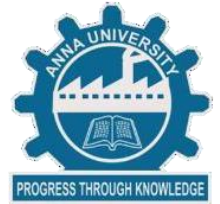


# PREDICTING THE ENERGY OUTPUT OF WIND TURBINE BASED ON WEATHER CONDITION



## IBMNALAIYATHIRANREPORT

*Submitted by*

**S.THAMIZHARASI - 814619106004**

**A.KAVIYA -814619106001**

**K.KAVIYA -814619106002**

**A.VANAJA-814619106302**

Date	18 November 2022
Team ID	<b>PNT2022TMID045948</b>
Project Name	<b>Predicting the output energy of wind turbine based on weather condition</b>
Maximum Marks	8 Marks



## TRICHY ENGINEERING COLLEGE

**TRICHY – 621 132**

## LIST OF CONTENTS:

S.NO	TITLE	PAGE NO
1.	INTRODUCTION	3
1.1	PROJECT OVERVIEW	3
1.2	PURPOSE	4
2.	LITERATURE SURVUY	4
2.1	EXISTING PROBLEM	4
2.2	REFERENCE	5
2.3	PROBLEM STATEMENT DEFINITION	6
3.	IDEATION & PROPOSED SOLUTION	6
3.1	EMPATHY MAP CANVAS	6
3.2	IDEATION & BRAINSTORMING	6
3.3	PROPOSED SOLUTION	7
3.4	PROBLEM SOLUTION FIT	8
4.	REQUIREMENT ANALYSIS	9
4.1	FUNCTIONAL REQUIREMENTS	9
4.2	NON-FUNCTIONAL REQUIREMENTS	10
5.	PROJECT DESIGN	11
5.1	DATA FLOW DIAGRAMS	11
5.2	SOLUTION & TECHNICAL ARCHITECTURE	12
6.	PROJECT PLANNING & SCHEDULING	13
6.1	SPRINT PLANNING & ESTIMATION	13
6.2	SPRINT DELIVERY SCHEDULE	14
7.	CODING & SOLUTIONING	14
7.1	FEATURE 1	14
8.	TESTING	19
8.1	TEST CASES	19
9.	RESULTS	20
9.1	PERFORMANCE METRICS	20
10.	ADVANTAGES & DISADVANTAGES	23
11.	CONCLUSION	23
12.	FUTURE SCOPE	24
13.	APPENDIX	24
13.1	SOURCE CODE	24
13.2	GITHUP & PROJECT DEMO LINK	

## **1.INTRODUCTION :**

Wind speed/power has received increasing attention around the earth due to its renewable nature as well as environmental friendliness.

With the global installed wind power capacity rapidly increasing, the wind industry is growing into a large-scale business. Reliable short-term wind speed forecasts play a practical and crucial role in wind energy conversion systems, such as the dynamic control of wind turbines and power system scheduling.

A precise forecast needs to overcome problems of variable energy production caused by fluctuating weather conditions.

Power generated by wind is highly dependent on the wind speed. Though it is highly non-linear, wind speed follows a certain pattern over a certain period of time. We exploit this time series pattern to gain useful information and use it for power prediction.

### **1.1 PROJECT OVERVIEW :**

#### **Category:**

Machine Learning

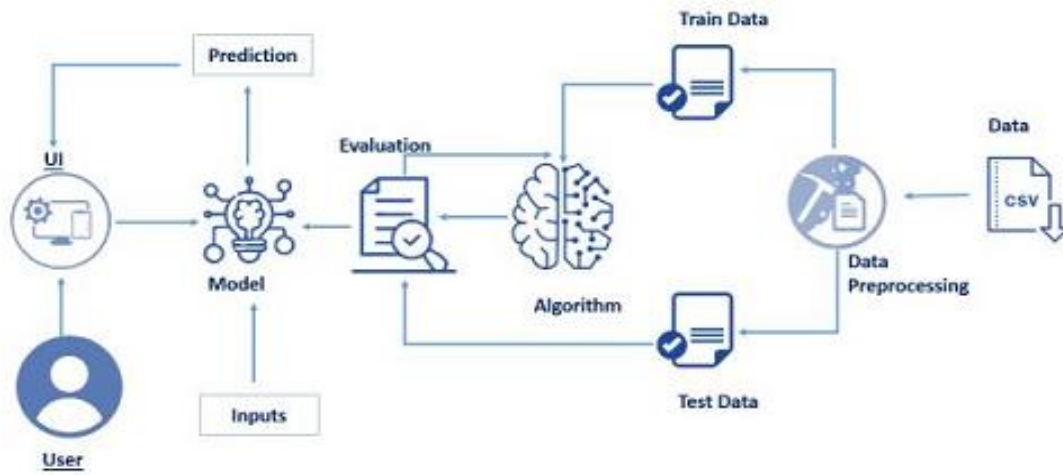
#### **Skills Required:**

Python, Python Web Frame Works, Python For Data Visualization, Data Preprocessing Techniques, Machine Learning, IBM Cloud, IBM Watson Studio, Python-Flask.

#### **Project Description:**

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.

#### **Technical Architecture:**



## 1.2 PURPOSE:

- 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.
- Wind energy **plays an increasing role in the supply of energy worldwide.**

## 2. LITERATURE SURVEY :

### 1.2 EXISTING PROBLEM:

- **Turbines produce noise and alter visual aesthetics :**

Wind farms have different impacts on the environment compared to conventional power plants, but similar concerns exist over both the noise produced by the turbine blades and the visual impacts on the landscape .

- **Sound and visual impact** are the two main public health and community concerns associated with operating wind turbines. Most of the sound generated by wind turbines is aerodynamic, caused by the movement of turbine blades through the air.

### 1.3 REFERENCE:

S.NO	AUTHOR/YEAR	TITTLE	TECHNIQUE USED	MERITS	DEMERITS
1.	A.Clifton/2012	Using machine learning to predict wind turbine power output.	Machine learning.	Strong function wind speed.	Affected by turbulence and shear.
2.	Aman Bahugun/2013	Predicting the energy output of	IBM WATSON AUTO AI	Predicted more accurately.	However,in another study it was found that the prediction

		wind turbine based on weather conditions watson auto AI	machine learning.		errors do not satisfy the KolmogoroveSmirnov test for normal distribution.
3.	Haroon Rashid/2020	Forecasting of wind turbine output power using machine Learning.	Machine learning	Accurate predicting of output power.	Absolute errors for the proposed model.
4.	Katya Vladislavleva/ 2019	Predicting the Energy Output of Wind Farms Based on Weather Data: Important Variables and their Correlation	wind energy, prediction, genetic programming, DataModeler	A good prediction of the energy output.	However, levels of production of wind energy are hard to predict as they rely on potentially unstable weather conditions present at the wind farm.
5.	J K Lundquist and P.Fleming1/2012	Using machine learning to predict wind turbine power output	Machine learning, classification and regression trees, wind energy, wind turbine	Reduce bias in power predictions that arise because of the different turbulence and shear at the new site, compared to the test site.	Changes of wind direction with height, non-uniform shear, and the state of the turbine were not considered here but may impact turbine deployment sites.
6.	Aoife M. Foley/2020	Review Current methods and advances in . forecasting of wind power generation	Meteorology Numerical weather prediction Probabilistic forecasting Wind integration wind power forecasting	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	Overall accurate wind power prediction reduces the financial and technical risk of uncertainty of wind power production for all electricity market participants.



## 3.2 IDEATION& BRAINSTORMING:

### BRAINSTROM & IDEA PRIORITIZATION:

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

#### **Define your problem statement**

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

#### **Brainstorm**

Write down any ideas that come to mind that address your problem statement.

#### **TIP**

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing.

#### **Group ideas**

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

#### **Prioritize**

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

#### **After you collaborate**

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

#### *Team gathering*

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

#### *Set the goal*

Think about the problem you'll be focusing on solving in the brainstorming session.

#### *Learn how to use the facilitation tools*

Use the Facilitation Superpower to run a happy.

## 3.3 PROPOSED SOLUTION:

### Proposed Solution Template:

Project team shall fill the following information in proposed solution template.

S .No.	Parameter	Description
1.	ProblemStatement(Problemto be solved)	Our time –tested kombi Box produce gives a higher priority to those forecasts having the lowest prediction error in respective weather situation.
2.	Idea/Solutiondescription	Our aim is to map weather data to energy production
3.	Novelty/ Uniqueness	A good overview on the different methods that were recently applied in forecasting of wind power generation can be found in .
4.	SocialImpact/ CustomerSatisfaction	If there's a lot of wind, you get more energy output than if there's less wind, which means you will likely want to do maintenance when the winds are low to minimize downtime.
5.	BusinessModel(RevenueModel)	Real time projections for solar power, including behind-the-meter generation Grid-oriented forecasts
6.	Scalabilityofthe Solution	The model prediction is then showcased on user interface to predict the energy output of wind turbine

### 3.4 PROBLEM SOLUTION FIT:

#### 1. CUSTOMERSEGMENT(S):

Windflowsoverthebladesccreatinglift (similartotheeffect onairplanewings),whichcauses the blades to turn.

The blades areconnected to a drive shaft that turns an electricgenerator,whichproduces (generates).

#### 1. JOBS-TO-BE-DONE/PROBLEMS:

Turbinesproducenoiseandaltervisualaesthetics

Wind farms have different impactson the environment compared toconventionalpowerplants,butsimilar concerns exist over both thenoiseproducedbytheturbinebladesandthevisualimpactsonthe landscape.

### 3. TRIGGERS:

The wind speed is always fluctuating andthus the energy content of the wind is alsoalways changing. Exactly how large thevariationisdependsboth ontheweather and onlocalsurfaceconditionsandobstacles.

### 4.EMOTIONS:

#### BEFORE / AFTER

**Before:**who live in close proximity to windturbines say they experience sleepdisturbances, headaches and concentrationproblems.

**After:**Between working long hours, climbingturbinesmultipletimes aday,anddealingwithextreme heat in the summer and cold in thewinter.

### 5.AVAILABLESOLUTIONS:



These residential windmills can generate electricity by churning the wind through its blades, which in turn rotates the turbine and generates power, which can meet the needs of a small family. The energy that is yielded from these wind turbines is clean, renewable, and also so cost-effective.

#### 6. CUSTOMER CONSTRAINTS:

- Intermittent
- Low operating costs
- Noise and visual pollution
- Efficient use of land space
- Some adverse environmental impact
- Wind energy is a job creator

#### 7. BEHAVIOUR:

The wind speed is always fluctuating and thus the energy content of the wind is also always changing.

Exactly how large the variation is depends both on the weather and on local surface conditions and obstacles.

#### 8. CHANNELS of BEHAVIOUR:

##### Online:

An efficient automated approach to wind farm operation monitoring is presented.

##### Offline:

Product is available for offline usage.

#### 9. PROBLEM ROOT CAUSE:

The degradation, weakening and debonding of the adhesive layers (in the trailing or leading edges or on the spar/shell joint) is one of the main processes leading to wind turbine blade failure.

#### 10. YOUR SOLUTION:

Larger rotor diameters allow wind turbines to sweep more area, capture more wind, and produce more electricity. A turbine with longer blades will be able to capture more of the available wind than shorter blades—even in areas with relatively less wind.

## 4. REQUIREMENT ANALYSIS:

### 4.1. FUNCTIONAL REQUIREMENTS:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story/Sub-Task)
FR-1	User Registration and logging in by entering their username and password.	Registration through Form.
FR-2	User Confirmation by validating the username with respect to the password	Confirmation via pop-up Message.
FR-3	Displaying the further information about The application.	By selecting the about button the details of the application will be displayed.

FR-4	Validatingthecityname.	System checks whether the city entered by theuser is present or not. If present it will collectthefurtherdetailseitwilldisplaythepop-upmessageas error inthecity.
FR-5	Checkingthedatatypeofthevalue.	Systemchecksforthedatatypeofthevalueenteredby theuser.
FR-6	Validatingallrequiredfields.	Before predicting the output the systemcheckswhetherallthevaluesareentere dbytheuserand checkswhetherallvaluesarecorrect.
FR-7	Displaying weatherConditionsforagiv encity.	ItdisplaystheweatherofthecitywhichhaveBeensel ected.
FR-8	Displaying predictedEnergyoutp utpower.	The predicted output will be displayed asamountofwindenergypowergenerated.

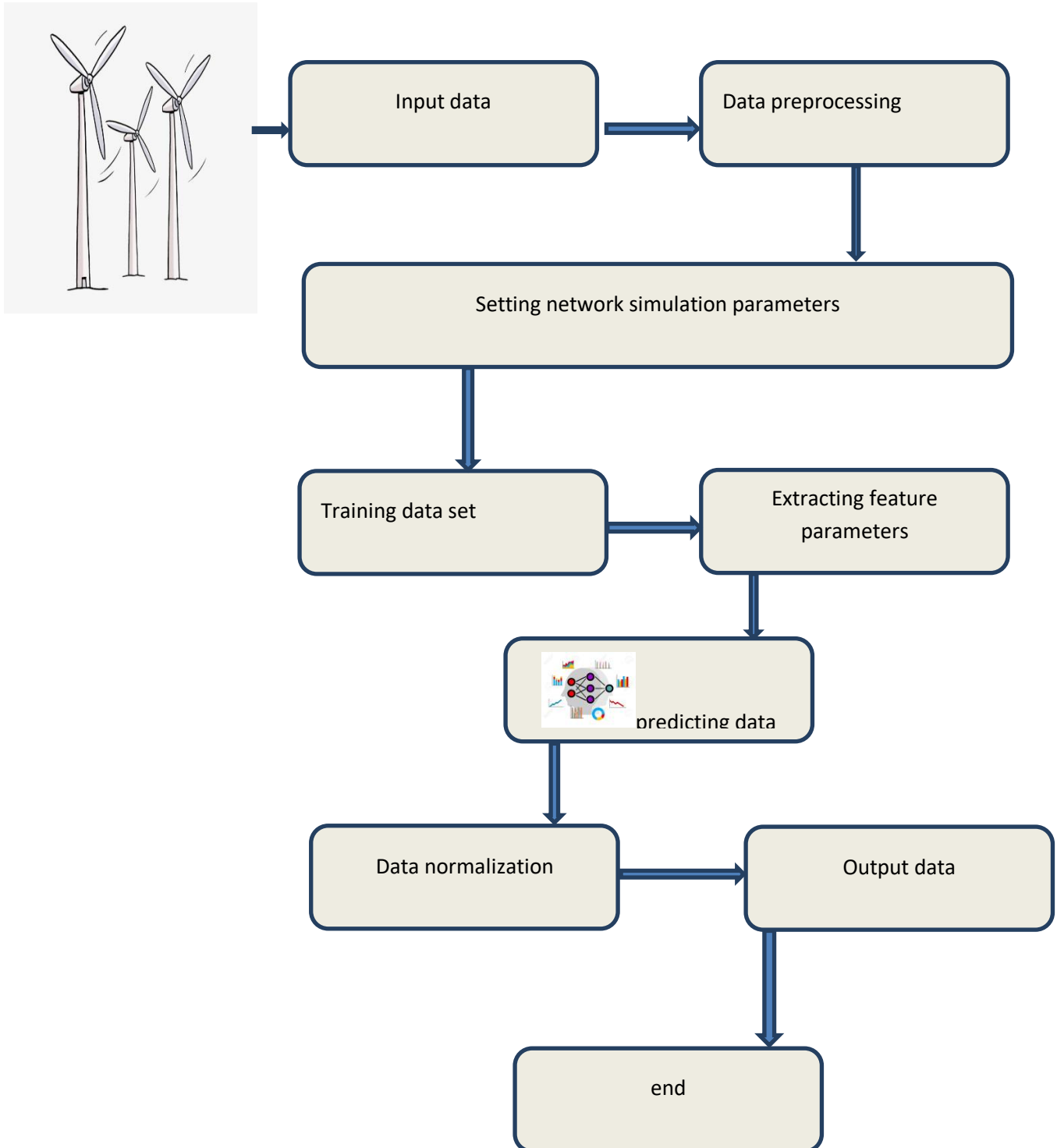
#### 4.2. NON-FUNCTIONAL REQUIREMENTS:

Followingarethennon-functionalrequirementsoftheproposedsolution.

FRNo.	Non-FunctionalRequirement	Description
NFR-1	Usability	Thesystemsatisfiestheusergoalsandtheapplicationi seasytouse.
NFR-2	Security	Thedataproviedtosystemwillbeprotectedfrom Attacksandunauthorizedaccess
NFR-3	Reliability	Thesystemwillprovidetheconsistencyinoutputwith outproducinganerror.
NFR-4	Performance	Theperformancewillneverdegradeeventhewor kloadis increased.
NFR-5	Availability	Theapplicationisavailablefor24*7
NFR-6	Scalability	Thesystemcanbeusedaswebapplicationaswellas mobile application with a sufficientinternet availability.

## 5. PROJECT DESIGN:

### 5.1 DATA FLOW DIAGRAMS:



## 5.2 SOLUTION & TECHNICAL ARCHITECTURE:

### Solution Architecture:

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- 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.
- The inlet condition of the wind farm is forecasted by the auto regressive model.
- We report on the correlation of the different variables for the energy output.

### Example - Solution Architecture Diagram:

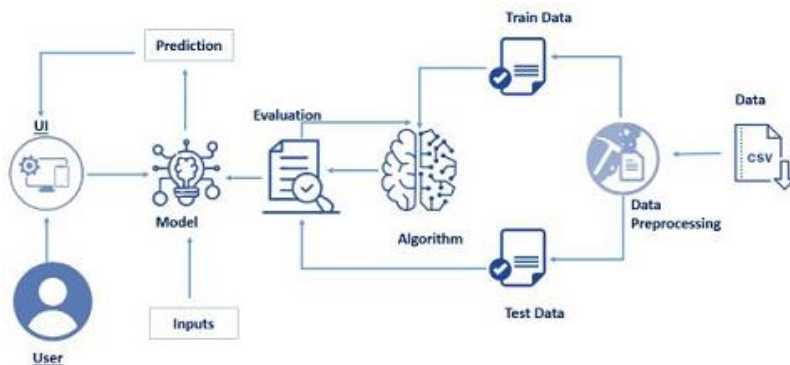


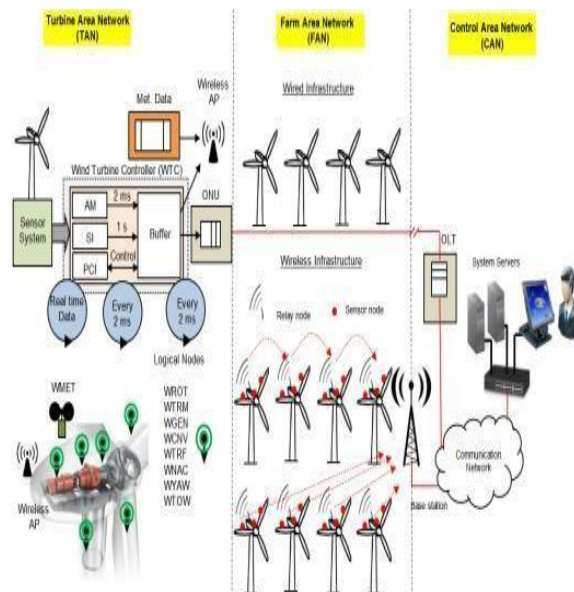
Figure 1: Architecture and data flow

Reference: <https://github.com/SmartPracticeschool/IISPS-INT-3437-Predicting-the-Energy-Output-of-Wind-Turbine-Based-on-Weather-Conditions-Watson-Auto->

### TechnicalArchitecture:

The Deliverables shall include the architectural diagrams below and the information as per the table 1 & table 2

**Example: Predicting the energy output of wind farm based on weather conditions.**



## Guidelines:

The proposed communication network architecture for the Smart-WPF consists of three networks: the turbine area network (TAN), the farm area network (FAN), and the control area network (CAN).

It consists of hierarchical architectures where Level 1 is a sensor network in a single wind turbine, Level 2 is the wind turbine-to-wind turbine interaction in the WPF, Level 3 is the local control center to wind turbine interaction, and Level 4 is the farm-to-farm interaction to optimize grid operation.

In order to implement hierarchical network architectures, a hybrid communication solution is considered. EPON-based architecture represents a wired solution, while ZigBee-Pro is considered for the wireless solution. In this work, Levels 1 and 2 are explained in more detail, while Levels 3 and 4 are out of the scope of this work.

## 6. PROJECT PLANNING & SCHEDULING:

### 6.1 SPRINT PLANNING & ESTIMATION:

Sprint	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)
Sprint-1	01 Nov 2022	14 Nov 2022	20

Sprint-2	07 Nov2022	14Nov2022	20
Sprint-3	07Nov2022	15Nov2022	20
Sprint-4	09 Nov2022	15Nov2022	20

## 6.2 SPRINT DELIVERY SCHEDULE:

Sprint	SprintStartDate	SprintEndDate(Planned)	Story PointsCompleted (ason PlannedEndDate)	SprintReleaseDate(Actual)	
Sprint-1	01 Nov2022	14 Nov2022	20	14 Nov 2022	
Sprint-2	07 Nov2022	14Nov2022	20	14 Nov2022	
Sprint-3	07Nov2022	15Nov2022	20	15 Nov2022	
Sprint-4	09 Nov2022	15Nov2022	20	15 Nov2022	

## 7. CODING & SOLUTIONING:

### 7.1 FEATURE 1:

Saving Wind Dataset.csv to Wind Dataset.csv

In [2]:

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

In [3]:

```
Data Preprocessing
path = "Wind Dataset.csv"
df = pd.read_csv(path)
df.rename(columns={"Date/Time":"Time", "LV ActivePower (kW)":"ActivePower(KW",
                  "Wind Direction(°)":"Wind_Direction"}),
df.head() inplace=True)
```

In [4]:

<b>0</b>	01.01 2018 00:00	380.047791	5.311336	416.328908259.994904
<b>1</b>	01.01 2018 00:10	453.769196	5.672167	519.917511 268.641113
<b>2</b>	01.01 2018 00:20	306.376587	5.216037	390.900016272.564789
<b>3</b>	01.01 2018 00:30	419.645904	5.659674	516.127569271.258087
<b>4</b>	01.01 2018	380.650696	5.577941	491.702972
		265.674286		

ActivePower(KW) 0  
WindSpeed(m/s) 0  
Theoretical\_Power\_Curve (KWh) 0  
Wind Direction (°) 0  
dtype: int64

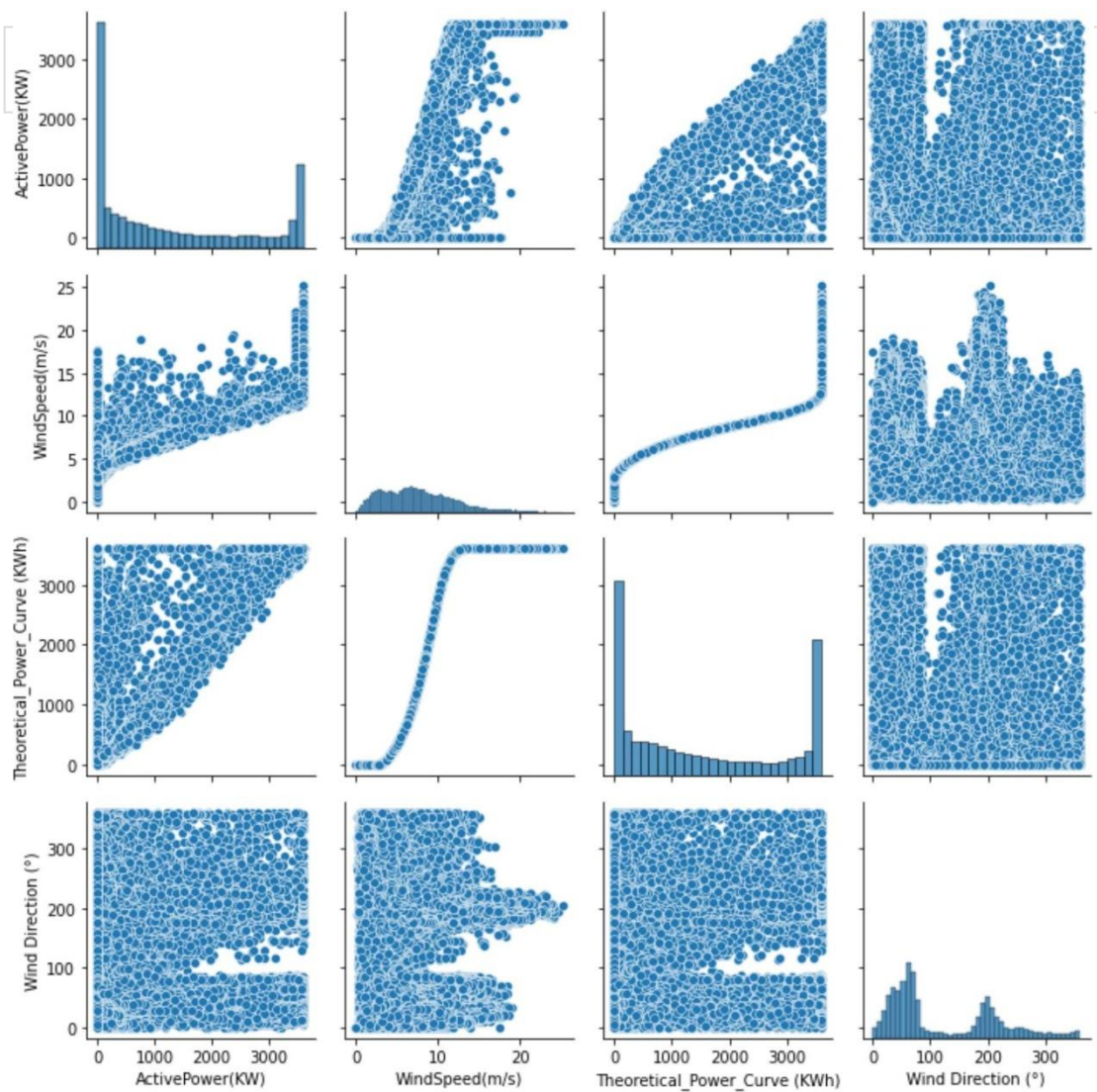
```
In [10]: Out[4]: rplot(df)
```

Wind Direction (°)  
ActivePower(KW) -0.062702  
WindSpeed(m/s) -0.077188  
Theoretical\_Power\_Curve (KWh) -0.099076  
Wind Direction (°) 1.000000

Out[6]:

In [5]:

Out[5]:



In [7]:

```
plt.figure(figsize=(10, 8))
corr = df.corr()
ax = sns.heatmap(corr, vmin = -1, vmax = 1, annot = True)
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)

print(corr)
```





```
In [8]: df["Time"] = pd.to_datetime(df["Time"], format = "%d %m %Y %H %M", errors =
```

Splitting the data to Train and Test

```
In [9]: y = df["ActivePower(KW)"]
X = df[["Theoretical_Power_Curve (KWh)", "WindSpeed(m/s)"]]

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

Model Building

```
In [10]: from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, r2_score

forest_model = RandomForestRegressor(n_estimators = 750, max_depth = 4, max_
forest_model.fit(train_X, train_y)
```

```
Out[10]: RandomForestRegressor(max_depth=4, max_leaf_nodes=500, n_estimators=750,
random_state=1)
```

```
In [11]: power_preds = forest_model.predict(val_X)

print(mean_absolute_error(val_y, power_preds))
print(r2_score(val_y, power_preds))
```

Wind Direction (°) 1.000000

0.911349642890175

In [12]: `joblib.dump(forest_model, "power_prediction.sav")`

Out[12]: ['power\_prediction.sav']

In [13]: `df`

Out[13]:

	Time	ActivePower(KW)	WindSpeed(m/s)	Theoretical_Power_Curve (KWh)	Wind Direction (°)
<b>0</b>	NaT	380.047791	5.311336	416.328908	259.994904
<b>1</b>	NaT	453.769196	5.672167	519.917511	268.641113
<b>2</b>	NaT	306.376587	5.216037	390.900016	272.564789
<b>3</b>	NaT	419.645904	5.659674	516.127569	271.258087
<b>4</b>	NaT	380.650696	5.577941	491.702972	265.674286
...	...	...	...	...	...
<b>50525</b>	NaT	2963.980957	11.404030	3397.190793	80.502724
<b>50526</b>	NaT	1684.353027	7.332648	1173.055771	84.062599
<b>50527</b>	NaT	2201.106934	8.435358	1788.284755	84.742500
<b>50528</b>	NaT	2515.694092	9.421366	2418.382503	84.297913
<b>50529</b>	NaT	2820.466064	9.979332	2779.184096	82.274620

50530 rows × 5 columns

8.TESTING:

8.1 TEST CASES:

TestingNo:1	TestingType	FunctionResult
	Functionalitytesting	Yes
	Usabilitytesting	Yes
	Interfacetesting	Yes
	Performancetesting	Yes(medium)
	Securitytesting	Yes

TestingNo:2	TestingType	FunctionResult
	Functionalitytesting	Yes
	Usabilitytesting	Yes
	Interfacetesting	Yes
	Performancetesting	Yes(medium)
	Securitytesting	Yes

TestingNo:3	Testing Type	Function Result
	Functionality testing	Yes
	Usability testing	Yes
	Interface testing	Yes
	Performance testing	Yes
	Security testing	Yes

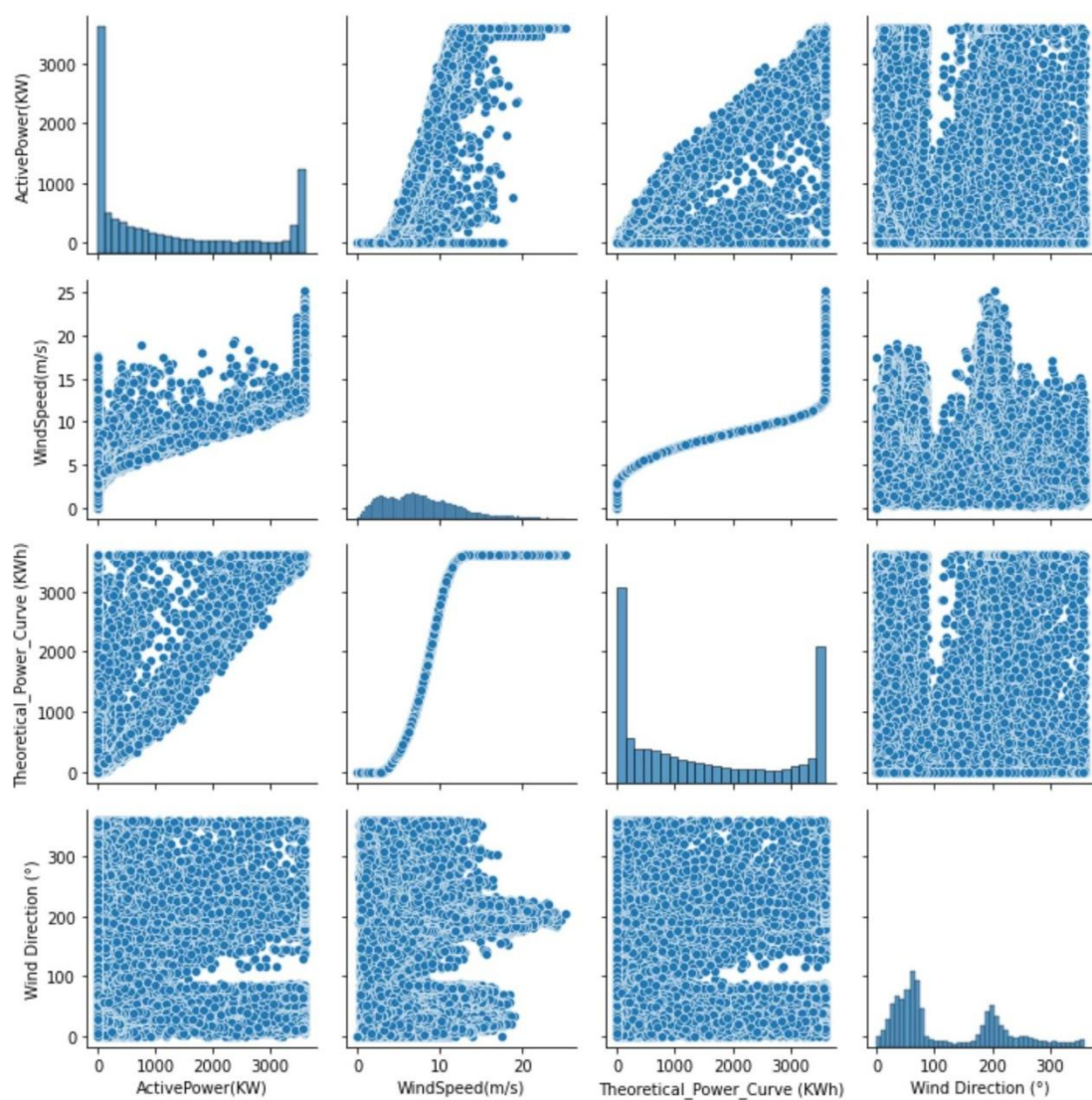
TestingNo:4	Testing Type	Function Result
	Functionality testing	Yes
	Usability testing	Yes
	Interface testing	Yes
	Performance testing	Yes
	Security testing	Yes

9.RESULT:

9.1 PERFORMANCE METRIES:

DATA PREPROCESSING:

<hr/>				
01 01				
0	2018	380.047791	5.311336	416.328908
259.994904				
00:00				
01 01				
1	2018	453.769196	5.672167	519.917511
268.641113				
00:10				
Out[4]:	01 01			
2	2018	306.376587	5.216037	390.900016
272.564789				
00:20				
01 01				
3	2018	419.645904	5.659674	516.127569
271.258087				
00:30				
01 01				
4	2018	380.650696	5.577941	491.702972
265.674286				





**MODEL BUILDING:**

```
Out[10]: RandomForestRegressor(max_depth=4, max_leaf_nodes=500, n_estimators=750,
                                random_state=1)
```

**FINAL RESULT:**

```
1 import numpy as np
2 from flask import Flask, request, jsonify, render_template
3 import joblib
4 import requests
5
6 # NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
7 API_KEY = "z5w_7QoNuv-PziHyHPjrv_fkckHyZvDd_vhj4lMS2dli"
8 token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":
9 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
10 mltoken = token_response.json()["access_token"]
11
12 header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
* Serving Flask app 'app_IBM' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Activate Windows  
Go to Settings to activate Windows.

## 10. ADVANTAGES& DISADVANTAGES:

### ADVANTAGES:

- Weather Underground Services provide very accurate Historical Weather Data which increased the accuracy of model.
- Website is more convenient to use due to zero storage.
- With Choosing city, Website can accurately predict power output using weather condition.

### DISADVANTAGES:

- Weather API is paid and the free version provide limited API requests per day.
- Android Website can be deployed on IBM Cloud.
- No free server available on IBM Cloud for deploying Backend.

## 11. CONCLUSION:

In this study we showed that wind energy output can be predicted from publicly available weather data with accuracy at best 80%  $R^2$  on the training range and at best 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 of prediction accuracy for 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.

## **12. FUTURE SCOPE:**

Most wind power forecasting models study ‘regular’ wind conditions.

The EU funded project called ‘Safewind’ aims to improve wind power prediction overchallenging and extreme weather periods and at different temporal and spatial scales.

Development activities are on-going to reduce error in the wind power prediction, to improve regionalized wind power forecasting for on - shore windfarms and to derive methods for wind power prediction for offshore wind farms.

It is possible that use of ensemble and combined weather prediction methods together may enhance forecasting.

If the error in wind power forecasting and prediction is reduced then electricity markets can trade with more certainty.

Contract errors as a function of time in electricity markets can be as high as 39% for a forecasting lead time of 4 h Gubina et al.

present a new tool called the WILMAR and ANEMOS scheduling Methodology (WALT) to reduce the number of thermal generators on stand-by or in reserve using the probability of generation outages and load shedding are system reliability criteria instead of generation adequacy based solely on generation outage.

The wind and load forecast errors are modelled using a Gaussian stochastic variable approach.

However, in another study it was found that the prediction errors do not satisfy the Kolmogorov-Smirnov test for normal distribution.

In Ramírez and Carta , it was shown that, the use of autocorrelated (and thus not independent) successive hourly mean wind speeds, though invalidating all of the usual statistical tests, has no appreciable effect on the shape of the pdf estimated from the data.

## **13. APPENDIX:**

### **13.1 SOURCE CODE:**

➤ **app.py**



```

import numpy as
npimport streamlit
as stimport pandas
as
pdimportdatetime

importplotly.graph_objectsasgoi
mportbase64

import
timeimporttensor
flow

st.set_page_config(page_ti
tle="DEEPWIND",page_icon
="🚢"
)

old_models=tensorflow.keras.models.load_model('model.h5')

#setbackground,usebase64toreadlocalfiledef
get_base64_of_bin_file(bin_file):

    withopen(bin_file,'rb')asf:d
        ata=f.read()

    returnbase64.b64encode(data).decode()

defset_png_as_page_bg(png_file):

bin_str=get_base64_of_bin_file(png_file)
page_bg_img=""

<style>

    body {

        background-
image:url("data:image/png;base64,%s");background
-size:cover;

```

```
}

</style>

'''%bin_str

st.markdown(page_bg_img,unsafe_allow_html=True)re
turn

set_png_as_page_bg('gr.gif')

defhome():

    return"welcome"


def
predict(temperature,pressure,wind_speed,wind_direction):values=np.array([[tempe
rature,pressure,wind_speed,wind_direction]])
```

```
prediction=old_models.predict(values.reshape(-
1,1,4),batch_size=1)print(prediction)
```

```
returnprediction
```

```
defmain():
```

```
st.sidebar.markdown("<h1style='text-align:center;color:black;'>unsafe_allow_html=True)
```

Navigation Bar  </h1>",

```
nav =st.sidebar.radio("",["Home", "User defined Prediction", "Forecasting"])
```

```
ifnav == "Home":
```

```
st.markdown("<h1 style='color:black; text_align:center;font-family:times new roman;font-size:20pt; font-weight: bold;'>DEEPWINDS </h1>", unsafe_allow_html=True)
```

```
st.markdown("<h1style='color:brown;text_align:center;font-weight:bold;font-size:19pt;'>MadebyQuadTechieswith</h1>",unsafe_allow_html=True)
```

```
st.markdown("<h1 style='color:black; text_align:center;font-family:times new roman;font-weight:bold;font-size:16pt;'>♦ WINDPOWERPREDICTIONDLWEB-APP</h1>",unsafe_allow_html=True)
```

```
if nav == "User definedPrediction":
```

```
set_png_as_page_bg('gra(1).jpg')
```

```
st.markdown("<h1 style='text-align: center; color: green;'>UserInput Parameters </h1>", unsafe_allow_html=True)
```

```
withst.beta_expander("Preferences"):
```

```
st.markdown("<h1style='text-align:left;font-weight:bold;color:black;background-
```

```
color:white;font-size:11pt;'>Temperature<math>\frac{1}{2}</math>(°C) </h1>",unsafe_allow_html=True)
```

```
col1,col2=st.beta_columns(2)
```

```
withcol1:
```

```
min_temp=st.number_input('Minimum Temperature(°C)',min_value=-89,max_value=55,value=-15,step=1)
```

```
withcol2:
```

```
max_temp=st.number_input('🔥 Maximum Temperature(°C)',min_value=-88,max_value=56,value=50,step=1)
```

```
st.markdown("<h1 style='text-align: left; font-weight:bold;color:black;background-color:white;font-size:11pt;'> Wind Speed 🌬️(m/s) </h1>",unsafe_allow_html=True)
```

```
col1,col2 =  
st.beta_columns(2)with col1:
```

```
min_speed=st.number_input(' MinimumWindSpeed  
(m/s)',min_value=0,max_value=99,value=1,step=1)with  
col2:
```

```
max_speed=st.number_input(' MaximumWindSpeed  
(m/s)',min_value=2,max_value=100,value=27,step=1)st.  
write("")
```

```
temperature=st.slider('Temperature 🌡️ 1/2 [°C]', min_value=min_temp, step=1,  
max_value=max_temp,value=max_temp)
```

```
pressure =st.slider('Pressure 🏹 [atm]', 0.9, 1.0, 1.0)
```

```

wind_speed = st.slider('Wind Speed 🌬️ [m/s]', min_value=min_speed, step=1,
max_value=max_speed, value=max_speed)

wind_direction = st.slider('Wind Direction 🌬️ ➡️ [deg]', 0, 1, 360)
result=""

if st.button("Predict"):

    result =
predict(temperature, pressure, wind_speed, wind_direction) st.balloons(
)

st.success('Predicted Power is {} kW'.format(result))

if nav == "Forecasting 🌤️":
set_png_as_page_bg('04.gif')

st.markdown("<h1 style='text-align: center; color: black ;'> 🌤️ FORECASTING 🌤️ </h1>",
unsafe_allow_html=True)

#Setup fileupload

st.markdown("<h1 style='text-align:center; color:white;background-color:black;font-
size:14pt'> 🌤️ Upload your CSV or Excel file. (200MB max) 🌤️ </h1>", unsafe_allow_html=True)

uploaded_file = st.file_uploader(label="", type=['csv',
'xlsx']) global df

if uploaded_file is not None:

    print(uploaded_file)

st.markdown("<h1 style='text-align:center; color:black;background-
color:lightgreen;font-size:14pt'> 🌤️ Fileupload
successful 🌤️ </h1>", unsafe_allow_html=True)

try:

    df = pd.read_csv(uploaded_file)
t.write(df)

except Exception as e:

```

```

df =
pd.read_excel(uploaded_file).write(df)

st.markdown("<h1style='text-align:center;color:black;background-color:powderblue;font-size:14pt'>INPUTDATA</h1>", unsafe_allow_html=True)

trace
=go.Scatter(x=df['DateTime'],y=df['Powergeneratedbysystem |(kW)'],mode='lines',name='Data
')

layout=go.Layout(title="",xaxis={'title':"Date"},yaxis={'title' :"Powergeneratedbysystem
|(kW)"})

fig = go.Figure(data=[trace],
layout=layout).write(fig)

df1=df.reset_index()['Powergeneratedbysystem |(kW)']i
mportmatplotlib.pyplot asplt

st.write("\n")

```

```
st.markdown("<h1style='text-align:center;color:black;background-color:powderblue;font-size:14pt'>INPUTDATAINTERMSOFNO. OFHOURS</h1>",
```

```
unsafe_allow_html=True)
```

```
trace = go.Scatter(x = df1.index,y = df['Power generated by system | (kW)'],mode = 'lines',name = 'Data' )
```

```
layout = go.Layout(title = "",xaxis = {'title' : "No. of hours"},yaxis = {'title' : "Powergeneratedby system (kW)"})
```

```
fig=go.Figure(data=[trace],layout=layout)#fig.show()
```

```
st.write(fig)
```

```
from sklearn.preprocessing import  
MinMaxScaler scaler=MinMaxScaler(feature_range=  
(0,1))df1=scaler.fit_transform(np.array(df1).reshape  
(-1,1))
```

```
##splitting dataset into train and test  
splittraining_size=int(len(df1)*0.65)test_size=  
len(df1)-training_size
```

```
train_data,test_data=df1[0:training_size:],df1[training_size:len(df1),:1]
```

```
import numpy
```

```
# convert an array of values into a dataset
```

```
matrix#convertan  
arrayofvaluesintoadatasetmatrixdefcreate_data  
set(dataset,time_step=1):
```

```
dataX,dataY=[],[]
```

```
foriinrange(len(dataset)-time_step-1):
```

```
a = dataset[i:(i+time_step), 0]###i=0, 0,1,2,3-----  
dataX.append(a)
```

```

        dataY.append(dataset[i + time_step,
0]))returnnumpy.array(dataX),numpy.array(data
Y)

#reshapeintoX=t,t+1,t+2,t+3andY=t+4tim
e_step=30

X_train, y_train = create_dataset(train_data,
time_step)X_test,ytest=create_dataset(test_data,
time_step)

#reshapeinputtobe[samples,timesteps,features]whichisrequiredforLSTMX_trai
n=X_train.reshape(X_train.shape[0],X_train.shape[1], 1)

X_test=X_test.reshape(X_test.shape[0],X_test.shape[1],1)#
Create theBILSTMmodel

fromtensorflow.keras.modelsimportSequential
fromtensorflow.keras.layersimportDense

fromtensorflow.keras.layersimportLSTM

fromtensorflow.keras.layersimportBidirectional
model=Sequential()

model.add(Bidirectional(LSTM(250,input_shape=(1,30))))m
odel.add(Dense(1))

model.compile(loss='mae',optimizer='adam')

```



```
model.fit(X_train,y_train,validation_data=(X_test,ytest),epochs=10,batch_size=64,verbose=1)
import tensorflow as tf
```

```
train_predict=model.predict(X_train)test_predict=model.predict(X_test)
```

```
#Transform back to original
from train_predict=scaler.inverse_transform(train_predict)test_predict=
scaler.inverse_transform(test_predict)
```

```
###Calculate RMSE performance metrics
import math
```

```
from sklearn.metrics import mean_squared_error
math.sqrt(mean_squared_error(y_train,train_predict))
```

```
###Test Data RMSE
math.sqrt(mean_squared_error(ytest,test_predict))
```

```
### Plotting
```

```
# shift train predictions for plotting
look_back=30
```

```
trainPredictPlot=numpy.empty_like(df1)
trainPredictPlot[:,:] = np.nan
```

```
trainPredictPlot[look_back:len(train_predict)+look_back,:]=train_predict#
shift test predictions for plotting
```

```
testPredictPlot =
numpy.empty_like(df1)
testPredictPlot[:, :] = numpy.nan
```

```
testPredictPlot[len(train_predict)+(look_back*2)+1:len(df1)-1, :] =
test_predict#plot baseline and predictions
```

```
st.markdown("<h1 style='text-align:center;color:black;background-color:powderblue;font-size:14pt'><img alt='TRAIN AND TEST DATA'></h1>",unsafe_allow_html=True)
```

```
#plt.plot(scaler.inverse_transform(df1))
plt.plot(scaler.inverse_transform(df1), color="blue", linewidth=1, linestyle="-")
plt.xlabel('No. of hours')
```

```

    # Set the y axis label of the current
axis=plt.ylabel('Power generated by system |(kW)')

plt.plot(trainPredictPlot,label='Train Data',color="black",linewidth=2, linestyle="--")
plt.plot(testPredictPlot,label='Test Data',color="orange",linewidth=2, linestyle="--")
plt.legend(loc="upperleft")

plt.show()
st.pyplot(plt)

x_input=test_data[len(test_data)-30:].reshape(1,-1)
temp_input=list(x_input)
temp_input=temp_input[0].tolist()

#demonstrate prediction for next 24 hours
from numpy import array

lst_output=[
]
n_steps=30
i=0

```

```

        while(i<24):if(len(temp_input)>30):#print(temp_input)

x_input=np.array(temp_input[1:])
x_input=x_input.reshape(1,-1)

x_input = x_input.reshape((1, n_steps,
1))yhat = model.predict(x_input,
verbose=0)temp_input.extend(yhat[0].tolist())temp_input=temp_input[1:]lst_output.extend(yhat.tolist())

i=i+1

        else:

x_input = x_input.reshape((1,
n_steps,1))yhat=model.predict(x_input,verbose=0)

        print(yhat[0])temp_input.extend(yhat[0].tolist())

        print(len(temp_input))
        lst_output.extend(yhat.tolist())
        i=i+1

        print(lst_output)day_new=np.arange(1,31)day_pred=np.arange(len(df1),len(df1)+24)

        import matplotlib.pyplot as pltprint(len(df1))progress=st.progress(0)

        for i in
range(100):time.sleep(0.1)progress.progress(i+1)s
t.balloons()

```

```
st.markdown("<h1style='text-align:center;color:black;background-color:powderblue;font-size:14pt'> PREDICTEDRESULTSFORNEXT24 HOURS </h1>",
```

```
unsafe_allow_html=True)plt.plot(day_pred,scaler.inverse_transform(lst_output),color="green",  
linewidth=1.5, linestyle="--",marker='*',markerfacecolor='yellow',markersize=7)
```

```
plt.legend('GTTP',loc="upperleft")
```

```
plt.xlabel('No.ofhours')
```

```
# Set the y axis label of the current  
axis=plt.ylabel('Powergeneratedbysystem|(kW)')
```

```
st.pyplot(plt)
```

```
st.markdown("<h1 style='text-align: center; color:black ;background-color:yellow;font-size:14pt'> G-GivenData, \n T-Train Data,\n T-TestData,\n P-Predicted Results</h1>",unsafe_allow_html=True)
```

```
st.write(scaler.inverse_transform(lst_output))
```

```
if __name__ == "__main__":
```

```
    main()
```

### ➤ **model.py**

```
import pandas as
```

```
pd
```

```
import datetime
```

```
import numpy
```

```
from keras.models
```

```
import Sequential
```

```
from keras.layers
```

```
import Dense
```

```
from keras.layers import LSTM
```

```
from keras.layers import
```

```
Bidirectional
```

```
import pandas as
```

```
pd
```

```
import keras
```

```
"""Loading data"""
```

```
df =
```

```
pd.read_excel('
```

```
Dataset.csv')
```

```
df = df.drop(columns=['
```

```
Date Time']
```

```
)]""" Cleaning
```

```
Data
```

```
"""# dataframe.d
```

```
rop['Date'].valu
```

```
es
```

```
df['Power generated by system | (kW)'].replace(0,  
np.nan,  
inplace=True)df['Powergeneratedbysystem | (kW)'].fillna  
(method='ffill',inplace=True)
```

```
X=df.drop(columns=['Powergeneratedb  
y system | (kW)'])Y = df[['Power  
generated by system |  
(kW)']]X=np.array(X).reshape(-1,1,4)
```

```
Y=np.array(Y).reshape(-1,1,1)
```

```
model=Sequential()
```

```
model.add(Bidirectional(LSTM(100,  
activation='relu',input_shape=(-  
1,1,4))))model.add(Dense(1))
```

```
model.compile(loss='mae',optimizer='adam',metrics=['accuracy'])
```

```
model.fit(X,Y,epochs=100,callbacks=[keras.callbacks.EarlyStopping(patience=3)])
```

```
test_data = np.array([[  
4.858,0.989741,6.651,273]])o=model  
.predict(test_data.reshape(-  
1,1,4),batch_size=1)print(o)
```

```
# Saving  
model to  
diskmodels=  
model.save('model.h5').
```