



# **1 : INTRODUCTION**

## **1.1 : Project Overview**

Predict the output power of a wind turbine given weather conditions at any time. Using Machine Learning to predict energy output based on previous performance data and real-time weather parameters will aid in integrating with the grid and realizing its full potential. The wind speed and direction can be entered as inputs, and the model will predict the turbine's output power. To find the best-fitting model, various machine learning models were evaluated.

## **1.2 : Purpose**

Because of the unpredictability of wind speed and direction (weather condition). As a result, the power generated by a wind turbine is irregular and unpredictable. Season, temperature, yearly currents, humidity, pressure, location, altitude, height of the turbine, blade size, blade pitch, and many other factors influence power generation. Because of the erratic nature of the output power, integrating this renewable energy source into the grid is extremely difficult. As a result, wind farms lose revenue because they are unable to supply power to the grid at the appropriate time.

# **2 : LITERATURE SURVEY**

## **2.1 : Existing Problem**

The existing solution has been investigated. It has been deduced that the wind mill's output energy is roughly predicted manually or using some basic software. Because of the numerous variables influencing the output, the obtained result is frequently unreliable, with a very high margin of error. There are thousands of Wind Mills all over the world. Even when two wind turbines are located very close to each other, the output performance varies significantly. A generalised, yet customizable, solution is required to meet the needs of a single wind turbine.

## 2.2 : References

S.NO	Paper Title	Author	Journal Name	Publication year	Description
1	Energy Modelling Output of Wind System based on Wind Speed	P. R. Anisha, C. Kishor Kumar Reddy & Nuzhat Yasmeen	Springer	2021	In this paper, they have connected a relationship with different parameters to the energy output. To deal with the interaction of the different parameters, we use random forest regression of machine learning algorithms.
2	Predicting the Energy Output of Wind Turbine Based on Weather Condition	Abdelkader Harrouz, Ilhami Colak, Korhan Kayisli	IEEE	2019	The main problem in wind energy is that it become quite difficult to predict the power output exactly and this hinders the accuracy. In this paper, there are three different wind models are modelled and simulated with choosing the complete and correct models.
3	A review of wind power and wind speed forecasting methods with different time horizons	Saurabh S. Soman, Hamidreza Zareipour , Om Malik	IEEE	2010	This paper provides insight on the foremost forecasting techniques, associated with wind power and speed, based on numeric weather prediction (NWP), statistical approaches, artificial neural network (ANN) and hybrid techniques over different time-scales.
4	Day-Ahead Wind Power Forecasting in Poland Based on Numerical Weather Prediction	Bogdan Bochenek , Jakub Jurasz , Adam Jaczewski , Gabriel Stachur, Piotr Sekuła	MDPI	2021	This paper examines the possibility to predict day-ahead wind power based on different machine learning methods not for a specific wind farm but at national level.

5	Predicting the Wind Turbine Power Generation based on Weather Conditions	S Preethi; H Prithika; M Pramila; S Birundha	IEEE	2021	In this paper, an end-to-end web application has been developed to predict and forecast the wind turbine's power generation based on the weather conditions. The prediction model has been developed using Bidirectional Long Short-Term Memory which is a unique kind of RNN (Recurrent Neural Network).
---	--	--	------	------	---

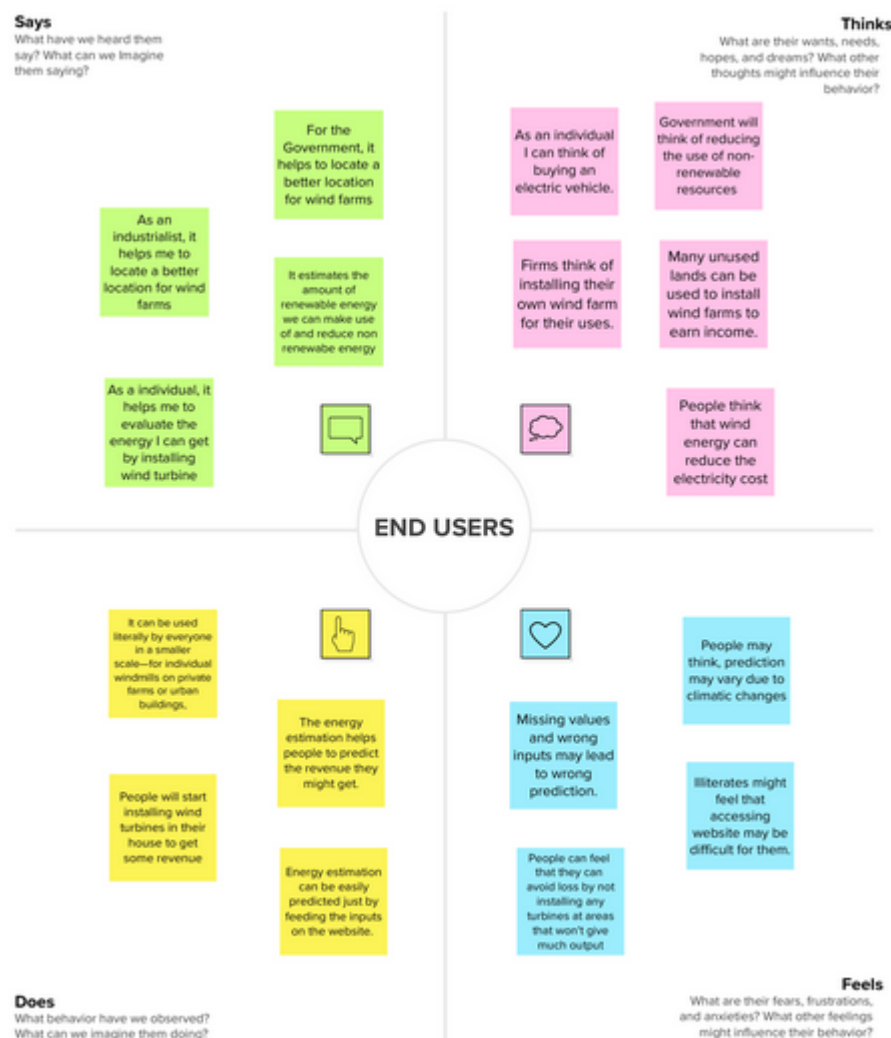
6	The Use of Machine Learning and Performance Concept to Monitor and Predict Wind Power Output	Kelvin Palhares Bastos Sathler; Athanasios Kolios	IEEE	2022	Here, the model to analyse the performance through Meteorological Mast Data (Met Mast Data) and then uses it as an input to monitor and predict power output. As a result, this model achieves high accuracy and can be key to understanding the wind turbine behaviour throughout its lifespan.
7	The Intelligent Methods Used in Prediction the Wind Speed and Output Power of Wind Farm	Xinyan Zhang; Chongchong Chen; Weiqing Wang; Yi Dai	IEEE	2012	In this paper, for prediction method they used BP neural network, wavelet BP neural network. The simulation results shows that the method used in this paper can give a better prediction.
8	Forecasting of Wind Turbine Output Power Using Machine learning	Haroon Rashid; Waqar Haider; Canras Batunlu	IEEE	2020	In this paper, they have predicted the output power of the wind turbines using the random forest regressor algorithm. The model is trained using the data from 2017. The wind direction, wind speed and outdoor temperature are used as input parameters to predict output power.

## 2.3 : Problem Statement Definition

Predict the output power of a wind turbine given weather conditions at any time. Using Machine Learning to predict energy output based on previous performance data and real-time weather parameters will aid in integrating with the grid and realising its full potential. Accurate wind power forecasting reduces the need for additional balancing energy and reserve power for wind power integration. Accurate power output is required to make efficient use of wind energy. When the power output of a wind mill at a given time is known, we can integrate it with the grid and use this renewable energy source instead of conventional non-renewable sources.


## 3 : IDEATION AND PROPOSED SOLUTION

### 3.1 : Empathy Map Canvas



### 3.2 : Ideation And Brainstorming

Template



## Brainstorm & 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.

10 minutes to prepare

1 hour to collaborate

2-3 people recommended

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.

10 minutes

1

Team gathering

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

2

Set the goal

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

3

Learn how to use the facilitation tools

Use the Facilitator Superpowers to run a happy and productive session.

Open article

→

Pragadesh M

Wind power is gaining popularity due to its renewable nature and environmental friendliness.

we can carry out publicly available weather and energy data for a wind farm.

we predict energy based on weather data and analyze the important parameters.

We map weather data to energy prediction and derive analysis.

Santhosh M

A wind farm's energy output is strongly dependent on the weather conditions and its location.

A real time prediction system of output power is crucial for a wind farm.

We investigate the effect of various weather conditions on wind farm energy output.

Fuzzy model approach provides an interpretable model structure

Haran M

We should forecast the precise output to avoid costly overproduction.

Energy suppliers can more efficiently coordinate the collaborative for various energy sources using the accurate output.

We analyze the correlation of different components that characterize the weather conditions

We collect the historical data through the Supervisory Control and Data Acquisition systems of wind farms and their fitting curves

Nirmal kumar K

We use symbolic regression modeling to cope up with interaction of the different parameters.

Rotor RPM wind direction is taken into consideration for determination

User can upload their own real time dataset (.csv or .xlsx format) for forecasting.

We check frequency of wind speed and determine it's output energy

Pragadesh M

Diameter of the rotor of a wind turbine plays a major role

Wind directions, wind speed and its outdoor temperature are input parameters

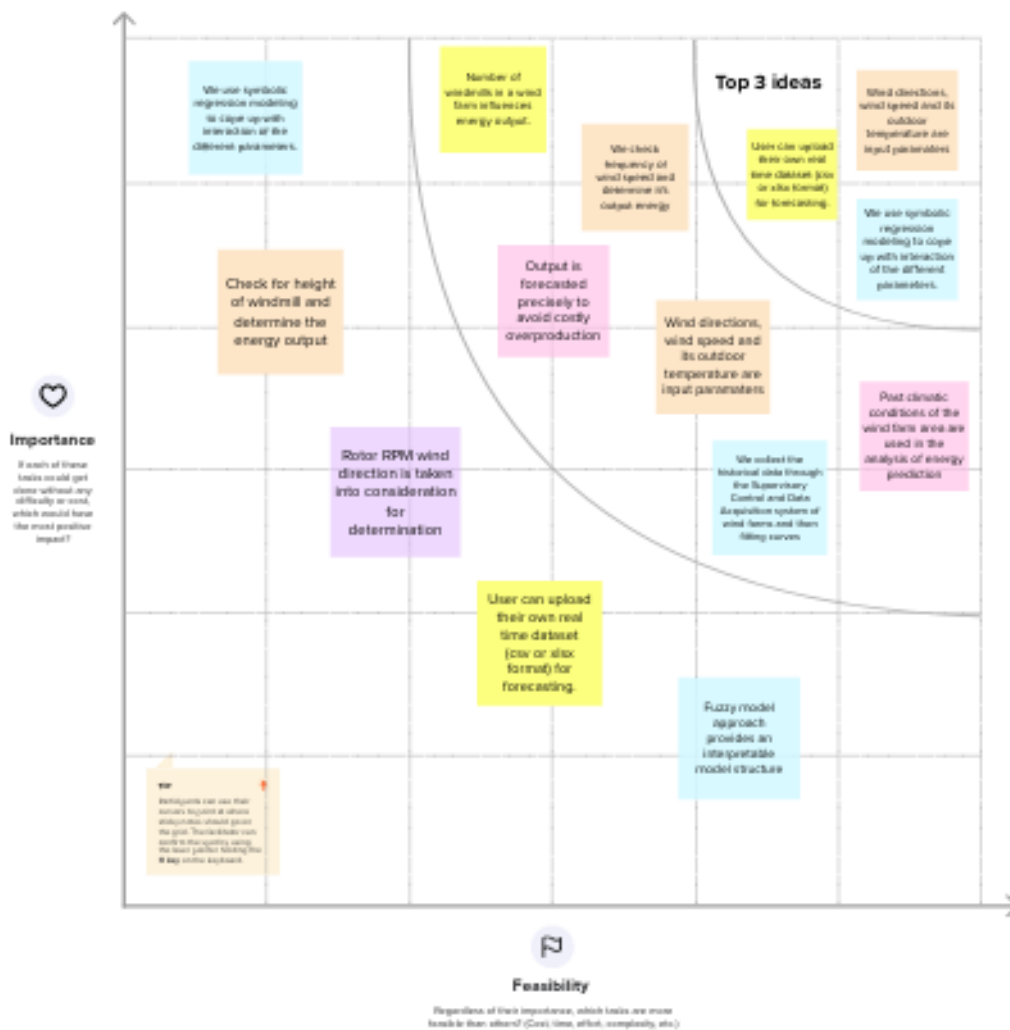
Number of windmills in a wind farm influences energy output.

Past climatic conditions of the wind farm area are used in the analysis of energy prediction

### Group ideas

20 minutes

Continuously update the algorithm with the actual and predicted value



### 3.3 : Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Wind Energy is the widely used Renewable energy source, but it is not a sustained source. The power generated is affected by various environmental factors. Thus it cannot be relied upon completely, thereby reducing its efficiency.
2.	Idea / Solution description	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.
3.	Novelty / Uniqueness	This model takes in the previous years energy outputs and correlate 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.
4.	Social Impact / Customer Satisfaction	This model helps in increasing the usage of renewable energy. It optimizes the operation of Wind Turbines. The cost of Implementing this solution makes it an Unformidable one.
5.	Business Model (Revenue Model)	Wind Energy Companies will be able to increase their energy output thereby 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.
6.	Scalability of the Solution	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 to get accurate results.



## 3.4 : Problem Solution Fit

Project Design Phase-I - Solution Fit

Team ID:PNT2022TMID31617

Predicting the energy output of Wind Turbine based on Weather conditions

Define CS, fit into CC	<p><b>1. CUSTOMER SEGMENT(S)</b> <span>CS</span></p> <p>Who is your customer?</p> <ul style="list-style-type: none"> <li>➤ Individuals</li> <li>➤ Electricity suppliers</li> <li>➤ Industrialist</li> <li>➤ Government</li> </ul>	<p><b>6. CUSTOMER CONSTRAINTS</b> <span>CC</span></p> <p>What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.</p> <ol style="list-style-type: none"> <li>1. Illiterates may feel difficulty in accessing the website.</li> <li>2. Network connection</li> <li>3. Feeding missing or wrong inputs</li> </ol>	<p><b>5. AVAILABLE SOLUTIONS</b> <span>AS</span></p> <p>Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros &amp; cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking</p> <p>Manual calculations based on past climatic conditions which consumes large amount of time were tried in the past.</p> <p>Pros: Consumes less time Cost-effective</p> <p>Cons: Network connectivity</p>	Explore AS, differentiate
Focus on J&P, tap into BE, understand RC	<p><b>2. JOBS-TO-BE-DONE / PROBLEMS</b> <span>J&amp;P</span></p> <p>Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.</p> <p>Since there's no proper platform for wind energy prediction, we predict the energy output of wind turbine in order to earn some revenue and to locate a better place for wind farms.</p>	<p><b>9. PROBLEM ROOT CAUSE</b> <span>RC</span></p> <p>What is the real reason that this problem exists? What is the back story behind the need to do this job? i.e. customers have to do it because of the change in regulations.</p> <p>Failures occur because of locating wind farms in unsuitable environment.</p>	<p><b>7. BEHAVIOUR</b> <span>BE</span></p> <p>What does your customer do to address the problem and get the job done?</p> <p>Since wind speed is constantly changing, so is the wind's energy content. The amount of fluctuation depends on the local surface conditions and obstructions as well as the weather.</p>	Focus on J&P, tap into BE, understand RC

## 4 : REQUIREMENT ANALYSIS

### 4.1 : Functional Requirements

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via meeting Confirmation via mail
FR-3	User Requirements	Knowledge about inputting the data Teaching on using the ML Model
FR-4	User Infrastructure	A system to support ML data modelling A Suitable GPU and CPU
FR-5	User Network	Network infrastructure to connect the wind mill to the Control station
FR-6	User Cost	User has to spend only for attaining the software, additional components are not required

### 4.2 : Non – Functional Requirements

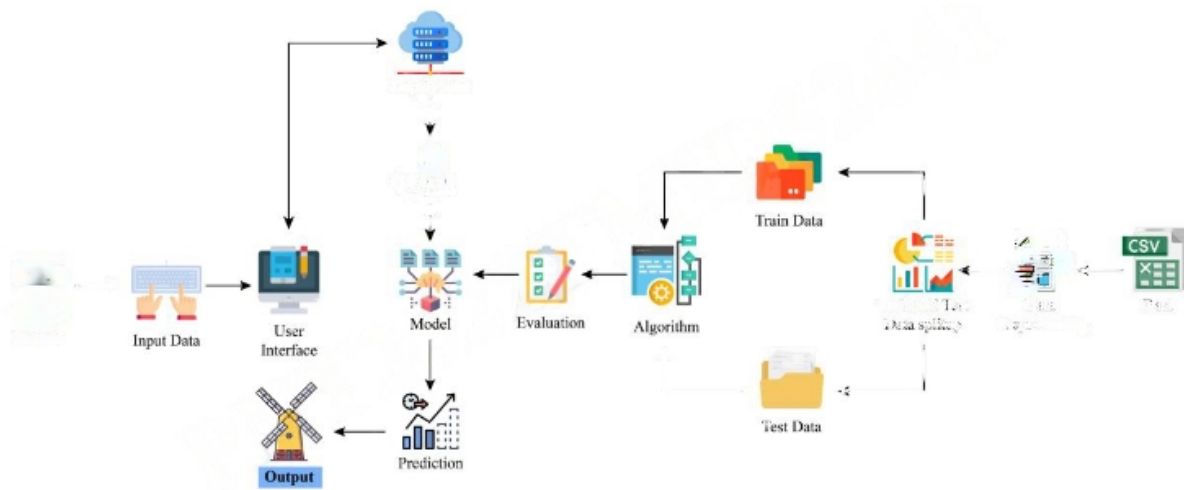
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	It can be used for various Wind mill and trained for specific models.
NFR-2	Security	The data can be stored in a secure cloud.
NFR-3	Reliability	The predicted will be accurate and can be relied upon.
NFR-4	Performance	The performance of the model depends on the Computer it runs on the accuracy of the data it is fed with.
NFR-5	Availability	It is available as a software package.
NFR-6	Scalability	It can be scaled up and interconnected with other Wind Mills to create a connected Wind Form.

## 5 : PROJECT DESIGN

## 5.1 : Data Flow Diagrams

1. The Weather data and wind speed data is fed as input to the ML model.
2. The previous years data stored in a cloud is fed as input to the ML model.
3. The expected power production is predicted.
4. The predicted value is sent to the user, and also stored in cloud.
5. The predicted output power is compared with the actual power generated.
6. The error in prediction is calculated and used as a feedback to train the model.
7. The model accuracy increases over the course.

## 5.2 : Solution and Technical Architecture



**Table-1 : Components & Technologies:**

S.No	Component	Description	Technology
1.	User Interface	API	HTML, CSS, JavaScript / Angular Js / React Js etc.
2.	Application Logic-1	Data Pre-processing	Java / Python
3.	Application Logic-2	Data Input	IBM Watson STT service
4.	Database	Previous Year data	MySQL, NoSQL, etc.
5.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
6.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
7.	External API	Purpose of External API used in the application	IBM Weather API, etc.
8.	Machine Learning Model	Purpose of Machine Learning Model	Weather prediction Model, etc.
9.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration:	Local, Cloud Foundry, Kubernetes, etc.

**Table-2: Application Characteristics:**

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	FLASK
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	SHA-256, Encryptions, IAM Controls, OWASP etc.
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	Cloud
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	Distributed cloud service
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	SDN

### 5.3 : User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user I can buy the ML model and train it customize according to the needs.	I can access my account / dashboard	High	Sprint-1
		USN-2	My Identity can be verified through a mail.	I can receive confirmation email & click confirm	High	Sprint-1
	Login	USN-1	As a user, I can log into the application by entering email & password.	I can login into the admin window	Low	Sprint-1
		USN-2	For different Wind Mill, various login can be used.	Different id and password can be used for different Wind Mills.	High	Sprint-2
	Dashboard	USN-3	The various functionalities can be viewed and navigated from the dashboard.	The options are clear and easily understandable.	Medium	Sprint-2

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer Care Executive	Queries	USN-1	The Customer Care executive answered my call and guided me.	He was calm and helped me through the process,	Medium	Sprint-2
	Initial Setup	USN-2	After the ML model has been bought the sales executive helped me setup the model.	He was clear and knowledgeable.	High	Sprint-3
Administrator	Remote Access	USN-1	I have remote access to all the models and can be worked on.	The remote access works flawlessly.	Medium	Sprint-4
		USN-2	The integration with cloud was easy and simple.	The cloud access is very good.	Low	Sprint-4

## 6 : PROJECT PLANNING AND SCHEDULING

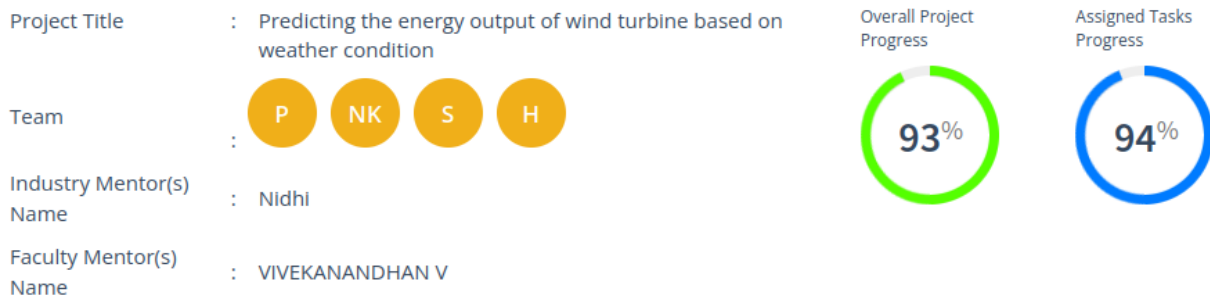
### 6.1 : Sprint Planning and Estimation

Sprint	Milestone
Sprint 1	<ol style="list-style-type: none"><li>1. User Registers into the application through entering Email Id and Password for confirmation.</li><li>2. User Receives a confirmation mail for their registered Email.</li><li>3. User can also register to the application through Mobile number.</li><li>4. User logs in into the website using Email Id and password.</li></ol>
Sprint 2	<ol style="list-style-type: none"><li>1. User can access the dashboard.</li><li>2. User enters the required details of weather conditions to get the desired turbine power output based on our model's prediction.</li></ol>
Sprint 3	<ol style="list-style-type: none"><li>1. Application stores the predictions, that can be used for future analysis.</li><li>2. The data stored has to be maintained securely.</li></ol>
Sprint 4	<ol style="list-style-type: none"><li>1. Administrator should properly maintain the website and update it whenever required.</li></ol>

### 6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	27 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	04 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	09 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	16 Nov 2022

## 6.3: Reports from JIRA



## 7 : CODING AND SOLUTIONING

### 7.1 : Feature 1

**Dataset taken for training :**

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

This is the Excel sheet visualization of the dataset that has been taken for the ML Model. It contains 5 attributes. Date/Time, Active power generated, Theoretical power generated, Wind speed and Wind Direction. The data set has 50,000+ samples. It has data of a single wind mill's power production over a period of one year. The samples are taken at an interval of every 10 mins making 144 samples per day.



## 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 :

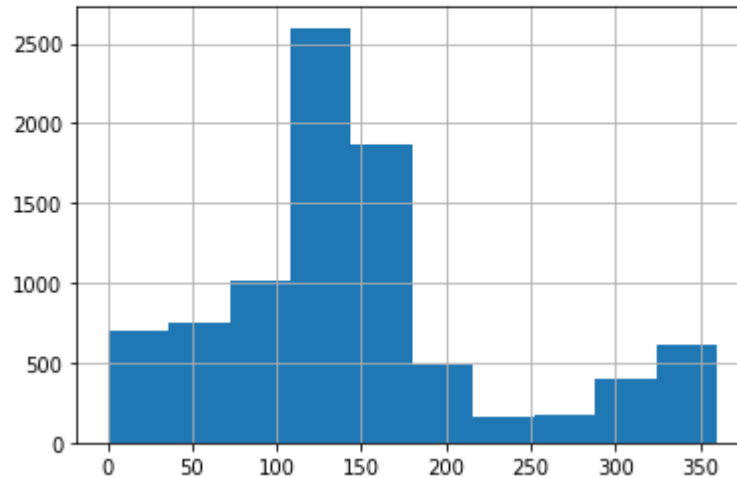


The plot gives a visualization of the direction of the Wind at the location over the year. It also shows the speed of the wind in that particular direction. From this we can infer that this particular Wind Mill experiences wind in a North-easterly direction primarily and south-westerly direction occasionally.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

## Feature Inference :

```
In [5]: df['WindDirection'].hist()
```

```
Out[5]: <AxesSubplot:>
```



The above graph plots the weightage of each attribute of the dataset. It helps to understand the dataset quickly and easily. The usual Windspeed in the region can be seen as 2 to 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.

## Data Pre-processing:

```
data.shape
```

```
(50530, 5)
```

```
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

## Splitting Data :

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

	WindSpeed(m/s)	WindDirection
0	5.311336	259.994904
1	5.672167	268.641113
2	5.216037	272.564789
3	5.659674	271.258087
4	5.577941	265.674286

The features are then split as dependent and independent variables 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 Regression Models :

```
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_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    random_state=6,
                                                    test_size=0.25)
```

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, Random Forest Regressor, Decision Tree Regressor and Support Vector Regression.

## Fitting the Models with dataset :

```
## Ensemble Learning
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import VotingRegressor
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
from xgboost import XGBRegressor
from sklearn.metrics import accuracy_score, r2_score, mean_squared_error
xgr=XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=0.7,
                  colsample_bynode=1, colsample_bytree=0.3, gamma=0.2,
                  importance_type='gain', learning_rate=0.03, max_delta_step=0,
                  max_depth=8, min_child_weight=25, missing=None, n_estimators=800,
                  n_jobs=1, nthread=None, objective='reg:linear', random_state=0,
                  reg_alpha=0.2, reg_lambda=0.8, scale_pos_weight=1, seed=None,
                  silent=None, subsample=0.1, verbosity=1)
sm=SVR(gamma='auto', C=50, epsilon=0.3)
rf=RandomForestRegressor(n_estimators=500, max_depth=4)
lr=LinearRegression()
model=VotingRegressor([('lr',lr), ('rf',rf), ('sm', sm), ('xgr',xgr)], weights=[1,1,2,3])
```

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

## Checking the Metrics :

### 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)))
```

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 model has the highest accuracy of 92%.

**Saving the Model :**

Offset	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+a	+b	+c	+d	+e	+f	Equivalent ASCII characters
00000000	00	00	00	3f	02	00	00	00	00	00	00	00	00	00	00	00	?
00000010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000080	00	00	00	00	00	00	00	00	00	0a	00	00	00	00	00	00	
00000090	72	65	67	3a	6c	69	6e	65	61	72	06	00	00	00	00	00	r e g : l i n e a r
000000a0	00	00	67	62	74	72	65	65	64	00	00	00	01	00	00	00	g b t r e e d
000000b0	02	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

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 library of IBM Watson Machine Learning is getting installed.

#### Authenticate and set Space

t1xJwH\_pNvesyStso2tawTlpypHX0HEQJVMev99cmAtK

```
In [28]: wml_credentials = {
          "apikey": "iJ8f02zR1zKFzMmJarCCyngkg2xF1jaKtkVucFJAQJ1h",
          "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. Looking for 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 model to 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 IBM Watson.

```
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 :

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

To import the required libraries

```
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/predict', methods=['POST'])
def predict():
    int_features = [float(x) for x in request.form.values()]
    final_features = np.array(int_features)
    #prediction = model.predict(final_features)
    t = {"array": final_features}
    t = json.dumps(t, cls=NumpyArrayEncoder)
    payload_scoring = {"input_data": [
{
    "fields" : [ "f0", "f1", "f2", "f3" ],
    "values" : t
}}]}
```

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 our ML model to do the prediction and the output will be displayed at the next web page. It is the connection between the Frontend and backend.

## HTML Code :

```
</head>

<body>
<div class="login">
<h1>Predict Energy output of a Wind Turbine</h1>

    <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">
        <input type="text" name="wind_speed" placeholder="Wind Speed (in m/s)" required="require
        <input type="text" name="wind_direction" placeholder="Wind Direction (in Degrees)" rec
        <input type="text" name="pressure" placeholder="Atmospheric Pressure (in atmos)" required=
        <input type="text" name="air_temperature" placeholder="Air Temperature (in degree Celsius)

        <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
    </form>

    <br>
    <br>
    {{ prediction_text }}
```

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

```
templates > index.html > html > body > div.login > form
<!DOCTYPE html>
<html >
<!-- From https://codepen.io/frytyler/pen/E6dtg-->
<head>
    <meta charset="UTF-8">
    <title>Wind Turbine Energy Predictor</title>
    <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
    <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
    <link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
    <link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet'
    <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

</head>

<body>
<div class="login">
<h1>Predict Energy output of a Wind Turbine</h1>

    <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">
        <input type="text" name="wind_speed" placeholder="Wind Speed (in m/s)" required="require
        <input type="text" name="wind_direction" placeholder="Wind Direction (in Degrees)" rec
        <input type="text" name="pressure" placeholder="Atmospheric Pressure (in atmos)" required=
        <input type="text" name="air_temperature" placeholder="Air Temperature (in degree Celsius)

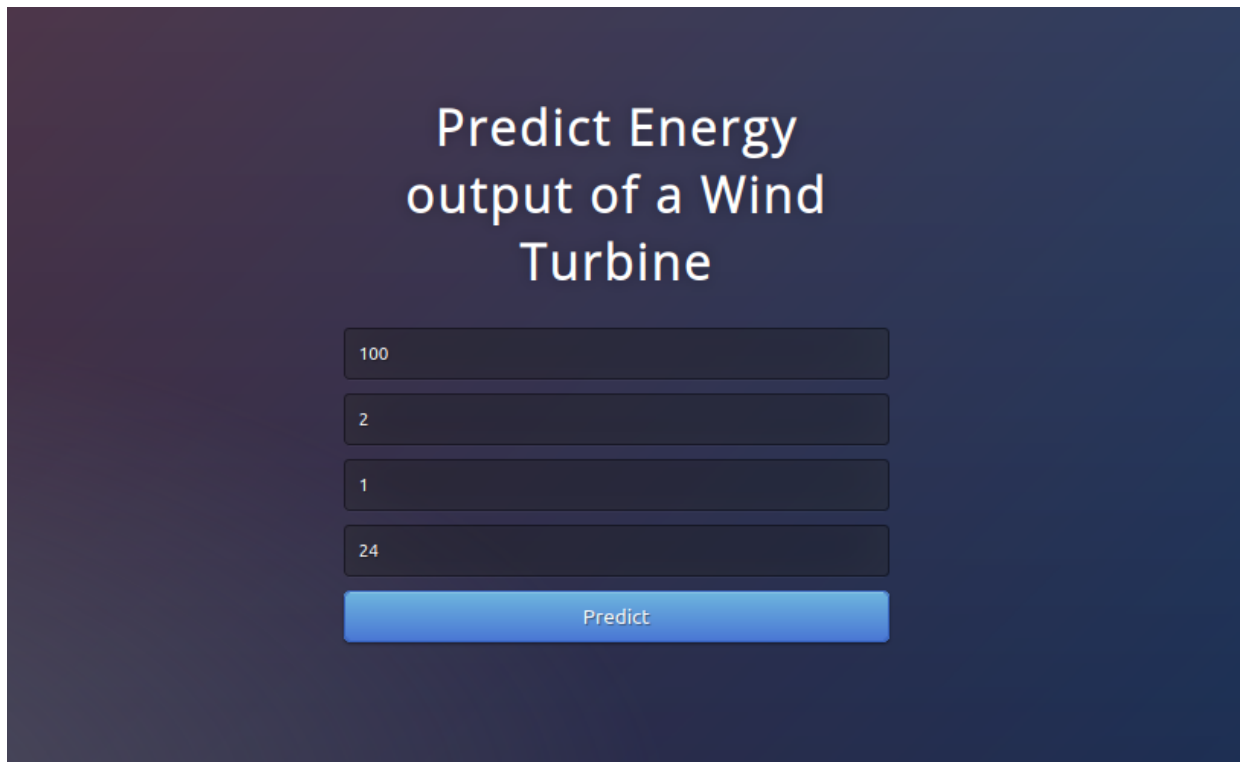
        <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
    </form>

<br>
```

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 :



The Home Page features a dark blue gradient background. At the top center, the title "Predict Energy output of a Wind Turbine" is displayed in a large, white, sans-serif font. Below the title, there are four horizontal input fields, each containing a white numerical value: "100", "2", "1", and "24". These fields are stacked vertically. At the bottom of the input section is a wide, rectangular button with a blue-to-white gradient and the word "Predict" centered in white text.

Predict Energy  
output of a Wind  
Turbine

100

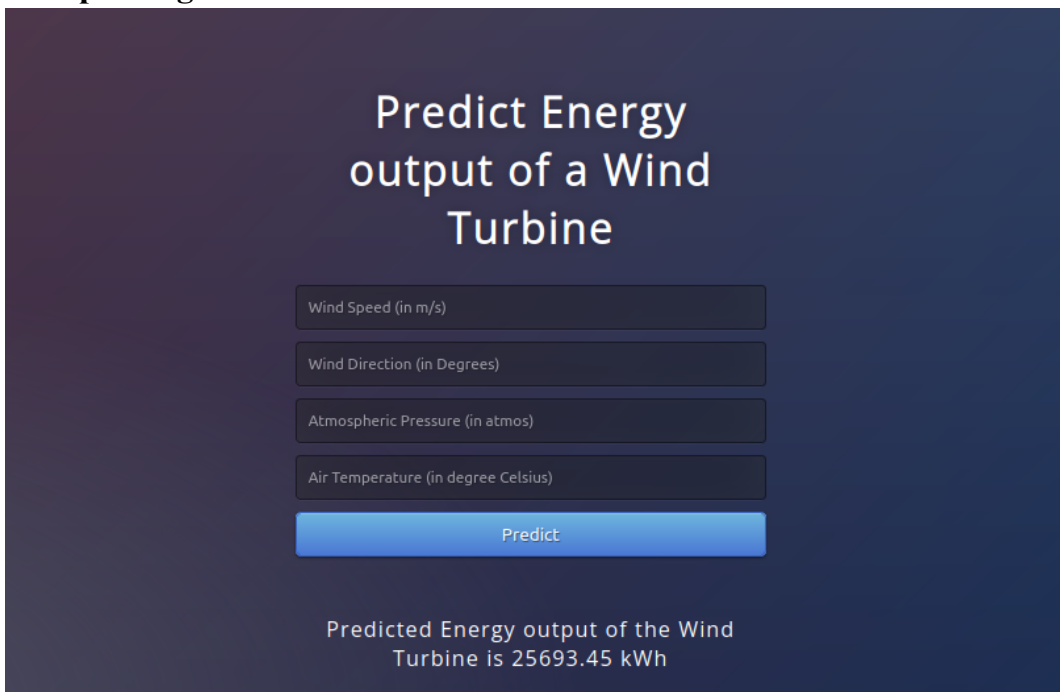
2

1

24

Predict

### Output Page :



The Output Page has the same dark blue gradient background and title as the Home Page. Below the title, there are four horizontal input fields, each with a white label: "Wind Speed (in m/s)", "Wind Direction (in Degrees)", "Atmospheric Pressure (in atmos)", and "Air Temperature (in degree Celsius)". These fields are stacked vertically. Below them is a wide, rectangular button with a blue-to-white gradient and the word "Predict" centered in white text. At the bottom of the page, the predicted energy output is displayed in white text: "Predicted Energy output of the Wind Turbine is 25693.45 kWh".

Predict Energy  
output of a Wind  
Turbine

Wind Speed (in m/s)

Wind Direction (in Degrees)

Atmospheric Pressure (in atmos)

Air Temperature (in degree Celsius)

Predict

Predicted Energy output of the Wind  
Turbine is 25693.45 kWh

## 8 : TESTING

### 8.1 : Test Cases

Wind Speed (m/s)	Wind Direction (°)	Predicted Power Output (KW)
10.0	2.3	1721.18 kWh
23	1.2	5202.57 kWh
13	1	2524.41 kWh
5.99	1	618.01 kWh
34	2	343.34 kWh

### 8.2 : User Acceptance Testing

Numerous people have participated in comprehensive testing of the project. The user interface was quite simple for them to utilise. The web sites were appealing and colourful. The website did not contain any extraneous information. Any novice could learn to use it since it was clear and uncomplicated. The data entry format was very straightforward. The user is not required to enter any units. He only needed to input the figure. The average forecast time is about 3 seconds, which is quite short. The main factor affecting this delay is internet access. The IBM cloud has been used to host the model. So long as an IBM access key is provided, the model may be accessed remotely from any machine using the API that is currently accessible.

## **9 : RESULTS**

### **9.1 : Performance Metrics**

When compared to other models, the model we applied here performs better in terms of speed and accuracy. This has been chosen as the best model for the application after we evaluated the performance metrics of 5 models. All of the test scenarios saw good performance from the model. During the testing process, there weren't any hiccups or delays with the API that was developed.

## **10 : Advantages and Disadvantages**

### **10.1 : Advantages**

This does not necessitate the installation of any additional equipment at the Wind turbine. The existing Sensors can be used to obtain Weather parameters in order to predict power output. With weather stations located all over the world, data can be obtained in real time. The prediction can be performed at the windmill control station. The algorithm is easily adaptable to work with any Wind Turbine. This model takes the previous year's energy outputs and correlates them with the weather and other factors that influenced them. We can use this model to give the weather conditions as input and get the energy output. It also changes the algorithm dynamically based on the predicted and actual output values. This model contributes to the increased use of renewable energy. It improves the performance of wind turbines. Because of the high cost of implementation, this solution is unbeatable. Wind energy companies will be able to increase their energy output, resulting in increased revenue. Wind energy can be relied on as a reliable source because we can predict total power output at any given time.

### **10.2 : Disadvantages**

Windmill companies are hesitant to rely entirely on this model. Data availability is challenging for all individual Wind Mills. Because the Wind Mill is in a remote location, connecting it all proves difficult and costly. The cost of data storage is very high because the data for output power and other attributes will be stored in the cloud. This is costly for the company. For the prediction process, the model requires Weather inputs. Errors in input values such as wind speed, wind direction, temperature, altitude, and humidity can result in prediction errors due to inaccuracy in the instruments used. The model struggles to predict sudden changes in weather conditions. Because climate conditions change around the world every year, data from the previous year is meaningless. Windmill efficiency loss is difficult to calculate and varies from one wind mill to the next. Human-made changes, such as building infrastructure in the wind path, can have a significant impact on the prediction, which cannot be given as input. Because the entire model is hosted in the cloud, a server crash or loss of internet access may leave the company with no other option.

## 11 : CONCLUSION

The project's goal was to forecast wind turbine output energy based on environmental factors. The Linear Regression model was highly accurate. This machine learning model would be especially beneficial to the energy sector. Because we are all shifting to renewable resources, we must exercise extreme caution in how we use them. Based on weather data, this model can predict whether or not installing wind turbines to generate wind energy in a specific location will be profitable. As a result, costs are reduced and the solution is optimized.

## 12 : FUTURE SCOPE

Additional work in this project could include including more features in model training to study the effect on output. A long data history (dataset of more than 3 years) can be used for training to improve accuracy. The application can be upgraded so that the sensor input values are automatically fed into the model rather than the user entering them manually. More web pages can be created so that the user can control multiple Wind Mills using the same API. To move between Windmills, use the navigation tabs. The dashboard can be made interactive for the user by displaying a real-time graph of the prediction and actual power.

## 13 : APPENDIX

### 13.1 : Source Code

#### Model Training :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error, r2_score
import joblib
%matplotlib inline
data =
pd.read_csv('https://raw.githubusercontent.com/Pragadesh-45/Datasets/main/T1.csv')
data.rename(columns = {'LV ActivePower (kW)': 'ActivePower(kW)',
                        "Wind Speed (m/s)": "WindSpeed(m/s)",
                        "Wind Direction (°)": "WindDirection", "Theoretical_Power_Curve
(KWh)": "TheoreticalPowerCurve(KWh)"},
```

```

        inplace = True)
data.head()
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import VotingRegressor
from sklearn.svm import SVR
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import VotingRegressor
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import accuracy_score,r2_score,mean_squared_error
xgr=XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=0.7,
                  colsample_bynode=1, colsample_bytree=0.3, gamma=0.2,
                  importance_type='gain', learning_rate=0.03, max_delta_step=0,
                  max_depth=8, min_child_weight=25, missing=None, n_estimators=800,
                  n_jobs=1, nthread=None, objective='reg:linear', random_state=0,
                  reg_alpha=0.2, reg_lambda=0.8, scale_pos_weight=1, seed=None,
                  silent=None, subsample=0.1, verbosity=1)
sm=SVR(gamma='auto',C=50,epsilon=0.3)
rf=RandomForestRegressor(n_estimators=500,max_depth=4)
lr=LinearRegression()
model=VotingRegressor([('lr',lr), ('rf',rf),('sm', sm),('xgr',xgr)],weights=[1,1,2,3])

```

## IBM Cloud Deployment :

```

!pip install ibm_watson_machine_learning
from ibm_watson_machine_learning import APIClient
wml_credentials = {
    "url": "https://us-south.ml.cloud.ibm.com",
    "apikey": "VRPKD1qVgUgqtLpDzESBUnBSykdvOGmUNou5Z065rfbq"
}
client = APIClient(wml_credentials)

def guid_from_space_name(client,space_name):
    space = client.spaces.get_details()
    #print(space)
    return(next(item for item in space['resources'] if item['entity']['name'] ==
space_name)['metadata']['id'])

space_uid = guid_from_space_name(client,'Model')
print("Space UID = " + space_uid)

deployment = wml_client.deployments.create(
    artifact_uid=model_id,
    meta_props=deployment_props
)

```

## FLASK Application

```

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

app = Flask(__name__)
model = pickle.load(open('Training File/model0.pkl','rb'))

@app.route('/')
def home():
    return render_template('index.html')
@app.route('/predict', methods=['POST'])
def predict():
    int_features = [float(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)

    output = round(prediction[0], 2)
    return render_template('index.html', prediction_text='Predicted Energy
output of the Wind Turbine is {} kWh'.format(output))
@app.route('/predict_api',methods=['POST'])
def predict_api():
    data = request.get_json(force=True)
    prediction = model.predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)
if __name__=="__main__":
    app.run(debug=True)

```

## Home Web Page :

```

<!DOCTYPE html>
<html >
<head>
<meta charset="UTF-8">
<meta charset="UTF-8">
<title>Wind Turbine Energy Predictor</title>
<link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

```



```
<</head>

<body>
  <div class="login">
    <h1>Predict Energy output of a Wind Turbine</h1>
    <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}" method="post">
      <input type="text" name="wind_speed" placeholder="Wind Speed (in m/s)"
required="required" />
      <input type="text" name="wind_direction" placeholder="Wind Direction (in Degrees)"
required="required" />
      <input type="text" name="pressure" placeholder="Atmospheric Pressure (in atmos)"
required="required" />
      <input type="text" name="air_temperature" placeholder="Air Temperature (in degree
Celsius)" required="required" />
      <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
    </form>
  <br>
  <br>
  {{ prediction_text }}
</div>
</body>
</html>
```

## CSS:

```
input {
  width: 100%;
  margin-bottom: 10px;
  background: rgba(0,0,0,0.3);
  border: none;
  outline: none;
  padding: 10px;
  font-size: 13px;
  color: #fff;
  text-shadow: 1px 1px 1px rgba(0,0,0,0.3);
  border: 1px solid rgba(0,0,0,0.3);
  border-radius: 4px;
  box-shadow: inset 0 -5px 45px rgba(100,100,100,0.2), 0 1px 1px rgba(255,255,255,0.2);
  -webkit-transition: box-shadow .5s ease;
  -moz-transition: box-shadow .5s ease;
  -o-transition: box-shadow .5s ease;
  -ms-transition: box-shadow .5s ease;
  transition: box-shadow .5s ease;
}
input:focus { box-shadow: inset 0 -5px 45px rgba(100,100,100,0.4), 0 1px 1px rgba(255,255,255,0.2); }
```

## **13.2 : GitHub & Project Demo Link**

**GitHub Repo :** <https://github.com/IBM-EPBL/IBM-Project-47060-1660796300/>