Problem Statement

Linear Regression

Import Libraries

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

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

Out[2]:

	WEAPON SOURCE COUNTRY	WEAPON DEPLOYMENT LOCATION	Data.Source	Location.Cordinates.Latitude	Location.Cordinates.Longit
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.
...	...	...	...	...	
2041	CHINA	Lop Nor	HFS	41.69	8
2042	INDIA	Pokhran	HFS	27.07	7
2043	INDIA	Pokhran	NRD	27.07	7
2044	PAKIST	Chagai	HFS	28.90	6
2045	PAKIST	Kharan	HFS	28.49	6

2046 rows × 16 columns

To display top 10 rows

In [3]:
```
c=a.head(15)
c
```

Out[3]:

	WEAPON SOURCE COUNTRY	WEAPON DEPLOYMENT LOCATION	Data.Source	Location.Cordinates.Latitude	Location.Cordinates.Longitud
0	USA	Alamogordo	DOE	32.54	-105.5
1	USA	Hiroshima	DOE	34.23	132.2

	WEAPON SOURCE COUNTRY	WEAPON DEPLOYMENT LOCATION	Data.Source	Location.Cordinates.Latitude	Location.Cordinates.Longitud
2	USA	Nagasaki	DOE	32.45	129.5
3	USA	Bikini	DOE	11.35	165.2
4	USA	Bikini	DOE	11.35	165.2
5	USA	Enewetak	DOE	11.30	162.1
6	USA	Enewetak	DOE	11.30	162.1
7	USA	Enewetak	DOE	11.30	162.1
8	USSR	Semi Kazakh	DOE	48.00	76.0
9	USA	Nts	DOE	37.00	-116.0
10	USA	Nts	DOE	37.00	-116.0
11	USA	Nts	DOE	37.00	-116.0
12	USA	Nts	DOE	37.00	-116.0
13	USA	Nts	DOE	37.00	-116.0
14	USA	Enewetak	DOE	11.30	162.1

To find Missing values

In [4]:

```
c.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15 entries, 0 to 14
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   WEAPON SOURCE COUNTRY           15 non-null     object 
 1   WEAPON DEPLOYMENT LOCATION      15 non-null     object 
 2   Data.Source                     15 non-null     object 
 3   Location.Cordinates.Latitude    15 non-null     float64
 4   Location.Cordinates.Longitude   15 non-null     float64
 5   Data.Magnitude.Body             15 non-null     float64
 6   Data.Magnitude.Surface          15 non-null     float64
 7   Location.Cordinates.Depth       15 non-null     float64
 8   Data.Yeild.Lower                15 non-null     float64
 9   Data.Yeild.Upper                15 non-null     float64
 10  Data.Purpose                    15 non-null     object 
 11  Data.Name                       15 non-null     object 
 12  Data.Type                       15 non-null     object 
 13  Date.Day                        15 non-null     int64  
 14  Date.Month                      15 non-null     int64  
 15  Date.Year                       15 non-null     int64  
dtypes: float64(7), int64(3), object(6)
memory usage: 2.0+ KB
```

To display summary of statistics

In [5]:

```
a.describe()
```

Out[5]:

	Location.Cordinates.Latitude	Location.Cordinates.Longitude	Data.Magnitude.Body	Data.Magni
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	

To display column heading

In [6]:

```
a.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')
```

Pairplot

In [7]:

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

Out[7]:

	WEAPON SOURCE COUNTRY	WEAPON DEPLOYMENT LOCATION	Data.Source	Location.Cordinates.Latitude	Location.Cordinates.Longit
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.
...	...	...	...	...	
2041	CHINA	Lop Nor	HFS	41.69	8
2042	INDIA	Pokhran	HFS	27.07	7
2043	INDIA	Pokhran	NRD	27.07	7
2044	PAKIST	Chagai	HFS	28.90	6
2045	PAKIST	Kharan	HFS	28.49	6.

2046 rows × 16 columns

In [8]:
```
s.columns
```

Out[8]:
```
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

In [12]:
```
g=c[['Date.Day', 'Date.Month']]
h=c['Date.Year']
```

To split dataset into training end test

In [13]:
```
from sklearn.model_selection import train_test_split
g_train,g_test,h_train,h_test=train_test_split(g,h,test_size=0.6)
```

To run the model

In [14]:
```
from sklearn.linear_model import LinearRegression
```

In [15]:
```
lr=LinearRegression()
lr.fit(g_train,h_train)
```

Out[15]:
```
LinearRegression()
```

In [16]:
```
print(lr.intercept_)
```

```
1953.17314441559
```

Coeffecient

In [17]:
```
coeff=pd.DataFrame(lr.coef_,g.columns,columns=['Co-effecient'])
coeff
```

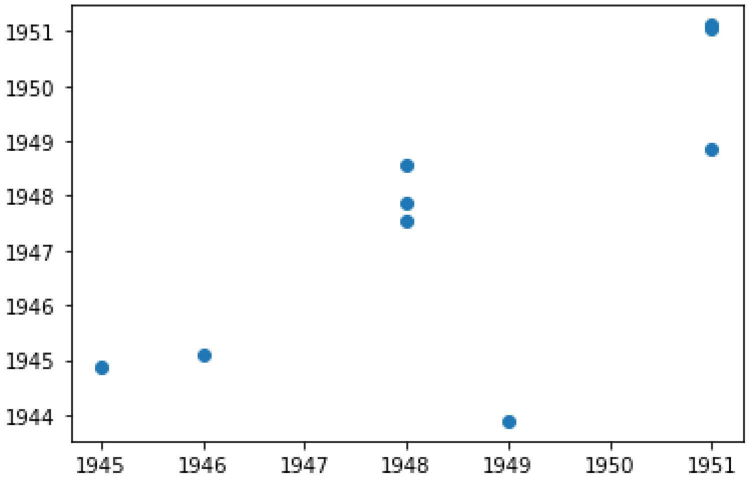
Out[17]:

	Co-effecient
Date.Day	-0.041670
Date.Month	-1.009497

Best Fit line

In [18]:
```
prediction=lr.predict(g_test)
plt.scatter(h_test,prediction)
```

Out[18]: <matplotlib.collections.PathCollection at 0x20a33e58190>



To find score

In [19]:
```
print(lr.score(g_test,h_test))
```

0.15945778074845496

Import Lasso and ridge

In [20]:
```
from sklearn.linear_model import Ridge,Lasso
```

Ridge

In [21]:
```
ri=Ridge(alpha=5)
ri.fit(g_train,h_train)
```

Out[21]: Ridge(alpha=5)

In [22]:
```
ri.score(g_test,h_test)
```

Out[22]: 0.2545529281809321

In [23]:
```
ri.score(g_train,h_train)
```

Out[23]: 0.9844307302728446

Lasso

In [24]:
```
l=Lasso(alpha=6)
l.fit(g_train,h_train)
```

Out[24]: Lasso(alpha=6)

In [25]:
```
l.score(g_test,h_test)
```

Out[25]: 0.26660855085315205

In [26]:
```
ri.score(g_train,h_train)
```

Out[26]: 0.9844307302728446

ElasticNet

In [27]:
```
from sklearn.linear_model import ElasticNet
e=ElasticNet()
e.fit(g_train,h_train)
```

Out[27]: ElasticNet()

Coeffecient,intercept

In [28]:
```
print(e.coef_)
```

[-0.03958092 -0.88535988]

In [29]:
```
print(e.intercept_)
```

1952.603536717083

Prediction

In [30]:
```
d=e.predict(g_test)
d
```

Out[30]:
```
array([1948.78503078, 1950.64949208, 1947.87466969, 1950.79323604,
       1945.3227531 , 1948.50796436, 1947.62260448, 1944.37281109,
       1945.45607556])
```

In [31]:
```
print(e.score(g_test,h_test))
```

0.28578827701632137

Evaluation

In [32]:
```
from sklearn import metrics
print("Mean Absolute error:",metrics.mean_absolute_error(h_test,d))
```

```
Mean Absolute error: 1.0307553048549432
```

In [33]:

```
print("Mean Squared error:",metrics.mean_squared_error(h_test,d))
```

```
Mean Squared error: 3.033195465510931
```

In [34]:

```
print("Mean Squared error:",np.sqrt(metrics.mean_squared_error(h_test,d)))
```

```
Mean Squared error: 1.741607150166458
```

Model Saving

In [35]:

```
import pickle
filename="pre"
pickle.dump(lr,open(filename,"wb"))
```

In [36]:

```
filename='pre'
model = pickle.load(open(filename,'rb'))
```

In [39]:

```
eral=[[15,10],[19,54]]
result=model.predict(eral)
result
```

Out[39]:

```
array([1942.45313094, 1897.86860148])
```

In [ ]: