Methodology

This section describes the technical workflow adopted for data processing, model development, training, and evaluation in the project.

Data Preprocessing

Effective data preprocessing is crucial for building accurate and reliable machine learning models. The raw datasets sourced from Kaggle contained operational energy generation metrics and corresponding weather variables. Without careful preprocessing, any noise or inconsistencies in this data could have negatively impacted model accuracy.

- Data Cleaning: The initial cleaning process involved scanning for and removing duplicate records to maintain the integrity of the time series. Subsequently, missing values were examined. For weather variables, gaps were filled using interpolation techniques that considered the temporal proximity of surrounding values, while forward-filling was applied to missing energy load or generation values to preserve historical consistency. Special care was taken not to artificially smooth critical events such as sudden spikes or dips that represent real-world phenomena.
- Handling Outliers: Outliers detected during exploratory data analysis were analyzed for relevance. Outliers linked to extreme weather conditions or grid instabilities were retained because they offered valuable insights into peak load and renewable generation behavior. Random or erroneous entries, if any, were removed cautiously.
- Feature Engineering: Lag features were created to introduce temporal dependencies into the models. For example, solar generation and wind speed values lagged by 1, 3, 6, and 24 hours were added as additional features, allowing the models to learn short- and medium-term patterns. Rolling window statistics such as moving averages over 6-hour and 12-hour intervals were also computed to capture smoothing trends.

  Time-based cyclical features such as "Hour of Day" and "Day of Week" were transformed using sine and cosine encoding, ensuring that the periodicity of these variables was mathematically preserved. Additionally, "Season" was encoded as a categorical feature based on the month extracted from the timestamp.
- Normalization: Continuous features such as temperature, wind speed, and humidity were scaled using Min-Max normalization to fit within the [0, 1] range. This scaling was crucial for deep learning models like LSTM, which are sensitive to feature magnitudes.
- Data Splitting: Once preprocessing was completed, the dataset was chronologically split into training, validation, and test subsets. The chronological order was maintained strictly to prevent any leakage of future information into the past.

  These preprocessing steps ensured that the dataset was not only clean but also rich in features that could help machine learning models learn complex temporal dependencies and deliver more accurate renewable energy forecasts.

Model Selection

The selection of appropriate machine learning models is critical to the success of any predictive project. In this study, three models — Random Forest, XGBoost, and LSTM — were selected based on their complementary strengths in handling structured, time-series, and complex datasets.

- **Random Forest:** Random Forest is an ensemble learning technique that combines multiple decision trees to generate a more robust and generalized model. Each decision tree is trained on a bootstrap sample of the data, and the final prediction is obtained by averaging the outputs (for regression tasks). Random Forest models are highly effective at capturing non-linear feature interactions without requiring extensive feature scaling or transformation. Their robustness against overfitting, especially when the number of trees is high, makes them suitable for noisy real-world energy datasets.
- **XGBoost:** XGBoost (Extreme Gradient Boosting) is an optimized gradient boosting framework that builds models sequentially by minimizing a loss function. Each new model corrects errors made by previous models, leading to a highly accurate and resilient predictive system. XGBoost introduces regularization (both L1 and L2) which helps prevent overfitting, a common issue in ensemble methods. Furthermore, its ability to handle missing values internally and prioritize important features automatically made it a strong candidate for this project.
- **LSTM (Long Short-Term Memory Networks):** LSTM networks are a special kind of Recurrent Neural Network (RNN) capable of learning long-term dependencies. They address the vanishing gradient problem associated with traditional RNNs, enabling the network to remember information over longer sequences. Given the sequential and temporally dependent nature of energy production and weather datasets, LSTM was ideal for capturing patterns over hours, days, and even seasons. Its architecture includes memory cells and gating mechanisms (input, forget, and output gates) that regulate information flow, allowing the model to retain and discard information as needed.

The rationale for choosing this combination was to allow a comparison between traditional ensemble tree methods (Random Forest and XGBoost) and advanced deep learning techniques (LSTM). This enabled the study to evaluate whether the additional complexity of deep learning models provided significant predictive advantages over more interpretable ensemble methods.

Training and Validation

Proper training and validation strategies are essential to ensure that the models developed can generalize well to unseen data, particularly in a time-series forecasting context.

- **Train-Test Split:** A chronological split of the data was performed to maintain the temporal order and avoid information leakage. 80% of the earliest data points were allocated for model training, and the remaining 20% were reserved for testing. This ensures that models are evaluated on future unseen data, simulating real-world forecasting conditions.

- **Cross-Validation:** For Random Forest and XGBoost models, time-series cross-validation was employed. Unlike random k-fold cross-validation, time-series cross-validation respects the order of observations, splitting the data into sequential training and validation sets. This method helps in assessing model performance over different time intervals and reduces the risk of overfitting to a particular temporal segment.
- **Hyperparameter Tuning:** Grid search and random search techniques were applied to tune hyperparameters such as the number of estimators, maximum tree depth, and learning rate for Random Forest and XGBoost models. This tuning process optimized the models for better bias-variance tradeoff.
- **Early Stopping for LSTM:** For the LSTM model, an early stopping mechanism was implemented by monitoring the validation loss during training. If the validation loss did not improve after a defined number of epochs (patience parameter), training was halted to prevent overfitting. The batch size, number of epochs, number of LSTM units, and dropout rates were tuned carefully based on performance on the validation set.
- **Data Scaling:** Before feeding the data into the LSTM model, all input features were scaled between 0 and 1 using Min-Max normalization, as neural networks are sensitive to input feature magnitudes.
- **Model Saving:** The best-performing models (based on validation performance) were saved and later loaded for final evaluation on the test set to ensure consistency and reproducibility.

Evaluation Metrics

Accurate evaluation of model performance is vital to validate predictive effectiveness and to enable fair comparison across different models. In this project, two key metrics were utilized:

- **Root Mean Squared Error (RMSE):** RMSE is a widely used measure of the differences between values predicted by a model and the values actually observed. It is calculated as the square root of the average squared differences between the predicted and actual values:

  where is the actual value, is the predicted value, and is the number of observations. A lower RMSE indicates a model that predicts closer to actual values. It penalizes larger errors more heavily due to the squaring term, making it sensitive to outliers.

- **R² Score (Coefficient of Determination):** The R² score indicates the proportion of variance in the dependent variable that is predictable from the independent variables. It is calculated as:

  where is the mean of the actual values. An R² value of 1 indicates perfect prediction, while an R² of 0 indicates that the model performs no better than simply predicting the mean.

Rationale for Metric Selection:

- RMSE was selected because it provides an absolute measure of average prediction error, which is meaningful for operational use where large errors can be costly.

- R^2 was selected because it offers an intuitive understanding of how much of the variability in renewable energy generation is explained by the model.

  By using both metrics together, a comprehensive evaluation of model accuracy (RMSE) and explanatory power (R^2) was achieved, offering deeper insights into the performance of Random Forest, XGBoost, and LSTM models.

  Tools and Development Environment

  The entire project was developed and executed using a cloud-based environment that ensured scalability, easy collaboration, and efficient computational management. The core tools and platforms used include:

- **Programming Language: Python**
  Python was selected due to its extensive ecosystem of libraries, flexibility, active developer community, and suitability for machine learning and deep learning projects.
- **Development Environment: Google Colaboratory (Colab)**
  Google Colab, a free cloud-based environment hosted by Google, was used for coding, model training, and evaluation. It provided access to powerful GPU resources, reduced local hardware dependency, and facilitated easy sharing of notebooks with peers and supervisors. The seamless integration with Google Drive also enabled secure and automatic saving of project files.
- **Key Libraries:**
  - **Pandas:** For data loading, cleaning, and manipulation operations, especially when merging energy and weather datasets.
  - **NumPy:** For efficient mathematical and matrix operations.
  - **Scikit-learn:** For implementing Random Forest models, performing preprocessing, cross-validation, and evaluation metrics computation.
  - **XGBoost:** Direct library usage for advanced gradient boosting techniques, including feature importance visualization and regularization options.
  - **TensorFlow/Keras:** For building, training, and validating the Long Short-Term Memory (LSTM) deep learning model, leveraging GPU acceleration available in Google Colab.
  - **Matplotlib and Seaborn:** For creating data visualizations, including exploratory plots, residual plots, and model comparison graphs.
- **Version Control:**
  GitHub was used for version control to manage different versions of scripts and notebooks systematically. The public repository ensures transparency, reproducibility, and backup.
- **Hardware Configuration:**
  Since the project was executed on Google Colab, it leveraged Google's cloud infrastructure, typically providing access to Intel Xeon CPUs, Tesla T4/K80 GPUs,

and up to 12GB RAM. This cloud-based approach enabled faster training of the LSTM model and efficient processing of the large integrated dataset.

- **Cloud Backup:**
  All datasets, intermediate results, and final project outputs were backed up securely on Google Drive, ensuring high availability and reducing the risk of accidental data loss.

  By leveraging this robust set of cloud-based tools and computational resources, the project maintained a high standard of technical rigor, reproducibility, efficiency, and scalability throughout the development lifecycle.