PREDICTING THE ENERGY OUTPUT OF WIND TURBINE BASED ON WEATHER CONDITION

Team ID:PNT2022TMID46139

Bachelor of Engineering

Computer Science And Engineering

Dhanalakshmi Srinivasan Institute of Technology

Samayapuram

Faculty Evaluator : Mr.S.Sasikumar
Faculty Mentor : Mrs.J.Fahamitha

Team Members :-
R.Ragha sudha
G.Arthika
M.Mahalakshmi
E.Srimitha
R.Mariya

1.INTRODUCTION

1.1: Project Overview

Predict the output power of a Wind Turbine at any given time provided with Weather Conditions. Using Machine Learning that takes on previous performance data and real time weather parameters to predict the energy output will help in integrating with the grid and make use of its full potential. The wind speed and wind direction can be given as input and the model will predict the output power of the turbine. Different machine learning models have been evaluated to determine the best fitting model.

1.1: Purpose

Due to the unpredictable nature of Wind speed and direction(weather condition). Because of this the power generated by a wind mill is irregular and unpredictable. The power generated depends on a large number of variables like, season, temperature, yearly currents, humidity, pressure, location, altitude, height off the turbine, blade size, blade pitch and many more. Owing to the irregular nature of the output power it is very difficult to integrate this source of renewable energy with the grid. In consequence Wind Farms loss revenue unable to supply the power at the right time to the grid.

2. LITERATURE SURVEY

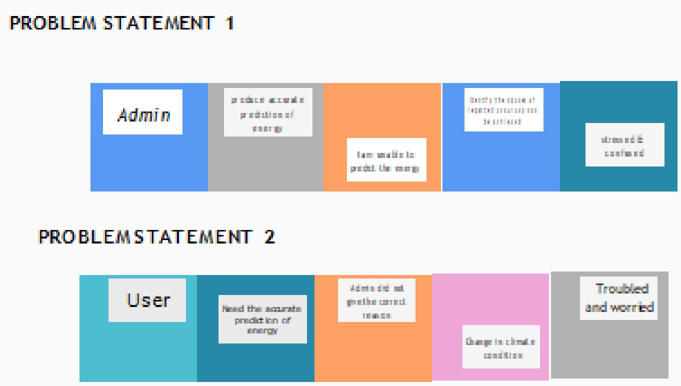
SI.NO	TITLE	ABSTRACT	MERITS	DEMERITS
1.	Predicting The Energy Output Of Wind FarmsBased On Weather Data: Important Variables And Their Correlation	The energy output of the wind farm is highly depend on the weather conditions present atthe wind fram.	Wind energyoutput can be predicted from publicly available weather data with accuracy at best 80%	Default settings to run the symbolic regression experiments as well as variable importance.
2.	Wind powerforecasting based on time series model using deep learning algorithms.	Wind energy is created due to uneven heating of the earthsurface and coriolis acceleration	To minimize risk and to improve performance.	Concerning to predictdifficult operation problems.
3.	Using machine learning to predict wind turbine power output	In this work, new aerostructural simulations of a generic 1.5 MW turbine are used to rank atmospheric influences on power output.	Simulations of a utility-scale wind turbine have been used to develop a database	Application of the data to wind turbine deployment sitesdoes not require any new instrumentation compared to what is currently used.

2.2 References

https://hpi.de/friedrich/docs/paper/RE1.pdf

2.3: Problem Statement Definition

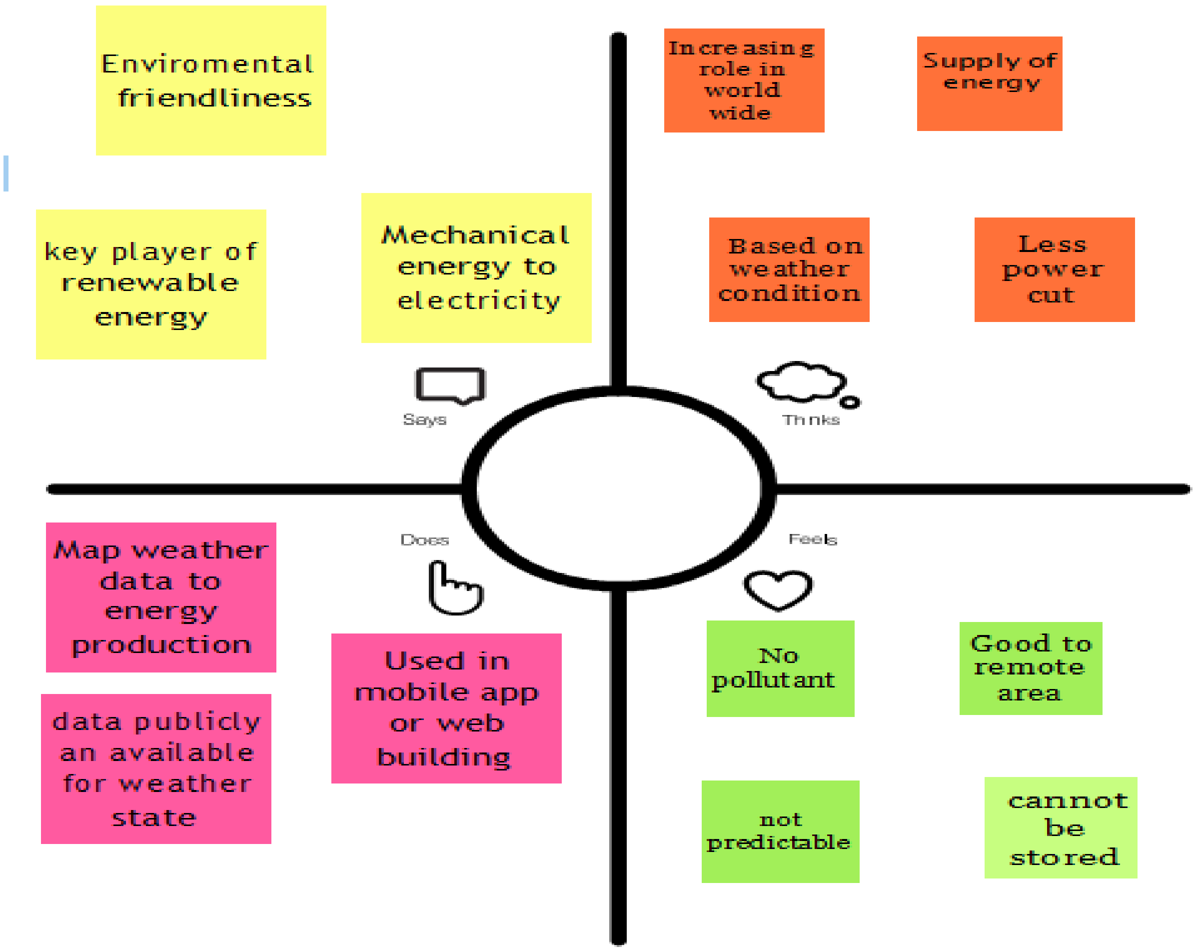
Predict the output power of a Wind Turbine at any given time provided with Weather Conditions. Using Machine Learning that takes on previous performance data and real time weather parameters to predict the energy output will help in integrating with the grid and make use of its full potential. Accurate wind power forecasting reduces the need for additional balancing energy and reserve power to integrate wind power. To make use wind energy efficiently the accurate power output is required. When power output of a wind mill at a given time isknown we can integrate it with grid and make use of this renewable source of energy rather than conventional non-renewable source.



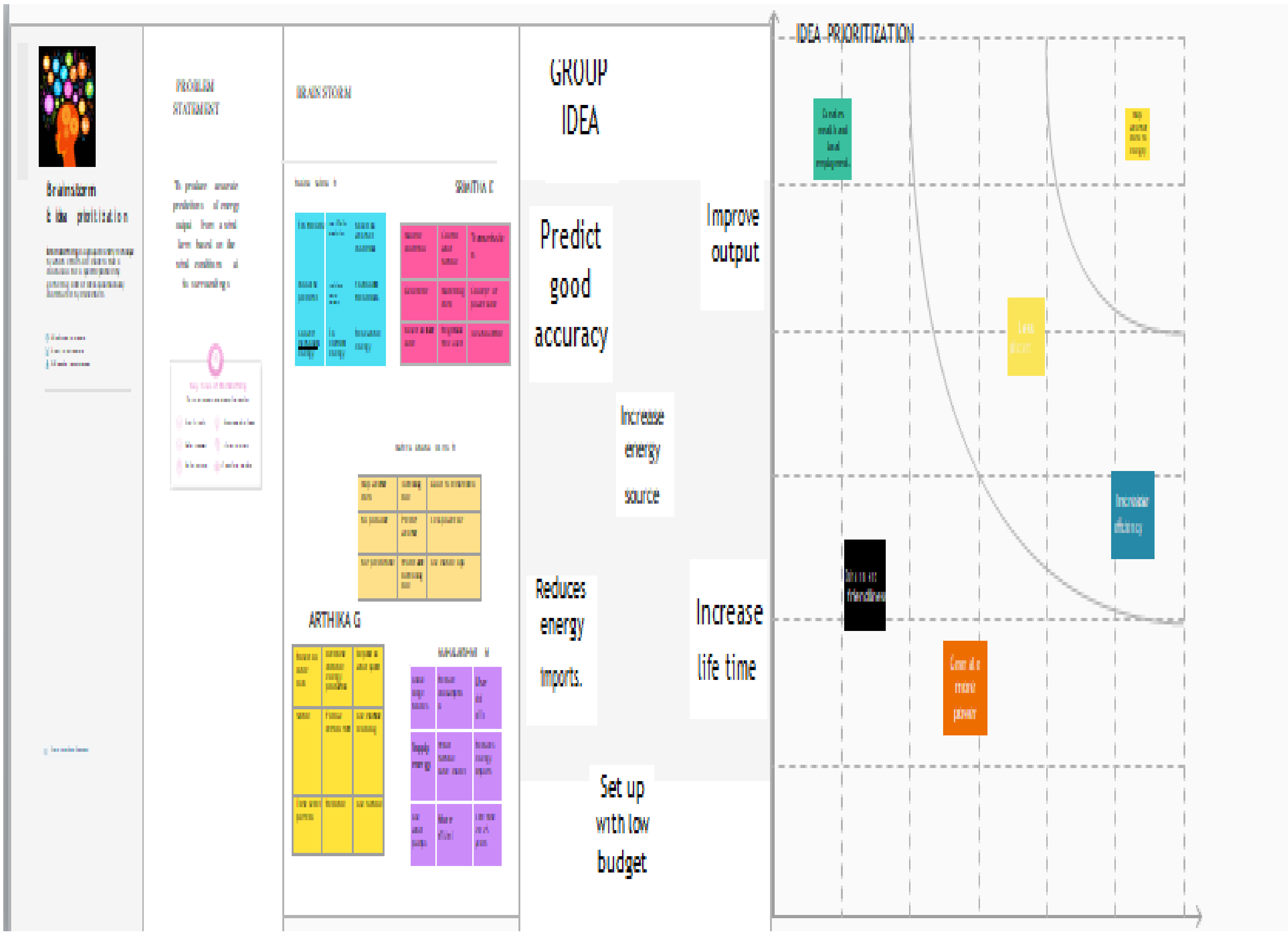
Problem Statement (PS)	I am	I'm trying to	But	Because	Which makes me feel
PS-1	Admin	Produce accurate prediction of energy	I am unable to predict the energy	Weather condition	stressed & confused
PS-2	User	Need the accurate prediction of energy	Admin did not give the correct reason	Change in climate condition	Troubled and worried

3. IDEATION AND PROPOSED SOLUTION

3.1 : Empathy Map Canvas



3.2: Ideation And Brainstorming

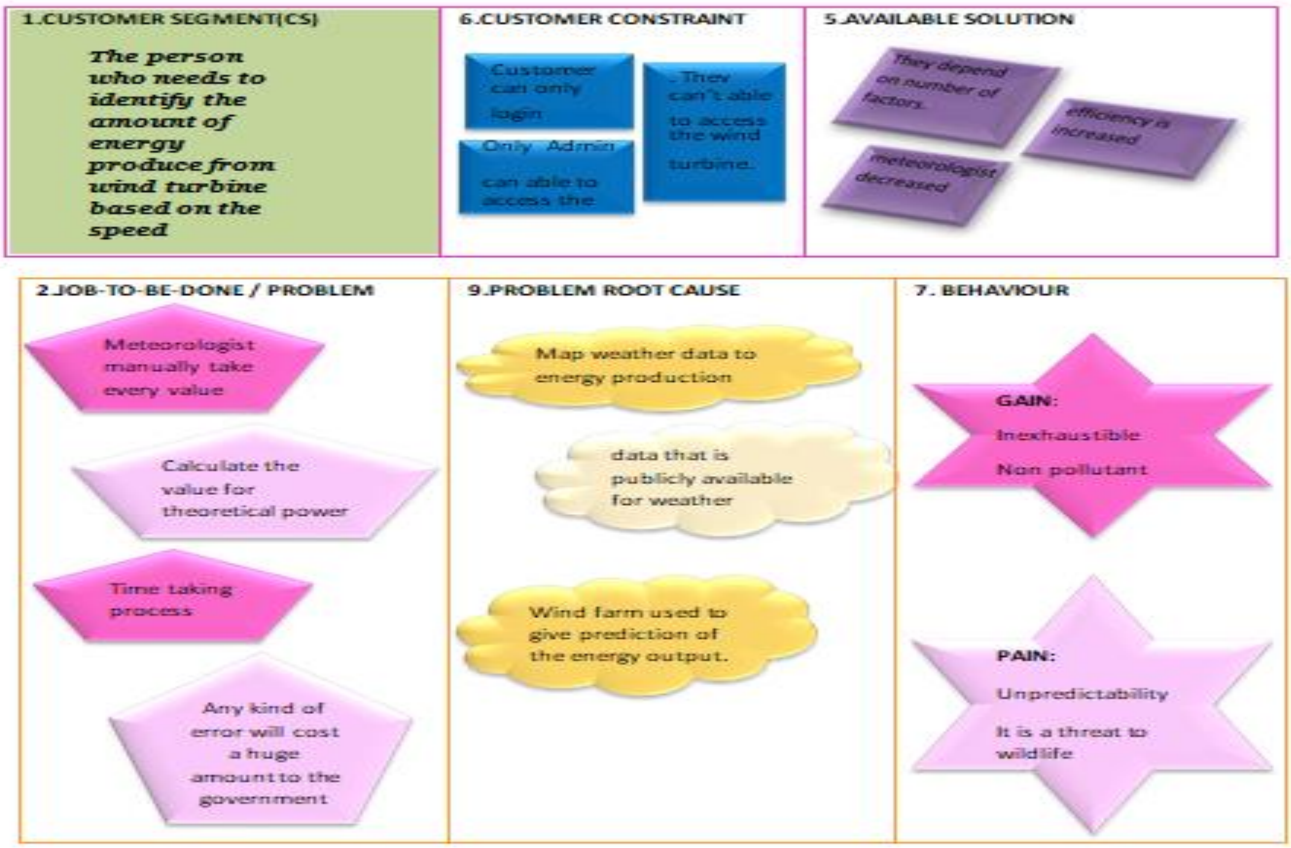


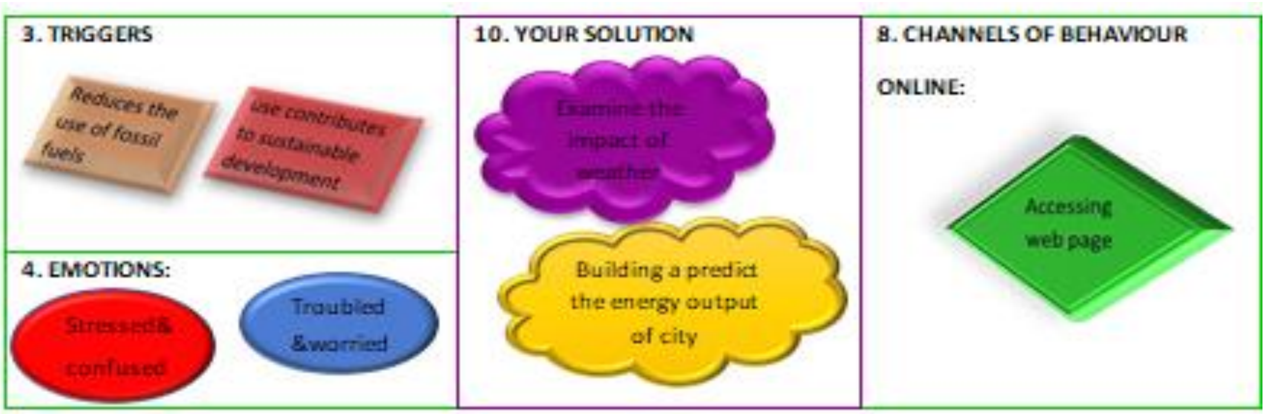
3.3: Proposed Solution

SI.NO	PARAMETER	DESCRIPTION
1.	Problem statement (problem to be solved)	Now, meteorologists have to manually take down every value and then calculate the value for theoretical power. This a very timetaking process and there are chances foe human errors. As this decides how much energy will be produced, any king of error will cost a huge amount to the government. Also,there is no fixed formulafor calculating theoretical power. They dependon number of factors. Hence, we have come up with the solution such thatthe work for meteorologist is decreased and also efficiency is increased.
2.	Idea/solution desceription	Our aim is to map weather data to energy production. We wish to showthat even data that is publicly availablefor weather stations close to wind farms can be used to give prediction of the energy output. Furthermore, we examine the impact of different weather conditions on the energy output of techniques to predict the energy output of wind farms. We are building to predict the energy output of wind turbines and weather conditins of a city.
3.	Novelty / uniqueness	Wind energy is a source of renewable energy. It reducesthe use of fossil fuels, which are the origin of greenhouse gases that cause global warming. Producing electricity through wind energy and its efficient use contribute to sustainable development. The uniqueness of wind energy: Renewable energy Inexhaustible Not pollutant Reduces the use of fossils fuels zReduce energy imports Creates wealth and local employment
4.	Social impact/ customer satisfaction	The environmental impact of electricity generation from wind power is minor when compared to that of fossil fuel power. Habitat loss and fragmentation are the greatest impacts of wind farms on wildlife. Onshore wind farms can have significant impacts on the landscape, as typically they need to be spread over more than other power stations. It also generate noise and at a residential distance of 300 metres this may be around 45dB. Construction of offshore wind farms may create underwater noise.

5.	Businesss model (Revenue model)	Wind energy projects provide many economic benefits. Direct employment Land leasepayments Local taxrevenue Wind energytourism
6.	Scalability of the solution	This model can be used as API in mobile app or web building. We are developing a web application which is built using node red service. We make use of the scoring endpoint to give user input values to be deployed model. The model prediction is then showcased on user interface to predict the energy output of wind turbine

3.4: Problem SolutionFit





4: REQUIREMENT ANALYSIS

4.1 : Functional Requirements

FR No.	Functional Requirement (Epic)	Sub Requirement (Story/ Sub-Task)
FR-1	User Registration	Registration through Form
FR-2	User Confirmation	Confirmation viaEmail
FR-3	Essentiality	• City name • Wind speed • Wind direction • Weather condition
FR-4	Output	Energy Predicated in KWh

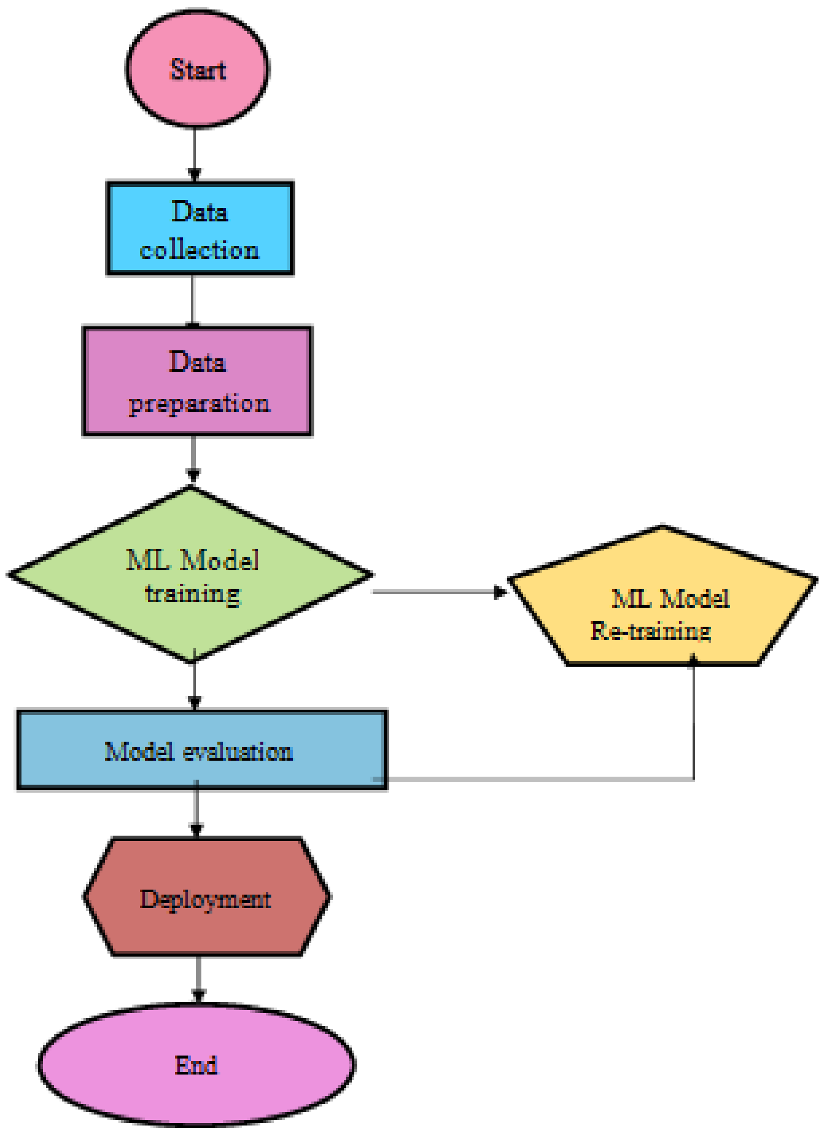
4.2 : Non – FunctionalRequirements

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	• Easy to learn • User friendly • Efficient
NFR-2	**Security**	Privacy - User can have Own accounts to securetheirdata.
NFR-3	**Reliability**	Wind Energy is reliable because it is both unlimitedand domestic
NFR-4	**Performance**	Accuracy is high due to combination of multiple MLmodels to predict the output .
NFR-5	**Availability**	This is a web based application so we canaccess inany device that have a web browser with good Internet facility.
NFR-6	**Scalability**	It can be extended further to provide API which canbe used by third party organizations such as Industries, Powersuppliers , Governmental ,etc.

5 : PROJECT DESIGN

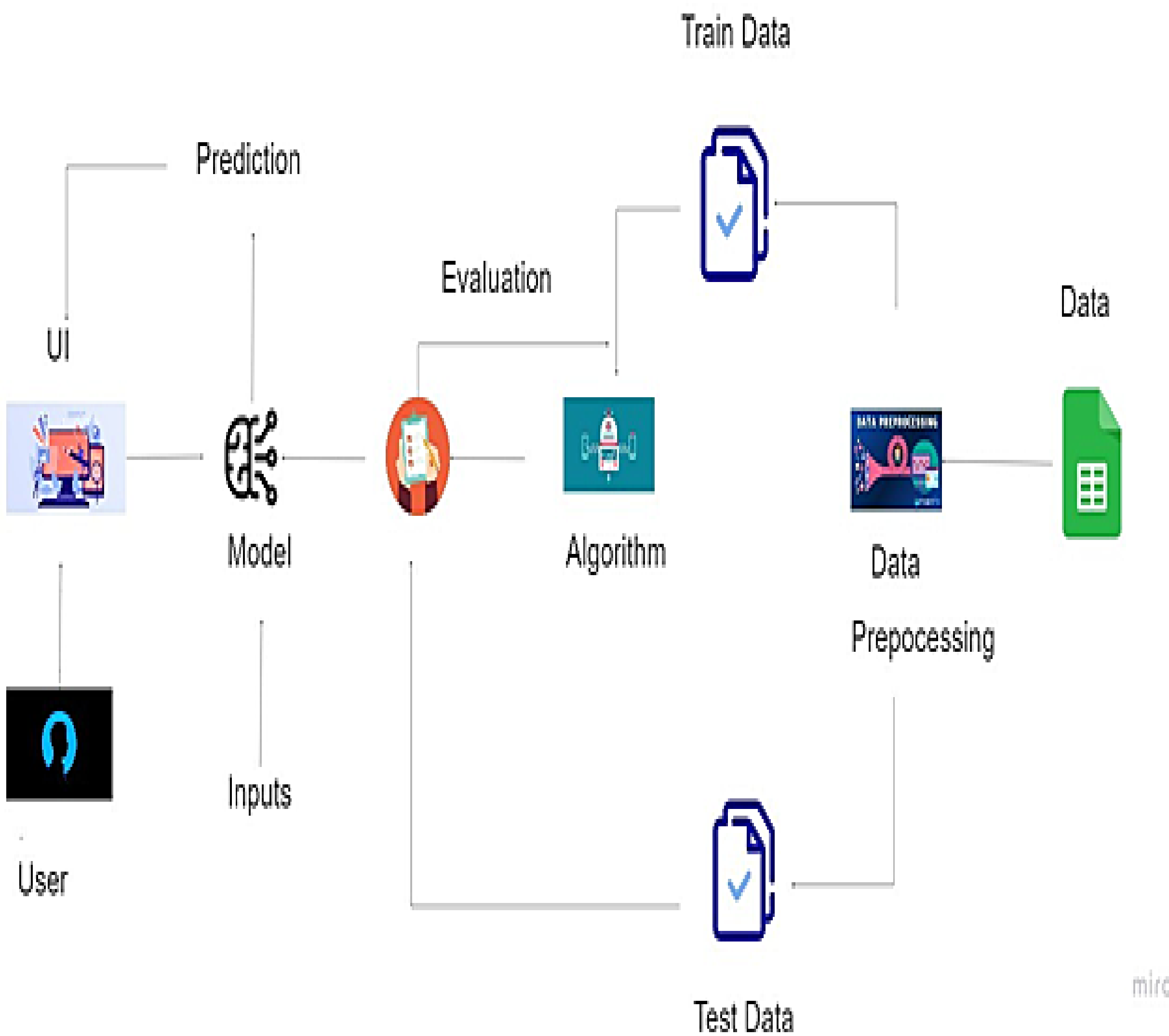
5.1 : Data Flow Diagrams

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



5.2: Solution and Technical Architecture

User Type	Functional Requirement (Epic)	User StoryNumber	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a customer, I can register for the application by entering my email, password,	I can accessmy account /dashboard	High	Sprint-1
	Login	USN-2	As a customer, I can loginto the applicationby entering correct email and password. I can access my a	I can accessmy account/dashboar d.	High	Sprint-1
	Dashboard	USN-3	As a customer, I can see all theorders raisedby me.	I get all the info needed inmy dashboard.	Low	Sprint-2
	Order creation	USN-4	As a customer, I can placemy order withthe detailed description of my query	I can askmy query	Medium	Sprint-2
	Address Column	USN-5	As a customer, I can haveconversati ons withthe assigned agent and get my queries Clarified	My queries are clarified	High	Sprint-2
	Forgot password	USN-6	As a customer, I can reset my password by this option incaseI forgot my old password.	I get accessto myaccount again	Medium	Sprint-2
	Order details	USN-7	As a Customer,I can see the current folder	I get a better understand	Medium	Sprint-2


Train Data
Prediction
Evaluation
Data
UI
Model
Algorithm
Data
Prepocessing
User
Inputs
Test Data
miro

DATA FLOW DIAGRAM AND USER STORIES

Admin (Mobile user)	Login	USN-1	As a admin, I can log into the application byentering Correct e-mail and password	I can access my account	High	Sprint-1
	Dashboar d	USN-2	As an admin I can see all the orders raised inthe entire system and lot more	I can assign agents byseeing those order.	High	Sprint-1
	Agent creation	USN-3	As an admin I can createan agent for clarifying the customers queries	I can create agents.	High	Sprint-2
	Assignme nt agent	USN-4	As an adminI can assign an agent for eachorder created by the customer.	Enable agent to clarifythe queries.	High	Sprint-1
	Forgot password	USN-5	As an admin I can reset my password by thisoption in case I forgot my old password.	I get access to my account.	High	Sprint-1

6 : PROJECT PLANNINGAND SCHEDULING

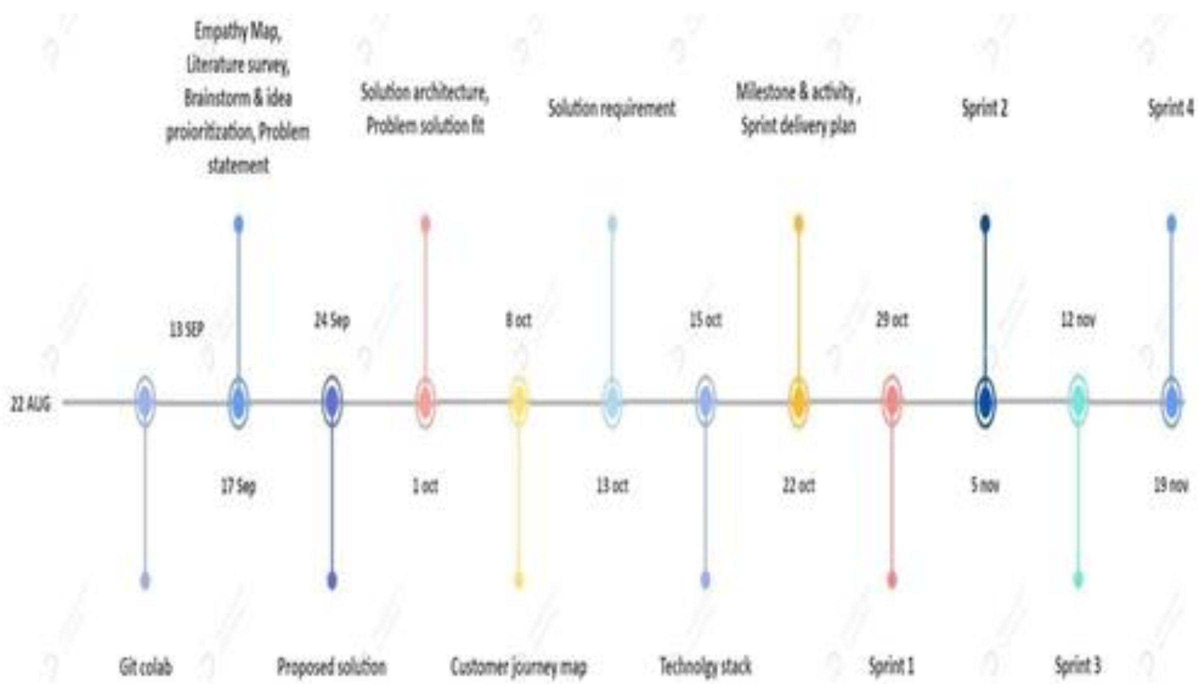
6.1 : Milestone and Activities List

ACTIVITY NO	ACTIVITY NAME	DETAILED ACTIVITY DESCRIPTION	Assigned to
1.	Preparation phase	Access the resources (courses) in project dashboard. Access the guided project workspace. Create github account & collaborate with project resporitory. Setup the laptop /computers based on the perquisites for each technology track.	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gnana Olivu R
2.	**Ideation phase**		
2.1.	Literature survey	Literature survey on the selected project & information gathering.	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gnana Olivu R
2.2.	Define the problem statement	Prepare the list of problem statement to understand the user needs.	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gnana Olivu R
2.3.	Empathy map	Preparation of empathy map canvas to capture the user pains & gains	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gnana Olivu R
2.4.	Brainstorm and idea prioritization	List the ideas by organizing the brainstorming session prioritize the top 3 ideas based on the feasibility and importance	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gnana Olivu R
3.	**Project design phase I**		
3.1.	Proposed solution	Preparation of proposed solution document, which	Ragha sudha R Mahalakshmi M Arthika G

		includes the novelty, feasibility of idea, business model, social impact, scalability of solution.	Srimitha E Mariya Gnana Olivu R
3.2.	Problem solution fit	Prepared problem is analyzed and make effective solution of the problem.	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gnana Olivu R
3.3.	Solution architecture	Prepare an architecture for solution.	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gnana Olivu R
4.	**Project design phase II**		
4.1.	Requirement analysis	Prepare the functional and non- functional requirement.	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gnana Olivu R
4.2.	Customer journey	Preparation of customer journey maps to understand the user interaction and experience with the application(entry to exit)	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gnana Olivu R
4.3.	Data flow diagram	Prepare the data flow diagram for project use level 0	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gnana Olivu R
4.4.	Technology architecture	Prepare technology architecture of the solution.	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gnana Olivu R
5.	**Project planning phase**		
5.1.	Milestone and task	Prepare milestone and activity list	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gnana Olivu R
5.2.	Sprint schedules	Prepare sprint delivery plan	Ragha sudha R Mahalakshmi M Arthika G

			Srimitha E Mariya Gnana Olivu R
6.	**Project development phase**		
6.1.	Coding & solutioning	Sprint-1 Delivery: Develop the code, Test and push it to Github.	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gnana Olivu R
6.2.	Acceptance testing	Sprint-2 Delivery: Develop the code, Test and push it to Github. Sprint-3 Delivery: Develop the code, Test and push it to Github.	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gnana Olivu R
6.3.	Performance testing	Sprint-4 Delivery: Develop the code, Test and push it to Github.	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gnana Olivu R

6.2 : Milestone With Timeline Chart



6.3 : Sprint Delivery Schedule

Product Backlogs, Sprint Schedule And Estimation

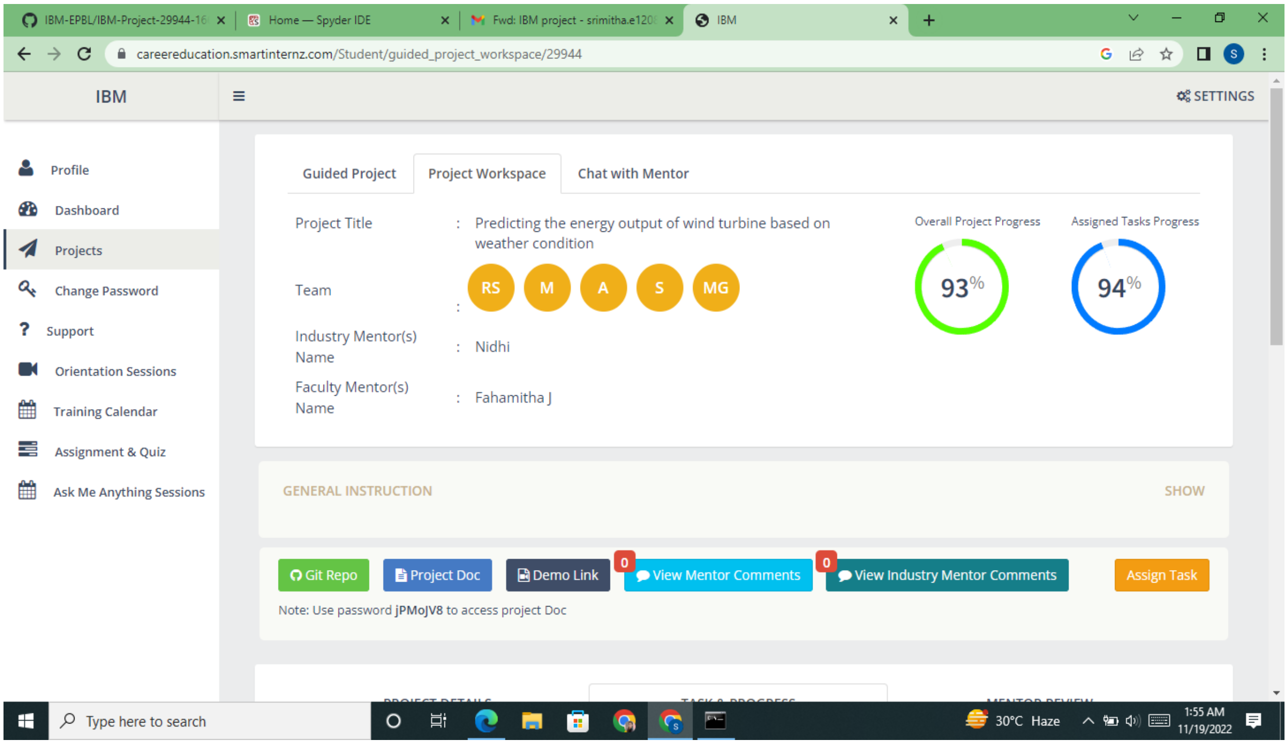
Sprint	Functional Requirement (EPIC)	User Story Number	User Story/ Task	Story Points	Priority	Team Members
Sprint 1	Registration	USN-1	As a user ,I can register for the application by entering email, password and conforming my password	5	High	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gana olivu R
Sprint 1		USN-2	As a user, I will receive confirmation email once I have registered for the application	4	High	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gana olivu R
Sprint 1		USN-3	As a user ,I can register for application through phone number.	4	Low	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gana olivu R
Sprint 1		USN-4	As a user, I can register for the application through Gmail	3	Medium	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gana olivu R
Sprint 1	Login(User)	USN-5	As a user, I can log into application by entering email& password	5	High	Ragha sudha R Mahalakshmi M Arthika G Srimitha E

						Mariya Gana olivu R
Sprint 2	Dashboard	USN-6	Once I have logged in , I can see my dashboard.	6	Medium	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gana olivu R
Sprint 2	Web access	USN-7	As a customer I can access the website to predict the weather conditions.	7	High	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gana olivu R
Sprint 2	Prediction	USN-8	As a customer when I enter the weather details the website should predict the approximate weather conditions.	7	High	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gana olivu R
Sprint 3	Analysis	USN -9	As a customer, I wish to store my prediction and make analysis.	10	Medium	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gana olivu R
Sprint 3	Security	USN-10	As a customer I expect my data to be secured.	10	Medium	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Ganana olivu R
Sprint 4	Database Access	USN- 11	An administrator I should maintain the website. And update the website regularly.	20	Low	Ragha sudha R Mahalakshmi M Arthika G Srimitha E Mariya Gana olivu R

Project Tracker,Velocity & BurndownChart:

Sprint	Total story points	Duration	Sprint startdate	Sprint enddate	Story points completed(as on planned End date)	Sprint Release date(Actual)
Sprint 1	20	6 days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint 2	20	6 days	31 Oct 2022	5 Nov 2022	20	5 Nov 2022
Sprint 3	20	6 days	7 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint 4	20	6 days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.4 : Reports from JIRA



7 : CODING AND SOLUTIONING

7.1 : Feature 1

Dataset taken for training

Date/Time	LV ActivePower (kW)	Wind Speed (m/s)	Theoretical_Power_Curve (KWh)	Wind Direction (°)
01 01 2018 00:00	380.0478	5.311336	416.3289	259.9949
01 01 2018 00:10	453.7692	5.672167	519.9175	268.6411
01 01 2018 00:20	306.3766	5.216037	390.9	272.5648
01 01 2018 00:30	419.6459	5.659674	516.1276	271.2581
01 01 2018 00:40	380.6507	5.577941	491.703	265.6743
01 01 2018 00:50	402.392	5.604052	499.4364	264.5786
01 01 2018 01:00	447.6057	5.793008	557.3724	266.1636
01 01 2018 01:10	387.2422	5.30605	414.8982	257.9495
01 01 2018 01:20	463.6512	5.584629	493.6777	253.4807
01 01 2018 01:30	439.7257	5.523228	475.7068	258.7238
01 01 2018 01:40	498.1817	5.724116	535.8414	251.851
01 01 2018 01:50	526.8162	5.934199	603.0141	265.5047
01 01 2018 02:00	710.5873	6.547414	824.6625	274.2329
01 01 2018 02:10	655.1943	6.199746	693.4726	266.7332
01 01 2018 02:20	754.7625	6.505383	808.0981	266.7604
01 01 2018 02:30	790.1733	6.634116	859.459	270.4932
01 01 2018 02:40	742.9853	6.378913	759.4345	266.5933
01 01 2018 02:50	748.2296	6.446653	785.281	265.5718
01 01 2018 03:00	736.6478	6.415083	773.1729	261.1587
01 01 2018 03:10	787.2462	6.437531	781.7712	257.5602
01 01 2018 03:20	722.8641	6.220024	700.7647	255.9265
01 01 2018 03:30	935.0334	6.898026	970.7366	250.0129
01 01 2018 03:40	1220.609	7.609711	1315.049	255.9857
01 01 2018 03:50	1053.772	7.288356	1151.266	255.4446

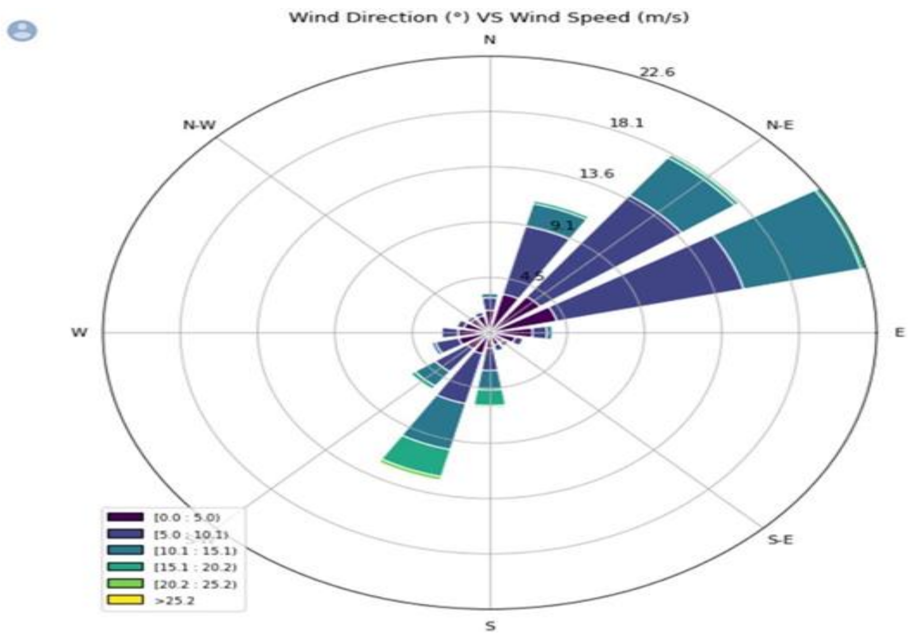
Dataset Description :

```
[ ] data.describe()
```

	ActivePower(kW)	WindSpeed(m/s)	TheoreticalPowerCurve(KWh)	WindDirection
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

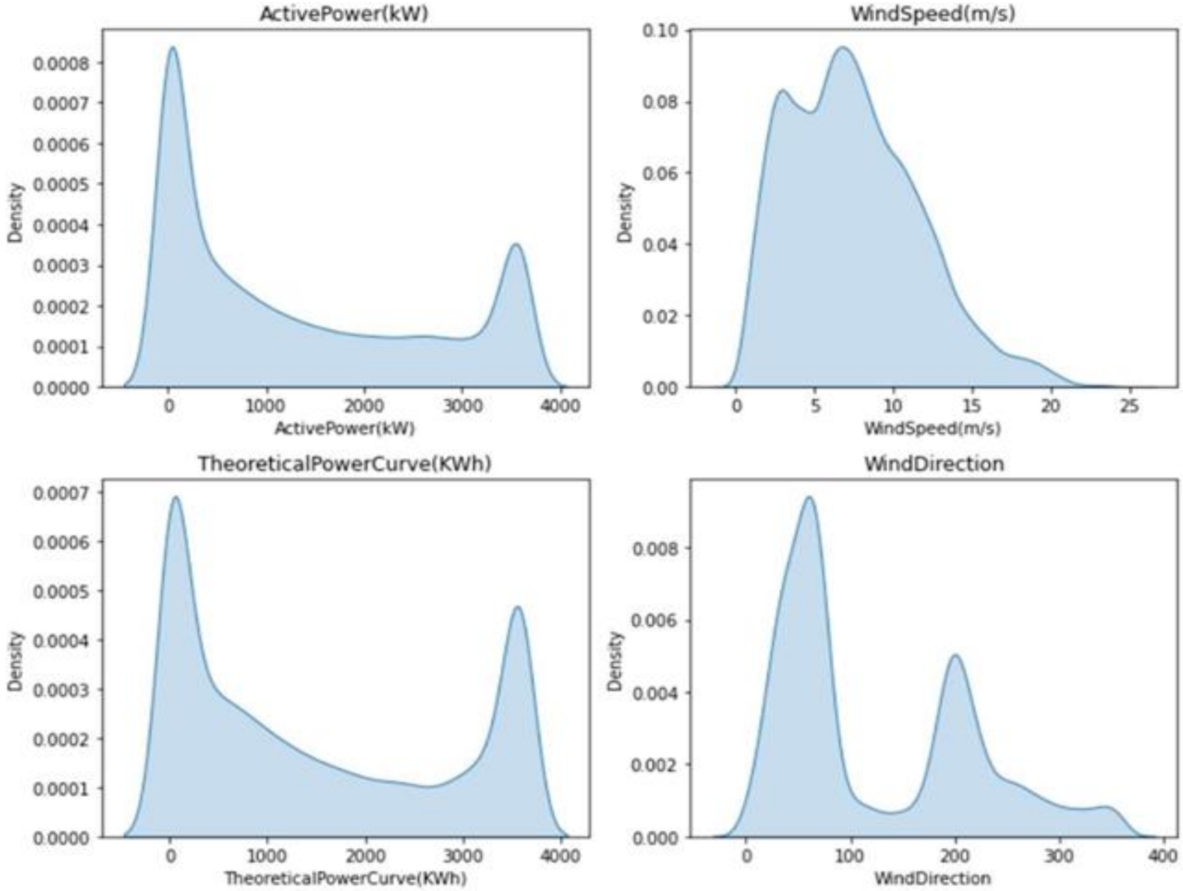
This command displays the various parameters like count, mean, Standard deviation, minimum value, maximum value for the four attributes.

Wind Direction vs Wind Speed



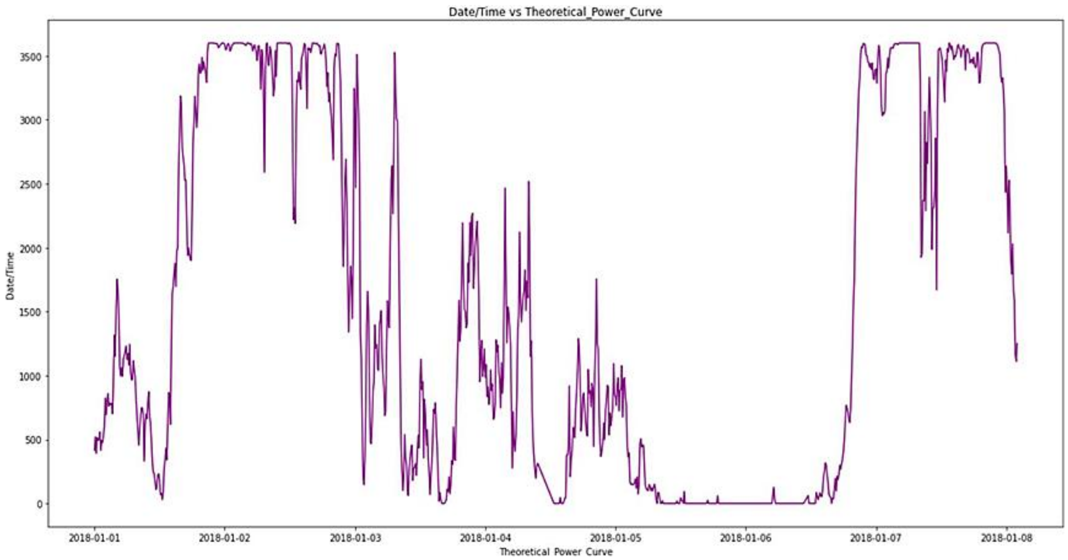
The plot gives a visualization of the direction of the Wind at the location over the year. It also shows the speed of the wind in that particular direction. From this we can infer that this particular Wind Mill experiences wind in a North-easterly direction primarily and South-westerly direction occasionally. The wind speed varies between 5 and 20 m/s.

Feature Inference:



The above graph plots the weightage of each attribute of the dataset. It helps to understand the dataset quickly and easily. The usual Wind speed in the region can be seen as 2to 12 m/s. The prominent Wind direction is 30° to 75° and mildly along 190° to 210° measures from magnetic north. The actual power generated is also less compared to the theoretical power calculated with the wind speed. This is due to the mechanical and aerodynamic losses faced by the wind mill.

Output Power Visualization :



This is a graph plotted with time as x-axis and power generated in y-axis. 1000 samples (8 days) data has been taken for viewing the plot clearly. It shows the trend in power generation. On one day there is maximum output and next two days the power output is less owing to low wind speed.

Data Pre-processing:

```
data.shape
```

```
(50530, 5)
```

This command returns the dimension of our dataset. We have 50530 rows and 5 columns which are the features of the dataset.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50530 entries, 0 to 50529
Data columns (total 5 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   Date/Time                      50530 non-null  object
 1   ActivePower(kW)                50530 non-null  float64
 2   WindSpeed(m/s)                 50530 non-null  float64
 3   TheoreticalPowerCurve(KWh)     50530 non-null  float64
 4   WindDirection                  50530 non-null  float64
dtypes: float64(4), object(1)
memory usage: 1.9+ MB
```

This command returns whether our dataset has any null values and the datatype of the features. From the output we can see that there is no null-data type in the dataset and the values are of 64 bit floating point integer.

Splitting Data :

```
X=data[['WindSpeed(m/s)','WindDirection']]
X.head()
```

```
y = data['ActivePower(kW)']
y.head()
```

```
0    380.047791
1    453.769196
2    306.376587
3    419.645905
4    380.650696
Name: ActivePower(kW), dtype: float64
```

```
X.to_csv('IndependentVariables.csv')
y.to_csv('DependentVariable.csv')
```

The features are then split as dependent and independent variable for training the model. Wind speed and wind direction is taken as independent variables whereas Active power generated is taken as dependent variable.

Importing the RegressionModels :

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.metrics import accuracy_score,r2_score,mean_squared_error
xgr=XGBRegressor()
rf=RandomForestRegressor()
lr=LinearRegression()
dt=DecisionTreeRegressor()
sm=SVR()
```

The above command is used to import the required libraries to train the various models. Here we use five regression models for training namely Linear Regressor, XGBRegressor, Random Forest Regressor, DecisionTree Regressor and Support Vector Regression.

Fitting the Models with dataset :

```
model_xg=xgr.fit(X_train,y_train)
y_xg=model_xg.predict(X_test)
model_rf=rf.fit(X_train,y_train)
y_rf=model_rf.predict(X_test)
model_lr=lr.fit(X_train,y_train)
y_lr=model_lr.predict(X_test)
model_dt=dt.fit(X_train,y_train)
y_dt=model_dt.predict(X_test)
model_sm=sm.fit(X_train,y_train)
y_sm=model_sm.predict(X_test)
```

The above command is used to train the data. The five models are being fitted individually with the training data.

Checking the Metrics:

```
print('R2-xgb',r2_score(y_test,y_xg))
print('RMSE-xgb',np.sqrt(mean_squared_error(y_test,y_xg)))

print('R2-rf',r2_score(y_test,y_rf))
print('RMSE-rf',np.sqrt(mean_squared_error(y_test,y_rf)))

print('R2-lr',r2_score(y_test,y_lr))
print('RMSE-lr',np.sqrt(mean_squared_error(y_test,y_lr)))

print('R2-dt',r2_score(y_test,y_dt))
print('RMSE-dt',np.sqrt(mean_squared_error(y_test,y_dt)))

print('R2-svm',r2_score(y_test,y_sm))
print('RMSE-svm',np.sqrt(mean_squared_error(y_test,y_sm)))
[14]
```

```
R2-xgb 0.9222746826171284
RMSE-xgb 364.85477293970644
R2-rf 0.9097702879938478
RMSE-rf 393.10952377367164
R2-lr 0.8368251429450982
RMSE-lr 528.6465476346768
R2-dt 0.8388459591904157
RMSE-dt 525.3628747175155
R2-svm 0.005368134807760105
RMSE-svm 1305.1786596858901
```

This command prints the score of all the five models that we have fitted. It displays the accuracy of each of the model. From the above statement we can see that XGBRegressor model has the highest accuracy of 92%.

XGBRegressor Model Training:

```
xg=XGBRegressor(colsample_bylevel=0.4, colsample_bytree=0.3, gamma=0.1,
                learning_rate=0.01, max_depth=6, min_child_weight=25,
                n_estimators=1500, reg_alpha=0.1, reg_lambda=0.8, subsample=0.6)
x=xgr.fit(X_train,y_train)
y1=x.predict(X_test)
r2_score(y_test,y1)
```

```
[07:14:27] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.

0.9222746826171284
```

Since XGBRegressor model best fits the model, we select that and give our dataset to obtain the trained model.

Saving the Model:

In [11]:
```
joblib.dump(forest_model, "power_prediction.sav")
```

Out[11]: ['power_prediction.sav']

In [12]:
```
df
```

Out[12]:

	Time	ActivePower(KW)	WindSpeed(m/s)	Theoretical_Power_Curve (KWh)	Wind Direction (°)
0	NaT	380.047791	5.311336	416.328908	259.994904
1	NaT	453.769196	5.672167	519.917511	268.641113
2	NaT	306.376587	5.216037	390.900016	272.564789
3	NaT	419.645905	5.659674	516.127569	271.258087
4	NaT	380.650696	5.577941	491.702972	265.674286
...	...	...	...	...	...
50525	NaT	2963.980957	11.404030	3397.190793	80.502724
50526	NaT	1684.353027	7.332648	1173.055771	84.062599
50527	NaT	2201.106934	8.435358	1788.284755	84.742500
50528	NaT	2515.694092	9.421366	2418.382503	84.297913
50529	NaT	2820.466064	9.979332	2779.184096	82.274620

50530 rows × 5 columns

In [13]:
```
import pickle
pickle.dump(forest_model,open("model.pkl","wb"))
```

This command saves our trained model as a .bin file. This file can then be called upon by our application to perform the prediction. This model accepts Wind speed and Wind Direction as input and gives the power generated as output.

7.2 : Feature 2

Deploying the Model in IBM Cloud :

IBM Deployment

```
In [18]: !pip install -U ibm-watson-machine-learning

Requirement already satisfied: ibm-watson-machine-learning in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (1.0.255)
Collecting ibm-watson-machine-learning
  Downloading ibm_watson_machine_learning-1.0.257-py3-none-any.whl (1.8 MB)
     |████████████████████████████████| 1.8 MB 31.6 MB/s eta 0:00:01
```

Here the required libraryof IBM Watson Machine Learningis getting installed.

Authenticate and set Space

t1xJwH_pNvesyStso2tawTlpypHX0HEQJVMev99cmAtK

```
In [28]: wml_credentials = {
             "apikey":"iJ8fO2zR1zKFzMmJarCCyrgkg2xF1jaKtkVucFJAQJ1h",
             "url":"https://eu-de.ml.cloud.ibm.com"
         }

In [29]: wml_client = APIClient(wml_credentials)

In [30]: wml_client.spaces.list()
         #e0a978b3-0ab3-4800-987d-a39e08695233

Note: 'limit' is not provided. Only first 50 records will be displayed if the number of records exceed 50
------------------------------------  -----------  ------------------------
ID                                    NAME         CREATED
e0a978b3-0ab3-4800-987d-a39e08695233  Wind Energy  2022-11-07T04:44:34.908Z
------------------------------------  -----------  ------------------------

In [32]: SPACE_ID= "e0a978b3-0ab3-4800-987d-a39e08695233"

In [33]: wml_client.set.default_space(SPACE_ID)

Out[33]: 'SUCCESS'
```

Using the unique API key generated in IBM Cloud and mentioning our server location. Using the API credentials a new space is created in IBM Watson. The space has its unique Space id.

```
In [36]: import sklearn
         sklearn.__version__

Out[36]: '1.0.2'

In [37]: MODEL_NAME = 'XGB_1'
         DEPLOYMENT_NAME = 'XGB_1'
         DEMO_MODEL = model_xg

In [38]: # Set Python Version
         software_spec_uid = wml_client.software_specifications.get_id_by_name('runtime-22.1-py3.9')

In [39]: software_spec_uid

Out[39]: '12b83a17-24d8-5082-900f-0ab31fbfd3cb'
```

Downloading the required ML model. Lookingfor the version that is being supported by IBM and downloading the correct version.Creating a new deployment space for the model.

```
In [40]: # Setup model meta
model_props = {
    wml_client.repository.ModelMetaNames.NAME: MODEL_NAME,
    wml_client.repository.ModelMetaNames.TYPE: 'scikit-learn_1.0',
    wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid
}
```

```
In [41]: #Save model

model_details = wml_client.repository.store_model(
    model=DEMO_MODEL,
    meta_props=model_props,
    training_data=X_train,
    training_target=y_train
)
```

To set up the model requirements and link it to the deployment space. Saving the modelto the space by mentioning the attributes of the model.

```
In [42]: model_details
```


```
Out[42]: {'entity': {'hybrid_pipeline_software_specs': [],
  'label_column': 'ActivePower(kW)',
  'schemas': {'input': [{'fields': [{'name': 'WindSpeed(m/s)',
       'type': 'float64'},
      {'name': 'WindDirection', 'type': 'float64'}],
     'id': '1',
     'type': 'struct'}],
   'output': []},
  'software_spec': {'id': '12b83a17-24d8-5082-900f-0ab31fbfd3cb',
   'name': 'runtime-22.1-py3.9'},
  'type': 'scikit-learn_1.0'},
 'metadata': {'created_at': '2022-11-07T04:56:31.773Z',
  'id': '7dd1db0c-ed59-4f73-b91b-e04cffd42347',
  'modified_at': '2022-11-07T04:56:34.488Z',
  'name': 'XGB_1',
  'owner': 'IBMid-666002NS6H',
  'resource_key': 'ae81f1ad-fa3a-4cb8-8dee-014487923830',
  'space_id': 'e0a978b3-0ab3-4800-987d-a39e08695233'},
 'system': {'warnings': []}}
```


To view the details of the model created.

```
In [44]: # Set meta
deployment_props = {
    wml_client.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT_NAME,
    wml_client.deployments.ConfigurationMetaNames.ONLINE: {}
}
```

To set the configuration of the deployment. Giving the name for the deployment in IBMWatson.

```
In [45]: # Deploy
deployment = wml_client.deployments.create(
    artifact_uid=model_id,
    meta_props=deployment_props
)
```

```
#######################################################################################

Synchronous deployment creation for uid: '7dd1db0c-ed59-4f73-b91b-e04cffd42347' started

#######################################################################################


initializing
Note: online_url is deprecated and will be removed in a future release. Use serving_urls instead.

ready


------------------------------------------------------------------------------------------------
Successfully finished deployment creation, deployment_uid='48a87a28-d849-4ab0-9203-ca3924b43312'
------------------------------------------------------------------------------------------------
```

Deploying the model in IBM Cloud using model id. An id is created for the model using which the model can be accessed online.

Flask Application :

```python
import flask
from flask import request, render_template
from flask_cors import CORS
import joblib
import pandas as pd
from xgboost import XGBRegressor
import requests
app = flask.Flask(__name__, static_url_path='')
CORS(app)
```

To import the required libraries.

```python
API_KEY = "iJ8fO2zR1zKFzMmJarCCyrgkg2xF1jaKtkVucFJAQJ1h"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":
 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}
```

The API key and model id are used to link to the model that has been trained in IBM Cloud.

```python
@app.route('/', methods=['GET'])
def sendHomePage():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predictSpecies():
    ws = float(request.form['ws'])
    wd = float(request.form['wd'])

    X = [[ws,wd]]
    xgr=XGBRegressor()
    df = pd.DataFrame(X, columns=['WindSpeed(m/s)','WindDirection'])
    payload_scoring = {"input_data": [{"field": [['ws', 'wd']], "values":X}]}

    response_scoring = requests.post('https://eu-de.ml.cloud.ibm.com/ml/v4/deployments/782741b9-1e46-
    headers={'Authorization': 'Bearer ' + mltoken})
    print(response_scoring)
    predictions = response_scoring.json()
    print(predictions)
    predict = predictions['predictions'][0]['values'][0][0]
    print("Final prediction :",predict)
    return render_template('predict.html',predict=predict)

if __name__ == '__main__':
    app.run()
```

This program serves as the backend for our Web page API and linking our Machine Learning model with it. The input that has been received from the home page is then sent to out ML model to do the prediction and the output will be displayed at the next web page. It isthe connection between the Frontend and backend.

HTML Code :

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>WIND TURBINE ENERGY PREDICTION</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/index.css') }}">
  </head>
  <body>
    <div class="container">
      <div class="glass">
        <h1 class="text" >WIND TURBINE <br>ENERGY PREDICTION</h1>
      <h2 class="text">Using XGBoost Model</h2>
        <br>
      <form method="POST" action="/predict">
        <p class="text">Wind Speed</p>
        <input name="ws" required />
        <p class="text">Wind Direction</p>
        <input name="wd" required />
        <br />
        <br />
        <button type="submit" class="submit">Submit</button>
      </div>
    </div>
  </body>
</html>
```

Code to design the home page. The page consists of a from where in the user can enter the wind speed and Wind directions. When submitted the values are given to the model.

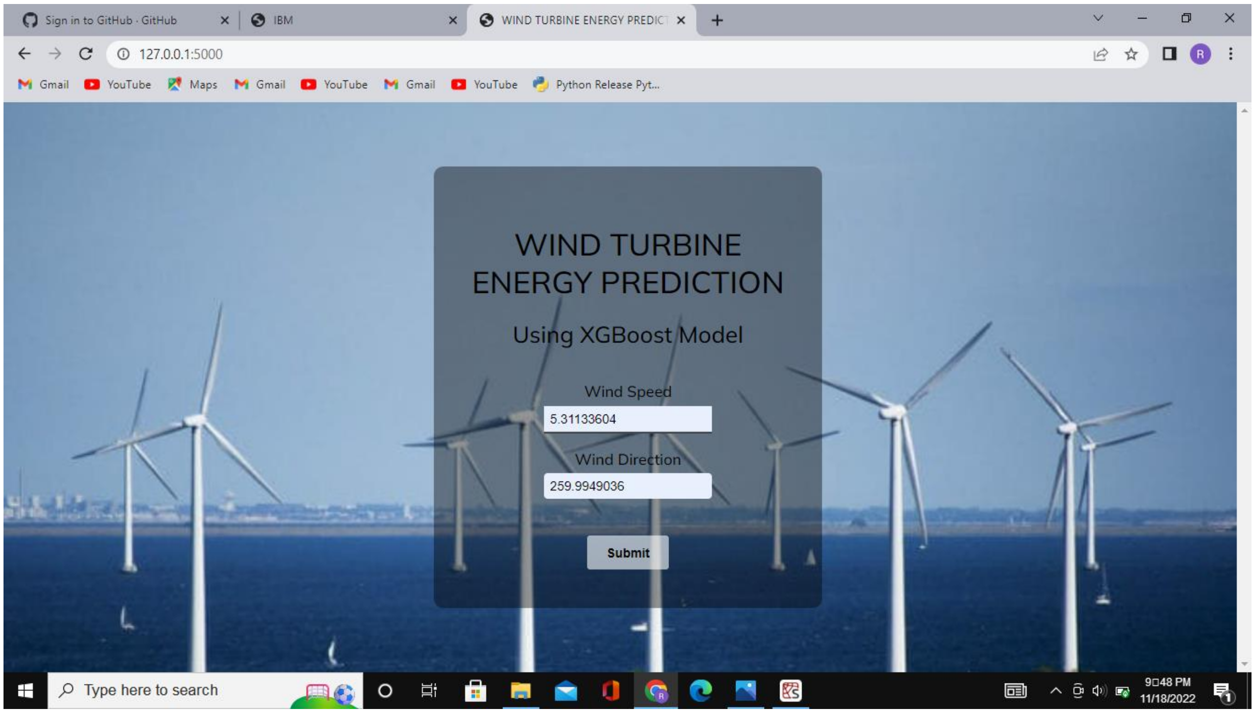
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="./css/index.css" />
    <title>Prediction</title>
</head>
<body>

    <div class="container">
        <div class="glassdoor">
            <h1 class="text">The predicted Output power is</h1>
            <h1 class="highlight">{{predict}}</h1>
            <a href="/" class="submit">Go Back</a>
        </div>
      </div>
</body>
</html>
```

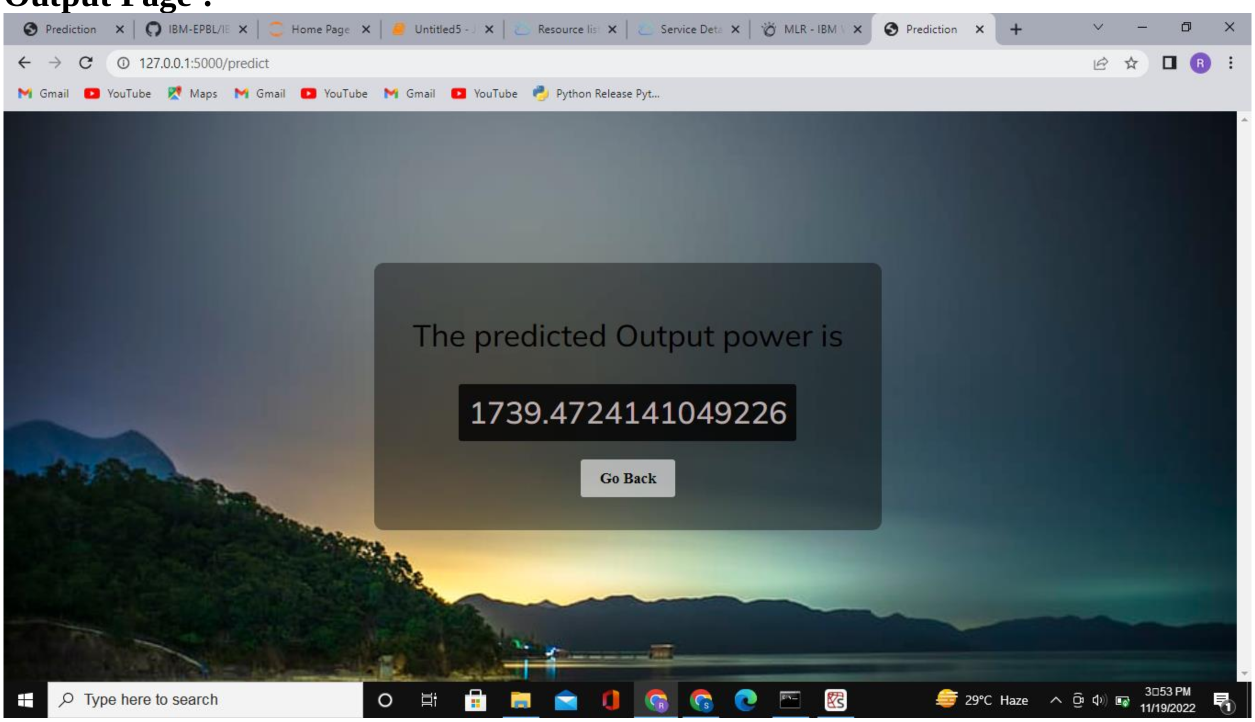
This page displays the output predicted value. This is a post method and hence receives the value from model and displays on the web page.

Web Page Design :

Home Page :



Output Page :



8 : TESTING

8.1 : Test Cases

Wind Speed (m/s)	Wind Direction (°)	Predicted Power Output (KW)
10.5	100.9	2695.02
6.6	290	751.88
30.7	220	3303.57
25.5	45	3595.69
19.1	0	1135.50
14.8	295	3758.29
8.3	180	1524.59
0.5	88	6.82
3.7	325	34.03
35.2	355	3819.80

8.2 : User Acceptance Testing

The project has been tested extensively with a number of users. The users found the interface very easy to use. The Web pages were colourful and attractive. There was no unnecessary details in the web page. It was clean and simple that any new user could master it. The data input format was also simple. The user need not enter any unit. He could simply enter the value. The prediction time is fairly low at an average time of 3 seconds. This delay primarily varies depending on the internet connectivity. The model has been hosted in IBM cloud. Thus with the API available, the model can be accessed remotely from any system provided IBM access key is given. The model predicts the power output close to the actual power generated. The users are satisfied with the predicted output power. Although the prediction is not very accurate it comes closer to the actual power. Various inputs have been given by the users to test the consistency of the model. The model proved itself and all the users accepted the model as a reliable and convenient

9 : RESULTS

9.1 : Performance Metrics

The XGBRegressor ML model that we have used have used here has better performance in speed and accuracy compared to other models. We have compared the performance metrics of 5 models and selected this as the best for the application. The model performed well for all the test cases. The model performed good with no glitches or lag found during the testing phases.

10 : Advantages and Disadvantages

10.1 : Advantages

This model takes in the previous years energy outputs and corelate it with the weather and other parameters that affected it. By using this model we can give the Weather conditions as input and obtain the energy output. It also dynamically alters the algorithm based on the predicted value and actual output value. This model helps in increasing the usage of renewable energy. It optimizes the operation of Wind Turbines.T he cost of Implementing this solution makes it an Unformidable one. Wind Energy Companies will be able to increase their energy output there by increasing revenue. Wind Energy can be trusted as a consistent source as we are able to predict the total power output for any given time. This doesn't require any additional equipment to be set up at the Wind turbine. The existing Sensors can be used to get the Weather parameters for predicting the power output. With Weather stations all across the world, the data can be obtained easily in real time. The prediction can be carried out at the control station of the Wind mills. The algorithm can be easily modified to work for every single Wind Turbine.

10.2 : Disadvantages

Wind Mill companies hesitate to completely rely on this model. Data availability is difficult for all the individual Wind Mills. The Wind Mill may be in a remote location, providingconnectivity to all of it proves challenging and expensive. Data Storage cost is very high, asthe data for the output power and other attributes will be stored in the cloud. This is expensive for the company. The model needs Weather inputs for the prediction process. Error in this input values like Wind speed, Wind Direction, Temperature, Altitude, Humidity due to the inaccuracy in the instruments that is being can result in errors in prediction. Sudden changes in weather conditions prove difficult for the model to predict. The changing Climatic conditions across the globe every year, means that the previous year data is insignificant. Efficiency loss at the wind mill is difficult to calculate and it varies from one wind mill to the other. Human made changes like building infrastructures in the wind path can greatly affect the prediction which cannot be given as input. Server crash or loss of internet can leave the company with no other choice as the entire model is hosted incloud.

11 : CONCLUSION

The XGBRegressor ML model that has been used above performs well for our dataset.The model is fast and consumes less resources. The API developed is also simple and user- friendly. By using this model, we could predict the output power of a wind turbine provided the required input parameter. This increases the use of Wind power and revenuefor thecompanies. The model is not 100% accuratebut it performs sufficiently. It can be concluded as the power output cannot be predicted very accurately as there are several parameters that could affect the output and all those outputs cannot be taken in for training as it can result in a very complex and overtrained model. The features that have high weightage are considered in this model.

12 : FUTURE SCOPE

The further works that can be done in this project is to include more features in model training to study the effect on the output. A long history of data (dataset of more than 3 years) can be used for training for increased accuracy. The application can be upgraded such that the input values from the sensors are directly fed to the model without the user entering it manually.More web pages can be designed so that the user can control more Wind Mill in the same API. Navigation tabs to move across various Wind mills. The dashboard can be made for User Interactive by making it to show real time graph of the prediction and actual power. Diagnosis of wind mill which performthe least can be done remotely.

13 : APPENDIX

13.1 : Source Code

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

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):
df
sns.pairplot(df)

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)

df["Time"] = pd.to_datetime(df["Time"], format = "%d %m %Y %H %M", errors = "coerce")

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:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error,r2_score
from xgboost import XGBRegressor
forest_model = RandomForestRegressor
```

```
forest_model = RandomForestRegressor(n_estimators = 750, max_depth = 4, max_leaf_nodes
= 500, random_state = 1)
forest_model.fit(train_X, train_y)
```

```
RandomForestRegressor(max_depth=4, max_leaf_nodes=500, n_estimators=750,
                      random_state=1)
```

```
power_preds = forest_model.predict(val_X)


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

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

```
['power_prediction.sav']
```

```
Df

import pickle
pickle.dump(forest_model,open("model.pkl","wb"))
```

IBM Cloud Deployment:

```
!pip install -U ibm-watson-machine-learning
from ibm_watson_machine_learning import
APIClientimport json
wml_credentials = {
   "apikey":"iJ8fO2zR1zKFzMmJarCCyrgkg2xF1jaKtkVucFJAQJ1h",
   "url":"https://eu-de.ml.cloud.ibm.com"
}
wml_client = APIClient(wml_credentials)
wml_client.spaces.list()
SPACE_ID= "e0a978b3-0ab3-4800-987d-a39e08695233"
wml_client.set.default_space(SPACE_ID)
wml_client.software_specifications.list(100)

import sklearnsklearn.
version
MODEL_NAME = 'XGB_1'
DEPLOYMENT_NAME = 'XGB_1'
DEMO_MODEL = model_xg
software_spec_uid = wml_client.software_specifications.get_id_by_name('runtime-
22.1-py3.9')
```

```
software_spec_uid

model_props = {

wml_client.repository.ModelMetaNames.NAME: MODEL_NAME,
wml_client.repository.ModelMetaNames.TYPE: 'scikit-learn_1.0',
wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:
software_spec_uid
}

model_details = wml_client.repository.store_model(
    model=DEMO_MODEL,
    meta_props=model_props, training_data=X_train,
    training_target=y_train
)

model_details
model_id = wml_client.repository.get_model_id(model_details)model_id
deployment_props = {
    wml_client.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT_NAME,
    wml_client.deployments.ConfigurationMetaNames.ONLINE: {}
}
deployment = wml_client.deployments.create(
    artifact_uid=model_id,
    meta_props=deployment_props
)
```

FLASK Application

```
import flask
from flask import request,render_template
from flask_corsimport CORS
import joblib
import pandas as pd
from xgboost import XGBRegressor
import requests
app = flask.Flask(_name_, static_url_path='')
CORS(app)

API_KEY = "iJ8fO2zR1zKFzMmJarCCyrgkg2xF1jaKtkVucFJAQJ1h"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey":
 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken= token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer' + mltoken}
@app.route('/', methods=['GET'])
```

```
def sendHomePage():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predictSpecies():
    ws = float(request.form['ws']) wd
    = float(request.form['wd'])

    X = [[ws,wd]]
    xgr=XGBRegressor()
    df = pd.DataFrame(X, columns=['WindSpeed(m/s)','WindDirection'])
    payload_scoring = {"input_data": [{"field": [['ws','wd']], "values":X}]}
    response_scoring = requests.post('https:/south.cloud.ibm.com/ml/v4/deployments/782741b9-
1e46-4126-943a- f0696c250c0e/predictions?version=2022-11-07'          ,
json=payload_scoring,
    headers={'Authorization': 'Bearer' + mltoken})
    print(response_scoring)
    predictions = response_scoring.json()
    print(predictions)
    predict = predictions['predictions'][0]['values'][0][0]
    print("Final prediction :",predict)
    return render_template('predict.html',predict=predict)

if __name__== '_main_':
    app.run()
```

Home Web Page :

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>WIND TURBINE ENERGYPREDICTION</title>
    <style>
    @importurl('https://fonts.googleapis.com/css2?family=Mulish:ital,wght@0,400;0,500;0,60;1
        ,400;1,500;1,600&display=swap');
html, body {
overflow-y: scroll;
overflow-x: hidden;padding: 0;
margin: 0;
  }
  body {
    height: 100vh;width:
    100vw;
  }
  body {
    scrollbar-gutter: 10px;
```

```
}
.container { height:
  100%;
  width: 100%;

  background-imageurl(".jpg");
  background-size: cover;
  background-repeat: no-repeat;

}
.container,form{
    display:flex;
    justify-content:
    center;align-items:
    center; flex-direction:
    column;
}
.glass,.glassdoor{
    padding:40px;
    background-color: rgba(0,0,0,.4);
    border-radius: 10px;
}
.glassdoor{
    height:200px;
    display: flex;
    flex-direction:
    column;align-items:
    center;
    justify-content:space-evenly;
    gap:10px;
}
input{
    margin-top: 5px;
    outline: 0; border:
    none;
    border-bottom: rgba(0,0,0,.7) 2px solid;
    background: transparent;
   padding: 6px;color:white;
    }
input:focus
{
margin-top: 5px;
    background-color: rgba(0,0,0,.45);
    border-bottom: transparent 2px solid;
    border:none;
    outline: 0;
    border-radius: 4px;padding:
    6px;
```

```
}
.text{
   font-family: "Mulish";
   color:rgba(255,255,255,.8);
   margin-bottom: 0;
   font-weight: 500;
   text-align: center;
}
.highlight{
   font-family: "Mulish";
   color:rgba(225, 214, 214, 0.8);
   margin-bottom: 10px;
   font-weight: 500;
   padding: 10px;
   background-color: rgba(0,0,0,.8);
   border-radius: 3px;
}
.submit{
   padding:10px    20px;
   border-radius:    3px;
   border: 0;
   background-color:rgba(255,255,255,.6);
   font-weight: 600;
}
.submit:hover{
   cursor:pointer;
}
a{
   outline:none;
   text-decoration:
   none;color:inherit;
}
   </style>
</head>
<body>
    <div class="container">
      <div class="glass">
        <h1 class="text" >WINDTURBINE <br>ENERGY PREDICTION</h1>
      <h2 class="text">Using XGBoostModel</h2>
        <br>
 <form method="POST" action="/predict">
        <p class="text">Wind Speed</p>
        <input name="ws" required/>
        <p class="text">Wind Direction</p>
        <input name="wd" required/>
        <br />
```

```
<br />
<button type="submit" class="submit">Submit</button>
</div>
</div>
</body>
</html>
```

Output Web Page :

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Prediction</title>
<style>
@importurl('https://fonts.googleapis.com/css2?family=Mulish:ital,wght@0,400;0,500;0,60;1
,400;1,500;1,600&display=swap');
html, body {
overflow-y: scroll;
overflow-x: hidden;padding: 0;
margin: 0;
}
body {
height: 100vh;width:
100vw;
}
body {
scrollbar-gutter: 10px;
}
.container { height:
100%;
width: 100%;

background-imageurl(".jpg");
background-size: cover;
background-repeat: no-repeat;

}
.container,form{
display:flex;
justify-content:
center;align-items:
center; flex-direction:
column;
```

```
}
.glass,.glassdoor{
   padding:40px;
   background-color: rgba(0,0,0,.4);
   border-radius: 10px;
}
.glassdoor{
   height:200px;
   display: flex;
   flex-direction:
   column;align-items:
   center;
   justify-content:space-evenly;
   gap:10px;
}
input{
   margin-top: 5px;
   outline: 0; border:
   none;
   border-bottom: rgba(0,0,0,.7) 2px solid;
   background: transparent;
  padding: 6px;color:white;
   }
input:focus
{
margin-top: 5px;
   background-color: rgba(0,0,0,.45);
   border-bottom: transparent 2px solid;
   border:none;
   outline: 0;
   border-radius: 4px;padding:
   6px;
}
.text{
   font-family: "Mulish";
   color:rgba(255,255,255,.8);
   margin-bottom: 0;
   font-weight: 500;
   text-align: center;
}
.highlight{
   font-family: "Mulish";
   color:rgba(225, 214, 214, 0.8);
   margin-bottom: 10px;
   font-weight: 500;
   padding: 10px;
   background-color: rgba(0,0,0,.8);
   border-radius: 3px;
```

```
}
.submit{
   padding:10px    20px;
   border-radius:    3px;
   border: 0;
   background-color:rgba(255,255,255,.6);
   font-weight: 600;
}
.submit:hover{
   cursor:pointer;
}
a{
   outline:none;
   text-decoration:
   none;color:inherit;
}
</style>
</head>
<body>
       <div class="container">
      <div class="glassdoor">
         <h1 class="text">The predictedOutput power is</h1>
         <h1  class="highlight">{{predict}}</h1>
         <a href="/" class="submit">Go Back</a>
      </div>
     </div>
</body>
</html
```

15 : GitHub & ProjectDemo Link

GitHub Repo : https://github.com/IBM-EPBL/IBM-Project-29944-1660134947

Project Demo Link :

https://drive.google.com/file/d/1dfNPncnvNqM3bvokuEwuFsHSSAut9jHR/view?usp=drivesdk