Problem Statement

A real estate agent want help to predict the house price for regions in USA.He gave us the dataset to work on to use linear regression model.Create a model that helps him to estimate of what the house would sell for

Import libraries ¶

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

In [2]:
```
# To import dataset
df=pd.read_csv('19_nuclear_explosions.csv')
df
```

Out[2]:

	WEAPON SOURCE COUNTRY	WEAPON DEPLOYMENT LOCATION	Data.Source	Location.Cordinates.Latitude	Location.Cordinates.Lc
0	USA	Alamogordo	DOE	32.54	
1	USA	Hiroshima	DOE	34.23	
2	USA	Nagasaki	DOE	32.45	
3	USA	Bikini	DOE	11.35	
4	USA	Bikini	DOE	11.35	
...	...	...	...	...	
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 [3]:
```
# To display top 10 rows
df.head(10)
```

Out[3]:

	WEAPON SOURCE COUNTRY	WEAPON DEPLOYMENT LOCATION	Data.Source	Location.Cordinates.Latitude	Location.Cordinates.Longi
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
5	USA	Enewetak	DOE	11.30	16
6	USA	Enewetak	DOE	11.30	16
7	USA	Enewetak	DOE	11.30	16
8	USSR	Semi Kazakh	DOE	48.00	7
9	USA	Nts	DOE	37.00	-11

Data Cleaning and Pre-Processing

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

Out[5]:

	Location.Cordinates.Latitude	Location.Cordinates.Longitude	Data.Magnitude.Body	Data.Mag
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 [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')
```

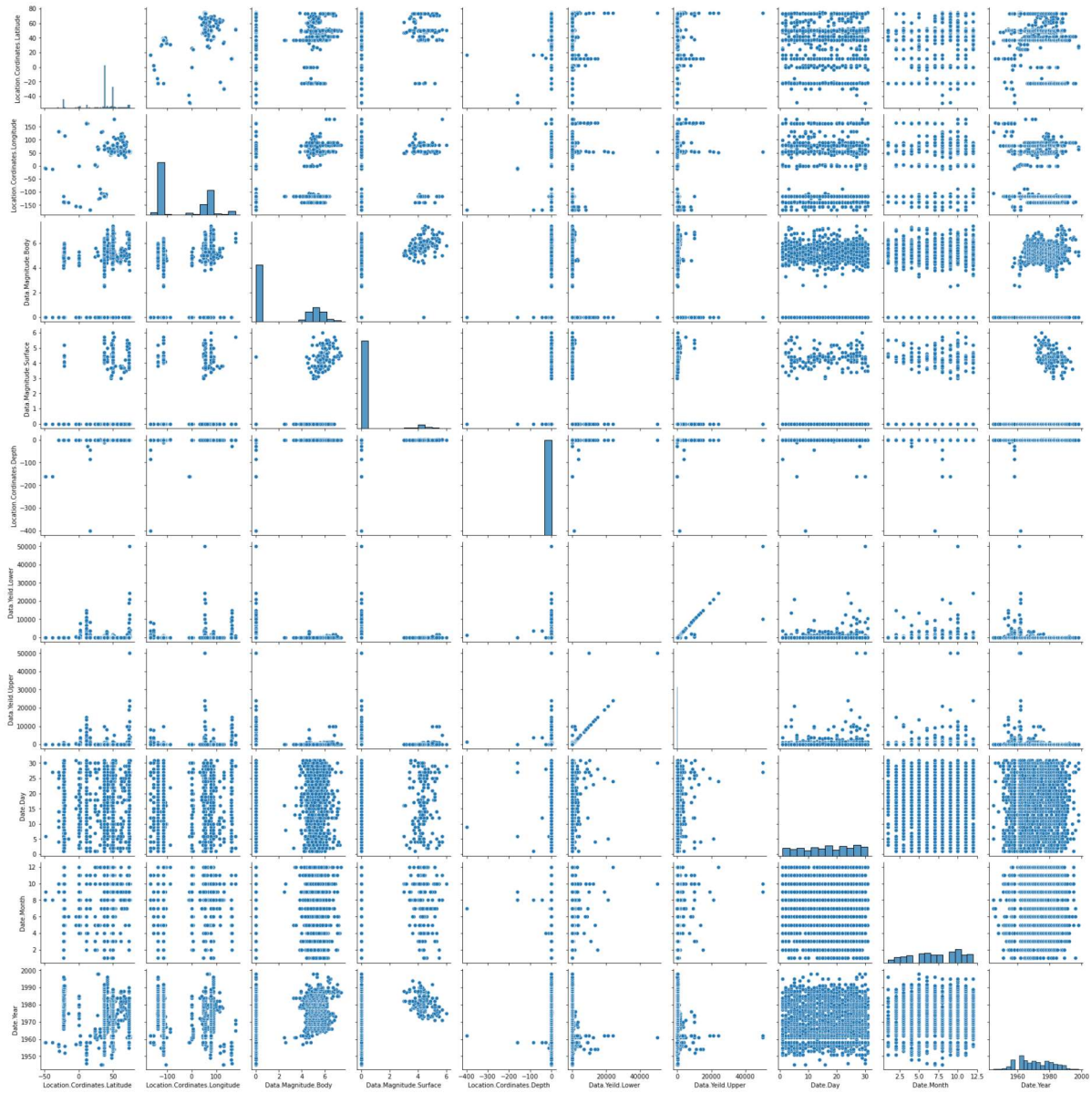
In [7]:
```
a = df.dropna(axis='columns')
a.columns
```

Out[7]:
```
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 [8]:
```
sns.pairplot(a)
```

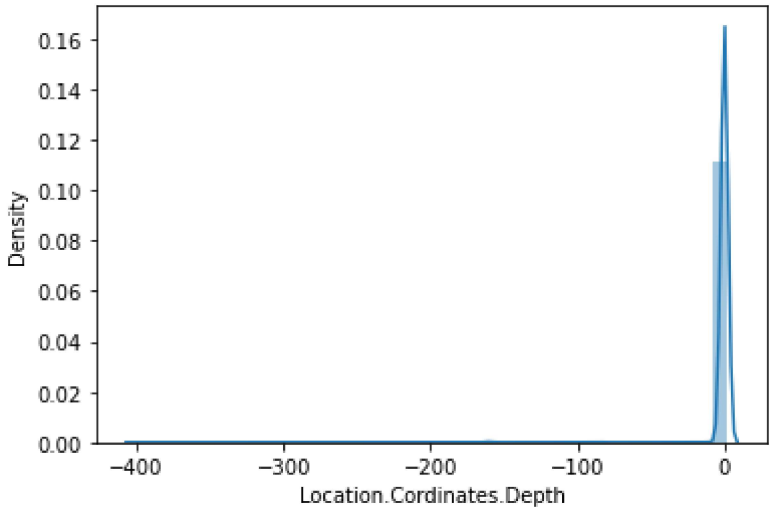
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x2198b8551c0>
```



In [9]:
```
sns.distplot(a['Location.Cordinates.Depth'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

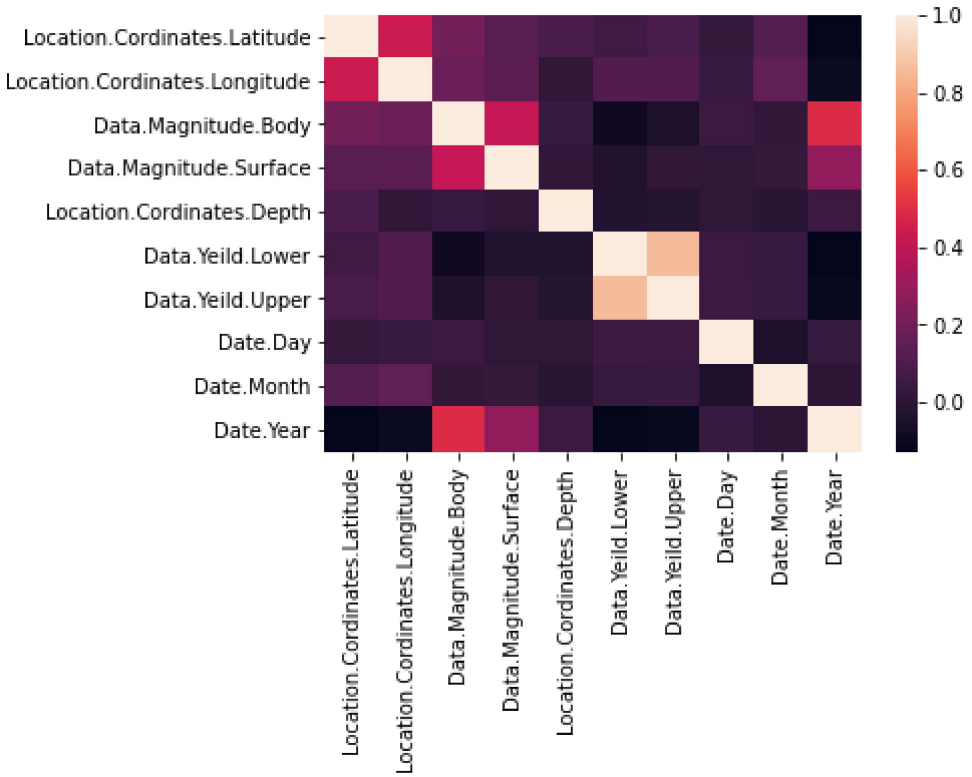
Out[9]:
```
<AxesSubplot:xlabel='Location.Cordinates.Depth', ylabel='Density'>
```



In [10]:
```
a1=a[[ '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', 'Date.Year']]
```

In [11]:
```
sns.heatmap(a1.corr())
```

Out[11]: <AxesSubplot:>



To Train the Model - Model Building

We are going to train Linear Regression model;We need to split out data into two variables x and y where x is independent variable (input) and y is dependent on x(output). We could ignore address column as it is not required for our model.

In [12]:
```
x=a1[['Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
       'Data.Magnitude.Body', 'Data.Magnitude.Surface']]
y=a1['Location.Cordinates.Depth']
```

To split my dataset into training and test data

In [13]:
```
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 [14]:
```
from sklearn.linear_model import LinearRegression

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

Out[14]:
```
LinearRegression()
```

In [15]:
```
print(lr.intercept_)
```

```
-2.1359414611190086
```

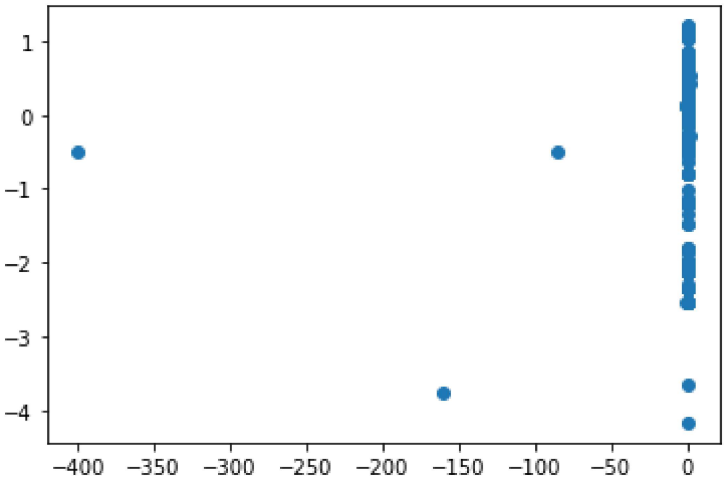
In [16]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
Location.Cordinates.Latitude	0.043881
Location.Cordinates.Longitude	-0.005459
Data.Magnitude.Body	0.067554
Data.Magnitude.Surface	-0.026241

In [17]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[17]:
```
<matplotlib.collections.PathCollection at 0x2199b589250>
```



In [18]:
```
print(lr.score(x_test,y_test))
```

```
0.002741424271516024
```

In [19]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [20]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[20]:
```
Ridge(alpha=10)
```

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

Out[21]: 0.025222225811931565

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

Out[22]: 0.0027408269045537947

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

Out[23]: 0.0027408269045537947

In [24]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[24]: Lasso(alpha=10)

In [25]:
```
la.score(x_test,y_test)
```

Out[25]: 0.0011032527515616453

In [26]:
```
from sklearn.linear_model import ElasticNet
en = ElasticNet()
en.fit(x_train,y_train)
```

Out[26]: ElasticNet()

In [27]:
```
print(en.coef_)
```

```
[ 0.04362695 -0.00506922  0.          0.        ]
```

In [28]:
```
print(en.intercept_)
```

```
-1.9791682961607546
```

In [29]:
```
print(en.predict(x_test))
```

```
[-0.19687873  0.22718974 -0.1932197   0.22305854  0.38183335  0.22390035
  0.2335588  -0.20340281 -0.1233278  -0.17557212  0.22141485 -0.20295894
  0.92679232 -2.23433974 -2.32143749 -0.20216782  0.22801865  0.22305854
  0.2345229   0.22213531 -0.1932197   0.13330881  0.22441805  1.00979125
  0.4860621  -2.23433974  0.22424446  0.22305854 -0.19721761 -0.19562752
 -0.20649979  0.22305854 -0.1932197   0.97081848  0.22713705  0.96929771
 -2.30815771  0.23702142  0.22811097 -0.20245202  0.22206466 -0.20519221
  0.22305854  0.2355011  -2.22869542 -0.00883569 -2.23433974 -0.1932197
  0.22792816  0.23527251  0.57791941 -0.60522892 -0.20326724 -2.2300623
 -2.30815771  0.22980536  0.22305854 -1.09604686 -0.20205123  0.99668733
  0.0827644   0.23661788  0.22926124  0.22305854  0.22305854 -0.1932197
  0.22305854  0.22305854  0.22582824  0.23279778  0.2295313  -2.22925965
  0.22882114  0.22642289  0.22305854 -0.20034132  0.22305854  0.94688543
 -0.1932197  -0.20317223  0.22305854  0.22918628 -0.1932197   0.22305854
 -2.23433974  0.22661935  0.22305854  0.23159099  0.22305854 -2.23433974
  0.23499845  0.22305854 -0.19932916 -2.2358605   0.22632227  0.92679232
  0.22400174  0.22305854 -2.23017458 -0.19332109  0.22305854 -3.44748995
  0.92679232 -0.1932197   0.22305854  0.92679232  0.22305854 -0.1932197
  0.22816166 -0.1932197   0.22305854  1.00561232 -0.1932197   0.22290431
  0.22305854 -0.1932197  -2.23433974 -0.1233278  -2.30815771  0.22792816
  0.23113935 -0.11997895  0.22287574  0.95378273 -0.1932197   0.29605686
 -0.61981024  0.22305854 -1.09604686  0.22391388 -0.2019112   0.22792816
 -2.23433974 -0.19169894 -0.20589148 -2.23433974  0.22305854  0.22305854
 -0.61981024  0.22305854 -0.1932197   0.22762401 -0.20198502 -0.17557212
  0.22305854  0.22305854  0.22742124  0.23279978  0.22305854  0.22880178
 -0.2028567  -0.19981472 -0.13811629  0.2371497   0.22305854  0.23666274
 -0.1932197   0.22305854  0.22800959 -1.09604686  0.92679232 -0.27033517
  0.22742124 -0.2029865   0.22473234  0.22305854 -0.1932197  -0.11825858
  0.2376674  -1.16164969 -2.23433974  0.22305854 -0.20423274 -0.19230724
 -0.1932197   0.9938454   0.94548595  0.22305854 -0.20032897  0.92679232
 -2.30815771 -2.23433974  0.22636589 -2.23433974  0.23279778  0.45781267
  0.23772886  0.22854724  0.22305854  0.22800959  0.22305854 -0.1932197
  0.22670078  0.22703566  0.22305854  0.22646728 -1.36794703  0.22305854
  0.23884409  0.22742124  0.22869932  0.22305854 -0.13560006  0.22305854
 -2.30815771  0.234196   -3.60051003 -0.1932197   0.92679232 -0.197782
  0.01365236  0.22862543 -1.36794703 -1.09604686  0.94416009 -1.09604686
 -0.20124143  0.15915471 -0.1932197   0.92679232  0.23596363 -0.19641173
 -2.32143749  0.22534128  0.22786442  0.22305854 -2.30815771 -2.22853612
 -0.20376083 -2.23433974 -0.1932197   0.92679232  0.22305854  0.23281299
 -2.22823196 -0.19510609  0.22675869  0.22305854 -0.20732163  0.22305854
  0.96341425 -2.32143749 -2.23013526 -0.19926285 -0.1932197   0.23229086
  0.23597301  0.22742124  0.22883143  0.98613385 -0.19701401 -0.19937893
  0.22305854 -0.20004636  0.22877567  0.22403247 -0.1932197  -2.22889866
 -2.23433974 -0.1932197   0.22305854 -0.1932197  -0.19903251 -2.30815771
  0.10003508 -1.09604686 -1.9791683   0.22305854 -0.20468406  0.22603516
  0.22305854  0.22905448 -2.23433974  0.22742124 -0.1233278   0.63910789
 -0.11825858 -0.20566489 -0.1932197   0.22305854  0.22305854 -0.1932197
  0.22305854 -2.30815771  0.22305854  0.22228739  0.22305854 -0.1932197
  0.21053884  0.22291566  0.23229086 -0.61981024  0.22408316  0.22305854
 -0.1932197   0.22305854 -0.20380835  0.22305854  0.22394027  0.22305854
  0.21594612  0.92679232  0.92679232  0.22305854 -0.19169894  0.23645826
  0.22305854 -0.13123895  0.22305854 -0.1932197   0.22589889  0.22305854
 -0.61716224  0.22792816 -0.19176959 -2.23433974  0.2276747  -0.1932197
 -2.226152   -0.01962841  0.22305854  0.22305854  0.21945623 -0.19271278
  0.22792816  0.22913559  0.23221941  0.23143828  0.23823607  0.23995013
  0.20453198 -1.9791683   0.22305854  0.22403323  0.22305854  0.2376674
 -0.19964842 -0.1149985  -0.1932197  -0.20394363 -0.2007852   0.94341048
```

```
 0.22305854  0.22305854  0.22305854 -0.20345351 -0.20257336 -2.23433974
-0.40318457  0.21798932  0.22742124 -0.1932197   0.22383337  1.00124963
-1.9791683   0.22305854  0.22305854  0.22305854  0.94453445 -0.63608675
-0.40318457 -0.20732163 -2.23433974 -0.19962086  0.22331201  0.22305854
 0.22305854  0.22305854 -1.9791683  -2.23433974  0.22364689 -0.1932197
 0.22463     0.22305854  0.22742124 -0.1932197   0.22913559 -2.23433974
 0.22282504 -2.23433974  0.22021662 -0.61097843  0.07451737 -0.20295894
-0.18255432  0.22695424 -1.9791683   0.22742124  0.94299417 -1.9791683
-0.33347226  0.22768899  0.22305854  0.22742124  0.22305854 -0.1655417
-0.20411032  0.22305854  0.22795889  0.22305854  0.22305854 -0.1932197
 0.22254085  0.2263152   0.23328474  0.94282261  0.22305854  0.17831051
-0.18242943  0.22123204  0.80683307  0.06529779  0.83405415  0.23034301
-0.1932197  -0.1932197  -0.61981024 -0.1932197   0.22596035 -2.23433974
-1.09604686 -0.63680721 -2.32143749  0.21798932 -2.30815771 -0.19828892
-0.19220586  0.92679232  0.23184382  0.23676828 -0.6402774   0.92141578
 0.22305854 -0.23684666  0.92679232 -0.61981024 -1.16906628  0.22305854
 0.2265487   0.22742124 -0.20004794  0.22305854 -0.1932197   0.22305854
-2.30815771 -0.1932197  -1.9791683  -0.1932197   0.22873987 -2.30815771
-0.20321788  0.22305854 -2.223433   -0.23684666  0.22238877  0.2242552
 0.22305854  0.22305854 -2.30815771  0.22305854 -0.1233278   0.92679232
 0.2304891  -0.64464371 -0.1932197   0.23431855 -2.23433974 -0.20132602
 0.22305854  0.22305854  0.2229865   0.22908673 -0.1975824  -0.1932197
-0.1932197  -0.20161021  0.22875001  0.22305854  0.22305854 -0.1932197
-1.04729098  0.94475002 -0.1932197  -0.20194193 -0.13870464  0.22792816
-0.20132285 -0.1932197  -1.9791683  -0.20340598  0.23663708  0.92679232
-2.23433974  0.22305854  0.22779253 -0.19419046 -0.20111248  0.22551163
 0.22930762  0.22698497  0.22305854  0.22860807 -0.1932197   0.22699511
 0.22725671  0.23675     0.22893282  0.94592513  0.22792816  0.22305854
 0.2369162  -0.95746748  0.22305854  0.22305854  0.22403247 -0.61982259
-2.22779569 -2.23433974  0.92679232 -0.19606163 -0.20436622  0.23558743
 0.22305854  0.22305854  0.22792816  0.22920564  0.22864862  0.23868124
-2.23433974 -0.18152647  0.22305854 -1.9791683   0.22305854 -0.20047344
 0.23100863 -2.23433974  0.92679232 -0.20245202  0.22438871 -0.19946986
-2.23014908  0.22305854  0.23246289 -1.9791683  -0.40318457 -0.1932197
 0.23817432  0.22305854 -0.1932197  -2.32143749  0.22305854 -2.22674741
 0.2326042   0.22305854 -0.20180221 -0.2009604  -1.09604686 -0.40318457
 0.22305854 -0.20055486 -0.61981024  0.23279778  0.24519639  0.92679232
 0.97903695  0.22305854  0.22305854 -2.19071278 -0.20178415 -0.19423513
 0.22742124 -2.24356128 -0.2052757   0.22682352  0.22792816  0.22305854
 0.22305854  0.22604637  0.22305854 -0.1932197   0.9459359  -1.9791683
-0.11825858  0.22305854 -2.22882969  0.22305854  0.22305854 -0.1932197
 0.22305854 -0.11825858  0.23229086 -0.1932197  -1.9791683   0.23843855
 0.92679232  0.2272077  -3.95204475 -0.1932197  -2.32143749 -0.61204958
-2.22075392  0.22305854 -2.23433974  0.23327967  0.22742124 -0.20586075
-0.19828892 -0.20024295 -0.1932197   0.21918281 -2.30815771  0.22716024
 0.22305854 -2.23433974]
```

In [30]:
```
print(en.score(x_test,y_test))
```

```
0.002212907095477612
```

Evaluation Metrics

In [31]:
```
from sklearn import metrics
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,pre
```

```
Mean Absolytre Error: 1.604642314275139
Mean Squared Error: 312.1906187180114
Root Mean Squared Error: 17.668916738668827
```

In [32]:
```
import pickle
```

In [34]:
```
filename='prediction1'
pickle.dump(lr,open(filename,'wb'))
```

In [ ]:
```
```