

Access Dashboard here: https://app-cloud-fkkex5gmsredm7ur8oqhva.streamlit.app/

5.2 MISSING DATA ANALYSIS

This Streamlit application is designed for comprehensive power consumption analysis, including missing data imputation and anomaly detection. It provides an intuitive interface for users to upload their historical energy data and visualize key insights. It's important to note that the application preprocesses the data by removing any rows that have missing values in the weather-related columns before proceeding with power consumption prediction. The application requires an Excel file as input. This file must contain the following columns:

- ✓ **DATE:** Timestamp for each data point.
- ✓ **Power_Consumption(MU):** The core metric for power consumption (e.g., in Million Units). This column is where missing values will be addressed.
- ✓ **Temperature (F):** Environmental temperature.
- ✓ **Dew Point (F):** Dew point temperature.

- ✓ **Max Wind Speed (mps):** Highest wind speed recorded.
- ✓ **Avg Wind Speed (mps):** Average wind speed.
- ✓ **Atm Pressure (hPa):** Atmospheric pressure.
- ✓ **Humidity(g/m^3):** Humidity levels.

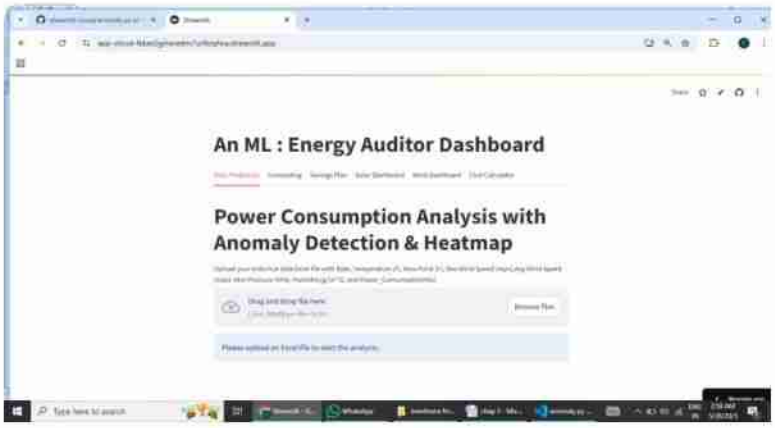


Fig 5.2 Data Prediction Dashboard from Streamlit to compute the Missing data

Upon uploading a suitable Excel file, the application generates several key outputs, Heatmap of Predicted Missing Power Consumption. This heatmap visually represents the average predicted power consumption for dates that initially had missing data. It helps in identifying any underlying daily or monthly patterns within the imputed values, validating the prediction quality.

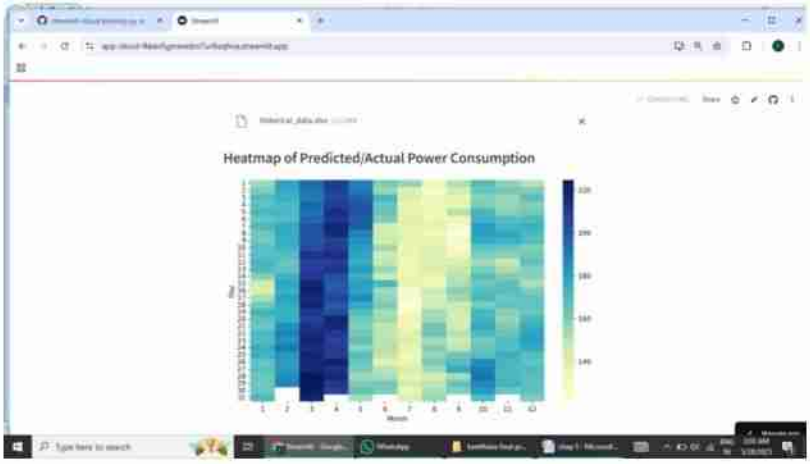


Fig 5.3 Heatmap of Missing and Present Power Consumption Values for the year 2024

The Random Forest Regressor is trained using known power consumption values and correlated weather features. It then predicts the missing power consumption values based on the weather conditions associated with those missing dates, providing an informed imputation rather than a simple statistical average.

Code for predicting missing values:

```
# From predict_missing_power function
if not X_missing.empty:
    model = RandomForestRegressor(n_estimators=200, random_state=42)
    model.fit(X_known, y_known) # Training on known data
    missing_data['Power_Consumption(MU)'] = model.predict(X_missing)
```

Power Consumption Trend with Anomaly Detection. A time-series plot displaying power consumption over time. This chart highlights the overall power consumption trend (including imputed values).Originally missing data points that have now been predicted (marked in green) to show their integration into the series. Detected anomalies (marked in red), indicating unusual consumption patterns that warrant further investigation.

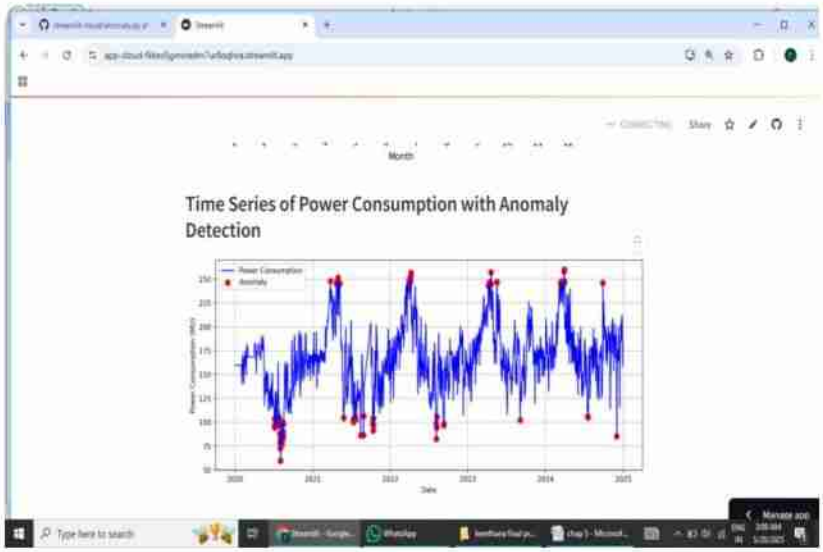


Fig 5.4 Anomaly detection of Power consumption graph to verify the abnormalities

Fig 5.5 Shows (first 10 rows) of the processed data, including the original measurements, the newly predicted power consumption values, a flag indicating if a value was originally missing (was_missing), and the anomaly status (anomaly). A convenient button to download the entire processed dataset (including original,imputed, and anomaly-flagged data) as a CSV file.

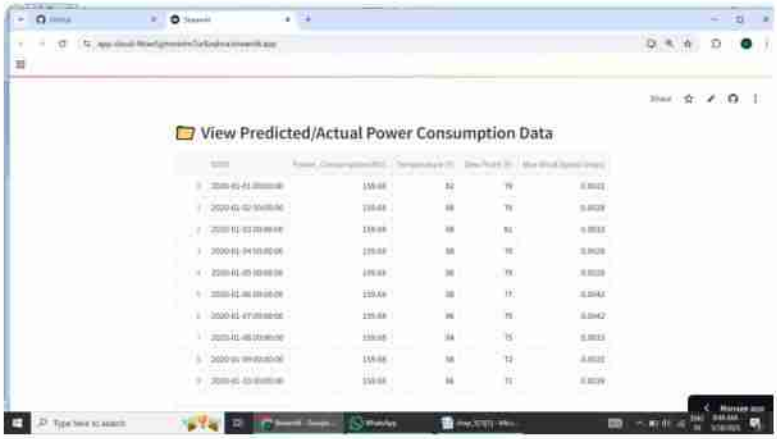


Fig 5.5 Sample Table from the Data prediction Dashboard

After the power consumption data is complete (post-imputation), Isolation Forest is applied to the entire series. It flags data points that statistically stand out from the majority, allowing users to investigate these potential issues. It operates on the principle that anomalies are few and different, making them "easier to isolate" than normal data points. It does this by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of that selected feature.

Code for detecting anomalies:

```
# From detect_anomalies function
iso_forest = IsolationForest(contamination=0.05, random_state=42)
data['anomaly'] = iso_forest.fit_predict(data[['Power_Consumption(MU)']]) # Applies the model and labels anomalies
anomalies = data[data['anomaly'] == -1] # Filters for identified anomalies
```

5.3 ENERGY FORECASTING USING THREE MODELS

5.3.1 Data preparation

This Streamlit application provides a comparative framework for forecasting power consumption using various machine learning models. It enables users to upload their historical data, receive forecasts, and evaluate model performance.

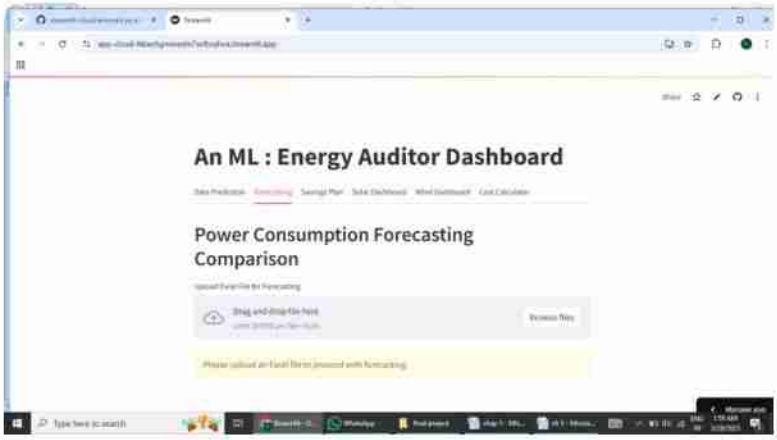
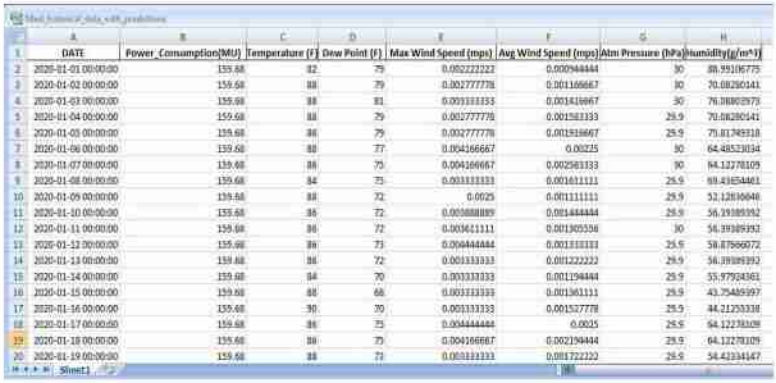


Fig 5.6 Future Forecasting Dashboard for Forecasting Power Consumption for 2025 to 2030

The application requires an Excel file as input. This file must contain the following columns:

- ✓ **DATE**: This column represents the date and time of each power consumption reading. It's crucial for creating time-based features and for visualizing the forecast.
- ✓ **Power_Consumption(MU):** This is the target variable, representing the power consumption (likely in Million Units, MU) that the models will predict.
- ✓ **Temperature (F):** Environmental temperature.
- ✓ **Dew Point (F):** Dew point temperature.
- ✓ **Max Wind Speed (mps):** Highest wind speed recorded.
- ✓ **Avg Wind Speed (mps):** Average wind speed.
- ✓ **Atm Pressure (hPa):** Atmospheric pressure.
- ✓ **Humidity(g/m^3):** Humidity levels.



	A	B	C	D	E	F	G	H
1	DATE	Power_Consumption(MU)	Temperature (F)	Dew Point (F)	Max Wind Speed (mps)	Avg Wind Speed (mps)	Atm Pressure (hPa)	Humidity(g/m^3)
2	2020-01-01 00:00:00	159.68	82	79	0.052222222	0.000944444	30	88.95106775
3	2020-01-02 00:00:00	159.68	88	79	0.002777778	0.00116667	30	70.08260141
4	2020-01-03 00:00:00	159.68	88	81	0.003333333	0.00141667	30	76.38802973
5	2020-01-04 00:00:00	159.68	88	79	0.002777778	0.00158333	29.9	70.08260141
6	2020-01-05 00:00:00	159.68	86	79	0.002777778	0.00191667	29.9	75.81749318
7	2020-01-06 00:00:00	159.68	88	77	0.004166667	0.00225	30	64.48523034
8	2020-01-07 00:00:00	159.68	86	75	0.004166667	0.00258333	30	64.12278109
9	2020-01-08 00:00:00	159.68	84	75	0.003333333	0.00161111	29.9	69.43654461
10	2020-01-09 00:00:00	159.68	88	72	0.0025	0.00111111	29.9	52.12830646
11	2020-01-10 00:00:00	159.68	86	72	0.00288889	0.00144444	29.9	56.39389392
12	2020-01-11 00:00:00	159.68	86	72	0.003611111	0.00130556	30	56.39389392
13	2020-01-12 00:00:00	159.68	86	73	0.004444444	0.00133333	29.9	58.87666072
14	2020-01-13 00:00:00	159.68	86	72	0.003333333	0.00122222	29.9	56.39389392
15	2020-01-14 00:00:00	159.68	84	70	0.003333333	0.00119444	29.9	55.97924361
16	2020-01-15 00:00:00	159.68	88	68	0.003333333	0.00136111	29.9	43.75485997
17	2020-01-16 00:00:00	159.68	90	70	0.003333333	0.00152778	29.9	44.21253338
18	2020-01-17 00:00:00	159.68	86	75	0.004444444	0.0025	29.9	64.12278109
19	2020-01-18 00:00:00	159.68	86	75	0.004166667	0.00219444	29.9	64.12278109
20	2020-01-19 00:00:00	159.68	88	73	0.003333333	0.00172222	29.9	54.42334147

Fig 5.7 Input Data for Power Consumption Forecasting

5.3.2 Model training and prediction

Upon uploading a suitable Excel file and initiating a forecast, the application delivers Individual Model Forecast Results, allowing users to select a single model (ANN, RF, or SVM) from the sidebar to inspect its specific performance. For the chosen model, it displays key Performance Metrics like Mean Squared Error (MSE), R-squared (R^2), and Mean Absolute Error (MAE), alongside a Forecasted Future Data Chart visualizing predicted power consumption for the next five years. Users can further explore detailed comparisons via expandable tables for Predicted vs. Actual Data and Forecasted Future Data, each showing the first 50 rows. To provide deeper insight into model reliability, an Error Distribution Plot (histogram with KDE) illustrates prediction error spread, while a Daily Power Consumption Changes Plot reveals day-over-day historical shifts, aiding in identifying volatility and trends.

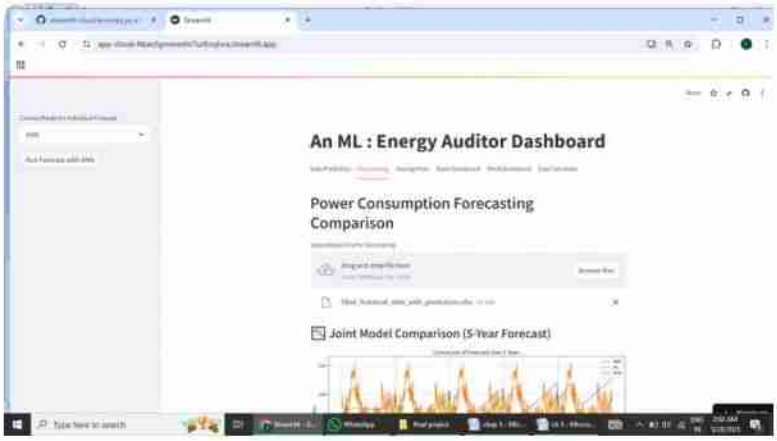


Fig 5.8 Forecasted Power Consumption Dashboard

Artificial Neural Networks (ANN)

From the sidebar, the ANN model is chosen and proceed to run forecast and it will shows the performance of the ANN model for the given dataset. An MLP Regressor from scikit-learn was used to build the ANN model. The model is trained for a maximum of 1000 iterations with a random state for reproducibility.

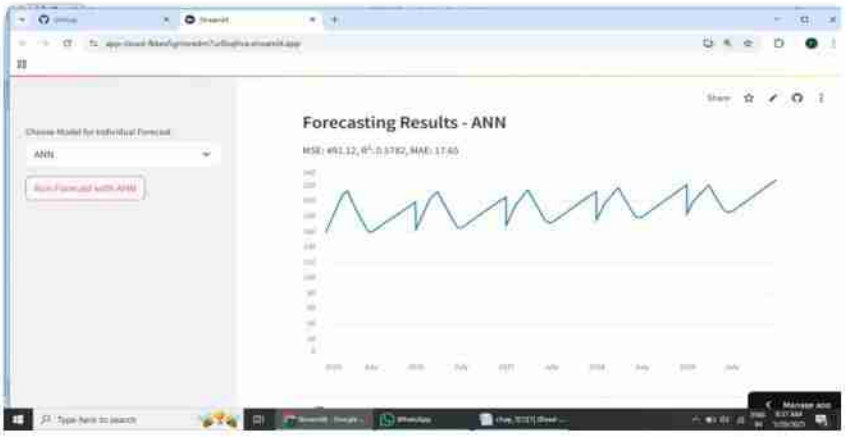


Fig 5.9 Forecasting results of ANN for Power Consumption

Fig 5.9 shows the plot for forecasting of power consumption for the year 2025 to 2030. Presents a line chart visualizing the power consumption forecast over the next five years.

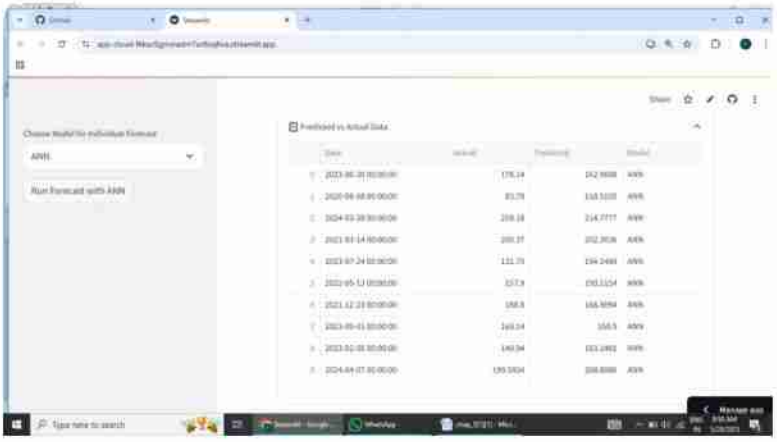


Fig 5.10 Actual vs Predicted (Sample) values of ANN for Power Consumption

Fig 5.10 shows a table displaying a sample (e.g., first 10 rows) of the historical data's actual power consumption values alongside the power consumption values predicted by the ANN model for those dates.

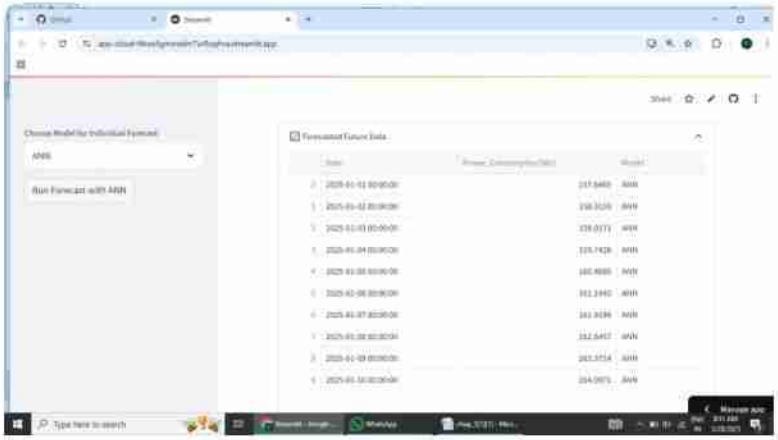


Fig 5.11 Forecasted Power Consumption values (Sample) of ANN

Fig 5.11 shows a table presenting a sample (e.g., first 10 rows) of the ANN model's predicted power consumption values for the future dates (next five years).

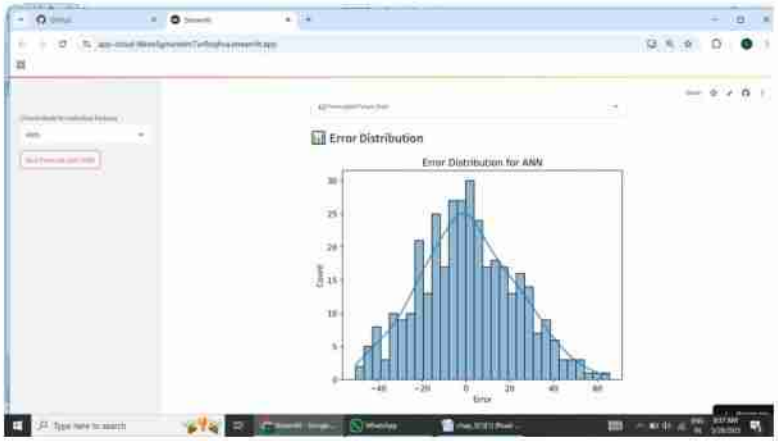


Fig 5.12 Error distribution of ANN for Metrics calculation

Fig 5.12 displays a histogram showing the distribution of the differences (errors) between the actual and predicted power consumption values from the test set. A Kernel Density Estimate (KDE) curve might be overlaid.

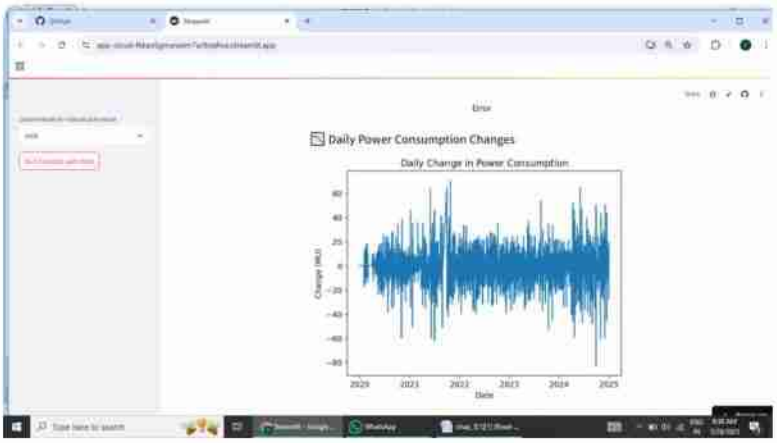


Fig 5.13 Daily change in power consumption of ANN from Collected Data

Fig 5.13 presents a line plot illustrating the day-to-day changes in the historical power consumption data.

Random Forest (RF)

From the sidebar, the RF model is chosen and proceed to run forecast and it will shows the performance of the RF model for the given dataset shown in Fig 5.14. A Random Forest Regressor from scikit-learn was employed. The model consists of 100 decision trees, and a random state is set for consistent results.

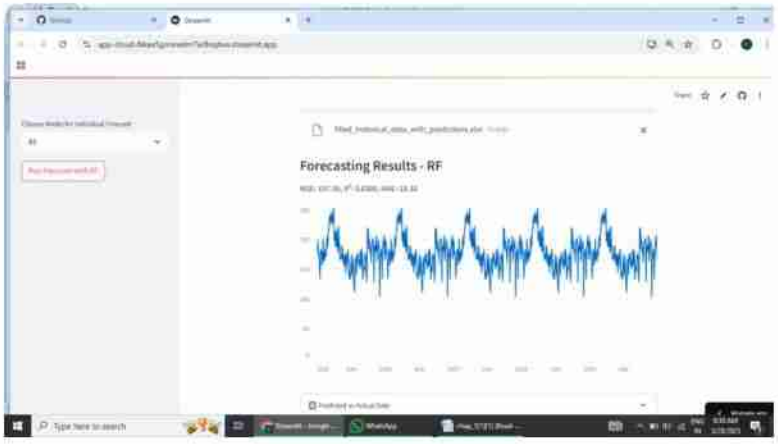


Fig 5.14 Forecasting results of RF for Power Consumption

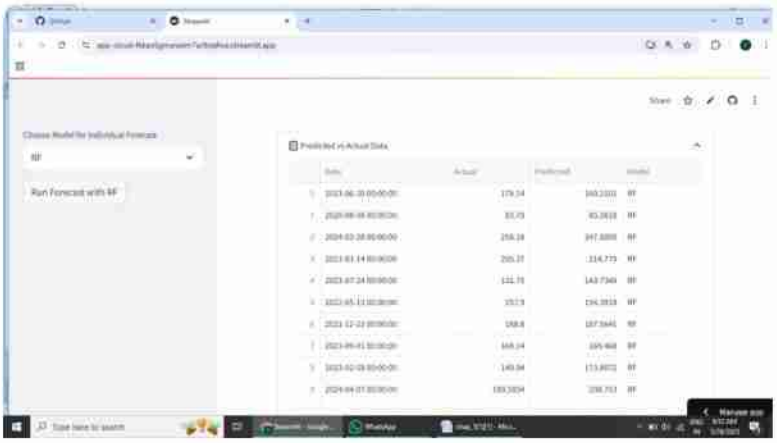


Table 5.15 Actual vs Predicted (Sample) values of RF

Table 5.15 shows an expandable table displaying a sample (e.g., first 10 rows) of the historical data's actual power consumption values alongside the power consumption values predicted by the Random Forest model for those dates.

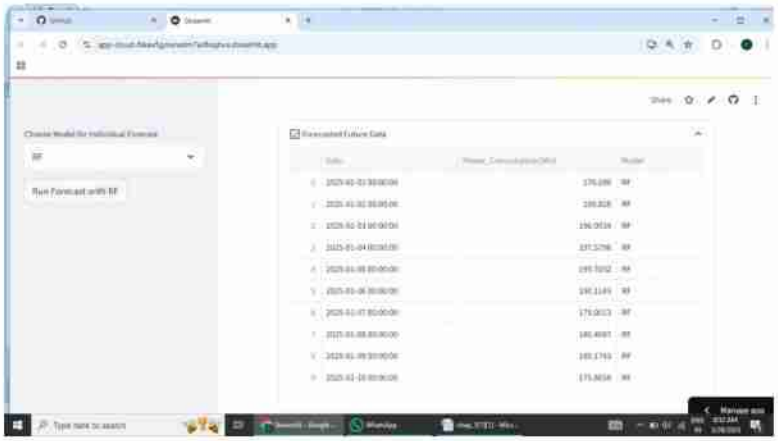


Fig 5.16 Forecasted Power Consumption values (Sample) of RF

Fig 5.16 shows an expandable table presenting a sample (e.g., first 10 rows) of the Random Forest model's predicted power consumption values for the future dates (next five years).

The distribution of the differences (errors) between the actual and predicted power consumption values from the test set for the Random Forest model. A Kernel Density Estimate (KDE) curve might be overlaid shown in Fig 5.17.

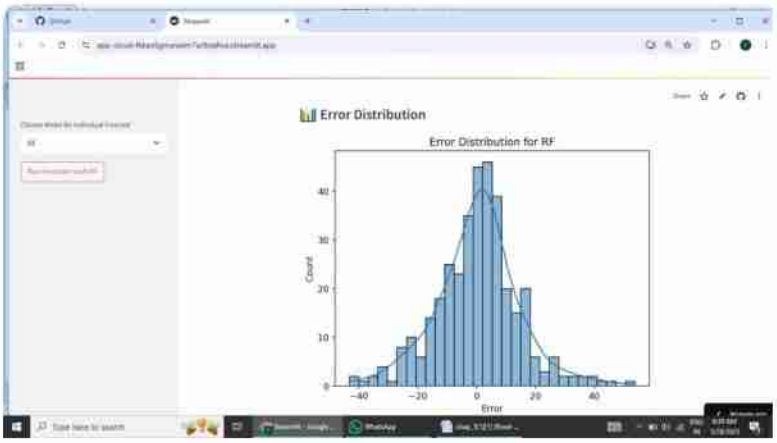


Fig 5.17 Error distribution of RF for Metrics Calculation

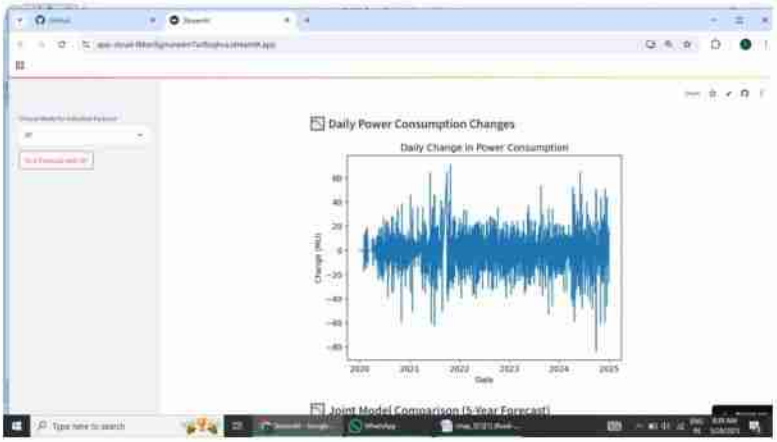


Fig 5.18 Daily change in power consumption of RF

Fig 5.18 presents the same line plot as above, illustrating the day-to-day changes in the historical power consumption data (this plot is based on the original data, not the model).

Support Vector Machine (SVM)

From the sidebar, the SVM model is chosen and proceed to run forecast and it will shows the performance of the SVM model for the given dataset. An SVR model from scikit-learn was utilized with a radial basis function (RBF) kernel. The hyper parameters C (regularization parameter), gamma (kernel coefficient), and epsilon (tube width) were set to specific values (C=100, gamma=0.1, epsilon=0.1).For each model,

the training data is used to fit the model parameters. After training, the models are used to predict power consumption on the test data and then used for future forecasting.

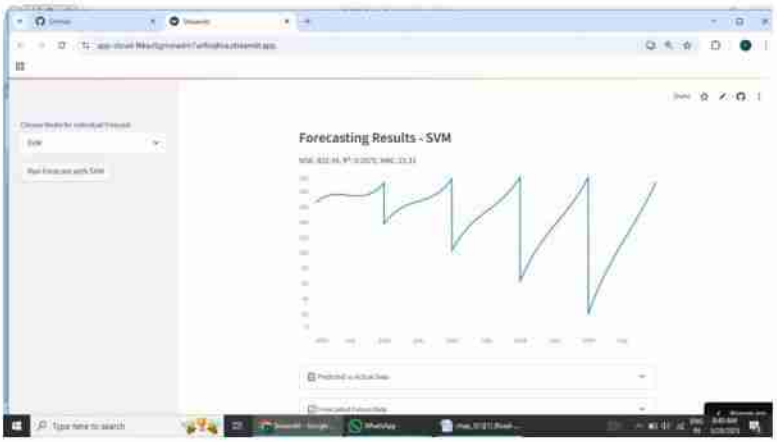


Fig 5.19 Forecasting results of SVM for Power Consumption

Fig 5.19 displays a line chart visualizing the power consumption forecast over the next five years generated by the Support Vector Regression.

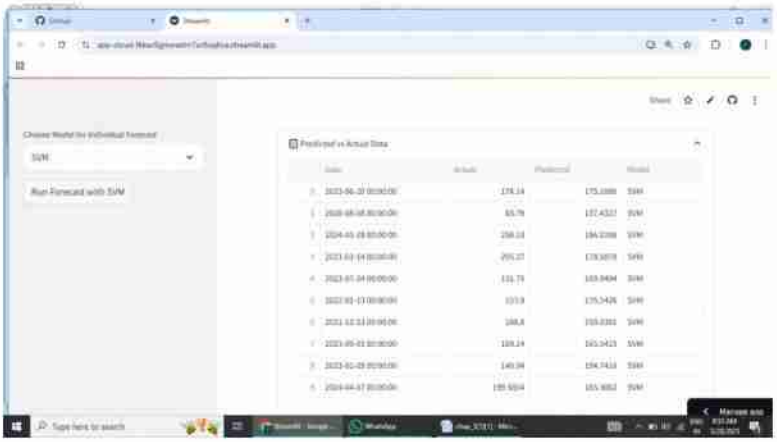


Fig 5.20 Actual vs Predicted (Sample) values of SVM

Fig 5.20 shows an expandable table displaying a sample (e.g., first 10 rows) of the historical data's actual Vs the predicted power consumption values .

Expandable table presenting a sample (e.g., first 10 rows) of the SVM model's predicted power consumption values for the future dates shown in Fig 5.21.

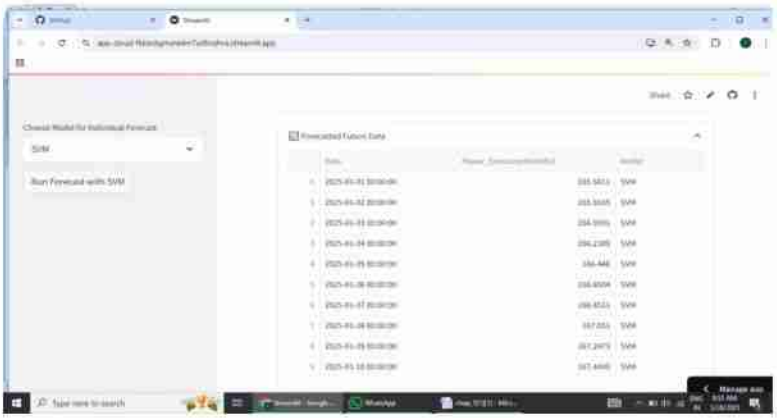


Fig 5.21 Forecasted Power Consumption values (Sample) of SVM

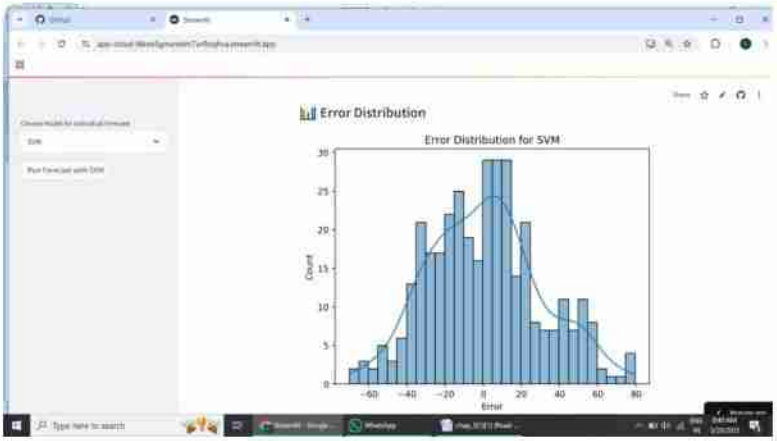


Fig 5.22 Error distribution of SVM for Metrics Calculation

Fig 5.22 displays a histogram showing the distribution of the differences (errors) between the actual and predicted power consumption values.

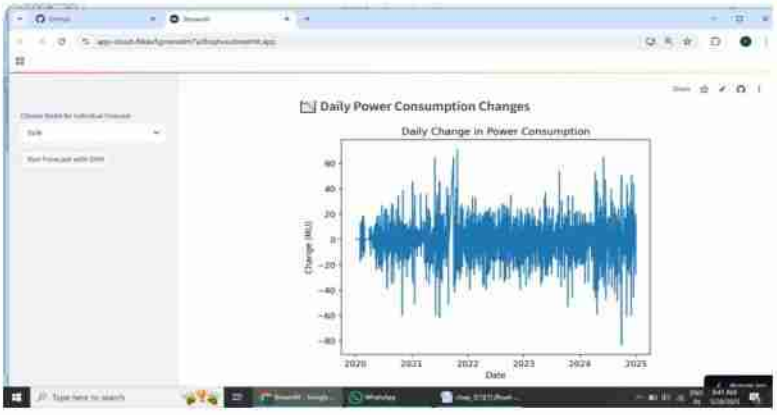


Fig 5.23 Daily change in power consumption of SVM

Fig 5.23 presents the same line plot as above, illustrating the day-to-day changes in the historical power consumption data.

Code for forecasting of models:

```
forecast_df = pd.DataFrame({
    'Date': future_dates,
    'Power_Consumption(MU)': future_predictions.flatten(),
    'Model': model_name
})
predicted_df = pd.DataFrame({
    'Date': pd.to_datetime(dict(
        year=(X_test[:, 0] * (x_scaler.data_max_[0] - x_scaler.data_min_[0]) +
x_scaler.data_min_[0]).astype(int),
        month=1, day=1)) +
            pd.to_timedelta(((X_test[:, 1] * (x_scaler.data_max_[1] -
x_scaler.data_min_[1]) + x_scaler.data_min_[
                1]) - 1).astype(int), unit='D'),
    'Actual': y_test_inv.flatten(),
    'Predicted': y_pred.flatten(),
    'Model': model_name
})
```

5.3.3 Model evaluation

The model evaluation metrics table will appear when these models are forecasting the power consumption datas for the given datas. The performance of each forecasting model is evaluated using three common regression metrics.

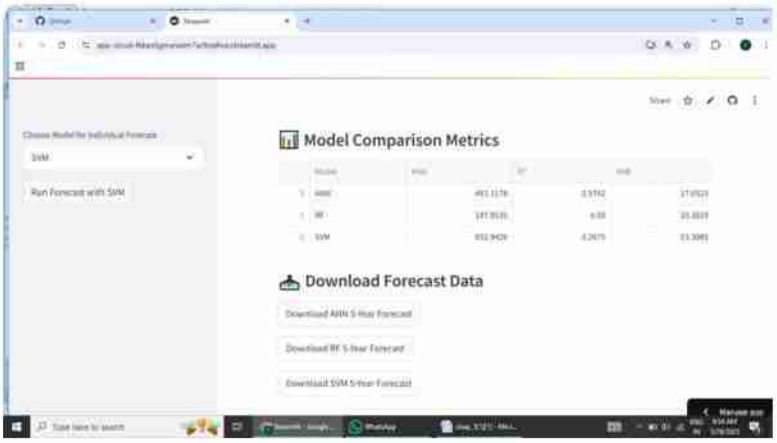


Fig 5.24 Evaluation Metrics

Fig 5.24 Displays calculated Mean Squared Error (MSE), R-squared (R^2), and Mean Absolute Error (MAE) values, quantifying the model's performance.

Based on the evaluation metrics, the Random Forest (RF) model demonstrates the strongest forecasting performance with the lowest Mean Squared Error (MSE) of 197.9535 and Mean Absolute Error (MAE) of 10.3819, alongside a high R-squared (R^2) value of 0.83, indicating that it explains a significant portion of the variance in power consumption. The Artificial Neural Network (ANN) model shows moderate performance, while the Support Vector Regression (SVR) model exhibits the weakest predictive capability with the highest MSE and MAE, and the lowest R-squared value. Therefore, for predicting power consumption in this context, the Random Forest model appears to be the most accurate and reliable choice among the evaluated models.

Code for calculation of errors:

```
y_pred_scaled = model.predict(X_test)
    y_pred = y_scaler.inverse_transform(y_pred_scaled.reshape(-1, 1))
    y_test_inv = y_scaler.inverse_transform(y_test)
    mse = mean_squared_error(y_test_inv, y_pred)
    r2 = r2_score(y_test_inv, y_pred)
```

mae = mean_absolute_error(y_test_inv, y_pred)

5.3.4 Visualization and comparision

The forecasted power consumption for each model over the five-year period is visualized using Matplotlib, allowing for a direct comparison of the trends predicted by the ANN, RF, and SVM. whose future trends is plotted in Fig 5.25, evaluating their performance, and visualizing the predicted trends for future energy planning.

Fig 5.25 Model comparision of three models for the forecasted power consumption values

Additionally, the evaluation metrics (MSE, R^2, MAE) for each model are presented in a table to facilitate a quantitative comparison of their performance. The distribution of prediction errors for a selected individual model is also visualized using a histogram with a kernel density estimate (KDE) using Seaborn. Furthermore, the daily changes in historical power consumption are plotted to provide context on the inherent variability of the data. The forecasted data for each model is also made available for download in Excel format shown in Fig 5.26. This module provides a comprehensive approach to forecasting future power consumption by implementing and comparing multiple machine learning models.

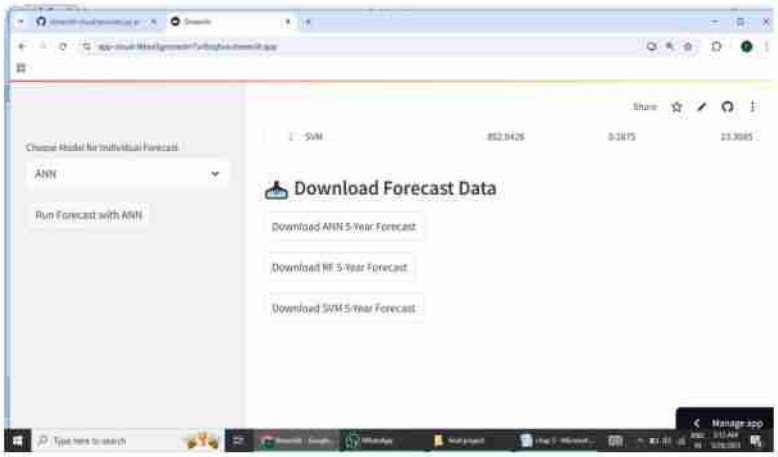


Fig 5.26 Forecasted Data Download Button

5.4 BASIC ENERGY SAVINGS PLAN

This dashboard provides users with tailored energy-saving guidelines categorized by sector, including Education, Industry, and Residential shown in Fig 5.27 .

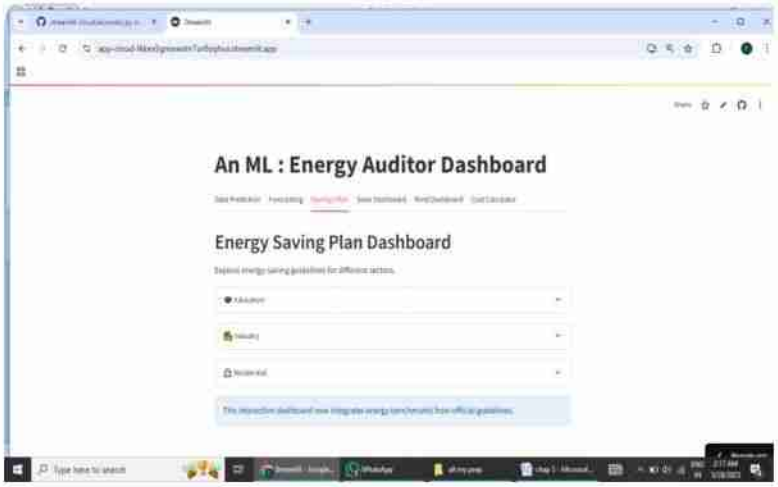


Fig 5.27 Energy savings plan dashboard

For the Education sector, a list of general energy-saving practices is presented, encouraging behavioral changes and infrastructure upgrades. These guidelines are shown in Fig 5.28 aim to optimize energy usage in manufacturing processes and equipment.

Fig 5.28 Energy Savings Plan Dashboard For Education Sector

The Industry sector offers specific recommendations aligned with energy conservation guidelines for Cement, Iron & Steel, and Textile industries, incorporating industry benchmarks for parameters like air ratio in kilns, boiler efficiency, power factor correction, and lighting power density. These guidelines are shown in Fig 5.29 aim to optimize energy usage in manufacturing processes and equipment.

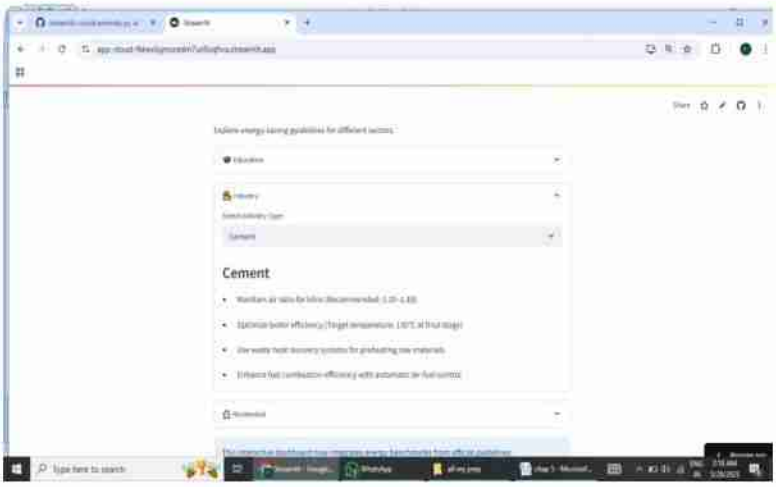


Fig 5.29 Energy savings plan of industrial sector

The Residential sector (shown in Fig 5.30) is further divided into Apartments, Houses, and Neighborhoods, offering context-specific advice ranging from individual appliance choices and usage habits to community-level initiatives like solar-powered street lighting and energy audits.

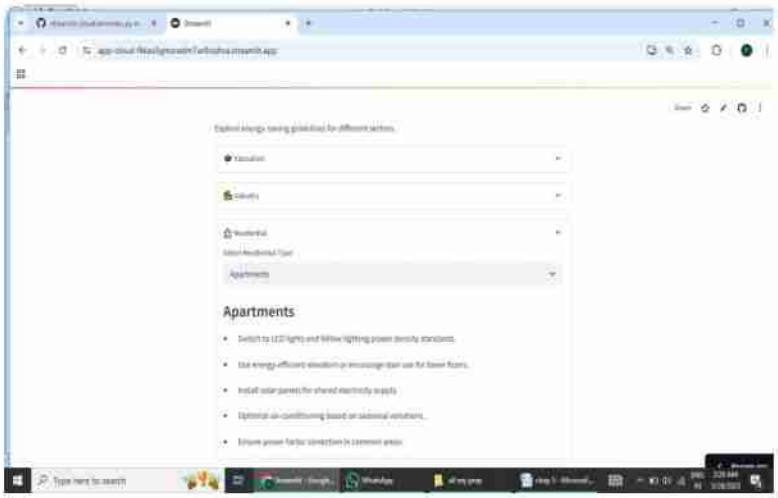


Fig 5.30 Energy savings plan for residential sector

5.5 SOLAR ENERGY INSTALLATION

The "Solar Energy Dashboard" provides tools for solar system sizing, energy prediction, and cost calculation.

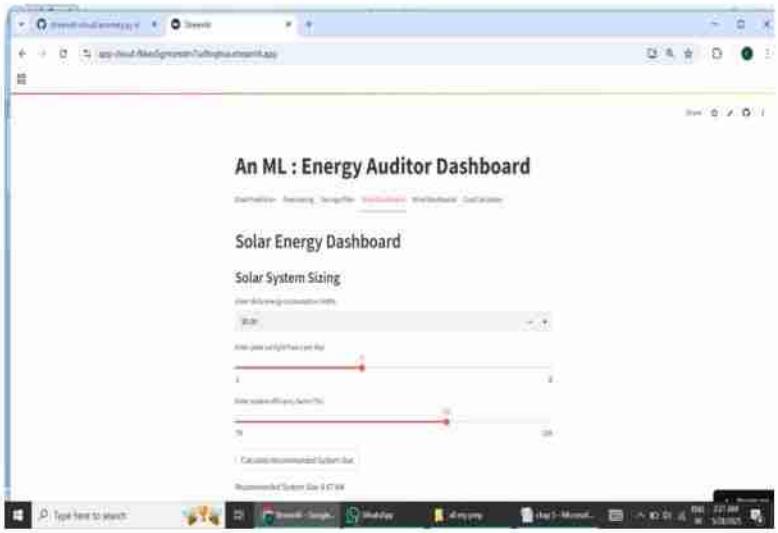


Fig 5.31 Calculation of solar system size

To begin, under "Solar System Sizing," you can input your daily energy consumption in kilowatt-hours (kWh) and the peak sunlight hours per day, along with a system efficiency factor. Upon clicking "Calculate Recommended System Size," the

app will display the estimated system size in kilowatts (kW) which is shown in Fig 5.31.

	A	B
1	DATE	solar irradiance
2	2020/01/01 00:00:00	5.69
3	2020/01/02 00:00:00	5.69
4	2020/01/03 00:00:00	5.33
5	2020/01/04 00:00:00	5.4
6	2020/01/05 00:00:00	5.32
7	2020/01/06 00:00:00	5.72
8	2020/01/07 00:00:00	5.61
9	2020/01/08 00:00:00	5.42
10	2020/01/09 00:00:00	5.3
11	2020/01/10 00:00:00	5.24
12	2020/01/11 00:00:00	5.82
13	2020/01/12 00:00:00	5.98
14	2020/01/13 00:00:00	5.49
15	2020/01/14 00:00:00	5.55

Sheet1

Fig 5.32 Solar Irradiance Data

Upon navigating to the "Solar Energy Prediction" section, users are prompted to upload an Excel file containing essential data such as solar irradiance shown in Fig 5.32, which will then be combined with manually entered details like location and desired system size to complete the input phase.

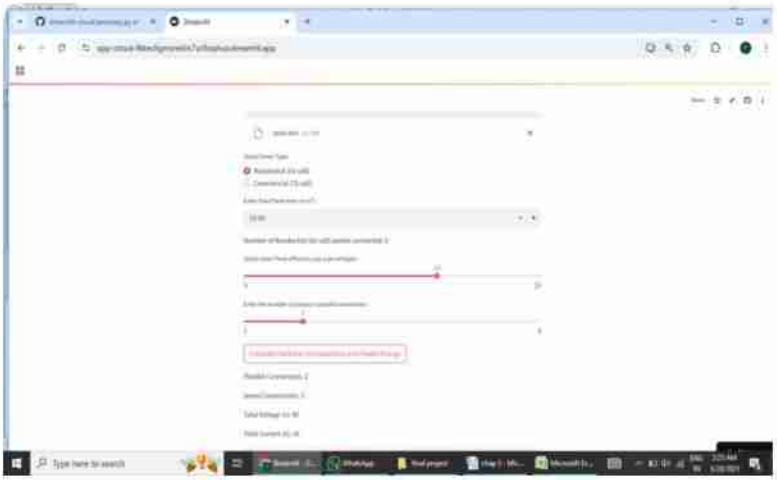


Fig 5.33 Calculation Of Electrical Characteristics

Following this, clicking the "Calculate Electrical Characteristics and Predict Energy" button, as depicted in Fig 5.33, triggers the system to process the inputs and display

crucial design metrics, including the optimal number of series and parallel connections for the solar array, along with its total voltage and current output.

Crucially, the tool will also generate a daily solar irradiance output prediction in kWh for the years 2025 through 2030, presenting this vital forecast both numerically and through a clear, illustrative visual representation to aid in understanding the system's projected long-term performance.

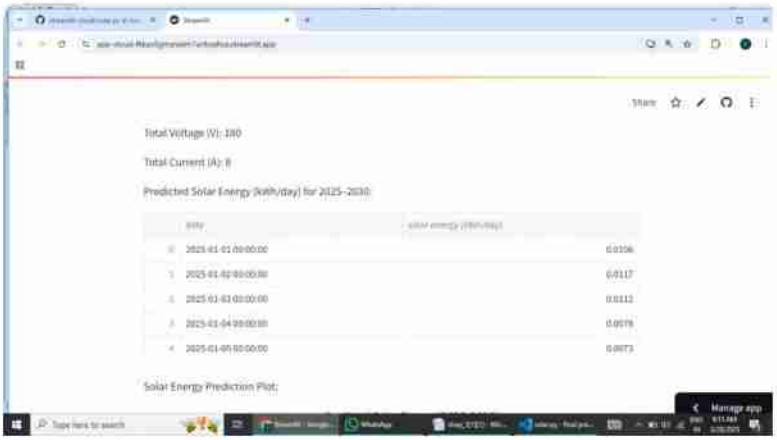


Fig 5.34 Predicted solar irradiance (sample) values

A sample of predicted value of solar irradiance for the year 2025 to 2030 is show in Fig 5.34.

Code for the prediction of solar irradiance:

```
future_dates = pd.date_range(start='2025-01-01', end='2030-12-31', freq='D')
    if not data['solar irradiance'].empty:
        future_data = pd.DataFrame({
            'date': future_dates,
            'solar irradiance': np.random.choice(data['solar irradiance'].fillna(data['solar irradiance'].mean()), size=len(future_dates)) * 100 # Simulated irradiance proxy with mean imputation
        })
```

Finally, the "Solar Installation Cost Calculator" shown in Fig 5.35, estimates the cost of a solar system based on the calculated system size or a manually chosen panel type (Residential or Commercial). It breaks down the costs for panels, inverter, battery (if included), mounting structure, wiring, and installation labor, providing a total estimated cost. You can also input a government subsidy percentage to see the final cost after the subsidy is applied.

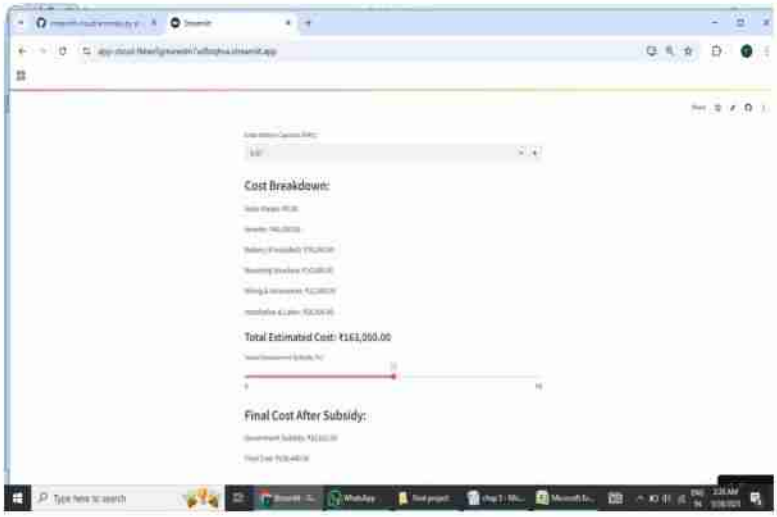


Fig 5.35 Calculation of solar installation cost

5.6 WIND ENERGY INSTALLATION

The "Wind Energy Dashboard" which is shown in Fig 5.36, is a Streamlit application designed for predicting wind energy generation.

Fig 5.36 Wind Energy Dashboard

Users begin by uploading an Excel file containing historical wind speed data, specifically with 'DATE' and 'Max Wind Speed (mps)' columns shown in Fig 5.37.

	A	B
	DATE	Max Wind Speed (mps)
	2020/01/01 00:00:00	0.002222222
	2020/01/02 00:00:00	0.002777778
	2020/01/03 00:00:00	0.003333333
	2020/01/04 00:00:00	0.002777778
	2020/01/05 00:00:00	0.002777778
	2020/01/06 00:00:00	0.004166667
	2020/01/07 00:00:00	0.004166667
	2020/01/08 00:00:00	0.003333333
	2020/01/09 00:00:00	0.0025
	2020/01/10 00:00:00	0.003888889
	2020/01/11 00:00:00	0.003611111
	2020/01/12 00:00:00	0.004444444
	2020/01/13 00:00:00	0.003333333
	2020/01/14 00:00:00	0.003333333

Sheet1 Sheet2 Sheet3

Fig 5.37 Maximum Wind Speed Data

Upon successful upload and verification of the column names, the application prompts for key wind turbine parameters, including the number of turbines, the swept area of each turbine in square meters, and the turbine efficiency as a percentage.

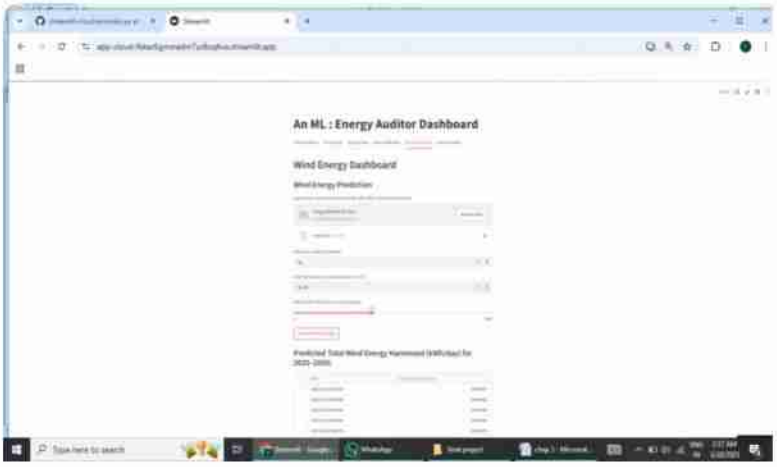


Fig 5.38 Prediction of wind speed

After these inputs, clicking the "Predict Wind Energy" button initiates the prediction process. The application forecasts the total wind energy that can be harnessed in kilowatt-hours per day for the period spanning 2025 to 2030 is shown in Fig 5.38. This prediction is based on a simulation using the uploaded historical wind speed data. The results are then displayed in a Fig 5.39 showing the predicted total energy output for different dates, accompanied by a line plot visualizing the trend of

predicted wind energy over time shown in Fig 5.40. This tool allows users to estimate the potential energy generation from a wind farm based on historical wind data and turbine specifications.

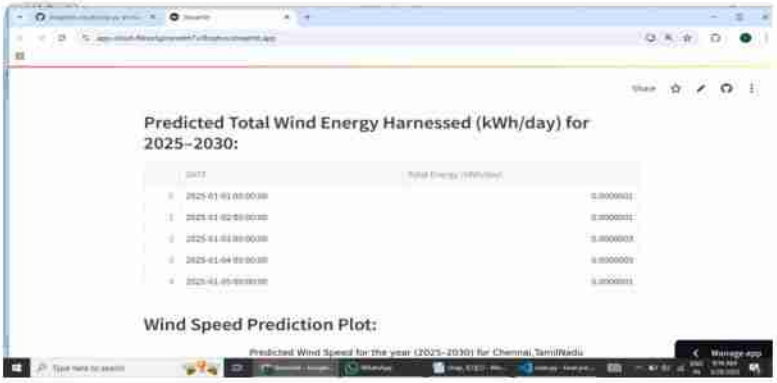


Fig 5.39 Predicted wind speed(sample) data

Code for wind speed prediction:

```
future_dates = pd.date_range(start='2025-01-01', end='2030-12-31', freq='D')
        # Ensure 'Max Wind Speed (mps)' is in the uploaded data before using it
        if not data['Max Wind Speed (mps)'].empty:
            future_data = pd.DataFrame({
                'DATE': future_dates,
                'Max Wind Speed (mps)': np.random.choice(data['Max Wind Speed (mps)'].fillna(data['Max Wind Speed (mps)'].mean()), size=len(future_dates))
            })
```

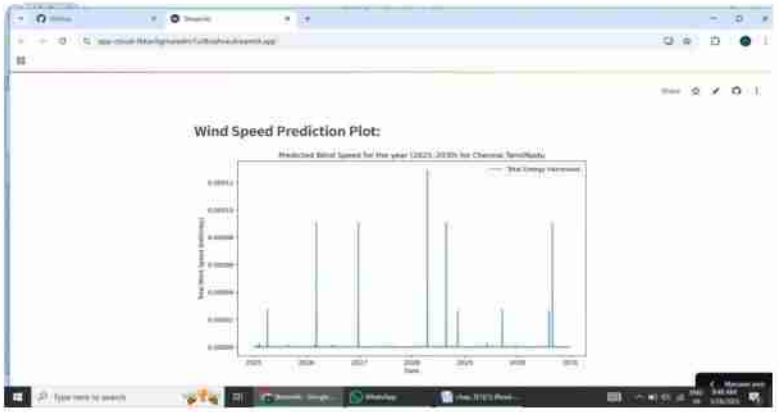


Fig 5.40 Visualization of wind energy prediction

5.7 COST AND ENERGY SAVING CALCULATOR

The "Electricity Cost and Savings Calculator" is shown in Fig 5.41, is a Streamlit application designed to help users estimate their monthly electricity costs and potential savings through the use of solar and wind energy.

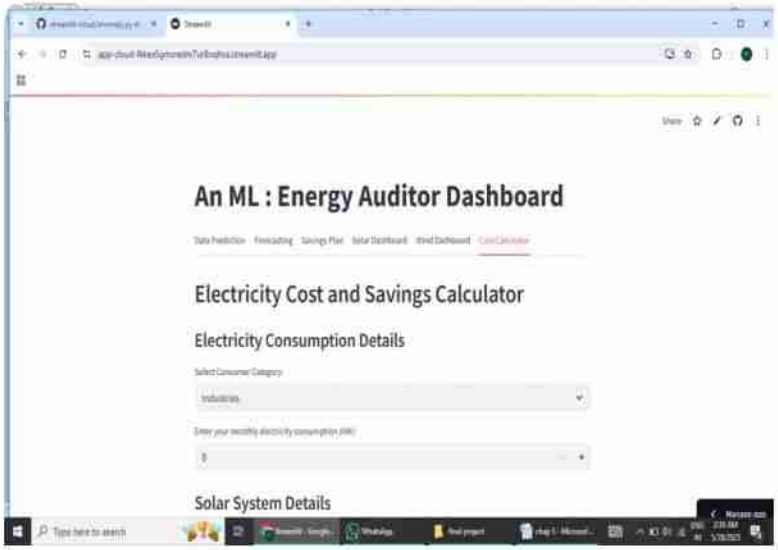


Fig 5.41 Cost Calculator Dashboard

Users begin by selecting their consumer category from a dropdown menu, which includes options like Industries, Government Institutions, various tiers of Domestic consumers, and more, each associated with the Tamil Nadu Electricity Regulatory Commission (TNERC) tariff rates as of July 2024. They then input their total monthly electricity consumption in kilowatts (kW). The application further allows users to specify the size of their solar system in kW, estimating the monthly solar energy generation based on an assumption of 4.5 hours of sunlight per day. Similarly, users can input their wind turbine capacity in kW and the average wind speed in meters per second, with the application providing an estimated monthly wind energy generation using a simplified power calculation.

Fig 5.42 Solar and wind details

Upon clicking the "Calculate Savings" button, the application computes and displays the total monthly consumption, the electricity cost without any renewable energy generation, the cost after accounting for the generated solar and wind energy, and the resulting savings in Indian Rupees.

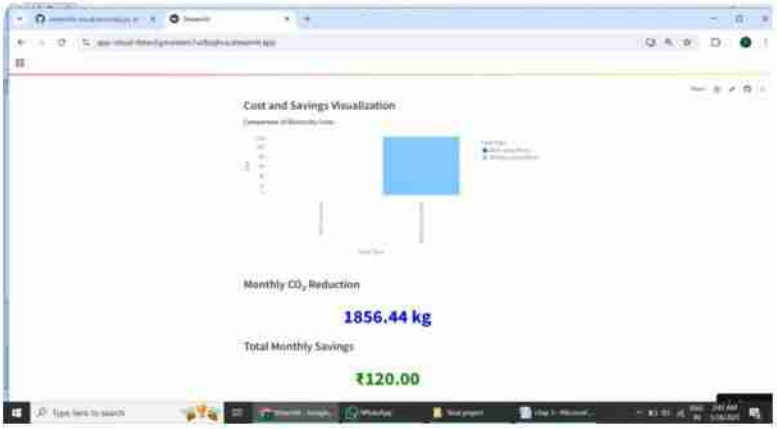


Fig 5.43 Total cost and savings

To provide a visual representation which is shown in Fig 5.43 is the application generated bar chart comparing the electricity costs with and without the integration of solar and wind energy, and prominently displays the total monthly savings. This tool enables users in Tamil Nadu, and potentially other regions with similar tariffs, to understand the economic benefits of incorporating renewable energy sources into their energy consumption.

5.8 CONCLUSION

This architecture offers a cloud-integrated, GitHub-powered energy auditing system that enhances real-time forecasting, accessibility, and automation. By combining machine learning, cloud computing, and interactive dashboards, industries can adopt data-driven energy management for sustainability and efficiency.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

This project presents a cloud-based energy auditing system designed to revolutionize traditional energy management. At its core, the system integrates advanced machine learning models, specifically Support Vector Regressor (SVR), Artificial Neural Networks (ANN), and Random Forest Regression(RFR), to achieve two primary objectives: accurate forecasting of energy consumption and the detection of inefficiencies within energy usage patterns. The research meticulously illustrates how this combination of cutting-edge predictive analytics and cloud deployment significantly elevates the precision, expands the scalability, and improves the overall usability of conventional energy auditing methodologies. This transition moves beyond manual data analysis to a proactive, data-driven approach.

Cloud platform provides Data Prediction for identifying energy dataset anomalies, Forecasting to predict future energy trends, and Savings Plan for developing and tracking energy savings initiatives. Additionally, it includes a Solar Dashboard for analyzing solar energy systems, a Wind Dashboard for insights into wind energy potential, and a Cost Calculator to estimate electricity costs and potential renewable energy savings. This integrated dashboard provides a user-friendly interface for a holistic approach to energy management.

A key innovation within this system is the strategic application of Artificial Neural Networks (ANN) not just for forecasting, but also for imputing missing data, ensuring the completeness and integrity of the datasets for more reliable analysis. The efficacy of the machine learning models is rigorously assessed using multiple error metrics, guaranteeing the robustness and accuracy of their predictions. Furthermore,

the system's practical implementation is facilitated through a Streamlit - based deployment, which provides a user friendly and interactive interface. This comprehensive design results in a highly robust and interactive solution, making sophisticated energy management tools accessible and highly effective, particularly within academic environments where efficient resource utilization is paramount.

6.2 FUTURE WORK

The "Energy and Anomaly Management Dashboard" project lays a robust foundation for future advancements in automated, cloud-based smart energy auditing systems, ultimately contributing to more efficient energy management and enhanced sustainability. To further enhance its capabilities, several key integrations and expansions are envisioned. Integrating IoT (Internet of Things) would enable the utilization of real-time sensor data, allowing for far more granular monitoring of energy consumption and generation at a device or localized level. This high-resolution data would pave the way for Dynamic Recommendations, where the dashboard could provide real-time alerts and personalized energy-saving tips based on current consumption patterns and detected anomalies. To extend accessibility and user convenience, developing Mobile App Support would allow users to monitor their energy usage and receive alerts on the go. Enhancing the financial aspects, the integration of Energy Cost Analysis with dynamic pricing models would enable more accurate forecasting of energy costs based on real-time market fluctuations. Finally, to maximize the impact of the system, Scalability is a crucial consideration, with the potential to expand the dashboard reach to other blocks or even entire campuses, facilitating broader energy optimization across multiple facilities. These future directions build upon the current framework to create a truly intelligent and adaptive energy management solution.

APPENDIX

MISSING DATA PREDICTION

```
import streamlit as st
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor, IsolationForest

@st.cache_data
def load_data(uploaded_file):
    if uploaded_file is not None:
        try:
            data = pd.read_excel(uploaded_file)
            if 'DATE' in data.columns:
                data['DATE'] = pd.to_datetime(data['DATE'])
            else:
                st.error("Error: 'DATE' column not found in the uploaded file.")
                return None
            required_columns = [
                'Temperature (F)', 'Dew Point (F)', 'Max Wind Speed (mps)',
                'Avg Wind Speed (mps)', 'Atm Pressure (hPa)', 'Humidity(g/m^3)',
                'Power_Consumption(MU)'
            ]
            if not all(col in data.columns for col in required_columns[:-1]):
                missing_cols = [col for col in required_columns[:-1] if col not in
        data.columns]
                st.error(f'Error: Missing required columns: {', '.join(missing_cols)} in the
```

```
        uploaded file.")
            return None
        data = data.dropna(subset=required_columns[:-1])
        return data
    except Exception as e:
        st.error(f"An error occurred while loading the data: {e}")
        return None
return None

@st.cache_data
def predict_missing_power(data):
    if data is None:
        return pd.DataFrame()
    if 'Power_Consumption(MU)' not in data.columns:
        st.error("Error: 'Power_Consumption(MU)' column not found for prediction.")
        return data
    known_data = data[data['Power_Consumption(MU)'].notna()]
    missing_data = data[data['Power_Consumption(MU)'].isna()]
    features = ['Temperature (F)', 'Dew Point (F)', 'Max Wind Speed (mps)',
                'Avg Wind Speed (mps)', 'Atm Pressure (hPa)', 'Humidity(g/m^3)']
    if not all(feature in known_data.columns for feature in features):
        missing_features = [f for f in features if f not in known_data.columns]
        st.error(f"Error: Missing features for prediction: {', '.join(missing_features)}")
        return data
    X_known = known_data[features]
    y_known = known_data['Power_Consumption(MU)']
    X_missing = missing_data[features]
    if not X_missing.empty and not X_known.empty:
        model = RandomForestRegressor(n_estimators=200, random_state=42)
```

```
        model.fit(X_known, y_known)
        missing_data['Power_Consumption(MU)'] = model.predict(X_missing)
    elif X_missing.empty:
        st.info("No missing power consumption values to predict.")
    elif X_known.empty:
        st.warning("Not enough data with known power consumption to train the
         prediction model.")
    filled_data = pd.concat([known_data, missing_data]).sort_values(by='DATE') if
         'DATE' in data.columns else pd.concat([known_data, missing_data])
    return filled_data


@st.cache_data
def prepare_heatmap_data(data):
    if data is None or data.empty:
        return pd.DataFrame()
    if 'DATE' not in data.columns:
        st.error("Error: 'DATE' column is required for heatmap generation.")
        return pd.DataFrame()
    data['Month'] = data['DATE'].dt.month
    data['Day'] = data['DATE'].dt.day
    if 'Power_Consumption(MU)' in data.columns:
         data['Power_Consumption(MU)'].fillna(data['Power_Consumption(MU)'].mea
         n(), inplace=True)
        heatmap_data = data.groupby(['Day',
         'Month'])['Power_Consumption(MU)'].mean().reset_index()
        heatmap_data_pivot = heatmap_data.pivot(index='Day', columns='Month',
         values='Power_Consumption(MU)')
        return heatmap_data_pivot
    else:
```

```
        st.error("Error: 'Power_Consumption(MU)' column not found for heatmap
         generation.")
        return pd.DataFrame()

@st.cache_data
def detect_anomalies(data):
    if data is None or data.empty:
        return pd.DataFrame()
    if 'Power_Consumption(MU)' not in data.columns:
        st.error("Error: 'Power_Consumption(MU)' column is required for anomaly
         detection.")
        return pd.DataFrame()
    if len(data) < 10:  # Example threshold, adjust as needed
        st.warning("Insufficient data for robust anomaly detection.")
        data['anomaly'] = 0  # Mark all as non-anomalous
        return data[data['anomaly'] == -1] # Return empty anomaly DataFrame
   iso_forest = IsolationForest(contamination=0.05, random_state=42)
    data['anomaly'] = iso_forest.fit_predict(data[['Power_Consumption(MU)']])
    anomalies = data[data['anomaly'] == -1]
    return anomalies

def show():
    st.title("Power Consumption Analysis with Anomaly Detection & Heatmap")

    uploaded_file = st.file_uploader("Upload your historical data Excel file with Date,
        Temperature (F), Dew Point (F), Max Wind Speed (mps),Avg Wind Speed
        (mps), Atm Pressure (hPa), Humidity(g/m^3), and Power_Consumption(MU)",
        type=["xlsx"])
    data = load_data(uploaded_file)
```

```
if data is not None:
    filled_data = predict_missing_power(data.copy()) # Use a copy to avoid
     modifying the original DataFrame
    heatmap_data_pivot = prepare_heatmap_data(filled_data.copy())
    anomalies = detect_anomalies(filled_data.copy())
    st.subheader("Heatmap of Predicted/Actual Power Consumption")
    if not heatmap_data_pivot.empty:
        fig1, ax1 = plt.subplots(figsize=(10, 6))
        sns.heatmap(heatmap_data_pivot, cmap='YlGnBu', annot=False, fmt=".2f",
     ax=ax1)
        st.pyplot(fig1)
    else:
        st.warning("No data available to generate heatmap.")
    st.subheader("Time Series of Power Consumption with Anomaly Detection")
    fig2, ax2 = plt.subplots(figsize=(12, 5))
    if filled_data is not None and 'DATE' in filled_data.columns and
     'Power_Consumption(MU)' in filled_data.columns:
      ax2.plot(filled_data['DATE'], filled_data['Power_Consumption(MU)'],
              label='Power Consumption', color='blue', linewidth=1.5)
      if not anomalies.empty and 'DATE' in anomalies.columns and
     'Power_Consumption(MU)' in anomalies.columns:
          ax2.scatter(anomalies['DATE'], anomalies['Power_Consumption(MU)'],
                    color='red', label='Anomaly', s=60, marker='o')
      ax2.set_xlabel('Date')
      ax2.set_ylabel('Power Consumption (MU)')
      ax2.legend()
      ax2.grid(True)
      st.pyplot(fig2)
    else:
```

```
            st.warning("Not enough data to plot the time series with anomaly detection.")
        st.subheader(" View Predicted/Actual Power Consumption Data")
        if filled_data is not None:
            st.dataframe(filled_data.head(10))
            st.download_button(
                label="Download Processed Data",
                data=filled_data.to_csv(index=False),
                file_name="Processed_Power_Consumption.csv",
                mime="text/csv"
            )
    else:
        st.info("Please upload an Excel file to start the analysis.")

if __name__ == "main":
    show()
```

COMPARISION OF ML MODELS FOR POWER CONSUMPTION FORECASTING

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import streamlit as st
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from datetime import timedelta
from io import BytesIO

def show():
    st.header("Power Consumption Forecasting Comparison")
    uploaded_file = st.file_uploader("Upload Excel File for Forecasting",
        type=['xlsx'])
    if uploaded_file is not None:
        try:
            df = pd.read_excel(uploaded_file)
        except Exception as e:
            st.error(f"Error reading file: {e}")
            st.stop()
        if 'DATE' not in df.columns:
            st.error("Error: The 'DATE' column is missing in the uploaded file.")
```

```
        st.stop()
    df['DATE'] = pd.to_datetime(df['DATE'])  # convert 'DATE' column to
     datetime
    if 'Power_Consumption(MU)' not in df.columns:
        st.error("Error: The 'Power_Consumption(MU)' column is missing in the
     uploaded file.")
        st.stop()
    model_choice = st.sidebar.selectbox("Choose Model for Individual Forecast",
     ['ANN', 'RF', 'SVM'])
    if st.sidebar.button(f"Run Forecast with {model_choice}"):
        try:
            forecast_df, mse, r2, mae, predicted_df = forecast_model(df.copy(),
     model_choice)
            st.subheader(f"Forecasting Results - {model_choice}")
            st.markdown(f"MSE: {mse:.2f}, R²: {r2:.4f}, MAE: {mae:.2f}")
st.line_chart(data=forecast_df.set_index('Date')['Power_Consumption(MU)'])
            with st.expander(" Predicted vs Actual Data"):
                st.dataframe(predicted_df.head(50))
            with st.expander(" Forecasted Future Data"):
                st.dataframe(forecast_df.head(50))
            st.subheader(" Error Distribution")
            predicted_df['Error'] = predicted_df['Actual'] - predicted_df['Predicted']
            fig_error, ax_error = plt.subplots()
            sns.histplot(predicted_df['Error'], bins=30, kde=True, ax=ax_error)
            ax_error.set_title(f"Error Distribution for {model_choice}")
            st.pyplot(fig_error)
            st.subheader("□ Daily Power Consumption Changes")
            df_sorted = df.sort_values('DATE').copy()
            df_sorted['Daily Change'] = df_sorted['Power_Consumption(MU)'].diff()
```

```
        fig_daily, ax_daily = plt.subplots()
        ax_daily.plot(df_sorted['DATE'], df_sorted['Daily Change'])
        ax_daily.set_title("Daily Change in Power Consumption")
        ax_daily.set_xlabel("Date")
        ax_daily.set_ylabel("Change (MU)")
        st.pyplot(fig_daily)
    except ValueError as ve:
        st.error(f"Error in forecasting with {model_choice}: {ve}")
    except Exception as e:
        st.error(f"An unexpected error occurred during forecasting with
 {model_choice}: {e}")
st.subheader("□ Joint Model Comparison (5-Year Forecast)")
try:
    ann_df, ann_mse, ann_r2, ann_mae, _ = forecast_model(df.copy(), 'ANN')
    rf_df, rf_mse, rf_r2, rf_mae, _ = forecast_model(df.copy(), 'RF')
    svm_df, svm_mse, svm_r2, svm_mae, _ = forecast_model(df.copy(), 'SVM')
    joint_df = pd.concat([ann_df.assign(Model='ANN'),
 rf_df.assign(Model='RF'), svm_df.assign(Model='SVM')])
    fig_joint, ax_joint = plt.subplots(figsize=(14, 7))
    for label, data in joint_df.groupby('Model'):
        ax_joint.plot(data['Date'], data['Power_Consumption(MU)'], label=label)
    ax_joint.legend()
    ax_joint.set_title("Comparison of Forecasts Over 5 Years")
    ax_joint.set_xlabel("Date")
    ax_joint.set_ylabel("Power Consumption (MU)")
    ax_joint.grid(True)
    st.pyplot(fig_joint)
    st.subheader(" Model Comparison Metrics")
    metrics_df = pd.DataFrame({
```

```
            'Model': ['ANN', 'RF', 'SVM'],
            'MSE': [ann_mse, rf_mse, svm_mse],
            'R²': [ann_r2, rf_r2, svm_r2],
            'MAE': [ann_mae, rf_mae, svm_mae]
        })
        st.dataframe(metrics_df)
        st.subheader("□ Download Forecast Data")
        for model, forecast_data in zip(['ANN', 'RF', 'SVM'], [ann_df, rf_df,
    svm_df]):
            excel_buffer_forecast = BytesIO()
            forecast_data.to_excel(excel_buffer_forecast, index=False)
            excel_buffer_forecast.seek(0)
            st.download_button(
                label=f"Download {model} 5-Year Forecast",
                data=excel_buffer_forecast,
                file_name=f"{model}_5yr_forecast.xlsx",
                mime="application/vnd.openxmlformats-
    officedocument.spreadsheetml.sheet"
            )
    except Exception as e:
        st.error(f"An error occurred during joint model comparison: {e}")

else:
    st.warning("Please upload an Excel file to proceed with forecasting.")
def load_data(file_path):
    df = pd.read_excel(file_path)
    df['DATE'] = pd.to_datetime(df['DATE'])
    return df
    df['Year'] = df['DATE'].dt.year
```

```
df['DayOfYear'] = df['DATE'].dt.dayofyear
X = df[['Year', 'DayOfYear']].values
y = df['Power_Consumption(MU)'].values.reshape(-1, 1)
x_scaler = MinMaxScaler()
y_scaler = MinMaxScaler()

X_scaled = x_scaler.fit_transform(X)
y_scaled = y_scaler.fit_transform(y)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled,
    test_size=0.2, random_state=42)
if model_name == 'ANN':
   model = MLPRegressor(hidden_layer_sizes=(100, 50), max_iter=1000,
    random_state=42)
elif model_name == 'RF':
   model = RandomForestRegressor(n_estimators=100, random_state=42)
elif model_name == 'SVM':
   model = SVR(C=100, gamma=0.1, epsilon=0.1)
else:
   raise ValueError("Invalid model name")
model.fit(X_train, y_train.ravel())
y_pred_scaled = model.predict(X_test)
y_pred = y_scaler.inverse_transform(y_pred_scaled.reshape(-1, 1))
y_test_inv = y_scaler.inverse_transform(y_test)
mse = mean_squared_error(y_test_inv, y_pred)
r2 = r2_score(y_test_inv, y_pred)
mae = mean_absolute_error(y_test_inv, y_pred)
last_date = df['DATE'].max()
future_dates = [last_date + timedelta(days=i) for i in range(1, 5 * 365 + 1)]
future_df = pd.DataFrame({
```

```
        'Date': future_dates,
        'Year': [d.year for d in future_dates],
        'DayOfYear': [d.timetuple().tm_yday for d in future_dates]
    })
    future_scaled = x_scaler.transform(future_df[['Year', 'DayOfYear']].values)
    future_predictions_scaled = model.predict(future_scaled)
    future_predictions =
        y_scaler.inverse_transform(future_predictions_scaled.reshape(-1, 1))
    forecast_df = pd.DataFrame({
        'Date': future_dates,
        'Power_Consumption(MU)': future_predictions.flatten(),
        'Model': model_name
    })
    predicted_df = pd.DataFrame({
        'Date': pd.to_datetime(dict(
            year=(X_test[:, 0] * (x_scaler.data_max_[0] - x_scaler.data_min_[0]) +
        x_scaler.data_min_[0]).astype(int),
            month=1, day=1)) +
                pd.to_timedelta(((X_test[:, 1] * (x_scaler.data_max_[1] -
        x_scaler.data_min_[1]) + x_scaler.data_min_[
                    1]) - 1).astype(int), unit='D'),
        'Actual': y_test_inv.flatten(),
        'Predicted': y_pred.flatten(),
        'Model': model_name
    })
    return forecast_df, mse, r2, mae, predicted_df

if __name__ == '__main__':
    show()
```

ENERGY SAVINGS PLAN

```
import streamlit as st
def show():
    st.header("Energy Saving Plan Dashboard")
    st.write("Explore energy-saving guidelines for different sectors.")
    energy_plans = {
"Education": [
"Turn off lights and fans after class",
"Use LED lighting in classrooms and halls",
"Maximize use of natural daylight",
"Maintain electrical equipment regularly",
"Install solar panels if possible",
"Encourage energy-saving habits",
"Use motion-sensor lights in restrooms and corridors",
"Switch off computers and projectors after use",
"Use fans and ventilation instead of ACs when possible",
"Create an energy monitoring team",
        ],
"Industry": {
"Cement": [
"Maintain air ratio for kilns (Recommended: 1.15–1.18)",
"Optimize boiler efficiency (Target temperature: 130°C at final stage)",
"Use waste heat recovery systems for preheating raw materials",
"Enhance fuel combustion efficiency with automatic air-fuel control",
            ],
"Iron & Steel": [
"Adopt waste heat recovery for flue gases (Min 52% efficiency)",
"Optimize power factor correction (Target > 0.95)",
"Upgrade furnace insulation (Maintain ceiling temperature: 120°C)",
```

"Use oxygen-enriched combustion systems to improve fuel utilization",

],

"Textile": [

"Reduce lighting power density (Recommended: 7.1–14.1 W/m²)",

"Install high-efficiency IE3 motors (Min efficiency: 93%)",

"Implement optimized compressed air systems (Leakage < 10%)",

"Use solar-powered heating systems for drying and processing",

]

},

"Residential": {

"Apartments": [

"Switch to LED lights and follow lighting power density standards.",

"Use energy-efficient elevators or encourage stair use for lower floors.",

"Install solar panels for shared electricity supply.",

"Optimize air-conditioning based on seasonal variations.",

"Ensure power factor correction in common areas.",

],

"Houses": [

"Use solar water heaters instead of electric geysers.",

"Regulate air-conditioning between 24–26°C and ensure filter maintenance.",

"Improve thermal insulation to reduce heat loss.",

"Use energy-efficient fans, refrigerators, and washing machines.",

"Seal windows and doors to enhance energy efficiency.",

],

"Neighborhoods": [

"Install solar-powered street lights.",

"Encourage carpooling and cycling for reduced emissions.",

"Implement monthly energy audits and awareness workshops.",

"Promote use of efficient power distribution transformers.",

```
"Develop community-level waste heat recovery programs.",
            ]
        }
    }
    with st.expander("□ Education"):
        for plan in energy_plans["Education"]:
            st.markdown(f"- {plan}")
    with st.expander(" Industry"):
        industry_types = list(energy_plans["Industry"].keys())
        selected_industry = st.selectbox("Select Industry Type:", industry_types,
         key="industry_selectbox")
        if selected_industry:
            st.subheader(selected_industry)
            for plan in energy_plans["Industry"][selected_industry]:
                st.markdown(f"- {plan}")
    with st.expander("□ Residential"):
        residential_types = list(energy_plans["Residential"].keys())
        selected_residential = st.selectbox("Select Residential Type:", residential_types,
         key="residential_selectbox")
        if selected_residential:
            st.subheader(selected_residential)
            for plan in energy_plans["Residential"][selected_residential]:
                st.markdown(f"- {plan}")
    st.info("This interactive dashboard now integrates energy benchmarks from
         official guidelines.")

if __name__ == "__main__":
    show()
```

SOLAR ENERGY UTILIZATION

```
import streamlit as st
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

def show():
    st.header("Solar Energy Dashboard")
    PANEL_COST_PER_WATT = {"Residential": 50, "Commercial": 45}  # ₹ per
        watt
    INVERTER_COST = 40_000  # ₹ (fixed cost for inverter)
    BATTERY_COST_PER_KWH = 15_000  # ₹ per kWh (optional)
    STRUCTURE_COST = 10_000  # ₹ (fixed cost for mounting structure)
    WIRING_ACCESSORIES_COST = 12_000  # ₹ (fixed cost for wiring and
        accessories)
    INSTALLATION_LABOR_COST = 25_000  # ₹ (fixed cost for labor)
    RESIDENTIAL_PANEL_AREA = 1.6  # m² (for 60-cell panels)
    COMMERCIAL_PANEL_AREA = 2.0  # m² (for 72-cell panels)
    VOLTAGE_PER_PANEL = 30  # Typical voltage for a panel in series
        connection
    CURRENT_PER_PANEL = 8  # Typical current for a panel in parallel
        connection
    st.subheader("Solar System Sizing")
    energy_consumption = st.number_input("Enter daily energy consumption
        (kWh):", min_value=1.0, value=30.0)
    sunlight_hours = st.slider("Enter peak sunlight hours per day:", min_value=3,
        max_value=8, value=5)
    efficiency_factor = st.slider("Enter system efficiency factor (%):", min_value=70,
```

```
    max_value=100, value=90) / 100
calculated_system_size = 0.0
calculate_button_pressed = st.button("Calculate Recommended System Size")
if calculate_button_pressed:
    if energy_consumption > 0 and sunlight_hours > 0 and efficiency_factor > 0:
        calculated_system_size = energy_consumption / (sunlight_hours *
    efficiency_factor)
        st.write(f"Recommended System Size: {calculated_system_size:.2f} kW")
st.subheader("Solar Energy Prediction")
uploaded_file = st.file_uploader("Upload your solar irradiance data Excel file
    with DATE, solar irradiance", type=["xlsx"])
if uploaded_file:
    try:
        data = pd.read_excel(uploaded_file, sheet_name='Sheet1')
        data.columns = data.columns.str.strip().str.lower()  # Normalize column
    names to lowercase
        data['date'] = pd.to_datetime(data['date'], errors='coerce')  # Convert Excel
    date to datetime
        if 'solar irradiance' not in data.columns:
            st.error("Error: 'solar irradiance' column not found in the uploaded file.")
        else:
            panel_type = st.radio("Select Panel Type:", options=["Residential (60-
    cell)", "Commercial (72-cell)"])
            total_area = st.number_input("Enter Total Panel Area (in m²):",
    min_value=1.0, value=10.0)
            if panel_type == "Residential (60-cell)":
                panel_area_per_unit = RESIDENTIAL_PANEL_AREA
            else:
                panel_area_per_unit = COMMERCIAL_PANEL_AREA
```

```
        num_panels = int(total_area / panel_area_per_unit)
        st.write(f"Number of {panel_type} panels connected: {num_panels}")
        panel_efficiency = st.slider("Select Solar Panel Efficiency (as a
percentage):", min_value=5, max_value=25, value=18) / 100
        num_parallel = st.slider("Enter the number of panels in parallel
connection:", min_value=1, max_value=num_panels, value=1)
        calculate_electrical_button = st.button("Calculate Electrical
Characteristics and Predict Energy")
        if calculate_electrical_button:
            num_series = num_panels // num_parallel
            st.write(f"Parallel Connections: {num_parallel}")
            st.write(f"Series Connections: {num_series}")
            total_voltage = VOLTAGE_PER_PANEL * num_series
            total_current = CURRENT_PER_PANEL * num_parallel
            st.write(f"Total Voltage (V): {total_voltage}")
            st.write(f"Total Current (A): {total_current}")
            future_dates = pd.date_range(start='2025-01-01', end='2030-12-31',
freq='D')
            if not data['solar irradiance'].empty:
                future_data = pd.DataFrame({
                    'date': future_dates,
                    'solar irradiance': np.random.choice(data['solar
irradiance'].fillna(data['solar irradiance'].mean()), size=len(future_dates)) *
100
                })
                future_data['solar energy (kWh/day)'] = (
                    future_data['solar irradiance'] * panel_efficiency * total_area /
1000
                )
```

```
                st.write("Predicted Solar Energy (kWh/day) for 2025–2030:")
                st.dataframe(future_data[['date', 'solar energy (kWh/day)']].head())
                st.write("Solar Energy Prediction Plot:")
                fig, ax = plt.subplots(figsize=(10, 6))
                ax.plot(future_data['date'], future_data['solar energy (kWh/day)'],
    label="Predicted Energy")
                ax.set_xlabel("Date")
                ax.set_ylabel("Solar Energy (kWh/day)")
                ax.set_title("Predicted Solar Energy (2025–2030)")
                ax.legend()
                st.pyplot(fig)
            else:
                st.warning("Warning: 'solar irradiance' data is empty in the uploaded
    file. Cannot generate prediction.")
    except Exception as e:
        st.error(f"An error occurred while processing the uploaded file: {e}")
st.markdown("---")
st.subheader("Solar Installation Cost Calculator")
panel_type_calc = st.radio("Select Panel Type for Cost Calculation:",
    options=["Residential", "Commercial"])
include_battery = st.checkbox("Include Battery Storage?")
battery_capacity_kwh = 0.0
if include_battery:
    battery_capacity_kwh = st.number_input("Enter Battery Capacity (kWh):",
    min_value=1.0, value=5.0)
panel_cost = PANEL_COST_PER_WATT[panel_type_calc] *
    calculated_system_size * 1000 if calculated_system_size > 0 else 0
inverter_cost = INVERTER_COST
battery_cost = battery_capacity_kwh * BATTERY_COST_PER_KWH
```

```
    total_cost = (
        panel_cost
        + inverter_cost
        + battery_cost
        + STRUCTURE_COST
        + WIRING_ACCESSORIES_COST
        + INSTALLATION_LABOR_COST
    )
    st.write("### Cost Breakdown:")
    st.write(f"Solar Panels: ₹{panel_cost:,.2f}")
    st.write(f"Inverter: ₹{inverter_cost:,.2f}")
    st.write(f"Battery (if included): ₹{battery_cost:,.2f}")
    st.write(f"Mounting Structure: ₹{STRUCTURE_COST:,.2f}")
    st.write(f"Wiring & Accessories: ₹{WIRING_ACCESSORIES_COST:,.2f}")
    st.write(f"Installation & Labor: ₹{INSTALLATION_LABOR_COST:,.2f}")
    st.write(f"#### Total Estimated Cost: ₹{total_cost:,.2f}")
    subsidy_percent = st.slider("Select Government Subsidy (%):", min_value=0,
        max_value=40, value=20)
    subsidy_amount = (subsidy_percent / 100) * total_cost
    final_cost = total_cost - subsidy_amount
    st.write("### Final Cost After Subsidy:")
    st.write(f"Government Subsidy: ₹{subsidy_amount:,.2f}")
    st.write(f"Final Cost: ₹{final_cost:,.2f}")

if __name__ == "__main__":
    show()
```

WIND ENERGY UTILIZATION

```
import streamlit as st
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

def show():
    st.header("Wind Energy Dashboard")
    st.subheader("Wind Energy Prediction")
    uploaded_file = st.file_uploader("Upload your wind speed data Excel file with
        DATE, max wind speed(mps)", type=["xlsx"])
    if uploaded_file:
        try:
            data = pd.read_excel(uploaded_file, sheet_name='Sheet1')
            if 'DATE' not in data.columns or 'Max Wind Speed (mps)' not in
        data.columns:
                st.error("Error: The uploaded file must contain 'DATE' and 'Max Wind
        Speed (mps)' columns.")
                return
            data['DATE'] = pd.to_datetime(data['DATE'])
            num_turbines = st.number_input("Enter the number of turbines:",
        min_value=1, value=10)
            swept_area = st.number_input("Enter the swept area of each turbine (in
        m²):", min_value=1.0, value=50.0)
            efficiency = st.slider("Enter turbine efficiency (as a percentage):",
        min_value=1, max_value=100, value=40) / 100
            air_density = 1.225  # kg/m³ (constant air density)
```

```
  predict_button = st.button("Predict Wind Energy")
  if predict_button:
      future_dates = pd.date_range(start='2025-01-01', end='2030-12-31',
freq='D')
      if not data['Max Wind Speed (mps)'].empty:
          future_data = pd.DataFrame({
              'DATE': future_dates,
              'Max Wind Speed (mps)': np.random.choice(data['Max Wind Speed
(mps)'].fillna(data['Max Wind Speed (mps)'].mean()), size=len(future_dates))
          })
          future_data['Wind Power (W)'] = 0.5 * air_density * swept_area *
(future_data['Max Wind Speed (mps)'] ** 3) * efficiency
          future_data['Energy per Turbine (kWh/day)'] = future_data['Wind
Power (W)'] * 24 / 1000  # Convert to kWh/day
          future_data['Total Energy (kWh/day)'] = future_data['Energy per
Turbine (kWh/day)'] * num_turbines
          st.subheader("Predicted Total Wind Energy Harnessed (kWh/day) for
2025–2030:")
          st.dataframe(future_data[['DATE', 'Total Energy (kWh/day)']].head())
          st.subheader("Wind Energy Prediction Plot:")
          fig, ax = plt.subplots(figsize=(10, 6))
          ax.plot(future_data['DATE'], future_data['Total Energy (kWh/day)'],
label="Total Energy Harnessed")
          ax.set_xlabel("Date")
          ax.set_ylabel("Total Energy (kWh/day)")
          ax.set_title("Predicted Wind Energy (2025–2030)")
          ax.legend()
          st.pyplot(fig)
      else:
```

```
                st.warning("Warning: 'Max Wind Speed (mps)' data is empty in the
        uploaded file. Cannot generate prediction.")
        except FileNotFoundError:
            st.error("Error: The uploaded file was not found.")
        except KeyError as e:
            st.error(f"Error: Column '{e}' not found in the uploaded file. Please ensure
        the file has 'DATE' and 'Max Wind Speed (mps)' columns.")
        except Exception as e:
            st.error(f"An error occurred while processing the uploaded file: {e}")

if __name__ == "__main__":
    show()
```

ENERGY SAVINGS COST CALCULATOR

```
import pandas as pd
import streamlit as st
import matplotlib.pyplot as plt
import altair as alt

electricity_cost_per_kwh = {
"Industries": 7.25,
"Govt. Institutions": 7.50,
"Private Institutions": 8.00,
"Miscellaneous": 9.10,
"Construction": 12.85,
"Domestic (<500 kW)": {"0-100 kW": 4.80, "101-200 kW": 6.00, "201-400 kW":
      7.15, "401-500 kW": 8.60},
"Domestic (>500 kW)": {"501-600 kW": 8.40, "601-800 kW": 9.45, "801-1000
      kW": 10.50, "Above 1000 kW": 11.55},
"LT Public Lighting": 8.55,
"LT Industries": 8.00,
"LT Agriculture": 4.80,
"LT Miscellaneous": 10.15,
"LT Construction": 12.85,
}

def calculate_domestic_cost(units_used):
   if units_used <= 100:
      return electricity_cost_per_kwh["Domestic (<500 kW)"]["0-100 kW"] *
       units_used
   elif 101 <= units_used <= 200:
      return electricity_cost_per_kwh["Domestic (<500 kW)"]["101-200 kW"] *
```

```
        units_used
    elif 201 <= units_used <= 400:
        return electricity_cost_per_kwh["Domestic (<500 kW)"]["201-400 kW"] *
        units_used
    elif 401 <= units_used <= 500:
        return electricity_cost_per_kwh["Domestic (<500 kW)"]["401-500 kW"] *
        units_used
    elif 501 <= units_used <= 600:
        return electricity_cost_per_kwh["Domestic (>500 kW)"]["501-600 kW"] *
        units_used
    elif 601 <= units_used <= 800:
        return electricity_cost_per_kwh["Domestic (>500 kW)"]["601-800 kW"] *
        units_used
    elif 801 <= units_used <= 1000:
        return electricity_cost_per_kwh["Domestic (>500 kW)"]["801-1000 kW"] *
        units_used
    else:
        return electricity_cost_per_kwh["Domestic (>500 kW)"]["Above 1000 kW"] *
        units_used
def calculate_cost(category, units_used):
    if category == "Domestic (<500 kW)" or category == "Domestic (>500 kW)":
        return calculate_domestic_cost(units_used)
    else:
        return electricity_cost_per_kwh[category] * units_used
def calculate_savings(units_used, solar_generation, wind_generation, category):
    cost_without_solar_wind = calculate_cost(category, units_used)
    net_consumption = max(0, units_used - solar_generation - wind_generation)  #
        Ensure non-negative
    cost_with_solar_wind = calculate_cost(category, net_consumption)
```

```
    return cost_without_solar_wind - cost_with_solar_wind

def show():
    st.header("Electricity Cost and Savings Calculator")
    st.subheader("Electricity Consumption Details")
    category = st.selectbox("Select Consumer Category:",
        list(electricity_cost_per_kwh.keys()))
    units_used = st.number_input("Enter your monthly electricity consumption
        (kW):", min_value=0)
    st.subheader("Solar System Details")
    solar_size_kw = st.number_input("Enter your solar system size (kW):",
        min_value=0.0)
    solar_generation_per_day = solar_size_kw * 4.5  # Assuming 4.5 hours of
        sunlight per day
    solar_generation_per_month = solar_generation_per_day * 30
    st.write(f"Estimated monthly solar generation: {solar_generation_per_month:.2f}
        kW")
    st.subheader("Wind Energy Details")
    wind_turbine_kw = st.number_input("Enter your wind turbine capacity (kW):",
        min_value=0.0)
    average_wind_speed = st.slider("Enter average wind speed (m/s):", min_value=0,
        max_value=25, value=10)
    wind_generation_per_month = 0.5 * 1.225 * (average_wind_speed ** 3) *
        (wind_turbine_kw/1000) * 24 * 30
    st.write(f"Estimated monthly wind generation: {wind_generation_per_month:.2f}
        kW")
    if st.button("Calculate Savings"):
       savings = calculate_savings(units_used, solar_generation_per_month,
        wind_generation_per_month, category)
```

```
    st.subheader("Results")
    st.write(f"Total Monthly Consumption: {units_used} kW")
    cost_without_solar_wind = calculate_cost(category, units_used)
    st.write(f"Cost without Solar and Wind: ₹{cost_without_solar_wind:.2f}")
    cost_with_solar_wind = calculate_cost(category, max(0, units_used -
     solar_generation_per_month - wind_generation_per_month))
    st.write(f"Cost with Solar and Wind: ₹{cost_with_solar_wind:.2f}")
    st.write(f"Savings: ₹{savings:.2f}")
    st.subheader("Cost and Savings Visualization")
    plot_data = pd.DataFrame({
        'Cost Type': ['Without Solar/Wind', 'With Solar/Wind'],
        'Cost': [cost_without_solar_wind, cost_with_solar_wind]
    })
    chart = alt.Chart(plot_data).mark_bar().encode(
        x='Cost Type',
        y='Cost',
        color='Cost Type',
        tooltip=['Cost Type', 'Cost']  # Add tooltip for interactivity
    ).properties(
        title='Comparison of Electricity Costs'
    )
    st.altair_chart(chart, use_container_width=True)  # Make chart responsive
    st.subheader("Total Monthly Savings")
    st.markdown(f"<h1 style='text-align: center; color:
     green;'>₹{savings:.2f}</h1>", unsafe_allow_html=True)

if __name__ == "__main__":
    show()
```

CLOUD DEPLOYMENT

```
import streamlit as st
from anomaly import show as show_anomaly
from compare import show as show_compare
from savingsplan import show as show_savingsplan
from solar import show as show_solar
from wind import show as show_wind
from costcalculator import show as show_costcalculator

def main():
    st.title("Energy and Anomaly Management Dashboard")
    tab1, tab2, tab3, tab4, tab5, tab6 = st.tabs(["Data Prediction",
"Forecasting",
"Savings Plan",
"Solar Dashboard",
"Wind Dashboard",
"Cost Calculator"])
    with tab1:
        show_anomaly()
    with tab2:
        show_compare()
    with tab3:
        show_savingsplan()
    with tab4:
        show_solar()
    with tab5:
        show_wind()

    with tab6:
```

```
        show_costcalculator()

if __name__ == "__main__":
    main()
```

REFERENCES

1) A. González-Briones, G. Hernández, J. M. Corchado, S. Omatu and M. S. Mohamad, "Machine Learning Models for Electricity Consumption Forecasting: A Review," 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS), Riyadh, Saudi Arabia, 2019,pp.1-6, doi: 10.1109/CAIS.2019.8769508.

2) A., Nagesh. (2021). Energy Audit System for Households using Machine Learning. International Journal of Innovative Technology and Exploring Engineering.10. 33-36. 10.35940/ijitee.G8895.0510721.

3) Ahmad, Tanveer & Chen, Huanxin. (2019). Nonlinear autoregressive and random forest approaches to forecasting electricity load for utility energy management systems. Sustainable Cities and Society. 45. 10.1016/j.scs.2018.12.013.

4) Barbour, Edward & González, Marta. (2018). Enhancing household-level load forecasts using daily load profile clustering.107-115. 10.1145/3276774.3276793.

5) Chang, Hong-Chan & Kuo, Cheng-Chien & Chen, Yu-Tung & Wu, Wei-Bin & Piedad, Eduardo Jr. (2018). Energy Consumption Level Prediction Based on Classification Approach with Machine Learning Technique. 10.11159/icert18.108.

6) Edwards, Richard & New, Joshua & Parker, Lynne. (2012). Predicting Future Hourly Residential Electrical Consumption: A Machine Learning Case

Study. Energy and Buildings - ENERG BLDG. 49. 591- 603. 10.1016/j.enbuild.2012.03.010.

7) E. Roponena, J. Kampars, A. Gailitis and J. Strods, "A Literature Review of Machine Learning Techniques for Cybersecurity in Data Centers," 2021 62nd International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS), Riga, Latvia, 2021, pp. 1-6, doi:10.1109/ITMS52826.2021.9615321.

8) Gajowniczek, K., & Ząbkowski, T. (2014). Short term electricity forecasting using individual smart meter data. Procedia Computer Science, 35, 589–597. https://doi.org/10.1016/j.procs.2014.08.140

9) H. Sequeira, P. Carreira, T. Goldschmidt and P. Vorst, "Energy Cloud: Real-Time Cloud-Native Energy Management System to Monitor and Analyze Energy Consumption in Multiple Industrial Sites," 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, London, UK, 2014, pp. 529-534, doi: 10.1109/UCC.2014.79.

10) K. Shahryari and A. Anvari-Moghaddam, "Demand Side Management Using the Internet of Energy Based on Fog and Cloud Computing," 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Exeter, UK,2017, pp.931-936,doi:10.1109/iThings-GreenCom-CPSCom SmartData.2017.143.

11) Luján, Emmanuel & Otero, Alejandro & Valenzuela Martínez, Sebastián & Steffenel, Luiz Angelo & Nesmachnow, Sergio. (2019). An integrated

platform for smart energy management: The CC-SEM project. Revista Facultad de Ingeniería. 10.17533/udea.redin.20191147.

12) M. Hans, P. Phad, V. Jogi and P. Udayakumar, "Energy Management of Smart Grid using Cloud Computing,"2018 International Conference on Information , Communication, Engineering and Technology (ICICET), Pune, India, 2018,pp.1-4,doi:10.1109/ICICET.2018.8533692.

13) Moon, Jihoon & Park, Jinwoong & Hwang, Eenjun & Jun, Sanghoon. (2018). Forecasting power consumption for higher educational institutions based on machine learning. The Journal of Supercomputing.74. 10.1007/s11227-017-2022-x.

14) Naveen, P., Ing, W. K., Danquah, M. K.,Sidhu, A. S., & Abu-Siada, A. (2016). Cloud computing for energy management in smart grid - an application survey. IOP Conference Series Materials Science and Engineering, 121, 012010.https://doi.org/10.1088/1757- 899x/121/1/012010

15) Ramalingam, Govindarajan & Meikandasivam, S. & Vijayakumar, D.. (2019). Cloud Computing Based Smart Energy Monitoring System. International Journal of Scientific & Technology Research. 8. 886-890.O. A. Olanrewaju, "Predicting Industrial Sector's Energy Consumption: Application of Support Vector Machine," 2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Macao, China, 2019, pp. 1597-1600,doi:10.1109/IEEM44572.2019.8978604.

16) R. Mohan, S. Devneni, S. Sumpreet, V. Mohan and N. Pachauri, "Comparative Analysis of Machine Learning Algorithms for the Building Energy Prediction," 2024 2nd International Conference on Device Intelligence,

Computing and Communication Technologies (DICCT), Dehradun, India, 2024, pp. 409-413, doi: 10.1109/DICCT61038.2024.10532823.

17) S. Iram et al., "An Innovative Machine Learning Technique for the Prediction of Weather Based Smart Home Energy Consumption," in IEEE Access, vol. 11,pp. 76300-76320, 2023, doi: 10.1109/ACCESS.2023.3287145.

18) Sankari, Siva & C Dr, Jayakumar. (2022). Short-Term Load Forecasting Using Random Forest with Entropy-Based Feature Selection. 10.1007/978-981-16- 6448-9_8.

19) Sereflisan, O., & Koyuncu, M. (2024). A Review Study on Energy Consumption in Cloud Computing. International Journal For Multidisciplinary Research. https://doi.org/10.36948/ijfmr.2024.v06i0 1.11927

20) Tso,G.K., & Yau,k.k. (2007). Predicting electricity energy consumption: A comparison of regression analysis, decision tree and nerual networks.Energy,32(9), 1761–1768.https://doi.org/10.1016/j.energy.2006.11.010

21) V. Lupu, "Cloud computing and continuous energy consumption monitoring," 2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC), Madeira, Portugal, 2017, pp. 119-123, doi: 10.1109/ICE.2017.8279878.

22) Yolchuyev, "A Novel Approach for Optimal Data Uploading to the Distributed Cloud Storage Systems," 2019 Big Data, Knowledge and Control Systems Engineering (BdKCSE), Sofia, Bulgaria, 2019,pp.1-6, doi: 10.1109/BdKCSE48644.2019.9010614.

23) Yu, Z., Haghighat, F., Fung, B. C., & Yoshino, H. (2010). A decision tree method for building energy demand modeling. Energy and Buildings, 42(10), 1637–1646. that perform the classification of the data set . https://doi.org/10.1016/j.enbuild.2010.04.0 06

24) Y. Jiang, Q. Wu, Y. Yang and X. Chi, "Design of a Home Energy Management System Based on Cloud Service," 2018 2nd IEEE Conference on Energy Internet and Energy System Integration (EI2), Beijing, China, 2018, pp. 1-5, doi: 10.1109/EI2.2018.8582274

25) Zhao, H., & Magoulès, F. (2012). Feature selection for predicting building energy consumption based on statistical learning method. Journal of Algorithms & Computational Technology, 6(1), 59–77. https://doi.org/10.1260/1748-3018.6.1.59

26) Z. Wu and W. Chu, "Sampling Strategy Analysis of Machine Learning Models for Energy Consumption Prediction," 2021 IEEE 9th International Conference on Smart Energy GridEngineering (SEGE), Oshawa, ON, Canada, 2021, pp. 77-81, doi: 10.1109/SEGE52446.2021.9534987

Reference links

a) https://www.srldc.in/Monthly-Reports

b) https://power.larc.nasa.gov/data-access-viewer/

c) https://www.wunderground.com/history/monthly/in/chennai/ICHENN51

d) https://github.com/energyauditeee/streamlit-cloud/blob/main/home.py

e) https://app-cloud-fkkex5gmsredm7ur8oqhva.streamlit.app/

LIST OF PUBLICATIONS

CONFERENCE PRESENTATION

Dr. P. Anbalagan, R. Keerthana, V. R. Preethi, J. K.Raghavarthinii, S. L. Sakthipriya, "A Literature Review For Cloud Based Efficient Energy Audit Management System Using Machine Learning" , May - 2025, National Conference On INNOVATIVE TRENDS in SCIENCE, ENGINEERING, TECHNOLOGY and MANAGEMENT - NCITSETM - 25

Paper Under Publication

Dr. P. Anbalagan, R. Keerthana, V. R. Preethi, J. K.Raghavarthinii, S. L. Sakthipriya, "A Literature Review For Cloud Based Efficient Energy Audit Management System Using Machine Learning", International Journal Of Engineering Research And Technology (IJERT), ISSN:2278-0181

SHREE VENKATESHWARA
HI-TECH ENGINEERING COLLEGE

(AUTONOMOUS)

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai.
Accredited by **NBA** (CIVIL, CSE, EEE, MECH) & **NAAC** with **'A'** Grade
Gobichettipalayam - 638 455, Erode (Dt), Tamilnadu

AICTE AQIS SCHEME for GOC SCHEME

Sponsored

National Conference

This is to certify that ~~Mr/Ms~~/Dr. ...P. ANBALAGAN.....AP(SL.G)/EEE..... *of* ...UNIVERSITY.....COLLEGE.....OF.....ENGINEERING..... *has presented the paper entitled* .A..LITERATURE...REVIEW....FOR....CLOUD....BASED...EFFICIENT..ENERGY AUDIT...MANAGEMENT.....SYSTEM..... *in the AICTE AQIS Scheme for GOC Scheme sponsored National Conference on INNOVATIVE TRENDS in SCIENCE, ENGINEERING, TECHNOLOGY and MANAGEMENT-NCITSETM-25 on 14th&15th May, 2025 at our college.*

Coordinator

15/05/2025

Principal

SHREE VENKATESHWARA
HI-TECH ENGINEERING COLLEGE

(AUTONOMOUS)

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai.
Accredited by **NBA** (CIVIL, CSE, EEE, MECH) & **NAAC** with **'A'** Grade
Gobichettipalayam - 638 455, Erode (Dt), Tamilnadu

AICTE AQIS SCHEME for GOC SCHEME

Sponsored

National Conference

This is to certify that ~~Mr~~/Ms/~~Dr~~. ...R.KEERTHANA.....IV-YEAR/EEE..... of ...UNIVERSITY...COLLEGE...OF.....ENGINEERING..... has presented the paper entitled ..A...LITERATURE...REVIEW...FOR...CLOUD...BASED..EFFICIENT..ENERGY AUDIT...MANAGEMENT....SYSTEM... in the AICTE AQIS Scheme for GOC Scheme sponsored National Conference on INNOVATIVE TRENDS in SCIENCE, ENGINEERING, TECHNOLOGY and MANAGEMENT-NCITSETM-25 on 14th&15th May, 2025 at our college.

15/05/2025

Coordinator **Principal**

SHREE VENKATESHWARA
HI-TECH ENGINEERING COLLEGE

(AUTONOMOUS)

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai.
Accredited by **NBA** (CIVIL, CSE, EEE, MECH) & **NAAC** with **'A'** Grade
Gobichettipalayam - 638 455, Erode (Dt), Tamilnadu

AICTE AQIS SCHEME for GOC SCHEME

Sponsored

National Conference

This is to certify that ~~Mr/Ms/Dr~~. ....V.R..PREETHI.....IV-YEAR/EEE..... of ....UNIVERSITY.....COLLEGE.....OF.....ENGINEERING..... has presented the paper entitled ...A...LITERATURE....REVIEW....FOR....CLOUD....BASED....EFFICIENT ENERGY AUDIT MANAGEMENT...SYSTEM in the AICTE AQIS Scheme for GOC Scheme sponsored National Conference on INNOVATIVE TRENDS in SCIENCE, ENGINEERING, TECHNOLOGY and MANAGEMENT-NCITSETM-25 on 14th&15th May, 2025 at our college.

Coordinator

15/05/2025

Principal

SHREE VENKATESHWARA
HI-TECH ENGINEERING COLLEGE

(AUTONOMOUS)

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai.
Accredited by **NBA** (CIVIL, CSE, EEE, MECH) & **NAAC** with '**A**' Grade
Gobichettipalayam - 638 455, Erode (Dt), Tamilnadu

AICTE AQIS SCHEME for GOC SCHEME

Sponsored

National Conference

This is to certify that Mr/Ms/Dr. J.K.RAGHAVARTHINI IV-YEAR/EEE *of* UNIVERSITY COLLEGE OF ENGINEERING *has presented the paper entitled* A LITERATURE REVIEW FOR CLOUD BASED EFFICIENT ENERGY AUDIT MANAGEMENT SYSTEM *in the AICTE AQIS Scheme for GOC Scheme sponsored National Conference on INNOVATIVE TRENDS in SCIENCE, ENGINEERING, TECHNOLOGY and MANAGEMENT-NCITSETM-25 on 14th&15th May, 2025 at our college.*

Coordinator

15/05/2025

Principal

SHREE VENKATESHWARA
HI-TECH ENGINEERING COLLEGE

(AUTONOMOUS)

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai.
Accredited by **NBA** (CIVIL, CSE, EEE, MECH) & **NAAC** with **'A'** Grade
Gobichettipalayam - 638 455, Erode (Dt), Tamilnadu

AICTE AQIS SCHEME for GOC SCHEME

Sponsored

National Conference

This is to certify that ~~Mr/Ms/Dr.~~ ....S.L.SAKTHIPRIYA.....IV-YEAR/EEE.... of ...UNIVERSITY....COLLEGE...OF.....ENGINEERING..... has presented the paper entitled .A..LITERATURE....REVIEW...FOR..CLOUD..BASED..EFFICIENT..ENERGY AUDIT...MANAGEMENT....SYSTEM..... in the AICTE AQIS Scheme for GOC Scheme sponsored National Conference on INNOVATIVE TRENDS in SCIENCE, ENGINEERING, TECHNOLOGY and MANAGEMENT-NCITSETM-25 on 14th&15th May, 2025 at our college.

Coordinator

15/05/2025

Principal