In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-pytho
n
# For example, here's several helpful packages to load

import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model, metrics# linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files
under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserve
d as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of
the current session
```

```
/kaggle/input/energyconsumedinindia/Powerdata.csv
/kaggle/input/india-power/Powerdata.csv
/kaggle/input/india-power/long_data.csv
```

In [2]:

```
df = pd.read_csv('../input/energyconsumedinindia/Powerdata.csv')
df2 = pd.read_csv('../input/india-power/Powerdata.csv')
```

In [3]:

```
df.shape
```

Out[3]:

```
(210, 5)
```

In [4]:

```
df2.head()
```

Out[4]:

	Date	Punjab	Haryana	Rajasthan	Delhi	UP	Uttarakhand	HP	J&K	Chandigarh	...	West Bengal	Sikkim	Arunachal Pradesh
0	28/10/2019	95.3	105.7	205.6	59.2	232.8	33.2	24.6	42.6	3.3	...	129.0	1.4	2.1
1	29/10/2019	96.5	111.5	213.6	60.2	240.9	33.8	24.1	43.1	3.2	...	142.0	1.4	2.0
2	30/10/2019	102.4	115.1	215.2	60.7	249.9	34.0	24.7	46.0	3.2	...	142.2	1.4	2.2
3	31/10/2019	103.9	116.2	216.8	60.6	256.7	33.7	24.7	44.8	3.1	...	141.3	1.4	2.2
4	01/11/2019	105.6	117.4	210.3	63.4	263.0	34.4	21.5	47.3	3.1	...	140.2	1.3	2.3

5 rows × 35 columns

In [5]:

```
df2.describe()
```

Out[5]:

Out[5]:

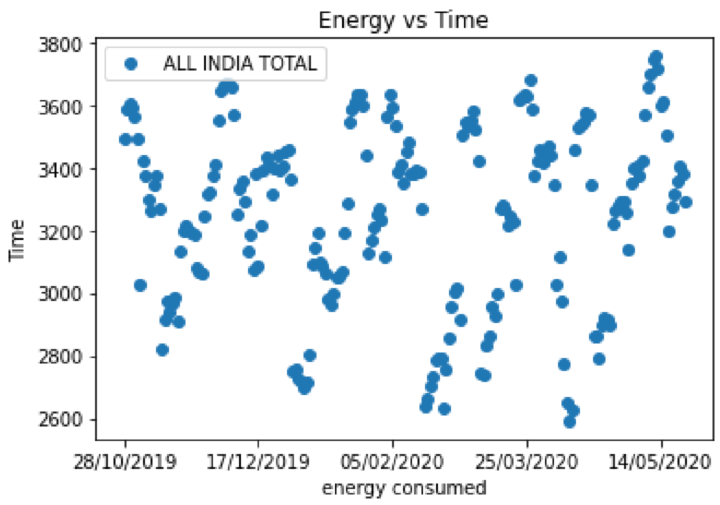
	Punjab	Haryana	Rajasthan	Delhi	UP	Uttarakhand	HP	J&K	Chandigarh	Chhatti
count	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000	210.
mean	104.077619	111.281905	210.673333	64.659524	269.191905	31.842381	24.453333	45.226190	3.480952	76.
std	19.233014	19.320635	31.857113	9.845594	33.564225	7.043567	6.388590	5.326384	0.704408	5.
min	56.100000	64.800000	105.800000	41.800000	186.800000	16.800000	11.800000	17.800000	2.200000	64.
25%	94.325000	98.050000	194.000000	60.300000	248.475000	26.575000	20.200000	42.425000	3.100000	72.
50%	105.650000	115.100000	215.750000	63.800000	268.650000	34.000000	27.200000	44.850000	3.400000	75.
75%	114.000000	125.175000	236.450000	71.325000	284.550000	37.200000	29.275000	49.175000	4.000000	79.
max	171.000000	158.000000	251.700000	95.200000	417.500000	42.200000	32.800000	54.100000	5.000000	91.

8 rows × 34 columns

Linear Regression

In [6]:

```
df2.plot(x='Date', y='ALL INDIA TOTAL', style='o')
plt.title('Energy vs Time')
plt.xlabel('energy consumed')
plt.ylabel('Time')
plt.show()
```



In [7]:

```
X = df.iloc[:, :-1].values
y = df.iloc[:, 1].values
```

In [8]:

```
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 [9]:

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[9]:

```
LinearRegression()
```

In [10]:

```
print(regressor.intercept_)
```

```
7.448092098690722
```

In [11]:

```
print(regressor.coef_)
```

```
[ 0.33343107  0.66656893 -0.33343107 -0.00226358]
```

In [12]:

```
df.head()
```

Out[12]:

	ALL INDIA TOTAL	MEAN	ABSOLUTE DIFFERENCE WITH MEAN	MEDIAN	ABSOLUTE DIFFERENCE MEDIAN
0	3494.4	3249.592381	244.807619	3290.4	204.0
1	3586.4	3249.592000	336.808000	3290.4	296.0
2	3604.8	3249.592000	355.208000	3290.4	314.4
3	3593.5	3249.592000	343.908000	3290.4	303.1
4	3565.2	3249.592000	315.608000	3290.4	274.8

In [13]:

```
y_pred = regressor.predict(X_test)
```

In [14]:

```
#df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
#df
```