DAY-10

Nuclear

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
df=pd.read_csv(r"C:\Users\user\Downloads\nuclear.csv")[0:500]
df
```

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
...	...	...	...	...	...
495	USA	Nts	DOE	37.00	-116.00
496	USSR	Semi Kazakh	MTM	50.00	78.00
497	USA	Johnston Is	DOE	16.45	-169.32
498	USSR	Mtr Russ	DOE	48.00	46.00
499	USSR	Nz Russ	UGS	73.40	54.90

500 rows × 16 columns

In [3]:
```
df.head(10)
```

Out[3]:

	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
5	USA	Enewetak	DOE	11.30	162.15
6	USA	Enewetak	DOE	11.30	162.15
7	USA	Enewetak	DOE	11.30	162.15
8	USSR	Semi Kazakh	DOE	48.00	76.00
9	USA	Nts	DOE	37.00	-116.00

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

Out[4]:

	Location.Cordinates.Latitude	Location.Cordinates.Longitude	Data.Magnitude.Body	Data.Magnitude
count	500.000000	500.000000	500.000000	
mean	36.870452	-5.856688	0.156600	
std	23.508740	112.909903	0.799309	
min	-49.500000	-169.320000	0.000000	
25%	26.000000	-116.000000	0.000000	
50%	37.000000	51.950000	0.000000	
75%	50.000000	78.000000	0.000000	
max	75.100000	165.200000	4.900000	

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 16 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   WEAPON SOURCE COUNTRY          500 non-null    object 
 1   WEAPON DEPLOYMENT LOCATION     500 non-null    object 
 2   Data.Source                    500 non-null    object 
 3   Location.Cordinates.Latitude   500 non-null    float64
 4   Location.Cordinates.Longitude  500 non-null    float64
 5   Data.Magnitude.Body            500 non-null    float64
 6   Data.Magnitude.Surface         500 non-null    float64
 7   Location.Cordinates.Depth      500 non-null    float64
 8   Data.Yeild.Lower               500 non-null    float64
 9   Data.Yeild.Upper               500 non-null    float64
 10  Data.Purpose                   500 non-null    object 
 11  Data.Name                      500 non-null    object 
 12  Data.Type                      500 non-null    object 
 13  Date.Day                       500 non-null    int64  
 14  Date.Month                     500 non-null    int64  
 15  Date.Year                      500 non-null    int64  
dtypes: float64(7), int64(3), object(6)
memory usage: 62.6+ KB
```

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')
```

In [7]:
```
x=df[['Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
       'Data.Magnitude.Body', 'Data.Magnitude.Surface',
       'Location.Cordinates.Depth', 'Data.Yeild.Lower', 'Data.Yeild.Upper', 'Date.Da
y=df['Date.Year']
```

In [8]:
```
#to split my dataset into traning 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)
```

In [9]:
```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(x_train,y_train)
```

Out[9]:
```
LinearRegression()
```

In [10]:
```
print(lr.intercept_)
```

```
1956.085113316194
```

In [11]:
```
print(lr.score(x_test,y_test))
```

```
-0.0016937691379574904
```

In [12]:
```
lr.score(x_train,y_train)
```

Out[12]:
```
0.11426610333689491
```

Ridge Regression

In [13]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [14]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[14]:
```
Ridge(alpha=10)
```

In [15]:
```
rr.score(x_test,y_test)
```

Out[15]:
```
-0.0023464494652720713
```

Lasso Regression

In [16]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[16]:
```
Lasso(alpha=10)
```

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

Out[17]:
```
-0.08134692102745356
```

In [18]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[18]:
```
ElasticNet()
```

In [19]:
```
print(en.intercept_)
```

```
1956.4905355643978
```

In [20]:
```
print(en.coef_)
```

```
[ 0.0298699  -0.00623519  0.          0.         -0.00810022 -0.00028627
  0.00038146 -0.          0.04594014]
```

```
In [21]: predict=(en.predict(x_test))
```

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

         -0.03656030531580434
```

Evaluation Metrix

```
In [23]: from sklearn import metrics
```

```
In [24]: print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,predict))

         Mean Absolute Error: 2.917323918471769
```

```
In [25]: print("Mean Square Error:",metrics.mean_squared_error(y_test,predict))

         Mean Square Error: 12.799400580692652
```

```
In [26]: print("Root Mean Square Error:",np.sqrt(metrics.mean_squared_error(y_test,predict))

         Root Mean Square Error: 3.5776249916240035
```

Model saving

```
In [27]: import pickle
```

```
In [28]: filename="predict"
         pickle.dump(lr,open(filename,'wb'))
```

```
In [ ]:
```