kaviyadevi 20106064

In [42]:

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

In [43]:

```
#to import dataset
data=pd.read_csv(r"C:\Users\user\Downloads\19_nuclear_explosions - 19_nuclear_exp
data
```

Out[43]:

	WEAPON SOURCE COUNTRY	WEAPON DEPLOYMENT LOCATION	Data.Source	Location.Cordinates.Latitude	Location.Cordinates.Long
0	USA	Alamogordo	DOE	32.54	-1
1	USA	Hiroshima	DOE	34.23	1
2	USA	Nagasaki	DOE	32.45	1
3	USA	Bikini	DOE	11.35	1
4	USA	Bikini	DOE	11.35	1
...	...	...	...	...	
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 [44]:

```
#to display top 5 rows
data.head()
```

Out[44]:

	WEAPON SOURCE COUNTRY	WEAPON DEPLOYMENT LOCATION	Data.Source	Location.Cordinates.Latitude	Location.Cordinates.Longitu
0	USA	Alamogordo	DOE	32.54	-105.
1	USA	Hiroshima	DOE	34.23	132.
2	USA	Nagasaki	DOE	32.45	129.
3	USA	Bikini	DOE	11.35	165.
4	USA	Bikini	DOE	11.35	165.

DATA CLEANING AND PREPROCESSING

In [45]:

```
#
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
```

In [46]:

```
#to display summary of statistics(here to know min max value)
data.describe()
```

Out[46]:

	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	

In [47]:

```
#to display the column heading
data.columns
```

Out[47]:

```
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')
```

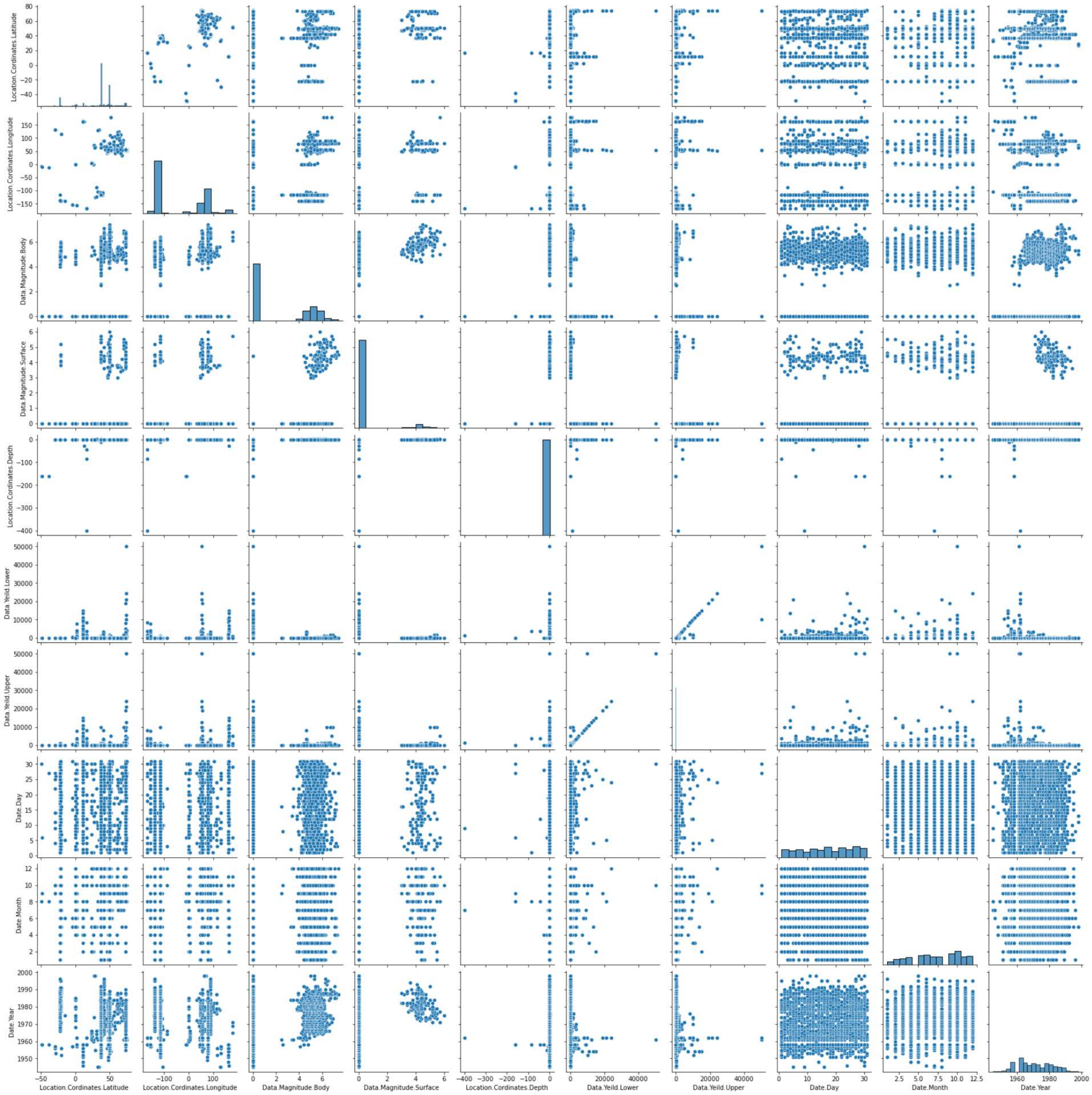
In [48]:

```
#here there is no missing values (identified through info() 5000 data are describ
```

EDA and DATA VISUALIZATION

In [49]:
```
sns.pairplot(data)
```

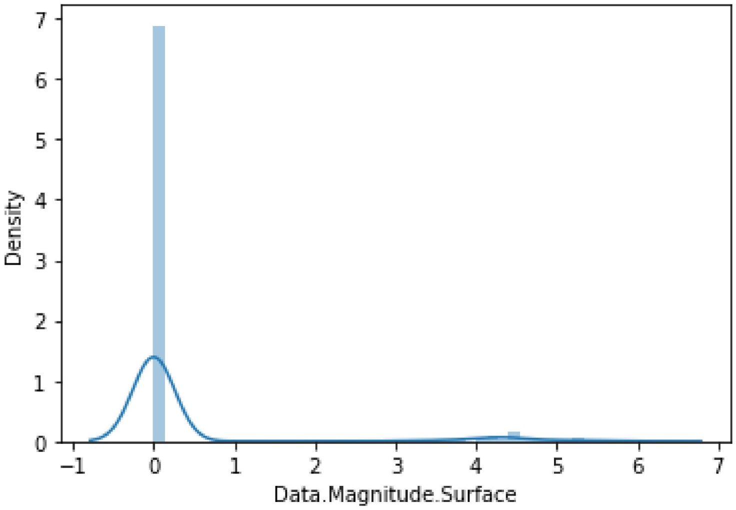
Out[49]:
```
<seaborn.axisgrid.PairGrid at 0x2391fe1cd30>
```



In [52]:
```
sns.distplot(data['Data.Magnitude.Surface'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

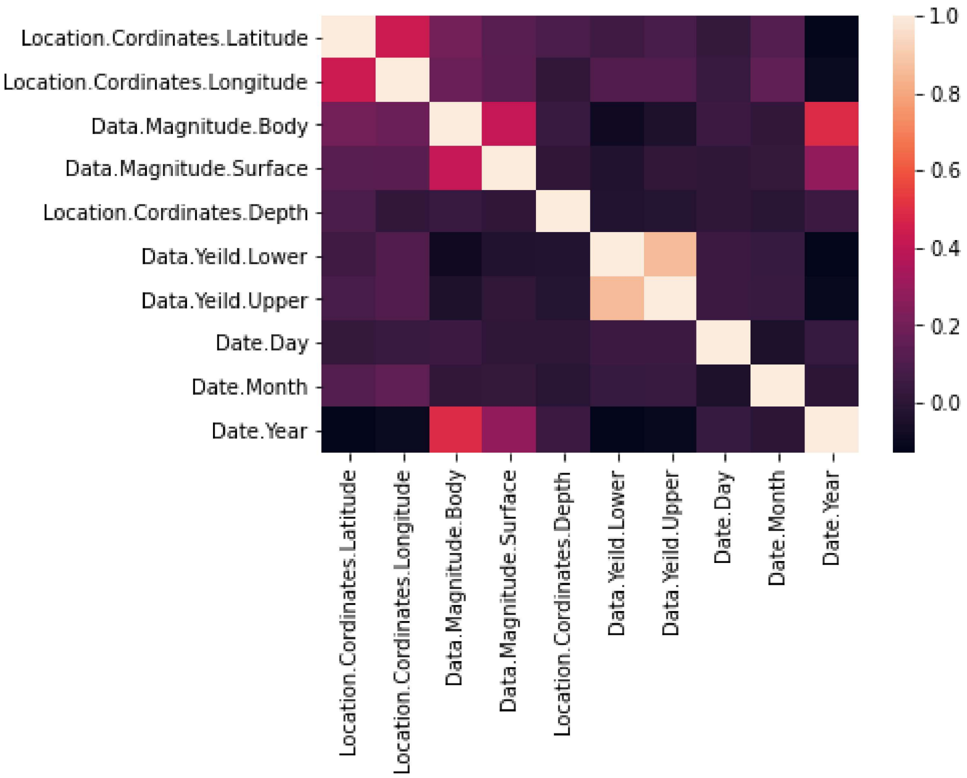
Out[52]:
```
<AxesSubplot:xlabel='Data.Magnitude.Surface', ylabel='Density'>
```



In [53]:
```
df=data[['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 [54]:
```
sns.heatmap(df.corr())
```

Out[54]:
```
<AxesSubplot:>
```



TO TRAIN MODEL

MODEL BUILDING We are going to train linear regression model; we need to split out the data into two variables x and y where x is independent variables (input) and y is dependent on x(output) we could ignore address column as it is not required for our model

In [17]:
```
x=df[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population']]
y=df['Price']
```

In [18]:
```
#to split my dataset into trainning and test

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 [19]:
```
from sklearn.linear_model import LinearRegression

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

Out[19]:
```
LinearRegression()
```

In [20]:
```
#to find intercept
print(lr.intercept_)
```

```
-2631179.446847313
```

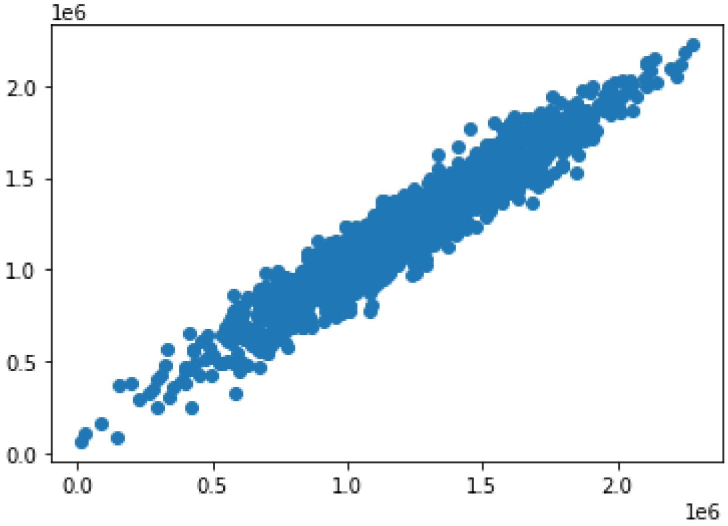
In [21]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[21]:

	Co-efficient
Avg. Area Income	21.479593
Avg. Area House Age	165312.826052
Avg. Area Number of Rooms	121223.545008
Avg. Area Number of Bedrooms	2293.701818
Area Population	15.118977

```
In [22]: prediction = lr.predict(x_test)
         plt.scatter(y_test,prediction)

Out[22]: <matplotlib.collections.PathCollection at 0x23922fedbb0>
```



```
In [23]: print(lr.score(x_test,y_test))

         0.9196122061704285
```

RIDGE AND LASSO REGRESSION

```
In [24]: from sklearn.linear_model import Ridge,Lasso

In [25]: rr=Ridge(alpha=10)
         rr.fit(x_train,y_train)

Out[25]: Ridge(alpha=10)

In [26]: rr.score(x_test,y_test)

Out[26]: 0.9196108618574063

In [27]: la=Lasso(alpha=10)
         la.fit(x_train,y_train)

Out[27]: Lasso(alpha=10)

In [28]: la.score(x_test,y_test)

Out[28]: 0.9196126427939018
```

In [29]:

```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[29]:

```
ElasticNet()
```

In [31]:

```
print(en.coef_)
```

```
[2.13485855e+01 1.08956510e+05 7.60150692e+04 1.48830865e+04
 1.49596622e+01]
```

In [33]:

```
print(en.predict(x_test))
```

```
[1233826.38851369 1039013.51432593 1449245.88937301 ... 1230827.74917279
 1120198.45040024 1159902.0622341 ]
```

In [34]:

```
print(en.score(x_test,y_test))
```

```
0.8832134954458345
```

EVALUATION METRICS

In [35]:

```
from sklearn import metrics
```

In [37]:

```
print("Mean Absolute error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute error 82122.1782559458
```

In [38]:

```
print("Mean Squared error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared error 10434724665.834866
```

In [41]:

```
print("Root Mean Absolute error",np.sqrt(metrics.mean_squared_error(y_test,predic
```

```
Root Mean Absolute error 102150.50007628385
```

MODEL SAVING

In [58]:

```
import pickle
```

In [59]:

```
filename='predict1'
pickle.dump(lr,open(filename,'wb'))
```