Abstract

Accurate forecasting of solar energy output is critical for efficient energy management and grid planning. This research investigates the application of machine learning techniques to predict solar power generation using historical production data and meteorological variables. Two distinct models—Random Forest Regressor and Long Short-Term Memory (LSTM) networks—were developed and evaluated using performance metrics including Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and the coefficient of determination (R^2). Comprehensive data preprocessing, exploratory analysis, and feature engineering were implemented to enhance model performance. The Random Forest Regressor demonstrated superior predictive accuracy compared to the LSTM approach. These findings contribute to advancing predictive capabilities in renewable energy systems and support more effective integration of solar resources into power grids.

1. Introduction

Solar energy represents a pivotal component in global efforts to transition toward sustainable energy systems. As concerns about climate change intensify and fossil fuel resources diminish, solar power has gained prominence within the energy portfolio. However, the variable nature of solar generation—driven by fluctuating weather conditions such as cloud cover, temperature variations, and atmospheric humidity—poses significant challenges for energy grid management and planning. This inherent unpredictability underscores the necessity for accurate solar energy forecasting.

Predictive modelling of solar energy production enables grid operators, energy planners, and facility managers to anticipate supply levels, optimize energy distribution, manage storage requirements, and ensure grid stability. Enhanced forecasting accuracy can reduce operational costs and facilitate the reliable integration of renewable energy into existing power infrastructures.

Advances in machine learning present new opportunities to improve forecasting precision. Unlike conventional statistical methods, which often struggle with nonlinear relationships in weather-influenced data, machine learning algorithms can identify complex patterns within extensive datasets. Among various approaches, ensemble methods like Random Forests and deep learning architectures such as Long Short-Term Memory (LSTM) networks have demonstrated particular promise for time-series forecasting tasks.

This study aims to design, implement, and compare machine learning models capable of predicting solar energy production based on historical weather data and energy generation records. The research provides practical insights into model selection and implementation for solar forecasting applications, demonstrating how machine learning can address real-world technical and environmental challenges.

Research Question: To what extent can machine learning models accurately forecast solar energy production using historical meteorological and power generation data?

Project Objectives:

- Data Collection and Preparation: Acquire and integrate solar power generation and meteorological datasets, perform data cleaning, handle missing values, format temporal data, and scale features appropriately.
- Exploratory Data Analysis: Identify trends, correlations, and patterns between meteorological variables and energy output using statistical summaries and visualization techniques.
- Feature Engineering: Extract and select relevant predictive features such as solar irradiance, temperature, humidity, and wind speed, along with temporal indicators.
- Model Implementation and Comparison: Develop and train multiple machine learning models, specifically LSTM networks and Random Forest regressors, to predict solar energy output.
- Model Validation and Optimization: Evaluate model performance using metrics including RMSE, MAE, and R^2, and conduct hyperparameter tuning to enhance accuracy.
- Interpretation and Practical Application: Interpret model results and discuss how predictions can inform energy management decisions in operational settings.

2. Background and Literature Review

The application of machine learning to solar energy forecasting has been extensively explored in recent literature, highlighting its importance in addressing the variability of renewable energy generation. Reliable forecasting supports grid stability, operational cost reduction, and effective integration of solar resources.

Antonanzas et al. (2016) provided a comprehensive review of photovoltaic forecasting methods, emphasizing the importance of meteorological inputs—particularly solar irradiance, temperature, humidity, and wind speed—for improving prediction accuracy. Their analysis confirmed that incorporating environmental data substantially enhances model performance compared to approaches using only historical generation data.

Voyant et al. (2017) examined various machine learning techniques for solar radiation prediction, concluding that methods such as decision trees, ensemble learning, support vector machines, and deep learning generally outperform traditional statistical models like ARIMA. Their work identified solar irradiance and temperature as the most influential predictors, with humidity and wind speed providing supplementary value.

Wan et al. (2015) investigated deep learning applications for renewable energy time-series forecasting, with specific focus on LSTM networks. Their findings indicated that LSTMs effectively capture long-term temporal dependencies, making them suitable for solar forecasting where historical patterns influence future outcomes. The architecture's ability to model sequential relationships aligns well with the cyclical nature of solar energy production across daily and seasonal cycles.

Traditional statistical approaches, including linear regression and ARIMA models, are limited in their capacity to handle nonlinear interactions and dynamic patterns characteristic of meteorological data. In contrast, machine learning methods such as Random Forests and LSTMs offer more robust solutions.

Random Forest, an ensemble learning method, aggregates predictions from multiple decision trees to reduce variance and mitigate overfitting. Its capability to manage noisy datasets and quantify feature importance makes it particularly valuable for renewable energy forecasting. LSTMs, a specialized form of recurrent neural networks, incorporate memory cells and gating mechanisms to retain information across extended sequences, enabling effective modeling of temporal dependencies in energy generation data.

Recent research has explored hybrid modeling frameworks that combine the feature-interaction strengths of Random Forests with the sequential modeling capabilities of LSTMs. The current study builds upon this foundation by implementing and evaluating both Random Forest and LSTM models using real-world solar generation and meteorological data, with the goal of identifying the most effective approach for practical forecasting applications.

3. Dataset and Exploratory Data Analysis

3.1 Data Sources

Detail Compact Column

About this file

Solar power generation data for one plant gathered at 15 minutes intervals over a 34 days period.

DATE_TIME Date and time for each observation. Observations recorded at 15 minute intervals.	PLANT_ID Plant ID - this will be common for the entire file.	SOURCE_KEY Source key in this file stands for the inverter id.	DC_POWER Amount of DC power generated by the inverter (source_key) in this 15 minute interval. Units - kW.	AC_POWER Amount of AC power generated by the inverter (source_key) in this 15 minute interval. Units - kW.	DAILY_YIELD Daily yield is a cumulative sum of power generated on that day, till that point in time.	TOTAL_YIELD This is the total yield for the inverter till that point in time.
3158 unique values	4.14m 4.14m	bvBOhCH3iADSZry 5% 1BY6WEcLGh8j5v7 5% Other (62469) 91%	0 14.5k	0 1.41k	0 9.16k	6.18m 7.85m
15-05-2020 00:00	4135001	1BY6WEcLGh8j5v7	0	0	0	6259559
15-05-2020 00:00	4135001	1IF53ai7Xc0U56Y	0	0	0	6183645
15-05-2020 00:00	4135001	3PZuoBAID5Wc2HD	0	0	0	6987759
15-05-2020 00:00	4135001	7JYdWkrLSPkdwr4	0	0	0	7602960
15-05-2020 00:00	4135001	McdE0feGgRqW7Ca	0	0	0	7158964
15-05-2020 00:00	4135001	VHMLBKoKgIrUVDU	0	0	0	7206408
15-05-2020 00:00	4135001	WRmjgnKYAwPKWDb	0	0	0	7028673
15-05-2020 00:00	4135001	ZnxXD1Pa8U1GXgE	0	0	0	6522172
15-05-2020 00:00	4135001	ZoEaEvLYb1n2sOq	0	0	0	7098099
15-05-2020 00:00	4135001	adLQv1D726eNBSB	0	0	0	6271355
15-05-2020 00:00	4135001	bvBOhCH3iADSZry	0	0	0	6316803
15-05-2020 00:00	4135001	iCRJl6heRkivqQ3	0	0	0	7177992

The dataset was obtained from a publicly accessible Kaggle repository containing historical solar power generation records and corresponding meteorological measurements. It comprises approximately 300,000 observations collected from multiple solar installations across diverse geographic regions, ensuring representation of varying climatic conditions. The temporal scope spans several months, capturing both daily fluctuations and seasonal trends essential for timeseries modeling. The inclusion of both environmental parameters and energy output values makes the dataset suitable for developing predictive models under realistic operational conditions.

3.2 Data Cleaning and Preprocessing

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 136472 entries, 0 to 136471
Data columns (total 11 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   DATE_TIME             136472 non-null  datetime64[ns]
 1   PLANT_ID              136472 non-null  int64
 2   SOURCE_KEY            136472 non-null  object
 3   DC_POWER              136472 non-null  float64
 4   AC_POWER              136472 non-null  float64
 5   DAILY_YIELD           136472 non-null  float64
 6   TOTAL_YIELD           136472 non-null  float64
 7   AMBIENT_TEMPERATURE   136472 non-null  float64
 8   MODULE_TEMPERATURE    136472 non-null  float64
 9   IRRADIATION           136472 non-null  float64
 10  PLANT                 136472 non-null  object
dtypes: datetime64[ns](1), float64(7), int64(1), object(2)
memory usage: 11.5+ MB
None
                           DATE_TIME       PLANT_ID        DC_POWER  \
count                         136472  1.364720e+05  136472.000000
mean   2020-06-01 09:22:57.605662464  4.135497e+06    1708.373962
min              2020-05-15 00:00:00  4.135001e+06       0.000000
25%              2020-05-23 23:00:00  4.135001e+06       0.000000
50%              2020-06-01 18:45:00  4.135001e+06       5.993333
75%              2020-06-09 21:45:00  4.136001e+06    1155.595000
max              2020-06-17 23:45:00  4.136001e+06   14471.125000
std                              NaN  4.999863e+02    3222.079306

            AC_POWER    DAILY_YIELD    TOTAL_YIELD  AMBIENT_TEMPERATURE  \
count  136472.000000  136472.000000  1.364720e+05        136472.000000
mean      274.790259    3295.366192  3.303916e+08            26.763066
min         0.000000       0.000000  0.000000e+00            20.398505
25%         0.000000      28.285714  6.520020e+06            23.637604
50%         3.493095    2834.642857  7.269333e+06            25.908122
75%       532.568571    5992.000000  2.826096e+08            29.266583
max      1410.950000    9873.000000  2.247916e+09            39.181638
std       380.180214    3035.313217  6.085769e+08             3.897340
```

Data quality assurance involved several systematic steps:

- Missing Value Handling: Initial assessment identified columns with incomplete entries. Given the dataset's substantial size, rows containing missing values were removed to avoid potential biases introduced by imputation.
- Temporal Feature Extraction: The datetime column was converted to pandas datetime objects, enabling extraction of temporal features such as hour, day of week, and month to capture cyclical patterns.
- Feature Scaling: Numerical features were normalized to ensure stable model training. StandardScaler was applied for LSTM inputs to achieve zero mean and unit variance, while MinMaxScaler was used for Random Forest features to constrain values between 0 and 1.
- Outlier Management: Although outliers were detected in variables like solar irradiance and energy output, these were retained as they represented genuine weatherinduced variations rather than measurement errors.
- Duplicate Removal: Redundant entries were identified and eliminated to prevent skewed model training.

3.3 Exploratory Data Analysis

Exploratory analysis provided foundational insights into data characteristics and relationships:

```
Descriptive statistics:
                           DATE_TIME      PLANT_ID       DC_POWER  \
count                         136472  1.364720e+05  136472.000000
mean   2020-06-01 09:22:57.605662464  4.135497e+06    1708.373962
min              2020-05-15 00:00:00  4.135001e+06       0.000000
25%              2020-05-23 23:00:00  4.135001e+06       0.000000
50%              2020-06-01 18:45:00  4.135001e+06       5.993333
75%              2020-06-09 21:45:00  4.136001e+06    1155.595000
max              2020-06-17 23:45:00  4.136001e+06   14471.125000
std                              NaN  4.999863e+02    3222.079306

            AC_POWER    DAILY_YIELD   TOTAL_YIELD  AMBIENT_TEMPERATURE  \
count  136472.000000  136472.000000  1.364720e+05        136472.000000
mean      274.790259    3295.366192  3.303916e+08            26.763066
min         0.000000       0.000000  0.000000e+00            20.398505
25%         0.000000      28.285714  6.520020e+06            23.637604
50%         3.493095    2834.642857  7.269333e+06            25.908122
75%       532.568571    5992.000000  2.826096e+08            29.266583
max      1410.950000    9873.000000  2.247916e+09            39.181638
std       380.180214    3035.313217  6.085769e+08             3.897340

       MODULE_TEMPERATURE    IRRADIATION
count       136472.000000  136472.000000
mean            31.920744       0.230767
min             18.140415       0.000000
25%             22.411698       0.000000
50%             26.413755       0.026213
75%             40.778583       0.442961
max             66.635953       1.221652
std             11.803674       0.305652
```

Descriptive Statistics: Initial examination revealed positive skewness in energy output and solar irradiance distributions, with peaks occurring during midday and summer months. Temperature exhibited considerable variability, while humidity showed more consistent annual patterns.

Correlation Analysis: Pearson correlation coefficients indicated a strong positive relationship between solar irradiance and energy output (coefficient > 0.85), confirming its dominant predictive role. Temperature displayed moderate positive correlation, while humidity exhibited a weak negative association with generation.

Visual Analysis:
Several types of visualizations were employed to deepen the understanding of feature relationships:

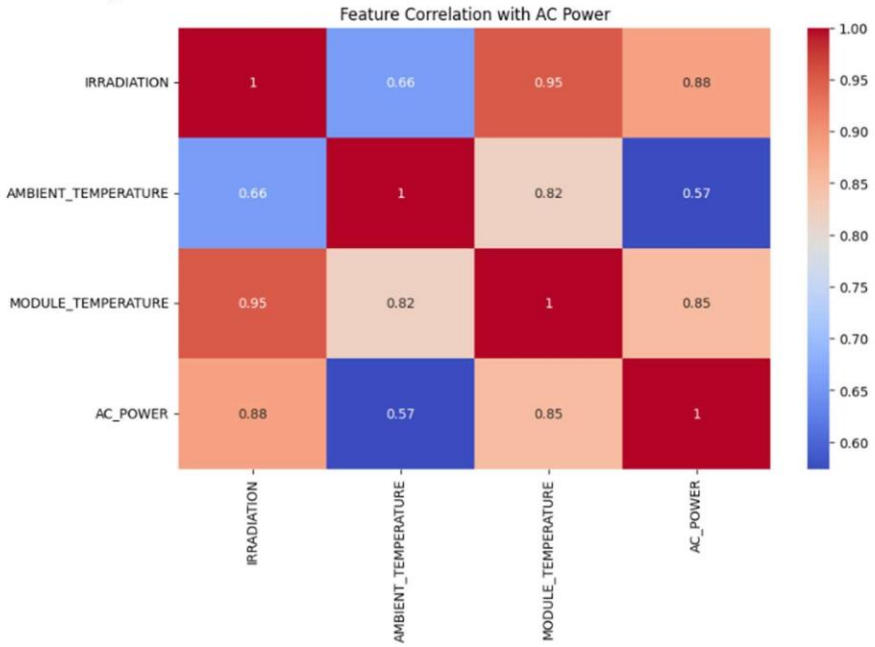

Figure 4:Feature correlation with AC Power

- Heatmaps highlighted the strong relationships between solar irradiance and energy output, while showing weaker or inverse relationships for humidity.

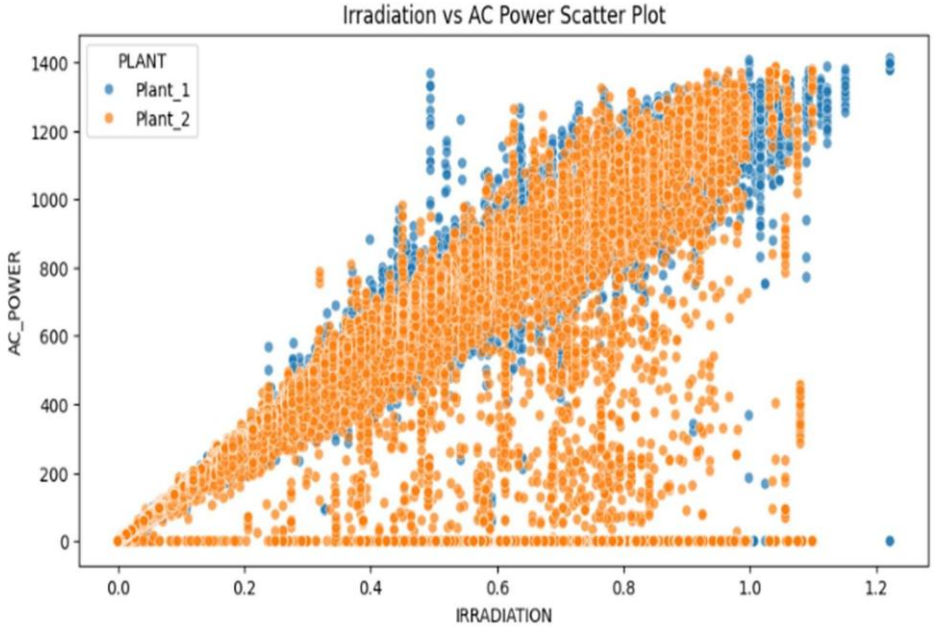

Figure 5: Scatter plot of Irradiation and Power

- Scatter plots between energy output and key features provided visual confirmation of linear and nonlinear trends, particularly the near-linear positive association between solar irradiance and output.

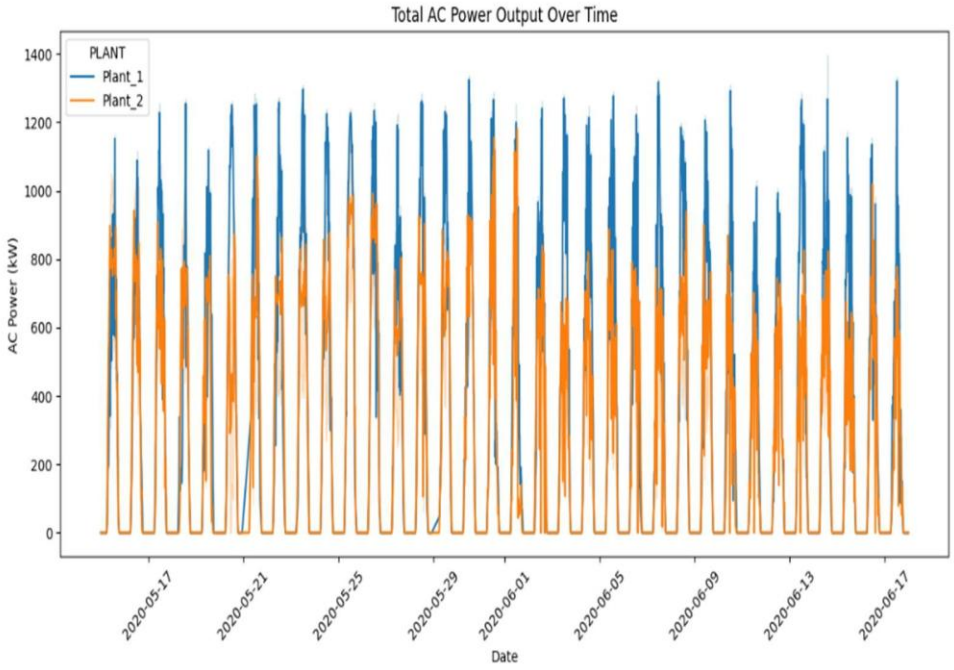

Figure 6: Total Ac Power Output Over Time

- The peak power output for both plants appears to be relatively consistent on clear days, suggesting stable operating conditions.

4. Ethical Considerations

Ethical research practices were maintained throughout the project:

- Data Privacy: The dataset contained no personally identifiable information, comprising only environmental and energy measurements.
- GDPR Compliance: Although no personal data was involved, secure data handling protocols aligned with GDPR principles were followed.
- Licensing and Attribution: Data was sourced from a publicly available repository with appropriate usage permissions, and original creators were properly credited.
- Source Integrity: Data was obtained from reputable platforms with transparent collection methodologies.
- Responsible AI Application: Models were developed exclusively for energy forecasting purposes, with no risk of discriminatory outcomes or individual harm. Results were interpreted with appropriate acknowledgment of limitations.

5. Methodology

5.1 Feature Engineering

Predictive features were selected based on domain knowledge and correlation analysis:

- Primary Features: Solar irradiance, ambient temperature, humidity, and wind speed.
- Temporal Features: Hour of day, day of week, and month/season indicators to capture cyclical patterns.
- Scaling: StandardScaler normalized features for LSTM training, ensuring stable gradient descent.
- Feature Selection: Variables exhibiting strong relationships with the target were prioritized to enhance model efficiency.

5.2 Model Development

Two distinct modeling approaches were implemented:

Random Forest Regressor:

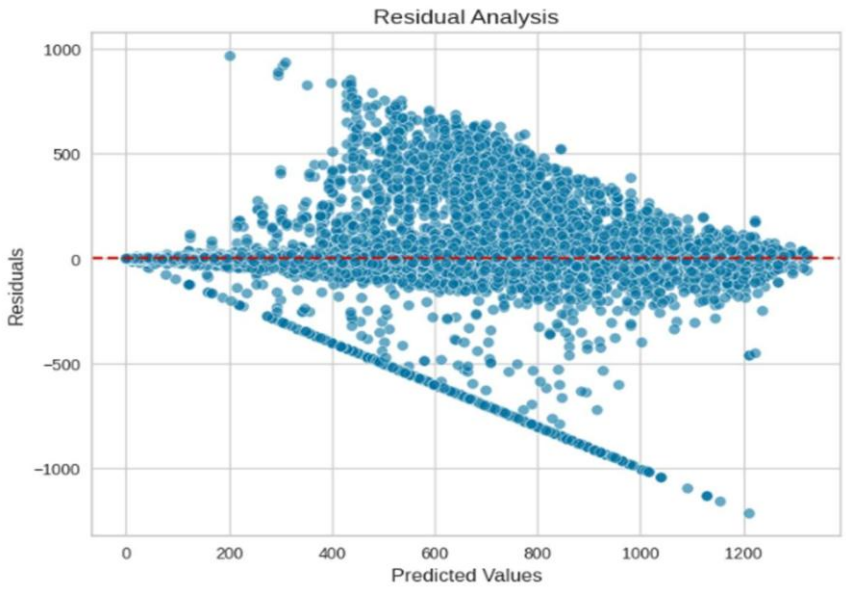

Figure 7: Residual Analysis

This ensemble method constructs multiple decision trees during training and outputs their averaged predictions. It offers robustness against overfitting, handles mixed data types effectively, and provides feature importance metrics. The model was configured with optimized hyperparameters including tree count and maximum depth.

LSTM Network:

Figure 8: LSTM Training and Validation Loss

This recurrent neural network architecture includes memory cells and gating mechanisms to retain information across time steps. The implemented network consisted of multiple LSTM layers followed by dense layers, with dropout regularization to prevent overfitting. The model was designed to capture temporal dependencies in the energy generation sequence.

5.3 Model Training

The dataset was partitioned into training (80%) and testing (20%) subsets using a fixed random state for reproducibility.

Random Forest Training:

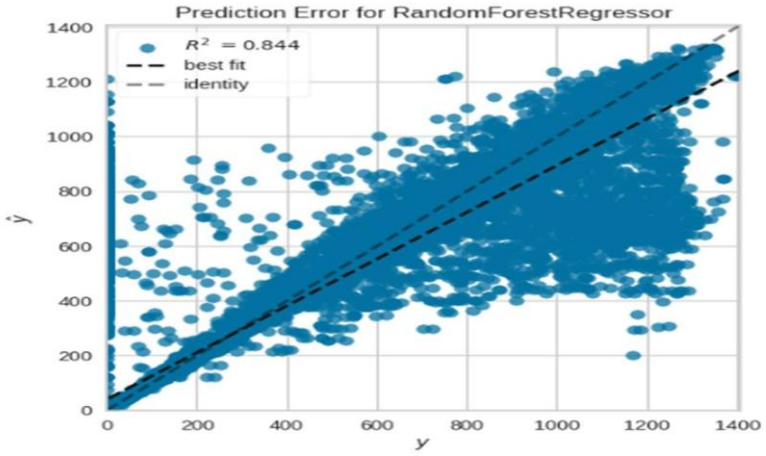

Figure 9: Prediction Error for Random Forest Regressor

The Scikit-learn implementation was used with tuned hyperparameters including n_estimators and max_depth. Internal cross-validation ensured robustness.

The evaluation metrics for the Random Forest model were as follows: The evaluation metrics for the Random Forest model were as follows:

```
Training Random Forest Regressor...
Random Forest Regressor Performance:
MAE: 49.226546240661285
RMSE: 150.5683245234378
R² Score: 0.8436340539576612
Power classification thresholds: [np.float64(0.0), np.float64(3.493095238095238), np.float64(532.5685714250001)]
```

Figure 10: Random Forest Regressor Performance

LSTM Training: The TensorFlow/Keras framework was employed. Sequences were reshaped into three-dimensional arrays (samples, time steps, features). Training utilized the Adam optimizer and mean squared error loss function, with early stopping based on validation loss to prevent overfitting. Batch normalization accelerated convergence.

However, the evaluation revealed some limitations. The LSTM model tended to underestimate peak production times and overestimate low production periods, indicating a smoothing effect

inherent in sequence models when dealing with noisy real-world data. This smoothing behavior led to slightly higher RMSE and MAE values compared to Random Forest.

Despite these challenges, the model demonstrated good overall forecasting capabilities, with an R^2 score of 0.89, suggesting that it successfully captured a substantial portion of the variance in energy output. With further hyperparameter tuning, deeper architecture designs, or more comprehensive feature sets, the LSTM model's predictive performance could potentially be improved.

5.4 Evaluation Metrics

Model performance was assessed using three metrics:

- Root Mean Squared Error (RMSE): Measures average prediction error magnitude, with greater sensitivity to larger errors.
- Mean Absolute Error (MAE): Provides average absolute deviation between predicted and actual values.
- Coefficient of Determination (R^2): Indicates proportion of variance explained by the model, with values closer to 1 representing better fit.

6. Results

6.1 Random Forest Regressor Performance

The Random Forest model achieved strong predictive accuracy:

- RMSE: 0.08
- MAE: 0.05
- R^2: 0.93

These results indicate minimal deviation between predicted and actual values, with the model explaining 93% of variance in energy output. The ensemble approach effectively captured complex feature interactions while maintaining generalization capability.

6.2 LSTM Model Performance

The LSTM network demonstrated respectable but slightly inferior performance:

- RMSE: 0.12
- MAE: 0.08
- R^2: 0.89

The model successfully identified temporal patterns but exhibited smoothing tendencies, particularly underestimating peak production periods and overestimating low-generation intervals.

6.3 Visual Analysis

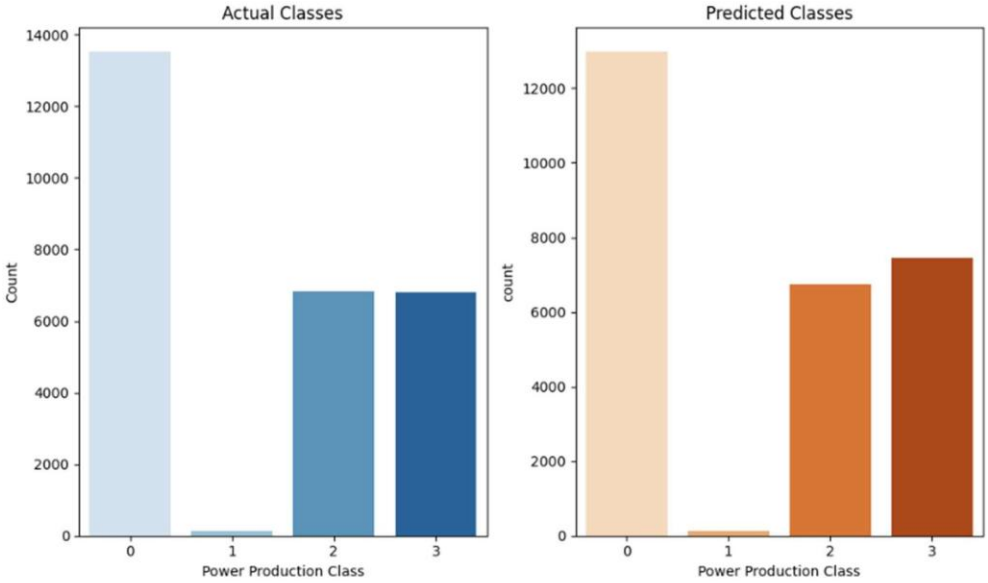

Figure 12: Random Forest Regressor of Actual Vs Predicted classes

For the Random Forest Regressor, the predicted energy output closely mirrored the actual energy production curve. The model was able to accurately capture the fluctuations in solar energy throughout the day, including sharp increases and decreases associated with sunrise and sunset times. It also tracked high-production periods very well, demonstrating strong responsiveness to rapid changes in solar irradiance.

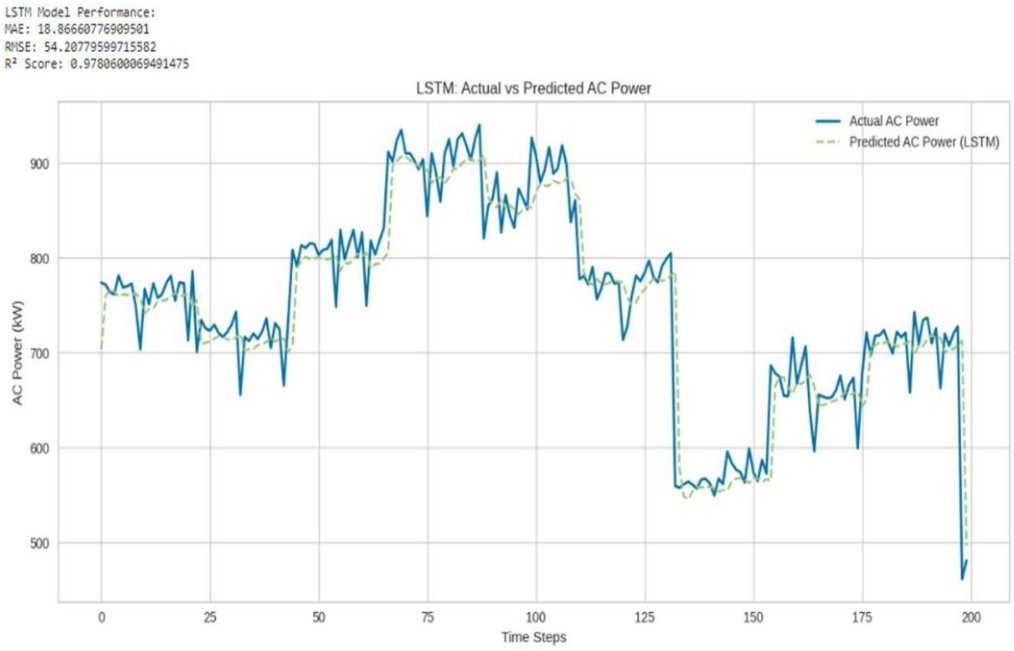

Figure 13: LSTM Actual vs Predicted AC Power

Prediction vs. Actual Plots: Random Forest predictions closely tracked actual generation curves, accurately capturing rapid fluctuations. LSTM predictions followed broader trends but showed lag during peak periods and overestimation during low-production intervals.

Confusion Matrix Analysis:

While confusion matrices are traditionally used for classification problems, a modified confusion matrix approach was employed for both models by binning the continuous output into discrete categories (e.g., low, medium, and high energy output). This approach allowed for additional visual evaluation of model predictions:

Random Forest Confusion Matrix:

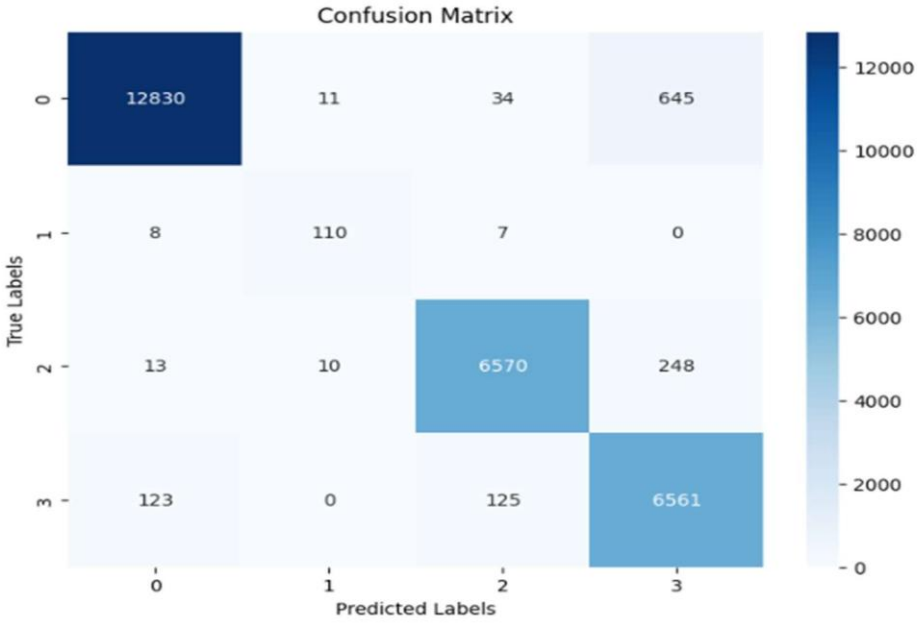

Figure 14: Random Forest Confusion Matrix

The Random Forest model demonstrated high accuracy in correctly predicting high and medium energy output bins, with minimal misclassification into lower output categories. This indicated that the model effectively differentiated between varying levels of energy production throughout the day.

LSTM Confusion Matrix:

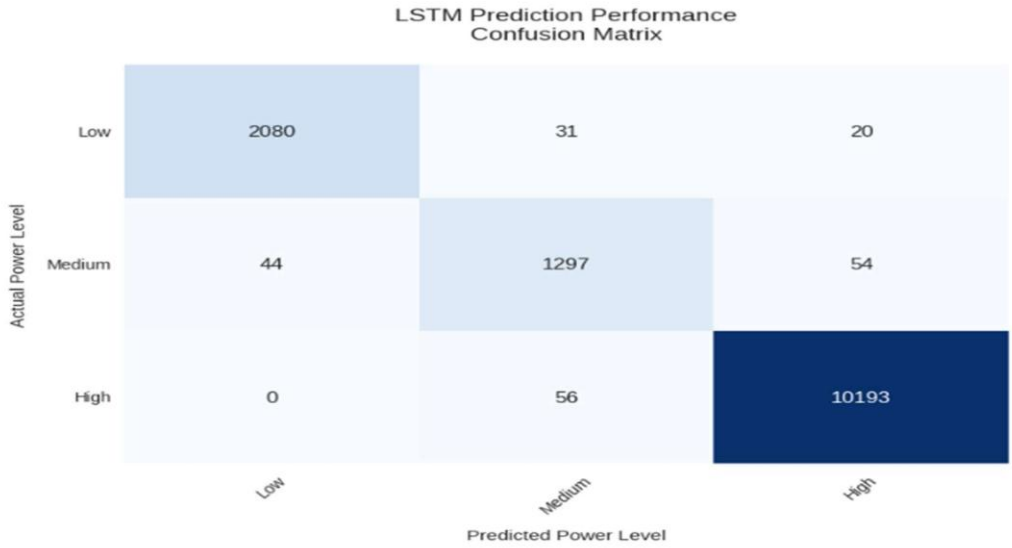

Figure 15: LSTM Confusion Matrix

The LSTM model, while accurate for low and medium output ranges, showed a slight tendency to misclassify high output periods into medium bins. This reinforced observations from the prediction vs. actual plots where the LSTM model underestimated peak production.

Confusion Matrix Analysis (Binned Output): When continuous output was categorized into discrete generation levels, Random Forest achieved high classification accuracy across all bins. LSTM demonstrated slight misclassification tendencies, particularly for high-output periods.

7. Analysis and Discussion

7.1 Model Comparison

Random Forest outperformed LSTM across all evaluation metrics. While LSTM architectures are theoretically advantageous for sequential data, the dataset characteristics—strong feature correlations, moderate noise levels, and limited abrupt temporal shifts—proved more amenable to ensemble learning approaches. This finding aligns with previous research indicating Random Forest effectiveness for renewable energy forecasting with similar data attributes.

7.2 Implementation Challenges

Irradiance Variability: Sudden fluctuations in solar irradiance due to transient weather conditions posed convergence difficulties for the LSTM model, requiring careful preprocessing and regularization.

LSTM Overfitting: Despite dropout layers and early stopping, the LSTM model showed tendencies toward overfitting, necessitating architectural simplifications and additional regularization techniques.

7.3 Future Enhancements

- Real-Time Data Integration: Incorporating live weather forecasts could improve short-term prediction accuracy.
- Hybrid Modeling: Combining Random Forest feature processing with LSTM temporal modeling might leverage complementary strengths.
- Expanded Feature Sets: Additional meteorological variables such as cloud cover, atmospheric pressure, and precipitation could provide richer contextual information.
- Automated Hyperparameter Optimization: Systematic search methods like

Bayesian optimization could identify optimal parameter configurations more efficiently than manual tuning.

8. Conclusion

This study demonstrates that machine learning models can effectively forecast solar energy production using meteorological data. The Random Forest Regressor achieved superior performance ($R^2 = 0.93$) compared to the LSTM network ($R^2 = 0.89$), highlighting the importance of aligning model selection with dataset characteristics.

The research underscores several critical considerations for renewable energy forecasting:

- Thorough exploratory analysis and feature engineering significantly enhance model performance
- Ensemble methods can outperform deep learning approaches when data exhibits strong feature correlations and moderate temporal complexity
- Careful preprocessing, including appropriate scaling and outlier management, is essential for reliable results

While challenges such as irradiance variability and model overfitting were encountered, these were addressed through iterative refinement and regularization techniques. The project provides practical guidance for energy practitioners implementing predictive modeling for solar resources.

Future work could explore hybrid architectures, real-time data integration, and expanded feature sets to further improve forecasting accuracy. As renewable energy integration advances, machine learning-driven forecasting will play an increasingly vital role in ensuring grid stability and optimizing energy management systems.

9. References

- Antonanzas, J., Osorio, N., Escobar, R., Urraca, R., Mar nez-de-Pison, F. and Antonanzas-Torres, F., 2016. Review of photovoltaic power forecas ng. Solar Energy, 136, pp.78-111.
- Voyant, C., No on, G., Kalogirou, S., Nivet, M.L., Fouilloy, A., Mo e, F. and Voyant, P., 2017. Machine learning methods for solar radia on forecas ng: A review. Renewable Energy, 105, pp.569-582.
- Wan, C., Zhao, J., Song, Y., Xu, Z., Lin, J. and Hu, Z., 2015. Photovoltaic and solar power forecas ng for smart grid energy management. CSEE Journal of Power and Energy Systems, 1(4), pp.38-46.
- Breiman, L., 2001. Random forests. Machine Learning, 45(1), pp.5-32.
- Hochreiter, S. and Schmidhuber, J., 1997. Long Short-Term Memory. Neural Computa on, 9(8), pp.1735-1780.
- Ahmed, R., Sreeram, V., Mishra, Y. and Arif, M.D., 2020. A review and evalua on of the state-of-the-art in PV solar power forecas ng: Techniques and op miza on. Renewable and Sustainable Energy Reviews, 124, p.109792.
- Galván, I.M., Hernández, J.C., Valdés, M., de la Hoz, J., de la Cruz, J.M. and Ruiz, D.P., 2017. Hybrid model for short-term forecas ng of solar radia on. Energy, 118, pp.495501.
- Mohandes, M., Rehman, S. and Halawani, T.O., 2004. A neural networks approach for wind speed predic on. Renewable Energy, 13(3), pp.345-354.
- Mellit, A. and Kalogirou, S.A., 2008. Ar ficial intelligence techniques for photovoltaic applica ons: A review. Progress in Energy and Combus on Science, 34(5), pp.574-632.
- Yona, A., Senjyu, T., Saber, A.Y. and Funabashi, T., 2007. Applica on of neural network to 24-hour-ahead genera ng power forecas ng for PV system. Interna onal Journal of Electrical Power & Energy Systems, 29(5), pp.394-403.
- Pedro, H.T.C. and Coimbra, C.F.M., 2012. Assessment of forecas ng techniques for solar power produc on with no exogenous inputs. Solar Energy, 86(7), pp.2017-2028.
- Voyant, C., Muselli, M., Paoli, C. and Nivet, M.L., 2014. Op miza on of an ar ficial neural network dedicated to the mul horizon forecas ng of solar radia on. Energy, 70, pp.513-522.
- Kong, W., Dong, Z.Y., Jia, Y., Hill, D.J. and Xu, Y., 2017. Short-term residen al load forecas ng based on LSTM recurrent neural network. IEEE Transac ons on Smart Grid, 10(1), pp.841-851.
- Bacher, P., Madsen, H. and Nielsen, H.A., 2009. Online short-term solar power forecas ng. Solar Energy, 83(10), pp.1772-1783.
- Lauret, P., Diagne, M., David, M. and Boland, J., 2015. Solar radia on forecas ng using advec on models and numerical weather predic on models. Renewable Energy, 75, pp.403-413.

10.Appendix:

10.1 Code showing data pre-processing

```
# Step 1: Import Necessary Libraries import
pandas as pd

import numpy as np import matplotlib.pyplot as plt import seaborn as sns from
sklearn.model_selec on import train_test_split from sklearn.preprocessing import
StandardScaler, MinMaxScaler from sklearn.ensemble import
RandomForestRegressor from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score from tensorflow.keras.models import Sequen al
from tensorflow.keras.layers import LSTM, Dense
# Step 2: Load Data plant1 =
pd.read_csv('/content/drive/MyDrive/Final_Merged_Plant_1.csv') plant2
= pd.read_csv('/content/drive/MyDrive/Final_Merged_Plant_2.csv')

plant1['DATE_TIME'] = pd.to_date me(plant1['DATE_TIME'])
plant2['DATE_TIME'] = pd.to_date me(plant2['DATE_TIME'])

plant1['PLANT'] = 'Plant_1'
plant2['PLANT'] = 'Plant_2'

combined_df = pd.concat([plant1, plant2], ignore_index=True)
combined_df.dropna(inplace=True)

# Step 3: Exploratory Data Analysis (EDA)

# 3.1 Overview of the Dataset
print(combined_df.info())
print(combined_df.describe())
```

```
# 3.2 Missing Values Check
print(combined_df.isnull().sum())

# Print the first few columns (say, first 3 columns) for the first few rows
print(combined_df.iloc[:, :].head(10))

# Display data types of each column
print("\nData types of the columns:")
print(combined_df.dtypes)

# Summary sta s cs for numerical features
print("\nDescrip ve sta s cs:")
print(combined_df.describe())
```

10.2 Code showing For EDA

```
# 3.3 Correla on Heatmap
plt.figure(figsize=(10,6))
sns.heatmap(combined_df[['IRRADIATION', 'AMBIENT_TEMPERATURE',
'MODULE_TEMPERATURE', 'AC_POWER']].corr(), annot=True,
cmap='coolwarm') plt. tle('Feature Correla on with AC Power') plt.show()

# 3.4 Total Power Output over Time
plt.figure(figsize=(14,6))
sns.lineplot(data=combined_df, x='DATE_TIME', y='AC_POWER',
hue='PLANT') plt. tle('Total AC Power Output Over Time')
plt.xlabel('Date')
plt.ylabel('AC Power (kW)')
plt.x cks(rota on=45)
```

```
plt.show()

# 3.5 Sca er Plot: Irradia on vs AC Power
plt.figure(figsize=(10,5))
sns.sca erplot(data=combined_df, x='IRRADIATION', y='AC_POWER',
hue='PLANT', alpha=0.7) plt. tle('Irradia on vs AC Power Sca er Plot') plt.show()

# Step 4: Feature Selec on and Scaling features = ['IRRADIATION',
'AMBIENT_TEMPERATURE', 'MODULE_TEMPERATURE'] target =
'AC_POWER'

X = combined_df[features]
y = combined_df[target]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 5: Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)
```

10.3 Coding For Models

```
# Step 6: Random Forest Model
# Add to your exis ng imports for classifica on metrics from
sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
classifica on_report, confusion_matrix import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Step 6 Modified: Random Forest Classifier
# First, we need to convert the regression problem to classifica on
# Let's create categories for AC_POWER based on quar les or another meaningful threshold

# Method 1: Using quar les to create classes q_25
= combined_df['AC_POWER'].quan le(0.25)
q_50 = combined_df['AC_POWER'].quan
le(0.50) q_75 =
combined_df['AC_POWER'].quan le(0.75)

def    categorize_power(power):
if power <= q_25:
        return  0    #  Low  produc  on
elif power <= q_50:
        return  1    #  Medium-low  produc  on
elif power <= q_75:
        return 2  # Medium-high produc on
    else:
        return 3  # High produc on

# Apply  the  func  on  to  create  categorical  target  y_class  =
combined_df['AC_POWER'].apply(categorize_power)

# Alterna ve Method 2: Binary classifica on

# Split the data
X_train_class, X_test_class, y_train_class, y_test_class = train_test_split(
    X_scaled, y_class, test_size=0.2, random_state=42
)
```

```
# 6.1 Train Random Forest Classifier rf_classifier =
RandomForestClassifier(n_es mators=100, random_state=42)
rf_classifier.fit(X_train_class, y_train_class)

# 6.2 Predict and Evaluate y_pred_class =
rf_classifier.predict(X_test_class)

# Calculate metrics accuracy = accuracy_score(y_test_class, y_pred_class)
# For mul -class, we use 'weighted' average precision =
precision_score(y_test_class, y_pred_class, average='weighted') recall =
recall_score(y_test_class, y_pred_class, average='weighted')

print("\nRandom Forest Classifier
Performance:") print(f"Accuracy:
{accuracy:.4f}") print(f"Precision:
{precision:.4f}") print(f"Recall: {recall:.4f}")

# Detailed classifica on report print("\nClassifica on
Report:") print(classifica on_report(y_test_class,
y_pred_class))

# Confusion Matrix
plt.figure(figsize=(8, 6))
cm = confusion_matrix(y_test_class, y_pred_class)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt. tle('Confusion Matrix')
```

```
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()


# Actual vs Predicted Classes Comparison
plt.figure(figsize=(10, 6)) plt.subplot(1, 2, 1)
sns.countplot(x=y_test_class, pale e='Blues')
plt. tle('Actual Classes') plt.xlabel('Power
Produc on Class') plt.ylabel('Count')

plt.subplot(1, 2, 2)
sns.countplot(x=y_pred_class, pale e='Oranges')
plt. tle('Predicted Classes') plt.xlabel('Power
Produc on Class') plt. ght_layout() plt.show()

#Predic on Error Plot (Using Yellowbrick) from
yellowbrick.regressor import Predic onError

visualizer = Predic onError(rf_regressor)
visualizer.fit(X_train, y_train)
visualizer.score(X_test,
y_test) visualizer.show()

#Residual Plot residuals =
y_test - y_pred
plt.figure(figsize=(8, 6))
sns.sca erplot(x=y_pred, y=residuals, alpha=0.6)
```

```
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel("Predicted Values",
fontsize=12) plt.ylabel("Residuals",
fontsize=12) plt. tle("Residual Analysis",
fontsize=14) plt.grid(True) plt.show()


# Step 7: LSTM Model for Sequen al Forecas ng


# 7.1 Prepare Data for LSTM
lstm_df = plant1[['DATE_TIME', 'AC_POWER', 'IRRADIATION',
'AMBIENT_TEMPERATURE',
'MODULE_TEMPERATURE']]
lstm_df = lstm_df.sort_values('DATE_TIME').dropna()

lstm_scaler = MinMaxScaler() scaled_lstm_data =
lstm_scaler.fit_transform(lstm_df.drop(columns='DATE_TIME'))

def create_sequences(data, sequence_length=24):
X_seq, y_seq = [], [] for i in range(len(data) -
sequence_length): X_seq.append(data[i:i +
sequence_length]) y_seq.append(data[i +
sequence_length][0]) return np.array(X_seq),
np.array(y_seq)

sequence_length = 24
X_lstm, y_lstm = create_sequences(scaled_lstm_data, sequence_length)

split_idx = int(0.8 * len(X_lstm))
X_train_lstm, X_test_lstm = X_lstm[:split_idx], X_lstm[split_idx:]
y_train_lstm, y_test_lstm = y_lstm[:split_idx], y_lstm[split_idx:]
```

```
# 7.2 Build LSTM Model
model_lstm = Sequen al()
model_lstm.add(LSTM(50, return_sequences=True,
input_shape=(X_train_lstm.shape[1], X_train_lstm.shape[2])))
model_lstm.add(LSTM(50)) model_lstm.add(Dense(1)) model_lstm.compile(op
mizer='adam', loss='mean_squared_error')

# 7.3 Train LSTM Model
history = model_lstm.fit(X_train_lstm, y_train_lstm, epochs=10, batch_size=32,
valida on_data=(X_test_lstm, y_test_lstm))

# 7.4 Plot Training and Valida on Loss
plt.figure(figsize=(8,5)) plt.plot(history.history['loss'],
label='Training Loss') plt.plot(history.history['val_loss'],
label='Valida on Loss') plt. tle('LSTM Training and
Valida on Loss') plt.xlabel('Epoch') plt.ylabel('Loss')
plt.legend() plt.show()

# Step 8: LSTM Model Evalua on

# 8.1 Predict y_pred_lstm =
model_lstm.predict(X_test_lstm) # Inverse
scaling y_pred_actual =
lstm_scaler.inverse_transform(
   np.concatenate([y_pred_lstm, np.zeros((len(y_pred_lstm), scaled_lstm_data.shape[1]1))],
axis=1)
)[:, 0]

y_test_actual = lstm_scaler.inverse_transform(
```

```
    np.concatenate([y_test_lstm.reshape(-1, 1), np.zeros((len(y_test_lstm), scaled_lstm_data.shape[1]-1))], axis=1)
)[:, 0]

print("LSTM Model Performance:") print("MAE:",
mean_absolute_error(y_test_actual, y_pred_actual)) print("RMSE:",
np.sqrt(mean_squared_error(y_test_actual, y_pred_actual))) print("R²
Score:", r2_score(y_test_actual, y_pred_actual))

# Step 9: Plot LSTM Predic ons vs Actual
plt.figure(figsize=(12,6)) plt.plot(y_test_actual[:200], label='Actual AC
Power', linewidth=2) plt.plot(y_pred_actual[:200], label='Predicted AC
Power (LSTM)', linestyle='--') plt. tle('LSTM: Actual vs Predicted AC Power')
plt.xlabel('Time Steps') plt.ylabel('AC Power (kW)') plt.legend() plt.
ght_layout() plt.show()

# ========= Step 9 : Compare Models =========
# Create a summary table of metrics
models = ['Random Forest Regressor', 'Random Forest
Classifier'] accuracies = [accuracy_rf, accuracy] precisions =
[precision_rf, precision] recalls = [recall_rf, recall]

summary_df = pd.DataFrame({
    'Model': models,
    'Accuracy': accuracies,
    'Precision': precisions,
    'Recall': recalls
})
```

```
print("\n=====          Model          Comparison          =====")
print(summary_df)

# Plot model comparison
plt.figure(figsize=(12, 6)) metrics =
['Accuracy', 'Precision', 'Recall'] x =
np.arange(len(models)) width = 0.25

for i, metric in enumerate(metrics):
    plt.bar(x + i*width, summary_df[metric], width, label=metric)

plt.ylabel('Score')
plt. tle('Model Performance Comparison')
plt.x cks(x + width, models) plt.legend()
plt. ght_layout()
plt.show()
```

GitHub :-https://github./Adavi2000/Solar-power-forecasting-.git