Prediction Of Full Load Electrical Power Output Of A Base Load Operated Combined Cycle Power Plant Using IBM Watson

Team Members

1. **Ananya Mahapatra - 19BAI10061**
2. **Sahil Ghule - 19BAI10076**
3. **Rishi Rajpal - 19BCY10027**
4. **Pranjal Roy - 19BAI10008**

1 INTRODUCTION

1.1 Overview

This Project examines and compares some machine learning regression methods to develop a predictive model, which can predict hourly full load electrical power output of a combined cycle power plant. The base load operation of a power plant is influenced by four main parameters, which are used as input variables in the dataset, such as ambient temperature, atmospheric pressure, relative humidity, and exhaust steam pressure. These parameters affect electrical power output, which is considered as the target variable. A web application is built to enter the inputs and view the result.

1.2 Purpose

The Combined Cycle Power Plant or combined cycle gas turbine, a gas turbine generator generates electricity and waste heat is used to make steam to generate additional electricity via a steam turbine. The gas turbine is one of the most efficient one for the conversion of gas fuels to mechanical power or electricity. Combined cycle power plants are frequently used for power production. These days prediction of power plant output based on operating parameters is a major concern.
Using this we predict the full load electrical power output of a base load power plant is important in order to maximize the profit from the available megawatt/hour.

2 LITERATURE SURVEY

2.1 Existing problem

Single-cycle gas turbine power plants generate electricity by using natural gas and compressed air. Air is drawn from the surroundings, compressed, and fed into the combustion chamber of the gas turbine. Here, natural gas is injected which mixes with the compressed air and ignited. The combustion produces a high-pressure, hot gas stream that flows through the turbine causing it to spin (at tremendous speeds). Consequently, this spins a generator which is connected to the turbine to produce electricity.

For single-cycle gas turbines, much of the energy is wasted as hot exhaust achieving an energy conversion efficiency of 35% at best. Combined cycle power plants exploit this inefficiency by capturing the waste heat using a heat recovery steam generator (HRSG), to produce even more power.

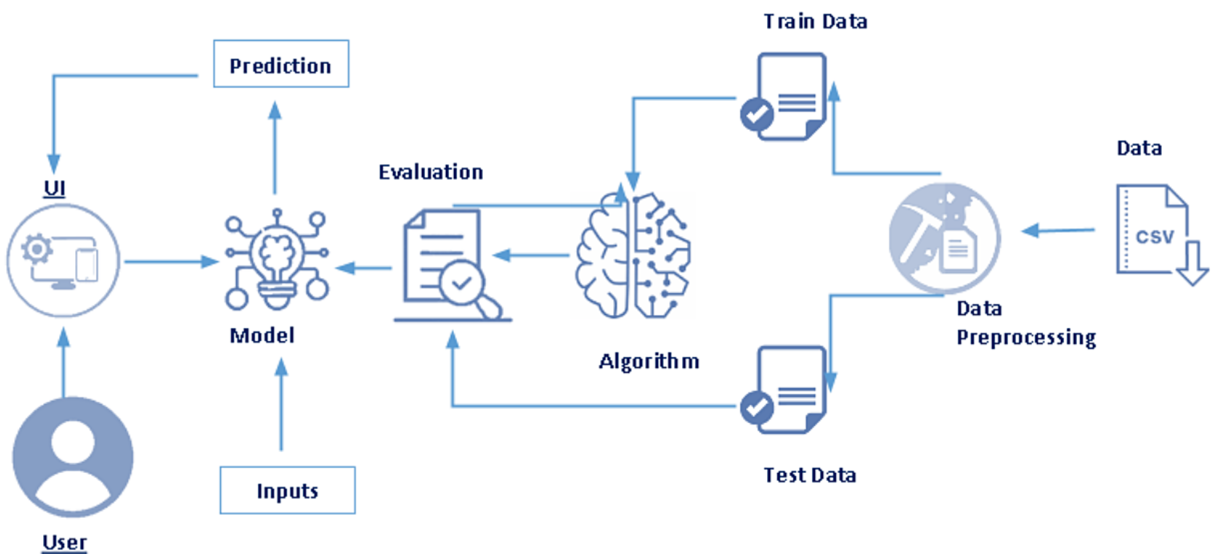
2.2 Proposed Solution

Combined cycle power plants are power generation plants that use both gas and steam turbines together to generate electricity. The waste heat generated from the gas turbine is used to produce steam which is fed to a steam turbine to generate even more electricity. This increases the power produced (up to 50% more) for the same amount of fuel, as well as increases the plant's efficiency to about 60%.

The Output power of the Combined Cycle Power Plant (CCPP) is dependent on a few parameters which are atmospheric pressure, exhaust steam pressure, ambient temperature, and relative humidity. Being able to predict the full load electrical power output is important for the efficient and economic operation of the power plant.

3 THEORETICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware/ Software requirement

1. Jupyter Notebook
2. Spyder
3. Visual studios
4. IBM Cloud
    a. IBM Watson studios
5. Flask
6. Packages:
    a. Pandas
    b. numpy
    c. scikit -learn
    d. matplotlib & seaborn
    e. Pickle

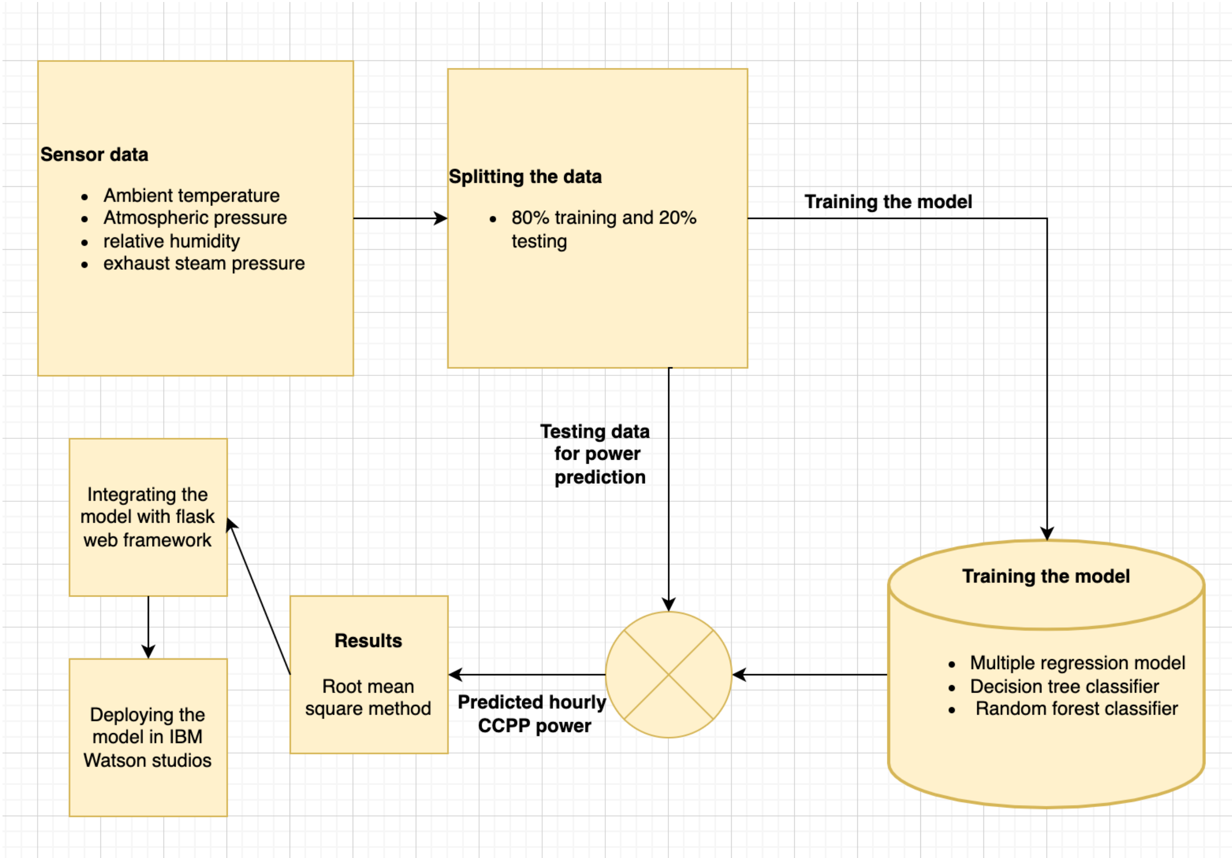
4 EXPERIMENTAL INVESTIGATIONS

1. The dataset which is used for building model is collected from a Combined Cycle Power Plant over 6 years (2006-2011).
2. The dataset contains 9568 datapoints.
3. We first build a model in jupyter notebook.
4. The data is split into training and testing set in 8:2 ratio.
5. The model is trained using Multiple regressor, Decision tree classifier and Random forest.
6. Random forest yields the most accurate result.
7. Then we integrate the machine learning model with the flask to build the web application.
8. The model is then deployed using IBM Watson Studios.

Project Workflow:

- User interacts with the UI (User Interface) to upload the input features.
- Uploaded features/input is analyzed by the model which is integrated.
- Once a model analyses the uploaded inputs, the prediction is showcased on the UI.

5 FLOWCHART



6 RESULT

After implementing our proposed model we (or the user) were successfully able to interact with the UI in order to predict the full load electrical power output of a base load power when provided with the information of influencing factors like temperature, Ambient pressure, Relative humidity, Exhaust Vaccum.

The prediction is later showcased in the UI.

7 ADVANTAGES & DISADVANTAGES

Our model uses random forest algorithm

Advantages:

- Random Forest is based on the **bagging** algorithm and uses **Ensemble Learning** technique. It creates as many trees on the subset of the data and combines the output of all the trees. In this way it **reduces overfitting** problem in decision trees and also **reduces the variance** and therefore **improves the accuracy.**
- **No feature scaling required:** No feature scaling (standardization and normalization) required in case of Random Forest as it uses rule based approach instead of distance calculation.
- Random Forest algorithm is very **stable**. Even if a new data point is introduced in the dataset, the overall algorithm is not affected much since the new data may impact one tree, but it is very hard for it to impact all the trees.

Disadvantages:

- **Complexity:** Random Forest creates a lot of trees (unlike only one tree in case of decision tree) and combines their outputs. By default, it creates 100 trees in Python sklearn library. To do so, this algorithm requires much more computational power and resources. On the other hand decision tree is simple and does not require so much computational resources.
- **Longer Training Period:** Random Forest require much more time to train as compared to decision trees as it generates a lot of trees (instead of one tree in case of decision tree) and makes decision on the majority of votes.

8 APPLICATIONS

This project can be successfully used in predicting the full load electrical power output of a base load power plant in order to maximize the profit from the available megawatt/hour in case of combined cycle power plant.

9 CONCLUSION

Combined cycle power plants create electricity by combining gas and steam engines. The waste heat from the gas turbine is used to generate steam, which is then pumped into a steam turbine to generate even more power. This improves the quantity of electricity produced (up to 50% more) for the same amount of fuel while also increasing the plant's efficiency to around 60%. The project focuses on predicting the full load electrical power output of a base load power plant in order to maximize the profit from the available megawatt/hour in case of combined cycle power plant. We achieve this by taking the ambient temperature, atmospheric pressure, relative humidity, and exhaust steam pressure as inputs to build the model which will predict the net hourly electrical energy output (EP) of the plant. The dataset is split into 80% training set and 20% testing set. The model is then trained using multiple regressor, decision tree classifier and random forest. The model with random forest yeilds the most accurate prediction with 96.52% accuracy. We integrate the prepared model with flask application. Then it is deployed using IBM Watson Studios.

10 FUTURE SCOPE

Currently, the application needs manual inputs inorder to predict the net hourly electrical energy. So, it can be further integrated and synchronised with the power plant, so the influencing factors such as ambient temperature, atmospheric pressure, relative humidity, and exhaust steam can be set according to the current environment.
For the future work, other algorithm could be used for the comparison such as Support Vector Machines (SVM).

11 BIBLIOGRAPHY

- https://medium.com/analytics-vidhya/prediction-of-the-output-power-of-a-combined-cycle-power-plant-using-machine-learning-a2ca01848eea
- https://www.researchgate.net/publication/345906727_Predicting_the_power_of_a_co

mbined_cycle_power_plant_using_machine_learning_methods
- https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3945086
- https://www.sciencedirect.com/topics/engineering/combined-cycle-power-plant
- https://www.hindawi.com/journals/wcmc/2021/9966395/
- http://theprofessionalspoint.blogspot.com/2019/02/advantages-and-disadvantages-of-random.html

APPENDIX

a. Source Code

In [21]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
```

In [22]:
```
# import the dataset from specified location
import os, 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.

if os.environ.get('RUNTIME_ENV_LOCATION_TYPE') == 'external':
    endpoint_87f2abeb901f48948c62b90ea66c915a = 'https://s3.us.cloud-object-storage.appdomain.cloud'
else:
    endpoint_87f2abeb901f48948c62b90ea66c915a = 'https://s3.private.us.cloud-object-storage.appdomain.cloud'

client_87f2abeb901f48948c62b90ea66c915a = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='bs8CI0xrvWdrjs6m70vPmD84N0e-qlEkRl7qC6wUVSHt',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url=endpoint_87f2abeb901f48948c62b90ea66c915a)

body = client_87f2abeb901f48948c62b90ea66c915a.get_object(Bucket='predictionoffullloadelectricalpow-donotdelete-pr-9qhe

data = pd.read_excel(body.read())
data.head()
```

Out[22]:

	AT	V	AP	RH	PE
0	14.96	41.76	1024.07	73.17	463.26
1	25.18	62.96	1020.04	59.08	444.37
2	5.11	39.40	1012.16	92.14	488.56
3	20.86	57.32	1010.24	76.64	446.48
4	10.82	37.50	1009.23	96.62	473.90

In [23]:
```
data.shape
```

Out[23]: (9568, 5)

In [24]:
```
# showing the data from top 5
data.head()
```

Out[24]:

	AT	V	AP	RH	PE
0	14.96	41.76	1024.07	73.17	463.26
1	25.18	62.96	1020.04	59.08	444.37
2	5.11	39.40	1012.16	92.14	488.56
3	20.86	57.32	1010.24	76.64	446.48
4	10.82	37.50	1009.23	96.62	473.90

In [25]:
```
# showing the data from bottom 5
data.tail()
```

Out[25]:

	AT	V	AP	RH	PE
9563	16.65	49.69	1014.01	91.00	460.03
9564	13.19	39.18	1023.67	66.78	469.62
9565	31.32	74.33	1012.92	36.48	429.57
9566	24.48	69.45	1013.86	62.39	435.74
9567	21.60	62.52	1017.23	67.87	453.28

In [26]:
```
# Print a concise summary of a DataFrame.
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9568 entries, 0 to 9567
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   AT      9568 non-null   float64
 1   V       9568 non-null   float64
 2   AP      9568 non-null   float64
 3   RH      9568 non-null   float64
 4   PE      9568 non-null   float64
dtypes: float64(5)
memory usage: 373.9 KB
```

In [27]:
```
# Computes a summary of statistics pertaining to the DataFrame columns
data.describe()
```

Out[27]:

	AT	V	AP	RH	PE
count	9568.000000	9568.000000	9568.000000	9568.000000	9568.000000
mean	19.651231	54.305804	1013.259078	73.308978	454.365009
std	7.452473	12.707893	5.938784	14.600269	17.066995
min	1.810000	25.360000	992.890000	25.560000	420.260000
25%	13.510000	41.740000	1009.100000	63.327500	439.750000
50%	20.345000	52.080000	1012.940000	74.975000	451.550000
75%	25.720000	66.540000	1017.260000	84.830000	468.430000
max	37.110000	81.560000	1033.300000	100.160000	495.760000

In [28]:
```
# It returns the number of
# missing values in the data set
data.isnull().sum()
```

Out[28]:
```
AT    0
V     0
AP    0
RH    0
PE    0
dtype: int64
```

In [29]:
```
data.head()
```

Out[29]:

	AT	V	AP	RH	PE
0	14.96	41.76	1024.07	73.17	463.26
1	25.18	62.96	1020.04	59.08	444.37
2	5.11	39.40	1012.16	92.14	488.56
3	20.86	57.32	1010.24	76.64	446.48
4	10.82	37.50	1009.23	96.62	473.90

In [35]:
```
# dividing the data into input and output
x=data.drop(['PE'],axis=1)
y=data['PE']
```

In [36]:
```
# importing the train_test_split from scikit-learn
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.2, random_state = 0)
```

In [37]:
```
# Returns size of xtrain
xtrain.shape
```

Out[37]: (7654, 4)

In [38]:
```
# Returns size of xtest
xtest.shape
```

Out[38]: (1914, 4)

In [ ]:

In [39]:
```
# Linear Regression
from sklearn.linear_model import LinearRegression
# Initializing the model
LRmodel = LinearRegression()
```

In [40]:
```
# Train the data with Linear Regreesion model
LRmodel.fit(xtrain, ytrain)
```

Out[40]: LinearRegression()

In [41]:
```
LRpred=LRmodel.predict(xtest)
```

In [42]:
```
# Importing R Square library
from sklearn.metrics import r2_score
```

In [43]: # Checking for accuracy score with actual data and predicted data

In [43]:
```
# Checking for accuracy score with actual data and predicted data
LRscore=r2_score(ytest, LRpred)
LRscore
```

Out[43]: 0.9325315554761303

In [44]:
```
# Decision Tree regressor
from sklearn.tree import DecisionTreeRegressor
# Intializing the model
DTRmodel=DecisionTreeRegressor()
```

In [45]:
```
# Train the data with Linear Regreesion model
DTRmodel.fit(xtrain, ytrain)
```

Out[45]: DecisionTreeRegressor()

In [46]:
```
DTRpred=DTRmodel.predict(xtest)
```

In [47]:
```
# Checking for accuracy score with actual data and predicted data
DTRscore=r2_score(ytest, DTRpred)
DTRscore
```

Out[47]: 0.9211520401091312

In [ ]:

In [48]:
```
# Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor
# Initializing the model
RFmodel=RandomForestRegressor()
```

In [49]:
```
# Train the data with Random Forest model
RFmodel.fit(xtrain, ytrain)
```

Out[49]: RandomForestRegressor()

In [50]:
```
RFpred=RFmodel.predict(xtest)
```

In [51]:
```
# Checking for accuracy score with actual data and predicted data
RFscore=r2_score(ytest, RFpred)
RFscore
```

Out[51]:
```
0.9652444049695488
```

In [52]:
```
# saving the model
pickle.dump(RFmodel, open('CCPP.pkl','wb'))
```

In [53]:
```
```

```
Requirement already satisfied: ibm_watson_machine_learning in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-pack
ages (1.0.176)
Requirement already satisfied: packaging in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_wat
son_machine_learning) (20.9)
Requirement already satisfied: importlib-metadata in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (fro
m ibm_watson_machine_learning) (3.10.0)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_wats
on_machine_learning) (0.8.9)
Requirement already satisfied: ibm-cos-sdk==2.7.* in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (fro
m ibm_watson_machine_learning) (2.7.0)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_watso
n_machine_learning) (2021.10.8)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_watso
n_machine_learning) (1.26.6)
Requirement already satisfied: pandas<1.3.0,>=0.24.2 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages
(from ibm_watson_machine_learning) (1.2.4)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_wats
on_machine_learning) (2.25.1)
```

In [54]:
```
from ibm_watson_machine_learning import APIClient
wml_credentials = {
    "url": "https://us-south.ml.cloud.ibm.com",
    "apikey": "SlTrzsYL9Ll-BlUtD6L1zkgObEyykis5yU_JET3hQ9hK"
}
client = APIClient(wml_credentials)
```

In [55]:
```
def guid_from_space_name(client, space_name):
    space = client.spaces.get_details()
    return(next(item for item in space['resources'] if item['entity']["name"]==space_name)['metadata']['id'])
```

In [56]:
```
space_uid = guid_from_space_name(client, 'Prediction_Deployment_Model')
print("Space UID = "+space_uid)
```

```
Space UID = dfddbf3a-449d-4a3f-ad5f-9042247606a4
```

In [57]:
```
client.set.default_space(space_uid)
```

Out[57]:
```
'SUCCESS'
```

In [58]:
```
client.software_specifications.list()
```

```
-----------------------------  ------------------------------------  ----
NAME                           ASSET_ID                              TYPE
default_py3.6                  0062b8c9-8b7d-44a0-a9b9-46c416adcbd9  base
pytorch-onnx_1.3-py3.7-edt     069ea134-3346-5748-b513-49120e15d288  base
scikit-learn_0.20-py3.6        09c5a1d0-9c1e-4473-a344-eb7b665ff687  base
spark-mllib_3.0-scala_2.12     09f4cff0-90a7-5899-b9ed-1ef348aebdee  base
ai-function_0.1-py3.6          0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda  base
shiny-r3.6                     0e6e79df-875e-4f24-8ae9-62dcc2148306  base
tensorflow_2.4-py3.7-horovod   1092590a-307d-563d-9b62-4eb7d64b3f22  base
pytorch_1.1-py3.6              10ac12d6-6b30-4ccd-8392-3e922c096a92  base
tensorflow_1.15-py3.6-ddl      111e41b3-de2d-5422-a4d6-bf776828c4b7  base
scikit-learn_0.22-py3.6        154010fa-5b3b-4ac1-82af-4d5ee5abbc85  base
default_r3.6                   1b70aec3-ab34-4b87-8aa0-a4a3c8296a36  base
```

In [66]:
```
software_spec_uid = client.software_specifications.get_uid_by_name("default_py3.8")
software_spec_uid
```

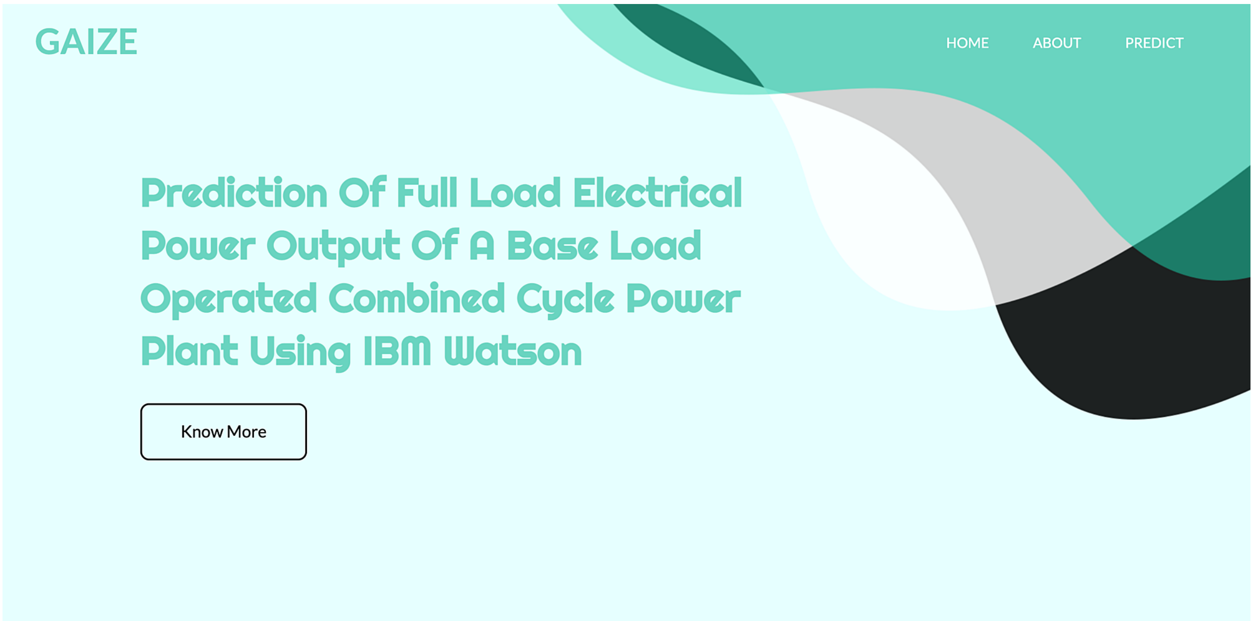
Out[66]:
```
'ab9e1b80-f2ce-592c-a7d2-4f2344f77194'
```

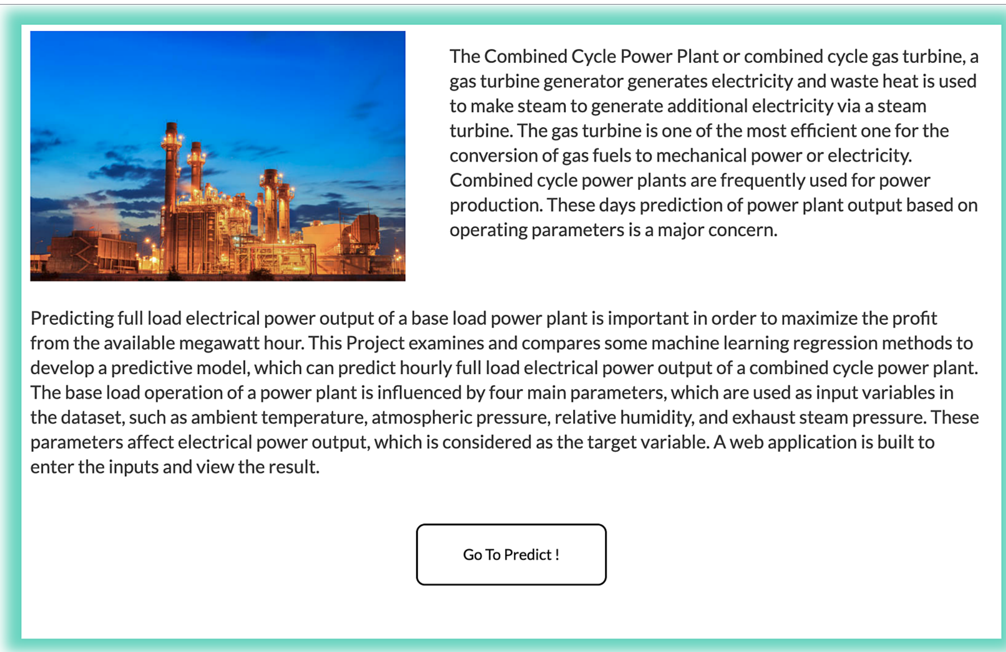
In [68]:
```
# Using Random Forest Model as it gave maximum accuracy of 96.52%

model_details = client.repository.store_model(model=RFmodel, meta_props={
    client.repository.ModelMetaNames.NAME: "Prediction_Model",
    client.repository.ModelMetaNames.TYPE: "scikit-learn_0.23",
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid
})
model_id = client.repository.get_model_uid(model_details)
model_id
```

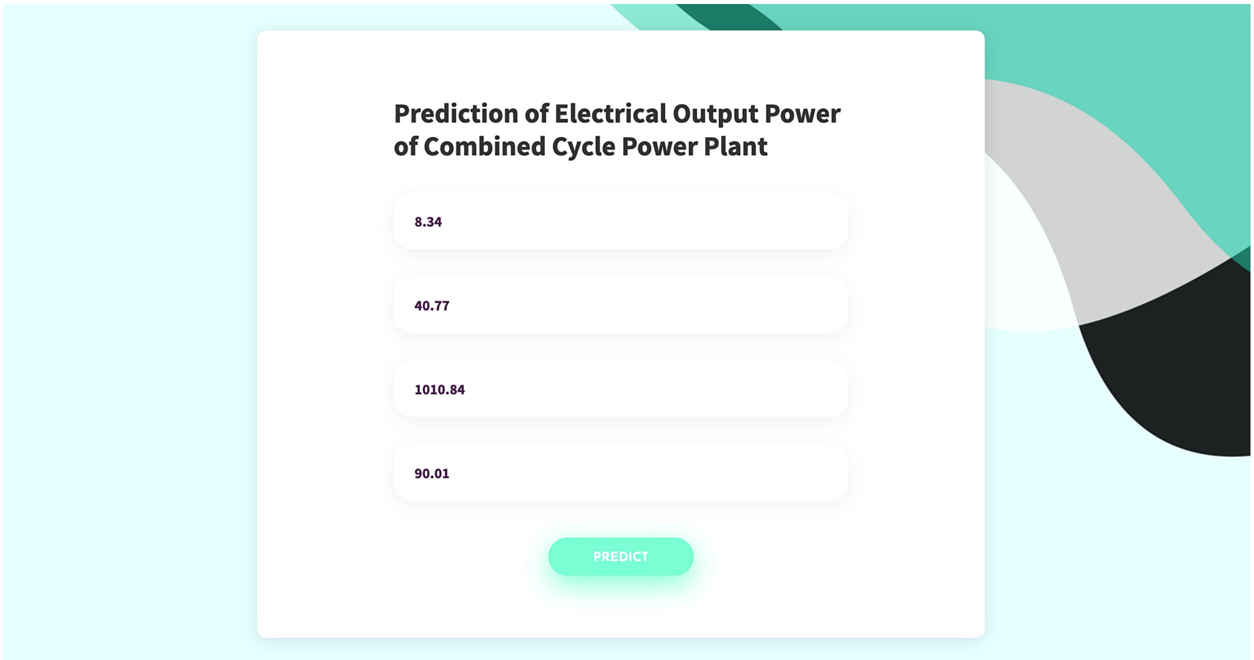
Out[68]:
```
'fc4353ca-c44c-4773-b983-9e6fa297c229'
```

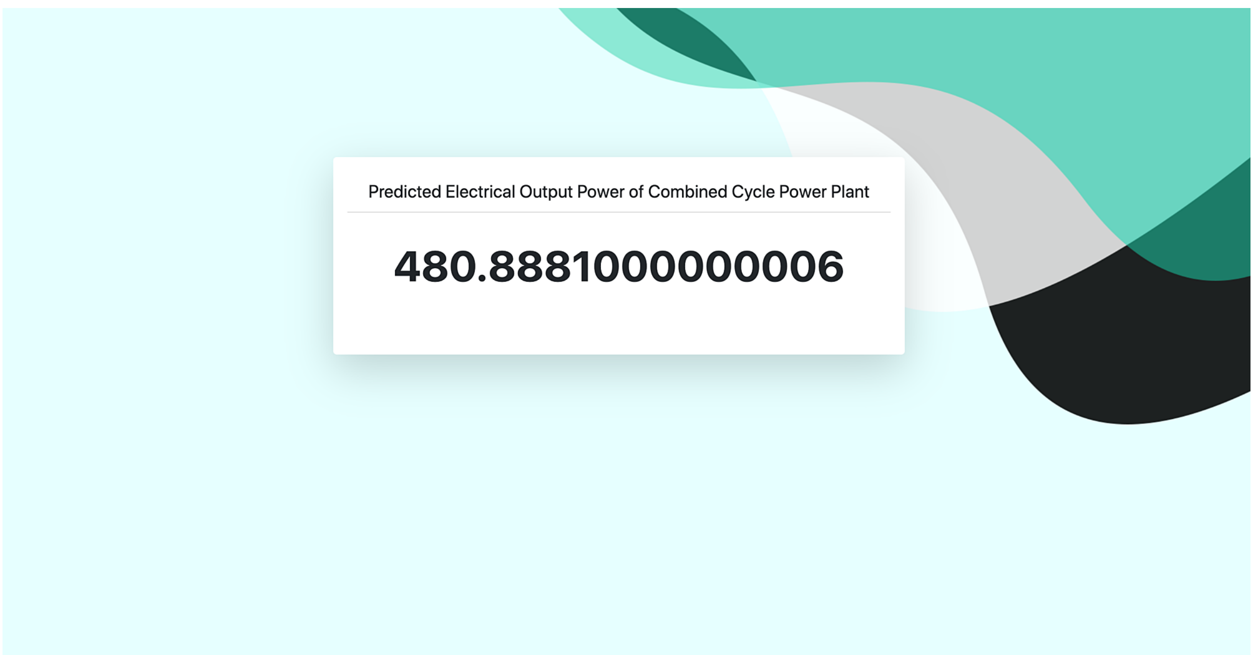
```python
from flask import Flask, render_template, request # Flask is a application
# used to run/serve our application
# request is used to access the file which is uploaded by the user in out application
# render_template is used for rendering the html pages
import pickle # pickle is used for serializing and de-serializing Python object structures

import requests

import json

# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "SlTrzsYL9Ll-BlUtD6L1zkgObEyykis5yU_JET3hQ9hK"
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=Flask(__name__) # our flask app

@app.route('/') # rendering the html template
def home():
    return render_template('home.html')
@app.route('/predict') # rendering the html template
def index() :
    return render_template("index.html")

@app.route('/data_predict', methods=['POST']) # route for our prediction
def predict():
    at = request.form['at'] # requesting for at data
    v = request.form['v'] # requesting for v data
    ap = request.form['ap'] # requesting for ap data
    rh = request.form['rh'] # requesting for rh data

    # coverting data into float format
    data = [[float(at), float(v), float(ap), float(rh)]]
    payload_scoring = {"input_data":[{"field": [['at', 'v', 'ap', 'rh',]],"values": data}]}
    response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/30246d1e-e48c-427c-90b8-
a0eb0ad79a0c/predictions?version=2022-01-19', json=payload_scoring, headers={'Authorization': 'Bearer ' + mltoken})
    # print(data)
    # print("Scoring response")
    # print(response_scoring.json())
    predictions =response_scoring.json()
    # print(predictions)
    print('Final Prediction Result',predictions['predictions'][0]['values'][0][0])
    pred = predictions['predictions'][0]['values'][0][0]
    # print(pred)




    # loading model which we saved
    #model = pickle.load(open('CCPP.pkl', 'rb'))

    #prediction= model.predict(data)[0]
    return render_template('predict.html', prediction=pred)

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

b. Output screenshot






Prediction of Electrical Output Power of Combined Cycle Power Plant
8.34
40.77
1010.84
90.01
PREDICT


Predicted Electrical Output Power of Combined Cycle Power Plant
480.8881000000006