Wind Energy Generation Prediction Using Machine Learning

Introduction

Project Title: Weather-Based Prediction of Wind Turbine Energy Output: A Next-Generation Approach to Renewable Energy Management

Team Details

Internship Program: SmartBridge Virtual Internship
Project Start Date: 29 January 2026
Team ID: LTVIP2026TMIDS84120
Team Size: 4
Team Leader: Pantham Naganjali
Team Members:

- Gurram Teja Sri
- Dadala Lakshman
- Steni Desabathula

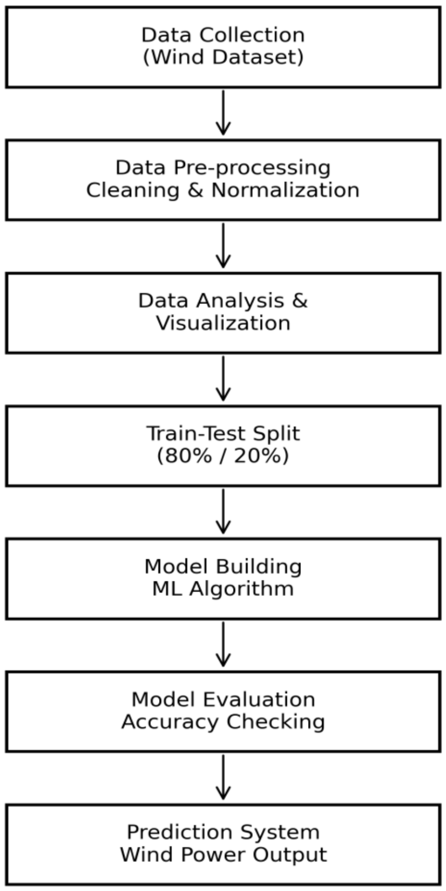
Project Description:

Renewable energy sources such as wind energy are playing a vital role in meeting the increasing energy demand of modern society. Wind power generation mainly depends upon environmental conditions such as wind speed, wind direction and atmospheric pressure. Due to the variability of wind conditions, accurate prediction of wind energy Generation becomes a challenging task.

Predicting the power generated from wind energy in advance helps in proper planning of energy distribution and reduces operational losses in power grids. Machine Learning techniques provide efficient methods to analyze historical wind data and predict future energy generation.

In this project, Machine Learning techniques are used to predict wind power generation based on environmental parameters. The dataset containing wind speed, wind direction and generated power is analyzed and processed. A prediction model is trained and tested using training and testing datasets to obtain accurate results.

Technical Architecture:


Data Collection (Wind Dataset)
Data Pre-processing Cleaning & Normalization
Data Analysis & Visualization
Train-Test Split (80% / 20%)
Model Building ML Algorithm
Model Evaluation Accuracy Checking
Prediction System Wind Power Output

Pre-Requisites:

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

Anaconda Navigator is an open-source software distribution used for package management and deployment of Python applications.

Python Packages:

- Open anaconda prompt as administrator
- Type “pip install numpy” and click enter.
- Type “pip install pandas” and click enter.
- Type “pip install scikit-learn” and click enter.
- Type ”pip install matplotlib” and click enter.

Prior Knowledge :

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

- **ML Concepts**
    - Supervised learning:

https://www.javatpoint.com/supervised-machine-learning

- Unsupervised learning:
  https://www.javatpoint.com/unsupervised-machine-learning

- Regression and classification

- Decision tree:
  https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

- Random forest:
  https://www.javatpoint.com/machine-learning-random-forest-algorithm

- KNN:
  https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

- Xgboost:
  https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/

- Evaluation metrics:
  https://www.analyticsvidhya.com/blog/2019/08/11- important-model-evaluation-error-metrics/

- **Flask Basics** : https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Objectives:

By the end of this project you will:

- Understand Machine Learning prediction concepts
- Gain knowledge about wind energy datasets
- Learn data preprocessing techniques
- Understand relationship between wind parameters and power generation
- Learn how to build prediction models

Project Flow:

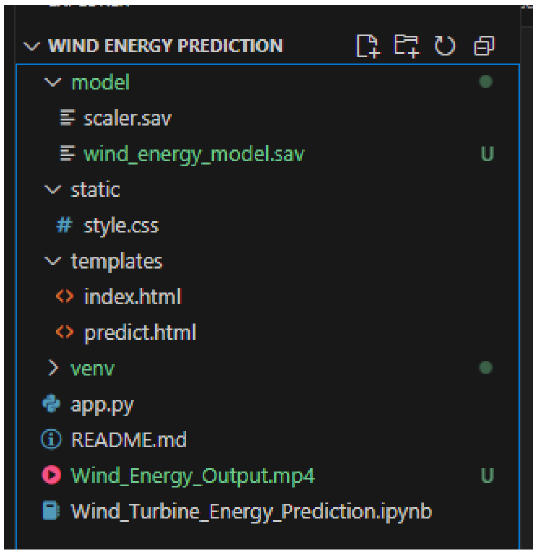
- User provides input dataset
- Dataset is analyzed
- Model is trained using historical wind data
- Model predicts the wind power generation

To accomplish this, we have to complete all the activities listed below,

- Data collection

- Collect the dataset or create the dataset
- Visualizing and analyzing data
- Univariate analysis
- Bivariate analysis
- Multivariate analysis
- Descriptive analysis
- Data pre-processing
- Checking for null values
- Handling outlier
- Handling categorical data
- Splitting data into train and test
- Model building
- Import the model building libraries
- Initializing the model
- Training and testing the model
- Evaluating performance of model
- Save the model
- Application Building
- Create an HTML file
- Build python code

Project Structure:
Create the Project folder which contains files as shown below


```
WIND ENERGY PREDICTION
  model
    scaler.sav
    wind_energy_model.sav        U
  static
    style.css
  templates
    index.html
    predict.html
  venv
  app.py
  README.md
  Wind_Energy_Output.mp4         U
  Wind_Turbine_Energy_Prediction.ipynb
```

- The **templates** folder contains all the HTML files such as *home.html*, *predict.html* which are used to build the user interface of the Wind Energy Prediction web application.
- The **Training** folder includes the dataset file (*dataset.csv*) and the Python script (*Wind_Turbine_Energy_Prediction.ipynb*) which are used for data preprocessing, training and testing the Machine Learning model.
- The **wind_energy_model.sav** file stores the trained Machine Learning model which is used by the application to predict wind power output based on user input parameters.
- The **app.py** file acts as the backend of the Flask application which connects the user interface with the trained model and displays the prediction results.

Milestone 1: Data Collection

Machine Learning depends heavily on data. Dataset can be downloaded from open-source platforms or created manually.

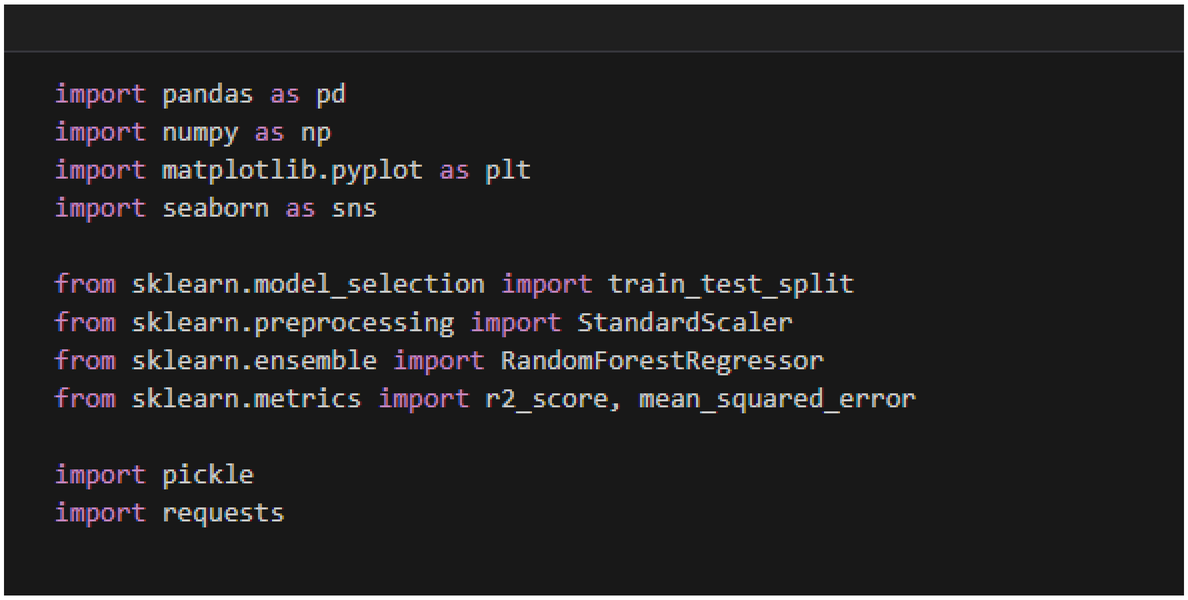
Link:

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

Milestone 2: Visualizing and Analyzing the Data

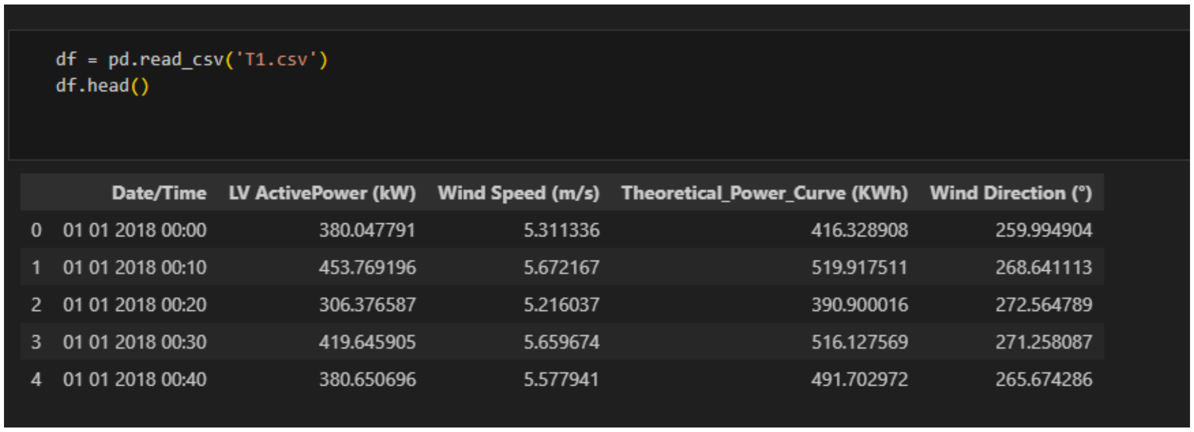
Activity 1: Importing Libraries

Required libraries such as NumPy, Pandas and Matplotlib are imported for dataset analysis.



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

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error

import pickle
import requests
```

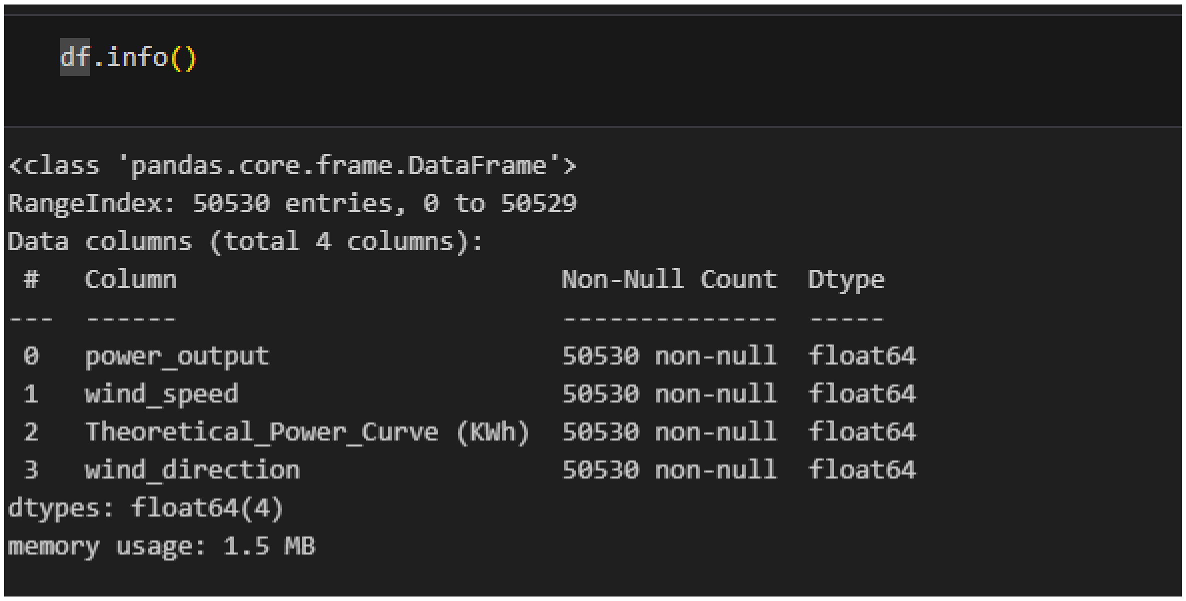
Activity 2: Reading the Dataset
Dataset is read using pandas from CSV, Excel or Text files.



```python
df = pd.read_csv('T1.csv')
df.head()
```

	Date/Time	LV ActivePower (kW)	Wind Speed (m/s)	Theoretical_Power_Curve (KWh)	Wind Direction (°)
0	01 01 2018 00:00	380.047791	5.311336	416.328908	259.994904
1	01 01 2018 00:10	453.769196	5.672167	519.917511	268.641113
2	01 01 2018 00:20	306.376587	5.216037	390.900016	272.564789
3	01 01 2018 00:30	419.645905	5.659674	516.127569	271.258087
4	01 01 2018 00:40	380.650696	5.577941	491.702972	265.674286

Activity 3: Dataset Analysis



```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50530 entries, 0 to 50529
Data columns (total 4 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   power_output                   50530 non-null  float64
 1   wind_speed                     50530 non-null  float64
 2   Theoretical_Power_Curve (KWh)  50530 non-null  float64
 3   wind_direction                 50530 non-null  float64
dtypes: float64(4)
memory usage: 1.5 MB
```

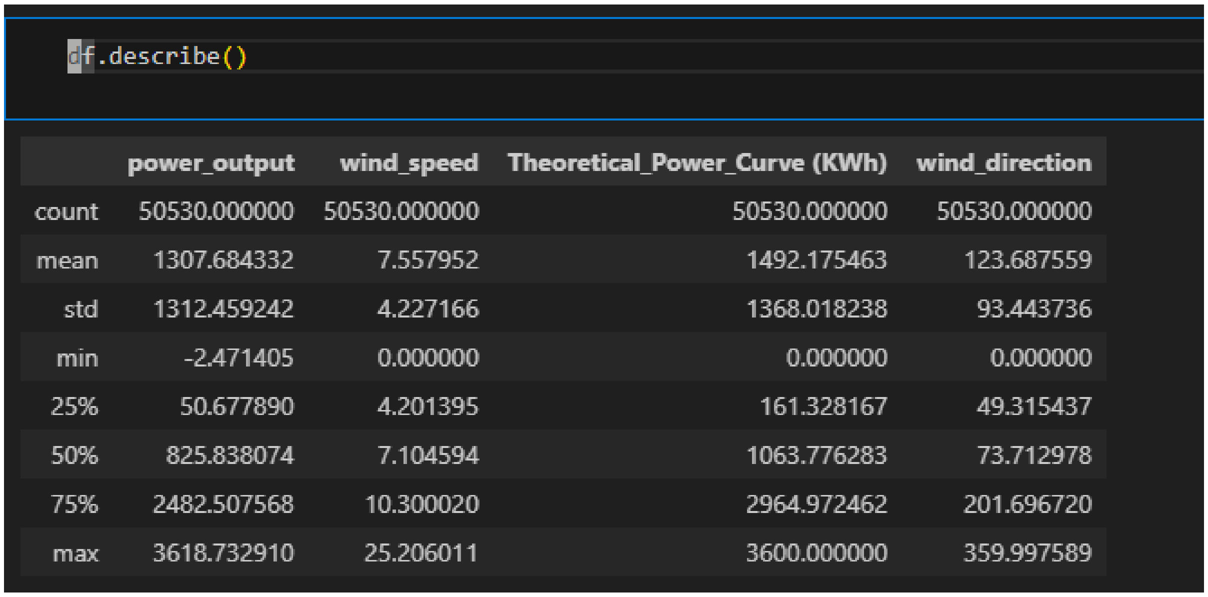
Dataset is analyzed to understand:
Wind speed variation
Wind direction
Power generation
Relationship between wind parameters and generated power is identified using visualization techniques.

Milestone 3: Data Pre-processing

Collected dataset is cleaned before training the model.
Steps involved:
Handling Missing Values
Handling Categorical Data
Normalization
Feature Selection
Splitting Dataset

Activity 1: Checking for Null Values

Dataset shape and data types are checked using:
df.shape df.info() df.isnull().sum()

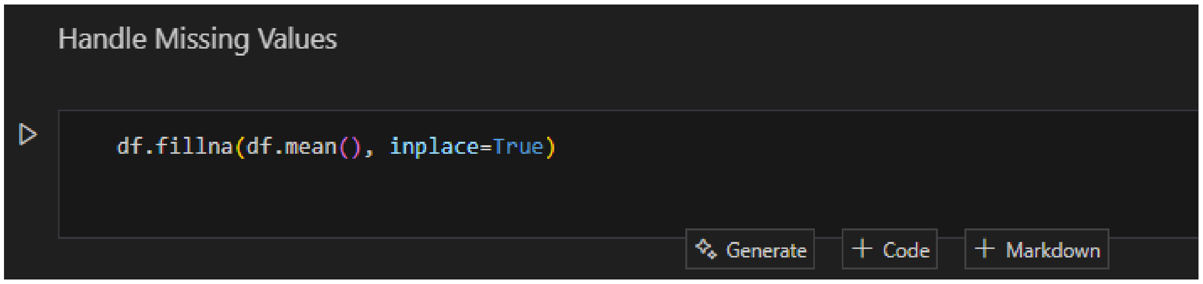


```
df.describe()
```

	power_output	wind_speed	Theoretical_Power_Curve (KWh)	wind_direction
count	50530.000000	50530.000000	50530.000000	50530.000000
mean	1307.684332	7.557952	1492.175463	123.687559
std	1312.459242	4.227166	1368.018238	93.443736
min	-2.471405	0.000000	0.000000	0.000000
25%	50.677890	4.201395	161.328167	49.315437
50%	825.838074	7.104594	1063.776283	73.712978
75%	2482.507568	10.300020	2964.972462	201.696720
max	3618.732910	25.206011	3600.000000	359.997589

Activity 2: Handling Categorical Values

Categorical values are converted into numerical format for model training.



Handle Missing Values

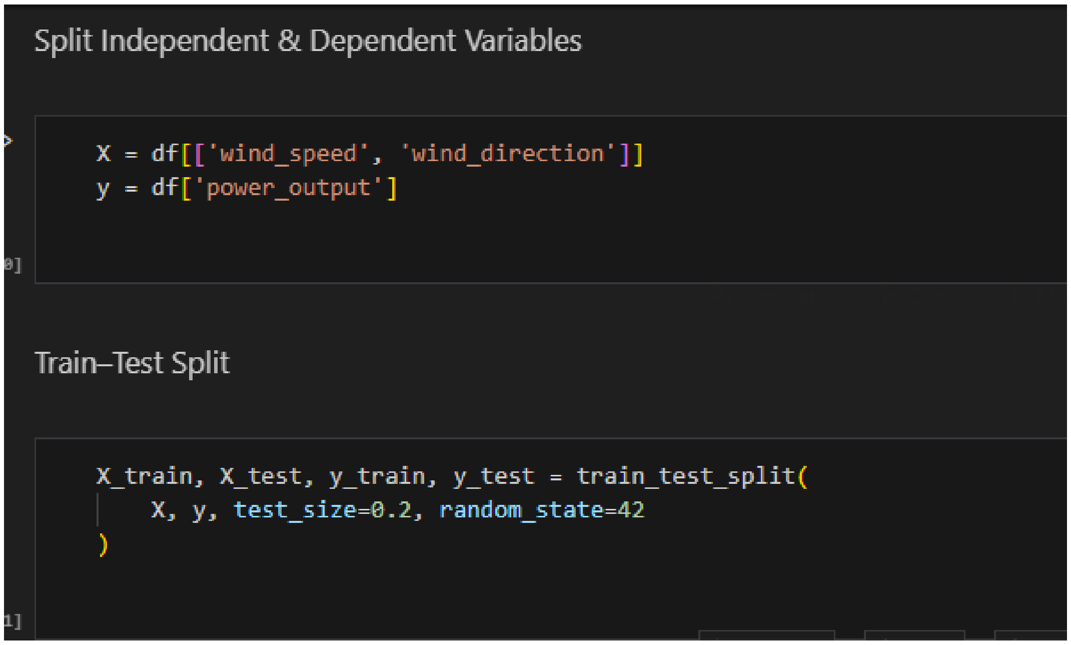
```
df.fillna(df.mean(), inplace=True)
```

Activity 3: Splitting Dataset

Datasetis split into:

Independent Variables (X)

Dependent Variable (Y)



Split Independent & Dependent Variables

```
X = df[['wind_speed', 'wind_direction']]
y = df['power_output']
```

Train–Test Split

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

Training and Testing datasets are created:

80% Training Data

20% Testing Data

Milestone 4: Model Building

After preprocessing, the dataset is trained using Machine Learning models.

Steps involved:

Import Model Libraries

Initialize Model

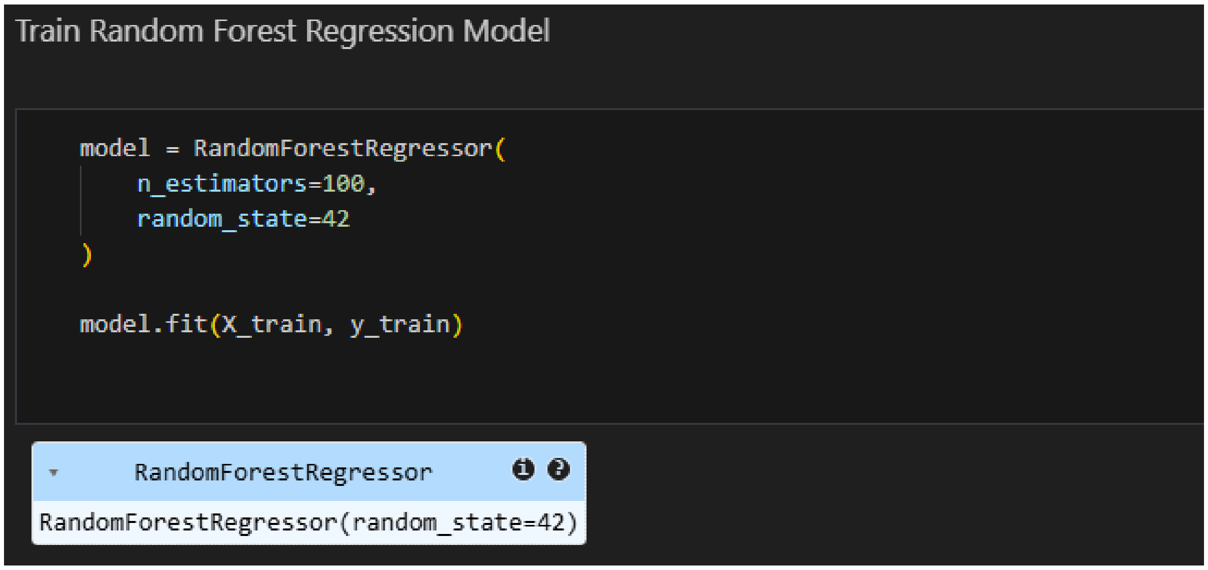
Train Model

Test Model

Evaluate Performance

Activity 1: Random forest model

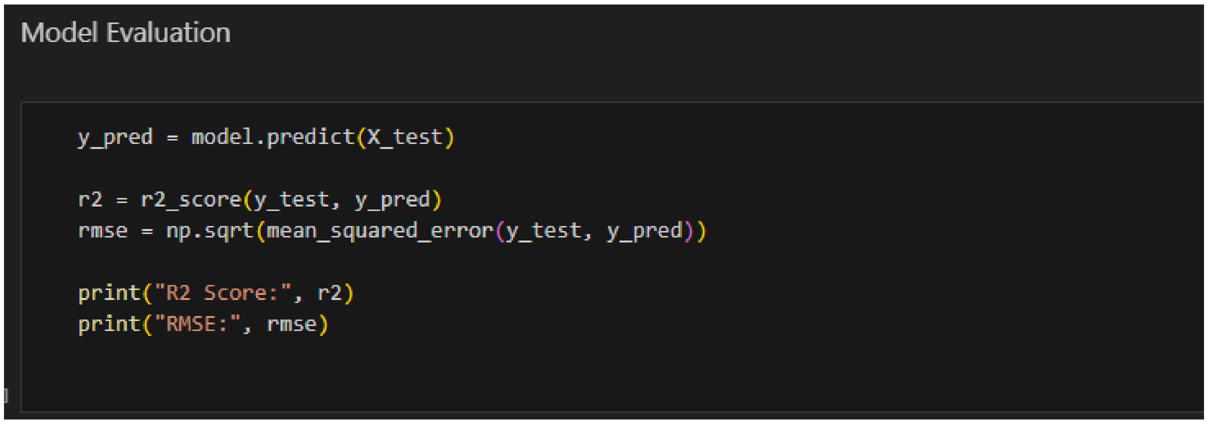
A function named randomForest is created and train and test data are passed as the parameters. Inside the function, RandomForestClassifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.


Train Random Forest Regression Model

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

model.fit(X_train, y_train)
```

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

Model Evaluation
Model performance is evaluated using:
Training Accuracy
Testing Accuracy


Model Evaluation

```
y_pred = model.predict(X_test)

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

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

The trained model is used to predict wind energy generation.

Milestone 5: Application Building

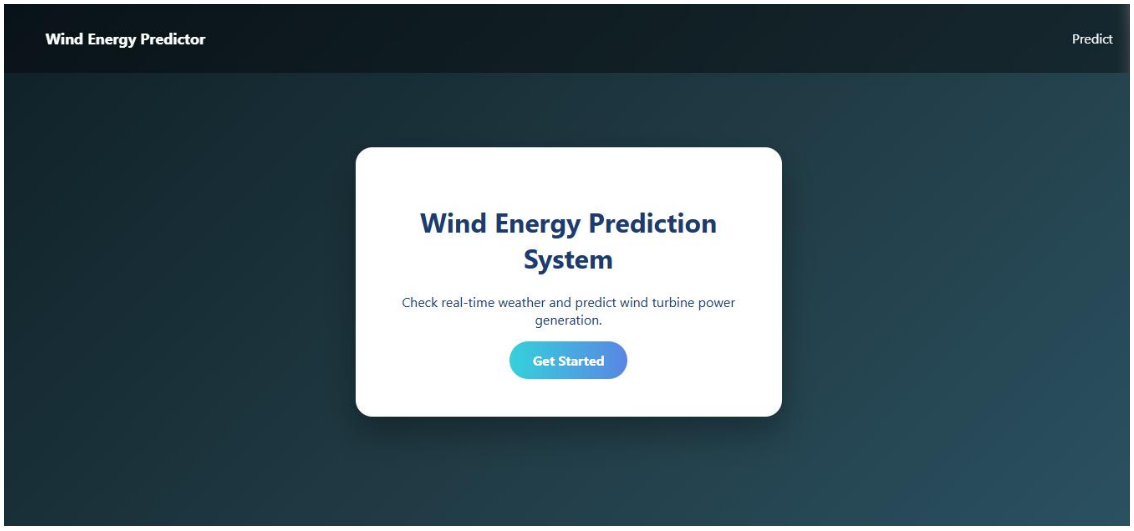
In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.
This section has the following tasks

- Building HTML Pages
- Building serverside script

Activity1: Building Html Pages:
For this project create three HTML files namely

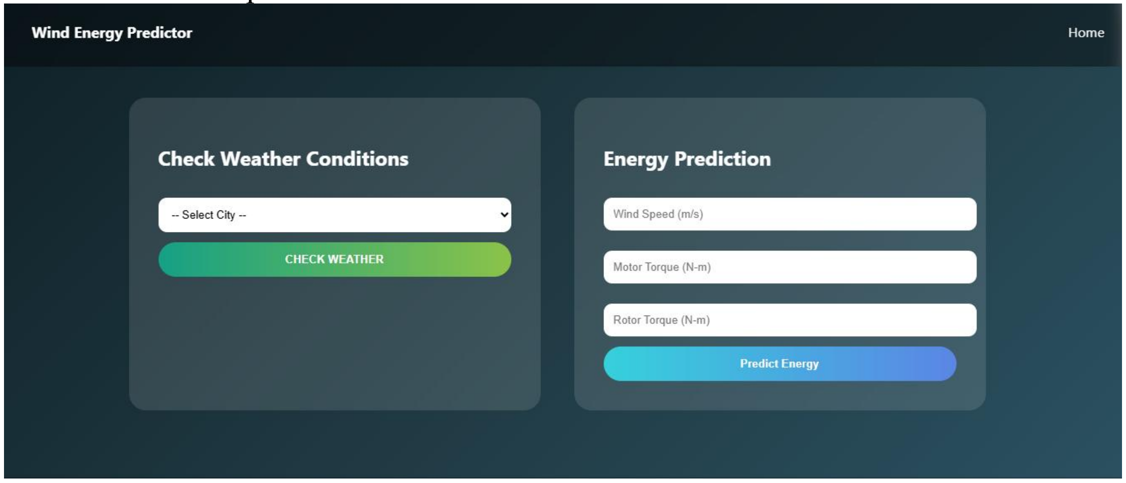
- home.html
- predict.html

and save them in templates folder.
Let's see how our home.html page looks like:


Wind Energy Predictor
Predict
Wind Energy Prediction System
Check real-time weather and predict wind turbine power generation.
Get Started

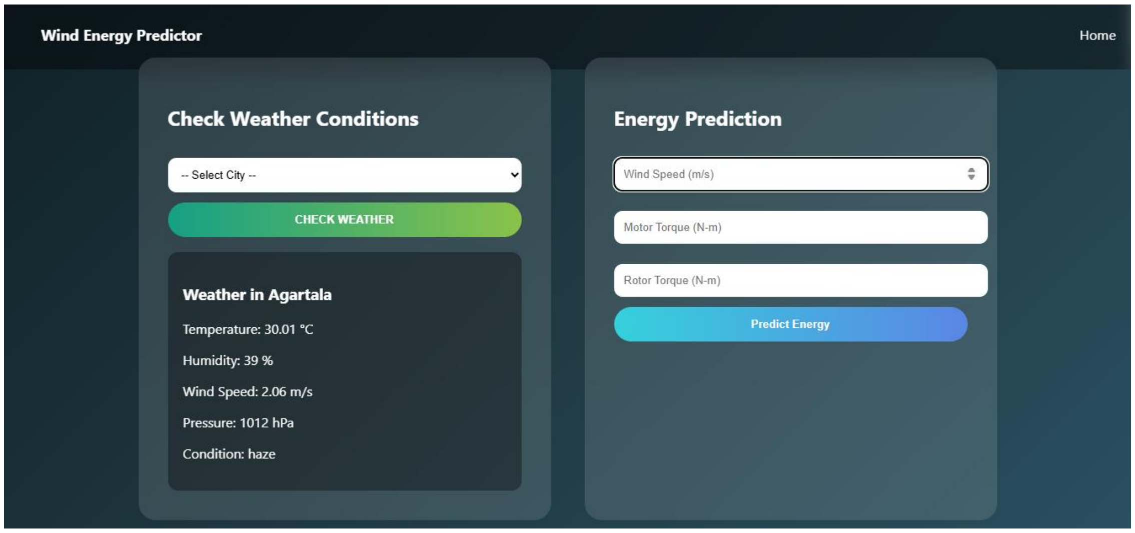
Now when you click on predict button from top right corner you will get redirected to predict.html

Lets look how our predict.html file looks like:



Now when you click on predict button from left bottom corner you will get the prediction of the energy

Lets look how our file looks like:



Weather in Agartala

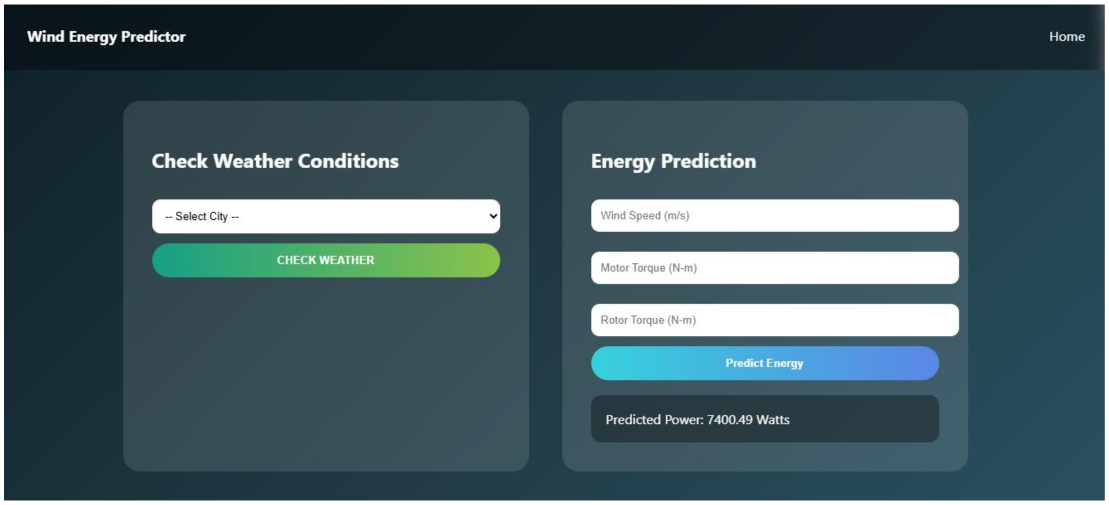
Temperature: 30.01 °C

Humidity: 39 %

Wind Speed: 2.06 m/s

Pressure: 1012 hPa

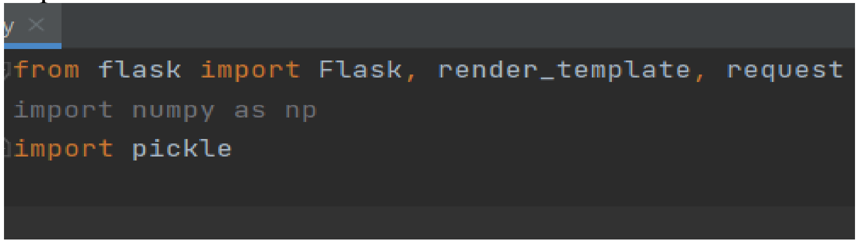
Condition: haze



Predicted Power: 7400.49 Watts

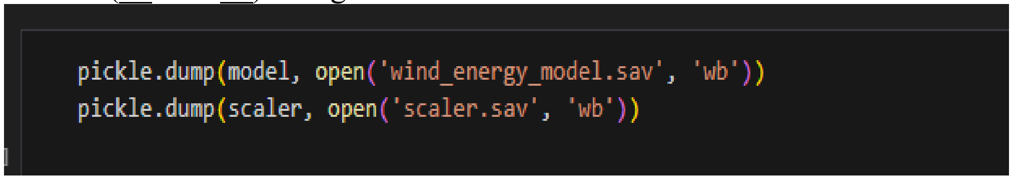
Activity 2: Build Python code:

Import the libraries



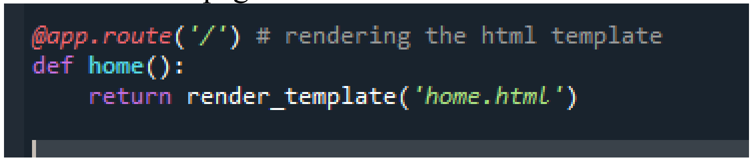
```
from flask import Flask, render_template, request
import numpy as np
import pickle
```

Load the saved model. Importing flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument.



```
    pickle.dump(model, open('wind_energy_model.sav', 'wb'))
    pickle.dump(scaler, open('scaler.sav', 'wb'))
```

Render HTML page:

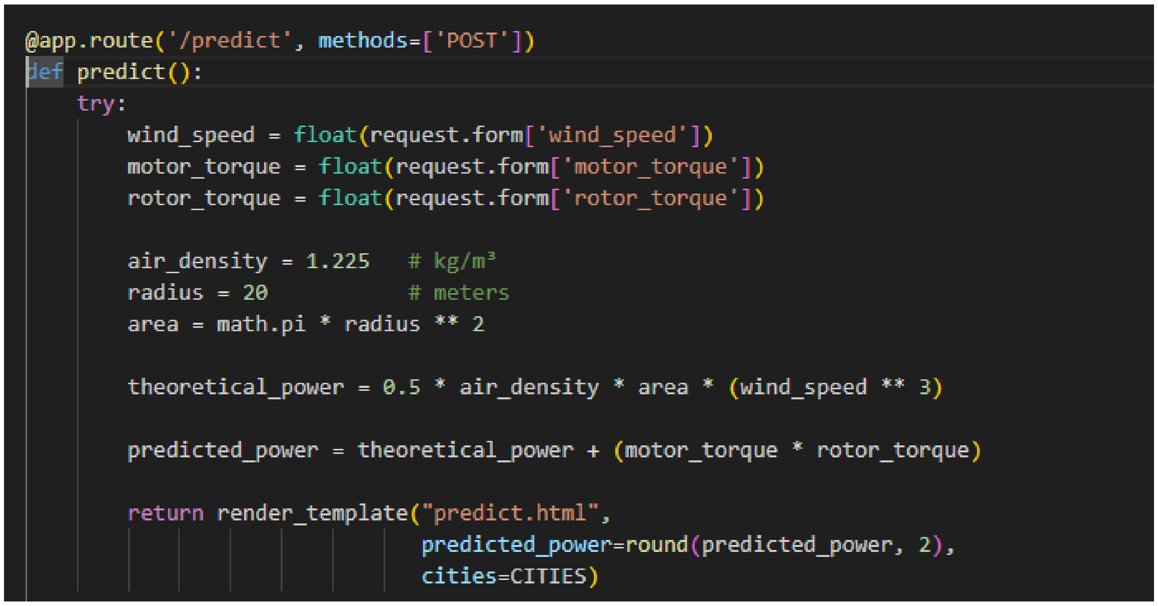


```
@app.route('/') # rendering the html template
def home():
    return render_template('home.html')
```

Here we will be using declared constructor to route to the HTML page which we have created earlier.

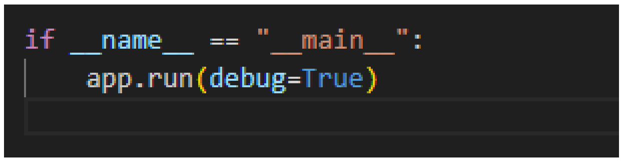
In the above example, '/' URL is bound with home.html function. Hence, when the home page of the web server is opened in browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:



```python
@app.route('/predict', methods=['POST'])
def predict():
    try:
        wind_speed = float(request.form['wind_speed'])
        motor_torque = float(request.form['motor_torque'])
        rotor_torque = float(request.form['rotor_torque'])

        air_density = 1.225   # kg/m³
        radius = 20           # meters
        area = math.pi * radius ** 2

        theoretical_power = 0.5 * air_density * area * (wind_speed ** 3)

        predicted_power = theoretical_power + (motor_torque * rotor_torque)

        return render_template("predict.html",
                               predicted_power=round(predicted_power, 2),
                               cities=CITIES)
```

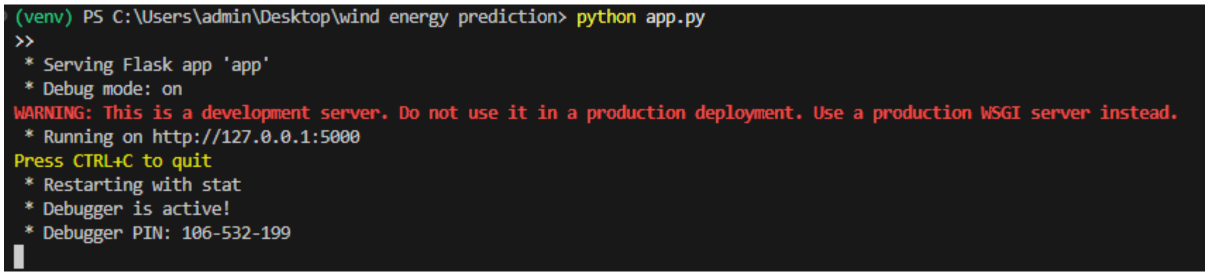
Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will rendered to the text that we have mentioned in the submit.html page earlier.
Main Function:



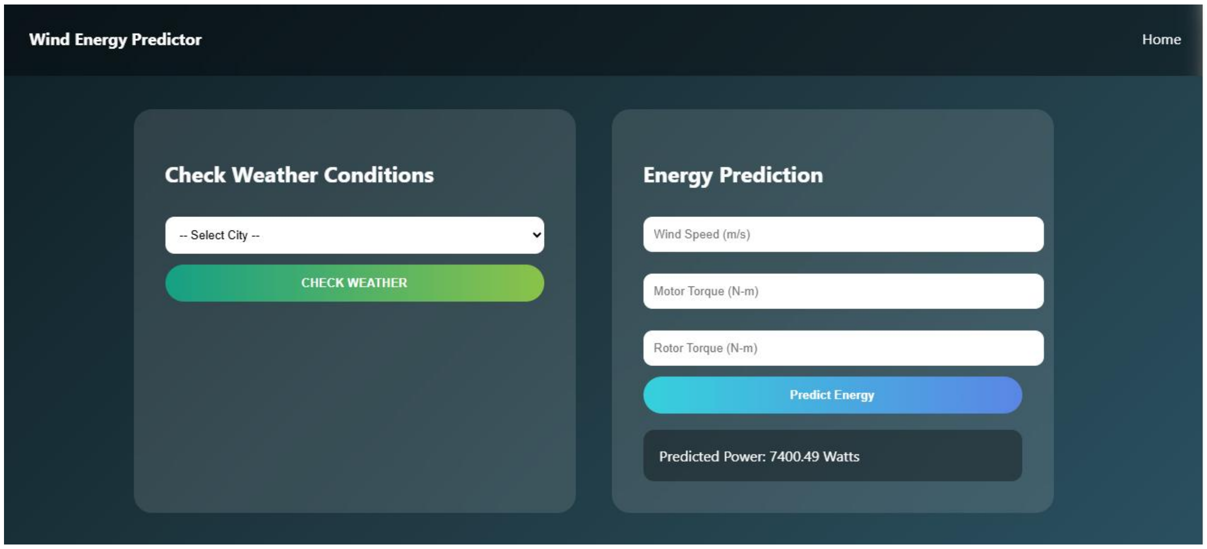
```python
if __name__ == "__main__":
    app.run(debug=True)
```

Activity 3: Run the application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.



```
(venv) PS C:\Users\admin\Desktop\wind energy prediction> python app.py
>>
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 106-532-199
```



Predicted Power: 7400.49 Watts

Conclusion:

Machine Learning techniques help in predicting wind energy generation effectively using historical environmental data. This prediction helps in efficient energy planning and management in renewable energy systems.