Ensemble learning methods for forecasting solar power generation depending on meteorological parameters

Akash Sathish Kumar Pillai, Sarun Sabu, Jesvin Varughese Jose
22300471, 22300623, 22304920
(Deggendorf Institute of Technology)
akash.sathish-kumar-pillai@stud.th-deg.de
sarun.sabu@stud.th-deg.de
jesvin.jose@stud.th-deg.de

***Abstract*—In this study, we analyze the application of machine learning techniques to predict solar power output based on various parameters. We employ three powerful regression algorithms, Decision Tree Regression, Random Forest Regression, and Gradient Boosting Regression.**

To enhance predictive accuracy, we took advantage of an ensemble approach by combining the individual models using Voting Regression. This ensemble model manipulating the strength of each base model, resulting in a robust and well-generalized predictor for solar power generation.

Furthermore, we present a user-friendly and interactive web-based User Interface (UI) that allows users to input relevant parameters and obtain real-time predictions for solar power output. The UI facilitates accessibility and usability, serve to both technical and non-technical users.

Our study aims to demonstrate the effectiveness of ensemble machine learning models in predicting solar power.

***Index Terms*—model, ensemble, user interface, prediction**

I. Introduction

As the world struggles with the imperative to shift towards sustainable energy sources, the prominence of renewable energy, especially solar power, has become increasingly evident. To mitigate the environmental impact of fossil fuels and meet escalating power demands, there is a global push towards harnessing the potential of renewable resources. The renewable energy landscape is evolving with projections indicating a significant increase in the share of electric power generation from renewable sources.

This paper delves into the realm of solar power generation and employs the applications of artificial intelligence via machine learning. In this changing environment, it becomes imperative to accurately forecast solar power generation. This paper describes a real-world case study that was carried out to investigate the impact of artificial intelligence on a solar photovoltaic power plant with a capacity of kW. The importance of these forecasts goes beyond operational effectiveness, instead, we concentrate on meteorological and environmental factors, offering a thorough examination of variables associated with weather and wider environmental effects in relation to solar power generation. This work aims to develop a reliable solar power prediction technique that is customized to the distinct features of the collected data, in light of the increasing number of solar power installations. Although there are many different prediction techniques, the study focuses on ensemble machine learning (EML) algorithms because of their proven success in developing accuracy. The approach, findings, and implications of utilizing an ensemble machine learning model for models such as decision tree regressor, bagging (random forest regressor), and boosting (gradient boosting) for regional solar power prediction will be covered in detail in the following sections.

The field data was gathered from the Solar Radiation Resource Assessment (SRRA) center's open source platform. It contains a number of factors that have been preprocessed (such as feature selection and data cleaning) and examined. In order to estimate solar power output under the influence of climatic conditions, the processed dataset has been fed into a number of machine learning models (Table I). As of right now, it has been determined that when it comes to prediction accuracy, the EML models outperform the other traditional regression models. In this work, three EML models have been selected: voting, bagging, and boosting. The testing set validates the algorithms' performance, which reveals a prediction accuracy of about 80%. Therefore, given a specific geographic location, the suggested EML algorithms can be highly helpful in predicting the performance of even large-scale solar power projects. Additionally, it can be deduced that the suggested model illustrates the beneficial use of these models through a user interface that is simple to navigate and can be utilized for prediction with appropriate identification of suitable parameters. By combining advanced machine learning methods with an intuitive user interface, solar power prediction apps are developed and renewable energy resources are utilized more effectively.

The work is structured as our proposed research and analysis technique is explained in Section 3. Section 3.A. goes into detail on the setup, pre-processing, analysis, and impact of various meteorological elements on the generation of solar power as well as the procedure of collecting datasets. In

Section 3.C, the different learning strategies employed in this research are discussed. Section 6 presents an analysis of the final result.

Sl. no.	Parameter
1	Temperature
2	Humidity
3	Precipitation
4	Snow
5	Cloud cover
6	Wind speed and direction
7	Zenith
8	Azimuth

TABLE I
FEATURES

II. LITERATURE REVIEW

D. Chakraborty et al. [1] in this article the authors focuses on to generalized method which provides insightful information for forecasting large-scale solar power plant performance under a range of meteorological scenarios using Ensemble Machine Learning models.

Leo et al. [2] in this paper the authors points out the use of random forests in predictive modeling, key aspects like he significance of strength and correlation between individual trees, the convergence of the generalization error, and the combination of tree predictors. It also proposes directions for future research, such as combining random features with boosting and developing theoretical frameworks to comprehend the behavior of the model as well as it also offers insights into how randomness contributes to precise classification and regression outcomes.

M. Alaraj et al. [3] developed a study which discusses the urgent necessity to switch from fossil fuels to renewable energy sources because of the rising expense of crude oil and environmental concerns, especially solar photovoltaic (SPV) technology. The paper highlights the difficulty of integrating solar power on a broad scale into the grid and suggests a unique machine learning method based on ensemble trees for precise SPV power forecasting.

A. Natckin et al. [4] proposed that gradient boosting machines (GBMs), highlighting their adaptability and success in a range of real-world applications and emphasizes how GBMs can be tailored to meet specific requirements and how they can be trained to accommodate a variety of loss functions. The course focuses a lot of emphasis on the useful applications of machine learning modeling in addition to exploring the theoretical foundations of gradient boosting techniques.

Ye Ren et al. [5] developed a thorough evaluation in solar irradiance and wind speed/power forecasting.The competitive and cooperative approaches to ensemble forecasting are further subdivided into cooperative methods based on pre- and post-processing and competitive methods based on diversity of data and parameters.

Amarasinghe et al. [6] in this article the authors emphasize the importance of executing an ensemble model approach that integrates deep learning approaches for accurate solar power forecasting across 21 solar photovoltaic installations located in Germany, in order to meet this difficulty in power system planning and operation. One of the most important steps is using a feature selection procedure to identify the most important meteorological characteristics for solar power generation.

III. METHODOLOGY

In the process of developing our deep learning system, we initiated by collecting diverse datasets relevant to our research objectives. The collected data underwent a thorough cleaning process, where we addressed missing values, outliers, and any inconsistencies to ensure the integrity and quality of our dataset. Subsequently, we employed standardization techniques to normalize the features, promoting uniformity and aiding the convergence of machine learning algorithms during the training phase. Visualization played a pivotal role in identifying trends and making informed decisions regarding feature engineering and model selection (Fig. 1.).

The heart of our system lies in the development of machine learning models. We partitioned our dataset into training and test sets to facilitate the model training process. Employing popular machine learning frameworks like bagging and boosting, we trained models to learn from the patterns in the data and make accurate predictions. Upon achieving satisfactory model performance, we transitioned to the deployment phase. An API (Application Programming Interface) provides a seamless and standardized interface for interacting with our trained models. This API allows for efficient integration with various applications and systems, enhancing the accessibility of our deep learning capabilities. To provide a user-friendly interface for end-users, the UI was designed with a focus on simplicity and flexibility to cater to a wide range of users, from domain experts to non-technical stakeholders.

This end-to-end methodology ensures the development of a robust and scalable deep learning system, encompassing data collection, cleaning, standardization, visualization, model creation, API development, and UI design, all contributing to a comprehensive solution for our research objectives.

A. Data Collection

Data collection is one of the major steps involved in every machine-learning project. In this case study, we collected data from free and open web sources. In addition to that, we clearly outlined the goals and objectives of the data collected and to understand what insights or knowledge the collected data aims. Effective data collection is a fundamental step in the research or decision-making process, laying the groundwork for subsequent data analysis, interpretation, and application.

B. Data Preprocessing

Data preprocessing is a pivotal step in the data science pipeline that involves cleaning, transforming, and organizing raw data into a format suitable for analysis or model training. The goal is to enhance the quality of the data, address any

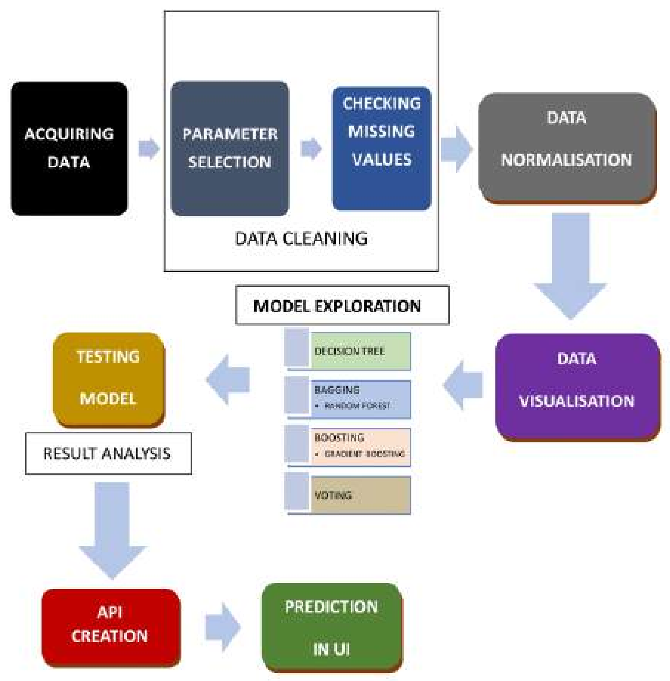


Fig. 1. Methodology Flow Chart

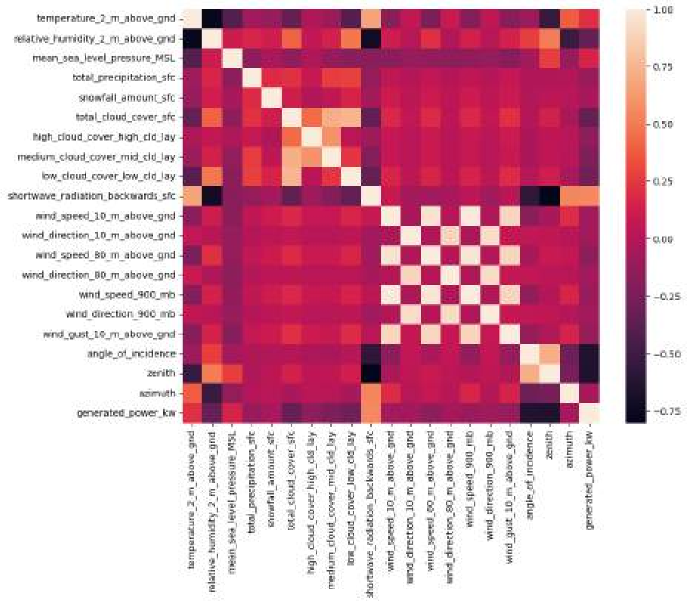


Fig. 2. Correlation Plot

inconsistencies or errors, and prepare it for meaningful insights or machine learning tasks. Initially handling missing data and dealing with outliers to prevent them from disproportionately influencing analysis or model training. Followed by data standardization for scaling numerical features to a standard range, ensuring uniformity and aiding the convergence of machine learning algorithms.

Exploratory Data Analysis (EDA) utilizes visualization libraries such as Matplotlib and Seaborn to explore data distributions, patterns, and relationships. Identify potential features and relationships that may be relevant to the research question. We conducted a thorough analysis of correlation among independent variables using a heatmap (Fig. 2.) as well as assessed the multicollinearity which, is not a significant concern for our models as we employ bagging and boosting techniques, which are known to be robust to correlated features.

C. Model Creation and Exploration

Initially, the dataset is divided into two sets as an 80% training set and a 20% testing set. This division allows for the model to be trained on the larger portion of the data. The training set is utilized to teach the model the underlying patterns and relationships within the data, while the testing set serves as a benchmark to assess how well the model generalizes to new data. This train-test split is a common practice in machine learning to ensure that the model's performance is robust and applicable beyond the data it was trained on. Model creation focuses on designing and training a machine learning model while, model exploration is dedicated to understanding and analyzing the trained model's behavior, strengths, and weaknesses. Both phases are crucial for developing reliable and effective machine learning solutions. Model exploration informs decisions about model deployment, improvements, and potential iterations based on a deeper understanding of how the model interacts with the data. In the realm of model creation, we focus on the prepared dataset, guided by the insights gleaned from captivating visualizations. Referring to the pair plot (Fig. 3.), it becomes evident that the majority of data points overlap, rendering linear and logistic models are unsuitable. These models rely on linear boundaries, and in this scenario, the dense overlap among data points impedes their effectiveness in distinguishing between classes. Alternative models with non-linear decision boundaries may be more appropriate for capturing the complex relationships present in the data.

By thoroughly exploring the model, we were able to make informed decisions regarding model deployment, improvements, and further iterations. This exploration process contributes to building trust in the model's predictions and enhances its applicability in real-world scenarios.

- Decision Tree Regressor
  The Decision Tree Regressor is a machine learning algorithm designed for regression tasks, where the goal is to predict a continuous target variable based on input features. This algorithm constructs a tree-like structure by recursively splitting the input space, creating decision nodes that guide predictions to leaf nodes. Each leaf node corresponds to a specific predicted value for the target variable. In equation (1), it represents entropy for the decision tree by which the nodes are executed in a decision tree. H(t) is the entropy at node t, c is the number of classes (distinct target values), and p(i/t) is the proportion of samples in class i at node t. The entropy is calculated for each node during the tree-building process. The de-

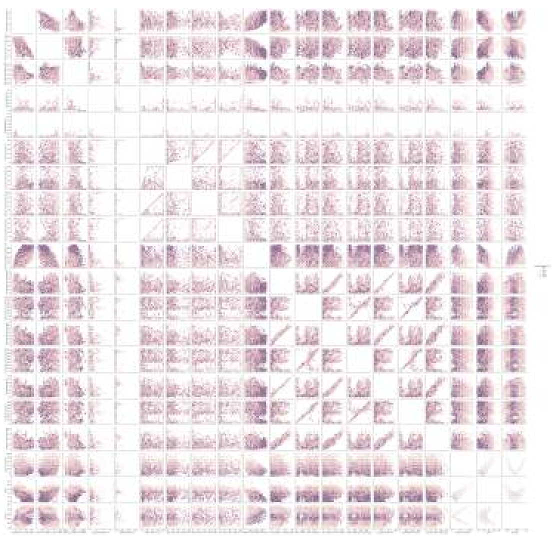


Fig. 3. pairplot of parameters in dataset

cision tree algorithm aims to maximize information gain when selecting the best feature to split on. The algorithm is particularly advantageous for its interpretability, as the resulting tree can be easily visualized. Key parameters, such as maximum depth and minimum samples per split, allow for control over the model's complexity.

$$H(t) = -\sum_{i=1}^{c} p(i|t) \log_2(p(i|t)) \tag{1}$$

- Random Forest Regression
  Random Forest Regression is a powerful ensemble learning algorithm that incorporates the principles of bagging (Bootstrap Aggregating) [3] to enhance predictive accuracy. Since the random forest algorithm constructs a collection of decision trees (the decision tree regressor aids in reducing the execution time for random forest regressor), where each tree is trained on a different subset of the data, and their predictions are aggregated to form a more robust and accurate model. In equation (2), Q(t) denotes the impurity improvement attained by splitting, H(t) is the impurity measure at node t, H(left) and H(right) are the impurity measures of both the left and right child nodes, N_t is the total number of samples at node t, and N_{left} and N_{right} are the number of samples in the left and right child nodes, respectively.

$$Q(t) = H(t) - \left(\frac{N_{\text{left}}}{N_t} H(\text{left}) + \frac{N_{\text{right}}}{N_t} H(\text{right}) \right) \tag{2}$$

- Gradient Boosting
  Boosting aims to minimize the expected loss, the loss function quantifies the penalty for the difference between the predicted values and the true labels. Gradient Boosting is an ensemble learning technique that builds a predictive model by iteratively combining weak learners, often decision trees, to correct errors made by the existing ensemble [5]. Sequentially added models focus on instances with prediction mistakes, optimizing the overall model for both bias and variance. The algorithm utilizes gradient descent to minimize a chosen loss function, adjusting the parameters of each weak learner to enhance predictive accuracy. Gradient Boosting Machines offer various optimizations, making Gradient Boosting suitable for diverse data types and tasks. While known for high predictive accuracy, its computational intensity and the need for careful hyperparameter tuning are considerations. Overall, Gradient Boosting stands out for its robustness and flexibility, making it a valuable tool in machine learning applications.

$$F^*(x) = \text{argmin}\, E(y,x)\psi(y, F(x)) \tag{3}$$

  The function $F^*(x)$ establishes a mapping from the input variable x to the output variable y. By minimizing ψ (y, F(x)) (the joint distribution of (x,y), the expected loss over this mapping is intended to be as small as possible. The minimization function is represented in equation (3). Building new weaker base learners with the highest correlation to the previously mentioned loss function's negative gradient is the key concept behind this technique. These newly created models or base learners, are sequentially combined to form the final ensemble. Each learner is trained by evaluating the error or loss in the ensemble.
  Gradient boosting machines are recognized for their configurational flexibility. Among various available loss functions, one can be selected to effectively train the ensemble, offering versatility in adapting to diverse scenarios.
- Voting Regressor
  The Voting Regressor in scikit-learn is used to ensemble the models used, by which it combines regression models to enhance predictive accuracy. Operating on the principle of aggregating predictions from diverse models, it offers two voting strategies 'hard' for majority voting and 'soft' for averaging predicted probabilities. Creating a Voting Regressor involves specifying base regression models and the chosen voting strategy. The Voting Regressor aggregates predictions from decision trees, random forests, and gradient boosting, offering a diverse set of models to improve overall predictive performance. Decision trees provide simplicity and interpretability, while random forests bring robustness and reduce overfitting. Gradient boosting, on the other hand, sequentially corrects errors and enhances predictive accuracy. By combining these models in a Voting Regressor, it could benefit from their complementary strengths, creating an ensemble that is resilient, accurate, and adaptable to different aspects of the data, its effectiveness relies on the diversity and independence of the underlying models, with careful consideration of hyperparameter tuning for the base models. Fine-tuning the hyperparameters of each base model and

the voting strategy is essential for achieving optimal results.

$$F(x_i) = \frac{\sum_{m=1}^{M} w_m \cdot f_m(x_i)}{\sum_{m=1}^{M} w_m} \tag{4}$$

If 'soft' voting is chosen, the final prediction for x_i is the weighted average of the predictions from all base models. The weights are typically proportional to the performance or confidence of each base model. In equation (4) the mathematical representation of x_i is represented as F(x_i) where, F(x_i) is the final prediction, $f_m(x_i)$ is the prediction of m-th base model and w_m is the weight assigned to the model. Whereas, in the case of 'hard' voting, the final prediction is opted with the one with the majority of votes.

D. Application Programming Interface

An API, or application programming interface, is a collection of protocols and tools for creating software applications that facilitate communication between different software applications. A well-designed API is crucial for providing a positive user experience and ensuring that users can easily navigate and interact with the application. It allows different software systems to communicate with each other, enabling us to seamless integration and data exchange. API defines the methods and data formats that applications can use to request and exchange information, providing a standardized way to interact with services or functionalities offered by other software components.

In this project, we have developed a Task Management System using Flask, a lightweight and extensible web framework for Python, and PyCharm, an integrated development environment (IDE) specifically designed for Python development. The goal of the project was to create a simple and efficient API for managing tasks, including functionalities for task creation, retrieval, and deletion. We have transformed our trained model into a pickel file which we utilized in Flask's simple and intuitive design to create API endpoints for managing tasks. Flask routes were defined for handling actions such as calling the model in the back end and retrieving data to the front end. Flask utilizes the WSGI (Web Server Gateway Interface) toolkit Werkzeug (Table II) to deal with the more complex parts of web development. It provides utilities for routing, debugging, and serving as the foundation for building web applications. We have facilitated the serialization of data into commonly used formats like JSON for data interchange in APIs.Flask provides a request object that simplifies the extraction of data from incoming HTTP requests. By leveraging the features of Flask and adhering to principles, we were able to create scalable, maintainable, and well-documented APIs. Flask's simplicity and extensibility, combined with its vibrant ecosystem, make it a powerful tool for building our web-based applications and services. In the context of our project, Flask serves as an excellent foundation for developing a Task Management System API with efficiency and ease.

E. User Interface

In this comprehensive project, we have not only developed a robust task management system API using Flask but have also seamlessly integrated a user-friendly HTML User Interface for an enhanced user experience. The interface comprises all the elements that users interact with on a screen, such as buttons, icons, photos, text, input fields, and more. The project showcases the synergy between Flask's back-end capabilities and the intuitiveness of HTML for front-end design. We leveraged Flask's Jinja2 template engine (Table II) to dynamically generate HTML pages based on back-end data. The UI components include an interactive page for feeding input parameters, which results in the prediction of solar power generation (Fig. 4.).

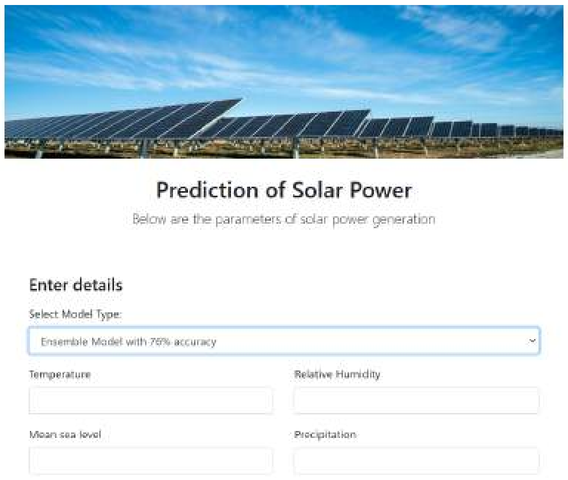
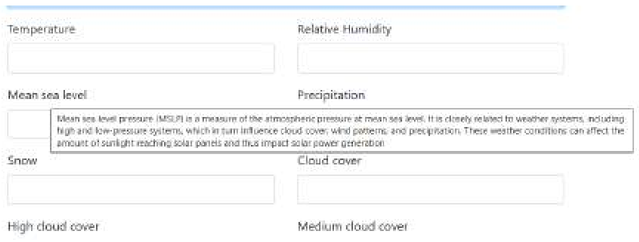

Prediction of Solar Power
Below are the parameters of solar power generation
Enter details
Select Model Type:
Ensemble Model with 76% accuracy
Temperature
Relative Humidity
Mean sea level
Precipitation

Fig. 4. User Interface

In addition to that, we have also implemented tooltips that provide explanations for each input parameter, ensuring that users can easily understand the purpose and function of each field. This added feature enhances the user experience by making the interface more informative and accessible to all users (Fig. 5.).



Mean sea level pressure (MSLP) is a measure of the atmospheric pressure at mean sea level. It is closely related to weather systems, including high and low-pressure systems, which in turn influence cloud cover, wind patterns, and precipitation. These weather conditions can affect the amount of sunlight reaching solar panels and thus impact solar power generation

Fig. 5. Tooltip

IV. Software Requirements

To develop a well-rounded machine learning system and seamlessly integrate the entire product, our team has leveraged a diverse set of software facilities. Beginning with the foundation of our project, we employed versatile programming tools for building essential components. We ensured smooth collaboration and version control through the adoption of Pycharm, providing a rugged platform for code management and collaboration. For the computational power required in training and running our models, we incorporated anaconda navigator tools, harnessing the parallel processing capabilities of NVIDIA GPUs. This strategic integration optimizes the efficiency of our machine learning model, enhancing its performance and expediting training processes. In tandem with these tools, we have also utilized popular machine learning frameworks like Flask, empowering our team to develop an API(Application programming interface). Additionally, the project is deployed in a user-friendly interface which can enhance the practicality and adoption of our machine learning system.

A. Anaconda Navigator

Anaconda Navigator is designed to simplify the management and deployment of data science and machine learning tools conveniently to navigate and access various components, such as Jupyter Notebooks, Spyder IDE, and other data science applications. In which our team opted for the Jupyter Notebook, which is an open source, an interactive web application for machine learning tasks where users can create and combine real time codes. This flexibility makes it an ideal environment for exploratory data analysis, prototyping, and collaborative research as well as it operates on a cell-based structure, where each cell can contain code, markdown text, or visual outputs. One of its notable features is its seamless integration with Anaconda's package management system, allowing users to effortlessly install, update, and manage Python packages and dependencies. The navigator's intuitive design caters to both beginners and experienced data scientists, facilitating a smooth workflow for tasks ranging from project development to the execution of complex machine learning models. With its emphasis on ease of use and comprehensive functionality, Anaconda Navigator significantly contributes to the efficiency and productivity of data science and deep learning projects.

B. PyCharm

PyCharm is a powerful integrated development environment (IDE) specifically tailored for Python programming. PyCharm supports the creation of virtual environments, isolating project dependencies for better manageability. It provides a smart code editor with advanced code completion, code analysis, and error highlighting, enabling developers to write clean and error-free code. The integrated version control system facilitates collaboration and code management within a team. In our study, we have used PyCharm's environment, robust debugging tools, unit testing support, and built-in terminal to contribute to bridging the back end and the front end.

C. Python libraries required

For the preparation of the model, we strategically employed key Python libraries like Pandas which facilitated data collection and pre-processing, while Matplotlib empowered insightful exploratory data analysis, NumPy handled data standardization, scikit-learn powered our machine learning models, and libraries including versions (Table II). This amalgamation of Python libraries ensured a streamlined and efficient end-to-end development process.

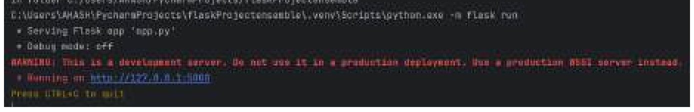
Sl. no.	Library	Version
1	Flask	3.0.0
2	Flask-Cors	4.0.0
3	Flask-MonitoringDashboard	3.2.2
4	numpy	1.26.2
5	pandas	2.1.4
6	scikit-learn	1.3.2
7	kneed	0.8.5
8	xgboost	2.0.2
9	matplotlib	3.8.2
10	Jinja2	3.1.2
11	Werkzeug	3.0.1
12	itsdangerous	2.1.2
13	certifi	2023.11.17
14	Markupsafe	2.1.3
15	gunicorn	21.2.0

TABLE II
LIBRARIES AND VERSIONS

V. HANDELING UI

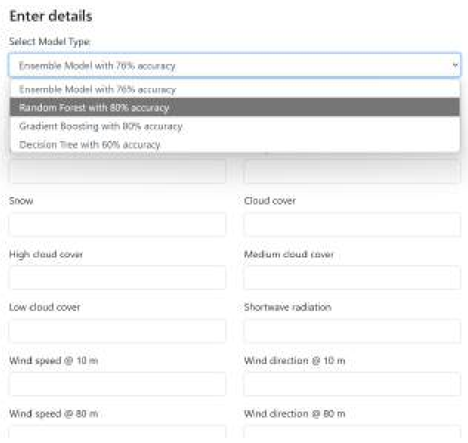
Users can access the graphical user interface through the deployment server link (Fig. 6.) generated in the Python environment, which offers a smooth entry point to a complex system while ensuring a comprehensive and user-friendly experience. The interface is designed to include various meteorological parameters (Table I), with each parameter represented by dedicated input boxes, allowing users to smoothly input the gathered data. The UI is created in a way that both technical and non-technical users can easily handle the system.

In addition to the meteorological inputs, a dedicated input value box is provided where users can select which prediction models they wish to use. The available models include Decision Tree, Random Forest Regression, Gradient Boosting, and an Ensemble Learning model (Fig. 7.). After completing the data input and selecting the desired models, the user simply needs to click the ”Start Predicting” button. This action initiates the interface to execute the chosen algorithms and provides the prediction of solar power generation output in kilowatts (kW). With this predictive capability and model selection feature, the interface becomes an even more powerful tool for analyzing and forecasting electricity generation based on meteorological data.



```
FLASK_DEBUG = 0
In folder C:/Users/AKASH/PycharmProjects/flaskProjectensemble
C:\Users\AKASH\PycharmProjects\flaskProjectensemble\.venv\Scripts\python.exe -m flask run
 * Serving Flask app 'app.py'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Fig. 6. Deployement Link


Enter details

Select Model Type

Ensemble Model with 76% accuracy

Ensemble Model with 76% accuracy
Random Forest with 80% accuracy
Gradient Boosting with 80% accuracy
Decision Tree with 60% accuracy

Snow | Cloud cover
High cloud cover | Medium cloud cover
Low cloud cover | Shortwave radiation
Wind speed @ 10 m | Wind direction @ 10 m
Wind speed @ 80 m | Wind direction @ 80 m

Fig. 7. Handeling UI

VI. Result and Analysis

We conducted a comprehensive analysis of model performance through the examination of residual plots (Fig. 8.) and scatter plots (Fig. 9.) comparing predicted and actual values. Residual analysis allows us to assess the goodness of fit by inspecting the distribution of the differences between predicted and actual values. The scatter plots visually illustrate the model's ability to capture patterns and trends in the data. These diagnostic tools provide valuable insights into the strengths and potential areas for improvement in our predictive model. The close examination of residuals aids in verifying assumptions and ensuring the robustness of our modeling approach.

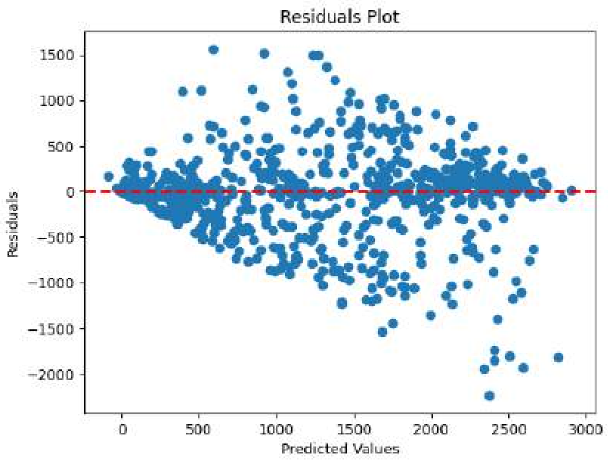


Fig. 8. Residual Plot

VII. Conclusion

In conclusion, the study focuses at how to anticipate solar power generation using ensemble machine learning (EML) techniques in order to increase the accuracy and robustness of solar power generation forecasts. As the demand for sustainable energy solutions continues to rise, the insights and

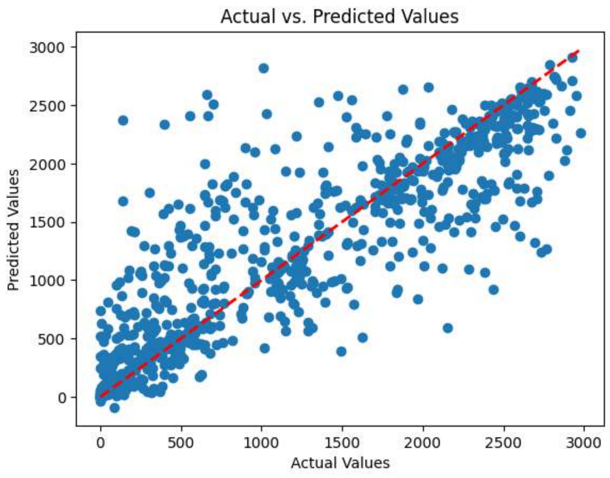


Fig. 9. Scattered Plot

tools generated through this project contribute to the ongoing efforts in optimizing solar power generation forecasting, in accordance with the meteorological conditions.

Furthermore, seamless interaction between the ensemble model and the user interface was made possible by the Flask-based user-friendly API implementation. The implementation of the model within a graphical user interface not only enables user involvement but also improves accessibility for those who wish to easily anticipate the generation of solar electricity. The model's effective implementation in a user interface highlights the created solution's practical applicability and provides consumers in the renewable energy industry and beyond with an approachable tool.

Additionally, it proposes potential extensions to newly constructed and existing solar power plants, as well as avenues for future research to improve the performance of machine learning algorithms and create better prediction algorithms.

Acknowledgment

The authors are thankful to everyone who provided significant assistance and contributions while the paper was being developed. We extend our gratitude to Professor Sunil Survaiya for their guidance and mentorship, which greatly enriched the quality and depth of our research. Special thanks to the Deggendorf Institute of Technology for their support and provision of resources. This work has been made possible through their commitment to advancing research in machine learning prediction.

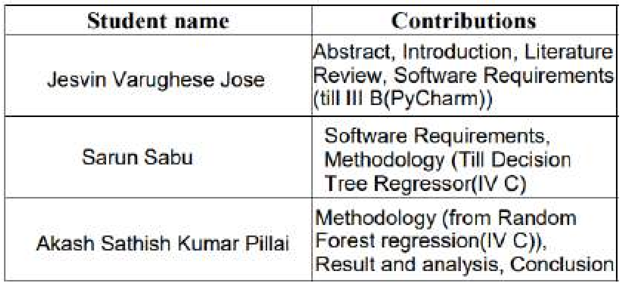
References

[1] D. Chakraborty, J. Mondal, H. B. Barua, A. Bhattacharjee Computational solar energy – Ensemble learning methods for prediction of solar power generation based on meteorological parameters in Eastern India, https://doi.org/10.1016/j.ref.2023.01.006

[2] Leo. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32, doi.org/10.1023/A:1010933404324

[3] M. Alaraj, A. Kumar, I. Alsaidan, M. Rizwan, M. Jamil, Energy production forecasting from solar photovoltaic plants based on meteorological parameters for qassim region, saudi arabia, IEEE Access 9 (2021) 83241–83251, https://doi.org/10.1109/ACCESS.2021.3087345

[4] A. Natekin, A. Knoll, Gradient boosting machines, a tutorial, Front. Neurorobot. 7 (2013) 21, https://doi.org/10.3389/fnbot.2013.00021
[5] Ye Ren and P.N. Sugathan and N. Srikanth, Ensemble methods for wind and solar power forecasting-A state of the art review, doi.org/10.1016/j.rser.2015.04.081
[6] P. A. G. M. Amarasinghe, N.S. Abeygunawardana, T.N. Jayasekara, E.A.J.P. Edirisinghe, S.K. Abeygunawardane, Ensemble models for solar power forecasting—a weather classification approach, doi: 10.3934/energy.2020.2.252



Student name	Contributions
Jesvin Varughese Jose	Abstract, Introduction, Literature Review, Software Requirements (till III B(PyCharm))
Sarun Sabu	Software Requirements, Methodology (Till Decision Tree Regressor(IV C)
Akash Sathish Kumar Pillai	Methodology (from Random Forest regression(IV C)), Result and analysis, Conclusion

Fig. 10. Author Contribution