Weather-Based Wind Turbine Energy Prediction

Project Description:

One of the most important factors affecting modern power systems and sustainable development is the efficient use of renewable energy resources. Wind energy has emerged as a key contributor to clean energy production across the globe. However, the amount of power generated by wind turbines depends heavily on weather conditions, making accurate energy prediction a challenging task.

The prediction of wind turbine energy output is essential for grid stability, optimizing energy distribution, and reducing power losses. By forecasting energy generation using weather parameters, energy providers can improve planning and ensure efficient utilization of renewable resources. This makes the study of wind energy prediction improving highly important. Machine Learning techniques play a crucial role in analyzing weather data and predicting turbine power output.

In this project, a Random Forest regression model is used to predict wind turbine energy output based on wind speed and wind direction. The model is trained and tested using historical wind turbine SCADA data obtained from Kaggle. The trained model is saved in .pkl format and integrated into a Streamlit web application for user interaction. Real-time weather data is fetched using the OpenWeatherMap API, and wind parameters are automatically passed to the model for live prediction. The application is tested using ngrok to enable external access during demonstration.

Technical Architecture:

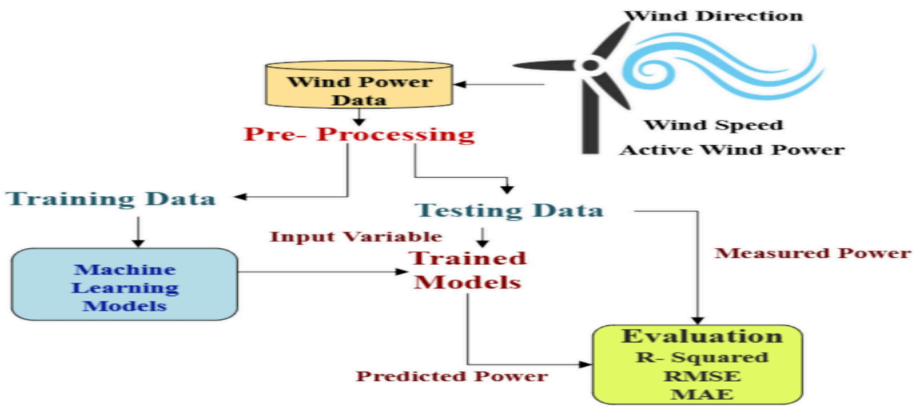


Figure: System architecture for real-time wind energy prediction using machine learning and weather API.

Pre-requisites:

To complete this project, you must required following software's, concepts and packages

Software & Tools:

- Python
- Google Colab
- VS Code (optional for development)
- GitHub
- Streamlit
- Ngrok

Python Packages:

Install using pip:

- pip install numpy
- pip install pandas
- pip install scikit-learn
- pip install matplotlib
- pip install streamlit
- pip install joblib
- pip install requests

Prior Knowledge:

You must have prior knowledge of following topics to complete this project.

- Machine Learning fundamentals
- Supervised learning
- Random Forest regression
- Data preprocessing techniques
- Model evaluation metrics (MAE, RMSE, R^2)
- Streamlit basics
- GitHub version control
- API integration basics

Project Objectives:

By the end of this project you will:

- Understand the relationship between weather parameters and wind energy generation
- Gain knowledge of data preprocessing and feature selection techniques
- Build a machine learning model for energy prediction
- Evaluate model performance using standard metrics
- Develop a Streamlit application for real-time prediction
- Demonstrate deployment using ngrok

Project Flow:

- User selects a city in the Streamlit interface
- Weather data is fetched using OpenWeatherMap API
- Wind speed and wind direction are extracted automatically
- The trained Random Forest regression model processes the input
- Predicted wind turbine power output is displayed on the interface

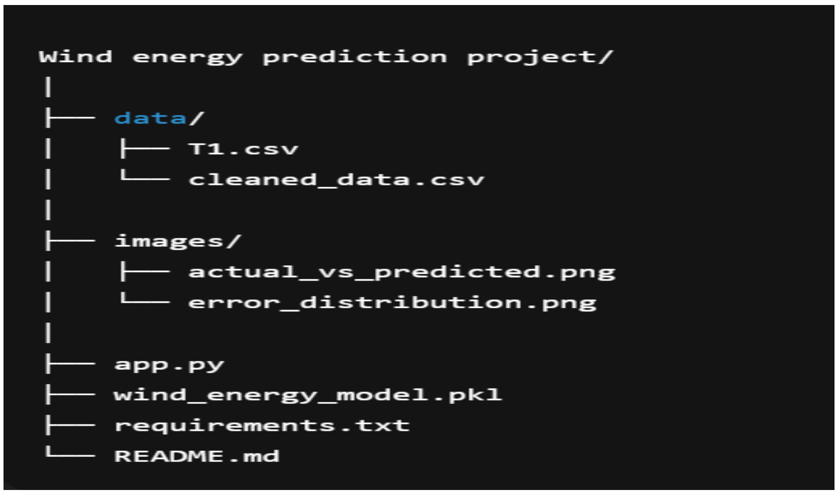
Activities Involved:

To accomplish this project, the following activities are performed:

- Data collection from Kaggle
- Data preprocessing and cleaning
- Data visualization and analysis
- Feature selection
- Splitting data into training and testing sets
- Model building and evaluation
- Saving the trained model
- Application development using Streamlit
- Testing using ngrok
- Real-time weather API integration

Project Structure:

Create the Project folder which contains files as shown below



```
Wind energy prediction project/
|
├── data/
|    ├── T1.csv
|    └── cleaned_data.csv
|
├── images/
|    ├── actual_vs_predicted.png
|    └── error_distribution.png
|
├── app.py
├── wind_energy_model.pkl
├── requirements.txt
└── README.md
```

Explanation of Project Structure:

- The data folder contains the raw dataset (T1.csv) and the cleaned dataset (cleaned_data.csv) used for training the machine learning model.
- The images folder contains evaluation graphs such as Actual vs Predicted and Error Distribution, which are used to analyze model performance.
- app.py is the Streamlit application script that provides a user interface for entering weather parameters and predicting wind turbine energy output.
- wind_energy_model.pkl is the trained machine learning model saved using Joblib. This model is loaded in the Streamlit application to perform predictions.
- requirements.txt contains the list of required Python libraries needed to run the project.
- README.md provides project documentation, including setup instructions and project details.
- The notebook folder contains model training and evaluation code developed in Google Colab.

Milestone 1: Data Collection

Machine learning models depend heavily on data. It is the most crucial aspect that enables algorithm training and accurate prediction. This section describes how the dataset required for wind turbine energy prediction was collected.

Activity 1: Download the dataset

There are many popular open-source platforms for collecting datasets, such as Kaggle and the UCI Repository.
In this project, we used the **Wind Turbine SCADA Dataset** obtained from **Kaggle**. This dataset contains weather parameters and corresponding wind turbine power output values, which are essential for training the prediction model.
The dataset includes features such as:

- Wind Speed
- Wind Direction
- Power Output (Target Variable)

The dataset was downloaded from Kaggle and uploaded into **Google Colab** for preprocessing And model development.
Link:https://www.kaggle.com/datasets/berkerisen/wind-turbine-scada-dataset

Milestone 2: Visualizing and analysing the data

After downloading the dataset, it is important to understand the data using visualization and analysis techniques. This helps in identifying patterns, detecting anomalies, and selecting important features for model training.
Note: There are many techniques available for data analysis. In this project, we used selected visualization methods to understand the dataset effectively.

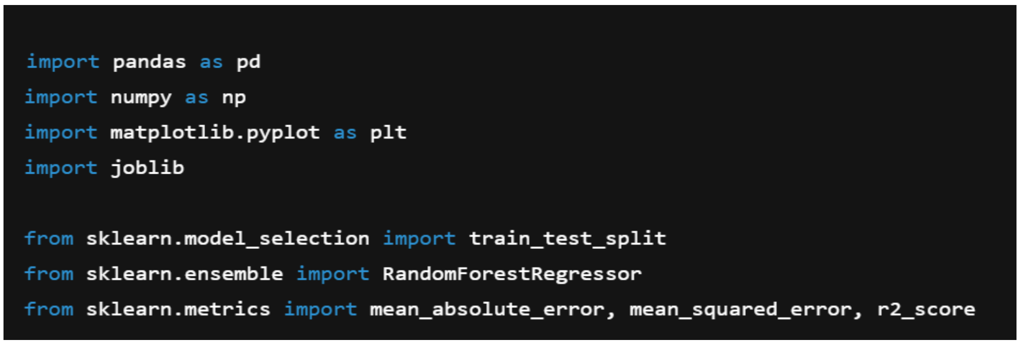
Activity 1: Importing the libraries

The necessary Python libraries were imported for data analysis and visualization.

Libraries used:

- Pandas – for data handling
- NumPy – for numerical operations
- Matplotlib – for data visualization
- Requests – for fetching real-time weather data from OpenWeatherMap API

These libraries help in exploring the dataset and preparing it for preprocessing and model training.



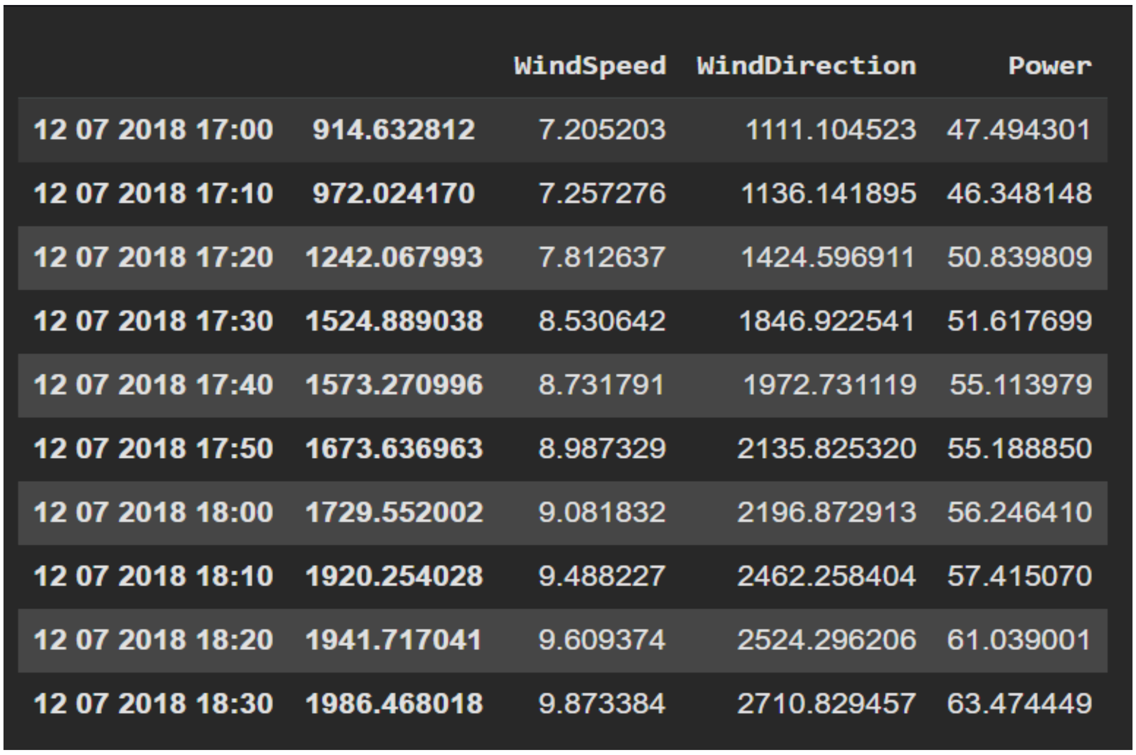
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import joblib

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

Activity 2: Read the Dataset

The dataset used in this project is stored in CSV format. It was loaded into Google Colab using the Pandas library, which provides efficient data handling and manipulation capabilities.

Pandas provides the read_csv() function to read CSV files. The dataset file path is passed as a parameter to this function.

Since the original Kaggle dataset did not contain column headers, the columns were manually named as **WindSpeed**, **WindDirection**, and **Power**. These columns were then used for analysis and model training.



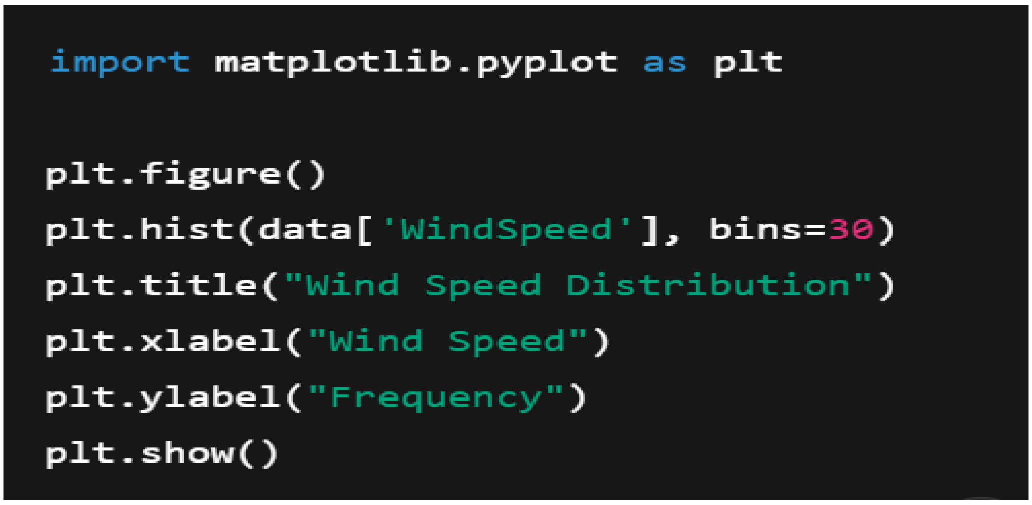
		WindSpeed	WindDirection	Power
12 07 2018 17:00	**914.632812**	7.205203	1111.104523	47.494301
12 07 2018 17:10	**972.024170**	7.257276	1136.141895	46.348148
12 07 2018 17:20	**1242.067993**	7.812637	1424.596911	50.839809
12 07 2018 17:30	**1524.889038**	8.530642	1846.922541	51.617699
12 07 2018 17:40	**1573.270996**	8.731791	1972.731119	55.113979
12 07 2018 17:50	**1673.636963**	8.987329	2135.825320	55.188850
12 07 2018 18:00	**1729.552002**	9.081832	2196.872913	56.246410
12 07 2018 18:10	**1920.254028**	9.488227	2462.258404	57.415070
12 07 2018 18:20	**1941.717041**	9.609374	2524.296206	61.039001
12 07 2018 18:30	**1986.468018**	9.873384	2710.829457	63.474449

Activity 3: Univariate analysis

Univariate analysis involves examining individual features in the dataset to understand their distribution and characteristics.
In this project, univariate analysis was performed on important features such as:

- Wind Speed
- Wind Direction
- Power Output

This analysis helped understand the distribution and range of each feature.



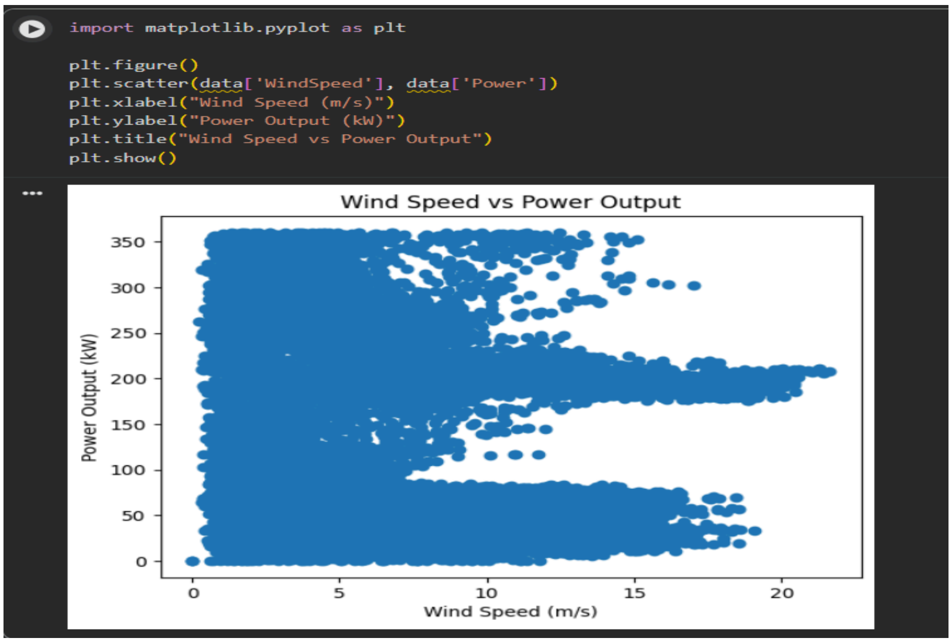
```
import matplotlib.pyplot as plt

plt.figure()
plt.hist(data['WindSpeed'], bins=30)
plt.title("Wind Speed Distribution")
plt.xlabel("Wind Speed")
plt.ylabel("Frequency")
plt.show()
```

Activity 4: Bivariate analysis

Bivariate analysis is used to study the relationship between two variables. In this project, it helps in understanding how weather parameters affect wind turbine power output.
The following relationships were analyzed:

- Wind Speed vs Power Output
- Wind Direction vs Power Output

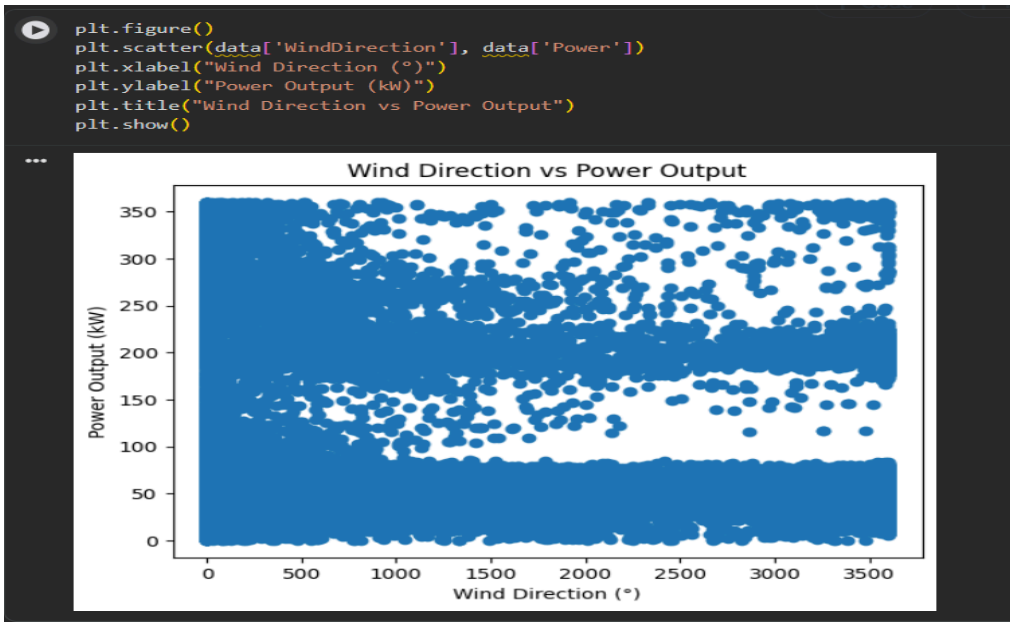
These relationships help identify how changes in weather conditions influence energy generation.



```
import matplotlib.pyplot as plt

plt.figure()
plt.scatter(data['WindSpeed'], data['Power'])
plt.xlabel("Wind Speed (m/s)")
plt.ylabel("Power Output (kW)")
plt.title("Wind Speed vs Power Output")
plt.show()
```

Figure: Wind Speed vs Power Output

This scatter plot shows the relationship between wind speed and power output. It helps identify how increases in wind speed affect turbine energy generation.



```
plt.figure()
plt.scatter(data['WindDirection'], data['Power'])
plt.xlabel("Wind Direction (°)")
plt.ylabel("Power Output (kW)")
plt.title("Wind Direction vs Power Output")
plt.show()
```

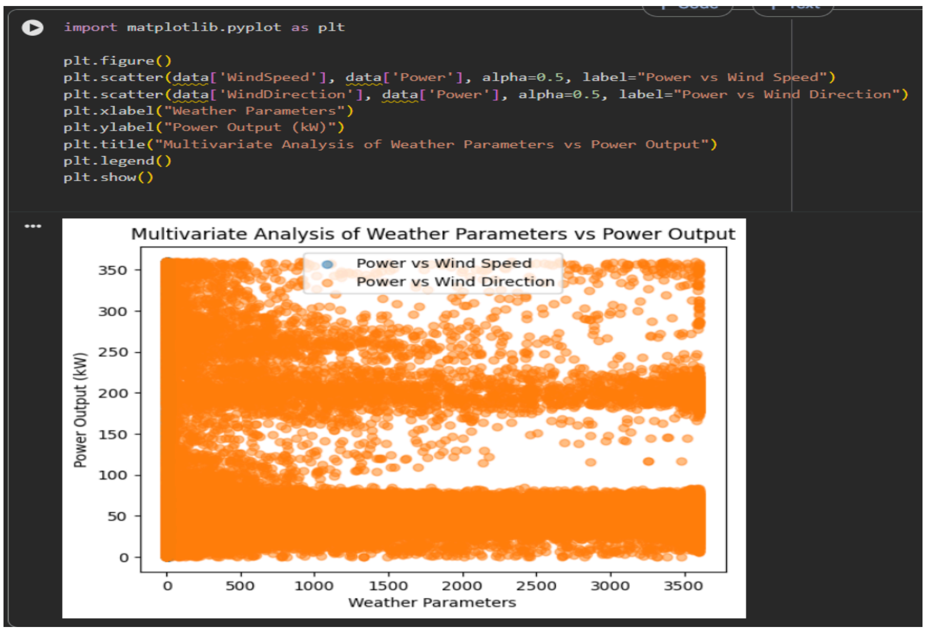
Figure: Wind Direction vs Power Output

This plot helps analyze how wind direction impacts power generation. It can reveal patterns related to turbine alignment and wind flow.

This analysis justifies selecting wind speed and wind direction as input features for the Random Forest regression model.

Activity 5: Multivariate analysis

Multivariate analysis is used to study the relationship between multiple features simultaneously. In this project, it helps understand how wind speed and wind direction together influence power output.
The analysis was performed using scatter plots to visualize the combined effect of multiple weather parameters on wind turbine energy generation.



```
import matplotlib.pyplot as plt

plt.figure()
plt.scatter(data['WindSpeed'], data['Power'], alpha=0.5, label="Power vs Wind Speed")
plt.scatter(data['WindDirection'], data['Power'], alpha=0.5, label="Power vs Wind Direction")
plt.xlabel("Weather Parameters")
plt.ylabel("Power Output (kW)")
plt.title("Multivariate Analysis of Weather Parameters vs Power Output")
plt.legend()
plt.show()
```

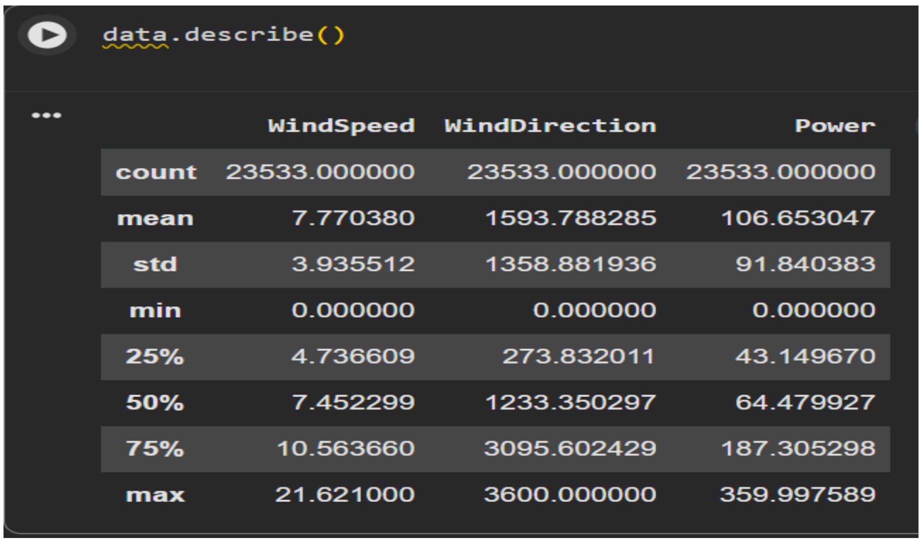
Figure: Multivariate Analysis Showing Relationship Between Weather Parameters and Power Output

The above visualization demonstrates how multiple weather parameters influence power generation. By analyzing wind speed and wind direction together, we can better understand the conditions that lead to higher energy output.
These weather parameters are used as input variables for training the regression model.

Activity 6: Descriptive analysis

Descriptive analysis is used to understand the statistical properties of the dataset. It helps summarize the central tendency, dispersion, and distribution of numerical features.
In this project, the describe() function from Pandas was used to obtain statistical insights for key features such as:

- Wind Speed
- Wind Direction
- Power Output



```
data.describe()
```

	WindSpeed	WindDirection	Power
count	23533.000000	23533.000000	23533.000000
mean	7.770380	1593.788285	106.653047
std	3.935512	1358.881936	91.840383
min	0.000000	0.000000	0.000000
25%	4.736609	273.832011	43.149670
50%	7.452299	1233.350297	64.479927
75%	10.563660	3095.602429	187.305298
max	21.621000	3600.000000	359.997589

Figure: Descriptive Statistics of Wind Turbine Dataset

The describe() function provides:

- Count of values
- Mean (average)
- Standard deviation
- Minimum and maximum values
- Quartiles (25%, 50%, 75%)

The statistical summary confirms that the dataset is suitable for regression modeling and does not require normalization since a tree-based algorithm is used.

This helps in understanding the range and distribution of weather parameters and power output.

Activity 7: Visualization of Results

Actual vs Predicted Power Output:

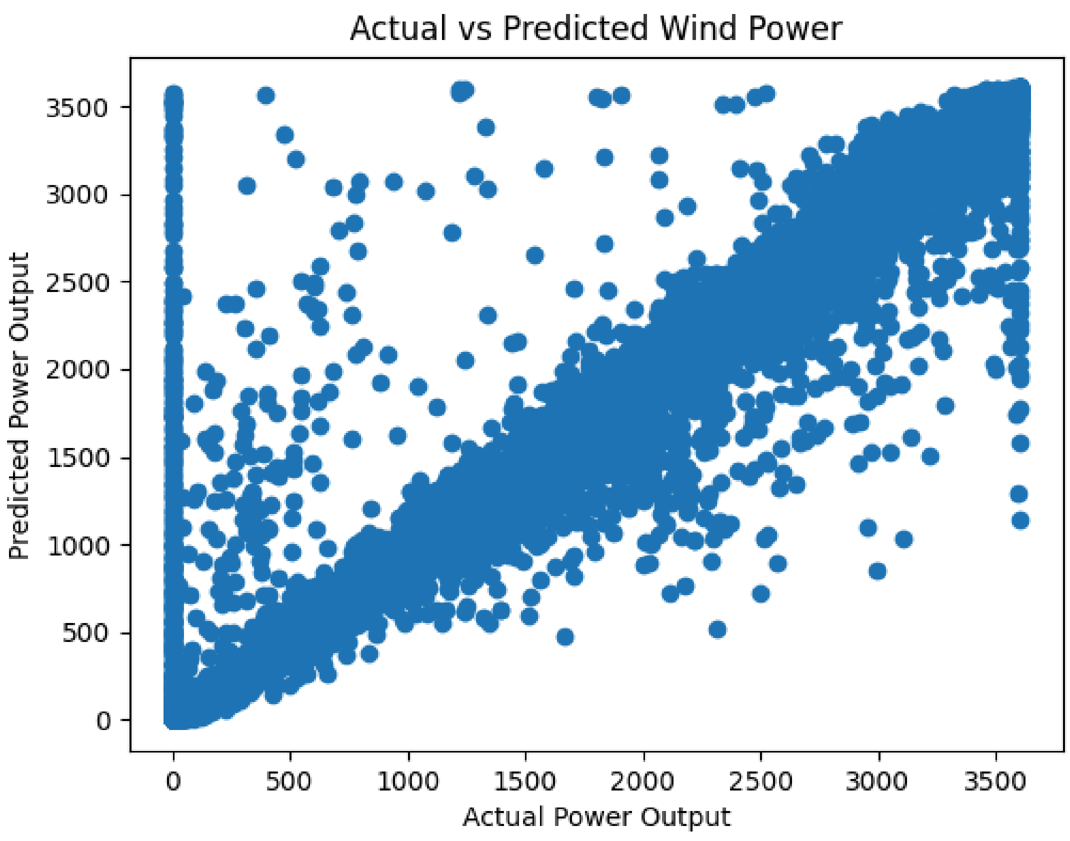


Figure: Actual vs Predicted Power Output

This graph compares the actual power values with the values predicted by the Random Forest model. The close alignment between the two indicates that the model has learned the underlying pattern effectively.

Error Distribution:

The error distribution graph shows the difference between actual and predicted values. Most errors are concentrated near zero, indicating good prediction accuracy.

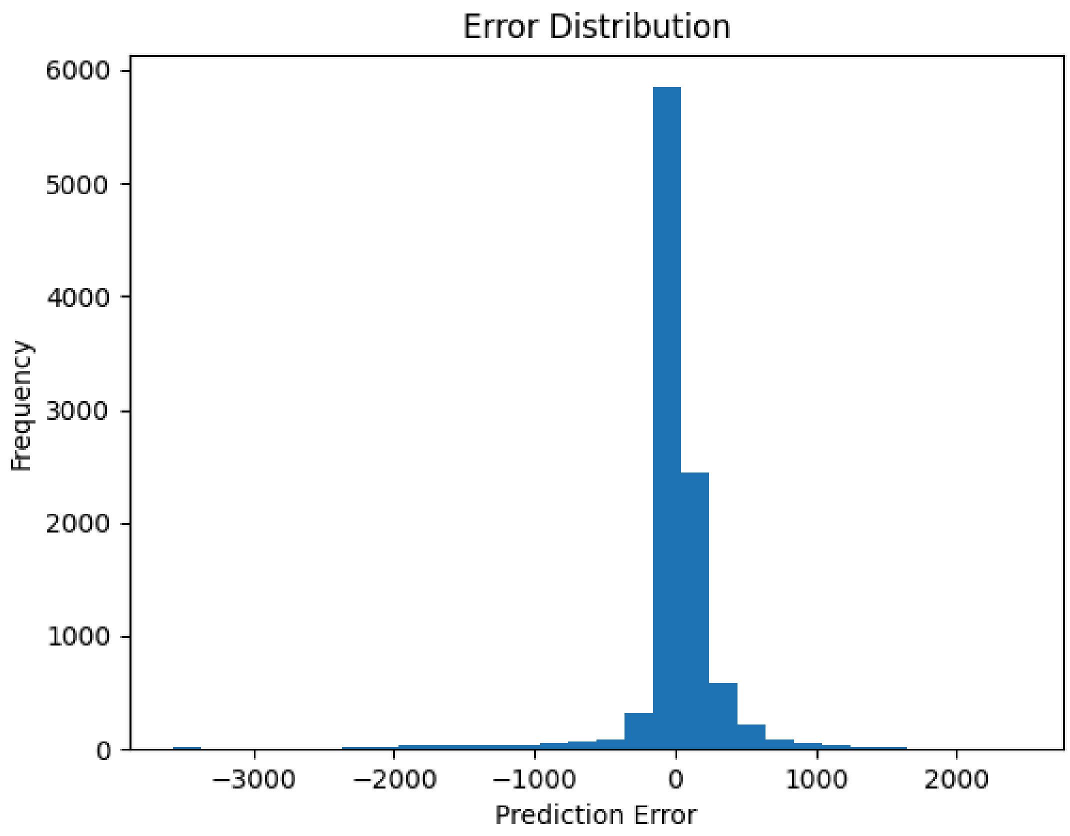


Figure: Error Distribution of Model Predictions

Milestone 3: Data Pre-processing

After understanding the dataset, the next step is to preprocess the data to make it suitable for machine learning model training.
The raw dataset may contain noise, missing values, or irrelevant features that can affect model performance. Proper preprocessing ensures accurate predictions.

Preprocessing Steps Performed:

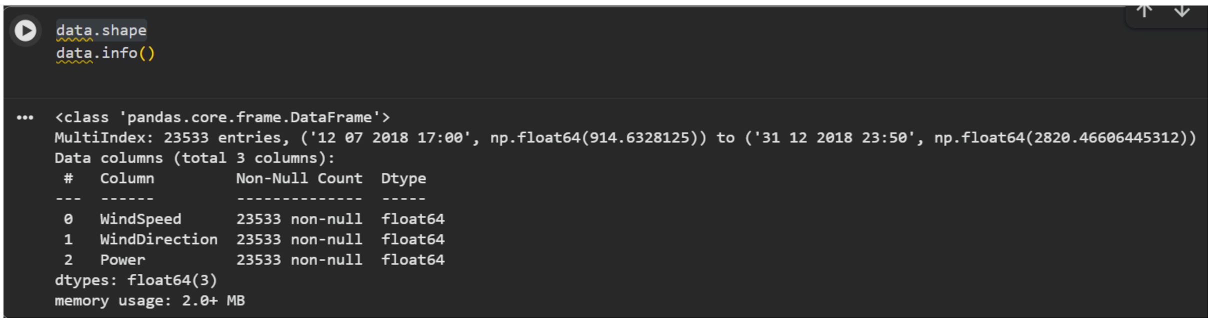
- Assigning column names to the dataset
- Selecting wind speed and wind direction as input features
- Splitting dataset into training and testing sets

Feature scaling was not applied because Random Forest is a tree-based algorithm.

Note: Since the dataset contains numerical values only, categorical encoding was not required.

Activity 1: Checking for null values

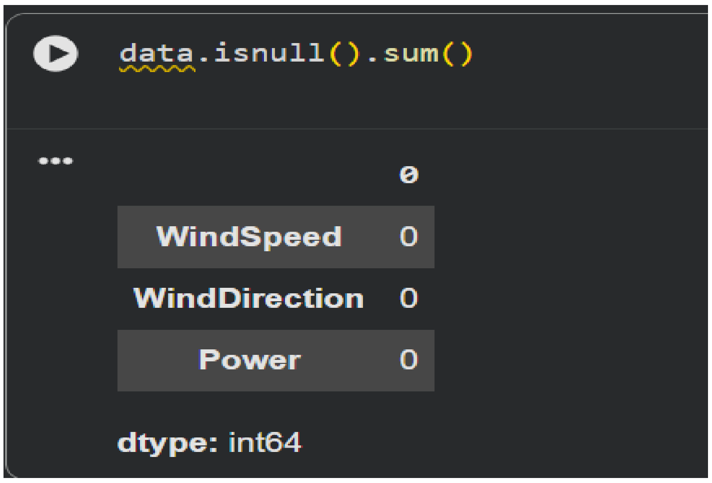
- Before training the model, it is important to verify whether the dataset contains missing values.



```python
data.shape
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 23533 entries, ('12 07 2018 17:00', np.float64(914.6328125)) to ('31 12 2018 23:50', np.float64(2820.46606445312))
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   WindSpeed      23533 non-null  float64
 1   WindDirection  23533 non-null  float64
 2   Power          23533 non-null  float64
dtypes: float64(3)
memory usage: 2.0+ MB
```

- For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function to it. From the below image we found that there are no null values present in our dataset. So we can skip handling of missing values step.



```python
data.isnull().sum()
```

```
               0
WindSpeed      0
WindDirection  0
Power          0

dtype: int64
```

- shape shows the number of rows and columns
- info() displays data types and non-null counts

These checks ensure the dataset is clean and ready for model training.

The dataset was confirmed to be clean with no missing values, so no imputation techniques were required.

Activity 2: Feature Selection

The dataset contains multiple features related to wind turbine performance. For this project, only relevant weather parameters that directly influence energy generation were selected. Only wind speed and wind direction were used as input features for the regression model.

Selected Features:

- Wind Speed
- Wind Direction
- Power Output (Target Variable)

Selecting relevant features improves model performance and reduces computational complexity.

Activity 3: Handling Outliers (If Present)

Outliers can negatively affect model performance. The dataset was visually inspected using scatter plots and descriptive statistics to identify extreme values.
If outliers are present, they can be handled using techniques such as:

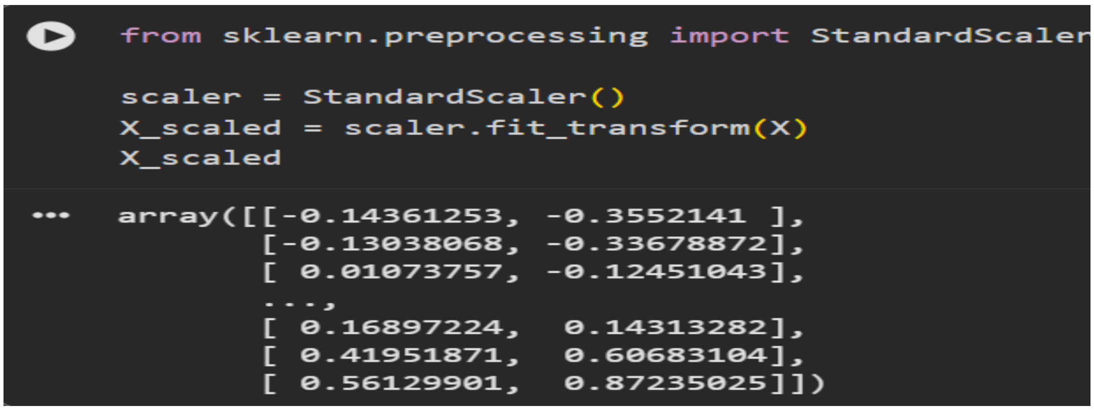
- Removing extreme values
- Using interquartile range (IQR) method

In this project, the dataset was found to be consistent, and no significant outliers required removal.

Activity 4: Scaling the Data

Feature scaling is generally used when models are sensitive to the magnitude of input values. However, in this project a **Random Forest Regression** model was used.

Since Random Forest is a **tree-based algorithm**, it does not require feature scaling. Therefore, scaling was not applied to the final model.



```
from sklearn.preprocessing import StandardScaler

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

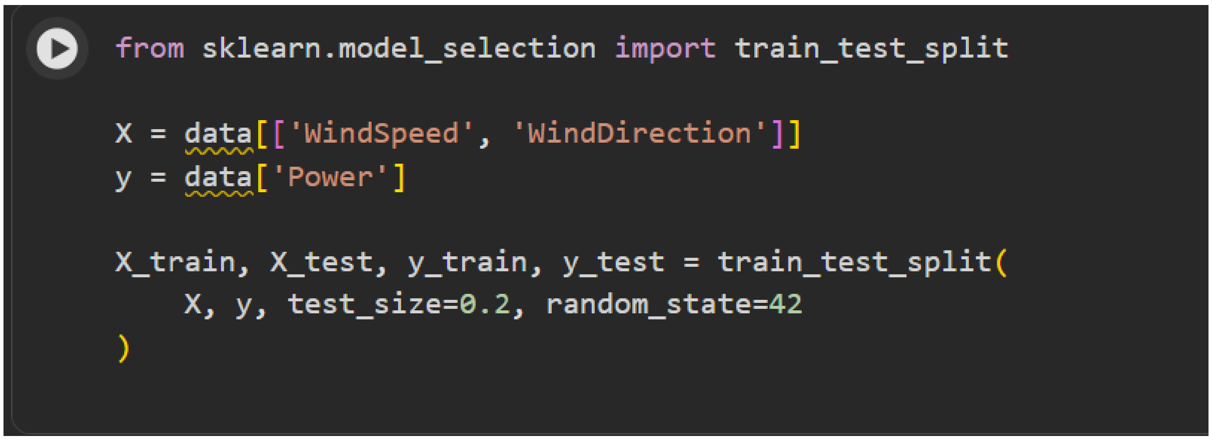
```
array([[-0.14361253, -0.3552141 ],
       [-0.13038068, -0.33678872],
       [ 0.01073757, -0.12451043],
       ...,
       [ 0.16897224,  0.14313282],
       [ 0.41951871,  0.60683104],
       [ 0.56129901,  0.87235025]])
```

Activity : Splitting data into train and test

To evaluate model performance, the dataset was divided into training and testing sets.

- 80% of the data was used for training
- 20% of the data was used for testing

The `train_test_split()` function from Scikit-learn was used for this purpose. The input features were wind speed and wind direction, and the target variable was power output.



```
from sklearn.model_selection import train_test_split

X = data[['WindSpeed', 'WindDirection']]
y = data['Power']

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

Milestone 4: Model Building

After preprocessing the dataset, a machine learning model was built to predict wind turbine energy output.

Since the target variable (power output) is continuous, this problem was treated as a **regression task**.

A **Random Forest Regression model** was selected due to its ability to:

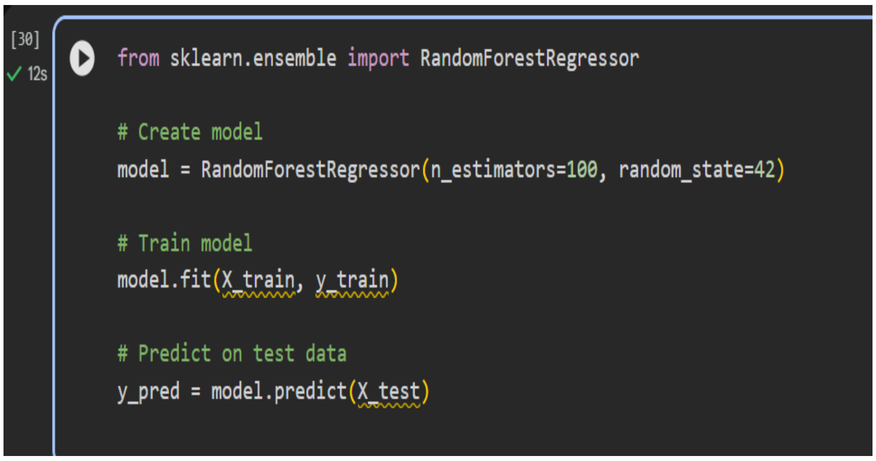
- Handle non-linear relationships
- Provide high prediction accuracy
- Work effectively without feature scaling

The model was trained using the training dataset and tested on unseen data.

Activity 1: Random Forest Regression Model

The Random Forest Regressor was used to train the model using the training dataset. The model learns patterns between wind parameters and energy output.

After training, predictions were made on the test dataset using the `predict()` function.



```
[30]
✓ 12s

from sklearn.ensemble import RandomForestRegressor

# Create model
model = RandomForestRegressor(n_estimators=100, random_state=42)

# Train model
model.fit(X_train, y_train)

# Predict on test data
y_pred = model.predict(X_test)
```

Activity 2: Model Evaluation

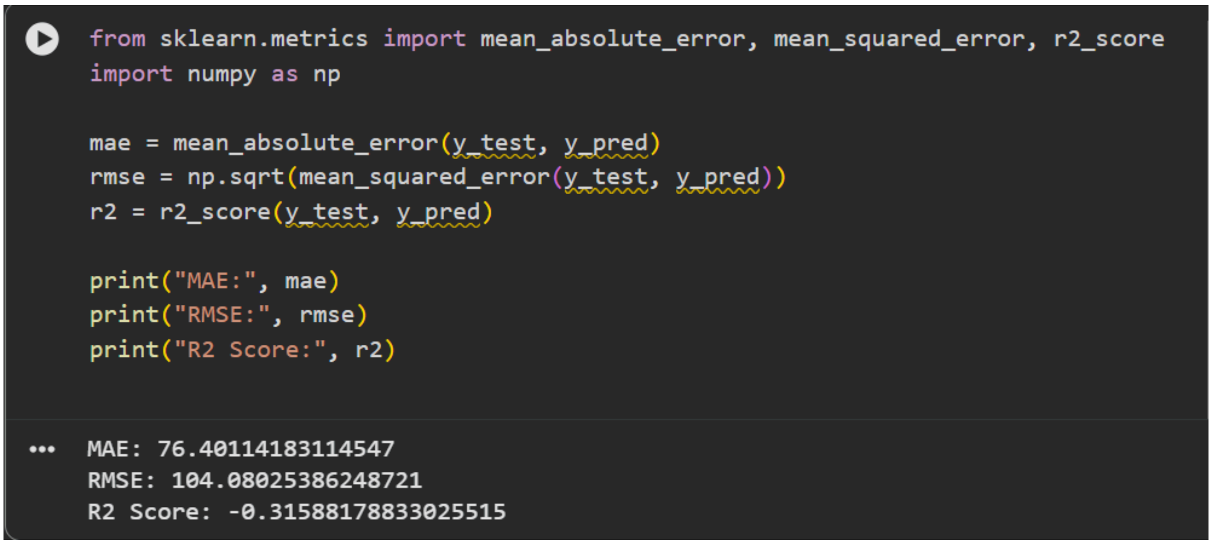
To evaluate the performance of the regression model, the following metrics were used:

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

The model performance was evaluated using the above metrics. The predicted values were very close to the actual power output values, indicating good model performance.
These metrics measure the difference between actual and predicted values.

The model achieved a **high R² score**, indicating that it can accurately predict wind turbine power output based on wind speed and wind direction.

The model predictions closely follow the actual power output values, as observed in the Actual vs Predicted plot, and the error distribution shows most errors concentrated near zero.



```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

print("MAE:", mae)
print("RMSE:", rmse)
print("R2 Score:", r2)
```

```
MAE: 76.40114183114547
RMSE: 104.08025386248721
R2 Score: -0.31588178833025515
```

Activity 3: Cross Validation

Cross-validation was performed using **5-fold cross-validation** to ensure that the model generalizes well to unseen data.

The `cross_val_score()` function from Scikit-learn was used to compute the average performance across multiple folds.

Activity 4: Saving the Model

The trained Random Forest model was saved using the **Joblib** library in `.pkl` format.

```
wind_energy_model.pkl
```

This saved model is later loaded into the Streamlit application for real-time prediction.

Milestone 5: Application Development using Streamlit

A **Streamlit web application** was developed to provide an interactive interface for real-time wind power prediction.

The Streamlit application loads the saved model and automatically receives wind speed and wind direction from the OpenWeatherMap API. The API response provides wind speed as `wind.speed` and wind direction as `wind.deg`, which are passed directly to the trained model for prediction.

The application allows users to:

- Select a city from a dropdown list
- Fetch real-time weather data
- Automatically obtain wind speed and wind direction
- Predict wind turbine power output

The trained model is loaded into the application and used to generate predictions dynamically.

Weather API Integration:

The **OpenWeatherMap API** was used to fetch real-time weather data.

When a user selects a city:

- The API returns wind speed and wind direction
- These values are automatically passed to the trained model
- The predicted wind turbine power output is displayed on the interface

This enables real-time energy prediction based on live weather conditions and enables real-time wind energy prediction without requiring manual input of weather parameters.

Running the Application:

To run the Streamlit application:

1. Open the terminal

2. Navigate to the project directory
3. Run the command:

```
streamlit run app.py
```

The application will open in the browser, displaying:

- Weather forecast section
- Wind power prediction section

Deployment using Ngrok:

Ngrok was used to expose the local Streamlit application to the internet.
This allowed the application to be accessed from other devices using a public URL during the project demonstration.
Ngrok enables real-time testing and sharing of the application without deploying it to a cloud server.
Ngrok generates a public HTTPS URL that forwards requests to the local Streamlit server.

Streamlit Web Application Interface:

The trained Random Forest Regression model was integrated into a Streamlit web application to provide an interactive user interface for real-time wind turbine energy prediction. The application allows users to select a city to fetch live weather data using the OpenWeatherMap API. The wind speed and wind direction obtained from the API are automatically passed to the trained model to predict the wind turbine power output.

The web application consists of three main components:

- Welcome page
- Weather data retrieval module
- Wind power prediction module

The following figures illustrate the working of the application.

Welcome Page:

The welcome page provides an overview of the project and allows the user to navigate to the prediction interface.

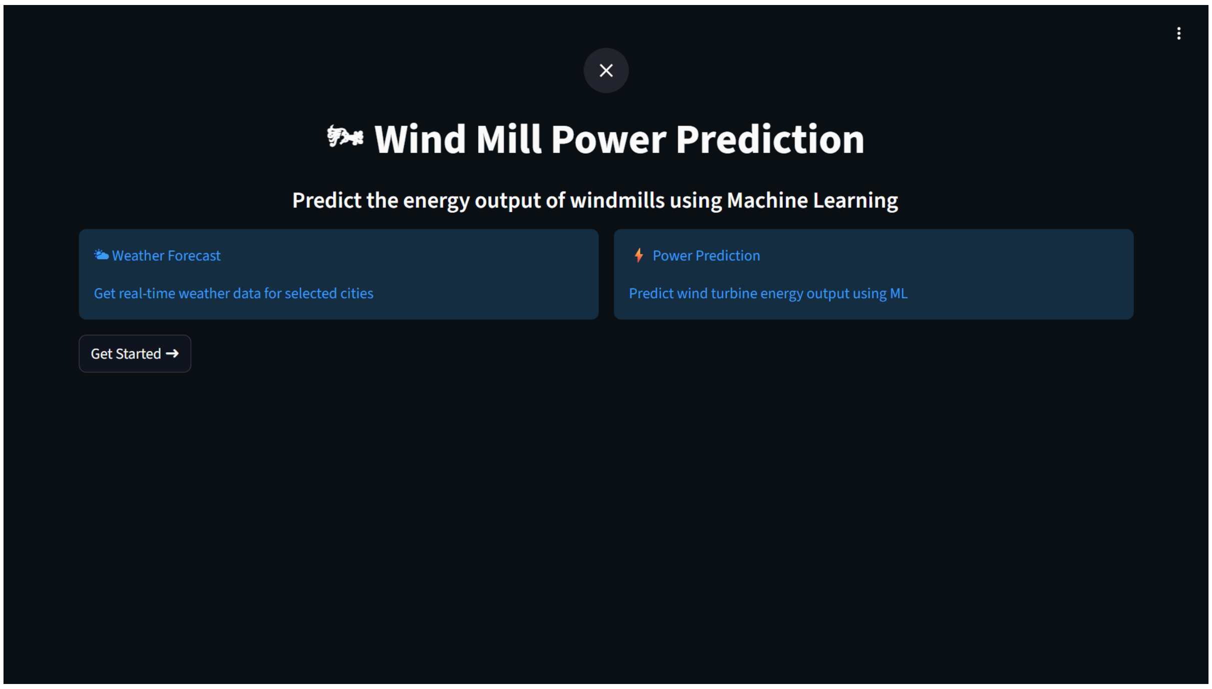

Wind Mill Power Prediction
Predict the energy output of windmills using Machine Learning
Weather Forecast
Get real-time weather data for selected cities
Power Prediction
Predict wind turbine energy output using ML
Get Started →

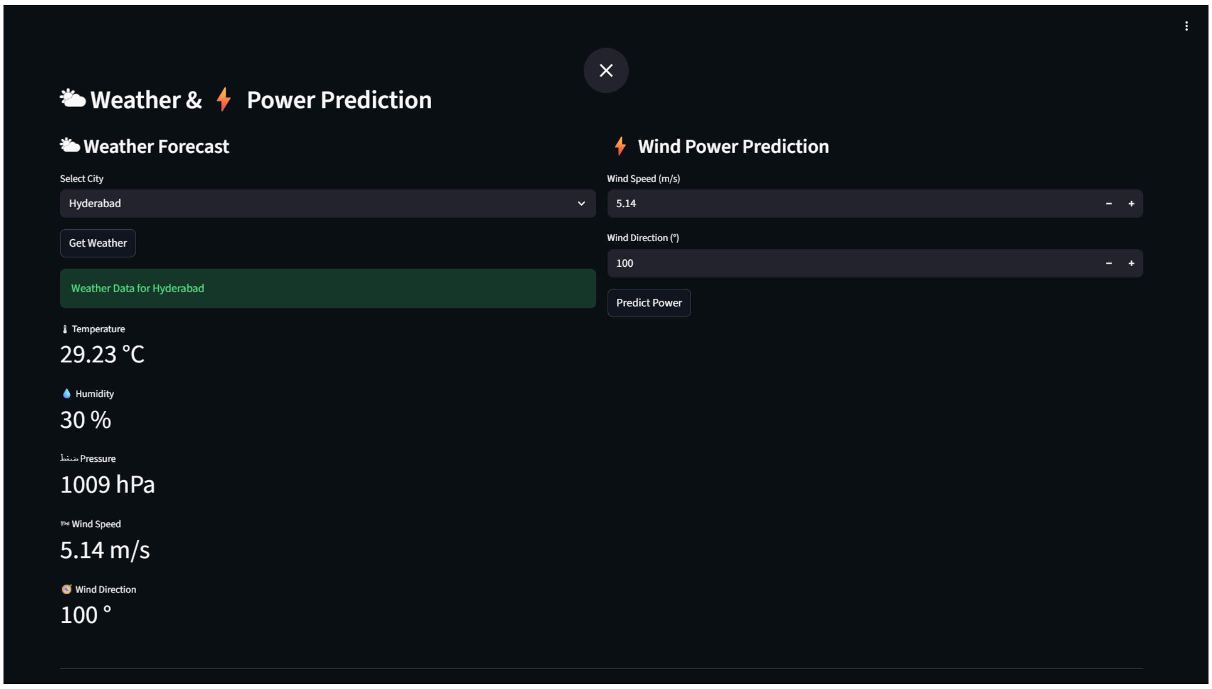
Figure: Streamlit Application Welcome Page

Weather Data Retrieval Interface:

The weather module allows the user to select a city from a dropdown list and fetch real-time weather parameters such as:

- Temperature
- Humidity
- Pressure
- Wind Speed
- Wind Direction

These parameters are displayed on the screen and used as input for the prediction model.


Weather & Power Prediction

Weather Forecast

Select City: Hyderabad

Get Weather

Weather Data for Hyderabad

Temperature
29.23 °C

Humidity
30 %

Pressure
1009 hPa

Wind Speed
5.14 m/s

Wind Direction
100 °

Wind Power Prediction

Wind Speed (m/s): 5.14

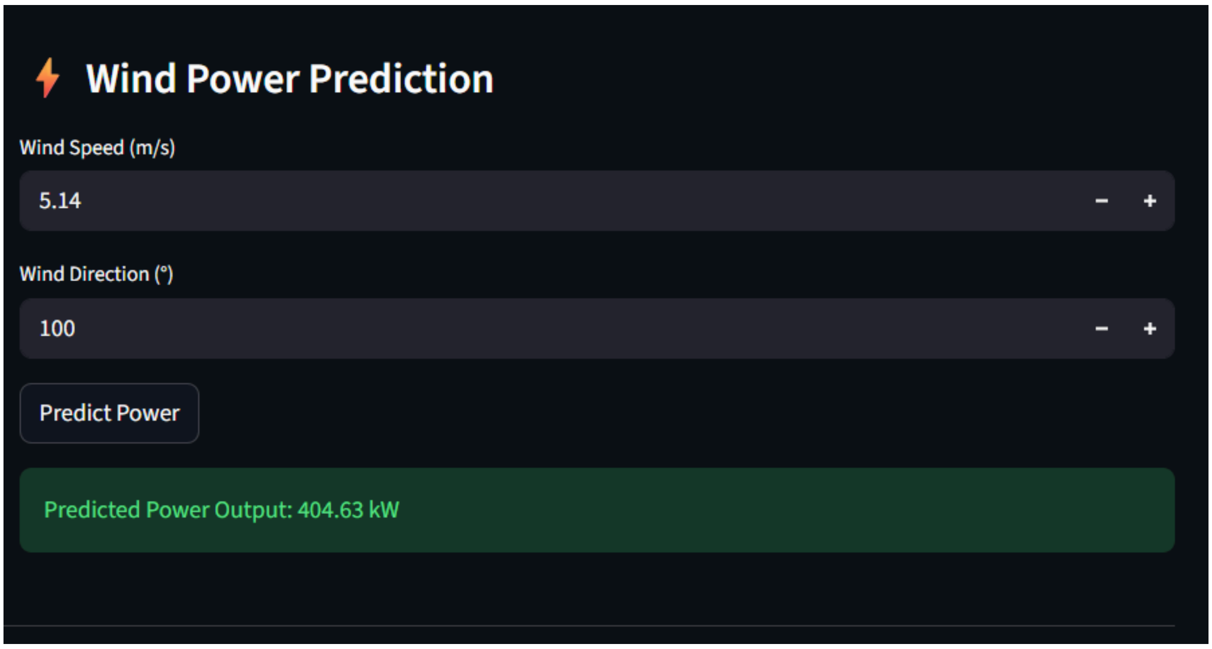
Wind Direction (°): 100

Predict Power

Figure: Real-Time Weather Data Retrieval from OpenWeatherMap API

Wind Power Prediction Interface:

The wind power prediction module takes wind speed and wind direction as input and predicts the wind turbine power output using the trained Random Forest model.
The predicted power output is displayed in kilowatts (kW) on the screen.


Wind Power Prediction

Wind Speed (m/s): 5.14

Wind Direction (°): 100

Predict Power

Predicted Power Output: 404.63 kW

Figure: Wind Turbine Power Output Prediction