PREDICT THE ELECTICAL POWER OUT FROM CCPP(COMBINED CYCLE POWER PLANT)

Submitted by

Sujitha V
Pushpa V
Padmappriya C
Venkatesh S

ABSTRACT:

- Predicting full load electrical power output of a base load power plant is important in order to maximize the profit from the available megawatt hours.

- This paper 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. The dataset, which consists of these input and target variables, was collected over a six-year period.

- First, based on these variables the best subset of the dataset is explored among all feature subsets in the experiments. Then, the most successful machine learning regression method is sought for predicting full load electrical power output.

INTRODUCTION:

- **Overview:**

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. Predicting full load electrical power output of a base load power plant is important in order to maximize the profit from the available megawatt hour.

- **Purpose:**

CCPP is designed as multi-shaft solution with gas and steam turbines to provide very flexible electricity production supporting peak, variable and intermediate load requirements

LITERATURE SURVEY:

- **Existing problem:**

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.
Predicting full load electrical power output of a base load power plant is important in order to maximize the profit from the available megawatt hour.

- **Proposed solution:**

  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.

EXPERIMENTAL INVESTIGATION:

- **Step1**:

Importing the libraries First step is usually importing the libraries that will be needed in the program.

- **Step 2:**

Import the Dataset We will need to locate the directory of the CSV file at first (it's more efficient to keep the dataset in the same directory as your program) and read it using a method called read_csv which can be found in the library called pandas.

- **Step 3:**

Taking Care of missing Data Sometimes you may find some data are missing in the dataset. We need to be equipped to handle the problem when we come across them. Obviously you could remove the entire line of data but what if you are unknowingly removing crucial information? Of course we would not want to do that. One of the most common ideas to handle the problem is to take a mean of all the values of the same column and have it to replace the missing data.

- **Step 4:**

Label Encoding Sometimes in the dataset we will find textual data like names, countries states, then the machine cannot do mathematical operations or cannot understand the

textual data. So the textual data are to be converted in to numerical format which is called as label encoding. we make use of label Encoder class to convert textual data in to Numerical data. In the given dataset country has textual data so we will be converting that particular columns textual data to numerical values.

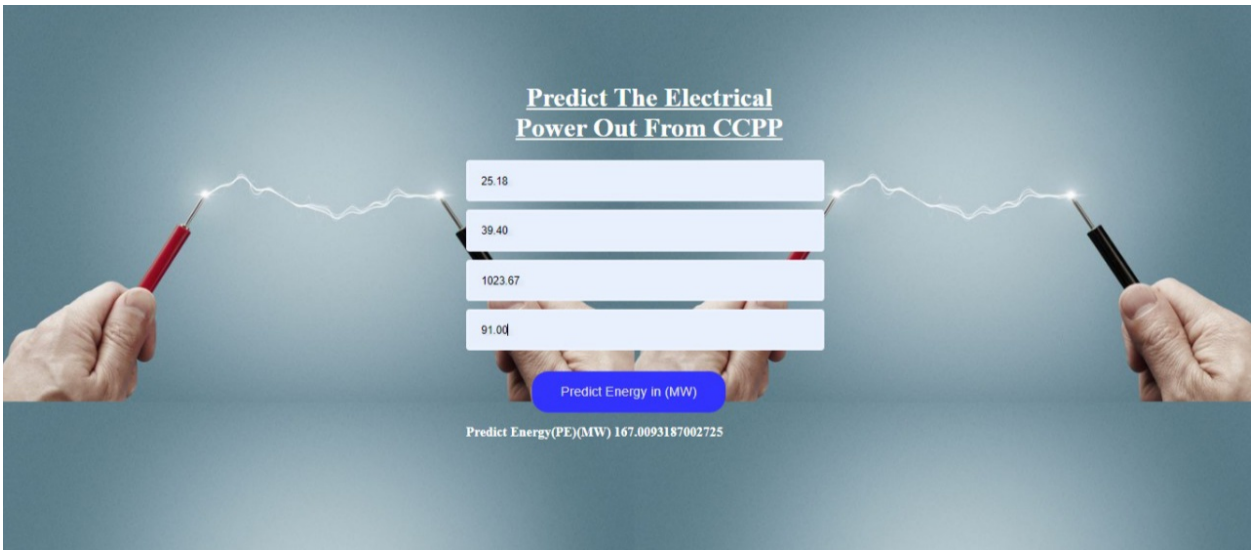
✓ **Step 5:**

Splitting The dataset in to Train set and Testing set Now we need to split our dataset into two sets — a Training set and a Test set. A general rule of the thumb is to allocate 80% of the dataset to training set and the remaining 20% to test set. For this task, we will import test_train_split from model_selection library of scikit. Now to build our training and test sets, we will create 4 sets— X_train (training part of the matrix of features), X_test (test part of the matrix of features), Y_train (training part of the dependent variables associated with the X train sets, and therefore also the same indices) , Y_test (test part of the dependent variables associated with the X test sets, and therefore also the same indices). We will assign to them the test_train_split, which takes the parameters — arrays (X and Y), test_size (if we give it the value 0.5, meaning 50%, it would split the dataset into half. Since an ideal choice is to allocate 20% of the dataset to test set, it is usually assigned as 0.2. 0.25 would mean 25%, just saying).

✓ **Step 6:**

Feature scaling: The final step of data preprocessing is to apply the very important feature scaling. It is a method used to standardize the range of independent variables or features of data. A lot of machine learning models are based on Euclidean distance. If, for example, the values in one column (x) is much higher than the value in another column (y), (x2-x1) squared will give a far greater value than (y2-y1) squared. So clearly, one square difference dominates over the other square difference. In the machine learning equations, the square difference with the lower value in comparison to the far greater value will almost be treated as if it does not exist. We do not want that to happen. That is why it is necessary to transform all our variables into the same scale. There are several ways of scaling the data. One way is called Standardization which may be used.

RESULT:



ADVANTAGES AND DISADVANTAGES:

- **High overall plant efficiency:** Efficiencies exceesing 50% can be attained.

- **Low investment costs:**

Because 2/3 of the output is produced in a GT and only 1/3 in the simple ST, the investment costs required are approximately 30% less than those for a conventional steam power plant.

- **Small amount of water required:**

The amount of cooling water required is only about 40 to 50% as much as for a steam plant.

- The technologies needed are more expensive and complex, so initial investments for building a **plant** is high.
- The Maintenance cost is high.
- It isn't suitable as a peak load **plant**.
- The natural **gas** used is a non_renewable and a highly flammable source.

CONCLUSION:

In this study, combined cycle power plants were investigated by energy, exergy and thermo economic analysis. General methodologies of these methods were discussed and also applied to case studies. The operating parameters of combined cycle power

plants were chosen to study their effect on overall thermal efficiency and exergy destruction in different components. An empirical correlation was determined for different set of operating variables and assessment parameters. Cost analysis was applied to a four stage intercooling, four stage reheating and three stage regenerating combined cycle by using software, cycle pad. Five different configurations were made and analyzed on cost basis. Cost of electricity production per MWh was calculated for each configuration.

FUTURE SCOPE:

In this thesis, an advance thermodynamic technique was used to analyze the different case studies of combined cycle power plant. After accomplishing energy, exergy and thermo economic analysis, a basis of starting optimization is ready to use. Thermo economic optimization and multidisciplinary optimization could be carried out for future works. Exergy and its relation to environment is an advance topic for future studies. Many researcher think about environment problems like air pollution, solid waste disposal etc may be reduce by using exergy techniques. More work is needed for improving the topics of exergy and its relation with economics, exergy and its relation to environments.

BIBILOGRAPHY:

- Jupyter
- Spyder
- Visual Studoi Code
- Flask
- HTML,CSS and Bootstrap(front-end Frame work)

APPENDIX:

Source Code:

- Notebook:

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [2]:
```
dataset = pd.read_csv('file:///C:/Users/PADMAPPRIYA/Downloads/dataset.csv')
dataset
```

Out[2]:

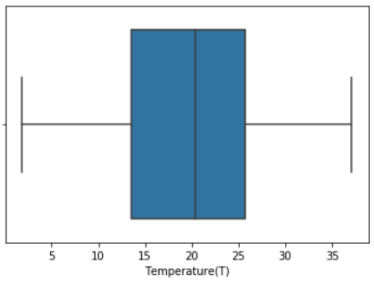
	Temperature(T)	Exahust vaccum(V)	Ambient pressure(AP)	Relative Humidity(RH)	Predict Energy(PE)(MW)
0	14.96	41.76	1024.07	73.17	463.26
1	25.18	62.96	1020.04	59.08	444.37
2	5.11	39.40	1012.16	92.14	488.56
3	20.86	57.32	1010.24	76.64	446.48
4	10.82	37.50	1009.23	96.62	473.90
5	26.27	59.44	1012.23	58.77	443.67
6	15.89	43.96	1014.02	75.24	467.35
7	9.48	44.71	1019.12	66.43	478.42
8	14.64	45.00	1021.78	41.25	475.98
9	11.74	43.56	1015.14	70.72	477.50
10	17.99	43.72	1008.64	75.04	453.02

In [3]:
```
dataset.isnull().any()
```

Out[3]:
```
Temperature(T)            False
Exahust vaccum(V)         False
Ambient pressure(AP)      False
Relative Humidity(RH)     False
Predict Energy(PE)(MW)    False
dtype: bool
```

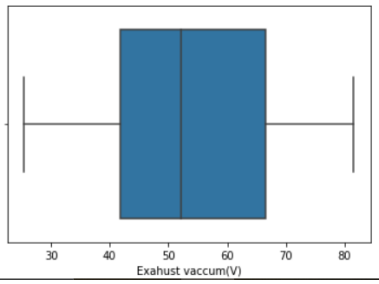
In [4]:
```
import seaborn as sns
sns.boxplot(dataset["Temperature(T)"])
```

Out[4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x23ced5624a8>
```



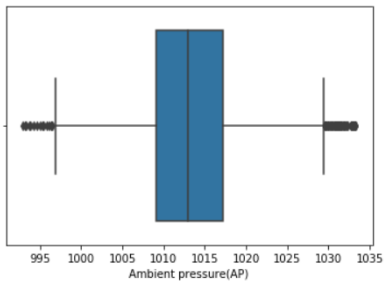
In [5]:
```
sns.boxplot(dataset["Exahust vaccum(V)"])
```

Out[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x23cee82aa20>
```



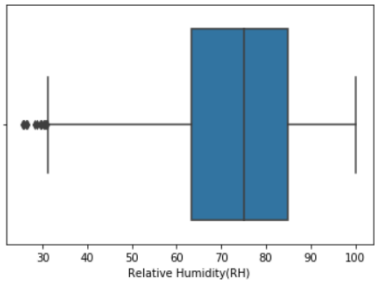
In [6]:
```
sns.boxplot(dataset["Ambient pressure(AP)"])
```

Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x23cee8c5f28>
```



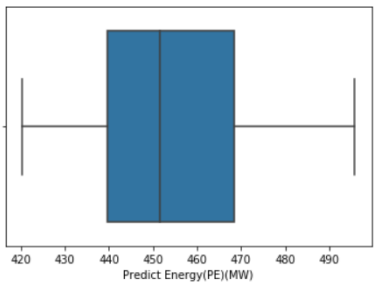
In [7]:
```
sns.boxplot(dataset["Relative Humidity(RH)"])
```

Out[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x23cee9336d8>
```



In [8]:
```
sns.boxplot(dataset["Predict Energy(PE)(MW)"])
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x23cee98ff60>
```



In [9]:
```
q1 = dataset.quantile(0.25)  #0.25, quartile 1 is 25%,
q3 = dataset.quantile(0.75)  #0.75, quartile 3 is 75%,
IQR = q3-q1
IQR
```

Out[9]:
```
Temperature(T)            12.2100
Exahust vaccum(V)         24.8000
Ambient pressure(AP)       8.1600
Relative Humidity(RH)     21.5025
Predict Energy(PE)(MW)    28.6800
dtype: float64
```

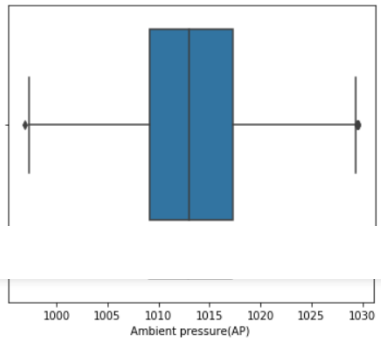
In [10]:
```
boston_iQr_outlier = dataset[~((dataset<(q1-1.5*IQR))|(dataset>(q3 + 1.5*IQR))).any(axis=1)]
boston_iQr_outlier
```

Out[10]:

	Temperature(T)	Exahust vaccum(V)	Ambient pressure(AP)	Relative Humidity(RH)	Predict Energy(PE)(MW)
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
5	26.27	59.44	1012.23	58.77	443.67
6	15.89	43.96	1014.02	75.24	467.35
7	9.48	44.71	1019.12	66.43	478.42
8	14.64	45.00	1021.78	41.25	475.98
9	11.74	43.56	1015.14	70.72	477.50
10	17.99	43.72	1008.64	75.04	453.02

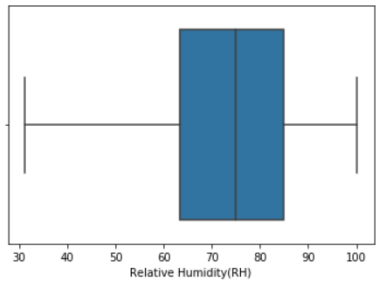
In [11]:
```
sns.boxplot(x=boston_iQr_outlier["Ambient pressure(AP)"])
```

Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x23ceea47b00>
```



In [12]:
```
sns.boxplot(boston_iQr_outlier["Relative Humidity(RH)"])
```

Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x23ceeac53c8>
```



#Divide Dependent&independent variable

In [13]:
```
x=boston_iQr_outlier.iloc[:,0:4].values
x
```

Out[13]:
```
array([[  14.96,   41.76, 1024.07,   73.17],
       [  25.18,   62.96, 1020.04,   59.08],
       [   5.11,   39.4 , 1012.16,   92.14],
       ...,
       [  31.32,   74.33, 1012.92,   36.48],
       [  24.48,   69.45, 1013.86,   62.39],
       [  21.6 ,   62.52, 1017.23,   67.87]])
```

In [14]:
```
y=boston_iQr_outlier.iloc[:,-1].values
y
```

Out[14]:
```
array([463.26, 444.37, 488.56, ..., 429.57, 435.74, 453.28])
```

In [15]:
```
#Normialization using standard scalar
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
```

In [16]:
```
x=sc.fit_transform(x)
x
```

Out[16]:
```
array([[-0.64089601, -0.99629561,  1.91454789, -0.01045211],
       [ 0.73517931,  0.67289863,  1.20806604, -0.97880806],
       [-1.96715256, -1.18211158, -0.17334265,  1.293289  ],
       ...,
       [ 1.56190167,  1.56812214, -0.04011034, -2.53202625],
       [ 0.64092758,  1.18389252,  0.12467699, -0.75132345],
       [ 0.25314902,  0.63825497,  0.7154571 , -0.3747024 ]])
```

In [17]:
```
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=0)
```

In [18]:
```
x_train
```

Out[18]:
```
array([[-0.3958415 , -1.35532985, -0.20665072, -0.30116508],
       [-1.26565036, -1.22069201, -0.18210793,  0.5613521 ],
       [ 0.81865942,  0.86343872, -0.67471687, -1.44614804],
       ...,
       [-1.19024898, -0.77032451,  1.29221276,  0.92697692],
       [-0.26658198, -0.27822715,  0.42269663,  1.13109453],
       [-0.42277057, -1.29155403,  1.81462366, -0.905958  ]])
```

In [19]:
```
from sklearn.linear_model import LinearRegression
```

In [20]:
```
mr=LinearRegression()
```

In [20]:
```
mr=LinearRegression()
```

In [21]:
```
mr.fit(x_train,y_train)
```

Out[21]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [22]:
```
from joblib import dump
dump(mr,"EP.save")
```

Out[22]:
```
['EP.save']
```

In [23]:
```
y_pred = mr.predict(x_test)
y_pred
```

Out[23]:
```
array([473.13673342, 468.91570496, 479.3603297 , ..., 431.62969387,
       470.83519265, 479.09530671])
```

In [24]:
```
y_test
```

Out[24]:
```
array([469.61, 468.74, 477.62, ..., 437.46, 476.44, 481.18])
```

In [25]:
```
mr.predict([[25.18,39.40,1023.67,91.00]])
```

Out[25]:
```
array([167.0093187])
```

In [26]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

Out[26]:
```
0.9248208361581015
```

OUTPUT:


Predict The Electrical
Power Out From CCPP
25.18
39.40
1023.67
91.00
Predict Energy in (MW)
Predict Energy(PE)(MW) 167.0093187002725