

WIND POWER GENERATION

Project ID : SPS_PRO_1530

Project Title : Wind Power Generation

1. PROJECT DESCRIPTION:

Problem Statement:

Wind energy plays an increasing role in the supply of energy world-wide. The energy output of a wind farm is highly 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.

Solution:

To resolve this problem we'll develop a web application that shows the estimated wind energy that could be generated from the parameters like wind speed, wind direction and time of the day. The time series in the web app will monitor the estimated power generation for a time wind speed, wind direction. The training of the model will be done with Wind Turbine Scada Dataset 2018. The attributes in the dataset are [Date/Time, LV ActivePower (kW), Wind Speed (m/s), Theoretical_Power_Curve (KWh), Wind Direction (°)]. User can use the time series of the web app to fetch the prime hours of production and hence can coordinate with the collaborators.

2. PROJECT SCOPE:

The project is developed keeping into mind how increasing energy demand can be fulfilled with a balance between renewable and non-renewable sources. The uncertainty associated with potential of renewable resources, makes it difficult to move towards them. The web application tries to bridge this gap between uncertainty in potential and increasing demand of energy. We present a detailed time series of expected production at the wind mills, which creates a scope for the energy suppliers to avoid the over consumption of non-renewable resources.

3. PROJECT SCHEDULE:

In this task, we discussed how to proceed for the project understanding all the required flow of the project .Assigned tasks to everyone.

4. TECHNOLOGIES USED:

1. IBM Cloud : IBM Cloud is a suite of **cloud** computing services from **IBM** that offers both platform as a service (PaaS) and infrastructure as a service (IaaS).

2. IBM Watson Studio: It is an integrated environment designed to make it easy to develop, train, manage models, and deploy AI-powered applications and is a SaaS solution delivered on the IBM Cloud.

3 . Node-RED : It is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things.

4 . IBM Machine Learning : **IBM Watson Machine Learning** is a full-service **IBM** Cloud offering that makes it easy for developers and data scientists to work together to integrate predictive capabilities with their applications.

5. Python : It is the heart of the entire project . **Python** is a high-level **programming** language designed to be easy to read and simple to implement. It is open source, which means it is free to use, even for commercial applications.

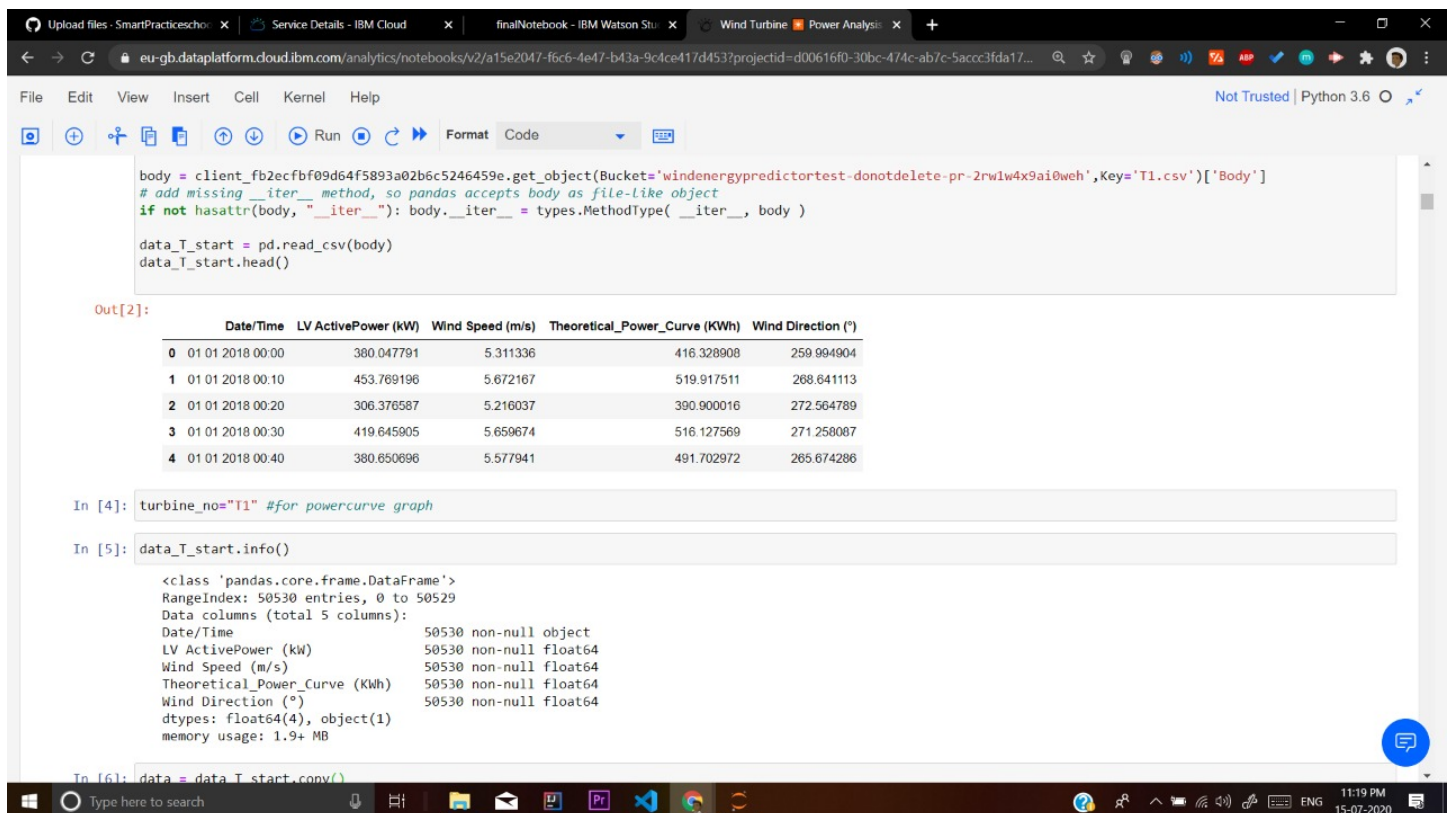
6. Weather Api : The Open weather api provide us the real time weather statistics in an JSON format.

5. OUR FLOW:

1. Dataset :

To create the web app, we first went into the search of a data-set. Our search stopped at 2018 Scada Data of a Wind Turbine in Turkey. As all machine learning algorithms, we also started with pre-processing of available data to overcome redundant features and overfitting.

RAW DATA :



The screenshot shows an IBM Watson Studio notebook interface. The code cell contains the following Python code:

```
body = client_fb2ecfbf09d64f5893a02b6c5246459e.get_object(Bucket='windenergypredictortest-donotdelete-pr-2rw1w4x9ai0weh', 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 )

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

The output of the code is displayed as follows:

Out[2]:

	Date/Time	LV ActivePower (kW)	Wind Speed (m/s)	Theoretical_Power_Curve (KWh)	Wind Direction (°)
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

Below the table, the notebook shows the execution of `data_T_start.info()`, which returns the following information:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50530 entries, 0 to 50529
Data columns (total 5 columns):
Date/Time                50530 non-null object
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: float64(4), object(1)
memory usage: 1.9+ MB
```

The bottom of the notebook shows the start of the next code cell: `In [6]: data = data_T_start.conv()`.

CLEANED DATA :

IBM Watson Studio

My projects / WindEnergyPredictorTest / Wind Turbine * Power Analysis

File Edit View Insert Cell Kernel Help

data_T_clean.head()

```
Out[86]:
```

	Time	ActivePower(kW)	WindSpeed(m/s)	Theoretical_Power_Curve (KWh)	Wind_Direction	Month	mean_WindSpeed	mean_Direction	Direction	Loss_Value(kW)	Loss(%)
0	01 01 2018 00:00	380.05	5.31	416.33	259.99	Jan	5.5	270	W	36.28	8.71
1	01 01 2018 00:10	453.77	5.67	519.92	268.64	Jan	5.5	270	W	66.15	12.72
2	01 01 2018 00:20	306.38	5.22	390.90	272.56	Jan	5.0	270	W	84.52	21.62
3	01 01 2018 00:30	419.65	5.66	516.13	271.26	Jan	5.5	270	W	96.48	18.69
4	01 01 2018 00:40	380.65	5.58	491.70	265.67	Jan	5.5	270	W	111.05	22.59

```
In [87]: #creating summary speed dataframe from clean data.
DepGroupT_speed = data_T_clean.groupby("mean_WindSpeed")
data_T_speed=DepGroupT_speed.mean()
#removing the unnecessary columns.
data_T_speed.drop(columns=["WindSpeed(m/s)", "Wind_Direction", "mean_Direction"], inplace=True)
#creating a windspeed column from index values.
listTspeed_Ws=data_T_speed.index.copy()
data_T_speed["WindSpeed(m/s)"]=listTspeed_Ws
#changing the place of columns.
data_T_speed=data_T_speed[["WindSpeed(m/s)", "ActivePower(kW)", "Theoretical_Power_Curve (KWh)", "Loss_Value(kW)", "Loss(%)"]]
#changing the index numbers.
data_T_speed["Index"]=list(range(1,len(data_T_speed.index)+1))
data_T_speed.set_index("Index", inplace=True)
#rounding the values to 2 digit
data_T_speed=data_T_speed.round({"WindSpeed(m/s)": 1, 'ActivePower(kW)': 2, 'Theoretical_Power_Curve (KWh)': 2, 'Loss_Value(kW)': 2, 'Loss(%)': 2})
#creating a count column that shows the number of wind speed from clean data.
data_T_speed["count"]=len(data_T_clean["mean_WindSpeed"][data_T_clean["mean_WindSpeed"]==i])
for i in data_T_speed["WindSpeed(m/s)"]

In [88]: data_T_speed
```

11:14 PM 15-07-2020

2. Creation Of Watson Studio Project :

We then created the watson studio project in IBM Cloud .

The screenshot displays the IBM Watson Studio web interface. At the top, a navigation bar includes the IBM Watson Studio logo, an 'Upgrade' button, a notification bell, and the user's account 'Jashmin Mishra's Account'. Below the navigation bar, a main header area contains two primary actions: 'Create a project' (with subtext 'Create a project, and then add the tools and assets you need.') and 'Search a catalog' (with subtext 'Find the assets you need in a catalog.').

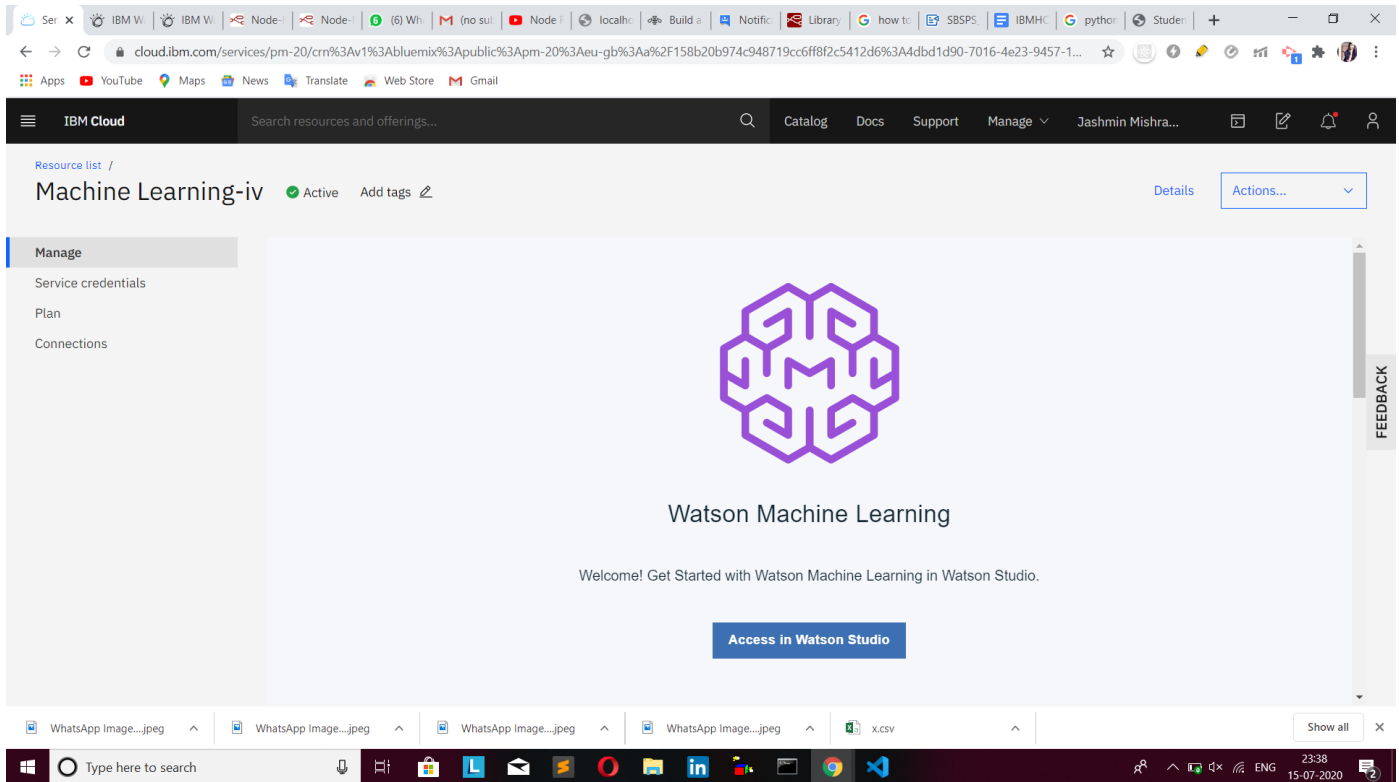
Below these actions, the 'Recently updated projects' section is visible, featuring a table with the following data:

Name	Role	Collaborators	Date created	Last updated
WindEnergyPredictorTest	Admin	JM AK GS	Jun 19, 2020	Jul 14, 2020

Below the table, the 'Your catalogs' section is shown, with a 'New Catalog' button. At the bottom of the interface, a 'Get started' button is visible. The browser's address bar shows the URL 'eu-gb.dataplatform.cloud.ibm.com/home?context=wdp&apps=data_science_experience&nocache=true'. The Windows taskbar at the bottom shows the search bar and several open applications, including WhatsApp, a file explorer, and a terminal.

3. Creation Of Machine Learning Service :

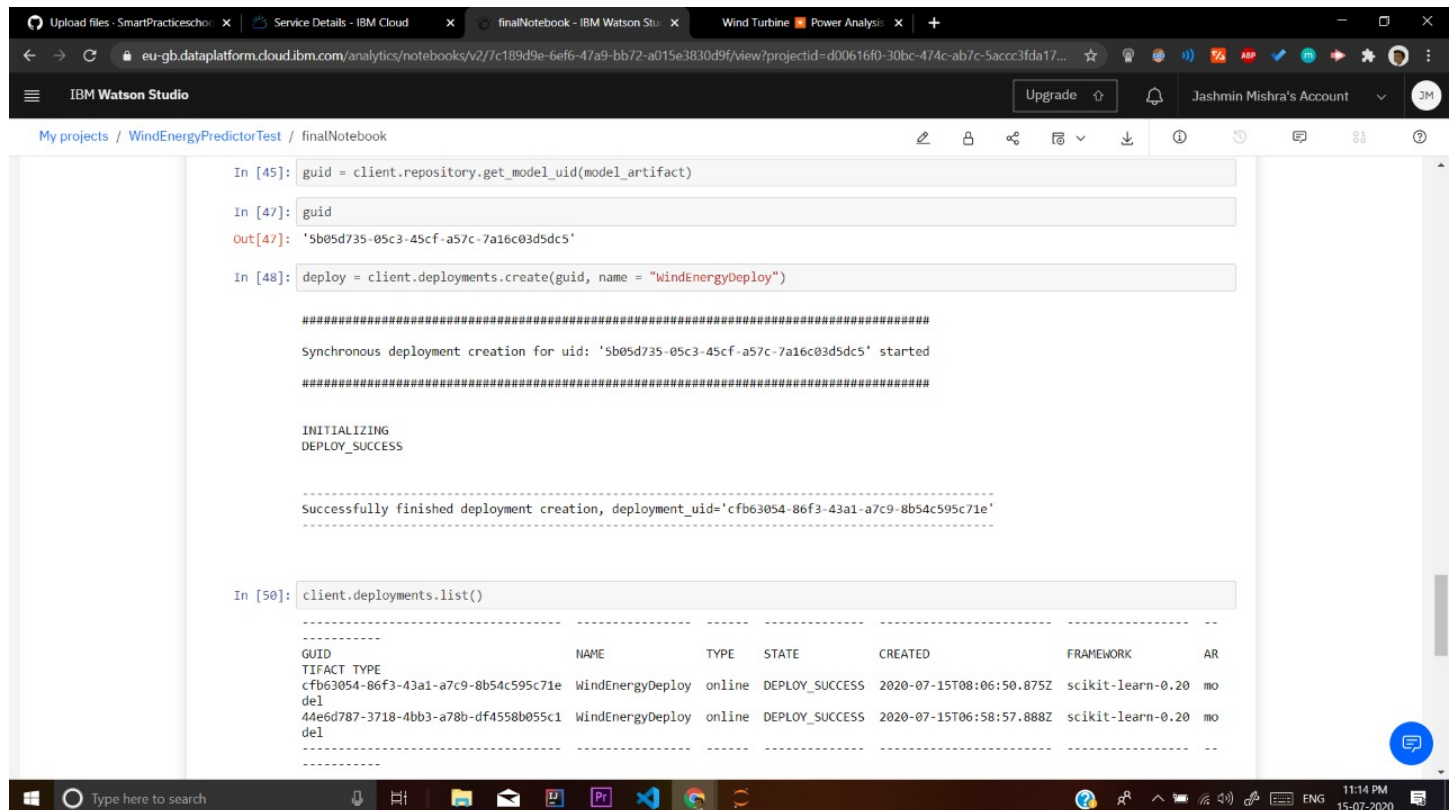
Then we created the machine learning service and integrated it .



4. Model Creation :

For our prediction we moved ahead with XGB Regressor model.

In Watson Assistant Studio we went on with a notebook containing all the required codes. We then split our dataset into train-test and trained our XGBRegressor model. In the notebook, using Watson Machine Learning client module we created an API and a scoring url so that prediction can be made in real time using our UI.



The screenshot shows a Jupyter Notebook interface within IBM Watson Studio. The browser address bar indicates the URL: eu-gb.dataplatform.cloud.ibm.com/analytics/notebooks/v2/7c189d9e-6ef6-47a9-bb72-a015e3830d9f/view?projectId=d00616f0-30bc-474c-ab7c-5acc3fda17... The notebook title is 'finalNotebook - IBM Watson Studio'. The code is being executed in a cell, and the output shows the successful creation and listing of a deployment.

```
In [45]: guid = client.repository.get_model_uid(model_artifact)

In [47]: guid
Out[47]: '5b05d735-05c3-45cf-a57c-7a16c03d5dc5'

In [48]: deploy = client.deployments.create(guid, name = "WindEnergyDeploy")

#####
Synchronous deployment creation for uid: '5b05d735-05c3-45cf-a57c-7a16c03d5dc5' started
#####

INITIALIZING
DEPLOY_SUCCESS

Successfully finished deployment creation, deployment_uid='cfb63054-86f3-43a1-a7c9-8b54c595c71e'

In [50]: client.deployments.list()
```

GUID	NAME	TYPE	STATE	CREATED	FRAMEWORK	AR
cfb63054-86f3-43a1-a7c9-8b54c595c71e	WindEnergyDeploy	online	DEPLOY_SUCCESS	2020-07-15T08:06:50.875Z	scikit-learn-0.20	mo
44e6d787-3718-4bb3-a78b-df4558b055c1	WindEnergyDeploy	online	DEPLOY_SUCCESS	2020-07-15T06:58:57.888Z	scikit-learn-0.20	mo


```
Upload files · SmartPracticescho... x Service Details - IBM Cloud x finalNotebook - IBM Watson Stu... x Wind Turbine Power Analysis: x +
eu-gb.dataplatform.cloud.ibm.com/analytics/notebooks/v2/7c189d9e-6ef6-47a9-bb72-a015e3830d9f/view?projectId=d00616f0-30bc-474c-ab7c-5acc3fda17...
IBM Watson Studio Upgrade Jashmin Mishra's Account 3M

My projects / WindEnergyPredictorTest / finalNotebook

In [6]: x_train=data[['WindSpeed','WindDirection']].values
        y_train=data['ActivePower'].values

In [ ]:

In [7]: from sklearn.metrics import mean_squared_error,r2_score
        import xgboost as xgb
        xgb.__version__

Out[7]: '0.82'

In [34]: model_xgb1 = xgb.XGBRegressor(max_depth = 4, learning_rate=2e-2,min_child_weight=1.1, reg_alpha=0.3,reg_lambda=0.7, nthread = -
        1)

In [35]: model_xgb1.fit(x_train,y_train)

Out[35]: XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
        colsample_bytree=1, gamma=0, importance_type='gain',
        learning_rate=0.02, max_delta_step=0, max_depth=4,
        min_child_weight=1.1, missing=None, n_estimators=100, n_jobs=1,
        nthread=-1, objective='reg:linear', random_state=0, reg_alpha=0.3,
        reg_lambda=0.7, scale_pos_weight=1, seed=None, silent=True,
        subsample=1)

In [36]: preds=model_xgb1.predict(x_train)
        score=mean_squared_error(y_train,preds)
        score**0.5

Out[36]: 450.03955092861816

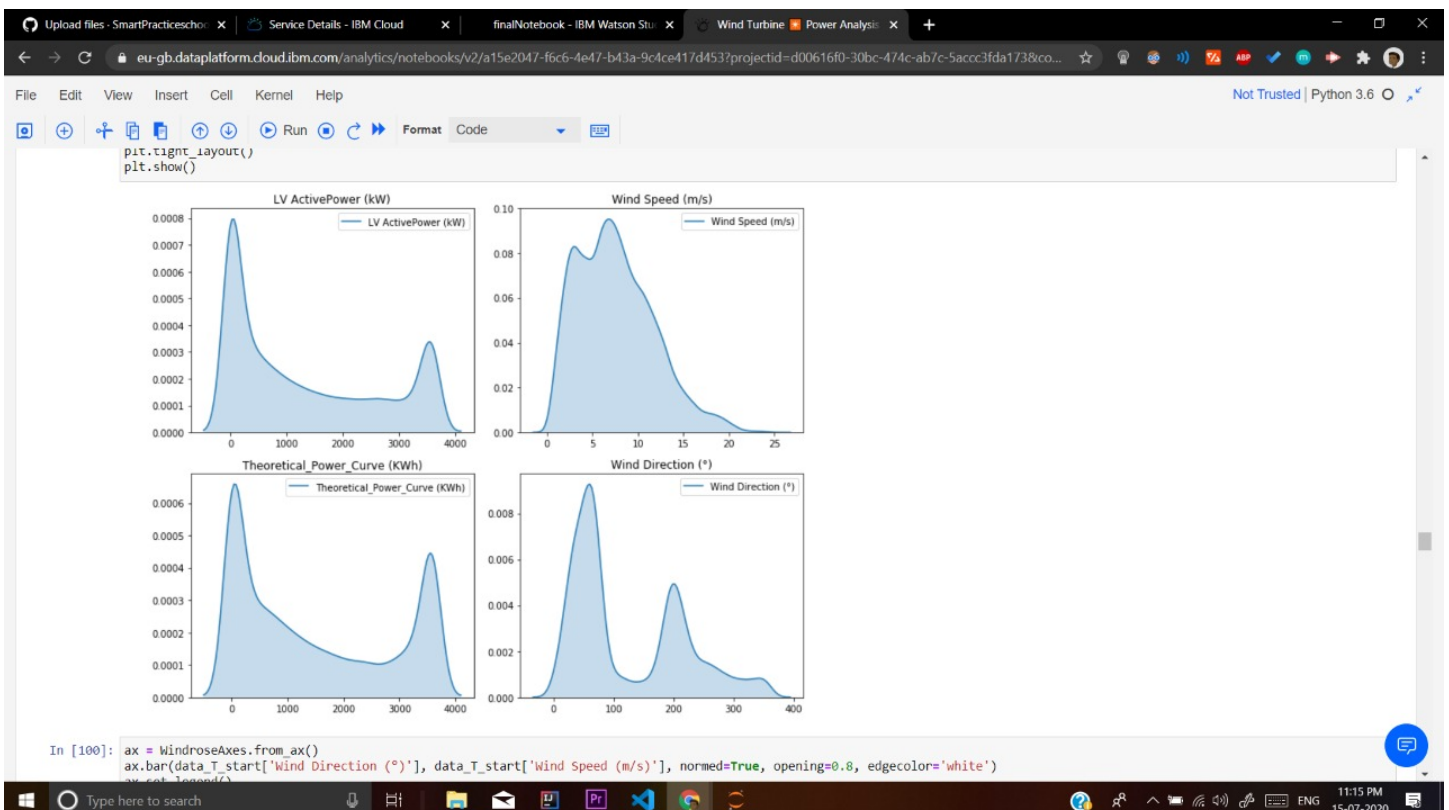
In [37]: r2_score(y_train,preds)

Out[37]: 0.8824186859885449

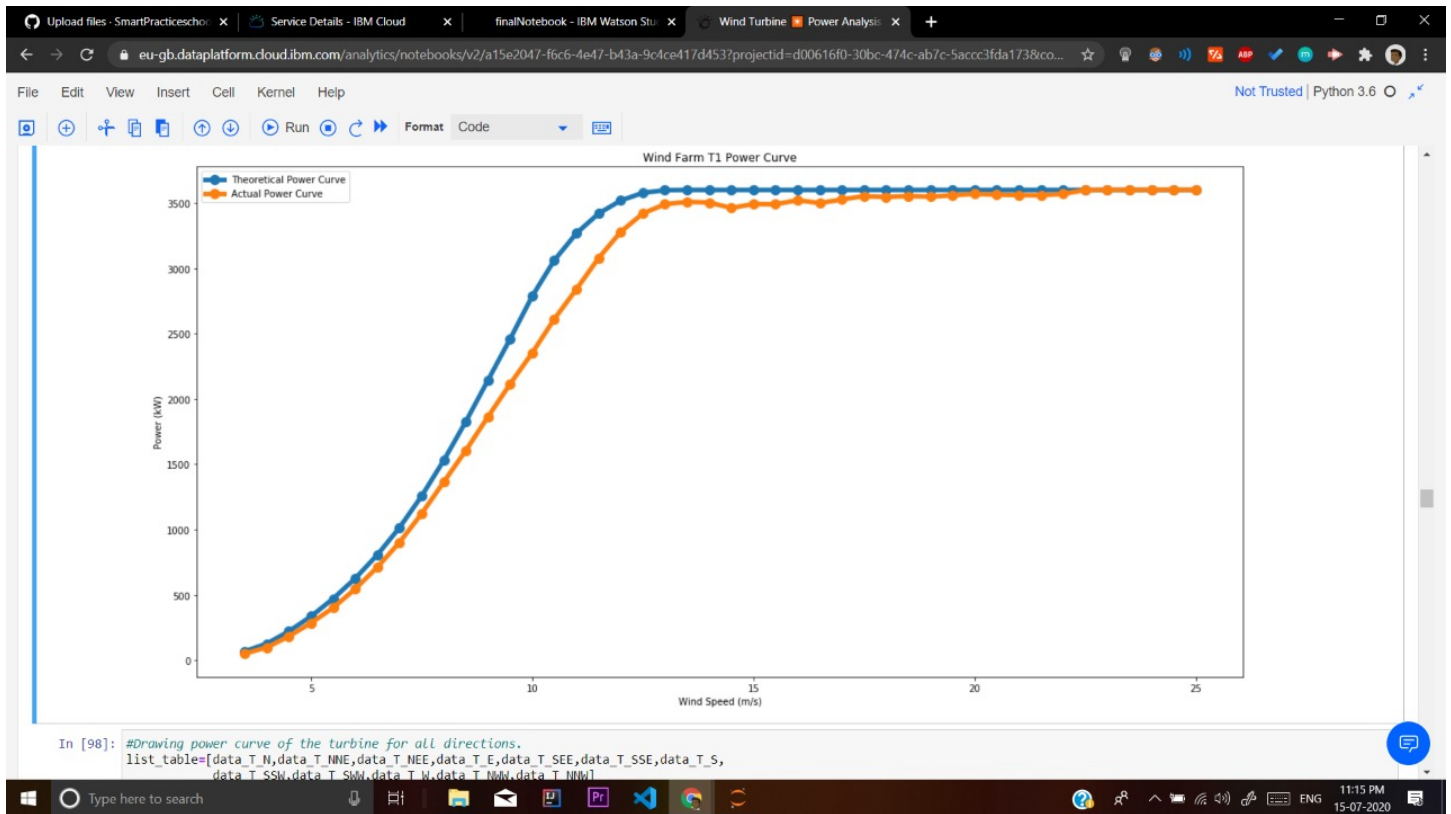
In [55]: model_xgb1.predict(np.array([4.1,250]).reshape(1,-1))

Out[55]: array([83.987441...dtype=float32])
```

GRAPH SHOWING WIND SPEED VS WIND DIRECTION VS POWER:

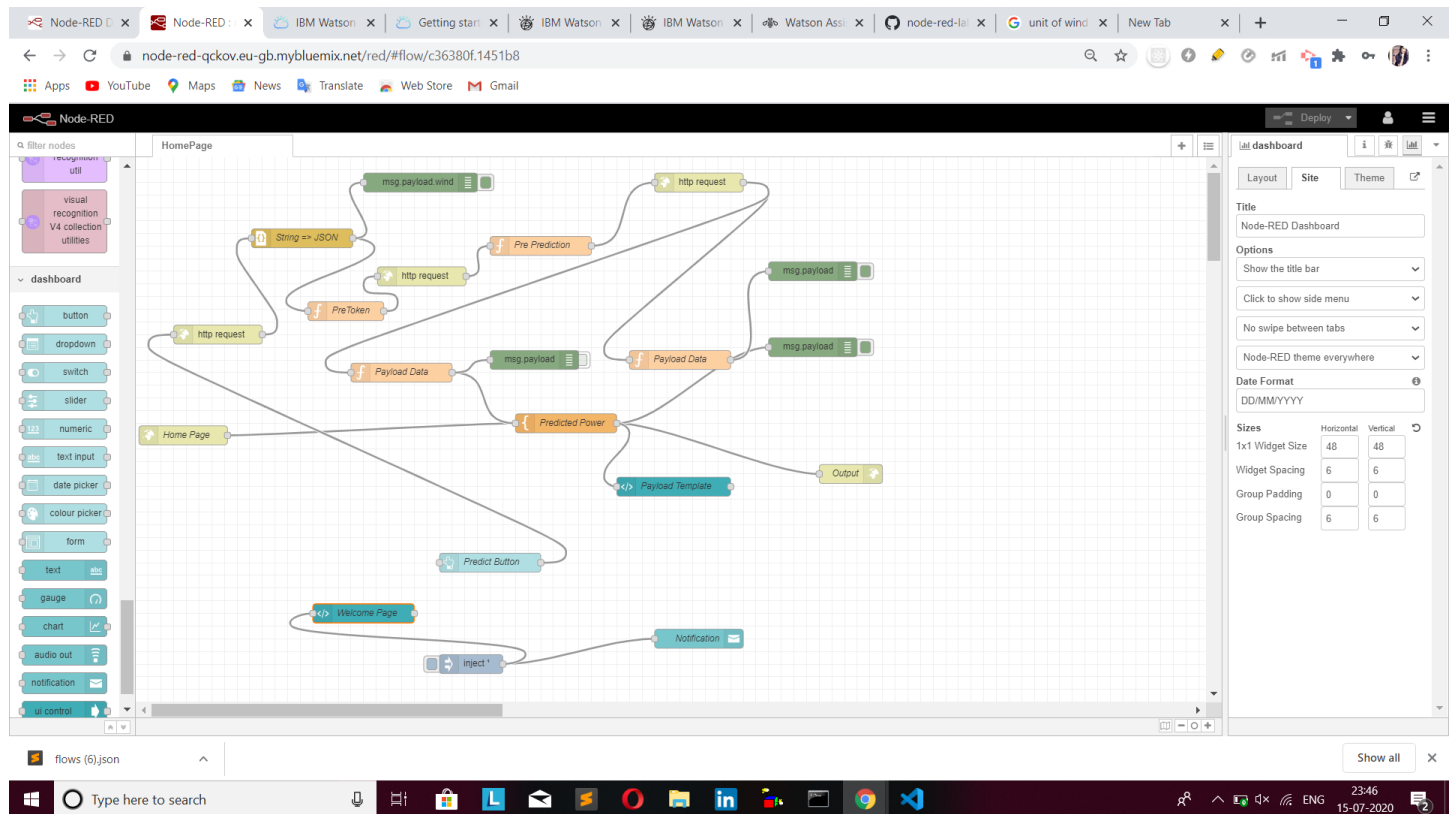


GRAPH SHOWING WINDSPEED VS POWER :



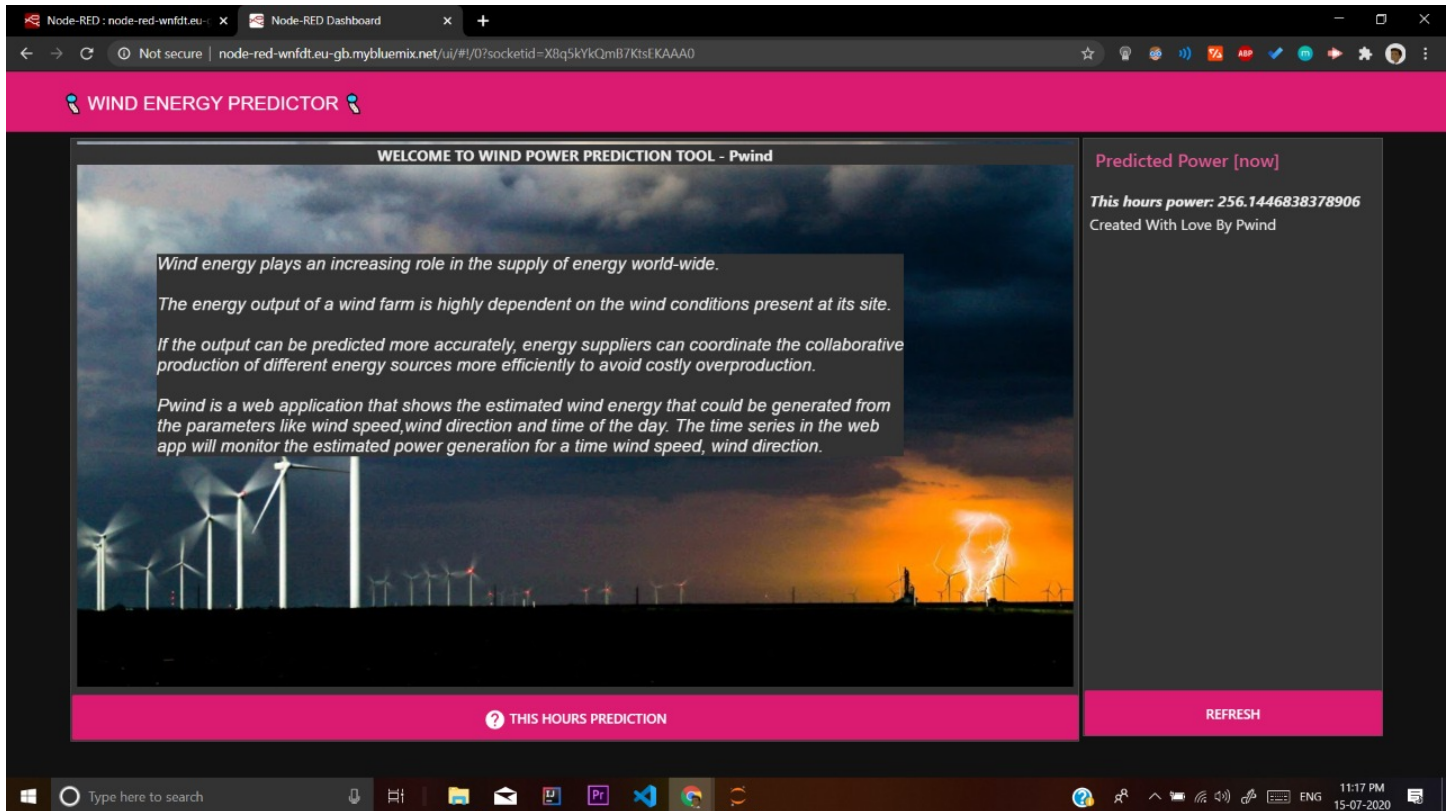
5. NODE RED APPLICATION :

We started our UI through creating flows in Node-RED. To provide real time weather report to our model, we looked forward to a weather API, which actually returns a json object containg required weather information to our node-RED flow.



5. WEB APP :

Finally the entire web application was integrated using git repository.



LINK TO OUR NODE RED APP FOR WIND POWER PREDICTOR :

<http://node-red-wnfddt.eu-gb.mybluemix.net/ui/>

GROUP MEMBERS :

1. AMAN KUMAR
2. JASHMIN MISHRA
3. GURUDEEP SINGH