PREDICTING THE ENERGY OUTPUT OF WIND TURBINE BASED ON WEATHER CONDITIONS

TEAM ID - PNT2022TMID52191

A PROJECT REPORT

Submitted By

FAHEEM ANWAR S. Team Lead (963219104006)

ANANTH KUMAR M. (963219104002)

MOHAMED UKKASHA SUBAKANI M. (963219104017)

RAJAGOPALARAMESH E. (963219104025)

SELVA SUBBIAH A. (963219104032)

In partial fulfillment for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PET ENGINEERING COLLEGE, VALLIOOR

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.

1.2 : 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

2.1 : Existing Problem

The Existing solution has been studied. It has been inferred that the output energy of the wind mill is being predicted approximately manually or using some basic software. The result obtained is often unreliable and the margin of error is very high owing to the multiple variables affecting the output. There are several thousand Wind Mills across the globe. The output performance varies significantly even between two windmills that are located very close to each other. A generalized at the same time customizable to meet the requirement of a single wind mill is required.

2.2 : References

S.NO	Paper Title	Author	Journal Name	Publication year	Description
1	Predicting the Energy Output of Wind Turbine Based on Weather Condition	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	Energy Modelling Output of Wind System based on Wind Speed	Abdelkader Harrouz, Ilhami Colak, Korhan Kayisli	IEEE	2019	The main problem in wind energy is that it becomes quite difficult to predict the power output exactly and this hinders the accuracy. In this paper, there are three different wind models are modeled 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 numerical 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 analyze 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 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 of wind energy efficiently, accurate output power is required. When output power of a windmill at a given time is known we can integrate it with grid and make use of this renewable source of energy rather than conventional non-renewable sources.

3 : IDEATION AND PROPOSED SOLUTION

3.1 : Empathy Map Canvas

What is empathy mapping?

An empathy map is **a collaborative tool teams can use to gain a deeper insight into their customers**. Much like a user persona, an empathy map can represent a group of users, such as a customer segment. The empathy map was originally created by Dave Gray and has gained much popularity within the agile community.
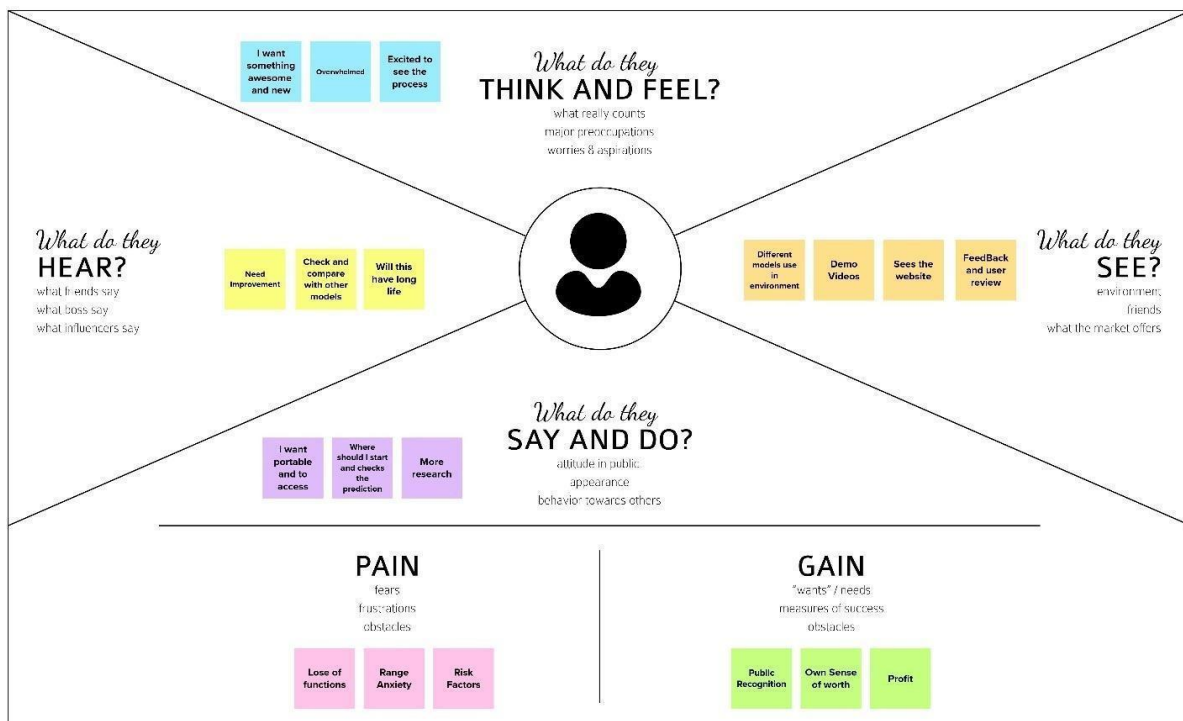
3.2 : Ideation And Brainstorming

Define your problem statement

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

What is brainstorming?

Brainstorming is **a method of generating ideas and sharing knowledge to solve a particular commercial or technical problem, in which participants are encouraged to think without interruption**. Brainstorming is a group activity where each participant shares their ideas as soon as they come to mind.

5 minutes

```
PROBLEM

How might we
[your problem
```

2

Brainstorm

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

⏱ 10 minutes

Faheem Anwar	Ananth Kumar	Selva Subbiah	Rajagopalaramesh
Predicting the wind current in summer / Analysis on wind based	Controlling speed of wind turbine / Using bigger blades	Analysis data on the site / Outcome on app	Predicting wind current in winter / Predicting wind in autumn
Predicting using AI / wind turbines	Using airbone wind turbines / using breaks to stop our wind turbine	Using lubricant in wind turbine / Using larger rotor diameters	wind direction / Placing solar panels to maximise the energy output

Mohamed Ukkasha Subakani

Predicting the wind current in summer	Using airbone wind turbines
Outcome an app	Predicting wind current in autumn

3

Group ideas

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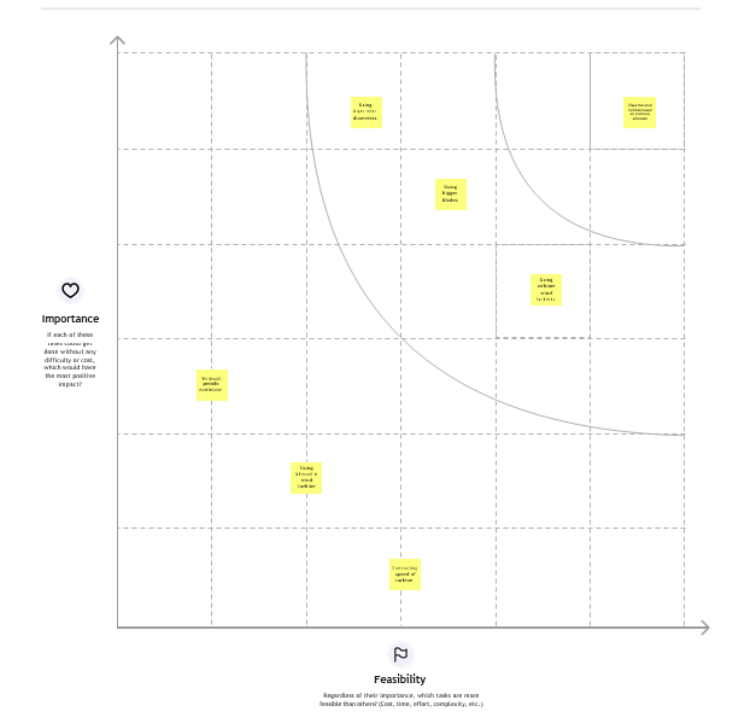
🕐 20 minutes

Predictions

Analysis

Using

4

Prioritize

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

🕐 20 minutes

Importance

If each of these areas could get done without any difficulty or cost, which would have the most positive impact?

Feasibility

Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

3.3 : Proposed Solution

S.No	Parameter	Description
1.	Problem Statement (Problem to be solved)	As wind is stochastic, wind power generation is different from traditional thermal generation. Given the unpredictability associated to the output of the wind farms, wind power forecasting is crucial in addressing the difficulties of balancing supply and demand in any electrical system.
2.	Idea / Solution description	Using a technique that combines physical models with statistical models, a prediction system is created. The auto regressive model in this System predicts the wind farm's inlet state.
3.	Novelty / Uniqueness	The energy outputs from the previous year are taken into account in this model, which also correlates them with the weather and other important factors. To obtain the energy output, we can input the weather conditions into this model. The algorithm is also adjusted dynamically based on the expected value and actual output value.
4.	Social Impact / Customer Satisfaction	A safe and environment-friendly alternative for the generation of electric energy, wind energy receives high levels of popular acceptance. In terms of the social aspect, unlike hydroelectric plants, wind power plants do not have significant negative environmental effects and allow for the coexistence of the production of electricity from the wind alongside the use of land for agriculture and animals.
5.	Business Model (Revenue Model)	Wind utility companies will be able to make more money if they can increase energy output. Wind energy is a trusted source since we can predict the total power output at any given time.
6.	Scalability of the Solution	The data collected from the weather stations can be accessed in real time easily. The weather features can be easily obtained through the sensors installed that helps in predicting energy output.

3.4 : Problem Solution Fit

1. CUSTOMER SEGMENT(S) `CS`

Who is your customer?

> ➢ Individuals
> ➢ Electricity suppliers
> ➢ Industrialist
> ➢ Government

6. CUSTOMER CONSTRAINTS `CC`

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.

1. Illiterates may feel difficulty in accessing the website.
2. Network connection
3. Feeding missing or wrong inputs

5. AVAILABLE SOLUTIONS `AS`

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 & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

Manual calculations based on past climatic conditions which consumes large amount of time were tried in the past.

Pros:
Consumes less time
Cost-effective

Cons:
Network connectivity

2. JOBS-TO-BE-DONE / `J&P`
PROBLEMS

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

Since there's no proper platform for wind energy prediction, we predict the energy output of windturbine in order to earn some revenue and to locate a better place for wind farms.

9. PROBLEM ROOT CAUSE `RC`

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.

Failures occur because of locating wind farms in unsuitable environment.

7. BEHAVIOUR `BE`

What does your customer do to address the problem and get the job done?

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.

3. TRIGGERS `TR`

What triggers customers to act?

Prediction of wind energy helps individuals and electricity suppliers to locate better location for wind farms and let them earn revenue.

4. EMOTIONS: BEFORE / AFTER `EM`

How do customers feel when they face a problem or a job and afterwards?

Before:
1. Stress, frustration
2. Fear of loss of investment

After:
1. Confidence, Happiness
2. Satisfaction, Relaxation

10. YOUR SOLUTION `SL`

A prediction system is developed with a method of combining statistical models and physical models. In this system, the inlet condition of the wind farm is forecasted by the auto regressive model.

Large turbine blades help capturing more of the available wind.

8. CHANNELS of BEHAVIOUR `CH`

8.1 ONLINE
What kind of actions do customers take online?
Checking on data updation.

8.2 OFFLINE
What kind of actions do customers take offline?
Monitoring and maintaining wind farms.

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
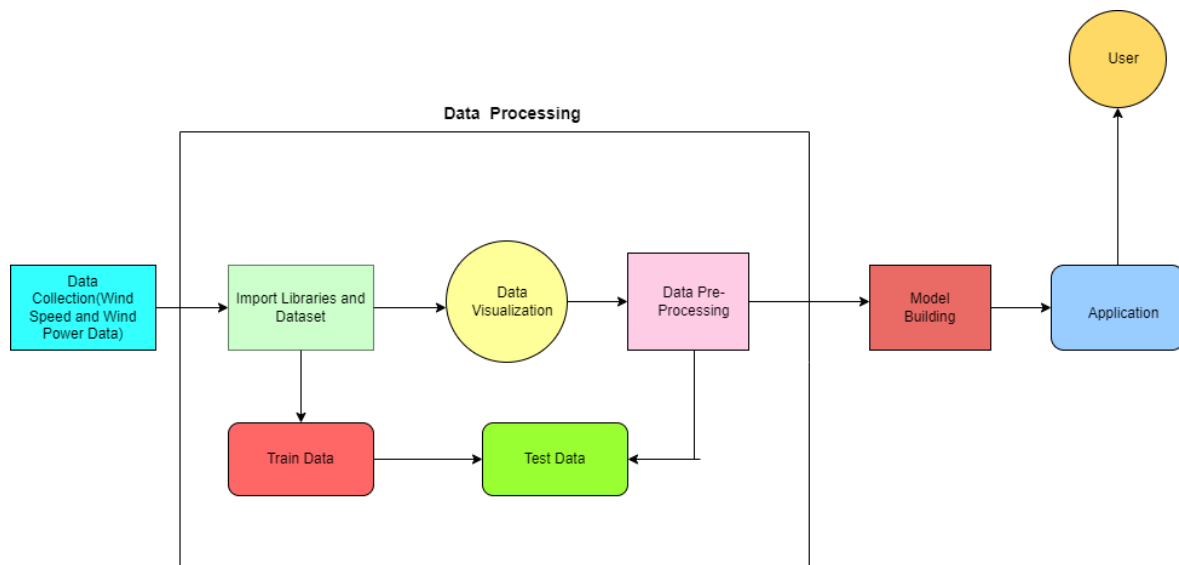
1. The Weather data and wind speed data is fed as input to the ML model.
2. The previous years data stored in a cloud is fed as input to the ML model.
3. The expected power production is predicted.
4. The predicted value is sent to the user, and also stored in cloud.
5. The predicted output power is 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
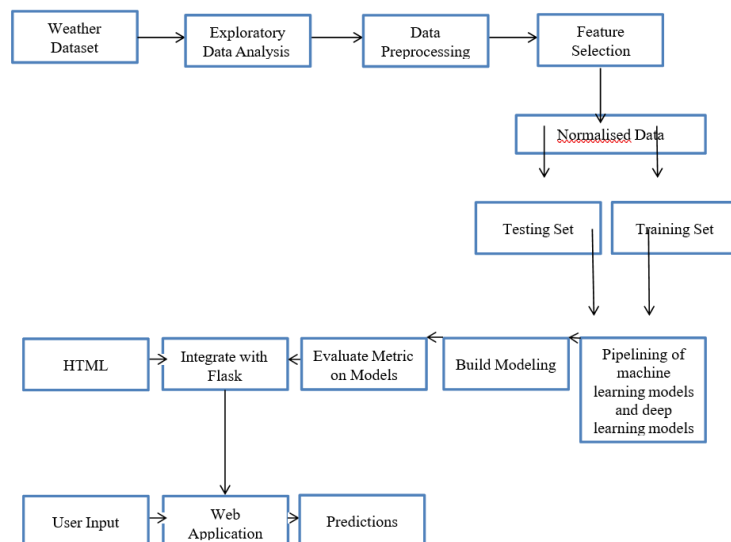
Table-1 : Components & Technologies:

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	1. User Registers into the application through entering Email Id and Password for confirmation. 2. User Receives a confirmation mail for their registered Email. 3. User can also register to the application through Mobile number. 4. User logs in into the website using Email Id and password.
Sprint 2	1. User can access the dashboard. 2. User enters the required details of weather conditions to get the desired turbine power output based on our model's prediction.
Sprint 3	1. Application stores the predictions, that can be used for future analysis. 2. The data stored has to be maintained securely.
Sprint 4	1. Administrator should properly maintain the website and update it whenever required.

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	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 : Reports from JIRA
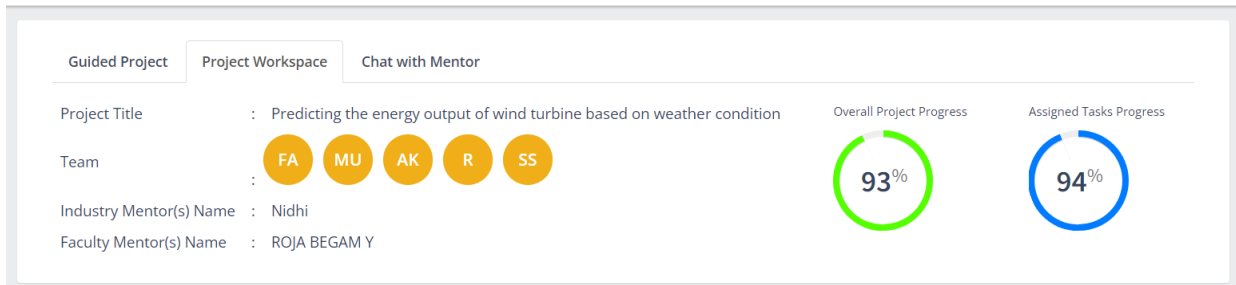
7 : CODING AND SOLUTIONING

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	380.0477905	5.31133604	416.3289078	259.9949036
01 01 2018 00:10	453.7691956	5.672166824	519.9175111	268.6411133
01 01 2018 00:20	306.3765869	5.216036797	390.9000158	272.5647888
01 01 2018 00:30	419.6459045	5.659674168	516.127569	271.2580872
01 01 2018 00:40	380.6506958	5.577940941	491.702972	265.6742859
01 01 2018 00:50	402.3919983	5.604052067	499.436385	264.5786133
01 01 2018 01:00	447.6057129	5.793007851	557.3723633	266.1636047
01 01 2018 01:10	387.2421875	5.306049824	414.8981788	257.9494934
01 01 2018 01:20	463.6512146	5.584629059	493.6776521	253.4806976
01 01 2018 01:30	439.725708	5.523228168	475.7067828	258.7237854
01 01 2018 01:40	498.1817017	5.724115849	535.841397	251.8509979
01 01 2018 01:50	526.8162231	5.934198856	603.0140765	265.5046997
01 01 2018 02:00	710.5872803	6.547413826	824.6625136	274.2329102
01 01 2018 02:10	655.1942749	6.199746132	693.4726411	266.7331848
01 01 2018 02:20	754.7625122	6.505383015	808.0981385	266.7604065
01 01 2018 02:30	790.1732788	6.634116173	859.4590208	270.4931946

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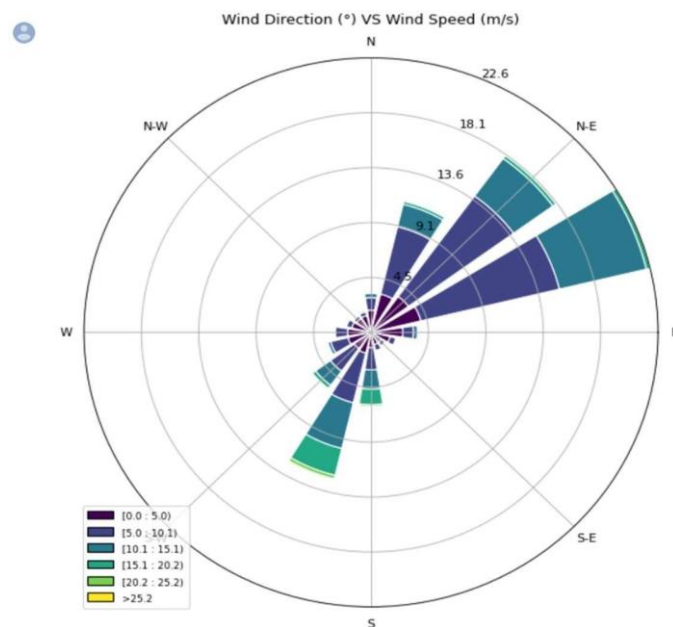
The plot gives a visualization of the direction of the Wind at the location over the year. It also shows the speed of the wind in that particular direction. From this we can infer that this particular WindMill experiences wind in a North-easterly direction primarily and south- westerly direction occasionally.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

Feature Inference :
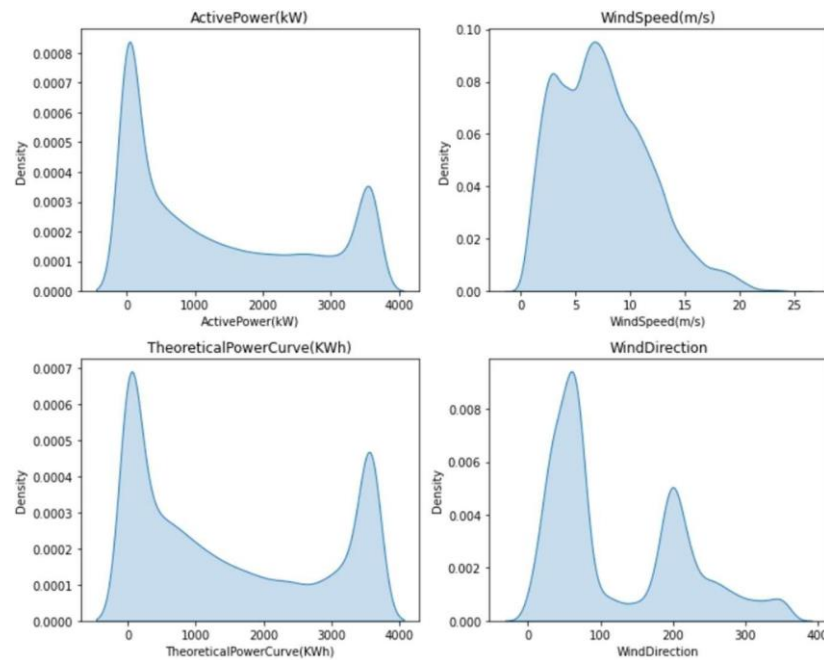
      The above graph plots the weightage of each attribute of the dataset. It helps to understand the dataset quickly and easily. The usual Wind Speed in the region can be seen as 2 to 12 m/s. The predominant Wind direction is 30° to 75° and mildly along 190° to 210° measures from magnetic north. The actual 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 windmill.

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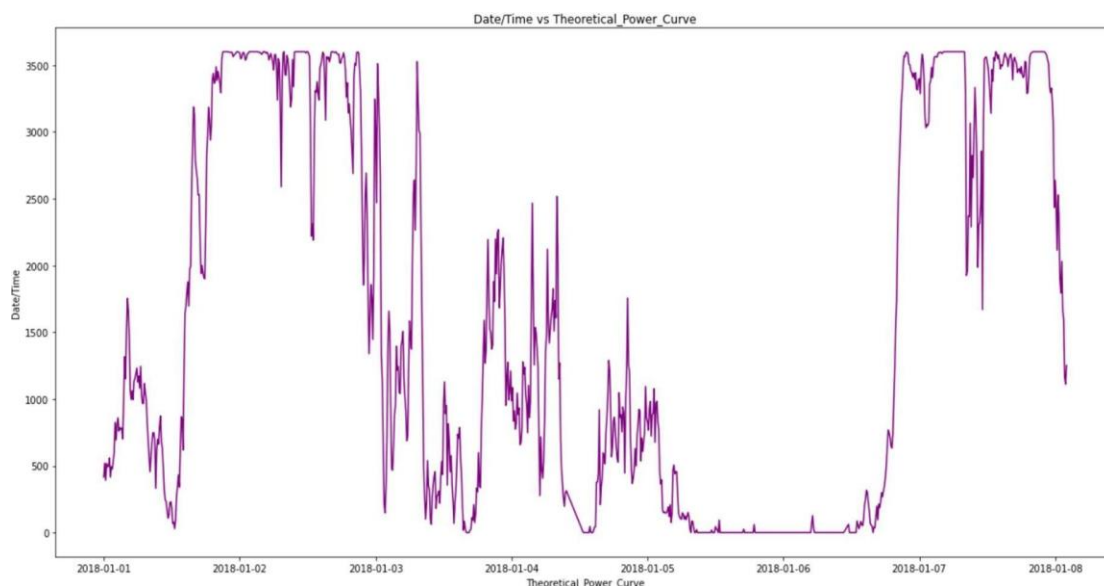
This is a graph plotted with time as x-axis and power generated in y-axis. 1000 samples (8 days) data has been taken for viewing the plot clearly. It shows the trend in power generation. On one day there is maximum output and the next two days the power output is less owing to low wind speed.

Data Pre-processing:

```
In [5]:  data.shape
Out[5]:  (50530, 5)
```

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

```
In [7]:  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 integers.

Splitting Data :

```
In [11]: X=data[['WindSpeed(m/s)','WindDirection']]
         X.head()
```

Out[11]:

	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

```
In [12]: y = data['ActivePower(kW)']
         y.head()
```

```
Out[12]: 0    380.047791
         1    453.769196
         2    306.376587
         3    419.645905
         4    380.650696
         Name: ActivePower(kW), dtype: float64
```

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 :

```
In [14]: 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, Decision Tree Regressor and Support Vector Regression.

Fitting the Models with dataset :

```
In [15]: 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 :

```
In [16]: 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)))

         R2-xgb 0.9197743106205652
         RMSE-xgb 370.6768884049128
         R2-rf 0.9095948832223242
         RMSE-rf 393.4914368167613
         R2-lr 0.8368251429450982
         RMSE-lr 528.6465476346768
         R2-dt 0.8436388583513368
         RMSE-dt 517.4914708383254
         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 models. From the above statement we can see that XGBRegressor model has the highest accuracy of 92%.

XGBRegressor Model Training :

```
In [22]: 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)

Out[22]: 0.9197743106205652
```

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

Saving the Model :

```
In [15]: model_xg.save_model('test_model.bin')
```

```
In [16]: data=[[5.311336,259.994904]]
         df = pd.DataFrame(data, columns=['WindSpeed(m/s)','WindDirection'])
         xgr.predict(df)

Out[16]: array([352.6657], dtype=float32)
```

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

```
In [21]:  wml_credentials = {
               "apikey":"HbsERqktGhGoDzwEicls3kLdXyrcM-Z071KIpvjmAKT3",
               "url":"https://us-south.ml.cloud.ibm.com"
          }

In [22]:  wml_client = APIClient(wml_credentials)

In [23]:  wml_client.spaces.list()

          Note: 'limit' is not provided. Only first 50 records will be displayed if the number of records exceed 50
          ----------------------------------  -----------  ------------------------
          ID                                  NAME         CREATED
          b6ebba24-8cb1-43b0-ae88-0940d085977d  Wind Energy  2022-11-17T15:05:25.014Z
          ----------------------------------  -----------  ------------------------

In [24]:  SPACE_ID= "b6ebba24-8cb1-43b0-ae88-0940d085977d"

In [25]:  wml_client.set.default_space(SPACE_ID)
Out[25]:  '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.

Save and Deploy the model

```
In [28]:  import sklearn
          sklearn.__version__
Out[28]:  '1.0.2'

In [29]:  MODEL_NAME = 'Wind Energy'
          DEPLOYMENT_NAME = 'Wind Energy'
          DEMO_MODEL= model_xg
```

Set Python Version

```
In [30]:  software_spec_uid=wml_client.software_specifications.get_id_by_name('runtime-22.1-py3.9')
```

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 [33]: model_details

Out[33]: {'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': '2022-11-17T16:53:03.438Z',
           'id': '230e3a72-2126-422f-85d7-977cdb7e40fe',
           'modified_at': '2022-11-17T16:53:06.728Z',
           'name': 'Wind Energy',
           'owner': 'IBMid-6640044N0G',
           'resource_key': '43fcad7d-b313-4606-9c8a-1817591ebe24',
           'space_id': 'b6ebba24-8cb1-43b0-ae88-0940d085977d'},
          '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.

Deploy

```
In [36]: deployment = wml_client.deployments.create(
             artifact_uid=model_id,
             meta_props=deployment_props
         )

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

Synchronous deployment creation for uid: '230e3a72-2126-422f-85d7-977cdb7e40fe' 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='198415bf-21da-4777-9224-d0a04107abeb'
-------------------------------------------------------------------------------
```

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 :

To import the required libraries:

```
 1   import flask
 2   import joblib
 3   import numpy as np
 4   import pandas as pd
 5   import requests
 6   from flask import render_template, request
 7   from flask_cors import CORS
 8   from xgboost import XGBRegressor
 9
10   app = flask.Flask(__name__, static_url_path='')
11   CORS(app)
12
```

```
13    API_KEY = "HbsERqktGhGoDzwEicls3kLdXyrcM-Z071KIpvjmAKT3"
14    token_response = requests.post('https://iam.cloud.ibm.com/identity/token',data={"apikey":
15        API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
16    mltoken = token_response.json()["access_token"]
17    header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}
18
```

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

```
18    @app.route('/', methods=['GET'])
19    def sendhomepage():
20        return render_template('Index Page.html')
21
22
23    @app.route('/predict', methods=['POST'])
24    def predictenergy():
25        print("working..")
26        ws = float(request.form['ws'])
27        wd = float(request.form['wd'])
28        X = [[ws, wd]]
29        xgr=XGBRegressor()
30        df=pd.DataFrame(X, columns=['WindSpeed(m/s)', 'WindDirection'])
31        payload_scoring = {"input_data": [{"fields": [['ws','wd']], "values":[3,4]}]}
32
33        response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/198415bf-21da-4777-9224-d0a04107abeb/predictio
34        headers={'Authorization': 'Bearer ' + mltoken})
35        print("response_scoring")
36        predictions = response_scoring.json()
37        print(predictions)
38        predict = predictions['predictions'][0]['values'][0][0]
39        print("Final prediction :",predict)
40        return render_template('Predict Page.html',predict=predict)
41
42
43    if __name__ == '__main__':
44        app.run()
45
```

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 is the connection between the Frontend and backend.

HTML Code :

```
ibm_autoai_flask > templates > <> Index Page.html > ⬡ html
  1    <!DOCTYPE html>
  2    <html lang="en">
  3      <head>
  4        <meta charset="UTF-8" />
  5        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  6        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  7        <title>WIND TURBINE ENERGY PREDICTION</title>
  8        <style>
  9            body{
 10                background-image: url('Windmill.jpg');
 11                background-size: 100%;
 12            }
 13        </style>
 14        <style>
 15            h1{text-align: center;}
 16            h2{text-align: center;}
 17            p{text-align: center;}
 18        </style>
 19      </head>
 20      <body>
 21        <div class="container">
 22          <div class="glass">
 23              <h1 class="text" >WIND TURBINE ENERGY PREDICTION</h1>
 24              <h2 class="text">Using XGBoost Model</h2>
 25              <br />
 26              <form method="POST" action="/predict">
 27                  <p class="text">Wind Speed</p>
 28                  <h1><label><input type="text" id="ws" name="ws"></label></h1>
 29                <p class="text">Wind Direction</p>
 30                <h1><label><input type="text" id="wd" name="wd"></label></h1>
 31                  <br />
 32                  <br />
 33                  <h1><button type="submit"  class="submit">Submit</button> </a></h1>
 34              </form>
 35            </div>
 36          </div>
 37        </body>
 38
```

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.

```
ibm_autoai_flask > templates > <> Predict Page.html > ...
  1   <!DOCTYPE html>
  2   <html lang="en">
  3   <head>
  4       <meta charset="UTF-8">
  5       <meta http-equiv="X-UA-Compatible" content="IE=edge">
  6       <meta name="viewport" content="width=device-width, initial-scale=1.0">
  7       <style>
  8           body{
  9               background-image: url('Windmill.jpg');
 10               background-size: 100%;
 11           }
 12       </style>
 13       <title>WIND TURBINE ENERGY PREDICTION</title>
 14       <style>
 15           h1{text-align: center;}
 16           h3{text-align: center;}
 17       </style>
 18   </head>
 19   <body>
 20
 21       <div class="container">
 22               <div class="glass door">
 23                   <h1 class="text">The predicted Output power is</h1><br>
 24                   <b><h3 class="highlight">{{predict}}</h3></b><br>
 25                   <a href=" http://127.0.0.1:5000" class="submit"><h1>Go Back</h1></a>
 26               </div>
 27       </div>
 28   </body>
 29   </html>
```
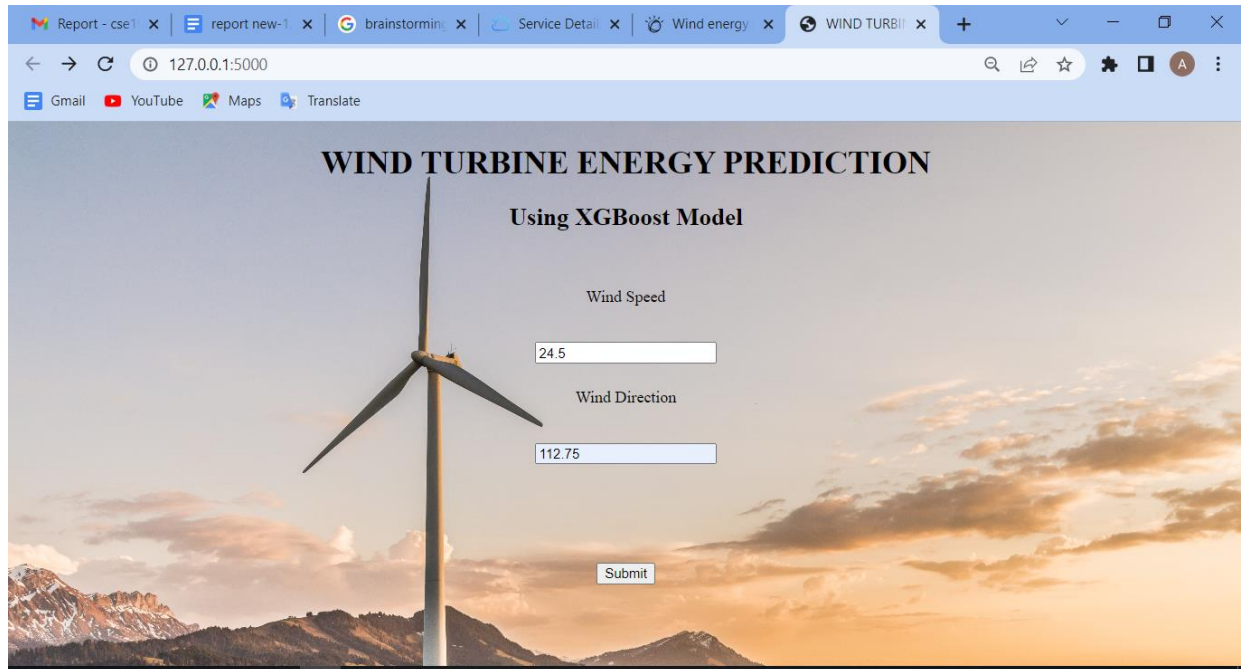
This page displays the output predicted value. This is a post method and hence receives the value from model and displays on the web page.

Web Page Design :

Home Page :

Output Page :

8 : TESTING

8.1 : Test Cases

Wind Speed (m/s)	Wind Direction (°)	Predicted Power Output (KW)
24.5	112.75	2492.15
31.4	143.62	2884.7
15.7	121	3343.995
72.5	456.73	3810.8886
29.8	87.9	2924.15
14.8	295	3758.29
8.3	180	1524.59
0.5	88	6.82
1.2	209	7.764
41.2	395.69	3810.8886

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 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 API developed also performed good with no glitches or lag found during the testing phase.

10 : Advantages and Disadvantages

10.1 : Advantages

This model takes in the previous years energy outputs and relate 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. The cost of Implementing this solution makes it an Unformidable one. 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. 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 WindMill may be in a remote location, providing connectivity to all of it proves challenging and expensive. Data Storage cost is very high, as the 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 windmill 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 in cloud.

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 revenue for the companies. The model is not 100% accurate but 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 work 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 show a real time graph of the prediction and actual power. Diagnosis of windmills which perform the least can be done remotely.

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('T1.csv')
```

```python
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()
data.shape
data.describe()
data.info()
data.isnull().any()
data['Date/Time'] = pd.to_datetime(data['Date/Time'],format='%d %m %Y
%H:%M')
data['year'] = data['Date/Time'].dt.year
data['month'] = data['Date/Time'].dt.month
data['day'] = data['Date/Time'].dt.day
data['Hour'] = data['Date/Time'].dt.hour
data['minute'] = data['Date/Time'].dt.minute
data.head()
data["Date/Time"] = pd.to_datetime(data["Date/Time"], format = "%d %m %Y
%H:%M", errors = "coerce")
data

X=data[['WindSpeed(m/s)','WindDirection']]
X.head()
y = data['ActivePower(kW)']
y.head()
X_train, X_test,y_train, y_test = train_test_split(X,y ,random_state=6,
test_size=0.25)

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()

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)

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)))
```

IBM Cloud Deployment :

```
!pip install -U ibm-watson-machine-learning
from ibm_watson_machine_learning import APIClient
import json
wml_credentials = {
   "apikey":"HbsERqktGhGoDzwEicls3kLdXyrcM-
   Z071KIpvjmAKT3", "url":"https://us-
   south.ml.cloud.ibm.com"
}
wml_client = APIClient(wml_credentials)
wml_client.spaces.list()
SPACE_ID= "b6ebba24-8cb1-43b0-ae88-0940d085977d"
wml_client.set.default_space(SPACE_ID)
wml_client.software_specifications.list(100)

import sklearn
sklearn.__version___
MODEL_NAME = 'Wind Energy'
DEPLOYMENT_NAME = 'Wind Energy'
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
import joblib
import numpy as np
import pandas as pd
import requests
from flask import render_template, request
from flask_cors import CORS
from xgboost import XGBRegressor

app = flask.Flask(__name__, static_url_path='')
CORS(app)

API_KEY = "HbsERqktGhGoDzwEicls3kLdXyrcM-Z071KIpvjmAKT3"
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():
```

```python
        return render_template('Index Page.html')


    @app.route('/predict', methods=['POST'])
    def predictenergy():
        print("working..")
        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": [{"fields": [['ws','wd']], "values":[3,4]}]}

        response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/198415bf-
    21da-4777-9224-d0a04107abeb/predictions?version=2022-11-17', 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 Page.html',predict=predict)


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

Home Web Page :

```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>
  <style>
    body{
      background-image: url('Windmill.jpg');
      background-size: 100%;
    }
  </style>
  <style>
    h1{text-align: center;}
    h2{text-align: center;}
    p{text-align: center;}
  </style>
 </head>
 <body>
  <div class="container">
   <div class="glass">
      <h1 class="text" >WIND TURBINE ENERGY PREDICTION</h1>
      <h2 class="text">Using XGBoost Model</h2>
      <br />
      <form method="POST" action="/predict">
         <p class="text">Wind Speed</p>
         <h1><label><input type="text" id="ws" name="ws"></label></h1>
        <p class="text">Wind Direction</p>
        <h1><label><input type="text" id="wd" name="wd"></label></h1>
         <br />
         <br />
         <h1><button type="submit"  class="submit">Submit</button> </a></h1>
       </form>
     </div>
    </div>
   </body>
<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">
  <style>
    body{
      background-image: url('Windmill.jpg');
      background-size: 100%;
    }
  </style>
  <title>WIND TURBINE ENERGY PREDICTION</title>
  <style>
    h1{text-align: center;}
    h3{text-align: center;}
  </style>
</head>
<body>

    <div class="container">
    <div class="glass door">
    <h1 class="text">The predicted Output power is</h1><br>
    <b><h3 class="highlight">2942.153564453125</h3></b><br>
      <a href=" http://127.0.0.1:5000" class="submit"><h1>Go Back</h1></a>
            </div>
    </div>
</body>
</html>
```

CSS:

```
@import
url('https://fonts.googleapis.
com/css2?family=Mulish:it
al,wght@0,400;0,500;0,600
;1 ,400;1,500;1,600
display=swap');


html,
body {

overflow-x: hidden;
padding: 0;
margin: 0;
}
body {
height: 100vh;
width: 100vw;
}
body {
scrollbar-gutter: 10px;
}
.container {
height: 100%;
width: 100%;
background-image: url("4.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{
```

```css
      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;
   }
```

13.2 : GitHub & Project Demo Link

GitHub Repo : https://github.com/IBM-EPBL/IBM-Project-45249-1660729044

Project Demo Link:
https://drive.google.com/file/d/1r6f2RkCICUUelVoyxP_5WuAUpICV5tU7/view?usp=share_link

Youtube Link : https://youtu.be/ltKtuS_7yxY