Predicting The Energy Output Of Wind Turbine Based On Weather Condition

Date: 17.06.2020

1. INTRODUCTION

1.1. OVERVIEW

Wind energy plays an increasing role in the supply of energy world-wide. The energy output of a wind farm is dependent on the wind conditions present at its site. If the output can be predicted more accurately, energy suppliers can coordinate the collaborative production of different energy sources more efficiently to avoid costly overproduction.

This project will suggest the best time to utilize the energy from wind farm.

1.2. PURPOSE

This project's objective is to develop a time series model to predict the power output of wind farm based on the weather condition in the site.

2. LITERATURE SURVEY

2.1. EXISTING PROBLEM

The power output of a single wind turbine is a direct function of the strength of the wind over the rotor swept area. The strength of the wind depends mostly on the wind speed.

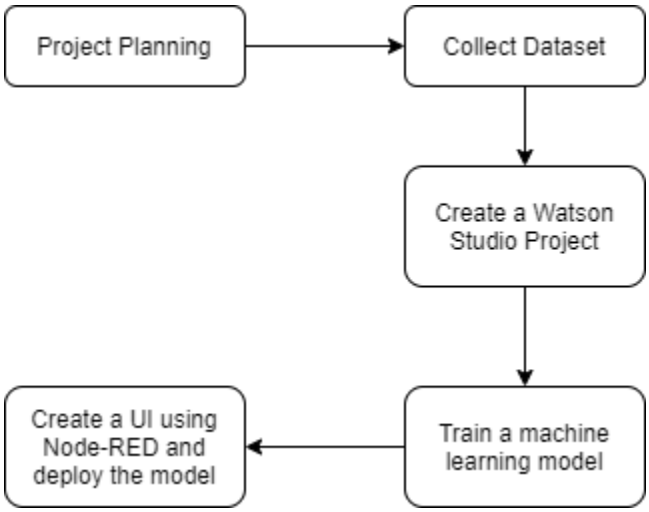
Another important aspect to consider is the fact that wind speed and direction fluctuate over time. Therefore, we must consider the time, wind speed and wind direction to optimize energy production.

2.2. PROPOSED SOLUTION

The proposed solution is to use linear regression to train a machine learning model. The input for the model will be time, wind speed and wind direction. A user friendly interface created using node-red would make it easy for the user to interact with the deployed model.

3. THEORETICAL ANALYSIS

3.1. BLOCK DIAGRAM



3.2. SOFTWARE DESIGNING

The software designing involves the following steps:

1) PROJECT PLANNING

This step involves deciding on which programming software to use and how the user shall access and interact with the model. The deadline must also be kept in mind during this step.

2) COLLECT DATA

This step involves collection of the dataset from kaggle.com.

3) CREATE IBM SERVICES

To deploy the application in IBM cloud, necessary services need to be created. This involves machine learning services for Watson Studio and cloud services.

4) CREATE A WATSON STUDIO PROJECT

Once the necessary services are created, a Watson Studio project can be created.

5) IMPORT THE DATASET

Watson studio needs the dataset as an asset to use it. Therefore, the dataset must be imported. It will be stored in the cloud object storage.

6) TRAIN THE MODEL

A notebook is created and python libraries like numpy, scikit learn are used to visualize the data and train the model.

7) BUILD NODE-RED FLOW

Once the model has been trained, it can be deployed. Node-red is used to provide a user friendly for for users to interact with the model and provide input and get the output.

4. EXPERIMENTAL INVESTIGATIONS

The Dataset:

The dataset is a csv file which has the following columns:

- Date / time
- Lv activepower (kw)
- Wind speed (m/s)
- Theoretical_power_curve (kwh)
- Wind direction (°)

Watson Studio:

Ibm's Watson studio is has many features which can be used for machine learning.

Preprocessing and cleaning dataset:

Before the dataset is trained, it needs to be processed and clean. For example, null values should be removed or set to 0. Also libraries to perform the above must be imported first. We can also convert the data to

a format in which it can be handled easily.

The necessary libraries are imported first.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Once the dataset is imported to the notebook, it is checked for null values.

```
df.isnull().sum()
```

```
Date/Time                       0
LV ActivePower (kW)             0
Wind Speed (m/s)                0
Theoretical_Power_Curve (KWh)   0
Wind Direction (°)              0
dtype: int64
```

The 'Date / Time' column is split into day, month, year and hour.

```
#Convert the Format of date and time
df['Date/Time'] = pd.to_datetime(df['Date/Time'],format='%d %m %Y %H:%M')
df.head()
df.info()
```

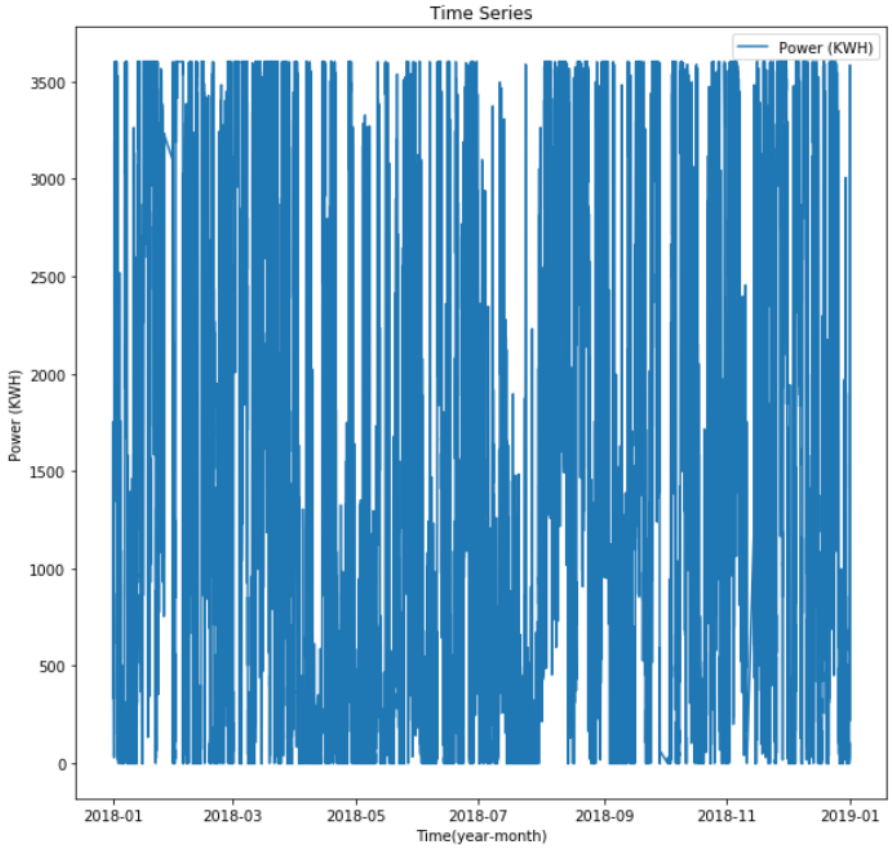
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50530 entries, 0 to 50529
Data columns (total 5 columns):
Date/Time                       50530 non-null datetime64[ns]
LV ActivePower (kW)             50530 non-null float64
Wind Speed (m/s)                50530 non-null float64
Theoretical_Power_Curve (KWh)   50530 non-null float64
Wind Direction (°)              50530 non-null float64
dtypes: datetime64[ns](1), float64(4)
memory usage: 1.9 MB
```

```
#Make another Coloumns for Month , Day , Hour , Year for accuurate Prediction.
df['Month']=df['Date/Time'].dt.month
df['Day']=df['Date/Time'].dt.day
df['Hour']=df['Date/Time'].dt.hour
df['Year']=df['Date/Time'].dt.year
```

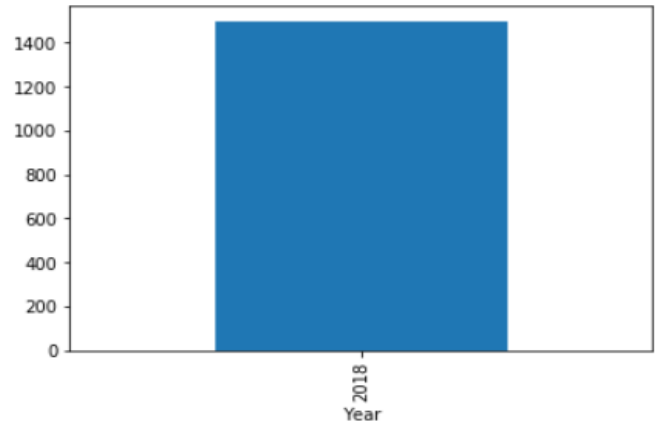
Data visualization:

Once the data has processed, graphs can be used to understand the data better. Plotting of the graphs is done with the help of mathplotlib.

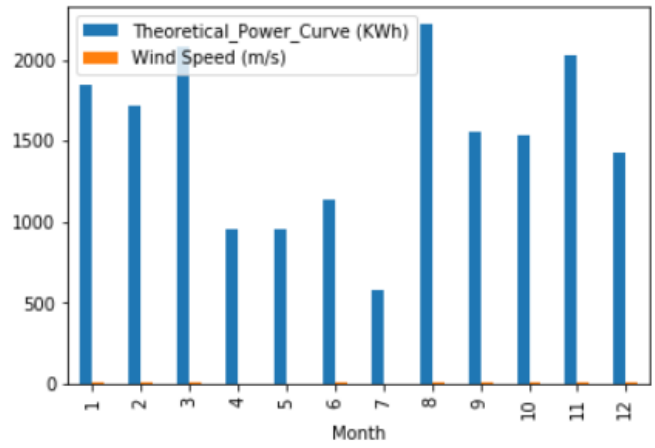
```
#Plotting Of Power Vs Time
df.index = df['Date/Time']                          # indexing the Datetime to get the time period on the x-axis.
ts = df['Theoretical_Power_Curve (KWh)']
plt.figure(figsize=(10,10))
plt.plot(ts, label='Power (KWH)')
plt.title('Time Series')
plt.xlabel("Time(year-month)")
plt.ylabel("Power (KWH)")
plt.legend(loc='best')
plt.show()
```



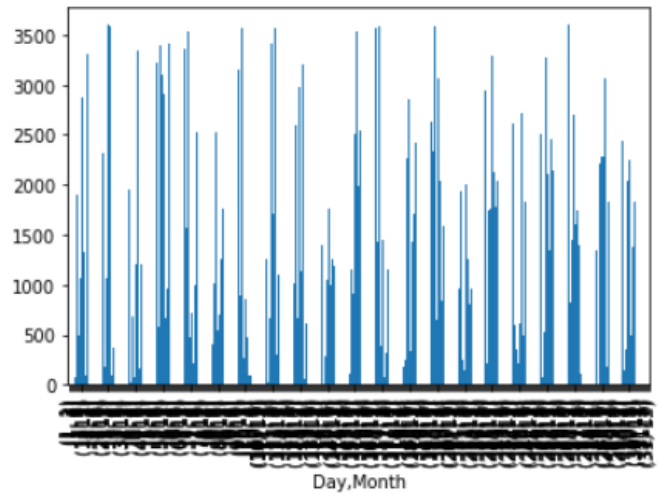

```
#Plotting Of Year/Month/Day
yr = df.groupby('Year')['Theoretical_Power_Curve (KWh)'].mean().plot.bar()
plt.show()
```



```
#Plotting Month , Theoretical Power Curve and Wind Speed
m = df.groupby('Month')['Theoretical_Power_Curve (KWh)','Wind Speed (m/s)'].mean().plot.bar()
plt.show()
```




```
#Plotting Day , month and Theoretical power
df.groupby(['Day','Month'])['Theoretical_Power_Curve (KWh)'].mean().plot.bar()
plt.show()
```



Training The Model:

Once visualization is done, the next step is to train the model.

Scikit-learn is used to train the model. Regression has been used here.

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state = 42)
```

```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline
model = Pipeline([('poly', PolynomialFeatures(degree=6)),('linear', LinearRegression(fit_intercept=False))])
```

```
model = model.fit(X_train,y_train)
```

Get Prediction:

Once the training is done, the predicted output can be obtained from the model.

```
pred_y = model.predict(X_test)
```

```
pred_y
```

```
array([3467.73307415, 2982.68008019,  776.11451813, ...,   20.47389759,
       1075.09855427,   17.81662961])
```

By comparing the predicted output with the values in the dataset, the accuracy of the model can be evaluated.

There are many was to measure the accuracy. Here, the mean square value and mean absolute error have been found in addition to the accuracy of the model.

```
from sklearn.metrics import mean_squared_error

print('Mean squared error: %.2f'
      % mean_squared_error(y_test, pred_y))

y_test.shape
```

```
Mean squared error: 494.71
(10106,)
```

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test, pred_y)
```

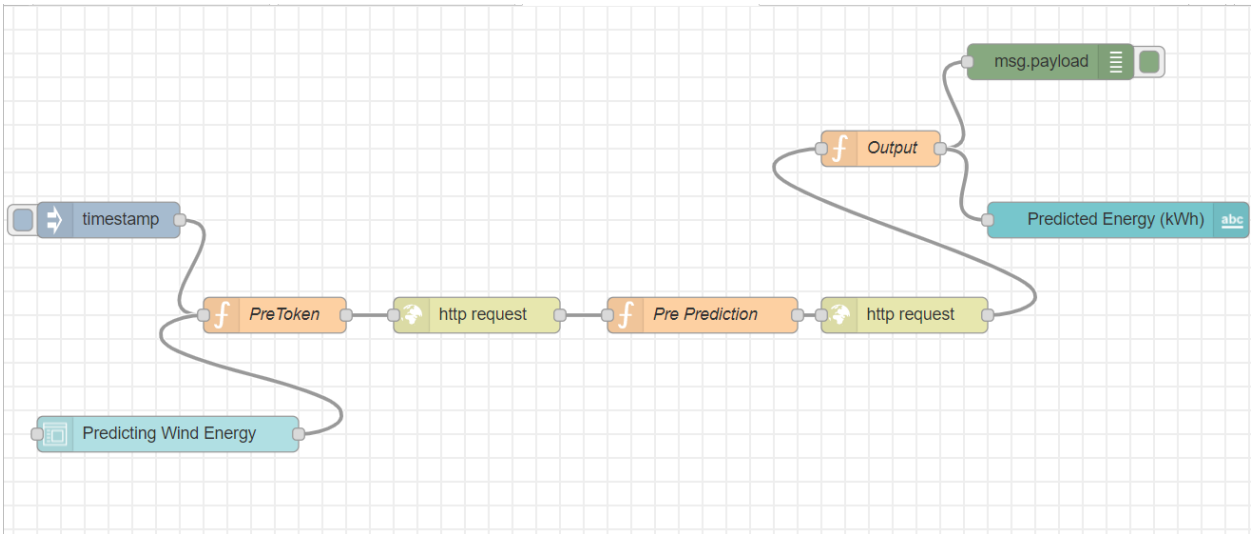
```
12.538318174969866
```

```
model.score(X_test,y_test)
```

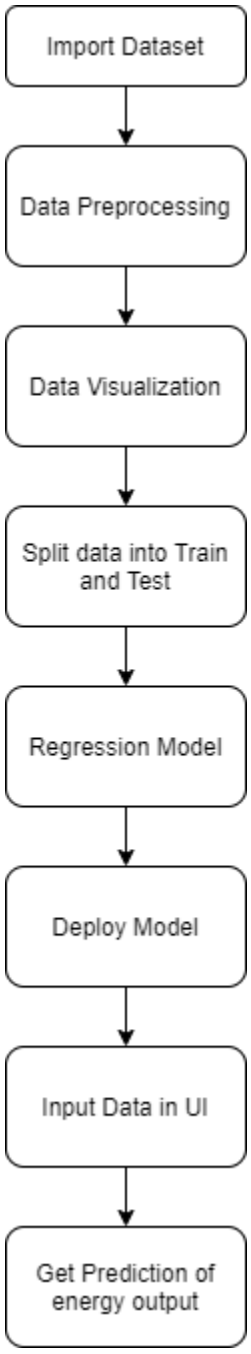
```
0.9997334768569658
```

Node-RED flow:

Once the model has been trained and deployed, a ui can be created for users to interact with the model. Node-RED has been used for this purpose. The Node-RED flow is shown below:



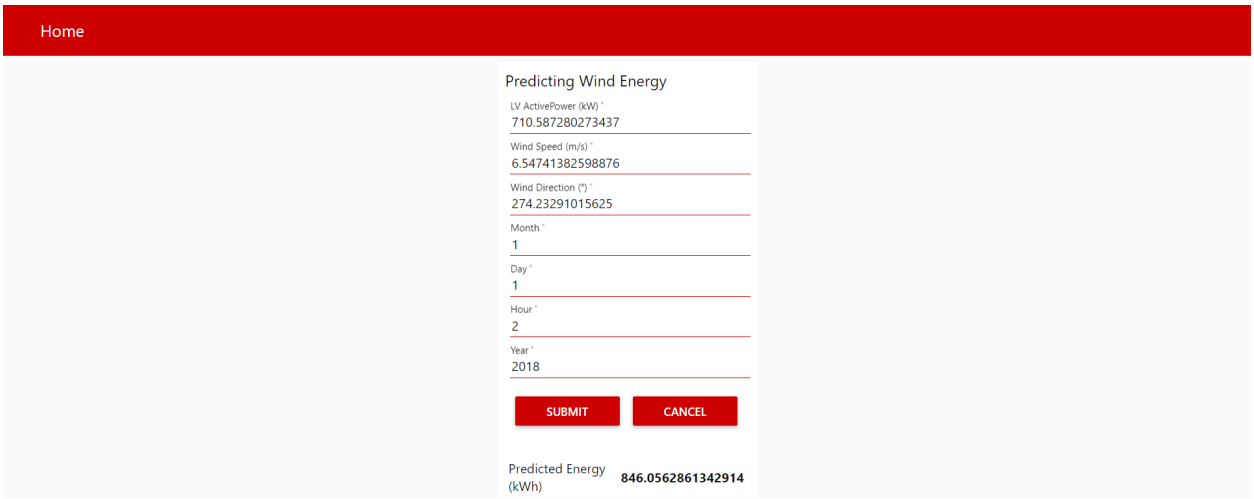
5. FLOWCHART


Import Dataset
Data Preprocessing
Data Visualization
Split data into Train and Test
Regression Model
Deploy Model
Input Data in UI
Get Prediction of energy output

6. RESULT

Web based UI was developed by integrating all the services using Node-RED.

The energy output of a wind turbine was predicted, given the weather conditions. From the result, it is possible to determine when the energy output was best.



7. ADVANTAGES AND DISADVANTAGES

Advantages:

One of the biggest advantages of embedding machine learning algorithms is their ability to improve over time. Machine learning technology typically improves efficiency and accuracy thanks to the ever-increasing amounts of data that are processed.

Using Node-RED also simplifies the effort put into a creating the front end.

Disadvantages:

Machine learning can be very time consuming especially when there is a large amount of data. Also, the machine learning model may not be as accurate as manual calculations.

Node-RED has very limited options and offers little customization for the UI.

8. APPLICATIONS

Through this project, wind farms can get a good overview on how the weather affects energy production and optimize their energy production.
Also, energy suppliers can coordinate the collaborative production of different energy sources more efficiently to avoid costly overproduction.

9. CONCLUSION

The end product is a webpage created and deployed on Node-RED app of IBM cloud. The backend of webpage is a regression model created and deployed on Watson Studio using machine learning service. This model can be used to predict the energy output of wind turbine based on weather conditions.

10. FUTURE SCOPE

The scalability and flexibility of the application can also be improved with advancement in technology and availability of new and improved resources. With more data, the predictions will also become more accurate.
Factors like season and type of turbine can also be considered in the future.

11. BIBLIOGRAPHY

1. Node-red Starter Application:
https://developer.ibm.com/tutorials/how-to-create-a-node-red-starter-application/
2. Watson Studio Cloud:
https://bookdown.org/caoying4work/watsonstudio-workshop/jn.html/
3. Dataset Reference:
https://www.kaggle.com/berkerisen/wind-turbine-scada-dataset
4. IBM Cloud Services:
https://www.youtube.com/watch?v=dbrglahdj48&list=plzpeuwuenmk2pytascakk4bzjayzhw23l
5. Information On Wind Energy:
https://hpi.de/friedrich/docs/paper/re1.pdf

APPENDIX

1. **Source Code:**

1.1. Node-RED flow:

[{"id":"e63bc915.781688","type":"tab","label":"Flow 1","disabled":false,"info":""},{"id":"10c7ec66.a19a04","type":"function","z":"e63bc915.781688","name":"PreToken","func":"global.set(\"lv\",msg.payload.lv)\nglobal.set(\"wSpd\",msg.payload.wSpd)\nglobal.set(\"wDir\",msg.payload.wDir)\nglobal.set(\"mon\",msg.payload.mon)\nglobal.set(\"day\",msg.payload.day)\nglobal.set(\"hr\",msg.payload.hr)\nglobal.set(\"yr\",msg.payload.yr)\nvar apikey=\"kDP_k4wL37ReoyyQR0gQmsH3gqoAwQel0hRhXuZV-Fjc\";\nmsg.headers={\"content-type\":\"application/x-www-form-urlencoded\"}\nmsg.payload={\"grant_type\":\"urn:ibm:params:oauth:grant-type:apikey\",\"apikey\":apikey}\nreturn msg;","outputs":1,"noerr":0,"x":240,"y":260,"wires":[["d8f3d09e.765d6"]]},{"id":"d8f3d09e.765d6","type":"http request","z":"e63bc915.781688","name":"","method":"POST","ret":"obj","paytoqs":false,"url":"https://iam.cloud.ibm.com/identity/token","tls":"","persist":false,"proxy":"","authType":"","x":410,"y":260,"wires":[["2ea9246f.e3db4c"]]},{"id":"64db4f5a.7953c","type":"inject","z":"e63bc915.781688","name":"","topic":"","payload":"","payloadType":"date","repeat":"","crontab":"","once":false,"onceDelay":0.1,"x":100,"y":180,"wires":[["10c7ec66.a19a04"]]},{"id":"7dfb2584.29cd6c","type":"debug","z":"e63bc915.781688","nam

```
e":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"payloa
d","targetType":"msg","x":893.0000114440918,"y":47.00000190734863,"wires":[]},{"i
d":"2ea9246f.e3db4c","type":"function","z":"e63bc915.781688","name":"Pre
Prediction","func":"var lv = global.get('lv')\nvar wSpd = global.get('wSpd')\nvar
wDir = global.get('wDir')\nvar mon = global.get('mon')\nvar day =
global.get('day')\nvar hr = global.get('hr')\nvar yr = global.get('yr')\nvar
token=msg.payload.access_token\nvar
instance_id=\"48bbedc7-82d9-42e5-b264-bb3233048f5d\"\nmsg.headers={'Cont
ent-Type': 'application/json',\"Authorization\":\"Bearer
\"+token,\"ML-Instance-ID\":instance_id}\nmsg.payload={\"fields\": [\"LV
ActivePower (kW)\",\"Wind Speed (m/s)\",\"Wind Direction (°)\", \"Month\",
\"Day\", \"Hour\", \"Year\"], \"values\": [[lv,wSpd,wDir,mon,day,hr,yr]]}\nreturn
msg;","outputs":1,"noerr":0,"x":600,"y":260,"wires":[["ca61f932.9d40f8"]]},{"id":"ca61f
932.9d40f8","type":"http
request","z":"e63bc915.781688","name":"","method":"POST","ret":"obj","paytoqs":fals
e,"url":"https://eu-gb.ml.cloud.ibm.com/v3/wml_instances/48bbedc7-82d9-42e5-
b264-bb3233048f5d/deployments/9ed3747a-b247-4f34-a2a1-c90965d36314/on
line","tls":"","persist":false,"proxy":"","authType":"","x":770,"y":260,"wires":[["a76f559f.3
c4728"]]},{"id":"264d0185.e5100e","type":"ui_form","z":"e63bc915.781688","name":""
,"label":"Predicting Wind
Energy","group":"a50d1238.d4866","order":1,"width":0,"height":0,"options":[{"label":"L
V ActivePower
(kW)","value":"lv","type":"number","required":true,"rows":null},{"label":"Wind Speed
(m/s)","value":"wSpd","type":"number","required":true,"rows":null},{"label":"Wind
Direction
(°)","value":"wDir","type":"number","required":true,"rows":null},{"label":"Month","value"
:"mon","type":"number","required":true,"rows":null},{"label":"Day","value":"day","type":"
number","required":true,"rows":null},{"label":"Hour","value":"hr","type":"number","requ
ired":true,"rows":null},{"label":"Year","value":"yr","type":"number","required":true,"rows
":null}],"formValue":{"lv":"","wSpd":"","wDir":"","mon":"","day":"","hr":"","yr":""},"payload":"","
submit":"Submit","cancel":"Cancel","topic":"","x":150,"y":360,"wires":[["10c7ec66.a19
a04"]]},{"id":"4e53bb45.b01814","type":"ui_text","z":"e63bc915.781688","group":"a50
d1238.d4866","order":2,"width":0,"height":0,"name":"","label":"Predicted Energy
(kWh)","format":"{{msg.payload}}","layout":"row-spread","x":950,"y":180,"wires":[]},{"i
d":"a76f559f.3c4728","type":"function","z":"e63bc915.781688","name":"Output","fun
c":"msg.payload=msg.payload.values[0][0]\nreturn
msg;","outputs":1,"noerr":0,"x":750,"y":120,"wires":[["7dfb2584.29cd6c","4e53bb45.b
```

01814"]]},{"id":"a50d1238.d4866","type":"ui_group","z":"","name":"Default","tab":"8713df88.c3364","order":1,"disp":false,"width":"6","collapse":false},{"id":"8713df88.c3364","type":"ui_tab","z":"","name":"Home","icon":"dashboard","disabled":false,"hidden":false}]

1.2. Notebook

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
import types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
client_c90b74d62daa49ba8b2ce9916a9d5144 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='YwfYuy02TXQ7JIGVrKb3qJfytBJB9LhOeSwhXRvrIsny',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.eu-geo.objectstorage.service.networklayer.com')

body = client_c90b74d62daa49ba8b2ce9916a9d5144.get_object(Bucket='windturbine-donotdelete-pr-bxj0ldnrvptfi6',Key='T1.csv')['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

df = pd.read_csv(body)
df.head()
```

```
df.isnull().sum()
```

```
#Convert the Format of date and time
df['Date/Time'] = pd.to_datetime(df['Date/Time'],format='%d %m %Y %H:%M')
df.head()
df.info()
```

```
#Make another Coloumns for Month , Day , Hour , Year for accuurate Prediction.
df['Month']=df['Date/Time'].dt.month
df['Day']=df['Date/Time'].dt.day
df['Hour']=df['Date/Time'].dt.hour
df['Year']=df['Date/Time'].dt.year
```

```
#Plotting Of Power Vs Time
df.index = df['Date/Time']                        # indexing the Datetime to get the time period on the x-axis.
ts = df['Theoretical_Power_Curve (KWh)']
plt.figure(figsize=(10,10))
plt.plot(ts, label='Power (KWH)')
plt.title('Time Series')
plt.xlabel("Time(year-month)")
plt.ylabel("Power (KWH)")
plt.legend(loc='best')
plt.show()
```

```
#Plotting Of Year/Month/Day
yr = df.groupby('Year')['Theoretical_Power_Curve (KWh)'].mean().plot.bar()
plt.show()
```

```
#Plotting Month , Theoretical Power Curve and Wind Speed
m = df.groupby('Month')['Theoretical_Power_Curve (KWh)','Wind Speed (m/s)'].mean().plot.bar()
plt.show()
```

```
#Plotting Day , month and Theoretical power
df.groupby(['Day','Month'])['Theoretical_Power_Curve (KWh)'].mean().plot.bar()
plt.show()
```

```
#Drop columns
x = df.drop(['Theoretical_Power_Curve (KWh)','Date/Time'],axis = 1)
x
```

```
y = df['Theoretical_Power_Curve (KWh)']
y
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state = 42)
```

```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline
model = Pipeline([('poly', PolynomialFeatures(degree=6)),('linear', LinearRegression(fit_intercept=False))])
```

```
model = model.fit(X_train,y_train)
```

```
pred_y = model.predict(X_test)
```

```
pred_y
```

```
from sklearn.metrics import mean_squared_error

print('Mean squared error: %.2f'
      % mean_squared_error(y_test, pred_y))

y_test.shape
```

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test, pred_y)
```

```
model.score(X_test,y_test)
```

```
plt.scatter(X_test.iloc[:,1],y_test, color='black')
plt.xticks(())
plt.yticks(())

plt.show()
```

```
plt.scatter(X_test.iloc[:,1],pred_y,color='blue')
plt.xticks(())
plt.yticks(())

plt.show()
```

```
plt.scatter(y_test,pred_y)
plt.show()
```

```
b = np.arange(100)
plt.figure(figsize=(20,10))
plt.plot(b, y_test[:100], color='blue', label='Actual Value')
plt.plot(b, pred_y[:100], color='red', label='Predicted Value')
plt.title('Visualization pred_y v/s y_test')
plt.xlabel('Reference line')
plt.ylabel('LV ActivePower (kW)')
plt.show()
```

```
from watson_machine_learning_client import WatsonMachineLearningAPIClient
```

```
wml_credentials = {
  "apikey": "kDP_k4wL37ReoyyQR0gQmsH3gqoAwQel0hRhXuZV-Fjc",
  "iam_apikey_description": "Auto-generated for key c88fab5b-7f9b-4c1a-b39a-12cad46a68fb",
  "iam_apikey_name": "WindEnergy",
  "iam_role_crn": "crn:v1:bluemix:public:iam::::serviceRole:Writer",
  "iam_serviceid_crn": "crn:v1:bluemix:public:iam-identity::a/a248d226cbac45618e7a934fe71c2e08::serviceid:ServiceId-9edae2f7-8805-4b1b-b89a-199f8d61e770",
  "instance_id": "48bbedc7-82d9-42e5-b264-bb3233048f5d",
  "url": "https://eu-gb.ml.cloud.ibm.com"
}
```

```
#Create a client
client = WatsonMachineLearningAPIClient(wml_credentials)
```

```
metadata = {
    client.repository.ModelMetaNames.AUTHOR_NAME : "Krishna",
    client.repository.ModelMetaNames.NAME : "WindEnergyPrediction"
}
```

```
#Store the model
stored_data = client.repository.store_model(model,meta_props = metadata)
```

```
guid = client.repository.get_model_uid(stored_data)
```

```
deploy = client.deployments.create(guid)
```

```
scoring_endpoint = client.deployments.get_scoring_url(deploy)
```

```
scoring_endpoint
```