

Data-Driven State-Wise Solar Energy Forecasting Using Advanced Machine Learning Models

Dr. P. Jayasaradadevi¹, Dr. M.V.S. Ramalakshmi², Dr. B. Ashok³

Department of MBA Business Analytics, Gayatri Vidya Parishad College for Degree and PG Courses (A),
Andhra University, India¹

<https://orcid.org/0000-0002-1256-5186>¹

Department of Management studies, Gayathri Vidya Parishad College for Degree and PG Courses (A),
Andhra University, India²

<https://orcid.org/0000-0003-0237-4017>²

Department of Management Studies, Gayatri Vidya Parishad College for Degree and PG Courses (A),
Andhra University, India³

<https://orcid.org/0000-0001-6323-0549>³

Abstract: In this paper, we present a machine learning framework for forecasting state-wise solar power generation in India, leveraging historical solar output and meteorological data. The models employed Random Forest, Gradient Boosting, and XGBoost are designed to capture the complex, nonlinear relationships inherent in weather-influenced energy generation. To enhance predictive performance and model robustness, hyperparameter tuning techniques are applied. Among the approaches, XGBoost outperforms others, demonstrating superior accuracy and adaptability. These findings emphasize the potential of machine learning in advancing solar energy forecasting and enabling data-driven, sustainable energy planning.

Keywords: Solar forecasting, XGBoost, Hyperparameter tuning, Machine learning, Renewable energy

1. INTRODUCTION

The demand for sustainable and renewable energy sources has risen sharply in recent years, making solar energy one of the most vital contributors to clean energy transitions. India, with its diverse geographical and climatic conditions, presents a unique opportunity for solar energy optimization on a regional scale. Accurate forecasting of solar power generation is crucial to ensure grid stability, efficient energy management, and policy-making.

This Study focuses on developing state-wise solar energy generation forecasts using machine learning models. By leveraging historical solar generation data and meteorological parameters such as temperature, humidity, cloud cover, and wind speed, the models aim to predict solar output across Indian states and union territories. These predictions will support stakeholders in making informed decisions for better energy planning and allocation.

The dataset required for this Study was provided by the organization, eliminating the need for manual data collection. It included pre-compiled historical solar generation and meteorological information across various states, which was further cleaned and preprocessed for analysis. The core objective was to apply and compare the performance of advanced machine learning models **XGBoost, Gradient Boosting, and Random Forest** to generate accurate and reliable solar forecasts.

A significant aspect of model development involved hyperparameter tuning to enhance prediction accuracy and prevent overfitting. Techniques like grid search and random search were employed to optimize parameters such as the number of estimators, maximum tree depth, learning rate, and regularization values. This ensured that each model was operating at its best capacity given the nature of the dataset.

Needs of the Study

- To accurately forecast state-wise solar energy generation.
- To handle the weather-driven variability in solar output using environmental features.
- To aid policy-makers and energy planners with reliable forecasts.
- To enable data-driven resource optimization and operational planning.
- To compare machine learning models and identify the most accurate prediction method.
- To apply hyperparameter tuning techniques (such as Grid Search and Random Search) to optimize model performance and prevent overfitting.

Objectives of the Study

- To preprocess historical weather and solar generation data provided across states, ensuring clean and usable datasets for model training.
- To identify key meteorological features, such as temperature, humidity, and wind speed, that influence solar energy output and enhance model accuracy.
- To develop predictive models using XGBoost, Gradient Boosting, and Random Forest, applying advanced machine learning techniques to forecast solar power generation.
- To perform hyperparameter tuning (using methods like Grid Search and Random Search) to optimize model performance and avoid overfitting, ensuring robust and reliable predictions.
- To compare model performance using evaluation metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R^2) to determine the best-performing model.
- To generate accurate, state-wise solar energy generation forecasts that can be used for energy planning and operational decisions.
- To support data-driven decisions in energy planning and policy formulation, ultimately aiding in more efficient integration of renewable energy sources into the grid.

II. REVIEW OF LITERATURE

The present study is anchored in the broader field of **predictive analytics**, which is grounded in statistical learning theory and data science principles. Predictive analytics enables the use of historical and real-time data to forecast future outcomes through mathematical modeling and algorithmic processing. In the context of solar energy forecasting, it serves as a foundation for developing models that analyze meteorological trends to estimate energy production. This approach is critical for integrating renewable energy sources into the power grid and supports planning efforts aimed at improving energy sustainability.

Machine learning theory, particularly in supervised learning, provides the methodological backbone of this study. The regression-based ensemble models—Random Forest, Gradient Boosting, and XGBoost—used here are known for their ability to capture complex, non-linear relationships within high-dimensional datasets. Ensemble learning, which aggregates predictions from multiple base models, has been shown to significantly enhance predictive accuracy and model robustness. This theoretical grounding allows for improved solar forecasting, especially when dealing with fluctuating environmental variables.

The increasing integration of solar energy into modern power systems has intensified the need for accurate forecasting models. Traditional statistical techniques often struggle to capture the nonlinear relationships between meteorological variables and solar power generation, leading researchers to explore machine learning approaches for enhanced prediction accuracy.

Voyant et al. (2017) conducted a comprehensive review of machine learning approaches for solar radiation forecasting and reported that artificial intelligence techniques such as Artificial Neural Networks (ANN), Support Vector Machines (SVM), and ensemble learning models significantly improve prediction accuracy compared to traditional statistical methods. Their study highlighted the effectiveness of machine learning in handling complex meteorological relationships. Antonanzas et al. (2016) reviewed photovoltaic power forecasting methodologies and emphasized the importance of meteorological variables such as solar irradiance, temperature, humidity, and cloud cover in determining forecasting accuracy. The authors concluded that advanced machine learning models outperform conventional forecasting approaches in capturing nonlinear weather patterns.

Breiman (2001) introduced the Random Forest algorithm, an ensemble learning method that combines multiple decision trees to improve prediction performance and reduce overfitting. The algorithm has been widely applied in renewable energy forecasting due to its robustness, scalability, and ability to model complex interactions among predictor variables. Friedman (2001) proposed Gradient Boosting Machines (GBM), which sequentially construct decision trees by minimizing prediction errors from previous iterations. The study demonstrated that boosting techniques can achieve superior predictive performance in regression tasks and are particularly effective in datasets with nonlinear relationships. Chen and Guestrin (2016) developed Extreme Gradient Boosting (XGBoost), an optimized boosting framework that incorporates regularization, parallel processing, and efficient tree construction techniques. Their research showed that XGBoost consistently achieves higher accuracy and faster computation than traditional boosting methods, making it suitable for large-scale forecasting applications.

Wang et al. (2018) investigated machine learning techniques for solar irradiance prediction and found that ensemble models such as Random Forest and Gradient Boosting significantly outperformed conventional regression methods. Their findings highlighted the importance of feature engineering and model optimization in enhancing forecasting accuracy. Ahmed et al. (2020) reviewed recent advancements in photovoltaic power forecasting and emphasized that hybrid and ensemble machine learning models provide superior forecasting performance. The study also stressed the importance of hyperparameter tuning and cross-validation techniques for improving model reliability and generalization.

Based on the reviewed literature, ensemble machine learning models such as Random Forest, Gradient Boosting, and XGBoost have demonstrated strong capabilities in forecasting renewable energy generation. These studies provide a strong foundation for the present research, which evaluates these algorithms for state-wise solar power prediction in India and examines the impact of hyperparameter tuning on forecasting accuracy.

Although previous studies have demonstrated the effectiveness of machine learning techniques for solar forecasting, most research has focused on solar irradiance prediction or localized photovoltaic systems. Limited studies have investigated state-wise solar power forecasting across India using hyperparameter-optimized ensemble learning models. Furthermore, comparative evaluations of Random Forest, Gradient Boosting, and XGBoost for large-scale regional solar forecasting remain scarce. Therefore, this study addresses this gap by developing and comparing advanced machine learning models for state-wise solar power prediction in India.

III.METHODOLOGY OF THE STUDY

1. Data Collection

- **Solar Generation Data:** The company provided historical daily solar energy generation data for various states across India, which serves as the target variable for the models (State_solar).
- **Meteorological Data:** The meteorological data, including features such as Temperature, Humidity, Precipitation, Pressure, Wind Speed, Wind Direction, Clouds, Population (Pop), Ultraviolet Index (UVI), and Sun Duration, were supplied by the company. These features are crucial for capturing weather-driven variability in solar energy output.

Dataset Description

The dataset contains weather-related and solar energy data recorded across different states over time. Each attribute in the dataset plays a crucial role in understanding the factors affecting solar energy generation. Below is a detailed breakdown of each parameter:

- **STATE:** Name of the state where data was recorded.
- **DATE:** The date corresponding to the data entry.
- **State solar:** Solar energy generation (target variable, but has missing values).
- **Temperature:** The average temperature recorded in Kelvin.
- **Humidity:** The relative humidity percentage.
- **Precipitation:** Amount of rainfall in millimetres.
- **Pressure:** Atmospheric pressure in hPa.
- **Wind Speed:** Speed of the wind in m/s.
- **Wind Directions:** The direction of the wind in degrees.
- **Clouds:** Cloud cover percentage.
- **POP (Probability of Precipitation):** Probability of rain occurrence.
- **UVI (UV Index):** Measurement of ultraviolet radiation exposure.
- **Sun Duration:** Duration of sunlight exposure in minutes.

2. Data Preprocessing

- **Data Cleaning:** Cleaned the data by handling missing values using imputation (mean or median) or removing incomplete records. Identified and removed outliers using statistical methods to prevent skewing the results.
- **Feature Engineering:** Created additional features such as rolling averages to capture long-term trends in weather parameters. Lag features were added to represent time-dependent information, helping the model better capture seasonal and cyclic patterns.
- **Data Splitting:** The dataset was divided into 80% training and 20% testing to ensure proper model training and evaluation on unseen data.
- **Feature Scaling:** Normalized numerical features (such as temperature and humidity) to a standard range to enhance the performance of models like XGBoost and Gradient Boosting.

3. Model Development

Model Selection: Three machine learning models were chosen for training:

- **XGBoost** (Breiman 2001): A powerful gradient boosting algorithm known for high performance and handling large datasets efficiently.
- **Gradient Boosting:** Another boosting model that builds trees sequentially, focusing on correcting errors made by the previous trees.
- **Random Forest:** An ensemble learning technique that builds multiple decision trees in parallel and aggregates their predictions.

Model Training:

- The selected models were trained using the training dataset, where meteorological features were used to predict solar generation (State_solar).
- **Hyperparameter Tuning:** Hyperparameter optimization was performed using techniques like grid search and random search to fine-tune model parameters (e.g., number of trees, learning rate, depth of trees) for each model. This was done to enhance model performance and prevent overfitting.

4. Model Evaluation

Performance Metrics: The models were evaluated using the following metrics:

- **Mean Absolute Error (MAE):** To assess the average magnitude of the errors in the predictions.
- **Root Mean Squared Error (RMSE):** To measure the prediction error, giving more weight to larger errors.
- **R-squared (R^2):** To evaluate how well the model fits the data and explains the variance in solar generation.

Cross-Validation: K-fold cross-validation was performed to ensure the models' robustness and generalization capability, preventing overfitting to the training data.

5. Forecast Generation

- Once the models were trained and evaluated, they were used to generate future solar generation forecasts for various states, leveraging the meteorological features available for the corresponding time periods.
- Predictions were made for several future days/weeks, depending on the available dataset, and results were compared to actual solar generation values (once available).

6. Model Comparison

- The performance of XGBoost, Gradient Boosting, and Random Forest was compared using the evaluation metrics (MAE, RMSE, R^2).
- The model that performed best on the test dataset, based on hyperparameter tuning and evaluation metrics, was selected as the final model for forecasting solar generation.

7. Final Model Selection

Based on model performance metrics, hyperparameter tuning results, and insights, the best-performing model (XGBoost, Gradient Boosting, or Random Forest) was selected for deployment.

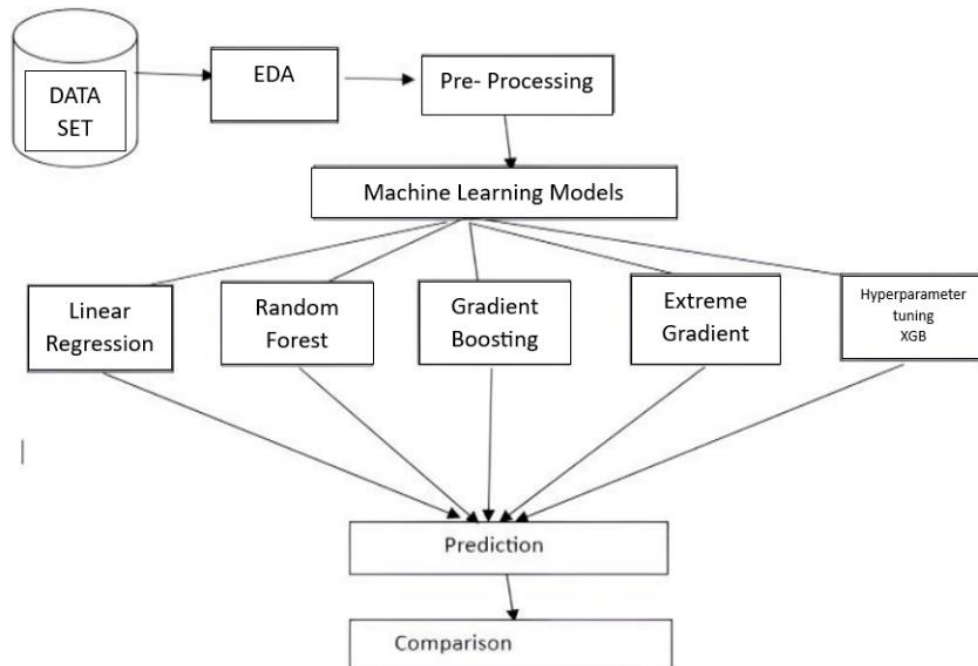


Fig 1: Process Flow Chart
Source: Prepared by the author based on the study data

IV.RESULTS AND DISCUSSION

1.1 Dataset

Data can be divided into two sets:

- **Historical Data:** Used for training the model.
- **Inference Data:** Used for making predictions.

1.2 Data Loading: Historical and inference datasets were loaded using `pandas.read_excel()`, confirming complete data in the historical set and missing `State_solar` values in the inference set for prediction.

1.3 EDA (Exploratory Data Analysis)

- Missing value analysis showed no nulls in the historical dataset, making it fit for training, while the inference dataset had 839 missing `State_solar` values expected for prediction.
- If `State_solar` was categorical, it was encoded using `LabelEncoder` to support regression modeling.
- Descriptive statistics confirmed complete numerical data with expected variability in features like Temperature and Humidity, while Precipitation remained consistently zero.
- The `.info()` output confirmed all 8117 records are complete with no missing values, and data types are appropriate—primarily `float64` for numerical features and `object` for categorical ones like `STATE` and `DATE`.
- The correlation heatmap shows how numerical features relate to each other and to the target variable `State_solar`. Notably, `Wind_Speed` (0.29) and `Temperature` (0.15) have a moderate positive correlation with `State_solar`, while `Humidity` (−0.24) and `Clouds` (−0.13) show negative correlations. Strong correlations between features like `Temperature` and `Sun_Duration` (0.74).

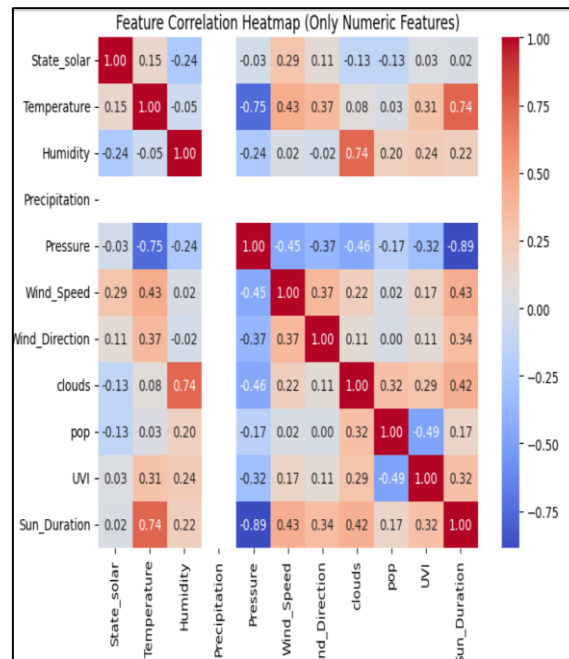


Fig 2 Correlation Matrix

- Outliers were detected and removed using the Interquartile Range (IQR) method. Outlier removal was successful, especially for Temperature, Wind Speed, and Pressure, where many extreme values were originally present.
- A custom function calculated the IQR and excluded values beyond 1.5 times the interquartile limits. This reduced the dataset from 8117 to 7554 records, confirming that 563 outlier rows were removed.

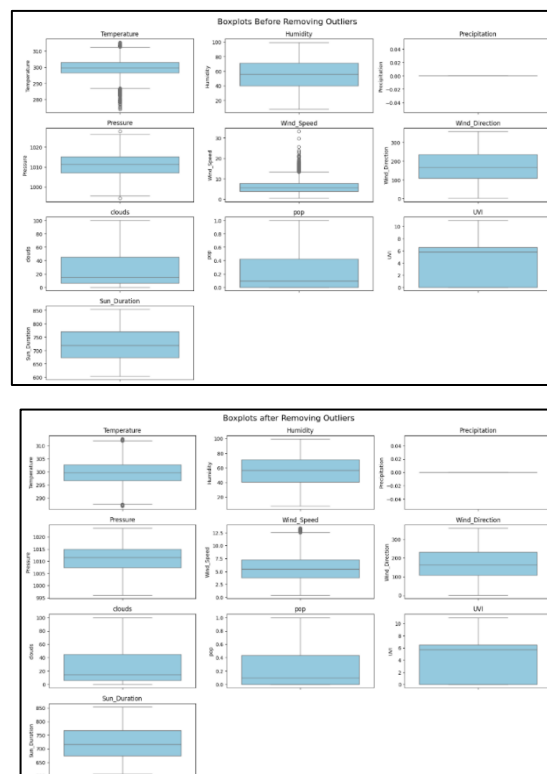


Fig 3 Graphical representation of before and after removal of Outlier

- Min-Max Scaling to normalize feature values between 0 and 1. The MinMaxScaler is first fitted to the training data X, then both the training data (X) and new data (X_new) are transformed using the same scaling parameters to ensure consistency.

1.4 Define Feature and Target

- The selected input features (Temperature, Humidity, Precipitation, Pressure, Wind_Speed, Wind_Direction, clouds, pop, UVI, and Sun_Duration) were extracted from both the historical dataset (X) and the inference dataset (X_new).
- The corresponding target variable for the historical dataset was State_solar (y), representing solar power output by state.

1.5 Model Selection & Training

➤ Linear Regression (Code)

```
# Initialize and train Linear Regression model
lr = LinearRegression()
lr.fit(X_normalized, y)
# Predict on inference data
y_pred_lr = lr.predict(X_new_normalized)
inference_data["Predicted_State_solar_LR"] = y_pred_lr
# Evaluate on training data
y_train_pred = lr.predict(X_normalized)
# Evaluation metrics
r2_lr = r2_score(y, y_train_pred)
mae_lr = mean_absolute_error(y, y_train_pred)
mse_lr = mean_squared_error(y, y_train_pred)
# Print results
print(f"Linear Regression R2: {r2_lr:.2f}")
print(f"Mean Absolute Error (MAE): {mae_lr:.2f}")
print(f"Mean Squared Error (MSE): {mse_lr:.2f}")
```

Model Evaluation Summary: The Linear Regression model showed limited performance with an R^2 score of 0.16, indicating it explains only 16% of the variance in solar generation. The MAE was 13.72, and the MSE was 463.5, reflecting considerable prediction errors. These results suggest that the model fails to capture complex, non-linear relationships in the data, highlighting the need for more advanced models like Random Forest.

➤ Random Forest Regression (Code)

```
# Initialize and train Random Forest Regressor
rf = RandomForestRegressor(n_estimators=200, max_depth=10, random_state=42)
rf.fit(X_normalized, y)
# Predict on inference data
y_pred_rf = rf.predict(X_new_normalized)
inference_data["Predicted_State_solar_RF"] = y_pred_rf
# Evaluate on training data
y_train_pred_rf = rf.predict(X_normalized)
# Evaluation metrics
r2_rf = r2_score(y, y_train_pred_rf)
mae_rf = mean_absolute_error(y, y_train_pred_rf)
mse_rf = mean_squared_error(y, y_train_pred_rf)
print(f"Random Forest R2: {r2_rf:.2f}")
print(f"Mean Absolute Error (MAE): {mae_rf:.2f}")
print(f"Mean Squared Error (MSE): {mse_rf:.2f}")
```

Model Evaluation Summary: The Random Forest model achieved an R^2 score of 0.80, MAE of 5.97 and an MSE of 111.11 significantly outperforming Linear Regression ($R^2 = 0.16$) by effectively capturing complex, non-linear relationships in the data. To further improve accuracy and R^2 , Gradient Boosting was introduced, leveraging sequential learning to better capture complex patterns in solar energy prediction.

➤ Gradient Boosting Regression (Code)

```
# Initialize and train the Gradient Boosting model
gb = GradientBoostingRegressor( n_estimators=300, learning_rate=0.1, max_depth=5,
random_state=42)
gb.fit(X_normalized, y)
# Predict on the inference data
y_pred_gb = gb.predict(X_new_normalized)
inference_data["Predicted_State_solar_GB"] = y_pred_gb
# Evaluate on training data
y_train_pred_gb = gb.predict(X_normalized)
# Evaluation metrics
r2_gb = r2_score(y, y_train_pred_gb)
mae_gb = mean_absolute_error(y, y_train_pred_gb)
mse_gb = mean_squared_error(y, y_train_pred_gb)
# Print results
print(f'Gradient Boosting R2: {r2_gb:.2f}')
print(f'Mean Absolute Error (MAE): {mae_gb:.2f}')
print(f'Mean Squared Error (MSE): {mse_gb:.2f}')
```

Model Evaluation Summary: Gradient Boosting achieved the best performance with an R^2 score of 0.92, MAE of 4.46, and MSE of 45.93 demonstrating high accuracy and strong predictive capability. It significantly outperforms Linear Regression ($R^2 = 0.16$) and Random Forest ($R^2 = 0.80$) by effectively capturing complex, non-linear relationships. To further boost accuracy and efficiency, XGBoost was introduced as an advanced, optimized variant of Gradient Boosting.

➤ Extreme Gradient Boosting - XGB Regression (Code):

```
# Initialize and train the XGBoost model
xgb = XGBRegressor(n_estimators=300, learning_rate=0.1, max_depth=5,
objective='reg:squarederror', random_state=42)
xgb.fit(X_normalized, y)
# Predict on the inference data
y_pred_xgb = xgb.predict(X_new_normalized)
inference_data["Predicted_State_solar_XGB"] = y_pred_xgb
# Evaluate on training data
y_train_pred_xgb = xgb.predict(X_normalized)
# Evaluation metrics
r2_xgb = r2_score(y, y_train_pred_xgb)
mae_xgb = mean_absolute_error(y, y_train_pred_xgb)
mse_xgb = mean_squared_error(y, y_train_pred_xgb)
# Print results
print(f'XGBoost R2: {r2_xgb:.2f}')
print(f'Mean Absolute Error (MAE): {mae_xgb:.2f}')
print(f'Mean Squared Error (MSE): {mse_xgb:.2f}')
```

Model Evaluation Summary: The XGBoost model achieved strong performance on the training data with an R^2 score of 0.90, MAE of 4.60, and MSE of 53.02, demonstrating high accuracy and effective error minimization. To further improve accuracy, hyperparameter tuning was applied to both Gradient Boosting and XGBoost, optimizing key parameters for better performance and robustness.

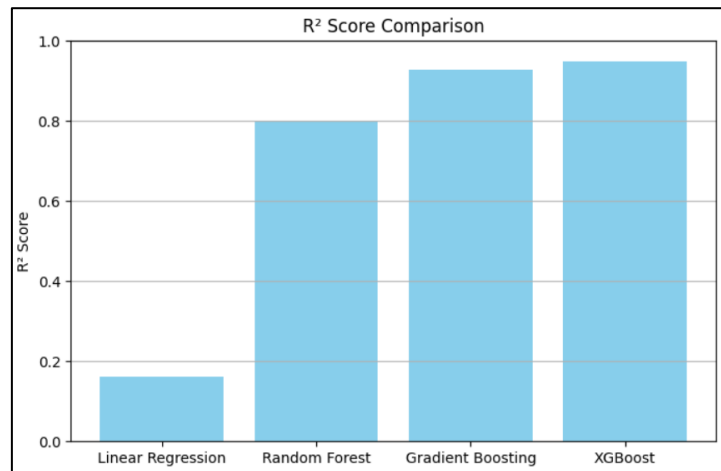


Fig 5: Graphical representation of Accuracy comparison

1.6 Hyperparameter Tuning

➤ Gradient Boosting Regression (Code)

```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
# Define hyperparameter grid for tuning
param_grid = {'n_estimators': [100, 200, 300], 'learning_rate': [0.05, 0.1, 0.2], 'max_depth': [3, 5, 7]}
# Initialize base model
gb_model = GradientBoostingRegressor(random_state=42)
# GridSearchCV for hyperparameter tuning
grid_search = GridSearchCV(estimator=gb_model, param_grid=param_grid, cv=5, scoring='r2',
n_jobs=-1, verbose=1)
# Fit the model with the best parameters
grid_search.fit(X_normalized, y)
# Extract the best estimator
best_gb = grid_search.best_estimator_
# Predict on the inference data
y_pred_gb = best_gb.predict(X_new_normalized)
inference_data["Predicted_State_solar_GB"] = y_pred_gb
# Evaluate on training data
y_train_pred_gb = best_gb.predict(X_normalized)
# Evaluation metrics
r2_gb = r2_score(y, y_train_pred_gb)
mae_gb = mean_absolute_error(y, y_train_pred_gb)
mse_gb = mean_squared_error(y, y_train_pred_gb)
print("Best Parameters:", grid_search.best_params_)
print(f"Tuned Gradient Boosting R²: {r2_gb:.2f}")
print(f"Mean Absolute Error (MAE): {mae_gb:.2f}")
print(f"Mean Squared Error (MSE): {mse_gb:.2f}")
```

Model Evaluation Summary: After hyperparameter tuning with GridSearchCV, the Gradient Boosting model's best parameters were: learning rate 0.05, max depth 7, and 200 estimators. This improved performance to an R² of 0.93, MAE of 3.93, and MSE of 39.21, demonstrating enhanced accuracy and reliability.

➤ Extreme Gradient Boosting - XGB Regression (Code):

```
from xgboost import XGBRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
# Define hyperparameter grid for tuning
param_grid_xgb = {'n_estimators': [100, 200, 300], 'learning_rate': [0.05, 0.1, 0.2], 'max_depth': [3, 5, 7]}
# Initialize base model
```

```
xgb_model = XGBRegressor(random_state=42, objective='reg:squarederror')
# GridSearchCV for hyperparameter tuning
grid_search_xgb = GridSearchCV(estimator=xgb_model, param_grid=param_grid_xgb, cv=5,
scoring='r2', n_jobs=-1, verbose=1)
# Fit the model with the best parameters
grid_search_xgb.fit(X_normalized, y)
# Extract the best estimator
best_xgb = grid_search_xgb.best_estimator_
# Predict on the inference data
y_pred_xgb = best_xgb.predict(X_new_normalized)
inference_data["Predicted_State_solar_XGB"] = y_pred_xgb
# Evaluate on training data
y_train_pred_xgb = best_xgb.predict(X_normalized)
# Evaluation metrics
r2_xgb = r2_score(y, y_train_pred_xgb)
mae_xgb = mean_absolute_error(y, y_train_pred_xgb)
mse_xgb = mean_squared_error(y, y_train_pred_xgb)
print("Best Parameters (XGBoost):", grid_search_xgb.best_params_)
print(f"Tuned XGBoost R2: {r2_xgb:.2f}")
print(f"Mean Absolute Error (MAE): {mae_xgb:.2f}")
print(f"Mean Squared Error (MSE): {mse_xgb:.2f}")
```

Model Evaluation Summary: After hyper parameter tuning with GridSearchCV, the XGBoost model's optimal parameters were learning rate 0.05, max depth 7, and 300 estimators. This boosted performance to an R^2 of 0.95, MAE of 3.27, and MSE of 28.71, demonstrating significant improvements in accuracy and reliability for solar state prediction.

Algorithms	Accuracy
Linear Regression	16%
Random Forest	80%
Gradient Boosting	92%
Extreme Gradient Boosting	90%
Hyperparameter tuning (GB)	93%
Hyperparameter tuning (XGB)	95%

Table 1: Model Accuracy Comparison

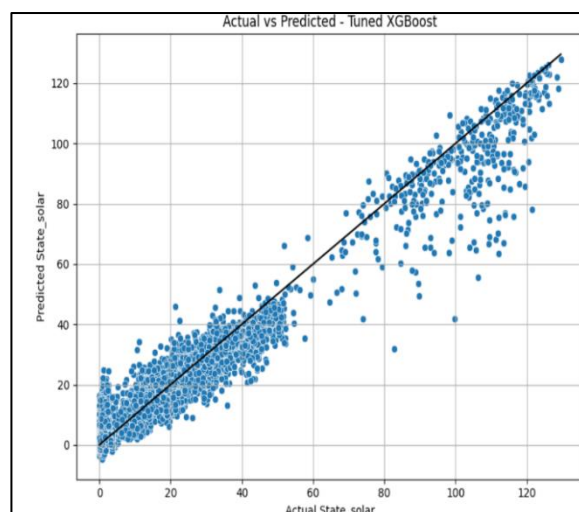


Fig 6: Actual vs Predicted Values (Tuned XGB)

In conclusion, applying hyperparameter tuning to the XGBoost model significantly enhanced its predictive accuracy and reliability, making it the most effective model for forecasting solar energy generation. Its ability to capture complex patterns with optimized parameters ensures robust and precise predictions, justifying its selection for final deployment.

V.CONCLUSION

This study presents a comprehensive approach to predicting state-wise solar power generation using advanced machine learning models. The process began with thorough **data preprocessing**, including exploratory data analysis (EDA), outlier removal via the IQR method, and feature normalization using Min-Max Scaling. The dataset, composed of historical records and inference data, was carefully cleaned and prepared to ensure modeling integrity.

Multiple regression models were evaluated:

- **Linear Regression** served as a baseline with limited predictive capability ($R^2 = 0.16$), unable to capture complex, non-linear dependencies in the data.
- **Random Forest Regression** improved the performance considerably ($R^2 = 0.80$), leveraging ensemble learning for better accuracy.
- **Gradient Boosting Regression** further enhanced results ($R^2 = 0.92$), capitalizing on sequential learning to correct residual errors.
- **XGBoost Regression**, an optimized form of Gradient Boosting, initially achieved excellent results ($R^2 = 0.90$) and was selected for further tuning.

To maximize model performance, **hyperparameter tuning using GridSearchCV** was applied to both Gradient Boosting and XGBoost models. Post tuning:

- **Gradient Boosting** improved to an R^2 of 0.93, MAE of 3.93, and MSE of 39.21.
- **XGBoost** outperformed all models, achieving an **R^2 of 0.95, MAE of 3.27, and MSE of 28.71**, indicating superior predictive accuracy and generalization.

Final Verdict:

The **XGBoost model, optimized via hyperparameter tuning**, delivered the most accurate and reliable predictions for solar energy generation across states. Its ability to handle non-linear relationships, minimize errors, and adapt to the complexity of environmental features like temperature, humidity, and sun duration makes it the **best-suited model for deployment** in real-world solar forecasting systems.

This conclusion underscores the **importance of model selection, parameter tuning, and feature engineering** in building robust machine learning solutions for energy prediction.

Data Availability Statement

The dataset used in this study was provided by the collaborating organization and is not publicly available due to confidentiality restrictions. Data may be made available by the corresponding author upon reasonable request and subject to organizational approval.

Conflict of Interest

The authors declare no conflict of interest.

REFERENCES

- [1]. Ahmed, R., Sreeram, V., Mishra, Y., & Arif, M. D. (2020). A review and evaluation of the state-of-the-art in PV solar power forecasting. *Renewable and Sustainable Energy Reviews*, 124, 109792. <https://doi.org/10.1016/j.rser.2020.109792>
- [2]. Antonanzas, J., Osorio, N., Escobar, R., Urraca, R., Martinez-de-Pison, F. J., & Antonanzas-Torres, F. (2016). Review of photovoltaic power forecasting. *Solar Energy*, 136, 78–111. <https://doi.org/10.1016/j.solener.2016.06.069>
- [3]. Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. <https://doi.org/10.1023/A:1010933404324>
- [4]. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). Association for Computing Machinery. <https://doi.org/10.1145/2939672.2939785>
- [5]. Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5), 1189–1232. <https://doi.org/10.1214/AOS/1013203451>
- [6]. Voyant, C., Notton, G., Kalogirou, S., Nivet, M. L., Paoli, C., Motte, F., & Foulloy, A. (2017). Machine learning methods for solar radiation forecasting: A review. *Renewable Energy*, 105, 569–582. <https://doi.org/10.1016/j.renene.2016.12.095>
- [7]. Wang, F., Mi, Z., Su, S., & Zhao, H. (2018). Short-term solar irradiance forecasting using machine learning techniques. *Energy Conversion and Management*, 164, 423–434.