In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [2]:
```
df = pd.read_csv("nuclear.csv").dropna()
df
```

Out[2]:

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

2046 rows × 16 columns

In [3]:
```
df.head()
```

Out[3]:

	WEAPON SOURCE COUNTRY	WEAPON DEPLOYMENT LOCATION	Data.Source	Location.Cordinates.Latitude	Location.Cordinates.Longi
0	USA	Alamogordo	DOE	32.54	-10
1	USA	Hiroshima	DOE	34.23	13
2	USA	Nagasaki	DOE	32.45	12
3	USA	Bikini	DOE	11.35	16
4	USA	Bikini	DOE	11.35	16

Data cleaning and pre processing

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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: 271.7+ KB
```

In [5]:
```
df.describe()
```

Out[5]:

	Location.Cordinates.Latitude	Location.Cordinates.Longitude	Data.Magnitude.Body	Data.Mag
count	2046.000000	2046.000000	2046.000000	
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	

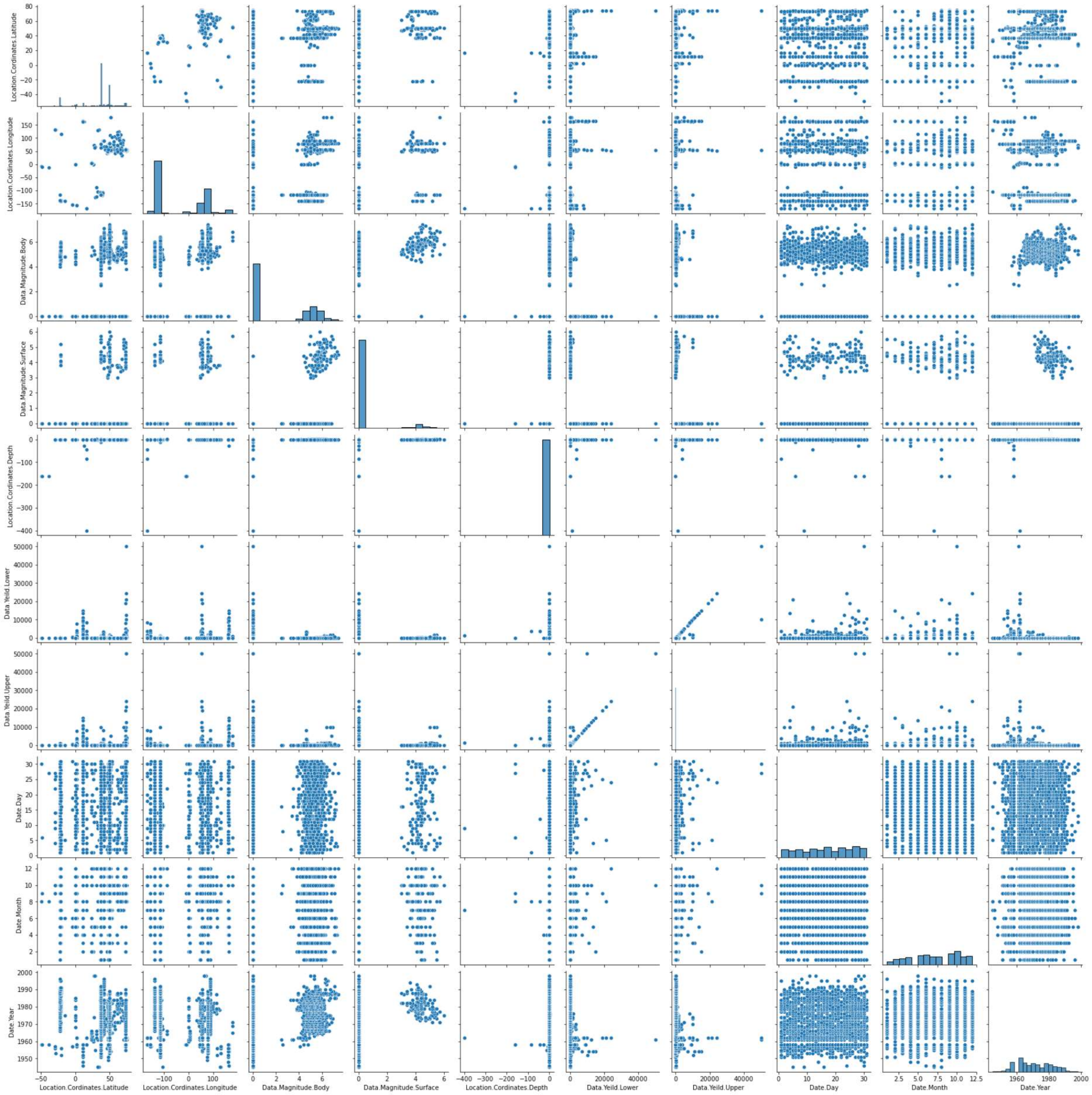
In [6]:
```
df.columns
```

Out[6]:
```
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')
```

EDA and VISUALIZATION

In [7]:
```
sns.pairplot(df)
```

Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x173014c1820>
```



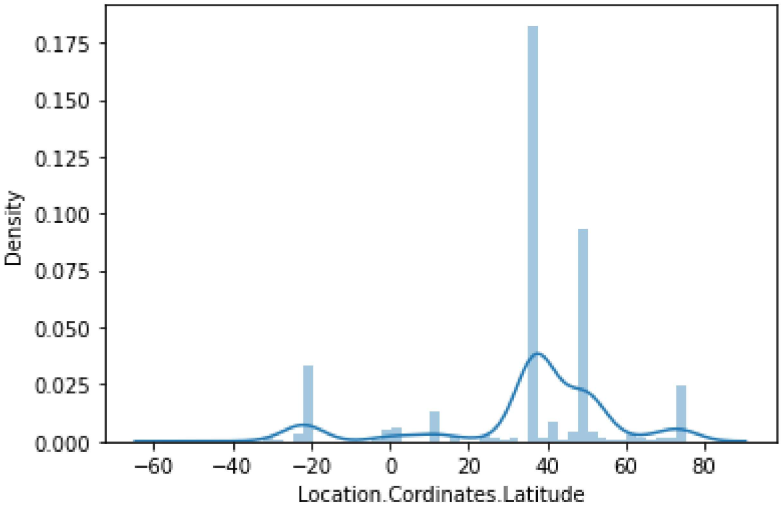
In [8]:

```
sns.distplot(df['Location.Cordinates.Latitude'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

Out[8]:

```
<AxesSubplot:xlabel='Location.Cordinates.Latitude', ylabel='Density'>
```

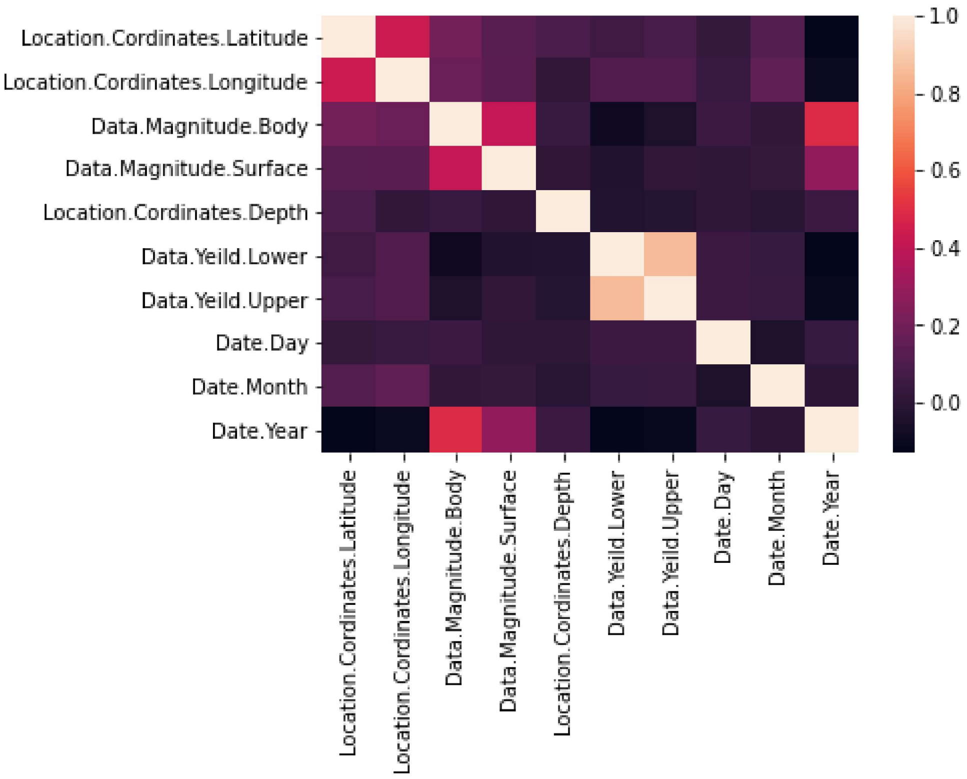


In [9]:

```
df1 = df[['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']]
```

```
In [10]: sns.heatmap(df1.corr())
```

```
Out[10]: <AxesSubplot:>
```



```
In [11]: x = df1[['Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
                'Data.Magnitude.Body',
                'Location.Cordinates.Depth', 'Data.Yeild.Lower', 'Data.Yeild.Upper', 'Da
                'Date.Year']]
         y = df1['Data.Magnitude.Surface']
```

split the data into training and test data

```
In [12]: x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3)
```

```
In [13]: lr = LinearRegression()
         lr.fit(x_train, y_train)
```

```
Out[13]: LinearRegression()
```

```
In [14]: lr.intercept_
```

```
Out[14]: -34.10720923778148
```

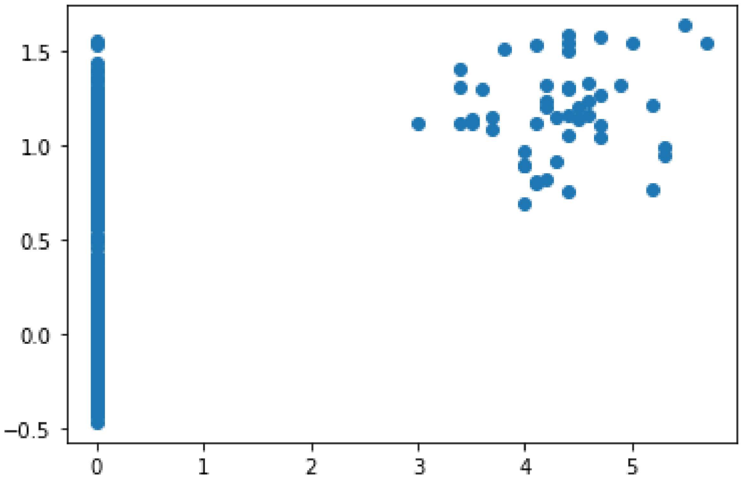
In [15]:
```
coeff = pd.DataFrame(lr.coef_, x.columns, columns =['Co-efficient'])
coeff
```

Out[15]:

	Co-efficient
Location.Cordinates.Latitude	0.003430
Location.Cordinates.Longitude	0.000812
Data.Magnitude.Body	0.147027
Location.Cordinates.Depth	-0.001198
Data.Yeild.Lower	-0.000049
Data.Yeild.Upper	0.000045
Date.Day	-0.003632
Date.Month	0.003313
Date.Year	0.017296

In [16]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test, prediction)
```

Out[16]:
```
<matplotlib.collections.PathCollection at 0x17312755670>
```



In [17]:
```
lr.score(x_test,y_test)
```

Out[17]:
```
0.19921989735087386
```

In [18]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [19]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
rr.score(x_train,y_train)
```

Out[19]:
```
0.1948024994212233
```

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

Out[20]:
```
0.19921997008820713
```

In [21]:
```
la = Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[21]:
```
Lasso(alpha=10)
```

In [22]:
```
la.score(x_test,y_test)
```

Out[22]:
```
0.017800264797578902
```

In [23]:
```
from sklearn.linear_model import ElasticNet
en = ElasticNet()
en.fit(x_train,y_train)
```

Out[23]:
```
ElasticNet()
```

In [24]:
```
print(en.coef_)
```

```
[ 4.13404901e-03  1.25583655e-03  5.13577239e-02 -0.00000000e+00
 -7.68123337e-05  5.26147310e-05 -0.00000000e+00  0.00000000e+00
  2.42579176e-02]
```

In [25]:
```
print(en.intercept_)
```

```
-47.667432481886046
```

In [26]:
```
print(en.predict(x_test))
```

```
[ 2.26031951e-01 -1.63147026e-01 -6.50630608e-02  8.33484951e-02
  7.56102611e-01  9.76700058e-01  6.44255909e-01 -3.33702575e-01
  5.76643363e-01  3.31232264e-01  1.55859706e-02  7.32735069e-01
  2.82889056e-01  4.71869158e-01  3.10424715e-02  8.47925579e-02
  2.31259625e-01  5.52484350e-01  2.22126915e-03  5.71104038e-02
  2.01774033e-01  2.31119226e-01 -4.08051431e-02  4.34963737e-01
 -3.83084374e-01 -3.56183068e-01  4.71728759e-01 -1.54218797e-01
  1.53258198e-01  9.23463959e-01 -1.63679373e-01  2.87860923e-01
  2.31201498e-01  8.36295005e-01  8.80134862e-01  3.70513710e-01
  7.03211295e-01 -3.32976647e-01  1.31411030e+00 -1.65472255e-02
  8.86071222e-01  7.56382277e-01  2.31258822e-01  2.01774033e-01
  7.99183670e-02  7.11737239e-01 -1.65472255e-02  6.19386307e-01
  5.71922637e-01  3.17627163e-01  7.26649733e-01  3.17846127e-01
  1.04357740e-01  1.95829244e-01  7.27195821e-01  4.80125474e-03
  8.98958349e-01  1.29273981e-01  5.50914268e-01  9.13277378e-01
  1.05468052e+00 -6.76360184e-02  2.48418856e-01 -2.67377866e-01
  6.68436373e-01  6.63762835e-01  8.04970033e-02  3.10714378e-01
  3.55574944e-01  9.65432603e-04  1.75394548e-01  6.43644224e-01
 -1.63146973e-01  5.51342695e-01  8.23830107e-02  1.75545424e-01
  4.86580404e-01  3.00083432e-01  6.35115078e-01  5.06130554e-01
```

In [27]:
```
print(en.score(x_test,y_test))
```

```
0.17049406052259697
```

Evaluation Metrics

In [28]:
```
from sklearn import metrics
```

In [29]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 0.6363042556370765
```

In [30]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 1.118977148440899
```

In [31]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,pred
```

```
Root Mean Squared Error: 1.057817162103593
```

In [32]:
```
import pickle
```

In [33]:
```
filename='prediction'
pickle.dump(lr,open(filename,'wb'))
```

In [34]:
```
import pandas as pd
import pickle
```

In [35]:
```
filename='prediction'
model = pickle.load(open(filename,'rb'))
```

In [37]:
```
real = [[10,20,30,44,55,66,99,77,88],[11,45,55,12,23,56,45,89,78]]
result = model.predict(real)
```

In [38]:
```
result
```

Out[38]:
```
array([-28.28074631, -24.47893551])
```