DATA COLLECTION

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

In [2]:
```
# To Import Dataset
sd=pd.read_csv(r"c:\Users\user\Downloads\19_nuclear_explosions.csv")
sd
```

Out[2]:

ody	Data.Magnitude.Surface	Location.Cordinates.Depth	Data.Yeild.Lower	Data.Yeild.Upper	Data.Purpose	Data.Name	Data.Type	Date.Day
0.0	0.0	-0.10	21.0	21.0	Wr	Trinity	Tower	16
0.0	0.0	-0.60	15.0	15.0	Combat	Littleboy	Airdrop	5
0.0	0.0	-0.60	21.0	21.0	Combat	Fatman	Airdrop	9
0.0	0.0	-0.20	21.0	21.0	We	Able	Airdrop	30
0.0	0.0	0.03	21.0	21.0	We	Baker	Uw	24
...	...	...	...	...	...	...	...	...
5.3	0.0	0.00	3.0	12.0	Wr	Nan	Ug	29
5.3	0.0	0.00	0.0	20.0	Wr	Shakti 1-3	Ug	11
0.0	0.0	0.00	0.0	1.0	Wr	Nan	Ug	13
0.0	0.0	0.00	0.0	35.0	Wr	Nan	Ug	28
5.0	0.0	0.00	0.0	18.0	Wr	Nan	Ug	30

In [3]:
```
# to display top 10 rows
sd.head(10)
```

Out[3]:

	WEAPON SOURCE COUNTRY	WEAPON DEPLOYMENT LOCATION	Data.Source	Location.Cordinates.Latitude	Location.Cordinates.Longitude	Data.Magnitude.Body	Data.Magnitu
0	USA	Alamogordo	DOE	32.54	-105.57	0.0	
1	USA	Hiroshima	DOE	34.23	132.27	0.0	
2	USA	Nagasaki	DOE	32.45	129.52	0.0	
3	USA	Bikini	DOE	11.35	165.20	0.0	
4	USA	Bikini	DOE	11.35	165.20	0.0	
5	USA	Enewetak	DOE	11.30	162.15	0.0	
6	USA	Enewetak	DOE	11.30	162.15	0.0	
7	USA	Enewetak	DOE	11.30	162.15	0.0	
8	USSR	Semi Kazakh	DOE	48.00	76.00	0.0	
9	USA	Nts	DOE	37.00	-116.00	0.0	

DATA CLEANING AND PRE_PROCESSING

In [4]:
```
sd.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
```

In [5]:
```
# to display summary of statistics
sd.describe()
```

Out[5]:

	Location.Cordinates.Latitude	Location.Cordinates.Longitude	Data.Magnitude.Body	Data.Magnitude.Surface	Location.Cordinates.Depth
count	2046.000000	2046.000000	2046.000000	2046.000000	2046.00000
mean	35.462429	-36.015037	2.145406	0.356696	-0.49082
std	23.352702	100.829355	2.625453	1.203569	10.98107
min	-49.500000	-169.320000	0.000000	0.000000	-400.00000
25%	37.000000	-116.051500	0.000000	0.000000	0.00000
50%	37.100000	-116.000000	0.000000	0.000000	0.00000
75%	49.870000	78.000000	5.100000	0.000000	0.00000
max	75.100000	179.220000	7.400000	6.000000	1.45100

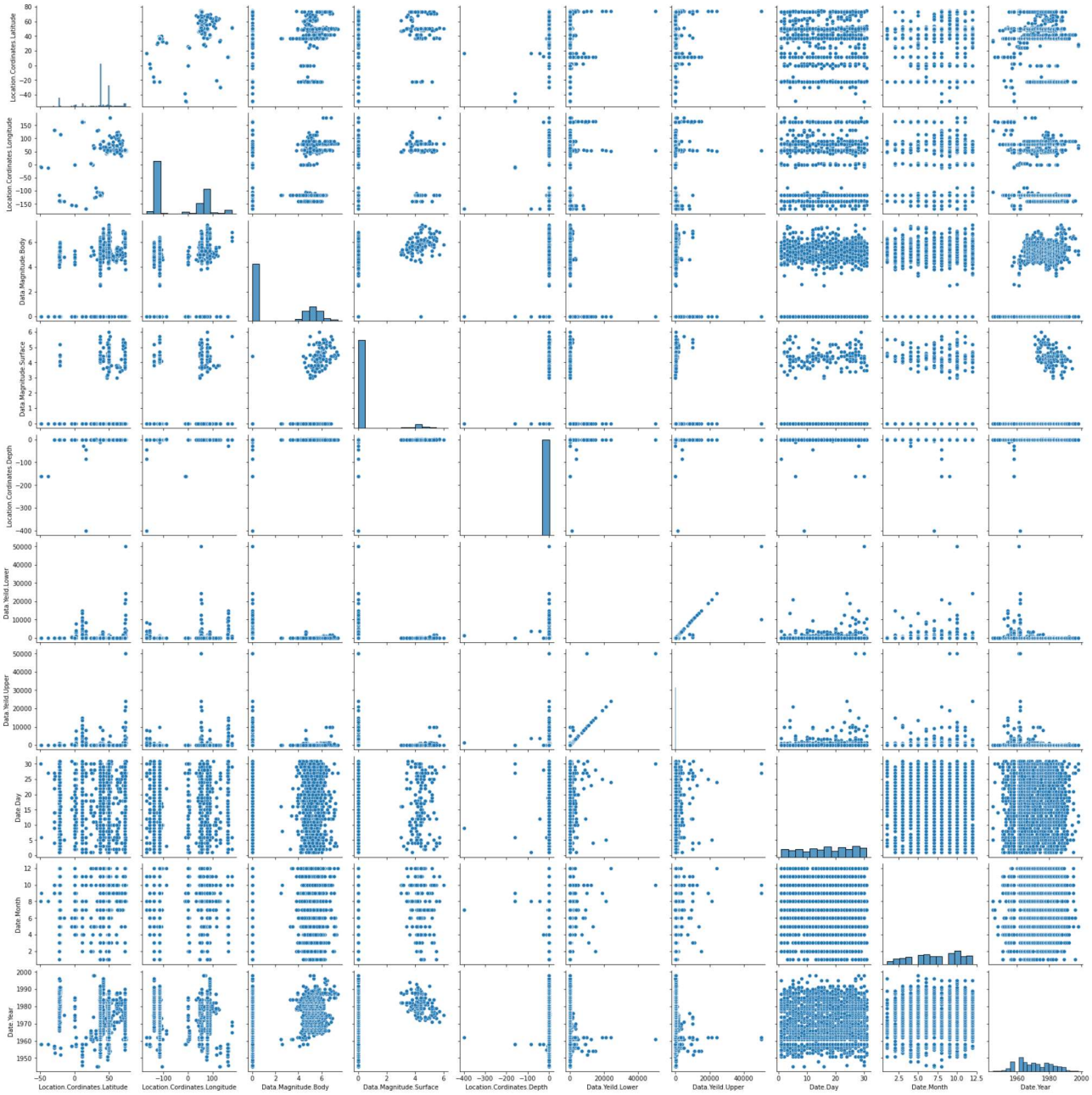
In [6]:
```
#to display colums heading
sd.columns
```

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(sd)
```

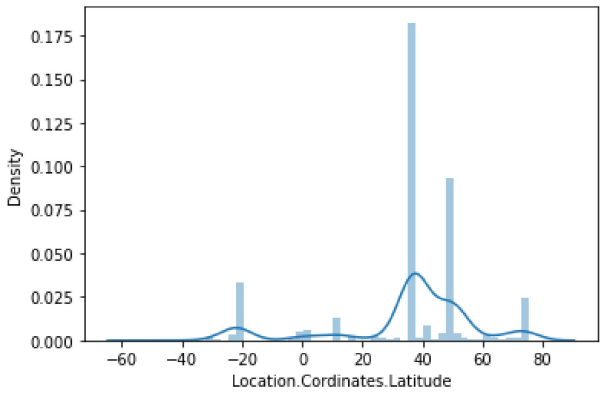
Out[7]: <seaborn.axisgrid.PairGrid at 0x1a86004dfa0>



In [8]:
```
sns.distplot(sd['Location.Cordinates.Latitude'])
```

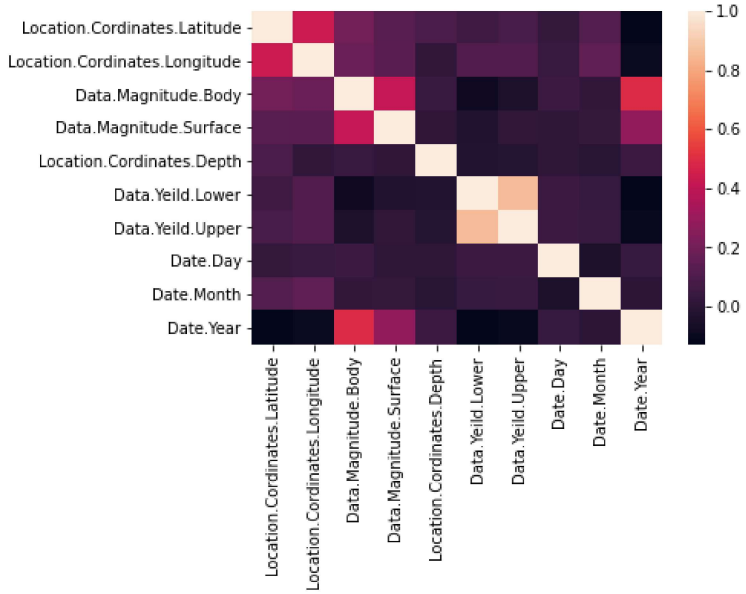
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a depr
ecated function and will be removed in a future version. Please adapt your code to use either `displot` (a fi
gure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[8]: <AxesSubplot:xlabel='Location.Cordinates.Latitude', ylabel='Density'>



In [9]:
```
sns.heatmap(sd.corr())
```

Out[9]: <AxesSubplot:>



In [10]:
```
sd1=sd[['Location.Cordinates.Latitude', 'Location.Cordinates.Longitude', 'Data.Yeild.Lower', 'Data.Yeild.Upper
```

TO TRAIN THE MODEL _MODEL BUILDING

we are goint train Liner Regression model; we need to split out the data into two varibles x and y where x is independent on x (output) and y is dependent on x(output) adress coloumn as it is not required our model

In [13]:
```
x= sd1[['Location.Cordinates.Longitude', 'Data.Yeild.Lower', 'Data.Yeild.Upper']]
y=sd1['Location.Cordinates.Latitude']
```

In [14]:
```
# To split my dataset  into training data 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 [15]:
```
from sklearn.linear_model import LinearRegression

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

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

In [16]:
```
from sklearn.linear_model import LinearRegression

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

Out[16]:
```
LinearRegression()
```

In [17]:
```
print(lr.intercept_)
```

```
38.71728683183676
```

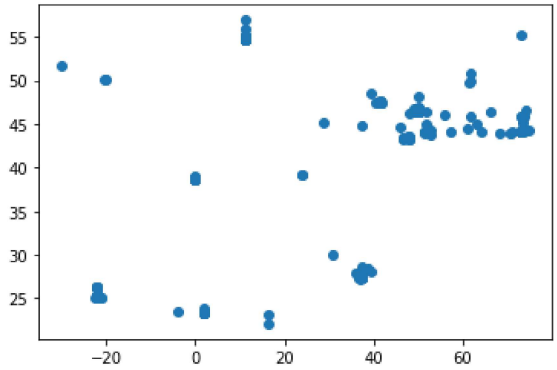
In [18]:
```
coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[18]:

	Co-efficient
Location.Cordinates.Longitude	0.098314
Data.Yeild.Lower	-0.001157
Data.Yeild.Upper	0.001288

In [19]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[19]:
```
<matplotlib.collections.PathCollection at 0x1a873f90b20>
```



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

```
0.20943286601582178
```

In [21]:
```
lr.score(x_train,y_train)
```

Out[21]:
```
0.18568813104547788
```

In [22]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [23]:
```
dr=Ridge(alpha=10)
dr.fit(x_train,y_train)
```

Out[23]:
```
Ridge(alpha=10)
```

In [24]:
```
dr.score(x_test,y_test)
```

Out[24]:
```
0.2094328507649108
```

In [25]:
```
dr.score(x_train,y_train)
```

Out[25]:
```
0.18568813104539394
```

In [26]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[26]:
```
Lasso(alpha=10)
```

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

Out[27]:
```
0.209179393024033
```

In [28]:
```
la.score(x_train,y_train)
```

Out[28]:
```
0.18566968442040788
```

ElasticNet

In [29]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[29]:
```
ElasticNet()
```

In [30]:
```
print(en.coef_)
```

```
[ 0.0982598  -0.00115564  0.00128726]
```

In [31]:
```
print(en.intercept_)
```

```
38.715268020888004
```

In [32]:
```
prediction=en.predict(x_test)
```

In [33]:
```
print(en.score(x_test,y_test))
```

```
0.20941998382571592
```

Evaluation metric

In [34]:
```
from sklearn import metrics
```

In [35]:
```
print("mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
mean Absolute Error: 14.392394029878112
```

In [36]:
```
print("mean squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
mean squared Error: 409.93358369061394
```

In [37]:
```
print("Root mean Absolytre Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root mean Absolytre Error: 20.246816631031503
```

Model Saving

In [38]:
```
import pickle
```

In [39]:
```
filename="prediction"
pickle.dump(lr,open(filename,'wb'))
```

In [ ]: