Importing libraries

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

Importing dataset

In [2]:
```
data=pd.read_csv(r"C:\Users\user\Downloads\nuclear explosion.csv")
data
```

Out[2]:

	WEAPON SOURCE COUNTRY	WEAPON DEPLOYMENT LOCATION	Data.Source	Location.Cordinates.Latitude	Location.Cordinates.Longitude
0	USA	Alamogordo	DOE	32.54	-105.57
1	USA	Hiroshima	DOE	34.23	132.27
2	USA	Nagasaki	DOE	32.45	129.52
3	USA	Bikini	DOE	11.35	165.20
4	USA	Bikini	DOE	11.35	165.20
...	...	...	...	...	...
2041	CHINA	Lop Nor	HFS	41.69	88.35
2042	INDIA	Pokhran	HFS	27.07	71.70
2043	INDIA	Pokhran	NRD	27.07	71.70
2044	PAKIST	Chagai	HFS	28.90	64.89
2045	PAKIST	Kharan	HFS	28.49	63.78

2046 rows × 16 columns

info

In [3]:
```
# to identify missing values
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2046 entries, 0 to 2045
Data columns (total 16 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   WEAPON SOURCE COUNTRY         2046 non-null   object
 1   WEAPON DEPLOYMENT LOCATION    2046 non-null   object
 2   Data.Source                   2046 non-null   object
```

```
 3   Location.Cordinates.Latitude   2046 non-null   float64
 4   Location.Cordinates.Longitude  2046 non-null   float64
 5   Data.Magnitude.Body            2046 non-null   float64
 6   Data.Magnitude.Surface         2046 non-null   float64
 7   Location.Cordinates.Depth      2046 non-null   float64
 8   Data.Yeild.Lower               2046 non-null   float64
 9   Data.Yeild.Upper               2046 non-null   float64
 10  Data.Purpose                   2046 non-null   object
 11  Data.Name                      2046 non-null   object
 12  Data.Type                      2046 non-null   object
 13  Date.Day                       2046 non-null   int64
 14  Date.Month                     2046 non-null   int64
 15  Date.Year                      2046 non-null   int64
dtypes: float64(7), int64(3), object(6)
memory usage: 255.9+ KB
```

describe

In [4]:

```
# to display summary of the dataset
data.describe()
```

Out[4]:

	Location.Cordinates.Latitude	Location.Cordinates.Longitude	Data.Magnitude.Body	Data.Magnitude
count	2046.000000	2046.000000	2046.000000	204
mean	35.462429	-36.015037	2.145406	
std	23.352702	100.829355	2.625453	
min	-49.500000	-169.320000	0.000000	
25%	37.000000	-116.051500	0.000000	
50%	37.100000	-116.000000	0.000000	
75%	49.870000	78.000000	5.100000	
max	75.100000	179.220000	7.400000	

columns

In [5]:

```
# to display headings of the dataset
data.columns
```

Out[5]:

```
Index(['WEAPON SOURCE COUNTRY', 'WEAPON DEPLOYMENT LOCATION', 'Data.Source',
       'Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
       'Data.Magnitude.Body', 'Data.Magnitude.Surface',
       'Location.Cordinates.Depth', 'Data.Yeild.Lower', 'Data.Yeild.Upper',
       'Data.Purpose', 'Data.Name', 'Data.Type', 'Date.Day', 'Date.Month',
       'Date.Year'],
      dtype='object')
```

In [6]:

```
a=data.dropna(axis=1)
a
```

Out[6]:

	WEAPON SOURCE COUNTRY	WEAPON DEPLOYMENT LOCATION	Data.Source	Location.Cordinates.Latitude	Location.Cordinates.Longitude
0	USA	Alamogordo	DOE	32.54	-105.57
1	USA	Hiroshima	DOE	34.23	132.27
2	USA	Nagasaki	DOE	32.45	129.52
3	USA	Bikini	DOE	11.35	165.20
4	USA	Bikini	DOE	11.35	165.20
...	...	...	...	...	...
2041	CHINA	Lop Nor	HFS	41.69	88.35
2042	INDIA	Pokhran	HFS	27.07	71.70
2043	INDIA	Pokhran	NRD	27.07	71.70
2044	PAKIST	Chagai	HFS	28.90	64.89
2045	PAKIST	Kharan	HFS	28.49	63.78

2046 rows × 16 columns

In [7]:

```
a.columns
```

Out[7]:

```
Index(['WEAPON SOURCE COUNTRY', 'WEAPON DEPLOYMENT LOCATION', 'Data.Source',
       'Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
       'Data.Magnitude.Body', 'Data.Magnitude.Surface',
       'Location.Cordinates.Depth', 'Data.Yeild.Lower', 'Data.Yeild.Upper',
       'Data.Purpose', 'Data.Name', 'Data.Type', 'Date.Day', 'Date.Month',
       'Date.Year'],
      dtype='object')
```

To train the model-Model Building

In [8]:

```
x=a[[ 'Date.Day']]
y=a['Date.Month']
```

In [9]:

```
# to split my dataset into training and test data
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

Linear regression

In [10]:

```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

Out[10]:
```
LinearRegression()
```

In [11]:
```
print(lr.intercept_)
```

```
7.479749493353967
```

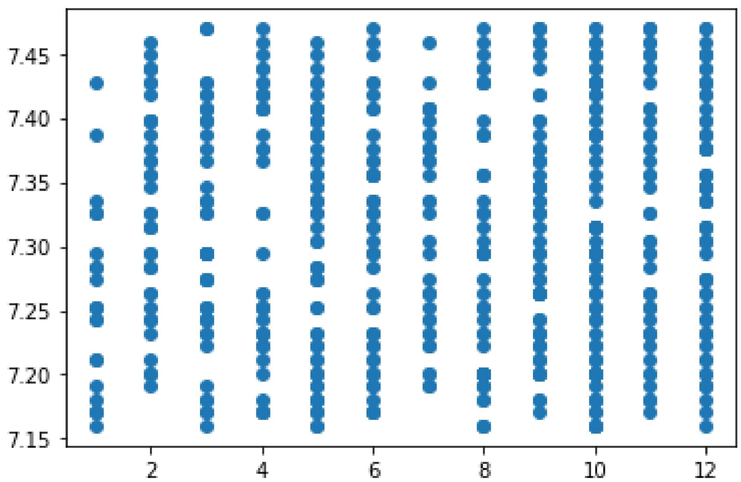
In [12]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[12]:

	Co-efficient
Date.Day	-0.010308

In [13]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[13]:
```
<matplotlib.collections.PathCollection at 0x1ec6a8ae5e0>
```



In [14]:
```
print(lr.score(x_test,y_test))
```

```
0.0018011058221996112
```

In [15]:
```
lr.score(x_train,y_train)
```

Out[15]:
```
0.0008956001221300802
```

Ridge regression

In [16]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [17]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[17]: 0.0018009516561215966

In [18]:
```
rr.score(x_train,y_train)
```

Out[18]: 0.0008956001149248438

Lasso regression

In [19]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
la.score(x_train,y_train)
```

Out[19]: 0.0

In [20]:
```
la.score(x_test,y_test)
```

Out[20]: -0.0006465534092729985

Elastic net regression

In [21]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[21]: ElasticNet()

In [22]:
```
print(en.coef_)
```

[-0.00386094]

In [23]:
```
print(en.intercept_)
```

7.372302733406749

In [24]:
```
predict=en.predict(x_test)
```

In [25]:
```
print(en.score(x_test,y_test))
```

0.00044095649985187446

In [26]:
```
from sklearn import metrics
```

In [27]:
```
print("Mean Absolute error:",metrics.mean_absolute_error(y_test,predict))
```

Mean Absolute error: 2.921026596447633

In [28]:
```
print("Mean Squared error:",metrics.mean_squared_error(y_test,predict))
```

```
Mean Squared error: 11.12608779976566
```

In [29]:
```
print("Root squared error:",np.sqrt(metrics.mean_squared_error(y_test,predict)))
```

```
Root squared error: 3.335579080124718
```

Model saving

In [30]:
```
import pickle
filename="prediction"
pickle.dump(lr,open(filename,'wb'))
filename='prediction'
model=pickle.load(open(filename,'rb'))
```

In [32]:
```
real=[[10],[7]]
result=model.predict(real)
result
```

Out[32]:
```
array([7.37666482, 7.40759022])
```

In [ ]: