In [1]:
```
import pandas as pd
import numpy as np
import re
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from math import sqrt
from urllib.request import urlopen
import json
```

# Combined Cycle Power Plant Dataset

## Dataset Information:

(Extracted from UCI's Machine learning repository)

"The dataset contains 9568 data points collected from a Combined Cycle Power Plant over 6 years (2006-2011), when the power plant was set to work with full load. Features consist of hourly average ambient variables Temperature (T) **[°C]** , Ambient Pressure (AP) **[milibar]** , Relative Humidity (RH) **[%]** and Exhaust Vacuum (V) to predict the net hourly electrical energy output (EP) **[MW]** of the plant."

We will train a model to predict power output given three variables:

- Temperature (T) ...... **[°C]**
- Ambient Pressure (AP) ...... **[milibar]**
- Relative Humidity (RH) ...... **[%]**

Then we will take average data for these three variables from diferent locations in the US and feed them to the model, so we can predict in which locations a power plant would have the highest energy output.

In [2]:
```
data = pd.read_excel('CCPP/Folds5x2_pp.xlsx')
data.head(10)
```

Out[2]:

| | 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 |
| **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 |

In [3]:
```
data = data.drop(['V'], axis = 1)

data.columns = ['Temperature', 'Pressure', 'Humidity', 'Power Out']

data.head()
```

Out[3]:

| | Temperature | Pressure | Humidity | Power Out |
|---|---|---|---|---|
| **0** | 14.96 | 1024.07 | 73.17 | 463.26 |
| **1** | 25.18 | 1020.04 | 59.08 | 444.37 |
| **2** | 5.11 | 1012.16 | 92.14 | 488.56 |
| **3** | 20.86 | 1010.24 | 76.64 | 446.48 |
| **4** | 10.82 | 1009.23 | 96.62 | 473.90 |

In [4]:
```
data.describe()
```

Out[4]:

| | Temperature | Pressure | Humidity | Power Out |
|---|---|---|---|---|
| **count** | 9568.000000 | 9568.000000 | 9568.000000 | 9568.000000 |
| **mean** | 19.651231 | 1013.259078 | 73.308978 | 454.365009 |
| **std** | 7.452473 | 5.938784 | 14.600269 | 17.066995 |
| **min** | 1.810000 | 992.890000 | 25.560000 | 420.260000 |
| **25%** | 13.510000 | 1009.100000 | 63.327500 | 439.750000 |
| **50%** | 20.345000 | 1012.940000 | 74.975000 | 451.550000 |
| **75%** | 25.720000 | 1017.260000 | 84.830000 | 468.430000 |
| **max** | 37.110000 | 1033.300000 | 100.160000 | 495.760000 |

In [5]:
```
data.corr()
```

Out[5]:

| | Temperature | Pressure | Humidity | Power Out |
|---|---|---|---|---|
| **Temperature** | 1.000000 | -0.507549 | -0.542535 | -0.948128 |
| **Pressure** | -0.507549 | 1.000000 | 0.099574 | 0.518429 |
| **Humidity** | -0.542535 | 0.099574 | 1.000000 | 0.389794 |
| **Power Out** | -0.948128 | 0.518429 | 0.389794 | 1.000000 |

# Weather Datasets

We are going to extract information about the average Temperature, Relative Humidity and Ambient Pressure in US from three different datasets with yearly data taken from different stations around US:

- US Weather Stations Temperatures Dataset.
- US Weather Stations Relative Humidity Dataset.
- Station metadata Dataset.

The last one contains infomation about stations all over the world. This include elevation over sea level. With this we can stimate the Ambient Pressure. We'll extract only the data for the stations in the US.

## Temperature Data

In [6]:
```
tempdata = pd.read_excel('Weather/temp.xlsx')
tempdata.head()
```

Out[6]:

| | wban | city | st | YRS | JAN | FEB | MAR | APR | MAY | JUN | JUL | AUG | SEP | OC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 13876 | BIRMINGHAM | AL | 30 | 43.8 | 47.7 | 55.2 | 62.5 | 70.6 | 77.7 | 81.1 | 80.7 | 74.7 | 64. |
| **1** | 3856 | HUNTSVILLE | AL | 30 | 41.5 | 45.7 | 53.5 | 61.8 | 70.3 | 77.7 | 80.6 | 80.1 | 73.7 | 62. |
| **2** | 13894 | MOBILE | AL | 30 | 50.4 | 53.8 | 60.2 | 66.4 | 74.1 | 79.8 | 81.8 | 81.6 | 77.5 | 68. |
| **3** | 13895 | MONTGOMERY | AL | 30 | 46.6 | 50.5 | 57.5 | 64.1 | 72.4 | 79.0 | 81.8 | 81.5 | 76.3 | 65. |
| **4** | 26451 | ANCHORAGE | AK | 30 | 17.1 | 20.2 | 26.6 | 36.8 | 47.8 | 55.2 | 58.8 | 56.7 | 48.6 | 34. |

In [7]:
```
tempdata = tempdata[['wban','city','st', 'ANN']]
tempdata.columns = ['Wban','City','State', 'Temp(°F)']
tempdata.head()
```

Out[7]:

| | Wban | City | State | Temp(°F) |
|---|---|---|---|---|
| **0** | 13876 | BIRMINGHAM | AL | 63.3 |
| **1** | 3856 | HUNTSVILLE | AL | 62.1 |
| **2** | 13894 | MOBILE | AL | 67.2 |
| **3** | 13895 | MONTGOMERY | AL | 65.1 |
| **4** | 26451 | ANCHORAGE | AK | 37.1 |

## Fahrenheit to Celsius conversion

```
T(°C) = (T(°F) - 32) / 1.8
```

In [8]:
```
tempF = np.array(tempdata['Temp(°F)'])
temperature = (tempF - 32)/ 1.8
tempdata['Temperature'] = temperature
tempdata = tempdata.drop('Temp(°F)', axis = 1)
tempdata.head()
```

Out[8]:

| | Wban | City | State | Temperature |
|---|---|---|---|---|
| **0** | 13876 | BIRMINGHAM | AL | 17.388889 |
| **1** | 3856 | HUNTSVILLE | AL | 16.722222 |
| **2** | 13894 | MOBILE | AL | 19.555556 |
| **3** | 13895 | MONTGOMERY | AL | 18.388889 |
| **4** | 26451 | ANCHORAGE | AK | 2.833333 |

In [9]:
```
tempdata.isnull().any()
```

Out[9]:
```
Wban           False
City           False
State          False
Temperature    False
dtype: bool
```

In [10]:
```
tempdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 263 entries, 0 to 262
Data columns (total 4 columns):
Wban           263 non-null int64
City           263 non-null object
State          263 non-null object
Temperature    263 non-null float64
dtypes: float64(1), int64(1), object(2)
memory usage: 8.3+ KB
```

# Humidity Data

In [11]:
```
humdata = pd.read_excel('Weather/relhum.xlsx')
humdata.head()
```

Out[11]:

| | Unnamed: 0 | POR | JAN | Unnamed: 3 | FEB | Unnamed: 5 | MAR | Unnamed: 7 | APR |
|---|---|---|---|---|---|---|---|---|---|
| **0** | NaN | NaN | M | A | M | A | M | A | M |
| **1** | 13876BIRMINGHAM,AL | 194801-201812 | 81 | 60 | 80 | 56 | 80 | 52 | 83 |
| **2** | 03856HUNTSVILLE,AL | 195809-201812 | 81 | 64 | 81 | 60 | 80 | 55 | 82 |
| **3** | 13894MOBILE,AL | 194801-201812 | 83 | 60 | 84 | 58 | 85 | 54 | 88 |
| **4** | 13895MONTGOMERY,AL | 194801-201812 | 83 | 59 | 82 | 56 | 83 | 52 | 87 |

5 rows × 28 columns

In [12]:
```
humdata.columns = ['0', 'POR', 'JAN', 'JAN', 'FEB', 'FEB', 'MAR',
        'MAR', 'APR', 'APR', 'MAY', 'MAY', 'JUN',
        'JUN', 'JUL', 'JUL', 'AUG', 'AUG', 'SEP',
        'SEP', 'OCT', 'OCT', 'NOV', 'NOV', 'DEC',
        'DEC', 'ANN1', 'ANN2']
```

In [13]:
```
humdata.head()
```

Out[13]:

| | 0 | POR | JAN | JAN | FEB | FEB | MAR | MAR | APR | APR | ... | SEP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | NaN | NaN | M | A | M | A | M | A | M | A | ... | M |
| **1** | 13876BIRMINGHAM,AL | 194801-201812 | 81 | 60 | 80 | 56 | 80 | 52 | 83 | 50 | ... | 87 |
| **2** | 03856HUNTSVILLE,AL | 195809-201812 | 81 | 64 | 81 | 60 | 80 | 55 | 82 | 51 | ... | 89 |
| **3** | 13894MOBILE,AL | 194801-201812 | 83 | 60 | 84 | 58 | 85 | 54 | 88 | 53 | ... | 90 |
| **4** | 13895MONTGOMERY,AL | 194801-201812 | 83 | 59 | 82 | 56 | 83 | 52 | 87 | 51 | ... | 90 |

5 rows × 28 columns

In [14]:
```
humdata = humdata[['0' ,'ANN1', 'ANN2']].drop([0])
humdata.head()
```

Out[14]:

| | 0 | ANN1 | ANN2 |
|---|---|---|---|
| 1 | 13876BIRMINGHAM,AL | 84 | 55 |
| 2 | 03856HUNTSVILLE,AL | 85 | 57 |
| 3 | 13894MOBILE,AL | 87 | 57 |
| 4 | 13895MONTGOMERY,AL | 87 | 55 |
| 5 | 26451ANCHORAGE,AK | 74 | 64 |

In [15]:
```
wban = humdata['0']
wban = [re.findall(r'[0-9]+', i)[0] for i in wban]
```

In [16]:
```
humdata['Wban'] = wban
humdata = humdata.drop(['0'], axis = 1)

humdata['Humidity'] = humdata[['ANN1', 'ANN2']].mean(axis=1)
humdata = humdata.drop(['ANN1', 'ANN2'], axis = 1)
humdata.head()
```

Out[16]:

| | Wban | Humidity |
|---|---|---|
| 1 | 13876 | 69.5 |
| 2 | 03856 | 71.0 |
| 3 | 13894 | 72.0 |
| 4 | 13895 | 71.0 |
| 5 | 26451 | 69.0 |

In [17]:
```
humdata['Wban'] = humdata['Wban'].astype('int64')
```

In [18]:
```
humdata.isnull().any()
```

Out[18]:
```
Wban        False
Humidity    False
dtype: bool
```

In [19]:
```
humdata.info()
```

```
&lt;class 'pandas.core.frame.DataFrame'&gt;
Int64Index: 266 entries, 1 to 266
Data columns (total 2 columns):
Wban         266 non-null int64
Humidity     266 non-null float64
dtypes: float64(1), int64(1)
memory usage: 6.2 KB
```

In [20]:
```
print (humdata['Wban'].dtype)
print (tempdata['Wban'].dtype)
```

```
int64
int64
```

## Pressure Data

In [21]:
```
presdata = pd.read_excel('Weather/stations.xltx')
presdata.head()
```

Out[21]:

| | Unnamed: 0 | Unnamed: 1 | Unnamed: 2 | Unnamed: 3 | Unnamed: 4 | Unnamed: 5 | Unnamed: 6 | Unnamed: 7 |
|---|---|---|---|---|---|---|---|---|
| **0** | USAF | WBAN | STATION NAME | CTRY | ST CALL | LAT | LON | ELEV(M) |
| **1** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **2** | 7018 | 99999 | WXPOD 7018 | NaN | NaN | 0 | 0 | 7018 |
| **3** | 7026 | 99999 | WXPOD 7026 | AF | NaN | 0 | 0 | 7026 |
| **4** | 7070 | 99999 | WXPOD 7070 | AF | NaN | 0 | 0 | 7070 |

In [22]:
```
columns = presdata.iloc[0]
presdata.columns = columns

presdata = presdata.drop([0])
presdata.head()
```

Out[22]:

| | USAF | WBAN | STATION NAME | CTRY | ST CALL | LAT | LON | ELEV(M) | BEGIN | END |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **2** | 7018 | 99999 | WXPOD 7018 | NaN | NaN | 0 | 0 | 7018 | 20110309 | 20130730 |
| **3** | 7026 | 99999 | WXPOD 7026 | AF | NaN | 0 | 0 | 7026 | 20120713 | 20170822 |
| **4** | 7070 | 99999 | WXPOD 7070 | AF | NaN | 0 | 0 | 7070 | 20140923 | 20150926 |
| **5** | 8260 | 99999 | WXPOD8270 | NaN | NaN | 0 | 0 | 0 | 20050101 | 20100920 |

In [23]:
```
presdata = presdata[presdata['CTRY']==('US')]
presdata.head()
```

Out[23]:

| | USAF | WBAN | STATION NAME | CTRY | ST CALL | LAT | LON | ELEV(M) | BEGIN | END |
|---|---|---|---|---|---|---|---|---|---|---|
| **13135** | 621010 | 99999 | MOORED BUOY | US | NaN | 50.6 | -2.933 | -999 | 20080721 | 20080721 |
| **13137** | 621110 | 99999 | MOORED BUOY | US | NaN | 58.9 | -0.2 | -999 | 20041118 | 20041118 |
| **13138** | 621130 | 99999 | MOORED BUOY | US | NaN | 58.4 | 0.3 | -999 | 20040726 | 20040726 |
| **13139** | 621160 | 99999 | MOORED BUOY | US | NaN | 58.1 | 1.8 | -999 | 20040829 | 20040829 |
| **13140** | 621170 | 99999 | MOORED BUOY | US | NaN | 57.9 | 0.1 | -999 | 20040726 | 20040726 |

In [24]:
```
presdata = presdata[['WBAN','ELEV(M)']]


presdata['ELEV(M)'] = pd.to_numeric(presdata['ELEV(M)'], errors='coerc
e')

presdata['WBAN'] = presdata['WBAN'].astype('int64')

presdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7326 entries, 13135 to 29745
Data columns (total 2 columns):
WBAN       7326 non-null int64
ELEV(M)    7209 non-null float64
dtypes: float64(1), int64(1)
memory usage: 171.7 KB
```

In [25]:
```
presdata.dropna(inplace=True)

presdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7209 entries, 13135 to 29745
Data columns (total 2 columns):
WBAN       7209 non-null int64
ELEV(M)    7209 non-null float64
dtypes: float64(1), int64(1)
memory usage: 169.0 KB
```

In [26]:
```
presdata.columns = ['Wban', 'Elevation']
presdata.head()
```

Out[26]:

| | Wban | Elevation |
|---|---|---|
| **13135** | 99999 | -999.0 |
| **13137** | 99999 | -999.0 |
| **13138** | 99999 | -999.0 |
| **13139** | 99999 | -999.0 |
| **13140** | 99999 | -999.0 |

In [27]:
```
presdata = presdata[presdata['Elevation']>=0]
presdata.head()
```

Out[27]:

| | Wban | Elevation |
|---|---|---|
| **13149** | 99999 | 0.0 |
| **14465** | 99999 | 1224.0 |
| **14466** | 93218 | 317.0 |
| **14467** | 99999 | 317.0 |
| **14468** | 93217 | 43.0 |

## Average ambient pressure calculation

For this calculation we will use the following expresion:

```
p ~ 101325 (1 - 2.25577^10-5 * h)^5
```

Where:

```
101325 = normal temperature and pressure at sea level (Pa)

p = air pressure (Pa)

h = altitude above sea level (m)
```

Pascal to milibar convertion:

```
1 Pa = 0.01 milibar
```

In [28]:
```
elev = np.array(presdata['Elevation'])


Pasc = 101325*(1 - 2.25577*10**(-5) * elev)**5


milibar = Pasc * 0.01
```

In [29]:
```
presdata['Pressure'] = milibar
presdata = presdata.drop('Elevation', axis = 1)

presdata.head()
```

Out[29]:

| | Wban | Pressure |
|---|---|---|
| **13149** | 99999 | 1013.250000 |
| **14465** | 99999 | 880.881800 |
| **14466** | 93218 | 977.536727 |
| **14467** | 99999 | 977.536727 |
| **14468** | 93217 | 1008.345357 |

In [30]:
```
presdata['Wban'].value_counts().head()
```

Out[30]:
```
99999    3548
23176       6
3849        5
14897       4
24255       4
Name: Wban, dtype: int64
```

In [31]:
```
presdata[presdata['Wban']==23176]
```

Out[31]:

| | Wban | Pressure |
|---|---|---|
| **19714** | 23176 | 849.441660 |
| **19722** | 23176 | 849.650098 |
| **19743** | 23176 | 849.650098 |
| **21748** | 23176 | 849.650098 |
| **21749** | 23176 | 849.650098 |
| **29011** | 23176 | 849.650098 |

In [32]:
```
presdata['Wban'] = presdata[presdata['Wban']!=99999]
presdata = presdata.drop_duplicates(subset = 'Wban')
presdata['Wban'].value_counts().head()
```

Out[32]:
```
54791.0    1
24164.0    1
24048.0    1
4898.0     1
3930.0     1
Name: Wban, dtype: int64
```

## Merging weather datasets into US Locations

In [33]:
```
USloc = tempdata.merge(humdata, how='inner')
print (len(USloc))
USloc.head(10)
```

```
260
```

Out[33]:

| | Wban | City | State | Temperature | Humidity |
|---|---|---|---|---|---|
| **0** | 13876 | BIRMINGHAM | AL | 17.388889 | 69.5 |
| **1** | 3856 | HUNTSVILLE | AL | 16.722222 | 71.0 |
| **2** | 13894 | MOBILE | AL | 19.555556 | 72.0 |
| **3** | 13895 | MONTGOMERY | AL | 18.388889 | 71.0 |
| **4** | 26451 | ANCHORAGE | AK | 2.833333 | 69.0 |
| **5** | 25308 | ANNETTE | AK | 8.111111 | 77.5 |
| **6** | 27502 | BARROW | AK | -11.222222 | 81.0 |
| **7** | 26615 | BETHEL | AK | -0.722222 | 78.5 |
| **8** | 26533 | BETTLES | AK | -4.722222 | 67.0 |
| **9** | 26415 | BIG DELTA | AK | -1.666667 | 62.5 |

In [34]:
```
usloc = USloc.merge(presdata, how='left')

print (len(usloc))
usloc.head(10)
```

```
260
```

Out[34]:

| | Wban | City | State | Temperature | Humidity | Pressure |
|---|---|---|---|---|---|---|
| **0** | 13876 | BIRMINGHAM | AL | 17.388889 | 69.5 | 992.002445 |
| **1** | 3856 | HUNTSVILLE | AL | 16.722222 | 71.0 | 991.699106 |
| **2** | 13894 | MOBILE | AL | 19.555556 | 72.0 | 1005.786554 |
| **3** | 13895 | MONTGOMERY | AL | 18.388889 | 71.0 | 1006.229708 |
| **4** | 26451 | ANCHORAGE | AK | 2.833333 | 69.0 | 1009.074145 |
| **5** | 25308 | ANNETTE | AK | 8.111111 | 77.5 | 1009.461485 |
| **6** | 27502 | BARROW | AK | -11.222222 | 81.0 | 1012.164777 |
| **7** | 26615 | BETHEL | AK | -0.722222 | 78.5 | 1009.700784 |
| **8** | 26533 | BETTLES | AK | -4.722222 | 67.0 | 991.362149 |
| **9** | 26415 | BIG DELTA | AK | -1.666667 | 62.5 | 969.545252 |

In [35]:
```
usloc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 260 entries, 0 to 259
Data columns (total 6 columns):
Wban           260 non-null int64
City           260 non-null object
State          260 non-null object
Temperature    260 non-null float64
Humidity       260 non-null float64
Pressure       251 non-null float64
dtypes: float64(3), int64(1), object(2)
memory usage: 14.2+ KB
```

In [36]:
```
usloc = usloc.dropna()

usloc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 251 entries, 0 to 250
Data columns (total 6 columns):
Wban           251 non-null int64
City           251 non-null object
State          251 non-null object
Temperature    251 non-null float64
Humidity       251 non-null float64
Pressure       251 non-null float64
dtypes: float64(3), int64(1), object(2)
memory usage: 13.7+ KB
```

# Multiple Linear Regression Model

We are going to create a model to preddict energy output (EP) [MW], taking Temperature (T) [°C] , Ambient Pressure (AP) [milibar] , Relative Humidity (RH) [%] as parameters.

We will use the dataset called 'data', which contains the values for the CCPP.

We will use scikit-learn, and train the model as a multiple linear regrssion

In [37]:
```
X = data[['Temperature', 'Pressure', 'Humidity']]
Y = data[['Power Out']]
```

In [38]:
```
Xtrain, Xtest, Ytrain, Ytest =  train_test_split(X,Y, test_size=0.25,
random_state=10)
```

In [39]:
```
regression = LinearRegression()
model = regression.fit(Xtrain, Ytrain)
```

In [40]:
```
Ypred = model.predict(Xtest)
Ypred
```

Out[40]:
```
array([[455.39353049],
       [447.47658286],
       [437.21191769],
       ...,
       [487.93736352],
       [440.41536484],
       [453.40714577]])
```

In [41]:
```
RMSE = sqrt(mean_squared_error(y_true = Ytest, y_pred = Ypred))
RMSE
```

Out[41]: 4.657571391074255

In [42]:
```
r2 = r2_score(Ytest, Ypred)
r2
```

Out[42]: 0.9266765701220129

# Performance prediction at US locations

In this step we will use the model previously trained with the CCPP data, to predict the expected power output of a CCPP situated in differents US locations based on their climate average conditions.

In [43]:
```
usweather = usloc[['Temperature', 'Pressure', 'Humidity']]

usweather.head()
```

Out[43]:

| | Temperature | Pressure | Humidity |
|---|---|---|---|
| **0** | 17.388889 | 992.002445 | 69.5 |
| **1** | 16.722222 | 991.699106 | 71.0 |
| **2** | 19.555556 | 1005.786554 | 72.0 |
| **3** | 18.388889 | 1006.229708 | 71.0 |
| **4** | 2.833333 | 1009.074145 | 69.0 |

In [44]:
```
X.head()
```

Out[44]:

| | Temperature | Pressure | Humidity |
|---|---|---|---|
| **0** | 14.96 | 1024.07 | 73.17 |
| **1** | 25.18 | 1020.04 | 59.08 |
| **2** | 5.11 | 1012.16 | 92.14 |
| **3** | 20.86 | 1010.24 | 76.64 |
| **4** | 10.82 | 1009.23 | 96.62 |

In [45]:
```
USpred = model.predict(usweather)
USpred[:5]
```

Out[45]:
```
array([[459.91897085],
       [461.1913595 ],
       [454.64134399],
       [457.62249043],
       [495.01181029]])
```

In [46]:
```
usloc['Power_out'] = USpred

usloc.head()
```

Out[46]:

| | Wban | City | State | Temperature | Humidity | Pressure | Power_out |
|---|---|---|---|---|---|---|---|
| **0** | 13876 | BIRMINGHAM | AL | 17.388889 | 69.5 | 992.002445 | 459.918971 |
| **1** | 3856 | HUNTSVILLE | AL | 16.722222 | 71.0 | 991.699106 | 461.191360 |
| **2** | 13894 | MOBILE | AL | 19.555556 | 72.0 | 1005.786554 | 454.641344 |
| **3** | 13895 | MONTGOMERY | AL | 18.388889 | 71.0 | 1006.229708 | 457.622490 |
| **4** | 26451 | ANCHORAGE | AK | 2.833333 | 69.0 | 1009.074145 | 495.011810 |

# Findings

In [47]:
```
usloc.Power_out.describe()
```

Out[47]:
```
count    251.000000
mean     472.345828
std       14.272938
min      440.563445
25%      462.034303
50%      473.768200
75%      480.319051
max      526.034436
Name: Power_out, dtype: float64
```

In [48]:
```
usloc.drop('Wban',axis=1).corr()
```

Out[48]:

| | Temperature | Humidity | Pressure | Power_out |
|---|---|---|---|---|
| **Temperature** | 1.000000 | -0.057025 | 0.236537 | -0.991844 |
| **Humidity** | -0.057025 | 1.000000 | 0.586223 | -0.048887 |
| **Pressure** | 0.236537 | 0.586223 | 1.000000 | -0.243565 |
| **Power_out** | -0.991844 | -0.048887 | -0.243565 | 1.000000 |

In [49]:
```
usloc.sort_values('Power_out', ascending=False).head(10)
```

Out[49]:

| | Wban | City | State | Temperature | Humidity | Pressure | Power_out |
|---|---|---|---|---|---|---|---|
| **6** | 27502 | BARROW | AK | -11.222222 | 81.0 | 1012.164777 | 526.034436 |
| **147** | 14755 | MT. WASHINGTON | NH | -2.555556 | 0.0 | 812.822902 | 516.454667 |
| **8** | 26533 | BETTLES | AK | -4.722222 | 67.0 | 991.362149 | 512.871904 |
| **17** | 26616 | KOTZEBUE | AK | -5.055556 | 78.0 | 1012.210452 | 512.005739 |
| **11** | 26411 | FAIRBANKS | AK | -2.388889 | 65.0 | 998.288099 | 507.921088 |
| **18** | 26510 | MCGRATH | AK | -2.555556 | 68.5 | 1001.703277 | 507.703816 |
| **19** | 26617 | NOME | AK | -2.555556 | 76.0 | 1012.792951 | 506.490944 |
| **12** | 26425 | GULKANA | AK | -2.111111 | 66.0 | 959.996103 | 506.043838 |
| **9** | 26415 | BIG DELTA | AK | -1.666667 | 62.5 | 969.545252 | 505.946333 |
| **7** | 26615 | BETHEL | AK | -0.722222 | 78.5 | 1009.700784 | 501.556013 |

In [50]:
```
usloc.sort_values('Power_out', ascending=False).tail(10)
```

Out[50]:

| | Wban | City | State | Temperature | Humidity | Pressure | Power_out |
|---|---|---|---|---|---|---|---|
| **61** | 12842 | TAMPA | FL | 23.000000 | 72.0 | 1012.587332 | 446.648664 |
| **70** | 21504 | HILO | HI | 23.277778 | 74.0 | 1011.925011 | 445.569949 |
| **211** | 12919 | BROWNSVILLE | TX | 23.611111 | 75.0 | 1012.416009 | 444.591033 |
| **53** | 12835 | FORT MYERS | FL | 23.944444 | 71.5 | 1012.724408 | 444.511729 |
| **72** | 22516 | KAHULUI | HI | 24.388889 | 67.5 | 1011.479853 | 444.228015 |
| **63** | 12844 | WEST PALM BEACH | FL | 24.111111 | 71.5 | 1012.587332 | 444.112604 |
| **73** | 22536 | LIHUE | HI | 24.333333 | 72.0 | 1009.769163 | 443.409941 |
| **71** | 22521 | HONOLULU | HI | 25.388889 | 65.0 | 1013.010029 | 442.398239 |
| **57** | 12839 | MIAMI | FL | 25.111111 | 71.5 | 1012.244709 | 441.730603 |
| **56** | 12836 | KEY WEST | FL | 25.444444 | 73.5 | 1013.215716 | 440.563445 |

In [51]:
```
statemean = usloc.groupby('State', as_index=False).mean()
statemean = statemean.sort_values('Power_out', ascending=False)
statemean.head()
```

Out[51]:

| | State | Wban | Temperature | Humidity | Pressure | Power_out |
|---|---|---|---|---|---|---|
| **30** | NH | 14750.00 | 2.722222 | 33.250 | 907.087592 | 499.751931 |
| **0** | AK | 26143.45 | 0.519444 | 74.325 | 1003.855505 | 499.293539 |
| **28** | ND | 36963.75 | 5.416667 | 70.875 | 968.073548 | 487.415893 |
| **21** | ME | 14685.50 | 6.222222 | 69.250 | 1001.692198 | 486.723986 |
| **23** | MN | 14920.80 | 5.755556 | 71.900 | 973.431288 | 486.548040 |

# Plotting color US map

In [52]:
```
city = pd.read_excel('uscities.xlsx')
city.head(10)
```

Out[52]:

| | city | city_ascii | state_id | state_name | county_fips | county_name | county_fips_all | co |
|---|---|---|---|---|---|---|---|---|
| **0** | South Creek | South Creek | WA | Washington | 53053 | Pierce | 53053 | |
| **1** | Roslyn | Roslyn | WA | Washington | 53037 | Kittitas | 53037 | |
| **2** | Sprague | Sprague | WA | Washington | 53043 | Lincoln | 53043 | |
| **3** | Gig Harbor | Gig Harbor | WA | Washington | 53053 | Pierce | 53053 | |
| **4** | Lake Cassidy | Lake Cassidy | WA | Washington | 53061 | Snohomish | 53061 | |
| **5** | Tenino | Tenino | WA | Washington | 53067 | Thurston | 53067 | |
| **6** | Jamestown | Jamestown | WA | Washington | 53009 | Clallam | 53009 | |
| **7** | Three Lakes | Three Lakes | WA | Washington | 53061 | Snohomish | 53061 | |
| **8** | Curlew Lake | Curlew Lake | WA | Washington | 53019 | Ferry | 53019 | |
| **9** | Chain Lake | Chain Lake | WA | Washington | 53061 | Snohomish | 53061 | |

In [53]:
```
city = city[['city','state_id', 'county_fips']]
city.head()
```

Out[53]:

| | city | state_id | county_fips |
|---|---|---|---|
| 0 | South Creek | WA | 53053 |
| 1 | Roslyn | WA | 53037 |
| 2 | Sprague | WA | 53043 |
| 3 | Gig Harbor | WA | 53053 |
| 4 | Lake Cassidy | WA | 53061 |

In [54]:
```
city.city = city.city.str.upper()
city.head()
```

Out[54]:

| | city | state_id | county_fips |
|---|---|---|---|
| 0 | SOUTH CREEK | WA | 53053 |
| 1 | ROSLYN | WA | 53037 |
| 2 | SPRAGUE | WA | 53043 |
| 3 | GIG HARBOR | WA | 53053 |
| 4 | LAKE CASSIDY | WA | 53061 |

In [55]:
```
DF = usloc[['City','State','Power_out']]
DF.columns = ['city','state_id','Power_out']
DF.city = DF.city.str.rstrip()
DF.head()
```

```
/Users/sandra/anaconda3/lib/python3.7/site-packages/pandas/core/gene
ric.py:5096: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
  self[name] = value
```

Out[55]:

| | city | state_id | Power_out |
|---|---|---|---|
| **0** | BIRMINGHAM | AL | 459.918971 |
| **1** | HUNTSVILLE | AL | 461.191360 |
| **2** | MOBILE | AL | 454.641344 |
| **3** | MONTGOMERY | AL | 457.622490 |
| **4** | ANCHORAGE | AK | 495.011810 |

In [56]:
```
city.head()
```

Out[56]:

| | city | state_id | county_fips |
|---|---|---|---|
| **0** | SOUTH CREEK | WA | 53053 |
| **1** | ROSLYN | WA | 53037 |
| **2** | SPRAGUE | WA | 53043 |
| **3** | GIG HARBOR | WA | 53053 |
| **4** | LAKE CASSIDY | WA | 53061 |

In [57]:
```
city.city[0]
```

Out[57]:
```
'SOUTH CREEK'
```

n=0 for i in city.city: if 'Huntsville' in i: print (city.iloc[n]) n =n+1

In [58]:
```
x = pd.merge(DF, city, on=['city','state_id'])
x.head()
```

Out[58]:

| | city | state_id | Power_out | county_fips |
|---|---|---|---|---|
| 0 | BIRMINGHAM | AL | 459.918971 | 1073 |
| 1 | HUNTSVILLE | AL | 461.191360 | 1089 |
| 2 | MOBILE | AL | 454.641344 | 1097 |
| 3 | MONTGOMERY | AL | 457.622490 | 1101 |
| 4 | ANCHORAGE | AK | 495.011810 | 2020 |

In [59]:
```
x.county_fips[x.county_fips<10000]= ('0'+ x.county_fips.astype(str))
```

```
/Users/sandra/anaconda3/lib/python3.7/site-packages/ipykernel_launch
er.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
```

In [60]:
```
len (x)
```

Out[60]: 232

In [61]:
```
x.head()
```

Out[61]:

| | city | state_id | Power_out | county_fips |
|---|---|---|---|---|
| 0 | BIRMINGHAM | AL | 459.918971 | 01073 |
| 1 | HUNTSVILLE | AL | 461.191360 | 01089 |
| 2 | MOBILE | AL | 454.641344 | 01097 |
| 3 | MONTGOMERY | AL | 457.622490 | 01101 |
| 4 | ANCHORAGE | AK | 495.011810 | 02020 |

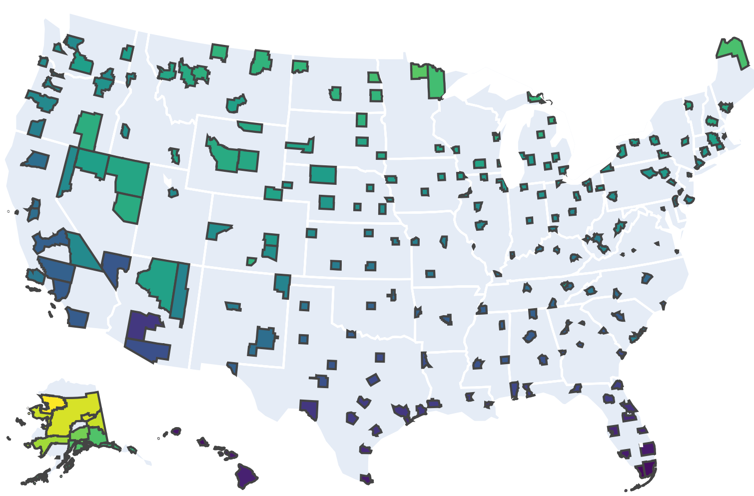
In [62]:

```
with urlopen('https://raw.githubusercontent.com/plotly/datasets/master
/geojson-counties-fips.json') as response:
    counties = json.load(response)




import plotly.express as px

fig = px.choropleth(x, geojson=counties, locations='county_fips', colo
r='Power_out',
                           color_continuous_scale="Viridis",
                           range_color=(439, 512),
                           scope="usa", labels={'Power_out':'Power Out
put [MW]'},
                          )
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```



# Combined Cycle Power Plant Performance

Sandra Poza

# Abstract

This project studies the relationship between weather conditions and performance of a combined cycle power plant (CCPP). We used a dataset containing CCPP performance to train a regression model that predict the power output of the plant.

Then with data from US weather datasets we predicted the performance it would have in different US locations.

It was found that it would perform better in coldest weather in northern areas.

# Motivation

Energy efficiency has become an essential matter in todays society. For climate, economic and sustainability reasons, every one gets benefit from a more effective use of energy.

In this project we try to find out where would a Combined Cycle Power Plant (CCPP) have the highest power output (which would mean a highest efficiency), based on the weather conditions.

# Dataset(s)

**Combined Cycle Power Plant Data Set:** The dataset contains 9568 data points collected from a Combined Cycle Power Plant over 6 years (2006-2011). Obtained from UCI's Machine learning repository.

**US Weather datasets:** Obtained from NOAA National Centers for Environmental Information

- **Temperatures dataset:** Contains monthly and yearly average temperatures for different US weather stations.
- **Relative Humidity dataset:** Contains monthly and yearly average relative humidity for different US weather stations.
- **Weather station locations dataset:** Contains weather stations information with elevation, used for ambient pressure calculation.

**US cities dataset:** Contains US cities information, including the fips codes. Obtained from https://simplemaps.com

# Data Preparation and Cleaning

- **Unit conversion**: Dataframes with different units for temperature and pressure

- **Dataframe formatting**: Weather datasets were originally in text format and some adjustments were needed when imported into pandas.

- **String extract**: in the humidity dataset, we had to extract the wban number that was in the same string as the city name.

- **String strip:** city column contained blank spaces at the right side.

- **Type**: data type changes.

- **Pressure calculation**: using altitude, the ambient pressure was calculated.

- **Merging dataframes**: temperature , Ambient Pressure, Relative Humidity dataframes into US locations and cities with fips numbers.

# Research Question(s)

This project aims to define the relationship between weather ambient conditions and performance of a **Combined Cycle Power Plant (CCPP).**

With this we would like to explore in which US locations a Combined Cycle Power Plant would have the better performance, based on the average weather conditions of these locations.

# Methods

The Combined Cycle Power Plant Dataset contains data of a CCPP working at full load over 6 years. This include the following features:

Temperature (T) [°C]
Ambient Pressure (AP) [milibar]
Relative Humidity (RH) [%]
**Energy output (EP) [MW] of the plant**

With this data we will train a model that predict **Energy output** based on the 3 first. We will use scikit-learn, and train the model as a multiple linear regression.

Then, from the weather datasets we are going to extract information about the average Temperature, Relative Humidity and Ambient Pressure in different US locations.

Then we will use the model previously trained with the CCPP data, and feed the weather average data from different locations to predict the expected power output of a CCPP in this locations.

We will use Matplotlib and Plotly for plotting results.

# Findings

**Multiple Linear Regression Model Accuracy**

The mean squared error for the model with test data is 4.65. Taking into consideration that the output of the model is in the order of 400-500 it is a good number.

```
RMSE = sqrt(mean_squared_error(y_true = Ytest, y_pred = Ypred))
RMSE
```

```
4.657571391074255
```

A better stimator is the R2 score. This is a statistical measure of how well the regression predictions approximate the real data points. It ranges from 0-1 being 1 the best possible result.
Our R2 score is 0.92, what means that **the model is highly accurate** on test data.

```
from sklearn.metrics import r2_score

r2 = r2_score(Ytest, Ypred)
r2
```

```
0.9266765701220129
```

# Findings

## Performance prediction at US locations

We can appreciate how the energy output is higher in the coldest areas like Alaska, and  lower in the warmer ones like Florida.

| | Wban | City | State | Temperature | Humidity | Pressure | Power_out |
|---|---|---|---|---|---|---|---|
| **6** | 27502 | BARROW | AK | -11.222222 | 81.0 | 1012.164777 | 526.034436 |
| **147** | 14755 | MT. WASHINGTON | NH | -2.555556 | 0.0 | 812.822902 | 516.454667 |
| **8** | 26533 | BETTLES | AK | -4.722222 | 67.0 | 991.362149 | 512.871904 |
| **17** | 26616 | KOTZEBUE | AK | -5.055556 | 78.0 | 1012.210452 | 512.005739 |
| **11** | 26411 | FAIRBANKS | AK | -2.388889 | 65.0 | 998.288099 | 507.921088 |
| **18** | 26510 | MCGRATH | AK | -2.555556 | 68.5 | 1001.703277 | 507.703816 |
| **19** | 26617 | NOME | AK | -2.555556 | 76.0 | 1012.792951 | 506.490944 |
| **12** | 26425 | GULKANA | AK | -2.111111 | 66.0 | 959.996103 | 506.043838 |
| **9** | 26415 | BIG DELTA | AK | -1.666667 | 62.5 | 969.545252 | 505.946333 |
| **7** | 26615 | BETHEL | AK | -0.722222 | 78.5 | 1009.700784 | 501.556013 |

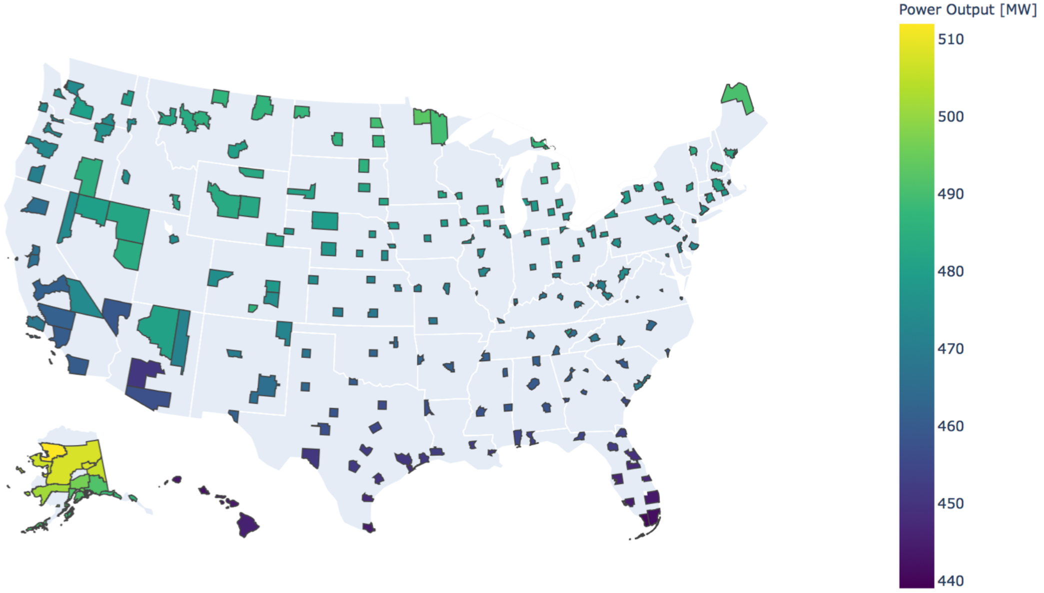
10 Highest Energy Output

| | Wban | City | State | Temperature | Humidity | Pressure | Power_out |
|---|---|---|---|---|---|---|---|
| **61** | 12842 | TAMPA | FL | 23.000000 | 72.0 | 1012.587332 | 446.648664 |
| **70** | 21504 | HILO | HI | 23.277778 | 74.0 | 1011.925011 | 445.569949 |
| **211** | 12919 | BROWNSVILLE | TX | 23.611111 | 75.0 | 1012.416009 | 444.591033 |
| **53** | 12835 | FORT MYERS | FL | 23.944444 | 71.5 | 1012.724408 | 444.511729 |
| **72** | 22516 | KAHULUI | HI | 24.388889 | 67.5 | 1011.479853 | 444.228015 |
| **63** | 12844 | WEST PALM BEACH | FL | 24.111111 | 71.5 | 1012.587332 | 444.112604 |
| **73** | 22536 | LIHUE | HI | 24.333333 | 72.0 | 1009.769163 | 443.409941 |
| **71** | 22521 | HONOLULU | HI | 25.388889 | 65.0 | 1013.010029 | 442.398239 |
| **57** | 12839 | MIAMI | FL | 25.111111 | 71.5 | 1012.244709 | 441.730603 |
| **56** | 12836 | KEY WEST | FL | 25.444444 | 73.5 | 1013.215716 | 440.563445 |

10 Lowest Energy Output

# Findings

## Performance prediction at US locations




This map shows the power output that it would be obtained by a CCPP situated in different US counties.

We can observe the relationship between southern, warmer, and more humid areas with lower power output

Also the northern, colder and drier areas whit highest power output.

It should be mentioned that Alaska is represented on the bottom left, but is in fact the northernmost state, with the lowest temperatures, so we can appreciate how that region is the most yellowish.

# Limitations

We did not have enough weather data to study the whole US map.

The weather data is average, so it doesn’t reflect seasonal changes.

The CCPP dataset measures was taken in a fixed location, so the pressure changes were not well represented.

# Conclusions

We can conclude, that the performance of a Combined Cycle Power Plant would be the best in the coldest  and driest possible weathers.

A CCPP situated in Alaska would have the highest efficiency, and one located in Florida the lowest. That means that the same power plant would consume less gas in Alaska to produce the same power output.

# Acknowledgements

**Combined Cycle Power Plant Data Set:** Obtained from UCI's Machine learning repository.

**US Weather datasets:** Obtained from NOAA National Centers for Environmental Information

**US cities dataset:** Obtained from https://simplemaps.com

I had no one give me feedback yet, but I would highly appreciate yours.

# References

https://plot.ly/python/choropleth-maps/

https://pandas.pydata.org

In [1]:

```
import pandas as pd
```

# Combined Cycle Power Plant Dataset

## Dataset Information:

(Extracted from UCI's Machine learning repository)

"The dataset contains 9568 data points collected from a Combined Cycle Power Plant over 6 years (2006-2011), when the power plant was set to work with full load. Features consist of hourly average ambient variables Temperature (T) **[°C]** , Ambient Pressure (AP) **[milibar]** , Relative Humidity (RH) **[%]** and Exhaust Vacuum (V) to predict the net hourly electrical energy output (EP) **[MW]** of the plant."

We will train a model to predict power output given three variables:

- Temperature (T) ...... **[°C]**
- Ambient Pressure (AP) ...... **[milibar]**
- Relative Humidity (RH) ...... **[%]**

Then we will take average data for these three variables from diferent locations in the US and feed them to the model, so we can predict in which locations a power plant would have the highest energy output.

In [ ]:

```

```

In [2]:

```
data = pd.read_excel('CCPP/Folds5x2_pp.xlsx')
data.head(10)
```

Out[2]:

| | 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 |
| **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 |

In [3]:

```
len(data)
```

Out[3]:

```
9568
```

In [4]:

```
data.columns
```

Out[4]:

```
Index(['AT', 'V', 'AP', 'RH', 'PE'], dtype='object')
```

In [5]:

```
data = data.drop(['V'], axis = 1)
```

In [6]:

```
data.columns = ['Temperature', 'Pressure', 'Humidity', 'Power Out']
```

In [7]:

```
data.head()
```

Out[7]:

| | Temperature | Pressure | Humidity | Power Out |
|---|---|---|---|---|
| **0** | 14.96 | 1024.07 | 73.17 | 463.26 |
| **1** | 25.18 | 1020.04 | 59.08 | 444.37 |
| **2** | 5.11 | 1012.16 | 92.14 | 488.56 |
| **3** | 20.86 | 1010.24 | 76.64 | 446.48 |
| **4** | 10.82 | 1009.23 | 96.62 | 473.90 |

In [8]:

```
data.describe()
```

Out[8]:

| | Temperature | Pressure | Humidity | Power Out |
|---|---|---|---|---|
| **count** | 9568.000000 | 9568.000000 | 9568.000000 | 9568.000000 |
| **mean** | 19.651231 | 1013.259078 | 73.308978 | 454.365009 |
| **std** | 7.452473 | 5.938784 | 14.600269 | 17.066995 |
| **min** | 1.810000 | 992.890000 | 25.560000 | 420.260000 |
| **25%** | 13.510000 | 1009.100000 | 63.327500 | 439.750000 |
| **50%** | 20.345000 | 1012.940000 | 74.975000 | 451.550000 |
| **75%** | 25.720000 | 1017.260000 | 84.830000 | 468.430000 |
| **max** | 37.110000 | 1033.300000 | 100.160000 | 495.760000 |

In [9]:

```
data.corr()
```

Out[9]:

| | Temperature | Pressure | Humidity | Power Out |
|---|---|---|---|---|
| **Temperature** | 1.000000 | -0.507549 | -0.542535 | -0.948128 |
| **Pressure** | -0.507549 | 1.000000 | 0.099574 | 0.518429 |
| **Humidity** | -0.542535 | 0.099574 | 1.000000 | 0.389794 |
| **Power Out** | -0.948128 | 0.518429 | 0.389794 | 1.000000 |

In [ ]:

```

```

# Weather Datasets

We are going to extract information about the average Temperature, Relative Humidity and Ambient Pressure in US from three different datasets with yearly data taken from different stations around US:

- US Weather Stations Temperatures Dataset.
- US Weather Stations Relative Humidity Dataset.
- Station metadata Dataset.

The last one contains infomation about stations all over the world. This include elevation over sea level. With this we can stimate the Ambient Pressure. We'll extract only the data for the stations in the US.

## Temperature Data

In [10]:

```
tempdata = pd.read_excel('Weather/temp.xlsx')
tempdata.head()
```

Out[10]:

| | wban | city | st | YRS | JAN | FEB | MAR | APR | MAY | JUN | JUL | AUG | SEP | OCT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 13876 | BIRMINGHAM | AL | 30 | 43.8 | 47.7 | 55.2 | 62.5 | 70.6 | 77.7 | 81.1 | 80.7 | 74.7 | 64.1 |
| **1** | 3856 | HUNTSVILLE | AL | 30 | 41.5 | 45.7 | 53.5 | 61.8 | 70.3 | 77.7 | 80.6 | 80.1 | 73.7 | 62.8 |
| **2** | 13894 | MOBILE | AL | 30 | 50.4 | 53.8 | 60.2 | 66.4 | 74.1 | 79.8 | 81.8 | 81.6 | 77.5 | 68.4 |
| **3** | 13895 | MONTGOMERY | AL | 30 | 46.6 | 50.5 | 57.5 | 64.1 | 72.4 | 79.0 | 81.8 | 81.5 | 76.3 | 65.9 |
| **4** | 26451 | ANCHORAGE | AK | 30 | 17.1 | 20.2 | 26.6 | 36.8 | 47.8 | 55.2 | 58.8 | 56.7 | 48.6 | 34.8 |

In [11]:

```
tempdata = tempdata[['wban','city','st', 'ANN']]
tempdata.columns = ['Wban','City','State', 'Temp(°F)']
tempdata.head()
```

Out[11]:

| | Wban | City | State | Temp(°F) |
|---|---|---|---|---|
| **0** | 13876 | BIRMINGHAM | AL | 63.3 |
| **1** | 3856 | HUNTSVILLE | AL | 62.1 |
| **2** | 13894 | MOBILE | AL | 67.2 |
| **3** | 13895 | MONTGOMERY | AL | 65.1 |
| **4** | 26451 | ANCHORAGE | AK | 37.1 |

## Fahrenheit to Celsius conversion

```
T(°C) = (T(°F) - 32) / 1.8
```

In [12]:

```
import numpy as np
tempF = np.array(tempdata['Temp(°F)'])
temperature = (tempF - 32)/ 1.8
tempdata['Temperature'] = temperature
tempdata = tempdata.drop('Temp(°F)', axis = 1)
tempdata.head()
```

Out[12]:

| | Wban | City | State | Temperature |
|---|---|---|---|---|
| **0** | 13876 | BIRMINGHAM | AL | 17.388889 |
| **1** | 3856 | HUNTSVILLE | AL | 16.722222 |
| **2** | 13894 | MOBILE | AL | 19.555556 |
| **3** | 13895 | MONTGOMERY | AL | 18.388889 |
| **4** | 26451 | ANCHORAGE | AK | 2.833333 |

In [13]:

```
tempdata.isnull().any()
```

Out[13]:

```
Wban           False
City           False
State          False
Temperature    False
dtype: bool
```

In [14]:

```
tempdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 263 entries, 0 to 262
Data columns (total 4 columns):
Wban           263 non-null int64
City           263 non-null object
State          263 non-null object
Temperature    263 non-null float64
dtypes: float64(1), int64(1), object(2)
memory usage: 8.3+ KB
```

## Humidity Data

In [15]:

```
humdata = pd.read_excel('Weather/relhum.xlsx')
humdata.head()
```

Out[15]:

| | Unnamed: 0 | POR | JAN | Unnamed: 3 | FEB | Unnamed: 5 | MAR | Unnamed: 7 | APR | Ur |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | NaN | NaN | M | A | M | A | M | A | M | |
| **1** | 13876BIRMINGHAM,AL | 194801-201812 | 81 | 60 | 80 | 56 | 80 | 52 | 83 | |
| **2** | 03856HUNTSVILLE,AL | 195809-201812 | 81 | 64 | 81 | 60 | 80 | 55 | 82 | |
| **3** | 13894MOBILE,AL | 194801-201812 | 83 | 60 | 84 | 58 | 85 | 54 | 88 | |
| **4** | 13895MONTGOMERY,AL | 194801-201812 | 83 | 59 | 82 | 56 | 83 | 52 | 87 | |

5 rows × 28 columns

In [16]:

```
humdata.columns
```

Out[16]:

```
Index(['Unnamed: 0', 'POR', 'JAN', 'Unnamed: 3', 'FEB', 'Unnamed: 5',
'MAR',
       'Unnamed: 7', 'APR', 'Unnamed: 9', 'MAY', 'Unnamed: 11', 'JUN',
       'Unnamed: 13', 'JUL', 'Unnamed: 15', 'AUG', 'Unnamed: 17', 'SEP
',
       'Unnamed: 19', 'OCT', 'Unnamed: 21', 'NOV', 'Unnamed: 23', 'DEC
',
       'Unnamed: 25', 'ANN', 'Unnamed: 27'],
      dtype='object')
```

In [17]:

```
humdata.columns = ['0', 'POR', 'JAN', 'JAN', 'FEB', 'FEB', 'MAR',
       'MAR', 'APR', 'APR', 'MAY', 'MAY', 'JUN',
       'JUN', 'JUL', 'JUL', 'AUG', 'AUG', 'SEP',
       'SEP', 'OCT', 'OCT', 'NOV', 'NOV', 'DEC',
       'DEC', 'ANN1', 'ANN2']
```

In [18]:

```
humdata.head()
```

Out[18]:

| | 0 | POR | JAN | JAN | FEB | FEB | MAR | MAR | APR | APR | ... | SEP | SEI |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | NaN | NaN | M | A | M | A | M | A | M | A | ... | M | / |
| **1** | 13876BIRMINGHAM,AL | 194801-201812 | 81 | 60 | 80 | 56 | 80 | 52 | 83 | 50 | ... | 87 | 5( |
| **2** | 03856HUNTSVILLE,AL | 195809-201812 | 81 | 64 | 81 | 60 | 80 | 55 | 82 | 51 | ... | 89 | 5( |
| **3** | 13894MOBILE,AL | 194801-201812 | 83 | 60 | 84 | 58 | 85 | 54 | 88 | 53 | ... | 90 | 5! |
| **4** | 13895MONTGOMERY,AL | 194801-201812 | 83 | 59 | 82 | 56 | 83 | 52 | 87 | 51 | ... | 90 | 5! |

5 rows × 28 columns

In [19]:

```
humdata = humdata[['0' ,'ANN1', 'ANN2']].drop([0])
humdata.head()
```

Out[19]:

| | 0 | ANN1 | ANN2 |
|---|---|---|---|
| 1 | 13876BIRMINGHAM,AL | 84 | 55 |
| 2 | 03856HUNTSVILLE,AL | 85 | 57 |
| 3 | 13894MOBILE,AL | 87 | 57 |
| 4 | 13895MONTGOMERY,AL | 87 | 55 |
| 5 | 26451ANCHORAGE,AK | 74 | 64 |

In [20]:

```
import re
wban = humdata['0']
wban = [re.findall(r'[0-9]+', i)[0] for i in wban]
```

In [21]:

```
humdata['Wban'] = wban
humdata = humdata.drop(['0'], axis = 1)
humdata.head()
```

Out[21]:

| | ANN1 | ANN2 | Wban |
|---|---|---|---|
| 1 | 84 | 55 | 13876 |
| 2 | 85 | 57 | 03856 |
| 3 | 87 | 57 | 13894 |
| 4 | 87 | 55 | 13895 |
| 5 | 74 | 64 | 26451 |

In [22]:

```
humdata['Humidity'] = humdata[['ANN1', 'ANN2']].mean(axis=1)
humdata = humdata.drop(['ANN1', 'ANN2'], axis = 1)
humdata.head()
```

Out[22]:

| | Wban | Humidity |
|---|---|---|
| **1** | 13876 | 69.5 |
| **2** | 03856 | 71.0 |
| **3** | 13894 | 72.0 |
| **4** | 13895 | 71.0 |
| **5** | 26451 | 69.0 |

In [23]:

```
humdata['Wban'] = humdata['Wban'].astype('int64')
```

In [24]:

```
humdata.isnull().any()
```

Out[24]:

```
Wban        False
Humidity    False
dtype: bool
```

In [25]:

```
humdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 266 entries, 1 to 266
Data columns (total 2 columns):
Wban        266 non-null int64
Humidity    266 non-null float64
dtypes: float64(1), int64(1)
memory usage: 6.2 KB
```

In [26]:

```


print (humdata['Wban'].dtype)
print (tempdata['Wban'].dtype)
```

```
int64
int64
```

## Pressure Data

In [27]:

```
presdata = pd.read_excel('Weather/stations.xltx')
presdata.head()
```

Out[27]:

| | Unnamed: 0 | Unnamed: 1 | Unnamed: 2 | Unnamed: 3 | Unnamed: 4 | Unnamed: 5 | Unnamed: 6 | Unnamed: 7 | Ur |
|---|---|---|---|---|---|---|---|---|---|
| **0** | USAF | WBAN | STATION NAME | CTRY | ST CALL | LAT | LON | ELEV(M) | |
| **1** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **2** | 7018 | 99999 | WXPOD 7018 | NaN | NaN | 0 | 0 | 7018 | 2 |
| **3** | 7026 | 99999 | WXPOD 7026 | AF | NaN | 0 | 0 | 7026 | 2 |
| **4** | 7070 | 99999 | WXPOD 7070 | AF | NaN | 0 | 0 | 7070 | 2 |

In [28]:

```
columns = presdata.iloc[0]
presdata.columns = columns
```

In [29]:

```
presdata = presdata.drop([0])
presdata.head()
```

Out[29]:

| | USAF | WBAN | STATION NAME | CTRY | ST CALL | LAT | LON | ELEV(M) | BEGIN | END |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **2** | 7018 | 99999 | WXPOD 7018 | NaN | NaN | 0 | 0 | 7018 | 20110309 | 20130730 |
| **3** | 7026 | 99999 | WXPOD 7026 | AF | NaN | 0 | 0 | 7026 | 20120713 | 20170822 |
| **4** | 7070 | 99999 | WXPOD 7070 | AF | NaN | 0 | 0 | 7070 | 20140923 | 20150926 |
| **5** | 8260 | 99999 | WXPOD8270 | NaN | NaN | 0 | 0 | 0 | 20050101 | 20100920 |

In [30]:

```
presdata = presdata[presdata['CTRY']==('US')]
presdata.head()
```

Out[30]:

| | USAF | WBAN | STATION NAME | CTRY | ST CALL | LAT | LON | ELEV(M) | BEGIN | END |
|---|---|---|---|---|---|---|---|---|---|---|
| **13135** | 621010 | 99999 | MOORED BUOY | US | NaN | 50.6 | -2.933 | -999 | 20080721 | 20080721 |
| **13137** | 621110 | 99999 | MOORED BUOY | US | NaN | 58.9 | -0.2 | -999 | 20041118 | 20041118 |
| **13138** | 621130 | 99999 | MOORED BUOY | US | NaN | 58.4 | 0.3 | -999 | 20040726 | 20040726 |
| **13139** | 621160 | 99999 | MOORED BUOY | US | NaN | 58.1 | 1.8 | -999 | 20040829 | 20040829 |
| **13140** | 621170 | 99999 | MOORED BUOY | US | NaN | 57.9 | 0.1 | -999 | 20040726 | 20040726 |

In [31]:

```
presdata = presdata[['WBAN','ELEV(M)']]
```

In [32]:

```
presdata['ELEV(M)'] = pd.to_numeric(presdata['ELEV(M)'], errors='coerce')
```

In [33]:

```
presdata['WBAN'] = presdata['WBAN'].astype('int64')
```

In [34]:

```
presdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7326 entries, 13135 to 29745
Data columns (total 2 columns):
WBAN       7326 non-null int64
ELEV(M)    7209 non-null float64
dtypes: float64(1), int64(1)
memory usage: 171.7 KB
```

In [35]:

```
presdata.dropna(inplace=True)
```

In [36]:

```
presdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7209 entries, 13135 to 29745
Data columns (total 2 columns):
WBAN        7209 non-null int64
ELEV(M)     7209 non-null float64
dtypes: float64(1), int64(1)
memory usage: 169.0 KB
```

In [37]:

```
presdata.columns = ['Wban', 'Elevation']
presdata.head()
```

Out[37]:

| | Wban | Elevation |
|---|---|---|
| **13135** | 99999 | -999.0 |
| **13137** | 99999 | -999.0 |
| **13138** | 99999 | -999.0 |
| **13139** | 99999 | -999.0 |
| **13140** | 99999 | -999.0 |

In [38]:

```
presdata = presdata[presdata['Elevation']>=0]
presdata.head()
```

Out[38]:

| | Wban | Elevation |
|---|---|---|
| **13149** | 99999 | 0.0 |
| **14465** | 99999 | 1224.0 |
| **14466** | 93218 | 317.0 |
| **14467** | 99999 | 317.0 |
| **14468** | 93217 | 43.0 |

## Average ambient pressure calculation

For this calculation we will use the following expresion:

```
p ~ 101325 (1 - 2.25577^10-5 * h)^5
```

Where:

```
101325 = normal temperature and pressure at sea level (Pa)

p = air pressure (Pa)

h = altitude above sea level (m)
```

Pascal to milibar convertion:

```
1 Pa = 0.01 milibar
```

In [39]:

```
import numpy as np


elev = np.array(presdata['Elevation'])


Pasc = 101325*(1 - 2.25577*10**(-5) * elev)**5


milibar = Pasc * 0.01
```

In [40]:

```
presdata['Pressure'] = milibar
presdata = presdata.drop('Elevation', axis = 1)
```

In [41]:

```
presdata.head()
```

Out[41]:

| | Wban | Pressure |
|---|---|---|
| **13149** | 99999 | 1013.250000 |
| **14465** | 99999 | 880.881800 |
| **14466** | 93218 | 977.536727 |
| **14467** | 99999 | 977.536727 |
| **14468** | 93217 | 1008.345357 |

In [42]:

```
presdata['Wban'].value_counts().head()
```

Out[42]:

```
99999    3548
23176       6
3849        5
14897       4
24255       4
Name: Wban, dtype: int64
```

In [43]:

```
presdata[presdata['Wban']==23176]
```

Out[43]:

| | Wban | Pressure |
|---|---|---|
| **19714** | 23176 | 849.441660 |
| **19722** | 23176 | 849.650098 |
| **19743** | 23176 | 849.650098 |
| **21748** | 23176 | 849.650098 |
| **21749** | 23176 | 849.650098 |
| **29011** | 23176 | 849.650098 |

In [44]:

```
presdata['Wban'] = presdata[presdata['Wban']!=99999]
presdata = presdata.drop_duplicates(subset = 'Wban')
presdata['Wban'].value_counts().head()
```

Out[44]:

```
54791.0    1
24164.0    1
24048.0    1
4898.0     1
3930.0     1
Name: Wban, dtype: int64
```

## Merging weather datasets into US Locations

In [45]:

```
USloc = tempdata.merge(humdata, how='inner')
print (len(USloc))
USloc.head(10)
```

```
260
```

Out[45]:

| | Wban | City | State | Temperature | Humidity |
|---|---|---|---|---|---|
| **0** | 13876 | BIRMINGHAM | AL | 17.388889 | 69.5 |
| **1** | 3856 | HUNTSVILLE | AL | 16.722222 | 71.0 |
| **2** | 13894 | MOBILE | AL | 19.555556 | 72.0 |
| **3** | 13895 | MONTGOMERY | AL | 18.388889 | 71.0 |
| **4** | 26451 | ANCHORAGE | AK | 2.833333 | 69.0 |
| **5** | 25308 | ANNETTE | AK | 8.111111 | 77.5 |
| **6** | 27502 | BARROW | AK | -11.222222 | 81.0 |
| **7** | 26615 | BETHEL | AK | -0.722222 | 78.5 |
| **8** | 26533 | BETTLES | AK | -4.722222 | 67.0 |
| **9** | 26415 | BIG DELTA | AK | -1.666667 | 62.5 |

In [46]:

```
usloc = USloc.merge(presdata, how='left')
```

In [47]:

```
print (len(usloc))
usloc.head(10)
```

260

Out[47]:

| | Wban | City | State | Temperature | Humidity | Pressure |
|---|---|---|---|---|---|---|
| **0** | 13876 | BIRMINGHAM | AL | 17.388889 | 69.5 | 992.002445 |
| **1** | 3856 | HUNTSVILLE | AL | 16.722222 | 71.0 | 991.699106 |
| **2** | 13894 | MOBILE | AL | 19.555556 | 72.0 | 1005.786554 |
| **3** | 13895 | MONTGOMERY | AL | 18.388889 | 71.0 | 1006.229708 |
| **4** | 26451 | ANCHORAGE | AK | 2.833333 | 69.0 | 1009.074145 |
| **5** | 25308 | ANNETTE | AK | 8.111111 | 77.5 | 1009.461485 |
| **6** | 27502 | BARROW | AK | -11.222222 | 81.0 | 1012.164777 |
| **7** | 26615 | BETHEL | AK | -0.722222 | 78.5 | 1009.700784 |
| **8** | 26533 | BETTLES | AK | -4.722222 | 67.0 | 991.362149 |
| **9** | 26415 | BIG DELTA | AK | -1.666667 | 62.5 | 969.545252 |

In [48]:

```
usloc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 260 entries, 0 to 259
Data columns (total 6 columns):
Wban           260 non-null int64
City           260 non-null object
State          260 non-null object
Temperature    260 non-null float64
Humidity       260 non-null float64
Pressure       251 non-null float64
dtypes: float64(3), int64(1), object(2)
memory usage: 14.2+ KB
```

In [49]:

```
usloc = usloc.dropna()
```

In [50]:

```
usloc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 251 entries, 0 to 250
Data columns (total 6 columns):
Wban           251 non-null int64
City           251 non-null object
State          251 non-null object
Temperature    251 non-null float64
Humidity       251 non-null float64
Pressure       251 non-null float64
dtypes: float64(3), int64(1), object(2)
memory usage: 13.7+ KB
```

# Multiple Linear Regression Model

We are going to create a model to preddict energy output (EP) [MW], taking Temperature (T) [°C] , Ambient Pressure (AP) [milibar] , Relative Humidity (RH) [%] as parameters.

We will use the dataset called 'data', which contains the values for the CCPP.

We will use scikit-learn, and train the model as a multiple linear regrssion

In [51]:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt
```

In [52]:

```
X = data[['Temperature', 'Pressure', 'Humidity']]
Y = data[['Power Out']]
```

In [53]:

```
Xtrain, Xtest, Ytrain, Ytest =  train_test_split(X,Y, test_size=0.25, random_state=
```

In [54]:

```
regression = LinearRegression()
model = regression.fit(Xtrain, Ytrain)
```

In [55]:

```
Ypred = model.predict(Xtest)
Ypred
```

Out[55]:

```
array([[455.39353049],
       [447.47658286],
       [437.21191769],
       ...,
       [487.93736352],
       [440.41536484],
       [453.40714577]])
```

In [56]:

```
RMSE = sqrt(mean_squared_error(y_true = Ytest, y_pred = Ypred))
RMSE
```

Out[56]:

```
4.657571391074255
```

In [57]:

```
from sklearn.metrics import r2_score

r2 = r2_score(Ytest, Ypred)
r2
```

Out[57]:

```
0.9266765701220129
```

# Performance prediction at US locations

In this step we will use the model previously trained with the CCPP data, to predict the expected power output of a CCPP situated in differents US locations based on their climate average conditions.

In [58]:

```
usweather = usloc[['Temperature', 'Pressure', 'Humidity']]
```

In [59]:

```
usweather.head()
```

Out[59]:

| | Temperature | Pressure | Humidity |
|---|---|---|---|
| **0** | 17.388889 | 992.002445 | 69.5 |
| **1** | 16.722222 | 991.699106 | 71.0 |
| **2** | 19.555556 | 1005.786554 | 72.0 |
| **3** | 18.388889 | 1006.229708 | 71.0 |
| **4** | 2.833333 | 1009.074145 | 69.0 |

In [60]:

```
X.head()
```

Out[60]:

| | Temperature | Pressure | Humidity |
|---|---|---|---|
| **0** | 14.96 | 1024.07 | 73.17 |
| **1** | 25.18 | 1020.04 | 59.08 |
| **2** | 5.11 | 1012.16 | 92.14 |
| **3** | 20.86 | 1010.24 | 76.64 |
| **4** | 10.82 | 1009.23 | 96.62 |

In [61]:

```
USpred = model.predict(usweather)
USpred[:5]
```

Out[61]:

```
array([[459.91897085],
       [461.1913595 ],
       [454.64134399],
       [457.62249043],
       [495.01181029]])
```

In [62]:

```
usloc['Power_out'] = USpred
```

In [63]:

```
usloc.head()
```

Out[63]:

| | Wban | City | State | Temperature | Humidity | Pressure | Power_out |
|---|---|---|---|---|---|---|---|
| **0** | 13876 | BIRMINGHAM | AL | 17.388889 | 69.5 | 992.002445 | 459.918971 |
| **1** | 3856 | HUNTSVILLE | AL | 16.722222 | 71.0 | 991.699106 | 461.191360 |
| **2** | 13894 | MOBILE | AL | 19.555556 | 72.0 | 1005.786554 | 454.641344 |
| **3** | 13895 | MONTGOMERY | AL | 18.388889 | 71.0 | 1006.229708 | 457.622490 |
| **4** | 26451 | ANCHORAGE | AK | 2.833333 | 69.0 | 1009.074145 | 495.011810 |

# Findings

In [64]:

```
usloc.Power_out.describe()
```

Out[64]:

```
count    251.000000
mean     472.345828
std       14.272938
min      440.563445
25%      462.034303
50%      473.768200
75%      480.319051
max      526.034436
Name: Power_out, dtype: float64
```

In [65]:

```
usloc.drop('Wban',axis=1).corr()
```

Out[65]:

| | Temperature | Humidity | Pressure | Power_out |
|---|---|---|---|---|
| **Temperature** | 1.000000 | -0.057025 | 0.236537 | -0.991844 |
| **Humidity** | -0.057025 | 1.000000 | 0.586223 | -0.048887 |
| **Pressure** | 0.236537 | 0.586223 | 1.000000 | -0.243565 |
| **Power_out** | -0.991844 | -0.048887 | -0.243565 | 1.000000 |

In [66]:

```
usloc.sort_values('Power_out', ascending=False).head(10)
```

Out[66]:

| | Wban | City | State | Temperature | Humidity | Pressure | Power_out |
|---|---|---|---|---|---|---|---|
| **6** | 27502 | BARROW | AK | -11.222222 | 81.0 | 1012.164777 | 526.034436 |
| **147** | 14755 | MT. WASHINGTON | NH | -2.555556 | 0.0 | 812.822902 | 516.454667 |
| **8** | 26533 | BETTLES | AK | -4.722222 | 67.0 | 991.362149 | 512.871904 |
| **17** | 26616 | KOTZEBUE | AK | -5.055556 | 78.0 | 1012.210452 | 512.005739 |
| **11** | 26411 | FAIRBANKS | AK | -2.388889 | 65.0 | 998.288099 | 507.921088 |
| **18** | 26510 | MCGRATH | AK | -2.555556 | 68.5 | 1001.703277 | 507.703816 |
| **19** | 26617 | NOME | AK | -2.555556 | 76.0 | 1012.792951 | 506.490944 |
| **12** | 26425 | GULKANA | AK | -2.111111 | 66.0 | 959.996103 | 506.043838 |
| **9** | 26415 | BIG DELTA | AK | -1.666667 | 62.5 | 969.545252 | 505.946333 |
| **7** | 26615 | BETHEL | AK | -0.722222 | 78.5 | 1009.700784 | 501.556013 |

In [67]:

```
usloc.sort_values('Power_out', ascending=False).tail(10)
```

Out[67]:

| | Wban | City | State | Temperature | Humidity | Pressure | Power_out |
|---|---|---|---|---|---|---|---|
| **61** | 12842 | TAMPA | FL | 23.000000 | 72.0 | 1012.587332 | 446.648664 |
| **70** | 21504 | HILO | HI | 23.277778 | 74.0 | 1011.925011 | 445.569949 |
| **211** | 12919 | BROWNSVILLE | TX | 23.611111 | 75.0 | 1012.416009 | 444.591033 |
| **53** | 12835 | FORT MYERS | FL | 23.944444 | 71.5 | 1012.724408 | 444.511729 |
| **72** | 22516 | KAHULUI | HI | 24.388889 | 67.5 | 1011.479853 | 444.228015 |
| **63** | 12844 | WEST PALM BEACH | FL | 24.111111 | 71.5 | 1012.587332 | 444.112604 |
| **73** | 22536 | LIHUE | HI | 24.333333 | 72.0 | 1009.769163 | 443.409941 |
| **71** | 22521 | HONOLULU | HI | 25.388889 | 65.0 | 1013.010029 | 442.398239 |
| **57** | 12839 | MIAMI | FL | 25.111111 | 71.5 | 1012.244709 | 441.730603 |
| **56** | 12836 | KEY WEST | FL | 25.444444 | 73.5 | 1013.215716 | 440.563445 |

In [68]:

```
statemean = usloc.groupby('State', as_index=False).mean()
statemean = statemean.sort_values('Power_out', ascending=False)
statemean.head()
```

Out[68]:

| | State | Wban | Temperature | Humidity | Pressure | Power_out |
|---|---|---|---|---|---|---|
| **30** | NH | 14750.00 | 2.722222 | 33.250 | 907.087592 | 499.751931 |
| **0** | AK | 26143.45 | 0.519444 | 74.325 | 1003.855505 | 499.293539 |
| **28** | ND | 36963.75 | 5.416667 | 70.875 | 968.073548 | 487.415893 |
| **21** | ME | 14685.50 | 6.222222 | 69.250 | 1001.692198 | 486.723986 |
| **23** | MN | 14920.80 | 5.755556 | 71.900 | 973.431288 | 486.548040 |

# Plotting color US map

In [69]:

```
city = pd.read_excel('uscities.xlsx')
city.head(10)
```

Out[69]:

| | city | city_ascii | state_id | state_name | county_fips | county_name | county_fips_all | county_name_all | |
|---|---|---|---|---|---|---|---|---|---|
| **0** | South Creek | South Creek | WA | Washington | 53053 | Pierce | 53053 | Pierce | 46.9 |
| **1** | Roslyn | Roslyn | WA | Washington | 53037 | Kittitas | 53037 | Kittitas | 47.2 |
| **2** | Sprague | Sprague | WA | Washington | 53043 | Lincoln | 53043 | Lincoln | 47.3 |
| **3** | Gig Harbor | Gig Harbor | WA | Washington | 53053 | Pierce | 53053 | Pierce | 47.3 |
| **4** | Lake Cassidy | Lake Cassidy | WA | Washington | 53061 | Snohomish | 53061 | Snohomish | 48.0 |
| **5** | Tenino | Tenino | WA | Washington | 53067 | Thurston | 53067 | Thurston | 46.8 |

In [70]:

```
city = city[['city','state_id', 'county_fips']]
city.head()
```

Out[70]:

| | city | state_id | county_fips |
|---|---|---|---|
| **0** | South Creek | WA | 53053 |
| **1** | Roslyn | WA | 53037 |
| **2** | Sprague | WA | 53043 |
| **3** | Gig Harbor | WA | 53053 |
| **4** | Lake Cassidy | WA | 53061 |

In [71]:

```
city.city = city.city.str.upper()
city.head()
```

Out[71]:

| | city | state_id | county_fips |
|---|---|---|---|
| **0** | SOUTH CREEK | WA | 53053 |
| **1** | ROSLYN | WA | 53037 |
| **2** | SPRAGUE | WA | 53043 |
| **3** | GIG HARBOR | WA | 53053 |
| **4** | LAKE CASSIDY | WA | 53061 |

In [72]:

```
DF = usloc[['City','State','Power_out']]
DF.columns = ['city','state_id','Power_out']
DF.city = DF.city.str.rstrip()
DF.head()
```

```
/Users/sandra/anaconda3/lib/python3.7/site-packages/pandas/core/generi
c.py:5096: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
(http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-vi
ew-versus-copy)
  self[name] = value
```

Out[72]:

| | city | state_id | Power_out |
|---|---|---|---|
| **0** | BIRMINGHAM | AL | 459.918971 |
| **1** | HUNTSVILLE | AL | 461.191360 |
| **2** | MOBILE | AL | 454.641344 |
| **3** | MONTGOMERY | AL | 457.622490 |
| **4** | ANCHORAGE | AK | 495.011810 |

In [73]:

```
city.head()
```

Out[73]:

| | city | state_id | county_fips |
|---|---|---|---|
| **0** | SOUTH CREEK | WA | 53053 |
| **1** | ROSLYN | WA | 53037 |
| **2** | SPRAGUE | WA | 53043 |
| **3** | GIG HARBOR | WA | 53053 |
| **4** | LAKE CASSIDY | WA | 53061 |

In [74]:

```
city.city[0]
```

Out[74]:

```
'SOUTH CREEK'
```

n=0 for i in city.city: if 'Huntsville' in i: print (city.iloc[n]) n =n+1

In [75]:

```
x = pd.merge(DF, city, on=['city','state_id'])
x.head()
```

Out[75]:

| | city | state_id | Power_out | county_fips |
|---|---|---|---|---|
| **0** | BIRMINGHAM | AL | 459.918971 | 1073 |
| **1** | HUNTSVILLE | AL | 461.191360 | 1089 |
| **2** | MOBILE | AL | 454.641344 | 1097 |
| **3** | MONTGOMERY | AL | 457.622490 | 1101 |
| **4** | ANCHORAGE | AK | 495.011810 | 2020 |

In [76]:

```
x.county_fips[x.county_fips<10000]= ('0'+ x.county_fips.astype(str))
```

```
/Users/sandra/anaconda3/lib/python3.7/site-packages/ipykernel_launcher
.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
(http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-vi
ew-versus-copy)
  """Entry point for launching an IPython kernel.
```

In [77]:

```
len (x)
```

Out[77]:

```
232
```

In [78]:

```
x.head()
```

Out[78]:

| | city | state_id | Power_out | county_fips |
|---|---|---|---|---|
| **0** | BIRMINGHAM | AL | 459.918971 | 01073 |
| **1** | HUNTSVILLE | AL | 461.191360 | 01089 |
| **2** | MOBILE | AL | 454.641344 | 01097 |
| **3** | MONTGOMERY | AL | 457.622490 | 01101 |
| **4** | ANCHORAGE | AK | 495.011810 | 02020 |

In [79]:

```
x.describe()
```

Out[79]:

| | Power_out |
|---|---|
| **count** | 232.000000 |
| **mean** | 471.852604 |
| **std** | 13.909673 |
| **min** | 440.563445 |
| **25%** | 461.496520 |
| **50%** | 473.667091 |
| **75%** | 480.007208 |
| **max** | 512.871904 |

In [ ]:

```

```

In [82]:

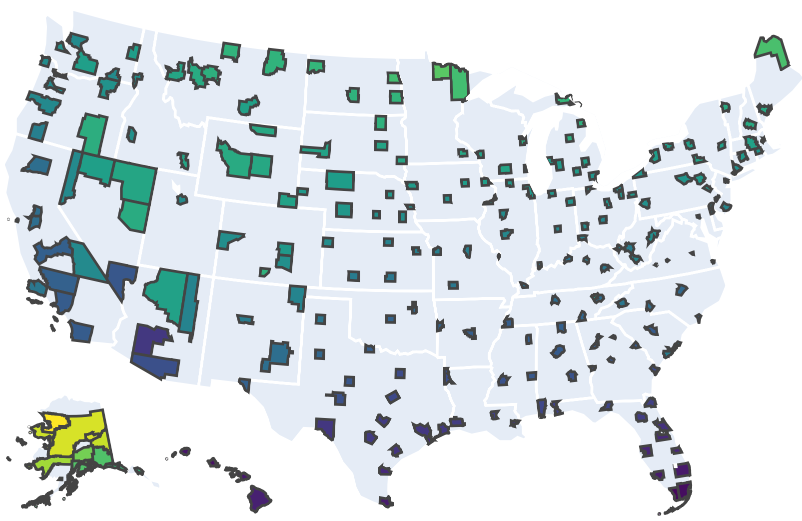
```
from urllib.request import urlopen
import json
with urlopen('https://raw.githubusercontent.com/plotly/datasets/master/geojson-count
    counties = json.load(response)



import plotly.express as px

fig = px.choropleth(x, geojson=counties, locations='county_fips', color='Power_out',
                           color_continuous_scale="Viridis",
                           range_color=(439, 512),
                           scope="usa", labels={'Power_out':'Power Output [MW]'},
                          )
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```

[(https://plot.ly/)](https://plot.ly/)



In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```