Type *Markdown* and LaTeX: α^2

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

In [2]:
```
#import dataset
df=pd.read_csv(r"E:\154\drive-download-20230731T110444Z-001\19_nuclear_explosions.csv
df
```

Out[2]:

	WEAPON SOURCE COUNTRY	WEAPON DEPLOYMENT LOCATION	Data.Source	Location.Cordinates.Latitude	Location.Cordinates.Longitude
0	USA	Alamogordo	DOE	32.54	-105.57
1	USA	Hiroshima	DOE	34.23	132.27
2	USA	Nagasaki	DOE	32.45	129.52
3	USA	Bikini	DOE	11.35	165.20
4	USA	Bikini	DOE	11.35	165.20
...	...	...	...	...	...
2041	CHINA	Lop Nor	HFS	41.69	88.35
2042	INDIA	Pokhran	HFS	27.07	71.70
2043	INDIA	Pokhran	NRD	27.07	71.70
2044	PAKIST	Chagai	HFS	28.90	64.89
2045	PAKIST	Kharan	HFS	28.49	63.78

2046 rows × 16 columns

In [3]:
```
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 [4]:
```
#to display top 5 rows
df.head()
```

Out[4]:

	WEAPON SOURCE COUNTRY	WEAPON DEPLOYMENT LOCATION	Data.Source	Location.Cordinates.Latitude	Location.Cordinates.Longitude	Da
0	USA	Alamogordo	DOE	32.54	-105.57	
1	USA	Hiroshima	DOE	34.23	132.27	
2	USA	Nagasaki	DOE	32.45	129.52	
3	USA	Bikini	DOE	11.35	165.20	
4	USA	Bikini	DOE	11.35	165.20	

Data cleaning and Pre-Processing

In [5]:
```
#To find null values
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 [6]:
```
# To display summary of statistics
df.describe()
```

Out[6]:

	Location.Cordinates.Latitude	Location.Cordinates.Longitude	Data.Magnitude.Body	Data.Magnitude.Su
count	2046.000000	2046.000000	2046.000000	2046.0
mean	35.462429	-36.015037	2.145406	0.3
std	23.352702	100.829355	2.625453	1.2
min	-49.500000	-169.320000	0.000000	0.0
25%	37.000000	-116.051500	0.000000	0.0
50%	37.100000	-116.000000	0.000000	0.0
75%	49.870000	78.000000	5.100000	0.0
max	75.100000	179.220000	7.400000	6.0

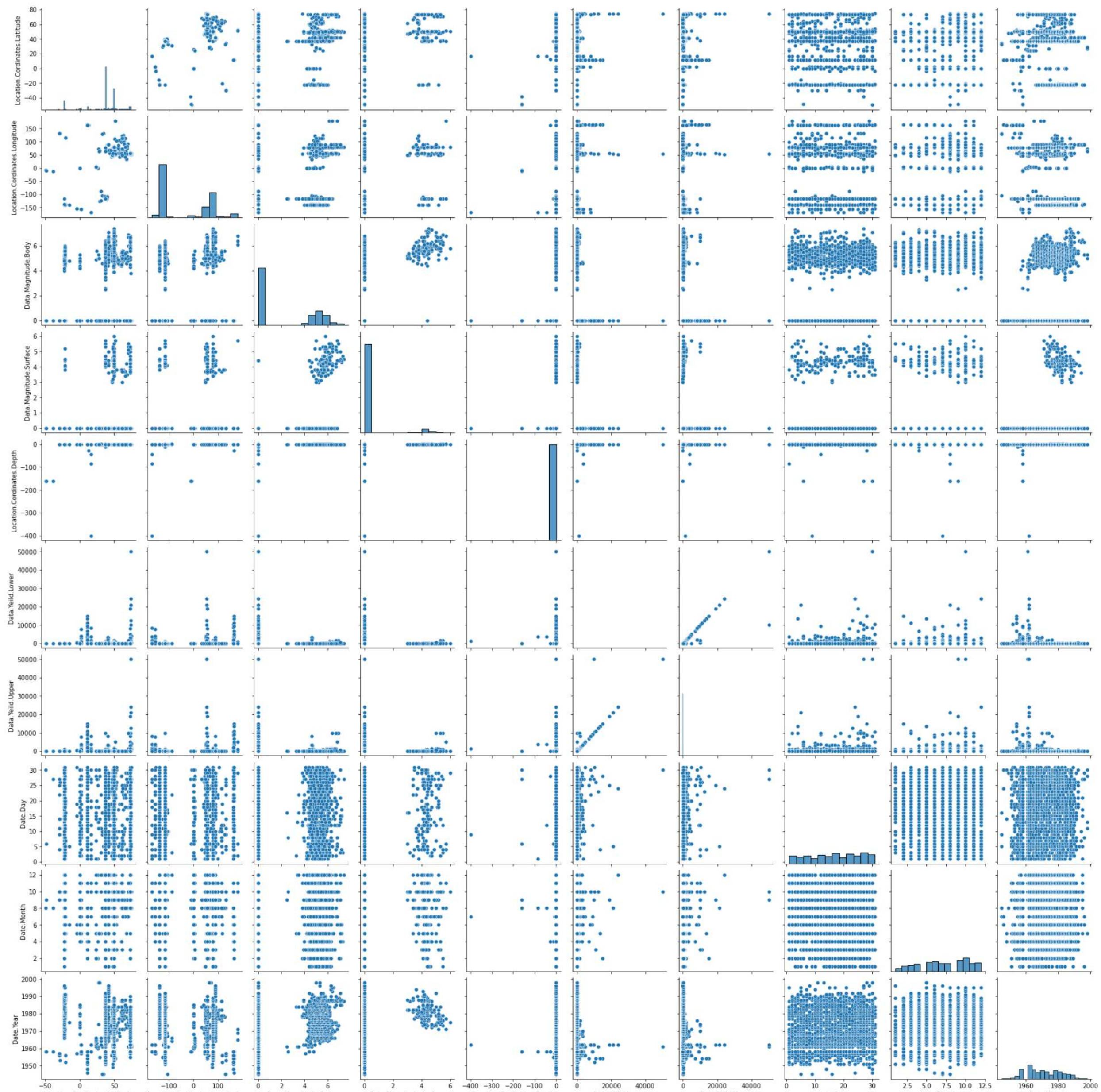
In [7]:
```
#To Display column heading
df.columns
```

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(df)
```

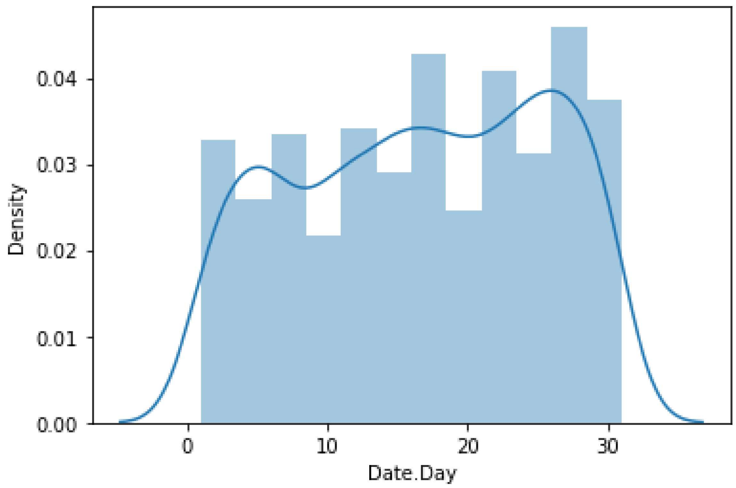
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x27d65cf33a0>
```



In [9]:
```
sns.distplot(df['Date.Day'])
```

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

Out[9]:
```
<AxesSubplot:xlabel='Date.Day', ylabel='Density'>
```

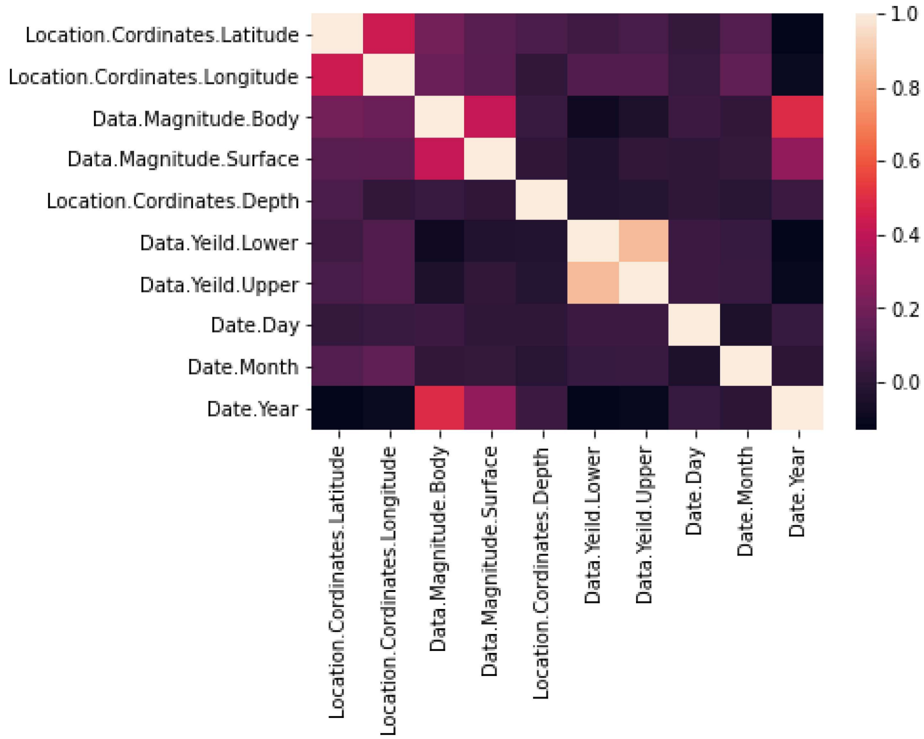


In [10]:
```
df1=df[[
        'Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
        'Data.Magnitude.Body', 'Data.Magnitude.Surface',
        'Location.Cordinates.Depth', 'Data.Yeild.Lower', 'Data.Yeild.Upper','Date.Day',
        'Date.Year']]
```

Plot Using Heat Map

In [11]:
```
sns.heatmap(df1.corr())
```

Out[11]:
```
<AxesSubplot:>
```



To Train The Model-Model Building

we are going to train Linera 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 required for our model

In [12]:
```
x=df1[[
        'Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
        'Data.Magnitude.Body', 'Data.Magnitude.Surface',
        'Location.Cordinates.Depth', 'Data.Yeild.Lower', 'Data.Yeild.Upper','Date.Day',
        'Date.Year']]
y=df1[ 'Date.Month']
```

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]:
```
lr.intercept_
```

Out[15]: 7.638334409421077e-14

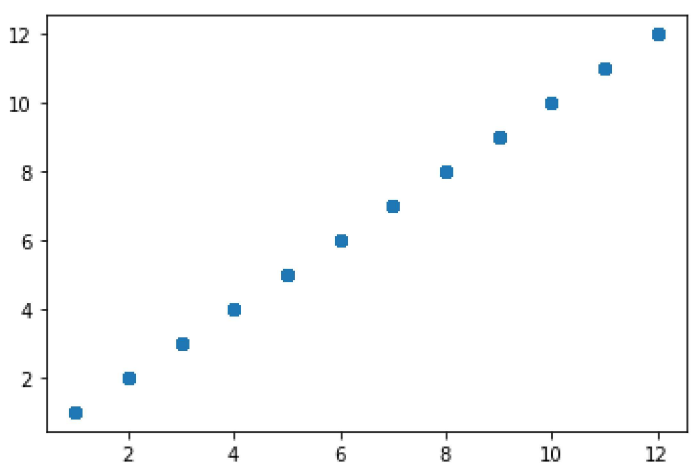
In [16]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
Location.Cordinates.Latitude	-9.276254e-18
Location.Cordinates.Longitude	-6.318855e-18
Data.Magnitude.Body	4.170730e-16
Data.Magnitude.Surface	-4.628821e-16
Location.Cordinates.Depth	-2.511849e-17
Data.Yeild.Lower	2.881462e-18
Data.Yeild.Upper	-1.797818e-18
Date.Day	4.543696e-17
Date.Month	1.000000e+00
Date.Year	-4.381773e-17

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

Out[17]: <matplotlib.collections.PathCollection at 0x27d79576220>



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

Out[18]: 1.0

Accuracy

In [19]:
```
lr.score(x_test,y_test)
```

Out[19]:
```
1.0
```

In [20]:
```
lr.score(x_train,y_train)
```

Out[20]:
```
1.0
```

In [21]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [22]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[22]:
```
Ridge(alpha=10)
```

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

Out[23]:
```
0.999999469698729
```

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]:
```
0.011582226663732764
```

ElasticNet

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

Out[26]:
```
ElasticNet()
```

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

```
[ 0.00000000e+00  4.14263078e-04  0.00000000e+00  0.00000000e+00
 -0.00000000e+00 -4.11040547e-06  5.75446955e-06 -0.00000000e+00
  9.01298555e-01  0.00000000e+00]
```

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

```
0.7421113055370583
```

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

```
[ 7.07328495  8.87806243  4.29930198  7.00326176  5.24860409  3.38860577
  6.09246001  4.34738333  8.80574392  7.08443902  6.09329095  8.90807663
 10.6565105   5.20157682 10.68917804  5.19645626 11.58986346  5.2005956
  6.10291681  7.00374619  6.10187278  9.70715743 10.60845599  8.9079844
 11.59085788  2.49676899  3.47845943  9.78741431  7.00318595  9.78757417
  9.7874222   8.88534477  9.77952539  6.10270139 11.50971311  2.4968477
  3.39806754 10.69133844  7.90519719  8.80574381  2.5679039   6.99448179
  5.2005818   9.70704448  4.3798077  11.50975454  6.9936474   6.10192178
  9.77829234  5.20130567  5.20064394  7.98608141  7.90445296  7.00326176
  3.3884244  11.50975454 10.65725858  8.88611083 11.59002622  4.30027827
  9.77790599  2.57750057  4.3798077   6.18237167  9.78759489  5.20066465
  9.77421872 10.60845599  8.80615316  8.8732688  10.68877218  9.7117966
 10.68882302  5.20056109  8.00893831  9.78753275  3.47843458  3.47848015
  8.88626361  9.77758215  2.49739343  7.08355837  3.47834251  6.9936474
  8.88022945 10.59889976  7.08358737 11.59120107  9.70704238  6.10262245
 11.50975454  2.49676899  4.2993661   2.57822297  3.4795056   3.39802315
  7.00392764  4.29903438  3.39797547  1.59517271  9.707145    8.88623834
  2.54765533  9.77510731  8.80573323  4.2993661  11.59483055  9.78741595
  4.2993661   9.78760732  2.49676899  9.77769886  7.0979423  11.50099114
  7.98494807  6.18241309  7.98481424  4.37063173  8.87380876 10.60836899
  5.32328779  7.90456032  8.80578975  7.9848149   6.21836191  7.88860417
  1.59641989  1.59547043  1.67572452  8.87382118  8.88764795 10.60845599
 11.50998043  8.80577312  7.00325762  6.10196321  9.70704292 11.5804888
  7.00314667  5.19105029  6.22014737  5.28090246  6.10194871  6.15001772
  3.39806754 11.59012157  3.4783195  11.50975454  5.20066465  3.4688572
  3.39803026  6.10196321  3.39809654  3.39871063  4.2993661   7.0836288
  7.90451061  5.20057423  8.87660769  5.20065223  3.47831975  8.80585225
  9.70704234  6.21707784  9.6875679   9.70715743 11.52082518  3.39869655
  8.87694002  8.93359376  1.68039436 11.59012157  9.82947261 10.67546796
  5.20060217  9.7781937   6.18233024  7.90441104  8.8861322   8.80574707
  4.29925134 11.58049208 11.59012572  6.1827735   9.79236257  7.90453961
 11.58051882 11.51042041  2.57810845  9.80937025  6.10182954 10.68911895
  9.70704246  4.29987869  2.49676899  2.57788475  4.36706768  7.90519719
  7.98329936  7.07330235 10.59892793  6.99373371  9.70704234 11.590163
  3.47845943  5.20062737  6.10192178 10.61312001 11.50975454  9.77508453
 10.68870859  9.78856732  6.10198392  5.28101926  5.20066465  4.2993661
  7.90456032  2.49666327  9.77873579  6.23170757  9.78761146  5.20046738
  8.92746474  3.4785129   8.80681247  7.00316262  4.29986627 11.50014017
 10.60845599  4.41453911  4.29918382  2.49675035  8.01967251 10.6083409
  4.2993545   8.79007425  7.00324934  9.78622369  2.49676899  4.28988776
  3.39870566  9.79551805  8.87998609  6.18278593  4.29990231  8.80585887
  5.1911366   6.99361862  7.07408495  8.80585887  8.87727195  6.15001773
  6.1834311   2.49772258  2.49722773  2.49666705  9.78264911  6.09243516
  4.28978834 10.59884162 11.59122261 11.50971311 10.59973733  9.80187939
 10.60910114  3.3979541   5.20066217  8.79621573  5.20061453  7.98043144
  4.2993661   7.0887208   5.20053749  6.99937309  5.20063938  6.15001773
 10.68871237  6.187676    6.18221516  8.80585887  8.80577188  9.78741069
  9.77777686  5.2783125   9.75521195  6.10195005 11.59211244  6.99373371
  8.80590444  6.09243516 11.590221    4.37960885  6.08617858  5.20066465
  7.99004261  6.10245509  9.78659876  7.90456032  3.39796626  2.49675656
  3.48296681 11.51040384  4.37971176  5.20066465  2.57702094 11.51039059
  7.98506394 10.67920454  8.88626733  9.77790599  7.07329489  4.2993661
  5.28102754  7.89503227  8.80574379  6.10196321  9.70715743  5.20053702
  2.57718573  6.18271136  7.11896581  6.08486695  2.49741001  2.57742008
  8.80574806  5.31578654  9.70715743  3.39795246  8.00445965  7.90455825
  5.24868135  5.2006029   7.9808457   2.49676899  6.09807454  1.66288105
  7.98480307  8.80577502  9.70704253  6.10196321 10.6094013   6.10271129
 11.59013152  2.4966539   9.78759075  9.80938176  4.28975173 10.59902977
  9.77464618  7.08366608  7.00418636  7.90454375  6.18343459  1.59548286
  8.87666313  8.8771566   5.2005808   5.20133053  8.00731899  7.00419879
```

```
  7.11974432  8.80585887  7.98395486  6.21709792  9.70715743  5.20062323
  5.28567651 10.67546796 11.50975454  1.59542072  9.70779058  9.70723141
 10.60939301  6.09319152  6.10190895  5.31591001  5.18364698 10.68882302
 11.50975454  2.49667346  9.77788133  6.10271796  6.9936474   1.59547043
 10.68931942  9.79175069  7.90456032  8.80650403  3.39987919  4.30001954
  6.99373371  8.85457926  6.18302098  2.49669007  4.30028655  1.59550792
  9.78752446  7.00385367  8.8767143   7.9848172  10.59970087 11.58048238
  2.49676899  8.87381994  9.7750837   6.09243516  9.77415493  9.77839987
  9.79132123  2.57726787  4.2992791   7.89606807  7.90456032  9.70704265
  7.08467165  8.8818957   7.90446331  9.70704234  9.78752985  3.39806754
  9.70703315  3.39795492  7.00326176  3.39900995  4.37965422  3.48315588
  6.10196321  2.57716875  5.28204555  3.4783195   1.59547043  8.88601127
  8.80586385  5.19184326  6.10192178  6.10194841  7.89503227  8.88724805
  2.49676899 11.50977111  5.2811104   5.20061908  6.18220596  7.08366194
  4.30009761  3.39803689  8.88613713  7.0979133   6.18238948  9.70811103
  2.57825901  4.2993661   7.00314996  8.8738129   4.29937453  8.87698405
 11.59020857  2.49665719  8.00314496  7.00329283  4.2993661  10.59881285
  7.13492725  5.20133053  6.18236338  9.78864603  7.06809804  9.78728149
  7.97537566  8.8058713   8.87921327  2.49665702  1.595487    5.20066465
  9.70704247  8.80574379  2.49773087  5.19188096  9.78740943 11.50975454
  9.7080986   9.7078233   8.80576187  2.49853506  4.29939095  3.44819338
  8.80585887  9.70715743  2.57713602  8.80585887  4.29925331  6.09234884
  5.31802921 10.6084506   7.00316312  7.90447647  7.00420293  2.56997896
  5.20066465  6.10187414  7.00391107  3.4694375   3.39779684  9.70704234
  9.69754306  1.59535699  5.28109383 10.60940958  6.10196321  7.00326176
  4.29938267 11.51038313  7.08368638  2.49651115  8.88622591  4.34737919
  5.20056929  3.38930414  8.80588787  9.78589647  7.00326176  7.90522205
 11.50975454  9.77495319  7.98401991  8.88631207  7.89503227 11.50975454
 10.60845599 11.51455754  5.31580643 10.68870833  9.70715743  5.2037663
  6.10290438  1.67594932  5.28221889  5.18372918  9.78755802 10.59967602
  1.59537261 11.51066671 10.6084477  11.50973797 10.68857446  7.90520548
  3.47834908  8.88611214  8.8537988   5.20055272  7.11967198  8.88622591
  8.80585887  7.98482443  7.89461801  5.20059395  2.49739343  6.10196321
  6.10193421  9.77951955  3.39802197  9.70700351  5.20119589  9.78792216
  6.99361862  7.0836288   5.1911366   4.37961842  2.49768944  4.37973313
  7.90539378  6.99384556  4.41461145  6.18337724  9.78589647  2.49676899
  9.77847319  7.0843361   4.3807851  11.59068803  7.98481851  7.0712226
  9.70704238  8.80579135  4.2993661   3.39805926 10.68874845  7.90454541
  9.77860472  4.28294029  8.80576351  9.77835811  7.00417393 11.50962198
  3.47834758  3.38857745  6.10196321 10.67982107  7.98486685  4.2993661
 11.59012158  4.37961805  3.39898386  2.49676899  9.70811103  7.11879319
  6.10268146  4.29062507  7.08469651  8.87855565  8.8861128  11.58042419
  4.37973313 11.59012157]
```

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

```
0.9903753561940183
```

Evaluation Metrics

In [31]:
```python
from sklearn import metrics
```

In [32]:
```python
print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error 3.2413449412753347e-15
```

In [33]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 1.644470652814272e-29
```

In [34]:
```
print("Root Mean Absolute Error:",np.sqrt(metrics.mean_squared_error(y_test,predictio
```

```
Root Mean Absolute Error: 4.0552073347909996e-15
```

In [35]:
```
print("Root Mean Absolute Error:",np.sqrt(metrics.mean_squared_error(y_test,predictio
```

```
Root Mean Absolute Error: 4.0552073347909996e-15
```

Model Saving

In [36]:
```
import pickle
```

In [37]:
```
filename="prediction"
pickle.dump(lr,open(filename,'wb'))
```

In [38]:
```
model=pickle.load(open(filename,'rb'))
```

In [39]:
```
real=[[10,20,30,40,50,78,45,56,87,58],[11,45,10,29,25,78,56,54,23,87]]
```

In [40]:
```
result =model.predict(real)
```

In [41]:
```
print(result)
```

```
[87. 23.]
```

In [ ]: