Predicting The Energy Output Of Wind Turbine Based On Weather Condition

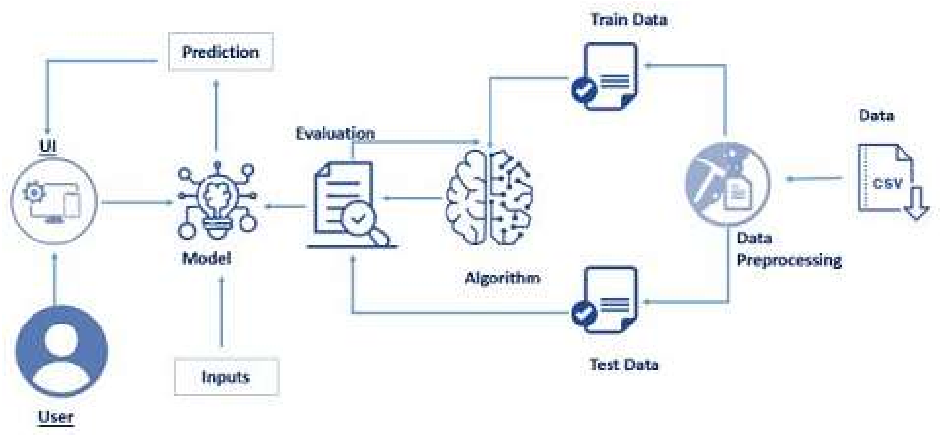
Department Of Computer Science And Engineering
Chalapathi Institute Of Engineering & Technology

Team Members:
1. **Gowri Bhavani.Bomma Reddy**
2. **Pratyusha.Gurram**
3. **Akhila.Dulipalla**
4. **Sandhya.Kanchupati**

Introduction:

Wind power generation differs from conventional thermal generation due to the stochastic nature of wind. Thus wind power forecasting plays a key role in dealing with the challenges of balancing supply and demand in any electricity system, given the uncertainty associated with the wind farm power output. Accurate wind power forecasting reduces the need for additional balancing energy and reserve power to integrate wind power. For a wind farm that converts wind energy into electricity power, a real-time prediction system of the output power is significant. In this guided project , a prediction system is developed with a method of combining statistical models and physical models. In this system, the inlet condition of the wind farm is forecasted by the auto regressive model.

Architecture:



Environment and Tools:

1.Numpy

2.Pandas

3.Matplotlib

4.Joblib

5.Seaborn

Dataset:

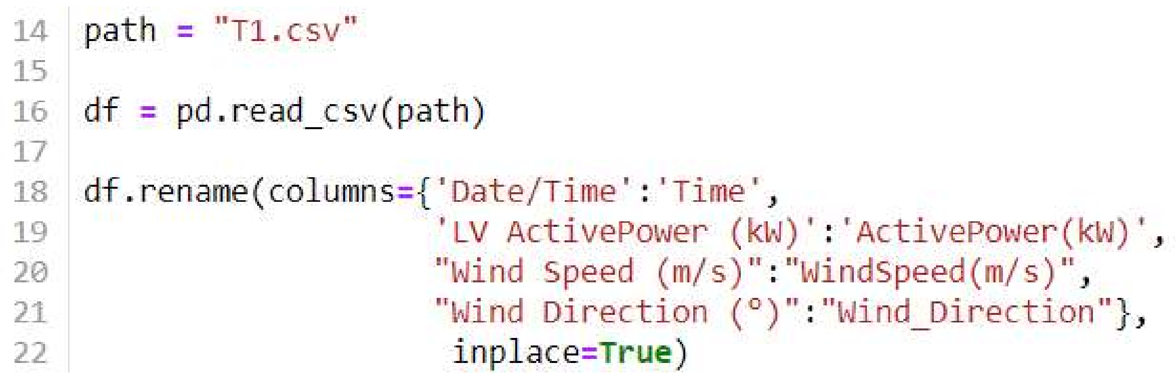
he dataset for wind energy prediction is to be collected. The dataset which is considered here will have the environmental conditions.

Data Preprocessing:

1. Processing the dataset.
2. Handling the null values.
3. Handling the categorical values if any.
4. Normalize the data if required.

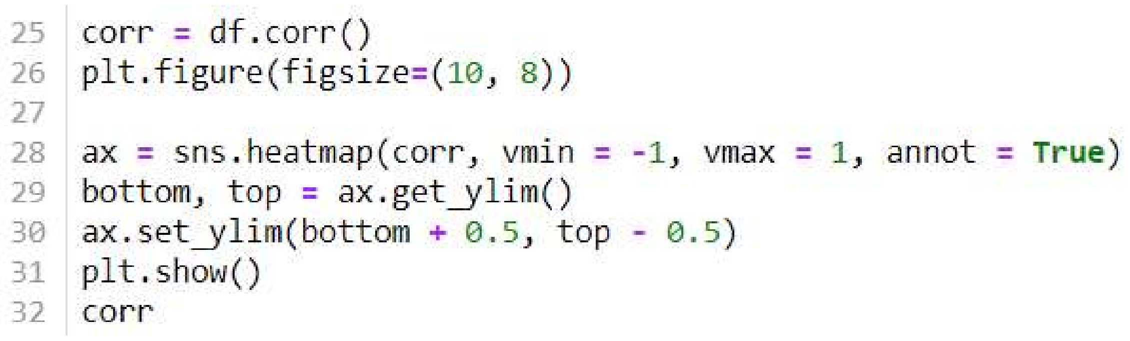
5. Identify the dependent and independent variables.
6. Split the dataset into train and test sets.

1.Analyze The Datasets:

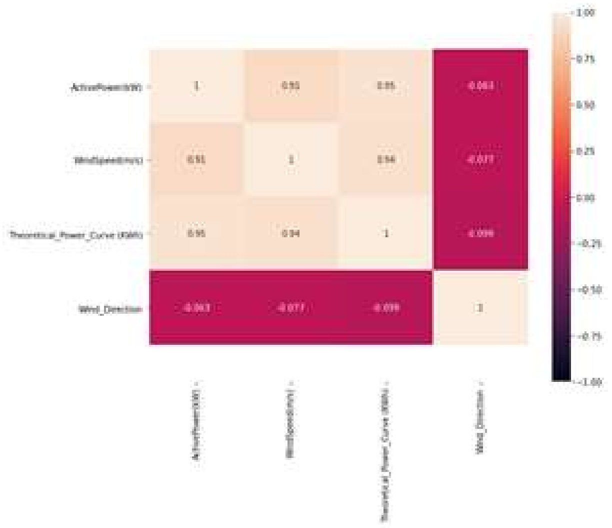


```
path = "T1.csv"

df = pd.read_csv(path)

df.rename(columns={'Date/Time':'Time',
                   'LV ActivePower (kW)':'ActivePower(kW)',
                   "Wind Speed (m/s)":"WindSpeed(m/s)",
                   "Wind Direction (°)":"Wind_Direction"},
                    inplace=True)
```

2.Check the correlation between the columns for dimensionality reduction:



```
corr = df.corr()
plt.figure(figsize=(10, 8))

ax = sns.heatmap(corr, vmin = -1, vmax = 1, annot = True)
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
plt.show()
corr
```



The heat map clearly tells us that there's no relation between wind direction and the Power generated but Wind speed, Theoretical power and Actual power generated to have a very positive correlation.

3.Drop the unnecessary columns if any

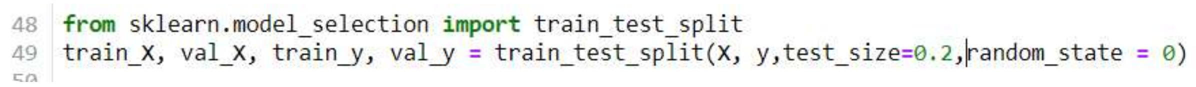
```python
df.drop(['Wind_Direction'],axis=1,inplace = True)
```

Splitting The Datasets

In this activity, the dependent and independent variables are to be identified. The independent columns are considered as x and the dependent column as y.

```python
y = df['ActivePower(kW)'] #'Theoretical_Power_Curve (KWh)'
X = df[['Theoretical_Power_Curve (KWh)','WindSpeed(m/s)']]#'ActivePower(kW)'
```

After identifying the dependent and independent variables, the dataset now has to be split into two sets, one set is used for training the model and the second set is used for testing how good the model is built. The split ratio we consider is 80% for training and 20% for testing.

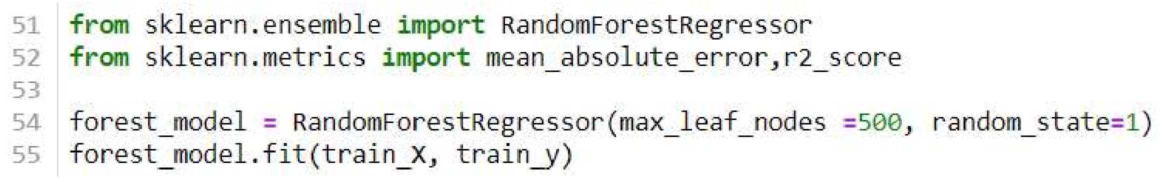


```python
from sklearn.model_selection import train_test_split
train_X, val_X, train_y, val_y = train_test_split(X, y,test_size=0.2,random_state = 0)
```

Model Building:

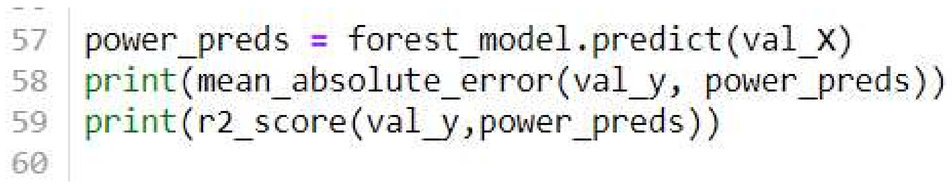
- Linear Regression
- Random Forest Regression / Classification
- Decision Tree Regression / Classification

We will be considering the Random Forest Regressor model and fit the data.



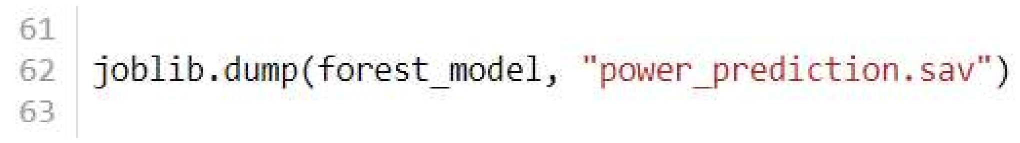
```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error,r2_score

forest_model = RandomForestRegressor(max_leaf_nodes =500, random_state=1)
forest_model.fit(train_X, train_y)
```

Check The Metrics Of The Model



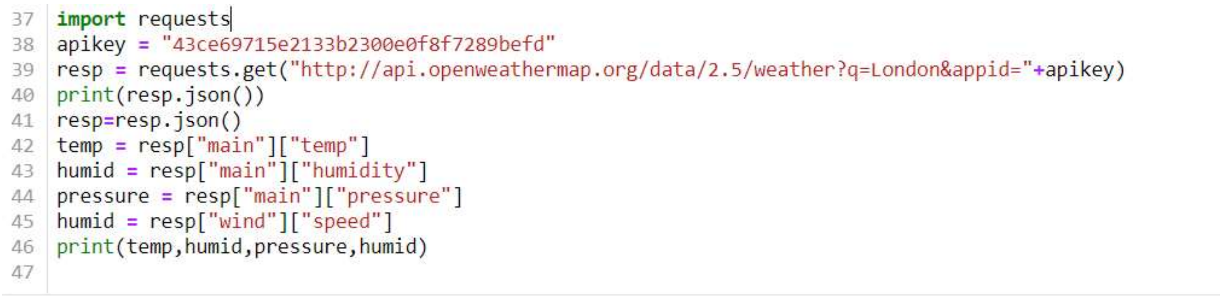
```python
power_preds = forest_model.predict(val_X)
print(mean_absolute_error(val_y, power_preds))
print(r2_score(val_y,power_preds))
```

Save The Model:



```python

joblib.dump(forest_model, "power_prediction.sav")

```

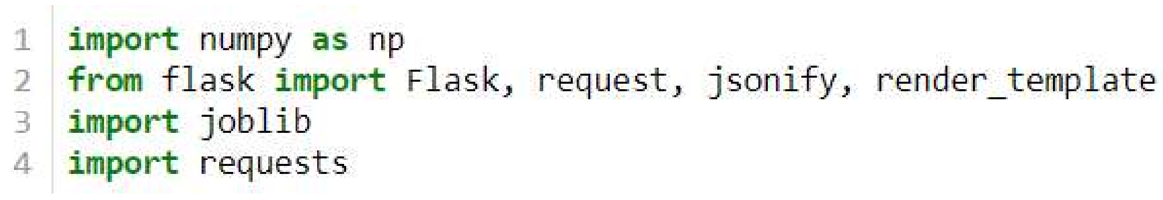
API Integration:



```python
import requests
apikey = "43ce69715e2133b2300e0f8f7289befd"
resp = requests.get("http://api.openweathermap.org/data/2.5/weather?q=London&appid="+apikey)
print(resp.json())
resp=resp.json()
temp = resp["main"]["temp"]
humid = resp["main"]["humidity"]
pressure = resp["main"]["pressure"]
humid = resp["wind"]["speed"]
print(temp,humid,pressure,humid)

```

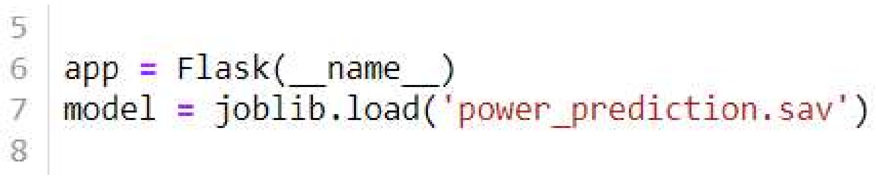
Build The Python

Step 1: Import required libraries Flask App:



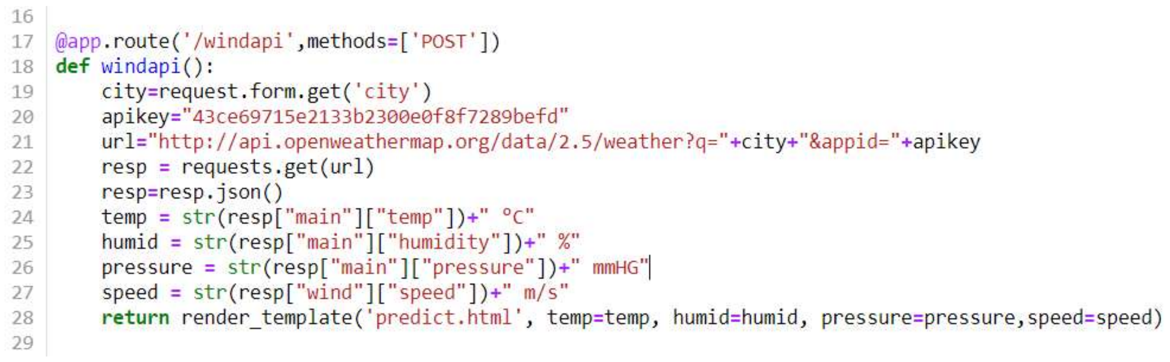
```python
import numpy as np
from flask import Flask, request, jsonify, render_template
import joblib
import requests
```

Step 2: Load the model and initialise flask app



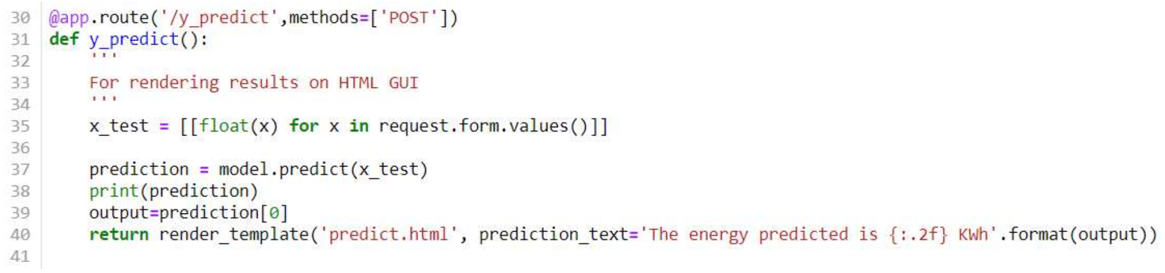
```python

app = Flask(__name__)
model = joblib.load('power_prediction.sav')

```

step3: Configure app.py for api request



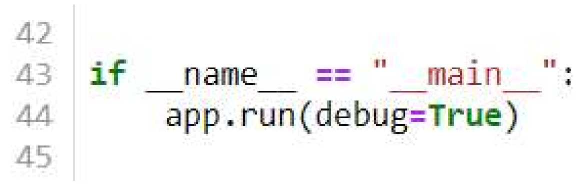
```python

@app.route('/windapi',methods=['POST'])
def windapi():
    city=request.form.get('city')
    apikey="43ce69715e2133b2300e0f8f7289befd"
    url="http://api.openweathermap.org/data/2.5/weather?q="+city+"&appid="+apikey
    resp = requests.get(url)
    resp=resp.json()
    temp = str(resp["main"]["temp"])+" °C"
    humid = str(resp["main"]["humidity"])+" %"
    pressure = str(resp["main"]["pressure"])+" mmHG"
    speed = str(resp["wind"]["speed"])+" m/s"
    return render_template('predict.html', temp=temp, humid=humid, pressure=pressure,speed=speed)

```

Step 4: Configure the file with predictions



```python
@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))

```

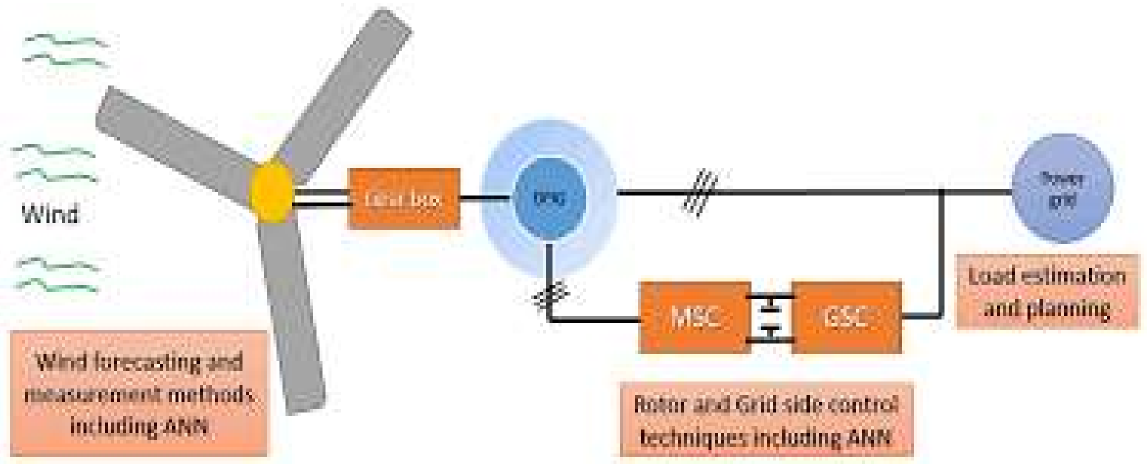
Step 5: Run the app



```python

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

```

Build An HTML Page

Execute And Test Your Model


Wind
Gear box
DFIG
Power grid
MSC
GSC
Load estimation and planning
Wind forecasting and measurement methods including ANN
Rotor and Grid side control techniques including ANN

5.Conclusion:

In this study, we showed that wind energy output can be predicted from publicly available weather data with accuracy up to 80% R2 on the training range and up to 85, 5% on the unseen test data. We identified the smallest space of input variables (windGust2 and dewPoint) where reported accuracy can be achieved, and provided clear trade-offs in prediction accuracy when decreasing the input space to the windGust2 variable. We demonstrated that an off-the-shelf data modeling and variable selection tool can be used with mostly default settings to run the symbolic regression experiments as well as variable importance, variable contribution analysis, ensemble selection, and validation.