Weather-Based Prediction of Wind Turbine Energy Output

Team details-

Team ID : LTVIP2026TMIDS89922

Team Size : 4

Team Leader : Koppineedi Lakshmi Priya

Team member : Bhavana Tamarapu

Team member : Sai Venkata Pavan Nagireddy

Team member : Videep Ramapogu

College name : Vishnu institute of technology

1. INTRODUCTION

1.1 Project Overview

The project titled “Weather-Based Prediction of Wind Turbine Energy Output” focuses on forecasting wind turbine power generation using machine learning techniques. Wind energy is one of the fastest-growing renewable energy sources globally, contributing significantly to sustainable energy production.

The system uses historical SCADA data from wind turbines (Kaggle dataset) and builds predictive models to estimate energy output (Active Power in kW) based on weather parameters such as:

- Wind Speed (m/s)
- Theoretical Power Curve (KWh)

The trained model is deployed using the Flask web framework, allowing users to:

- Fetch real-time weather data via OpenWeather API
- Automatically populate wind speed values
- Predict turbine energy output instantly

The project integrates Data Science + Machine Learning + Web Development, forming a complete Full Stack Data Science solution.

1.2 Purpose

The purpose of this project is to:

- Improve energy production forecasting
- Assist grid operators in balancing electricity supply
- Help wind farm operators plan maintenance
- Reduce operational inefficiencies
- Promote smart renewable energy management

2. IDEATION PHASE

2.1 Problem Statement

Wind energy production is highly dependent on weather conditions, especially wind speed. Due to unpredictable weather:

- Energy companies face difficulty forecasting power generation.
- Grid operators struggle with load balancing.
- Maintenance scheduling becomes inefficient.
- Overproduction or underproduction can cause financial losses.

Problem:
How can we accurately predict wind turbine energy output using weather parameters to improve operational efficiency and energy management?

2.2 Empathy Map Canvas

Users:

- Wind Farm Operators
- Energy Companies
- Grid Operators
- Renewable Energy Analysts

Think & Feel:

- Concerned about unpredictable wind patterns
- Need accurate forecasting
- Want reduced downtime
- Focused on operational efficiency

See:

- Fluctuating energy production
- Weather variability
- Grid instability

Say & Do:

- Monitor SCADA systems
- Analyze weather reports
- Schedule maintenance manually

Pain Points:

- Revenue loss due to poor prediction
- Maintenance during high-wind periods
- Grid imbalance issues

Gains:

- Accurate forecasting
- Better planning
- Improved profit margins
- Reduced downtime

2.3 Brainstorming

Possible solutions explored:

- Statistical regression models
- Time-series forecasting (ARIMA)
- Machine Learning regression algorithms
- Deep Learning models (LSTM)
- Real-time API integration
- Dashboard-based prediction system

Final approach chosen:

- Random Forest Regression
- Real-time Weather API integration
- Flask-based Web Application

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

1. User opens the web application.
2. User enters city name.
3. System fetches real-time weather data.
4. Wind speed auto-fills in prediction form.
5. User clicks “Predict”.
6. Model returns energy output.
7. User makes operational decisions.

3.2 Solution Requirements

Functional Requirements

- Fetch real-time weather data using OpenWeather API.
- Predict energy output using trained ML model.

- Display prediction result.
- Provide user-friendly interface.
- Auto-fill wind speed from API.
- Handle invalid input gracefully.

Non-Functional Requirements

- Fast prediction (< 2 seconds)
- High accuracy model (R^2 score)
- Scalable architecture
- Secure API handling
- Responsive UI design

3.3 Data Flow Diagram (DFD)

Level 0 DFD

User → Web Interface → Flask Backend → ML Model → Prediction Output

Level 1 DFD

1. User inputs city name
2. Flask sends request to OpenWeather API
3. Weather data retrieved
4. Data sent to ML Model
5. Model predicts energy output
6. Result displayed to user

3.4 Technology Stack

Programming Language:

- Python

Libraries:

- NumPy
- Pandas
- Matplotlib
- Seaborn
- Scikit-learn
- Joblib
- Requests

Machine Learning Algorithms:

- Linear Regression
- Decision Tree Regressor
- Random Forest Regressor (Final Model)

Web Framework:

- Flask

Frontend:

- HTML
- CSS

API:

- OpenWeather API

Dataset:

Wind Turbine SCADA Dataset from Kaggle
https://www.kaggle.com/datasets/sulymansifat/wind-turbine-scada-dataset

4. PROJECT DESIGN

4.1 Problem-Solution Fit

Problem	**Solution**
Unpredictable wind energy output	Machine Learning-based regression model
Grid imbalance	Forecasting system
Inefficient maintenance planning	Predict low-wind periods
Manual estimation errors	Automated prediction system

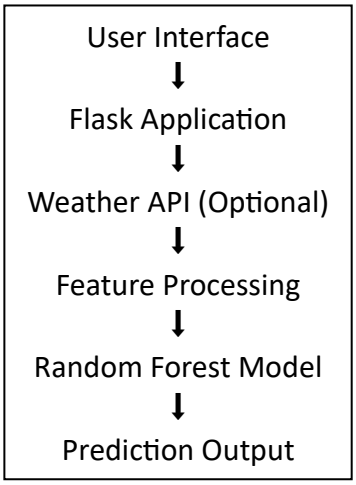
4.2 Proposed Solution

The system:

1. Preprocesses historical wind turbine dataset.
2. Performs exploratory data analysis.
3. Trains multiple regression models.
4. Evaluates models using:
    - R^2 Score
    - Mean Absolute Error
5. Selects Random Forest as best performer.
6. Saves model using joblib.
7. Deploys model via Flask app.
8. Integrates weather API.
9. Displays prediction on web dashboard.

4.3 Solution Architecture

Architecture Flow:



Model File: power_prediction.sav
Backend File: app.py
Templates: intro.html, predict.html

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

Phase	Duration
Problem Understanding	1 Week
Data Collection	1 Week
Data Cleaning & EDA	1 Week
Model Training	1 Week
Model Evaluation	1 Week
Web Application Development	1 Week
Testing & Debugging	1 Week
Documentation	1 Week

Total Duration: 8 Weeks

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Functional Testing

- Verified API returns correct weather data.
- Checked auto-fill functionality.
- Tested prediction for multiple inputs.
- Verified model loading.
- Tested invalid city input handling.

6.2 Performance Testing

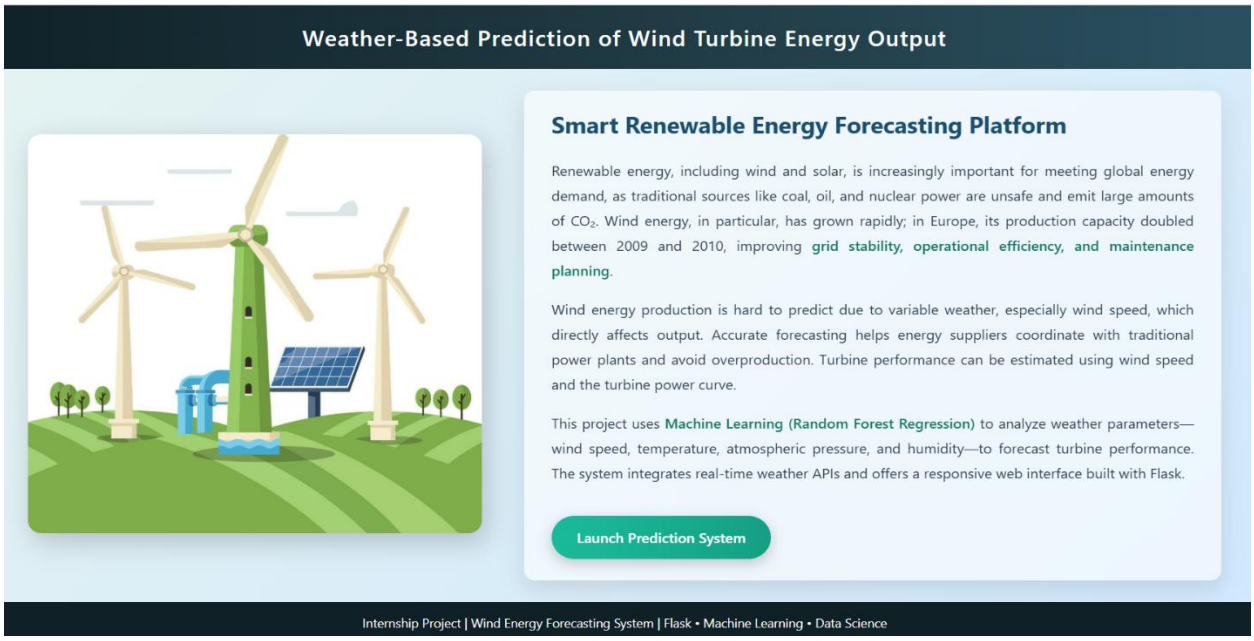
- Response time: < 2 seconds
- Model Accuracy:
    - Linear Regression R^2: 0.9056825575631955
    - Decision Tree R^2: 0.8277564120240467
    - Random Forest R^2: 0.9113496428907649

Random Forest chosen due to highest accuracy.

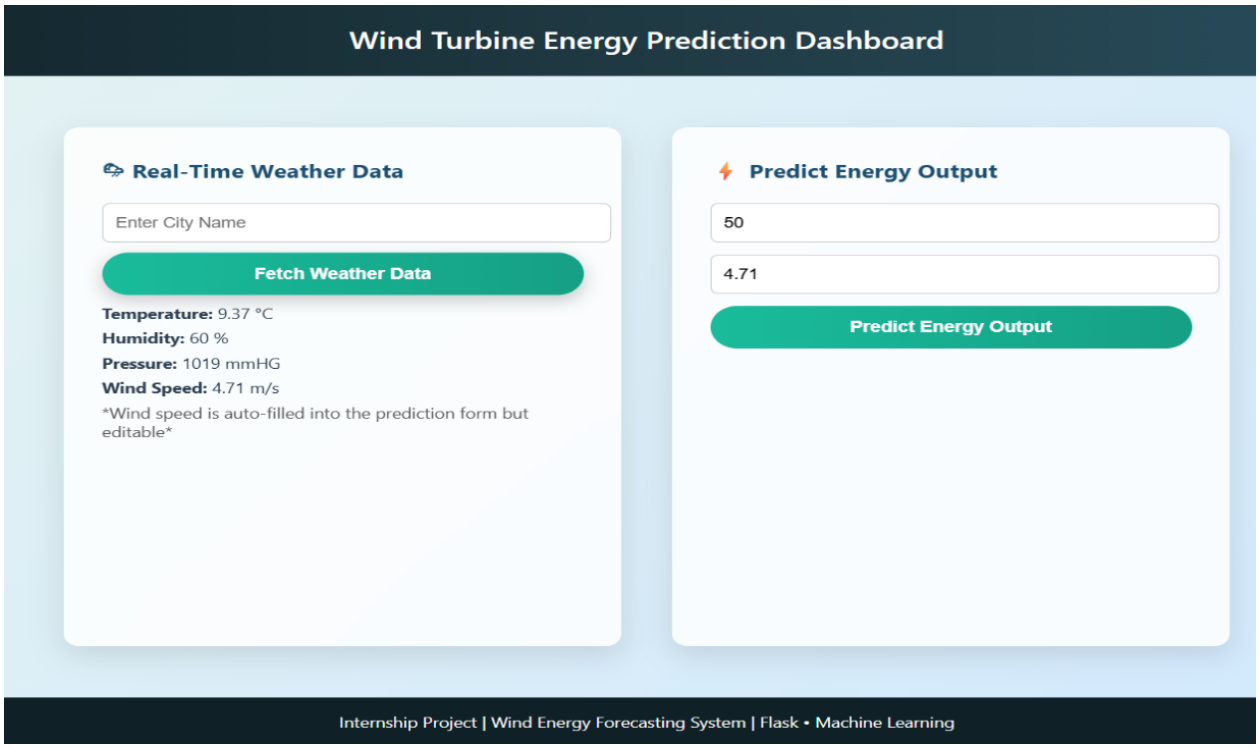
7. RESULTS

7.1 Output Screenshots

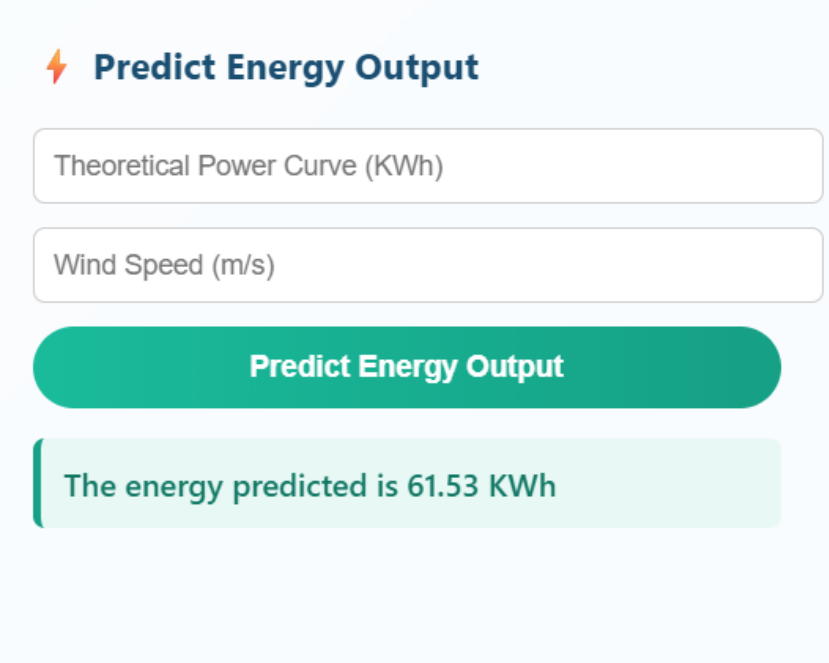
- Home Page



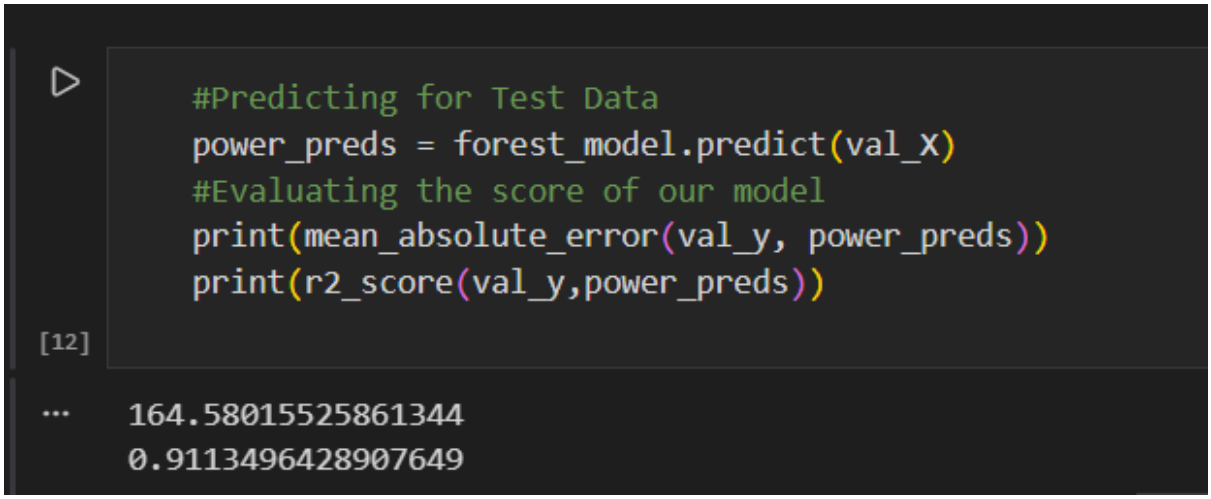
- Weather Fetching



- Prediction Result



- Model Accuracy Output



- Correlation Heatmap

8. ADVANTAGES & DISADVANTAGES

Advantages

✓ Improves energy forecasting
✓ Helps grid balancing
✓ Reduces maintenance downtime
✓ Real-time weather integration
✓ User-friendly interface

Disadvantages

✕ Accuracy depends on dataset quality
✕ Simplified theoretical power curve
✕ Limited features used for prediction
✕ API dependency

9. CONCLUSION

The project successfully demonstrates how machine learning can enhance renewable energy management. The Random Forest regression model provides accurate predictions of wind turbine energy output based on weather parameters.

The integration of real-time weather data and web deployment makes the system practical and scalable for real-world use.

10. FUTURE SCOPE

- Integrate more weather parameters (humidity, temperature, pressure)
- Use Deep Learning (LSTM for time-series forecasting)
- Add historical graph visualization
- Deploy on AWS / Azure / Heroku
- Implement user authentication
- Create Admin Dashboard
- Improve theoretical power curve modeling
- Integrate multiple turbine prediction

11. APPENDIX

Source Code

```
import numpy as np

from flask import Flask, request, jsonify, render_template

import joblib

import requests


app = Flask (__name__)
```

```
model = joblib.load('power_prediction.sav')


@app.route('/')
def home():
    return render_template('intro.html')
@app.route('/predict')
def predict():
    return render_template('predict.html')


@app.route('/windapi', methods=['POST'])
def windapi():
    city = request.form.get('city')


    f=open(r"D:/Docu/pyjunb/API_keys/weather api.txt")
    apikey=f.read()


    url = f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={apikey}"


    resp = requests.get(url)
    resp_json = resp.json()


    temp_kelvin = resp_json["main"]["temp"]
    temp_celsius = round(temp_kelvin - 273.15, 2)  # Convert to Celsius
    humid = str(resp_json["main"]["humidity"]) + " %"
    pressure = str(resp_json["main"]["pressure"]) + " mmHG"
    wind_speed = float(resp_json["wind"]["speed"])
```

```
wind_speed_str = str(wind_speed)  # numeric value for input


# Simple example theoretical power curve based on wind speed
def get_theoretical_power(ws):
    if ws < 3:
        return 0
    elif ws < 5:
        return 50
    elif ws < 8:
        return 150
    elif ws < 12:
        return 300
    else:
        return 500


theoretical_power = get_theoretical_power(wind_speed)


return render_template(
    'predict.html',
    temp=f"{temp_celsius} °C",
    humid=humid,
    pressure=pressure,
    speed=str(wind_speed) + " m/s",
    wind_speed_value=wind_speed,          # auto-fill wind speed
    theoretical_power=theoretical_power   # auto-fill theoretical power
)
```

```
@app.route('/y_predict', methods=['POST'])
def y_predict():
    '''
    For rendering results on HTML GUI
    '''
    x_test = [[float(x) for x in request.form.values()]]
    prediction = model.predict(x_test)
    print(prediction)
    output=prediction [0]
    return render_template('predict.html', prediction_text='The energy predicted is {:.2f} KWh'.format(output))

if __name__ == "__main__":
    app.run(debug=False)
```

Dataset Link

https://www.kaggle.com/datasets/sulymansifat/wind-turbine-scada-dataset

GitHub & Demo Link

https://github.com/priyakoppineedi/Weather-Based-Prediction-of-Wind-Turbine-Energy-Output