Problem statement

predicting the house price in USA.To create a model to help him estimate of what the house would sell for.

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

In [2]:
```
1 df=pd.read_csv("nuclear_explosions")
```

To display top 10 rows

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

Out[3]:

	WEAPON SOURCE COUNTRY	WEAPON DEPLOYMENT LOCATION	Data.Source	Location.Cordinates.Latitude	Location.Cordinates.Longitude	Data.Magnitude.Body	Data.Magnit
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]:
```
1 df.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]:
```
# Display the statistical summary
df.describe()
```

Out[5]:

	Location.Cordinates.Latitude	Location.Cordinates.Longitude	Data.Magnitude.Body	Data.Magnitude.Surface	Location.Cordinates.Dept
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 the col headings
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]:
```
cols=df.dropna(axis=1)
```

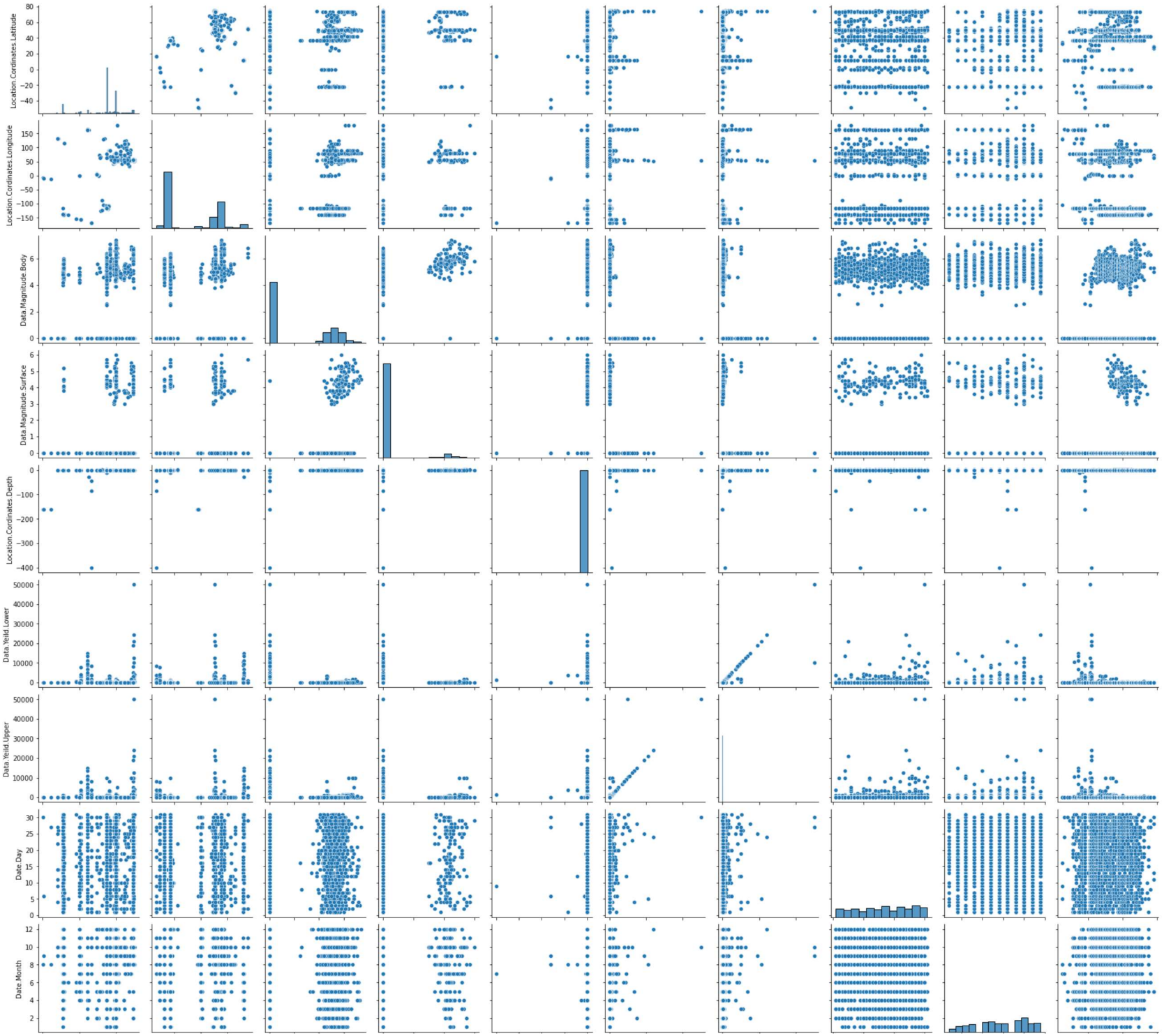
In [8]:
```
1 cols.columns
```

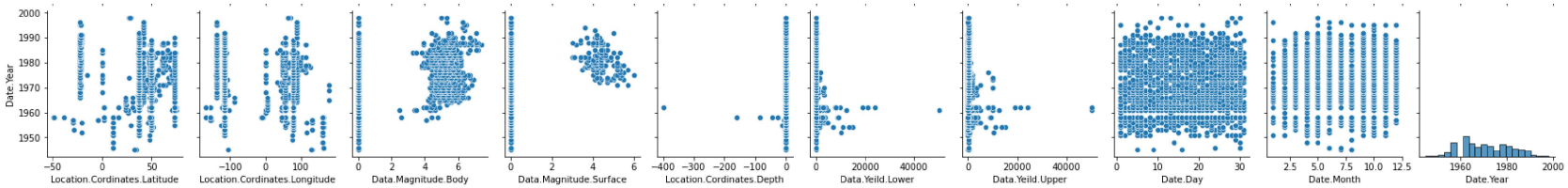
Out[8]:
```
Index(['WEAPON SOURCE COUNTRY', 'WEAPON DEPLOYMENT LOCATION', 'Data.Source',
       'Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
       'Data.Magnitude.Body', 'Data.Magnitude.Surface',
       'Location.Cordinates.Depth', 'Data.Yeild.Lower', 'Data.Yeild.Upper',
       'Data.Purpose', 'Data.Name', 'Data.Type', 'Date.Day', 'Date.Month',
       'Date.Year'],
      dtype='object')
```

EDA and Visualization

In [9]:
```
1 sns.pairplot(cols)
```

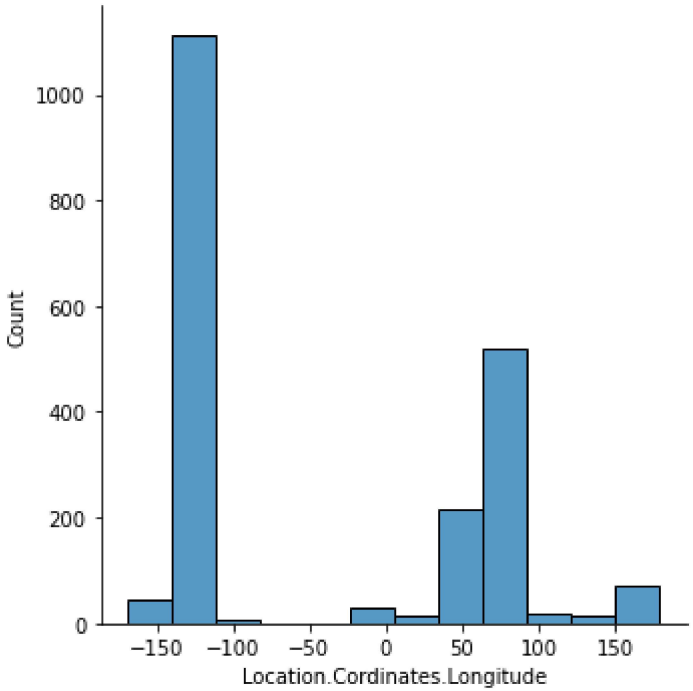
Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x25c48de2be0>
```


Location.Cordinates.Latitude
Location.Cordinates.Longitude
Data.Magnitude.Body
Data.Magnitude.Surface
Location.Cordinates.Depth
Data.Yeild.Lower
Data.Yeild.Upper
Date.Day
Date.Month



In [11]:
```
1 sns.displot(df['Location.Cordinates.Longitude'])
```

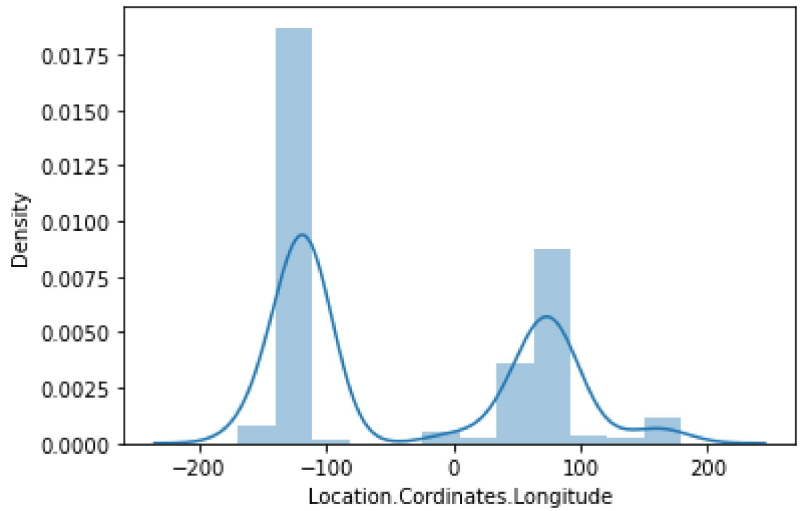
Out[11]: <seaborn.axisgrid.FacetGrid at 0x25c5b96efa0>



In [12]:
```
# We use displot in older version we get distplot use displot
sns.distplot(df['Location.Cordinates.Longitude'])
```

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

Out[12]: <AxesSubplot:xlabel='Location.Cordinates.Longitude', ylabel='Density'>



In [14]:

```
df1=cols[['Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
       'Data.Magnitude.Body', 'Data.Magnitude.Surface',
       'Location.Cordinates.Depth']]
df1
```

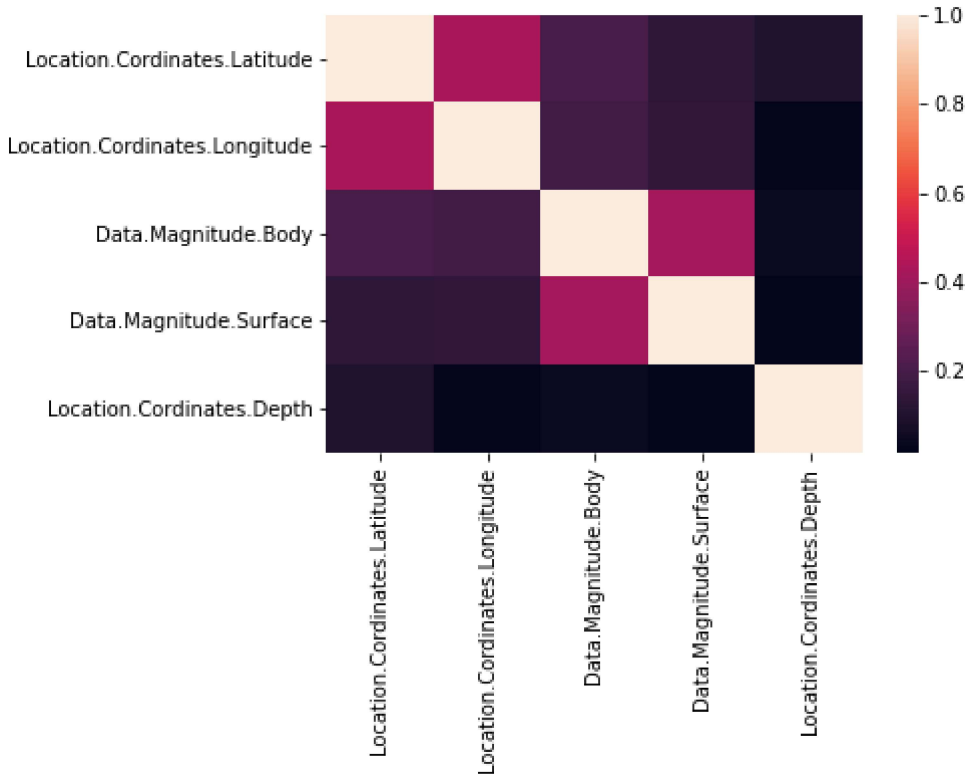
Out[14]:

	Location.Cordinates.Latitude	Location.Cordinates.Longitude	Data.Magnitude.Body	Data.Magnitude.Surface	Location.Cordinates.Depth
0	32.54	-105.57	0.0	0.0	-0.1(
1	34.23	132.27	0.0	0.0	-0.6(
2	32.45	129.52	0.0	0.0	-0.6(
3	11.35	165.20	0.0	0.0	-0.2(
4	11.35	165.20	0.0	0.0	0.0
...	...	...	...	...	..
2041	41.69	88.35	5.3	0.0	0.0(
2042	27.07	71.70	5.3	0.0	0.0(
2043	27.07	71.70	0.0	0.0	0.0(
2044	28.90	64.89	0.0	0.0	0.0(
2045	28.49	63.78	5.0	0.0	0.0(

2046 rows × 5 columns

In [15]:
```
1 sns.heatmap(df1.corr())
```

Out[15]: <AxesSubplot:>



To train the model - MODEL BUILD

Going to train linear regression model;We split our data into 2 variables x and y where x is independent var(input) and y is dependent on x(output), we could ignore address col as it is not required for our model

In [42]:
```
1 x=df1[['Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
2        'Data.Magnitude.Body', 'Data.Magnitude.Surface',
3        'Location.Cordinates.Depth']]
4 y=df1[['Location.Cordinates.Depth']]
```

To split the dataset into test data

In [43]:
```
# importing lib for splitting test data
from sklearn.model_selection import train_test_split
```

In [44]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [45]:
```
from sklearn.linear_model import LinearRegression

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

Out[45]:
```
LinearRegression()
```

In [46]:
```
print(lr.intercept_)
```

```
[2.83106871e-15]
```

In [47]:
```
print(lr.score(x_test,y_test))
```

```
1.0
```

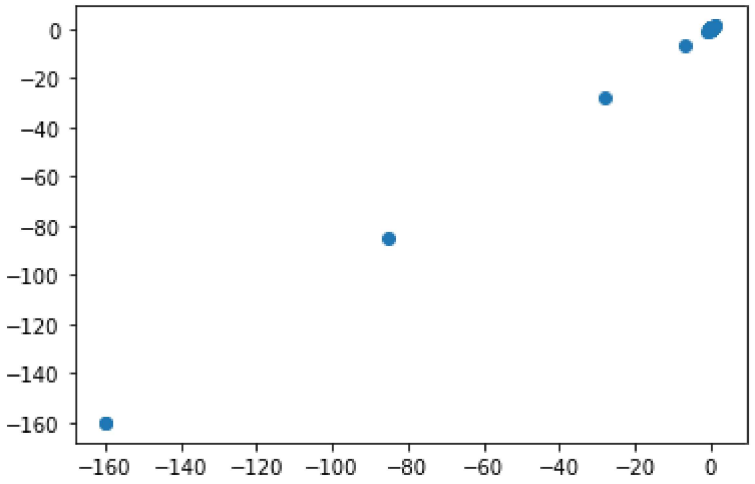
In [48]:
```
coeff=pd.DataFrame(lr.coef_)
coeff
```

Out[48]:

	0	1	2	3	4
0	-1.118438e-16	1.353222e-17	7.115108e-16	1.560122e-16	1.0

In [49]:
```
1 pred = lr.predict(x_test)
2 plt.scatter(y_test,pred)
```

Out[49]: <matplotlib.collections.PathCollection at 0x25c5ae66f70>



In [50]:
```
1 from sklearn.linear_model import Ridge,Lasso
```

In [51]:
```
1 rr=Ridge(alpha=10)
2 rr.fit(x_train,y_train)
```

Out[51]: Ridge(alpha=10)

In [52]:
```
1 rr.score(x_test,y_test)
```

Out[52]: 0.9999999971947974

In [53]:
```
1 la=Lasso(alpha=10)
2 la.fit(x_train,y_train)
```

Out[53]: Lasso(alpha=10)

In [54]:
```
1 la.score(x_test,y_test)
```

Out[54]: 0.99416709843095

ELASTIC NET

In [55]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[55]:
```
ElasticNet()
```

In [56]:
```
print(en.coef_)
```

```
[0.         0.         0.         0.         0.99239503]
```

In [57]:
```
print(en.intercept_)
```

```
[-0.00307039]
```

In [58]:
```
prediction=en.predict(x_test)
prediction
```

```
-3.07039470e-03, -3.07039470e-03, -3.07039470e-03, -3.07039470e-03,
-3.07039470e-03, -4.06278973e-03, -3.07039470e-03, -3.07039470e-03,
-3.07039470e-03, -3.07039470e-03, -3.07039470e-03, -3.07039470e-03,
 4.65340060e-01, -3.07039470e-03, -3.07039470e-03, -3.07039470e-03,
-3.07039470e-03, -3.07039470e-03, -3.07039470e-03, -3.07039470e-03,
-3.07039470e-03, -3.07039470e-03, -3.07039470e-03, -3.07039470e-03,
-4.06278973e-03, -3.07039470e-03,  5.34807712e-01, -3.07039470e-03,
-3.07039470e-03, -3.07039470e-03, -3.07039470e-03, -3.07039470e-03,
 5.92366624e-01, -2.77901313e+01, -3.07039470e-03, -3.07039470e-03,
 3.17473200e-01, -3.07039470e-03, -3.07039470e-03, -3.07039470e-03,
 2.11286932e-01, -3.07039470e-03, -3.07039470e-03, -3.07039470e-03,
-3.07039470e-03, -3.07039470e-03, -3.07039470e-03, -3.07039470e-03,
-3.07039470e-03, -3.07039470e-03, -3.07039470e-03, -3.07039470e-03,
-2.51169152e-01, -3.07039470e-03, -3.07039470e-03, -3.07039470e-03,
-3.07039470e-03,  1.95408611e-01, -3.07039470e-03, -3.07039470e-03,
-3.07039470e-03, -3.07039470e-03, -3.07039470e-03, -3.07039470e-03,
-2.01549401e-01, -3.07039470e-03,  6.92598522e-01, -3.07039470e-03,
-1.02309898e-01, -3.07039470e-03, -3.07039470e-03, -3.07039470e-03,
-3.07039470e-03, -3.07039470e-03, -3.07039470e-03, -4.06278973e-03,
```

In [59]:
```
1 print(en.score(x_test,y_test))
```

```
0.9999421137312923
```

EVALUATION METRICS

In [60]:
```
1 from sklearn import metrics
```

In [61]:
```
1 print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 0.008667694123444624
```

In [62]:
```
1 print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 0.005560099889522486
```

In [63]:
```
1 print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root Mean Squared Error: 0.07456607733763716
```

MODEL SAVING

In [64]:
```
1 import pickle
```

In [65]:
```
1 filename='prediction01'
2 pickle.dump(lr,open(filename,'wb'))
```

In [ ]:
```
1
```