Data Collection :

For This Wind Predication Model I Collect Data From
https://www.kaggle.com/berkerisen/wind-turbine-scada-dataset

Import Data :

After Downlaod The Wind Dataset From Above Link I Import Data Into Our Jupityer Notebook /Watson Studio Notebook By Following Code

Jupityer Notebook

```
df = pd.read_csv('C:\Users\Deepak Singh\Downloads\T1.csv',encoding = "utf-8")
```

Watson Studio Notebook

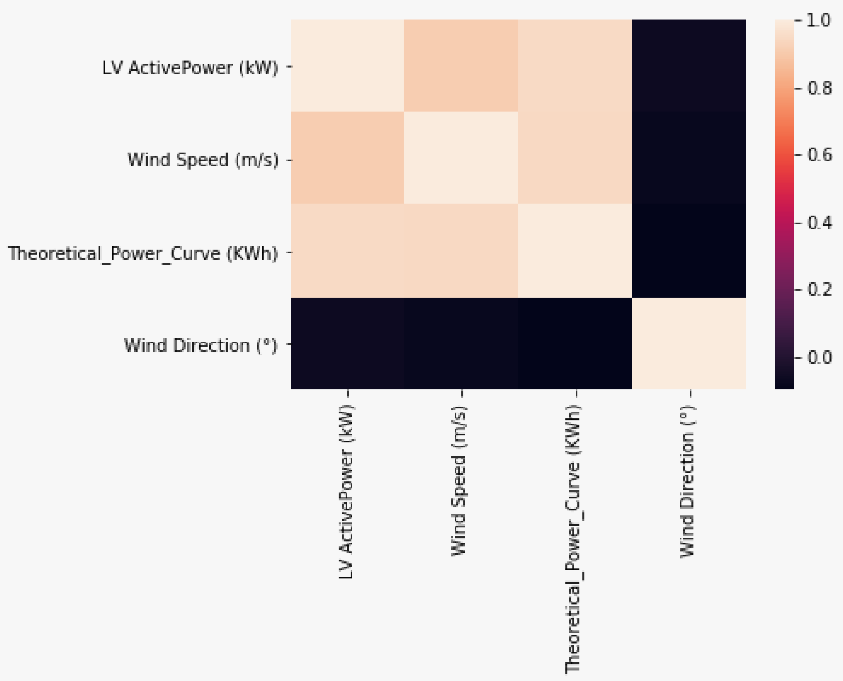
First I upload dataset on cloud stroage of IBM Cloud and from rigth section of notebook i import csv file as Pandas Dataframe and following code generated automatically in watson notebook.

```
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_9c69e53f6bf24045a870b6ce82b683ec = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='8TFJ2E4o2q72jqru2421UKRyvS_JNFqM0ICwVmI8C08G',
    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_9c69e53f6bf24045a870b6ce82b683ec.get_object(Bucket='newwind-donotdelete-pr-0dk3fsjy2
zdd3m',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,encoding='unicode_escape',sep='\t')
```

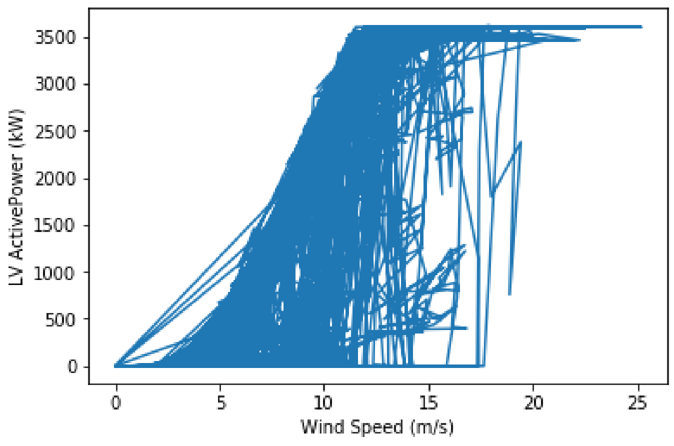
Data Visualize :

To Relate all Features of Data set and Find the what kind of correlation is held by attributes of data or find which features correspond to output we simply draw multivariate plots between all features and check from plots which feature is correspond to output.

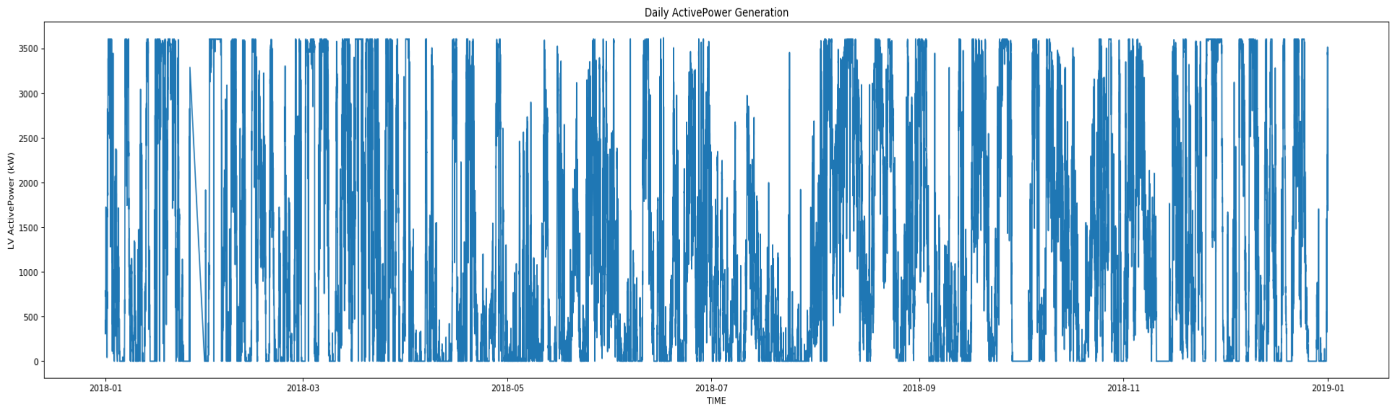
For correlate all features we use seaborn heatmaps and draw heatmap by using sns.heatmap(df.corr()) command.



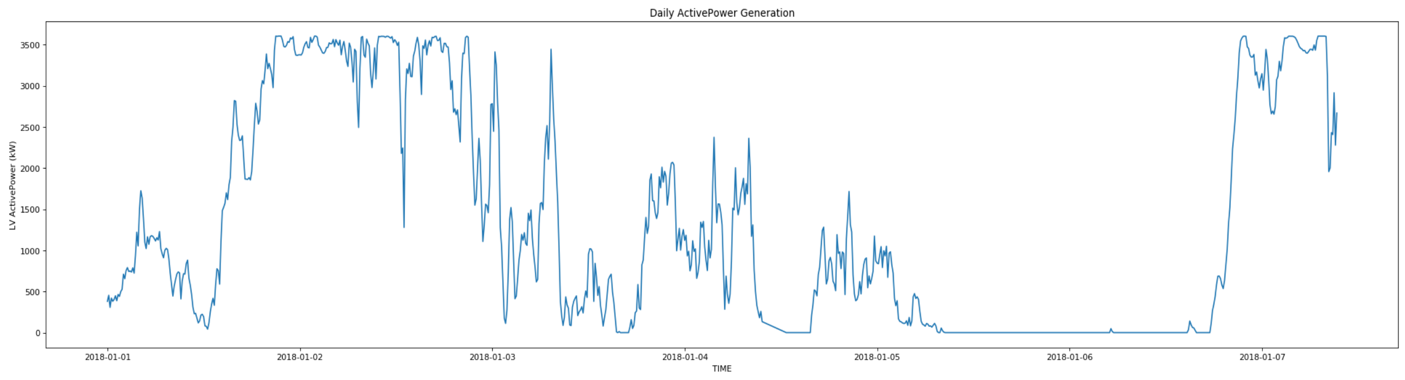
From above heatmaps we can view the correlation between all the features and we observe that Theoretical Power is very corresponding to ouput. Further we also draw individual plots of all features by matplotlib.



This is the graph we plot to observe the relation between Wind Speed and the output(LV Active Power).
We also Developed Time Series Graph To See How our Prediction is vary according to Date and Time. Here Below Graph is Between show relation between time and LV active power (Yearly)



Here Below Graph is Between show relation between time and LV active power (Weekly)

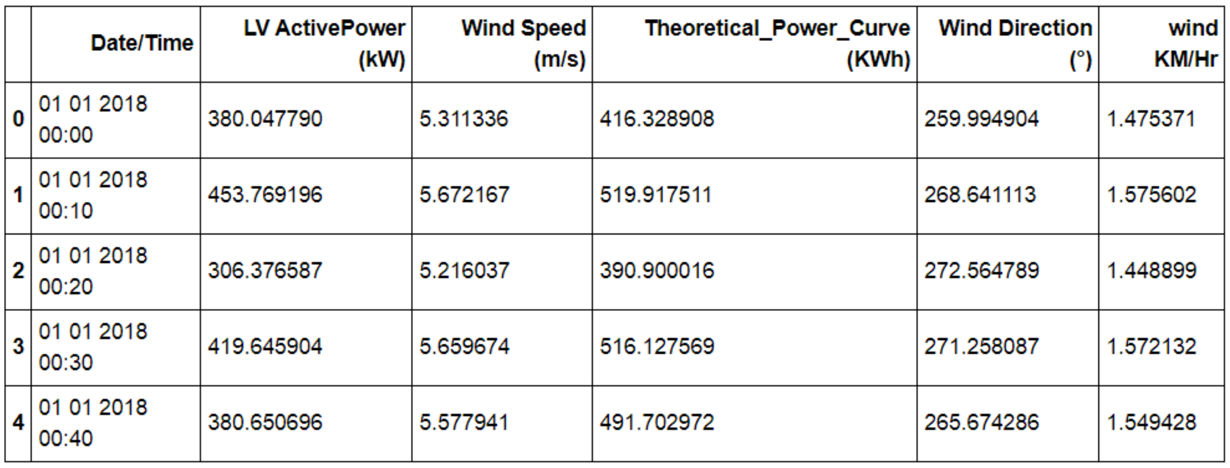


Feature Selection :

Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output. From above Heatmaps and Univarient graph we can see that wind speed a Theoretical Power contibute most to prediction output so we assume the wind speed is contribute more important that theoretical power because for calculating theoretical power we need wind parameter so we don't include that theoretical power.

1. **Create New Feature** : As we know wind speed is most closer to predication we add new feature called wind speed (km/h) to see that our model is give more accuracy as compared to base model.

x1 = x[:,0] * 5/18 This Command add wind speed (km/h) to our dataset.

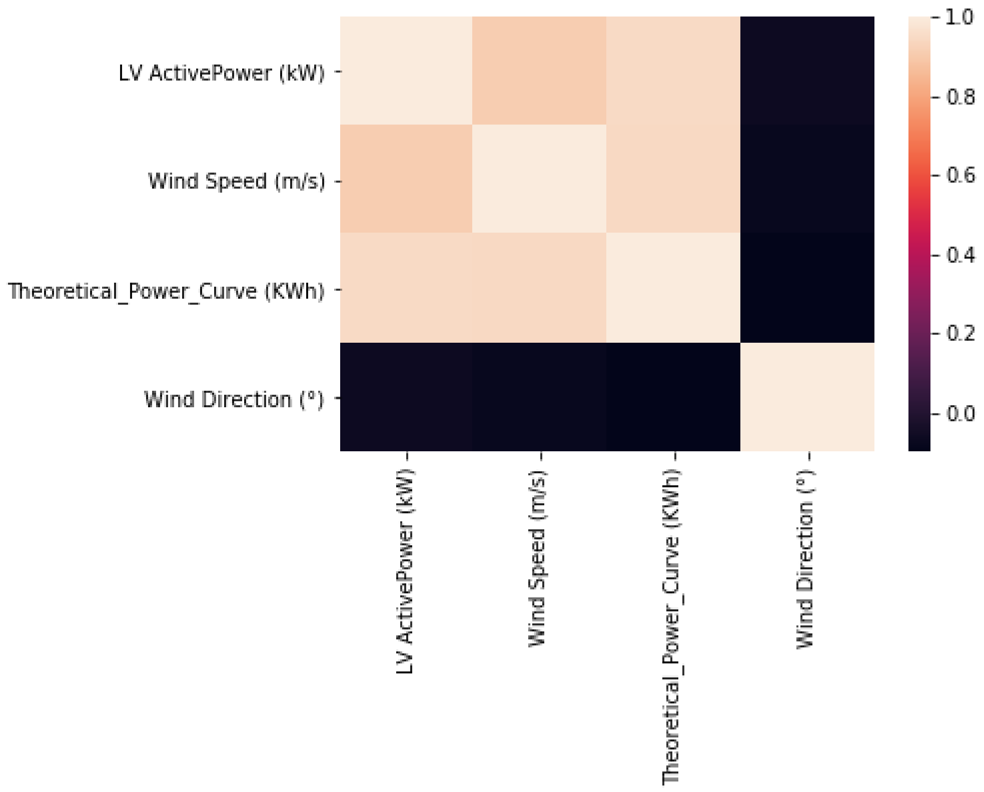


	Date/Time	LV ActivePower (kW)	Wind Speed (m/s)	Theoretical_Power_Curve (KWh)	Wind Direction (°)	wind KM/Hr
0	01 01 2018 00:00	380.047790	5.311336	416.328908	259.994904	1.475371
1	01 01 2018 00:10	453.769196	5.672167	519.917511	268.641113	1.575602
2	01 01 2018 00:20	306.376587	5.216037	390.900016	272.564789	1.448899
3	01 01 2018 00:30	419.645904	5.659674	516.127569	271.258087	1.572132
4	01 01 2018 00:40	380.650696	5.577941	491.702972	265.674286	1.549428

in dataset a new column wind speed(km/hr) is added.

2. **Select Best Feature :** From many observation and with the help of graph we determine that overall Wind Speed is responsible for predicating the energy output.

3. **Correlation :** for correlation we draw another heatmeaps to see correalted values.



Model Building :

We build our model is using XGboost algorithm first we build pipelines using some algorithm and select best algorithm and build model on them and further we try with neural network.

Pipeline :

```
pipeline_lr=Pipeline([('lr_classifier',LinearRegression())])
pipeline_svm=Pipeline([('dt_classifier',svm.SVR())])
pipeline_xgb=Pipeline([('rf_classifier',xg.XGBRegressor())])
```

Create Node Red:

This below is to Create a Node Red Instance on Ibm Cloud and all steps are taken from https://developer.ibm.com/components/node-red/tutorials/how-to-create-a-node-red-starter-application/

Step 1.Find the Node-RED Starter in the IBM Cloud catalog

1. Log in to IBM Cloud.
2. Open the catalog and search for node-red.
3. Click on the Node-RED App tile.

This will show you an overview of the Starter Kit and what it provides.

Step 2. Create your application

Now you need to create the Node-RED Starter application.

1. On the *Create* tab, a randomly generated App name will be suggested. Either accept that default name or provide a unique name for your application. This will become part of the application URL.
2. *Note:* If the name is not unique, you will see an error message and you must enter a different name before you can continue.
3. The Node-RED Starter application requires an instance of the Cloudant database service to store your application flow configuration. Select the region the service should be created in and what pricing plan it should use.
4. *Note:* You can only have one Cloudant instance using the Lite plan. If you have

already got an instance, you will be able to select it from the Pricing plan select box. You can have more than one Node-RED Starter application using the same Cloudant service instance.

5. Click the Create button to continue. This will create your application, but it is not yet deployed to IBM Cloud.

Step 3. Enable the Continuous Delivery feature

At this point, you have created the application and the resources it requires, but you have not deployed it anywhere to run. This step shows how to setup the Continuous Delivery feature that will deploy your application into the Cloud Foundry space of IBM Cloud.

1. On the next screen, click the Deploy your app button to enable the *Continuous Delivery* feature for your application.
2. You will need to create an IBM Cloud API key to allow the deployment process to access your resources. Click the New button to create the key. A message dialog will appear. You can accept the default values and confirm to close the dialog.
3. Increase the Memory allocation per instance slider to at least 128MB. If you do not increase the memory allocation, your Node-RED application might not have sufficient memory to run successfully.
4. The Node-RED Starter kit only supports deployment to the Cloud Foundry space of IBM Cloud. Select the region to deploy your application to. This should match the region you created your Cloudant instance in. Lite users might only be able to deploy to your default region.
5. Click Next to continue.
6. Configure the DevOps toolchain by selecting the region it should be created in – again, try to match the region you selected previously.

7. Click Create. This will take you back to the application details page.
8. After a few moments, the Deployment Automation section will refresh with the details of your newly created Delivery Pipeline. The Status field of the pipeline will eventually show In progress. That means your application is being built and deployed.
9. Click on the Status field to see the full status of the Delivery Pipeline.
10. The Deploy stage will take a few minutes to complete. You can click on the View logs and history link to check its progress. Eventually the Deploy stage will go green to show it has passed. This means your Node-RED Starter application is now running.

Step 4. Open the Node-RED application

Now that you've deployed your Node-RED application, let's open it up!

1. Back on the application details page, you should now see the App URL, Source and Deployment target fields filled in.
2. Click on the App URL to open up your Node-RED application in a new browser tab.

Create Node UI :

1. Drag form node and create reuired values fields.

2. Drag function node name as Pre-Token node.

3. Drag a http request node to generate access token using https://iam.cloud.ibm.com/identity/token. and select method is POST.

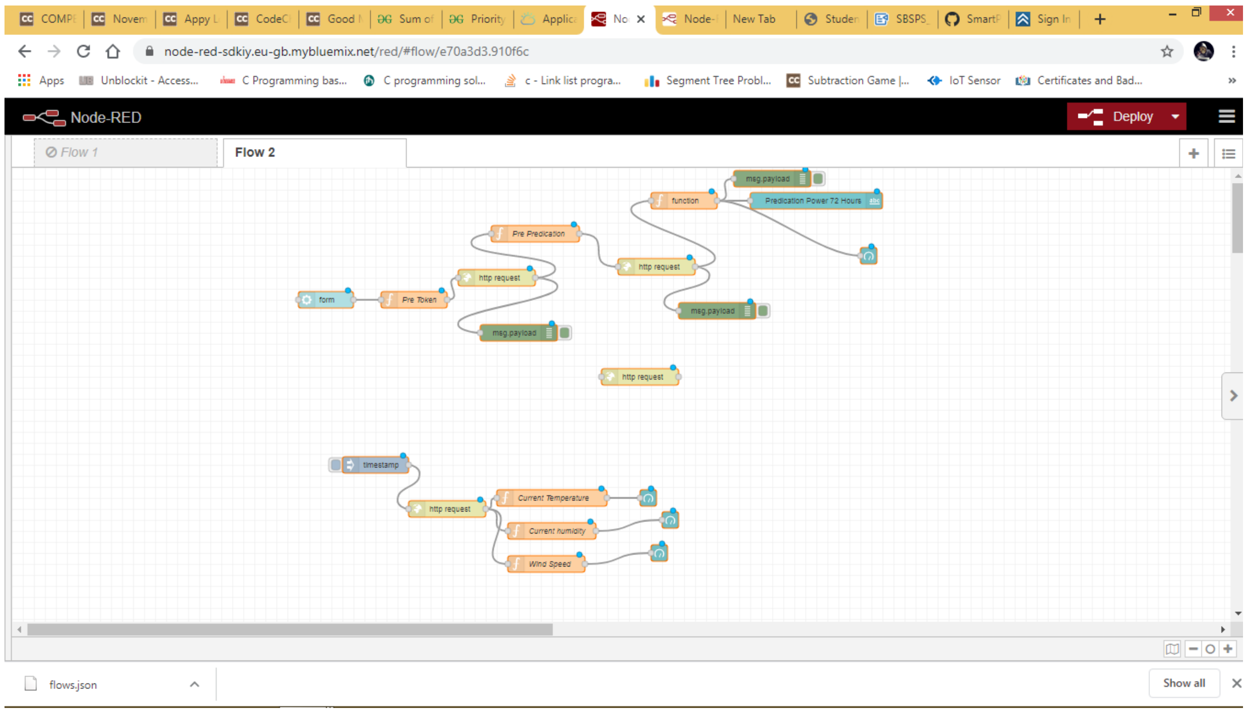
4. Drag function node name as Pre predication.

5. Drag a http request node select mehtod is POST and uri is scoring endpoint of model.

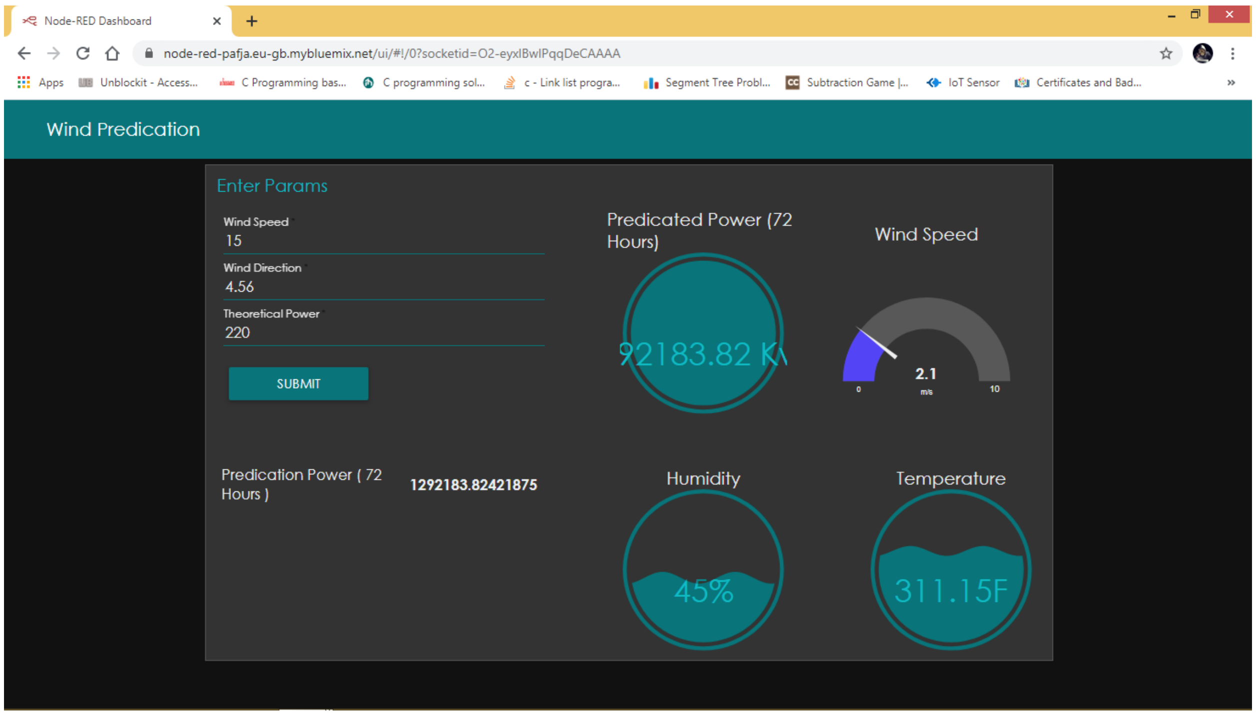
6. Drag a function node name as Output.

7. Drag a debug and text node to see prediction output in Node Red UI.

8. Connect all nodes with each other sequentially.



Below is screenshot of node red ui.



Wind Predication

Enter Params

Wind Speed
15

Wind Direction
4.56

Theoretical Power
220

Predicated Power (72 Hours)
92183.82 K\

Wind Speed
2.1

Predication Power (72 Hours) 1292183.82421875

Humidity
45%

Temperature
311.15F

Watson Studio :

here is step by step procedure to create a Watson studio project and configure with Machine Learning Services and Cloud Storage This following step is taken from https://developer.ibm.com/recipes/tutorials/create-an-ibm-watson-studio-project/

1. Login to IBM Cloud

2. Browse to Catalog

Search for Watson Studio. Watson Studio under AI gets listed.

3. Click on Watson Studio to create

Provide the Service Name. Choose a region / location to Deploy In. Select a resource group. Select the Pricing Plan that works for you. Click on create

4. Get Started

The Previous step completes deployment and Watson Studio would be ready for usage. Click on Get Started. This loads user Information.

5. **Create New Project**

   Click on New Project – You can select any of the options available. For Ex: select Experiment Assistant -> This automatically provisions the required Watson Machine Learning and Cloud Object Storage Services.
   If you do not have those services already provisioned, select the region in which you want those services to be provisioned and create the services and have them provisioned. When you do, you can see the service instance names for cloud storage and Watson Machine Learning services.

6. **Name the Project and descripe**

   Provide Project Name and Description and then click create.

7. **Create Access Token**

   All projects need a Access token that is used to access data assets. For ex: Files and Connections. These Access tokens are used by Platform APIs.
   So, in Settings, under Access Tokens, click on New Token. Provide a Name and Select Viewer or Editor as the roles and then click create.
   Now, if you go back to Access Tokens under Settings of your project, you would see the New Token you created.

After Successfully Perfrom this above Steps We create a notebook and AutoAI Model To predict the output of Wind Energy. but in our project we don,t use AutoAI model.

Build and ML Model On Watson Studio :

1. **Data Collection :**

   As above mention that we collect and download data from kaggle.

https://www.kaggle.com/berkerisen/wind-turbine-scada-dataset

2. **Import Data :**

```
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 i
ncludes your credentials.
# You might want to remove those credentials before you share the notebook.
client_9c69e53f6bf24045a870b6ce82b683ec =
ibm_boto3.client(service_name='s3',
 ibm_api_key_id='8TFJ2E4o2q72jqru2421UKRyvS_JNFqM0ICwVmI8C08G',
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_9c69e53f6bf24045a870b6ce82b683ec.get_object(Bucket='newwind-d
onotdelete-pr-0dk3fsjy2zdd3m',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,encoding='unicode_escape',sep='\t')
df.head()
```

3. **Data Visulization** : For Data Visulization we draw various plots.

```
# This command shows the relation between all features
sns.heatmap(dfs.corr())

# Graph between Theoretical_Power_Curve (KWh) And  "LV ActivePower (kW)
```

```
print(x[:,1].shape)
print(y.shape)
plt.plot(x[:,1],y)
plt.xlabel("Theoretical_Power_Curve (KWh)")
plt.ylabel("LV ActivePower (kW)")

# Graph between "Wind Direction (°)"  And  "LV ActivePower (kW)"
plt.plot(x[:,2],y)
plt.xlabel("Wind Direction (°)")
plt.ylabel("LV ActivePower (kW)")

# Graph between Wind Speed (m/s)"  And  "LV ActivePower (kW)"
plt.plot(x[:,0],y)
plt.xlabel("Wind Speed (m/s)")
plt.ylabel("LV ActivePower (kW)")

# THIS GRAPH SHOWS THAT LV ACTIVEPOWER GENERATION with respect to TIME
from matplotlib.pyplot import figure
figure(num=None, figsize=(30, 7), dpi=80, facecolor='w', edgecolor='k')
plt.plot(new_x,y)
plt.xlabel("TIME")
plt.ylabel("LV ActivePower (kW)")
plt.title("Daily ActivePower Generation")
plt.show()
```

4. **Feature Selection :** Select Best Feature for model

5. **Building Pipeline :** We build the pipeline using linear regression, support vector machines and XGboost and check which algorithm gives better accuracy.

```
pipeline_lr=Pipeline([('scalar1',StandardScaler()),
          ('lr_classifier',LinearRegression())])
pipeline_svm=Pipeline([('scalar2',StandardScaler()),
          ('dt_classifier',svm.SVR())])
pipeline_xgb=Pipeline([('scalar3',StandardScaler()),
            ('rf_classifier',xg.XGBRegressor())])
```

6. **Split Dataset into Train Data and Test Data :**

```
x= df.iloc[:,2:5].values
y=df.iloc[:,1].values
```

7. **Train Model**

```
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size=0.15,
random_state=42)

my_model=xg.XGBRegressor()
my_model.fit(x,y)
```

8. **Test the Model using Test dataset**

```
y_pred = my_model.predict
```

Deploy Model :

1. Create Machine Learning Credentials

```
from watson_machine_learning_client import WatsonMachineLearningAPIClient

wml_credientials = {
"apikey": "YcmRf3furEZvRswGtqFDQ2NoshyhwjW-Ul9--Kla-CaA",
"iam_apikey_description": "Auto-generated for key 13f2067b-a828-461a-81ae-4fb79d03d74e",
"iam_apikey_name": "New_Wind_Credientials_XGB",
"iam_role_crn": "crn:v1:bluemix:public:iam::::serviceRole:Writer",
"iam_serviceid_crn":
"crn:v1:bluemix:public:iam-identity::a/8c70ccdf93094ab4a01648da17c10a0f::serviceid:
ServiceId-ee2aadf4-8499-463e-b1d3-168e5745caf2",
"instance_id": "ce4c95b7-0bf0-4e31-8a40-2fa03f9b7a35",
"url": "https://eu-gb.ml.cloud.ibm.com"
}

client = WatsonMachineLearningAPIClient(wml_credientials)
```

2. Create Metadata

```
 metadata = {
                    client.repository.ModelMetaNames.NAME : 'Wind Predication XGB',
                   client.repository.ModelMetaNames.AUTHOR_NAME : 'Deepak Singh',
                   client.repository.ModelMetaNames.AUTHOR_EMAIL : 'ds467625@gmail.com'
}
```

3. Save Model

```
stored_data = client.repository.store_model(my_model,meta_props = metadata)
```

4. Extract GUID

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

5. Deploy Model from GUID

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

Synchronous deployment creation for uid: '77a8a195-5a5b-472c-b4c6-52dbb12878c6'
started

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


INITIALIZING
DEPLOY_SUCCESS
```

```
------------------------------------------------------------------------------------
Successfully finished deployment creation,
deployment_uid='49173376-dba5-4262-9486-3e68515415a8'
------------------------------------------------------------------------------------
```

Configure Model With Node Red Flow

1. Drag form node and create reuired values fields.

2. Drag function node name as Pre-Token node and write below code in pre token function body.

```
global.set("sp",msg.payload.sp)
global.set("de",msg.payload.de)
global.set("eg",msg.payload.eg)
var apikey = "YcmRf3furEZvRswGtqFDQ2NoshyhwjW-Ul9--Kla-CaA";
msg.headers = {"content-type":"application/x-www-form-urlencoded"}
msg.payload =
{"grant_type":"urn:ibm:params:oauth:grant-type:apikey","apikey":apikey}
return msg;.
```

   Here api key is taken from machine learning credientials.

3. Drag a http request node to generate access token using https://iam.cloud.ibm.com/identity/token. and select method is POST.

4. Drag function node name as Pre predication and write below code in pre predication.

```
var sp = global.get('sp')
var eg = global.get('eg')
var de = global.get('de')
var token = msg.payload.access_token
var instance_id = "ce4c95b7-0bf0-4e31-8a40-2fa03f9b7a35";
msg.headers =
{'Content-Type':'application/json',"Authorization":"Bearer"+token,"ML-instance-ID"
:instance_id}
msg.payload = {"fields":["fd","de","ped"],"values":[[sp,eg,de]]}
return msg;
```

Here Instance Id is take from Machine Learning Credientials.

5. Drag a http request node select mehtod is POST and uri is scoring endpoint of model.

   https://eu-gb.ml.cloud.ibm.com/v3/wml_instances/ce4c95b7-0bf0-4e31-8a40-2fa03f9b7a35/deployments/49173376-dba5-4262-9486-3e68515415a8/online

6. Drag a function node name as Output and write below code to function body

   ```
   msg.payload = msg.payload.values[0][0]
   return msg;
   ```

7. Drag a debug and text node to see prediction output in Node Red UI.

8. Connect all nodes with each other sequentially.