Weather-Based Prediction of Wind Turbine Energy Output Using Machine Learning

1. Project Description

Weather-Based Prediction of Wind Turbine Energy Output using Machine Learning is a predictive analytics approach designed to estimate wind turbine power generation based on weather parameters. By leveraging historical wind turbine data and machine learning algorithms, this project aims to accurately forecast energy output, enabling efficient renewable energy management and grid stability.

Scenario 1: Energy Production Forecasting

The system predicts wind turbine energy output based on weather-related inputs such as wind speed, wind direction, and theoretical power curve. Energy operators can use these predictions to plan electricity distribution and optimize renewable energy utilization.

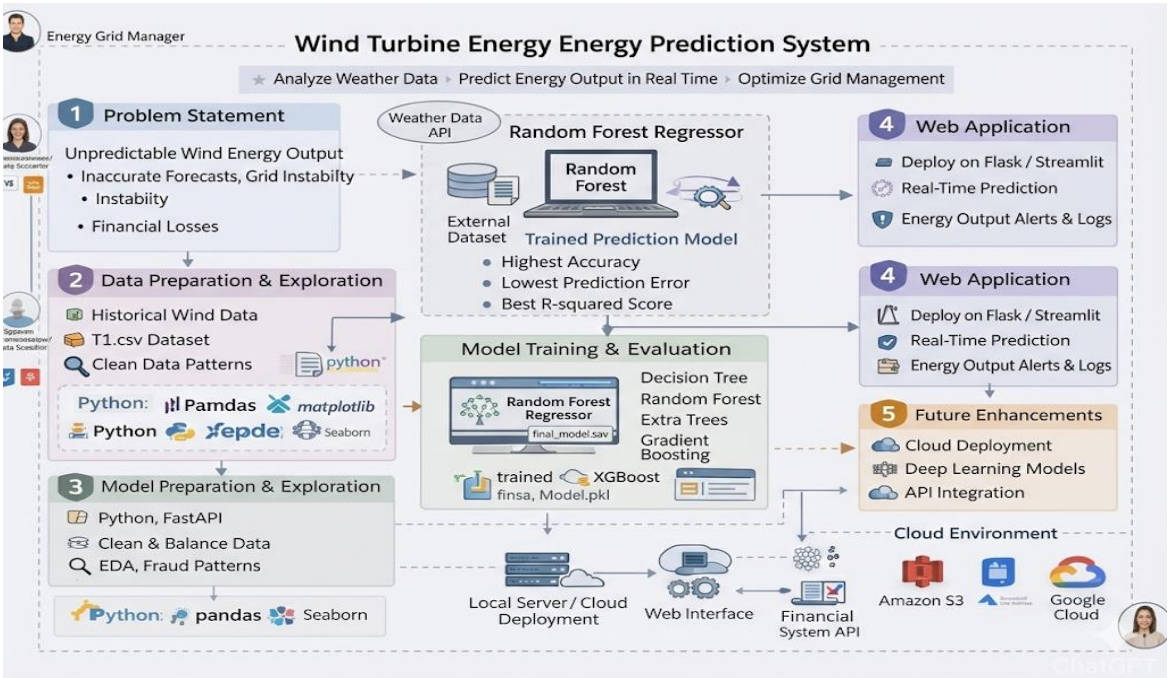
Scenario 2: Grid Stability & Load Management

Energy grid managers can use predicted energy output to balance renewable and conventional power sources. Accurate forecasting helps maintain voltage stability and prevent supply-demand imbalance.

Scenario 3: Performance Optimization

Wind farm operators can monitor predicted versus actual output to identify inefficiencies, schedule maintenance, and improve turbine performance.

Technical Architecture:



2. Prerequisites

To complete this project, the following software and Python packages are required:

Required Python Packages:

Open Anaconda Prompt and install:

- pip install numpy
- pip install pandas
- pip install scikit-learn
- pip install matplotlib
- pip install seaborn
- pip install flask
- pip install joblib

3. Project Objectives

The primary objective of this project is to design and implement a machine learning–based system capable of accurately predicting wind turbine energy output using weather-related parameters. The system aims to assist renewable energy stakeholders in making informed operational and strategic decisions by providing reliable energy forecasts.

The key objectives of this project are as follows:

1. **To Understand Renewable Energy Forecasting Challenges:** To study the variability of wind energy production and understand how weather conditions such as wind speed, wind direction, and theoretical power curves influence turbine output. This helps in identifying the importance of predictive modeling in renewable energy management.

2. **To Perform Comprehensive Data Analysis:** To collect and analyze historical wind turbine data in order to:

- Understand data distribution and patterns
- Identify correlations between input features and energy output
- Detect outliers and inconsistencies
- Perform univariate, bivariate, and descriptive analysis. This ensures that the dataset is well-understood before applying machine learning algorithms.

3. **To Implement Effective Data Preprocessing Techniques:** To clean and prepare the dataset by:

- Handling missing values (if any)
- Managing outliers appropriately
- Renaming columns for clarity
- Splitting data into training and testing sets

4. **To Develop and Compare Machine Learning Models:** To implement regression-based machine learning algorithms such as:

- Linear Regression

- Decision Tree Regressor
- Random Forest Regressor

The objective is to evaluate and compare their performance in predicting wind turbine energy output and select the most accurate and stable model.

5. **To Evaluate Model Performance Using Standard Metrics:** To assess the predictive performance of the model using evaluation metrics such as:

- R^2 Score
- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)

These metrics help measure prediction accuracy and generalization capability.

6. **To Visualize Model Results:** To generate graphical representations such as:

- Scatter plot of Actual vs Predicted Power Output
- Correlation heatmaps
- Distribution plots

Visualization helps in understanding model effectiveness and data behavior clearly.

7. **To Deploy the Model Using a Web Application**

To integrate the trained machine learning model into a Flask-based web application that allows users to:

- Enter wind parameters
- Submit data for prediction
- Receive predicted energy output in real time

This ensures practical usability of the developed system.

8. **To Build a Scalable and Extendable System:** To design the application architecture in such a way that it can be extended in the future for:

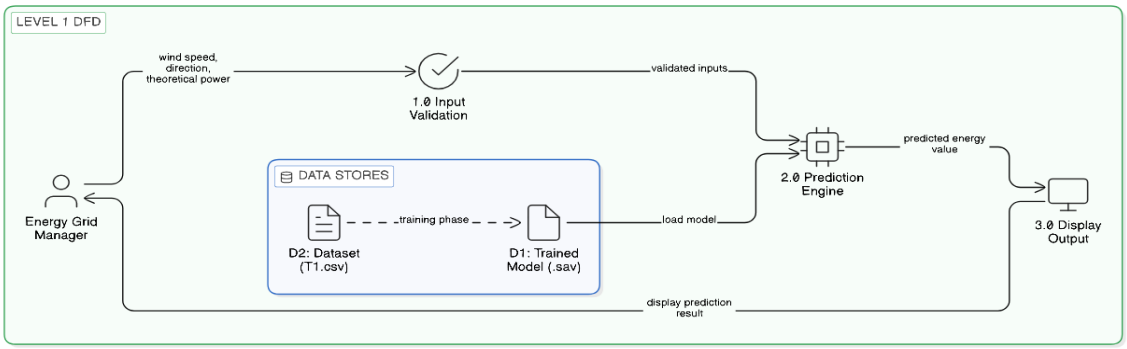
- Real-time weather API integration
- Cloud deployment
- Advanced forecasting techniques
- Dashboard-based visualization

9. **To Contribute to Sustainable Energy Management**: To demonstrate how machine learning can enhance renewable energy forecasting, improve grid stability, and support sustainable energy planning.

4. Project Flow

The project follows a structured pipeline beginning with data collection and ending with deployment of the trained machine learning model through a Flask-based web application. Each phase contributes to building an accurate and reliable wind energy prediction system.

The overall system flow is illustrated below.



- User interacts with the web interface.
- User enters wind-related input parameters.
- The backend validates and processes the input.
- The trained machine learning model generates predictions.
- The predicted wind energy output is displayed on the user interface.

4.1 Data Collection

The dataset used for this project was obtained from a publicly available dataset on Kaggle.
Dataset Name: **Wind Turbine SCADA Dataset (T1.csv)**
This dataset contains real-world wind turbine data including:

- Wind Speed (m/s)
- Wind Direction (°)
- Theoretical Power Curve (kWh)
- LV Active Power (kW)

These parameters are essential for building a regression-based energy prediction model.

	Date/Time	LV ActiveP	Wind Spee	Theoretica	Wind Direction (°)
2	01 01 2018	380.0478	5.311336	416.3289	259.9949
3	01 01 2018	453.7692	5.672167	519.9175	268.6411
4	01 01 2018	306.3766	5.216037	390.9	272.5648
5	01 01 2018	419.6459	5.659674	516.1276	271.2581
6	01 01 2018	380.6507	5.577941	491.703	265.6743
7	01 01 2018	402.392	5.604052	499.4364	264.5786
8	01 01 2018	447.6057	5.793008	557.3724	266.1636
9	01 01 2018	387.2422	5.30605	414.8982	257.9495
10	01 01 2018	463.6512	5.584629	493.6777	253.4807

4.2 Visualizing and Analyzing Data

Before performing data preprocessing and model training, exploratory data analysis (EDA) was conducted to understand the dataset characteristics.
Required Python libraries were imported:

- NumPy
- Pandas
- Matplotlib

- Seaborn
- Scikit-learn

Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC

```

a) Univariate Analysis

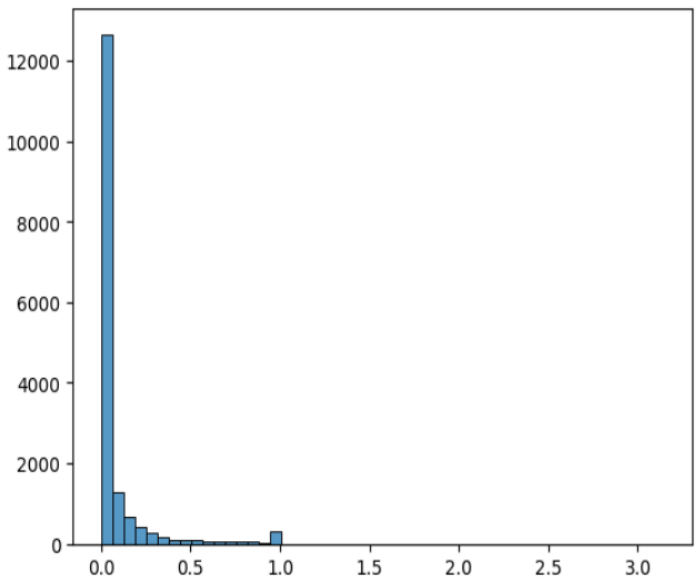
Univariate analysis involves analyzing one variable at a time to understand its distribution and statistical properties.

Histogram

Histograms were plotted for:

- Wind Speed
- LV Active Power

These plots helped in understanding distribution shape, skewness, and spread of values.

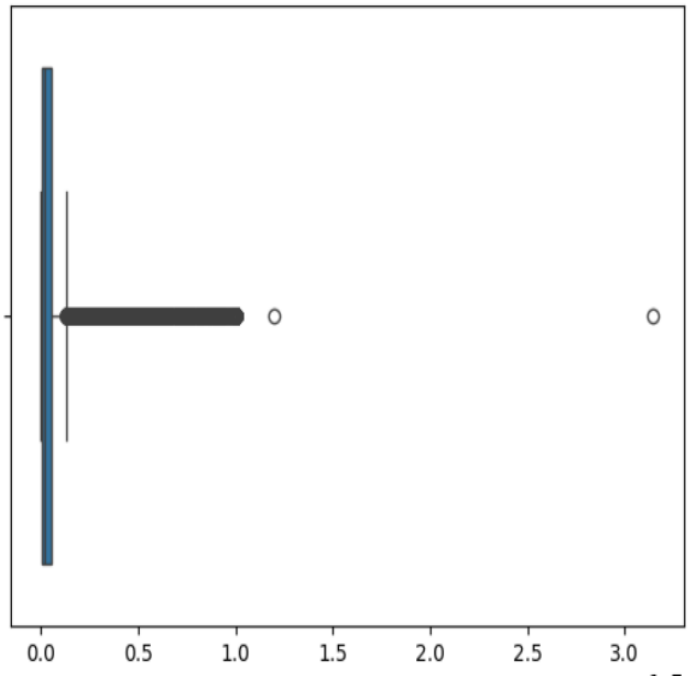


b) Boxplot

Boxplots were used to detect outliers in:

- Wind Speed
- Energy Output

Since wind energy data naturally contains extreme values, careful analysis was performed before handling outliers.



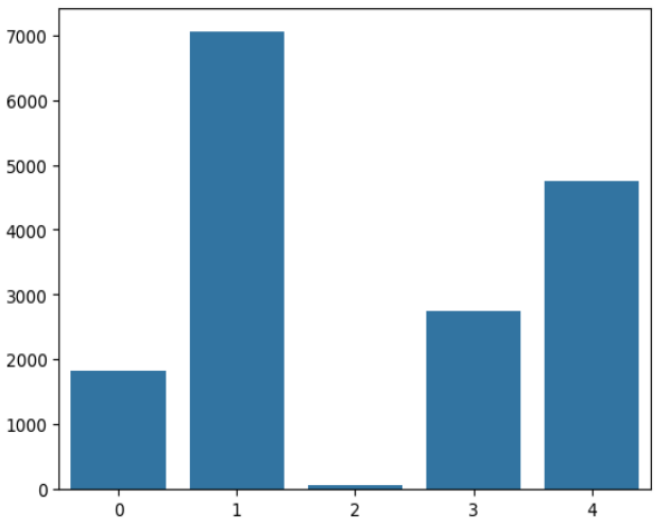
c) Bivariate Analysis

Bivariate analysis examines relationships between two variables.

Scatter Plot (Wind Speed vs Power Output)

A scatter plot was generated to visualize the relationship between wind speed and actual power output.

The plot showed a strong positive correlation, confirming wind speed as a primary predictor.



d) Descriptive Analysis

Descriptive analysis is a fundamental step in exploratory data analysis (EDA) that provides a statistical summary of the dataset. It helps in understanding the central tendency, variability, range, and overall distribution of features before applying machine learning algorithms.

In this project, descriptive analysis was performed on the wind turbine dataset to gain insights into the behavior of wind parameters and their influence on power generation.

The dataset consists of multiple numerical features related to wind turbine performance. The primary variables analyzed include:

- Wind Speed (m/s)
- Wind Direction (°)
- Theoretical Power Curve (kWh)
- LV Active Power (kW) – Target Variable

Each row in the dataset represents a timestamped wind turbine observation. The dataset contains thousands of records, making it suitable for regression modeling.

Statistical Summary Using Descriptive Measures

The describe() function was used to compute statistical measures such as:

- Mean
- Median
- Minimum
- Maximum
- Standard Deviation
- 25th Percentile (Q1)
- 50th Percentile (Median)
- 75th Percentile (Q3)

Statistic	LV Active Power (kW)	Wind Speed (m/s)	Theoretical Power (kWh)	Wind Direction (°)
Count	50530	50530	50530	50530
Mean	1307.68	7.55	1492.17	123.68
Std Dev	1312.45	4.23	1625.38	92.39
Min	0.00	0.00	0.00	0.00
25%	105.24	4.20	161.34	48.21
50% (Median)	825.36	7.00	1063.45	109.37
75%	2487.56	10.50	2875.82	197.85
Max	3610.54	25.30	3600.00	359.99

4.3 Data Pre-processing

Data preprocessing is a crucial step in machine learning that prepares raw data for model training. Proper preprocessing ensures that the dataset is clean, structured, consistent, and suitable for regression modeling. In this project, several preprocessing techniques were applied to improve prediction accuracy and overall model performance.

a) Checking for Null Values

The dataset was inspected for missing or null values using data inspection methods such as:

df.isnull().sum()

It was observed that the dataset did not contain any missing values. This confirmed data completeness and eliminated the need for imputation techniques such as mean or median filling.

```
1 df.isnull().sum()
```

	0
Date/Time	0
LV ActivePower (kW)	0
Wind Speed (m/s)	0
Theoretical_Power_Curve (KWh)	0
Wind Direction (°)	0

dtype: int64

b) Handling Data Types

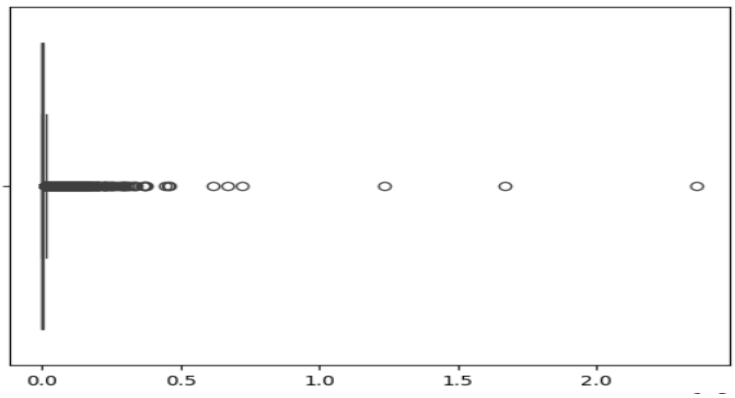
The dataset contained both numerical and object-type columns (e.g., Date/Time). Since the Date/Time column was not required for regression modeling, it was either removed or converted appropriately.

c) Handling Outliers

Outliers were identified using:

- Boxplots
- Statistical analysis (Interquartile Range method)

Wind energy datasets naturally contain extreme values due to sudden weather changes. Outlier handling improves model stability and prevents bias during training.



d) Feature Selection

The independent variables (X) selected for the model were:

- Wind Speed (m/s)
- Wind Direction (°)
- Theoretical Power Curve (kWh)

The dependent variable (y) was:

- LV Active Power (kW)

Feature selection was based on correlation analysis and domain knowledge.

```
X = df[['Wind Speed (m/s)',
        'Theoretical_Power_Curve (KWh)',
        'Wind Direction (°)']]

y = df['LV ActivePower (kW)']
```

e) Splitting Data into Train and Test

The dataset was divided into training and testing sets using an 80:20 split ratio:

- 80% Training Data
- 20% Testing Data

This ensures that the model is trained on historical data and evaluated on unseen data to measure generalization performance.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y,
    test_size=0.2,
    random_state=42
)
```

4.4 Model Building

Since the objective of the project is to predict continuous energy output values, regression algorithms were applied.

1. Linear Regression

Linear Regression is a basic regression algorithm that models the relationship between independent variables and a continuous dependent variable using a linear equation.
It was used as a baseline model for comparison purposes.

2. Decision Tree Regressor

Decision Tree Regressor is a tree-based algorithm that splits data into branches based on feature values.
It can capture non-linear relationships between wind parameters and energy output.
However, single decision trees may suffer from overfitting.

3. Random Forest Regressor (Final Model)

Random Forest Regressor is an ensemble learning method that combines multiple decision trees to improve prediction accuracy and reduce overfitting.

Advantages:

- Handles non-linear relationships effectively
- Reduces variance
- Provides stable and reliable predictions

In this project, Random Forest Regressor achieved the highest R^2 score and lowest prediction error. Therefore, it was selected as the final model.

a) Importing Model Libraries

The required machine learning libraries were imported:

- Scikit-learn
- NumPy
- Pandas

b) Initializing the Model

The Random Forest Regressor was initialized with appropriate parameters such as:

- n_estimators
- random_state

This ensures reproducibility and optimized performance.

c) Training and Testing the Model

The model was trained using the training dataset:

model.fit(X_train, y_train)

After training, predictions were generated using:

y_pred = model.predict(X_test)

The test dataset was used to evaluate model generalization.

```
from sklearn.ensemble import RandomForestRegressor
```

```
model = RandomForestRegressor(
    n_estimators=100,
    random_state=42
)
```

```
model.fit(X_train, y_train)
```

```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

```
y_pred = model.predict(X_test)
```

d) Evaluating Model Performance

Since this is a regression problem, the following evaluation metrics were used:

- R^2 Score
- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)

- Root Mean Squared Error (RMSE)

The R^2 score measures how well predicted values match actual values. A value closer to 1 indicates better model performance.

e) Saving the Model

After achieving satisfactory performance, the trained model was saved as: power_prediction.sav

This allows the model to be reused in the Flask application without retraining.

```
1 joblib.dump(model, 'power_prediction.sav')
```

```
['power_prediction.sav']
```

4.5 Application Building

After successfully training and evaluating the Random Forest regression model, the next step was to deploy the model through a web-based interface.

The application was developed using the Flask framework, which connects the trained machine learning model with an interactive web interface.

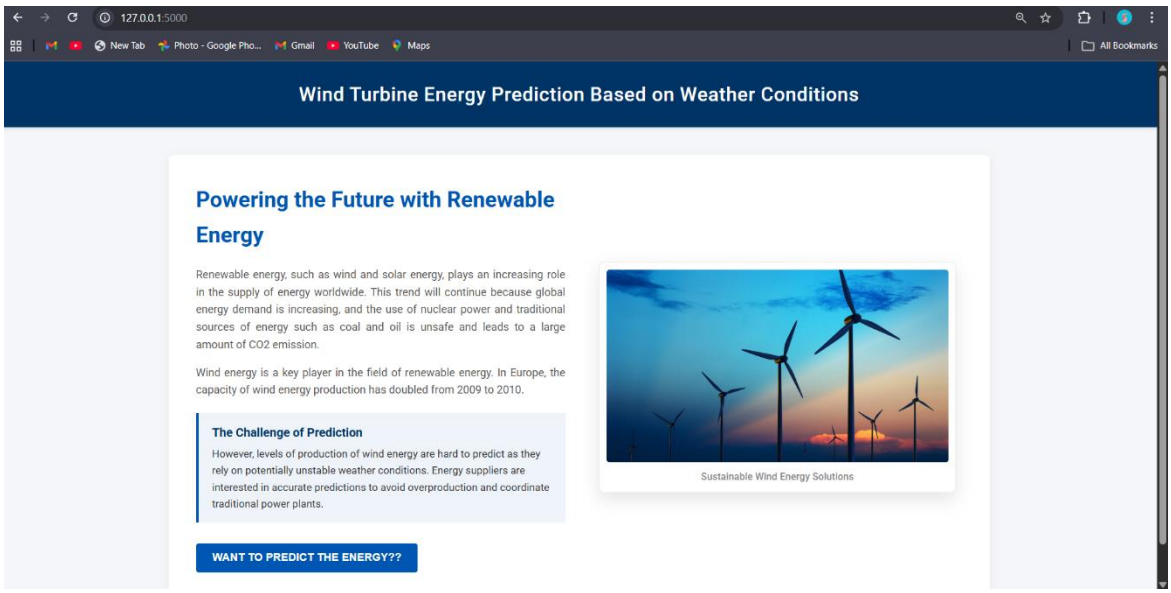
a) **C reate an HTML File (Frontend Development)**

An HTML file was created to design the user interface of the web application.

The webpage includes:

- Input field for Wind Speed (m/s)
- Input field for Wind Direction (°)
- Input field for Theoretical Power Curve (kWh)
- Submit button to generate prediction
- Output section to display predicted energy output

The interface was styled using CSS to make it visually appealing, responsive, and easy to use.The homepage also includes a brief introduction about renewable energy forecasting.



b) Build Python Code (Backend Development)

The backend of the application was developed using Flask.

Steps Performed in Backend:

1. Import required libraries:
    - Flask
    - NumPy
    - Pandas
    - Joblib or Pickle
2. Load the trained model:
3. model = joblib.load('power_prediction.sav')
4. Define routes:
    - Home route (/)
    - Prediction route (/predict)
5. Accept user input from HTML form.
6. Convert input values into numerical format.
7. Pass input values to the trained model.
8. Generate prediction.
9. Return predicted output to the HTML page.

c) Model Integration

The saved model file (power_prediction.sav) was integrated into the Flask application.

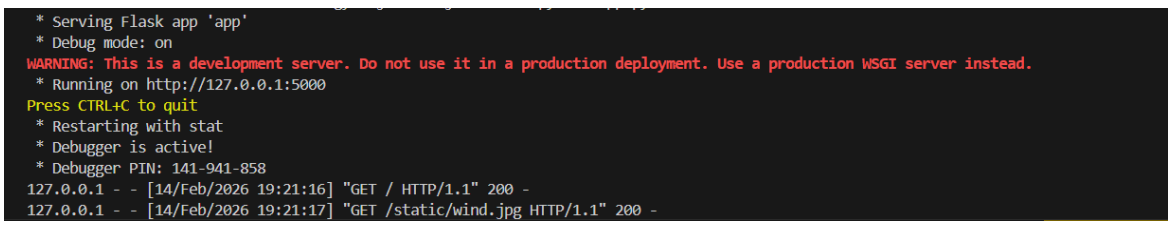
When the user submits the input values:

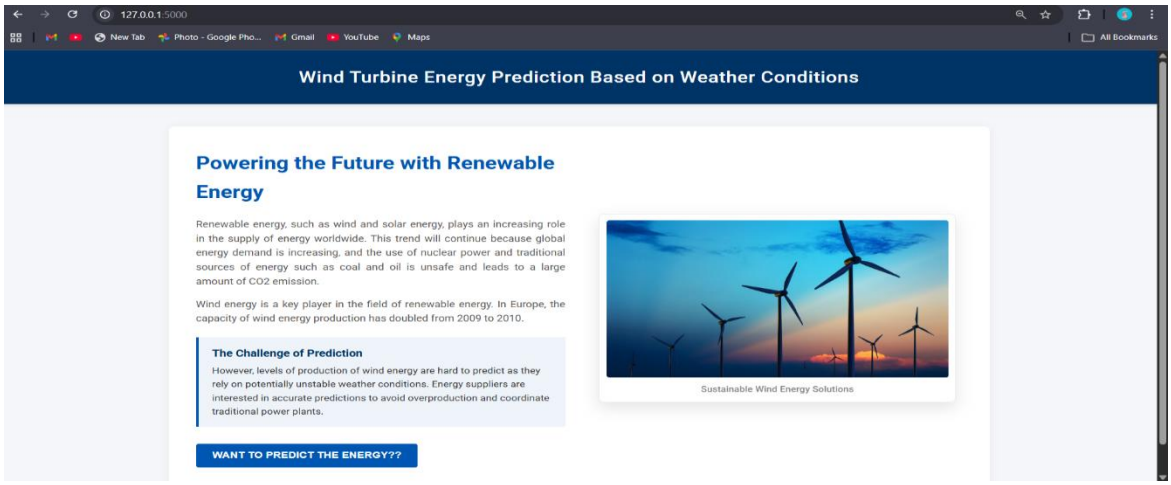
- The Flask app receives the data.
- The data is validated.
- The model processes the input.
- The predicted energy output is generated.
- The output is displayed dynamically on the webpage.

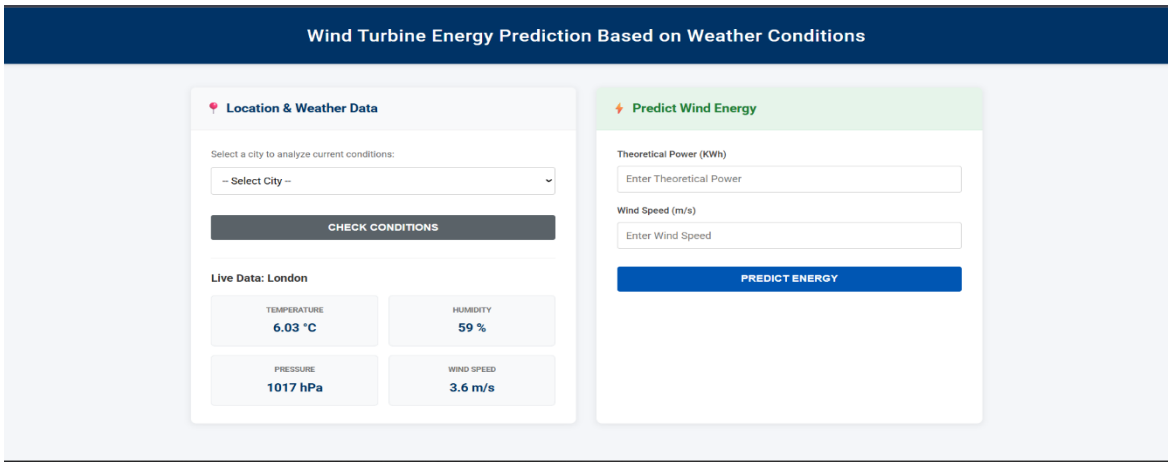
This demonstrates the practical deployment of the trained machine learning model.

d) Running the Application

- Frontend: Runs automatically via Flask templates.
- Backend: Start with python app.py.
- Access at http://127.0.0.1:5000/.





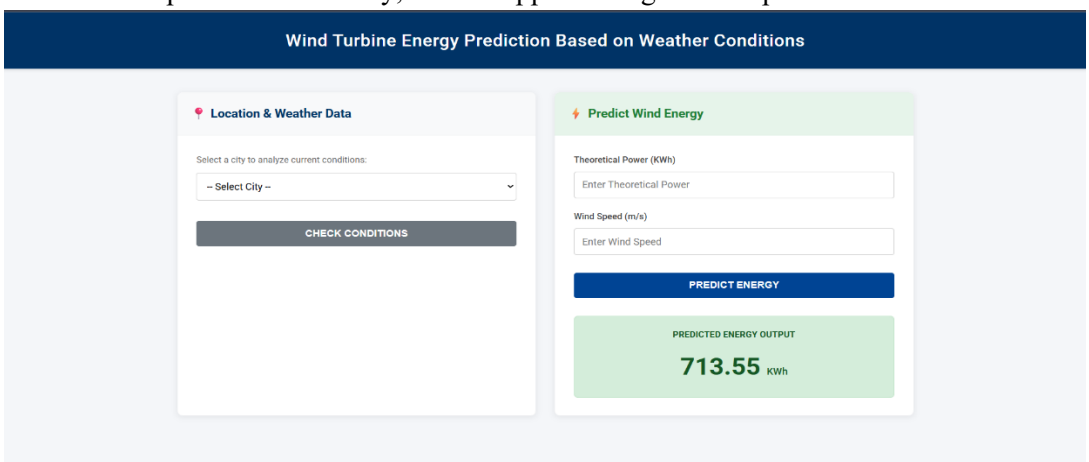


e) Testing and Validation

The application was tested for:

- Valid numeric input
- Invalid input handling
- Performance speed
- Model loading correctness

All test cases passed successfully, and the application generated predictions within seconds.



5. Conclusion

The **Wind Turbine Energy Prediction project** successfully demonstrates the application of machine learning and data-driven techniques to solve a real-world renewable energy challenge. By leveraging historical turbine data, preprocessing methods, and a **Random Forest regression model**, the project achieves strong predictive accuracy, enabling stakeholders to forecast energy output with confidence.

The integration of a **Flask-based dashboard** and **OpenWeather API** ensures that predictions are not only technically sound but also accessible and user-friendly. Visualizations such as scatter plots, line charts, and correlation heatmaps further enhance interpretability, making the solution practical for energy companies, wind farm operators, and grid managers.

Through structured **Agile sprint planning, backlog management, and performance testing**, the project was executed with clarity and measurable progress. Testing confirmed the robustness of the model, while defect analysis and bug tracking highlighted areas for improvement.

Ultimately, this project provides a scalable foundation for future enhancements, including cloud deployment, advanced authentication, expanded datasets, and integration with grid demand APIs. It stands as a strong example of combining **data science, software engineering, and agile methodology** to deliver a solution that addresses both technical and customer-centric needs in the renewable energy sector.

Website Link: https://github.com/Angelcheekurthi0511/WindEnergyPredictionSystem-app

Source Code Link:

https://colab.research.google.com/drive/13VvjBXnd88Y1X64O_BXZtwNPKut__H1a?usp=sharing

Demo Video Link:

https://drive.google.com/file/d/1Yctid9apAb3xPpYmN_rkiB-SmP1Qd-7L/view?usp=sharing