In [1]:

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

19_nuclear_explosions

In [2]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\19_nuclear_explosions.csv")
a
```

Out[2]:

	WEAPON SOURCE COUNTRY	WEAPON DEPLOYMENT LOCATION	Data.Source	Location.Cordinates.Latitude	Location.Cordinate
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

LINEAR REGRESSION

In [4]:

```
c=a.dropna(axis=1)
c
```

Out[4]:

ody	Data.Magnitude.Surface	Location.Cordinates.Depth	Data.Yeild.Lower	Data.Yeild.Upper	Data
0.0	0.0	-0.10	21.0	21.0	
0.0	0.0	-0.60	15.0	15.0	
0.0	0.0	-0.60	21.0	21.0	
0.0	0.0	-0.20	21.0	21.0	
0.0	0.0	0.03	21.0	21.0	
...	...	...	...	...	
5.3	0.0	0.00	3.0	12.0	
5.3	0.0	0.00	0.0	20.0	
0.0	0.0	0.00	0.0	1.0	
0.0	0.0	0.00	0.0	35.0	
5.0	0.0	0.00	0.0	18.0	

In [5]:

```
c.columns
```

Out[5]:

```
Index(['WEAPON SOURCE COUNTRY', 'WEAPON DEPLOYMENT LOCATION', 'Data.Sourc
e',
       'Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
       'Data.Magnitude.Body', 'Data.Magnitude.Surface',
       'Location.Cordinates.Depth', 'Data.Yeild.Lower', 'Data.Yeild.Uppe
r',
       'Data.Purpose', 'Data.Name', 'Data.Type', 'Date.Day', 'Date.Month',
       'Date.Year'],
      dtype='object')
```

In [8]:

```
x=a[['Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
       'Data.Magnitude.Body', 'Data.Magnitude.Surface']]
y=a['Date.Year']
```

In [9]:

```
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 [10]:

```
from sklearn.linear_model import LinearRegression

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

Out[10]:

```
LinearRegression()
```

In [11]:

```
print(lr.intercept_)
```

```
1968.685507614099
```

In [12]:

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

Out[12]:

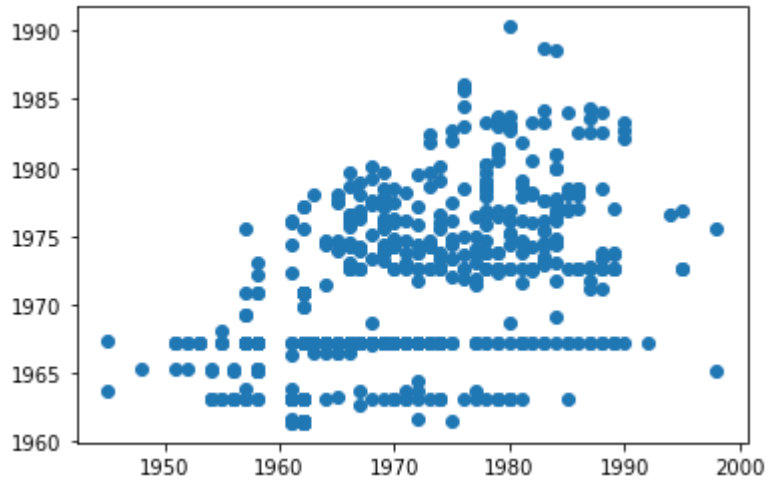
	Co-efficient
Location.Cordinates.Latitude	-0.087288
Location.Cordinates.Longitude	-0.015023
Data.Magnitude.Body	2.051086
Data.Magnitude.Surface	1.100463

In [13]:

```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[13]:

```
<matplotlib.collections.PathCollection at 0x2a625b64b20>
```



In [14]:

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

```
0.2690250260219136
```

In [15]:

```
lr.score(x_train,y_train)
```

Out[15]:

```
0.33935447564119725
```

RIDGE REGRESSION

In [16]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [17]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[17]:

```
Ridge(alpha=10)
```

In [18]:

```
rr.score(x_test,y_test)
```

Out[18]:

```
0.26911034381285104
```

LASSO REGRESSION

In [19]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[19]:

```
Lasso(alpha=10)
```

In [20]:

```
la.score(x_test,y_test)
```

Out[20]:

```
0.15352877970681045
```

ELASTIC NET

In [21]:

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

Out[21]:

```
ElasticNet()
```

In [22]:

```
print(p.coef_)
```

```
[-0.08135099 -0.0140463   1.91249546  0.65032937]
```

In [23]:

```
print(p.intercept_)
```

```
1968.983629811167
```

In [24]:

```
print(p.predict(x_test))
```

```
 1967.60301411 1963.82046892 1972.72578736 1967.60301411 1965.78675598
 1967.60301411 1975.81647913 1967.8193366  1967.59137331 1976.6196872
 1973.81285246 1965.78675598 1976.41919967 1973.00970591 1967.60301411
 1971.81276361 1977.73298406 1963.82046892 1967.60301411 1962.2724611
 1967.60301411 1972.71668131 1963.88543457 1972.83011395 1967.60301411
 1973.39575516 1967.60301411 1967.60301411 1972.72578736 1972.72578736
 1965.73984722 1967.60301411 1967.59052118 1975.22630222 1977.20736027
 1963.82046892 1962.2724611  1967.59487901 1963.82046892 1967.58862259
 1967.58823164 1967.59667576 1967.60301411 1967.60301411 1967.60301411
 1982.09701509 1967.60301411 1967.59256489 1967.58896781 1963.82046892
 1967.60301411 1963.82046892 1967.60301411 1964.34108128 1971.61762866
 1974.1662021  1967.60301411 1972.64443637 1967.60301411 1967.60301411
 1973.95855012 1969.58409408 1963.82046892 1967.60301411 1975.24419364
 1967.59628364 1967.60301411 1970.02372568 1967.60301411 1967.60301411
 1967.60301411 1971.80450053 1976.97394453 1967.60301411 1972.71174106
 1978.29608056 1962.2724611  1962.2724611  1967.59487901 1978.11416586
 1967.60301411 1967.60301411 1967.60301411 1963.82046892 1967.60301411
 1967.60301411 1975.74470869 1974.15545909 1974.63542803 1963.82046892
 1963.82046892 1973.39497911 1977.89865124 1967.56196388 1978.30485356
 1970.02372568 1967.60301411 1981.55350739 1973.97374927 1981.99628942
```

In [25]:

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

```
0.2729317077827548
```

EVALUATION METRICS

In [26]:

```
from sklearn import metrics
```

In [27]:

```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolytre Error: 6.7857384103334395
```

In [28]:

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

```
Mean Squared Error: 73.0081197322618
```

In [29]:

```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root Mean Squared Error: 8.544478903494454
```

MODEL SAVING

In [30]:

```
import pickle
```

In [31]:

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

In [32]:

```
filename="prediction"
model=pickle.load(open(filename,'rb'))
```

In [34]:

```
real=[[10,20,30,40],[12,43,98,45]]
result=model.predict(real)
```

In [35]:

```
result
```

Out[35]:

```
array([2073.06327932, 2217.51934189])
```

20_states

In [36]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\20_states.csv")
a
```

Out[36]:

	id	name	country_id	country_code	country_name	state_code	type	latitu
0	3901	Badakhshan	1	AF	Afghanistan	BDS	NaN	36.7347
1	3871	Badghis	1	AF	Afghanistan	BDG	NaN	35.1671
2	3875	Baghlan	1	AF	Afghanistan	BGL	NaN	36.1789
3	3884	Balkh	1	AF	Afghanistan	BAL	NaN	36.7550
4	3872	Bamyan	1	AF	Afghanistan	BAM	NaN	34.8100
...	...	...	...	...	...	...	...	
5072	1953	Mashonaland West Province	247	ZW	Zimbabwe	MW	NaN	-17.4851
5073	1960	Masvingo Province	247	ZW	Zimbabwe	MV	NaN	-20.6241
5074	1954	Matabeleland North Province	247	ZW	Zimbabwe	MN	NaN	-18.5331
5075	1952	Matabeleland South Province	247	ZW	Zimbabwe	MS	NaN	-21.0523
5076	1957	Midlands Province	247	ZW	Zimbabwe	MI	NaN	-19.0552

5077 rows × 9 columns

In [37]:

```
a.columns
```

Out[37]:

```
Index(['id', 'name', 'country_id', 'country_code', 'country_name',
       'state_code', 'type', 'latitude', 'longitude'],
      dtype='object')
```

In [42]:

```
x1=a[['id','country_id']]
y1=a['country_id']
```

In [43]:

```
from sklearn.model_selection import train_test_split
x1_train,x1_test,y1_train,y1_test=train_test_split(x1,y1,test_size=0.3)
```

In [44]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x1_train,y1_train)
```

Out[44]:

```
LinearRegression()
```

In [45]:

```
print(lr.intercept_)
```

```
2.842170943040401e-14
```

In [47]:

```
coeff=pd.DataFrame(lr.coef_,x1.columns,columns=['Co-efficient'])
coeff
```

Out[47]:

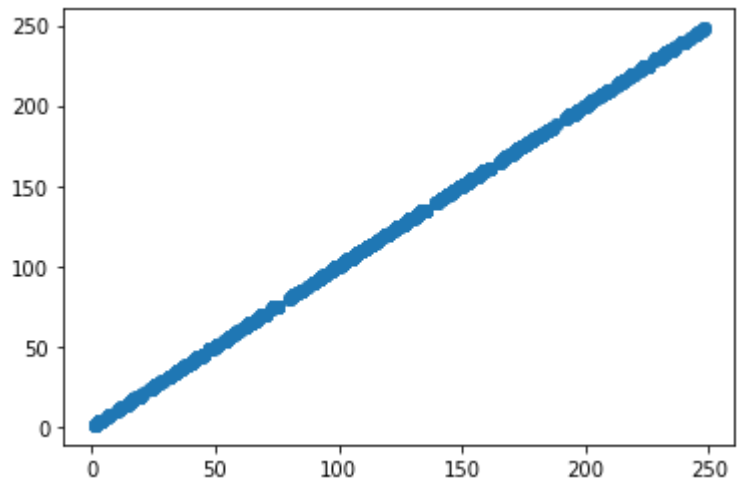
	Co-efficient
id	0.0
country_id	1.0

In [49]:

```
prediction=lr.predict(x1_test)
plt.scatter(y1_test,prediction)
```

Out[49]:

```
<matplotlib.collections.PathCollection at 0x2a6265b6c70>
```



In [50]:

```
print(lr.score(x1_test,y1_test))
```

```
1.0
```

In [51]:

```
lr.score(x1_train,y1_train)
```

Out[51]:

```
1.0
```

In [52]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [57]:

```
rr=Ridge(alpha=10)
rr.fit(x1_train,y1_train)
```

Out[57]:

```
Ridge(alpha=10)
```

In [58]:

```
rr.score(x1_test,y1_test)
```

Out[58]:

```
0.999999999999714
```

In [59]:

```
la=Lasso(alpha=10)
la.fit(x1_train,y1_train)
```

Out[59]:

```
Lasso(alpha=10)
```

In [61]:

```
la.score(x1_test,y1_test)
```

Out[61]:

```
0.9999963863596751
```

In [62]:

```
from sklearn.linear_model import ElasticNet
p=ElasticNet()
p.fit(x1_train,y1_train)
```

Out[62]:

```
ElasticNet()
```

In [63]:

```
print(p.coef_)
```

```
[1.08180632e-07 9.99809879e-01]
```

In [64]:

```
print(p.intercept_)
```

```
0.025010731354967675
```

In [65]:

```
print(p.predict(x1_test))
```

```
[200.98725779 200.98725216 198.98767968 ... 218.98375307 169.99284117
  99.00630147]
```

In [66]:

```
print(p.score(x1_test,y1_test))
```

```
0.999999963890732
```

In [67]:

```
from sklearn import metrics
```

In [68]:

```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y1_test,prediction))
```

```
Mean Absolytre Error: 8.770179100300088e-15
```

In [69]:

```
print("Mean Squared Error:",metrics.mean_squared_error(y1_test,prediction))
```

```
Mean Squared Error: 1.7703961691022298e-28
```

In [70]:

```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y1_test,prediction)))
```

```
Root Mean Squared Error: 1.3305623507007214e-14
```

In [71]:

```
import pickle
```

In [72]:

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

In [73]:

```
filename="prediction"
model=pickle.load(open(filename,'rb'))
```

In [74]:

```
real=[[10,20],[12,43]]
result=model.predict(real)
```

In [75]:

```
result
```

Out[75]:

```
array([20., 43.])
```

21_cities

In [76]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\21_cities.csv")
a
```

Out[76]:

	id	name	state_id	state_code	state_name	country_id	country_code	cou
0	52	Ashkāsham	3901	BDS	Badakhshan	1	AF	
1	68	Fayzabad	3901	BDS	Badakhshan	1	AF	
2	78	Jurm	3901	BDS	Badakhshan	1	AF	
3	84	Khandūd	3901	BDS	Badakhshan	1	AF	
4	115	Rāghistān	3901	BDS	Badakhshan	1	AF	
...	...	...	...	...	...	...	...	
150449	131496	Redcliff	1957	MI	Midlands Province	247	ZW	
150450	131502	Shangani	1957	MI	Midlands Province	247	ZW	
150451	131503	Shurugwi	1957	MI	Midlands Province	247	ZW	
150452	131504	Shurugwi District	1957	MI	Midlands Province	247	ZW	
150453	131508	Zvishavane District	1957	MI	Midlands Province	247	ZW	

150454 rows × 11 columns

In [77]:

```
a.columns
```

Out[77]:

```
Index(['id', 'name', 'state_id', 'state_code', 'state_name', 'country_id',
       'country_code', 'country_name', 'latitude', 'longitude', 'wikiDataI
d'],
      dtype='object')
```

In [78]:

```
x2=a[['id', 'state_id', 'country_id']]
y2=a['country_id']
```

In [79]:

```
from sklearn.model_selection import train_test_split
x2_train,x2_test,y2_train,y2_test=train_test_split(x2,y2,test_size=0.3)
```

In [80]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x2_train,y2_train)
```

Out[80]:

```
LinearRegression()
```

In [81]:

```
print(lr.intercept_)
```

```
-7.105427357601002e-13
```

In [82]:

```
coeff=pd.DataFrame(lr.coef_,x2.columns,columns=['Co-efficient'])
coeff
```

Out[82]:

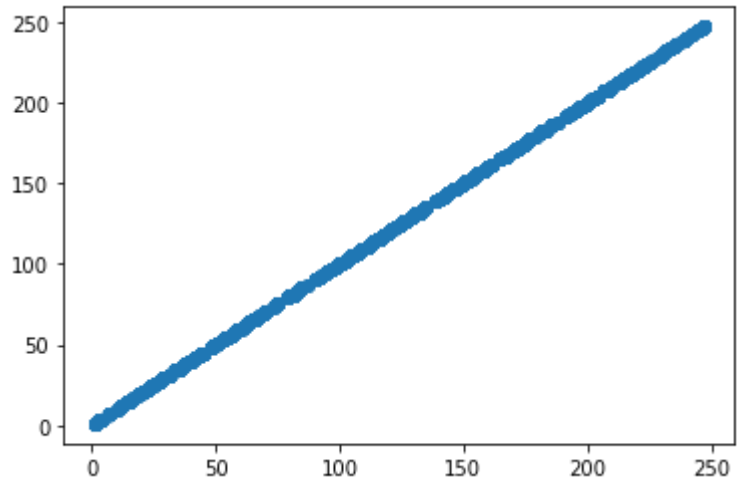
	Co-efficient
id	1.542866e-17
state_id	-1.692364e-16
country_id	1.000000e+00

In [83]:

```
prediction=lr.predict(x2_test)
plt.scatter(y2_test,prediction)
```

Out[83]:

```
<matplotlib.collections.PathCollection at 0x2a62758f7c0>
```



In [84]:

```
print(lr.score(x2_test,y2_test))
```

```
1.0
```

In [85]:

```
lr.score(x2_train,y2_train)
```

Out[85]:

```
1.0
```

In [86]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [87]:

```
rr=Ridge(alpha=10)
rr.fit(x2_train,y2_train)
```

Out[87]:

```
Ridge(alpha=10)
```

In [88]:

```
rr.score(x2_test,y2_test)
```

Out[88]:

```
0.9999999999999993
```

In [89]:

```
la=Lasso(alpha=10)
la.fit(x2_train,y2_train)
```

Out[89]:

```
Lasso(alpha=10)
```

In [90]:

```
la.score(x2_test,y2_test)
```

Out[90]:

```
0.9999932395709826
```

In [91]:

```
from sklearn.linear_model import ElasticNet
p=ElasticNet()
p.fit(x2_train,y2_train)
```

Out[91]:

```
ElasticNet()
```

In [92]:

```
print(p.coef_)
```

```
[ 3.38570771e-07 -2.24821008e-06  9.99657292e-01]
```

In [93]:

```
print(p.intercept_)
```

```
0.028336857837132357
```

In [94]:

```
print(p.predict(x2_test))
```

```
[181.99585823 173.99357293  15.01925693 ...  14.01699914 176.99297896
 107.00858367]
```

In [95]:

```
from sklearn import metrics
```

In [96]:

```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y2_test,prediction))
```

```
Mean Absolytre Error: 6.372333224339684e-13
```

In [97]:

```
print("Mean Squared Error:",metrics.mean_squared_error(y2_test,prediction))
```

```
Mean Squared Error: 5.469877411531514e-25
```

In [98]:

```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y2_test,prediction)))
```

```
Root Mean Squared Error: 7.395861958914264e-13
```

In [99]:

```
import pickle
```

In [100]:

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

In [101]:

```
filename="prediction"
model=pickle.load(open(filename,'rb'))
```

In [102]:

```
real=[[10,20,30],[12,43,98]]
result=model.predict(real)
```

In [103]:

```
result
```

Out[103]:

```
array([30., 98.])
```

22_countries

In [104]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\22_countries.csv")
a
```

Out[104]:

ıcy	currency_name	currency_symbol	tld	native	region	subregion	
FN	Afghan afghani	؋	.af	افغانستان	Asia	Southern Asia	[{zoneName:'Asia\/Ka
UR	Euro	€	.ax	Åland	Europe	Northern Europe	[{zoneName:'Europe\/M
ıLL	Albanian lek	Lek	.al	Shqipëria	Europe	Southern Europe	[{zoneName:'Europe\/
ZD	Algerian dinar	دج	.dz	الجزائر	Africa	Northern Africa	[{zoneName:'Africa\/
SD	US Dollar	$	.as	American Samoa	Oceania	Polynesia	[{zoneName:'Pacific\/I
...	...	...	...	...	...	...	
PF	CFP franc	₣	.wf	Wallis et Futuna	Oceania	Polynesia	[{zoneName:'Pacific
AD	Moroccan Dirham	MAD	.eh	الصحراء الغربية	Africa	Northern Africa	[{zoneName:'Africa\/
ER	Yemeni rial	﷼	.ye	اليَمَن	Asia	Western Asia	[{zoneName:'Asia\/Ad
MW	Zambian kwacha	ZK	.zm	Zambia	Africa	Eastern Africa	[{zoneName:'Africa\/L
WL	Zimbabwe Dollar	$	.zw	Zimbabwe	Africa	Eastern Africa	[{zoneName:'Africa\/I

In [105]:

```
c=a.dropna(axis=1)
c
```

Out[105]:

	id	name	iso3	numeric_code	phone_code	currency	currency_name	currency_
0	1	Afghanistan	AFG	4	93	AFN	Afghan afghani	
1	2	Aland Islands	ALA	248	+358-18	EUR	Euro	
2	3	Albania	ALB	8	355	ALL	Albanian lek	
3	4	Algeria	DZA	12	213	DZD	Algerian dinar	
4	5	American Samoa	ASM	16	+1-684	USD	US Dollar	
...	...	...	...	...	...	...	...	
245	243	Wallis And Futuna Islands	WLF	876	681	XPF	CFP franc	
246	244	Western Sahara	ESH	732	212	MAD	Moroccan Dirham	
247	245	Yemen	YEM	887	967	YER	Yemeni rial	
248	246	Zambia	ZMB	894	260	ZMW	Zambian kwacha	
249	247	Zimbabwe	ZWE	716	263	ZWL	Zimbabwe Dollar	

250 rows × 14 columns

In [107]:

```
c.columns
```

Out[107]:

```
Index(['id', 'name', 'iso3', 'numeric_code', 'phone_code', 'currency',
       'currency_name', 'currency_symbol', 'tld', 'timezones', 'latitude',
       'longitude', 'emoji', 'emojiU'],
      dtype='object')
```

In [108]:

```
x3=c[['id', 'numeric_code']]
y3=c['numeric_code']
```

In [109]:

```
from sklearn.model_selection import train_test_split
x3_train,x3_test,y3_train,y3_test=train_test_split(x3,y3,test_size=0.3)
```

In [110]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x3_train,y3_train)
```

Out[110]:

```
LinearRegression()
```

In [111]:

```
print(lr.intercept_)
```

```
-5.684341886080802e-14
```

In [112]:

```
coeff=pd.DataFrame(lr.coef_,x3.columns,columns=['Co-efficient'])
coeff
```

Out[112]:

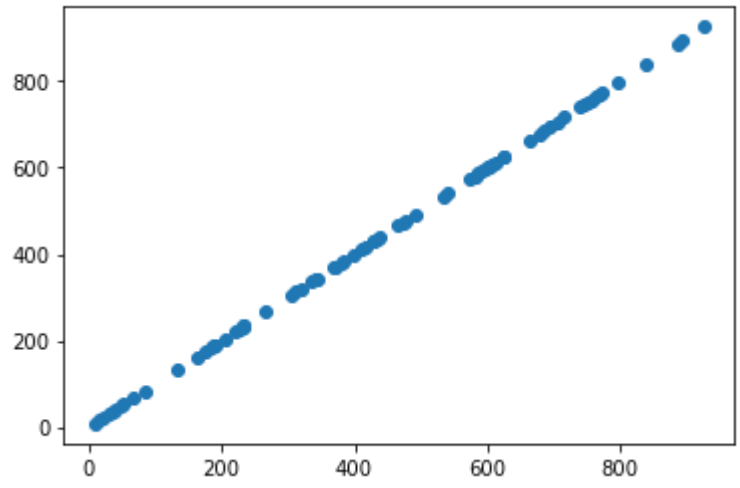
	Co-efficient
id	1.169303e-15
numeric_code	1.000000e+00

In [113]:

```
prediction=lr.predict(x3_test)
plt.scatter(y3_test,prediction)
```

Out[113]:

```
<matplotlib.collections.PathCollection at 0x2a628107b50>
```



In [114]:

```
print(lr.score(x3_test,y3_test))
```

```
1.0
```

In [115]:

```
lr.score(x3_train,y3_train)
```

Out[115]:

```
1.0
```

In [116]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [117]:

```
rr=Ridge(alpha=10)
rr.fit(x3_train,y3_train)
```

Out[117]:

```
Ridge(alpha=10)
```

In [118]:

```
rr.score(x3_test,y3_test)
```

Out[118]:

```
0.9999999999992337
```

In [119]:

```
la=Lasso(alpha=10)
la.fit(x3_train,y3_train)
```

Out[119]:

```
Lasso(alpha=10)
```

In [120]:

```
la.score(x3_test,y3_test)
```

Out[120]:

```
0.9999999762202638
```

In [121]:

```
from sklearn.linear_model import ElasticNet
p=ElasticNet()
p.fit(x3_train,y3_train)
```

Out[121]:

```
ElasticNet()
```

In [122]:

```
print(p.coef_)
```

```
[0.         0.99998458]
```

In [123]:

```
print(p.intercept_)
```

```
0.006706627748144456
```

In [124]:

```
print(p.predict(x3_test))
```

```
[304.00201908 573.9978558  625.99705398 539.99838007 312.00189572
 222.00328349 693.99600545 771.99480273 410.00038461 893.99292154
 607.99733153 437.99995286 465.99952111  31.00622862 751.99511112
 597.99748573 705.99582042 477.99933608 380.00084719 795.99443266
 430.00007622 336.00152566 590.99759367  40.00608985 715.99566622
 233.00311387 580.99774786 384.00078552 767.99486441 839.9937542
 685.99612881 533.99847258 603.99739321 599.99745489  52.00590481
 320.00177237 204.00356104 417.00027667 132.00467125  16.00645991
 344.0014023  226.00322181 368.00103223 743.99523447 585.99767077
 662.99648346   8.00658327  20.00639824 491.9991202  763.99492608
 184.00386943 473.99939776 886.99302948  48.00596649 925.99242811
 434.00001454 340.00146398 191.00376149 739.99529615 611.99726986
 372.00097055 174.00402362 162.00420866 755.99504944 232.00312929
  68.0056581  188.00380775 266.00260502 398.00056964  36.00615152
 234.00309845 623.99708482  84.00541138 428.00010706 677.99625217]
```

In [125]:

```
print(p.score(x3_test,y3_test))
```

```
0.9999999997622063
```

In [126]:

```
from sklearn import metrics
```

In [127]:

```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y3_test,prediction))
```

```
Mean Absolytre Error: 2.4750571962310157e-14
```

In [128]:

```
print("Mean Squared Error:",metrics.mean_squared_error(y3_test,prediction))
```

```
Mean Squared Error: 2.651456165756447e-27
```

In [129]:

```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y3_test,prediction)))
```

```
Root Mean Squared Error: 5.149229229463811e-14
```

In [130]:

```
import pickle
```

In [131]:

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

In [132]:

```
filename="prediction"
model=pickle.load(open(filename,'rb'))
```

In [135]:

```
real=[[45,80],[12,43]]
result=model.predict(real)
```

In [136]:

```
result
```

Out[136]:

```
array([80., 43.])
```

23_Vande Bharat

In [137]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\23_Vande Bharat.csv")
a
```

Out[137]:

ing ion	Terminal City	Terminal Station	Operator	No. of Cars	Frequency	Distance	Tr
elhi	Varanasi	Varanasi Junction	NR	16	Except Thursdays	759 km (472 mi)	
elhi	Katra	Shri Mata Vaishno Devi Katra	NR	16	Except Tuesdays	655 km (407 mi)	
ıbai ıtral	Gandhinagar	Gandhinagar Capital	WR	16	Except Wednesdays	522 km (324 mi)	
elhi	Andaura	Amb Andaura	NR	16	Except Fridays	412 km (256 mi)	
ınai ıtral	Mysuru	Mysore Junction	SR	16	Except Wednesdays	496 km (308 mi)	
pur tion	Nagpur	Nagpur Junction	SECR	8	Except Saturdays	412 km (256 mi)	
rah tion	Siliguri	New Jalpaiguri Junction	ER	16	Except Wednesdays	565 km (351 mi)	
ıam tion	Hyderabad	Secunderabad Junction	ECoR	16	Except Sundays	698 km (434 mi)	
pati vaji nus	Solapur	Solapur	CR	16	Except Wednesdays (22225) , Except Thursdays (...	452 km (281 mi)	
pati vaji nus	Shirdi	Sainagar Shirdi	CR	16	Except Tuesdays	339 km (211 mi)	
ganj tani ati)	Delhi	Hazrat Nizamuddin	WCR	16	Except Saturdays	702 km (436 mi)	
bad tion	Tirupati	Tirupati	SCR	16	Except Tuesdays	661 km (411 mi)	
ınai ıtral	Coimbatore	Coimbatore Junction	SR	8	Except Wednesdays	495 km (308 mi)	
elhi ent	Ajmer	Ajmer Junction	NWR	16	Except Wednesdays	428 km (266 mi)	
god	Thiruvananthapuram	Thiruvananthapuram Central	SR	16	Except Thursdays	587 km (365 mi)	
rah tion	Puri	Puri	SER	16	Except Thursdays	500 km (310 mi)	

ing ion	Terminal City	Terminal Station	Operator	No. of Cars	Frequency	Distance	Tr
har inal	Dehradun	Dehradun Terminal	NR	8	Except Wednesdays	304 km (189 mi)	
guri tion	Guwahati	Guwahati	NFR	8	Except Tuesdays	407 km (253 mi)	
pati vaji nus	Madgaon	Madgaon Junction	CR	16	Except Fridays\n(Non-Monsoon)	586 km (364 mi)	07h 4
pati vaji nus	Madgaon	Madgaon Junction	CR	16	Monday, Wednesday, Friday (22229)\nTuesday, Th...	586 km (364 mi)	05m\n(
tion	Ranchi	Ranchi Junction	ECR	8	Except Tuesdays	379 km (235 mi)	
City	Hubbali - Dharwad	Dharwad	SWR	8	Except Tuesdays	490 km (300 mi)	
ganj tani ati)	Jabalpur	Jabalpur Junction	WCR	8	Except Tuesdays	337 km (209 mi)	
tion	Bhopal	Bhopal Junction	WR	8	Except Sundays	250 km (160 mi)	
pur tion	Ahmedabad	Sabarmati Junction	NWR	8	Except Tuesdays	449 km (279 mi)	
pur tion	Charbagh	Lucknow Charbagh	NER	8	Except Saturdays	296 km (184 mi)	

In [138]:
```
a.columns
```

Out[138]:

```
Index(['Sr. No.', 'Train Name', 'Train Number', 'Originating City',
       'Originating Station', 'Terminal City', 'Terminal Station', 'Operat
or',
       'No. of Cars', 'Frequency', 'Distance', 'Travel Time', 'Speed',
       'Average Speed', 'Inauguration', 'Average occupancy'],
      dtype='object')
```

In [140]:
```
x4=a[['Sr. No.', 'No. of Cars']]
y4=a['No. of Cars']
```

In [141]:
```
from sklearn.model_selection import train_test_split
x4_train,x4_test,y4_train,y4_test=train_test_split(x4,y4,test_size=0.3)
```

In [142]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x4_train,y4_train)
```

Out[142]:

```
LinearRegression()
```

In [143]:

```
print(lr.intercept_)
```

```
3.552713678800501e-15
```

In [144]:

```
coeff=pd.DataFrame(lr.coef_,x4.columns,columns=['Co-efficient'])
coeff
```

Out[144]:

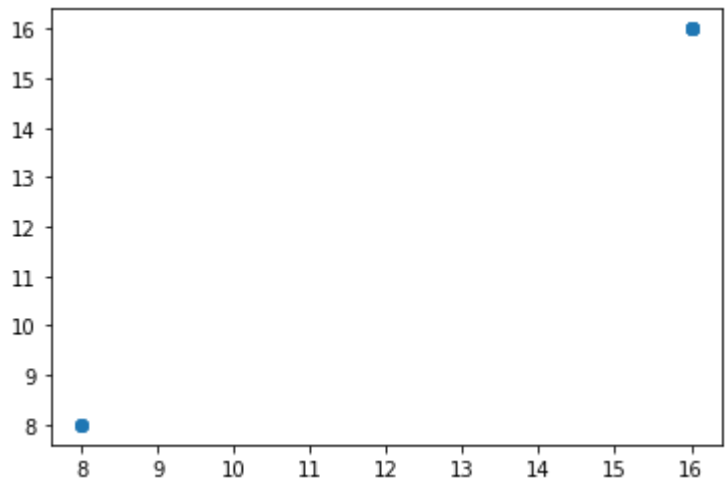
	Co-efficient
Sr. No.	-1.714157e-16
No. of Cars	1.000000e+00

In [145]:

```
prediction=lr.predict(x4_test)
plt.scatter(y4_test,prediction)
```

Out[145]:

```
<matplotlib.collections.PathCollection at 0x2a6285c4250>
```



In [146]:

```
print(lr.score(x4_test,y4_test))
```

```
1.0
```

In [147]:

```
lr.score(x4_train,y4_train)
```

Out[147]:

```
1.0
```

In [148]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [149]:

```
rr=Ridge(alpha=10)
rr.fit(x4_train,y4_train)
```

Out[149]:

```
Ridge(alpha=10)
```

In [150]:

```
rr.score(x4_test,y4_test)
```

Out[150]:

```
0.9968649065108404
```

In [151]:

```
la=Lasso(alpha=10)
la.fit(x4_train,y4_train)
```

Out[151]:

```
Lasso(alpha=10)
```

In [152]:

```
la.score(x4_test,y4_test)
```

Out[152]:

```
0.4887330385875225
```

In [153]:

```
from sklearn.linear_model import ElasticNet
p=ElasticNet()
p.fit(x4_train,y4_train)
```

Out[153]:

```
ElasticNet()
```

In [154]:

```
print(p.coef_)
```

```
[-0.0246848   0.90573469]
```

In [155]:

```
print(p.intercept_)
```

```
1.5399915791179613
```

In [156]:

```
print(p.predict(x4_test))
```

```
[15.80958346  8.19343401  8.21811881 15.63678989  8.63776033 15.78489867
 15.90832264 15.68615948]
```

In [157]:

```
print(p.score(x4_test,y4_test))
```

```
0.9932243333601749
```

In [158]:

```
from sklearn import metrics
```

In [159]:

```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y4_test,prediction))
```

```
Mean Absolytre Error: 7.771561172376096e-16
```

In [160]:

```
print("Mean Squared Error:",metrics.mean_squared_error(y4_test,prediction))
```

```
Mean Squared Error: 1.2818989709841442e-30
```

In [161]:

```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y4_test,prediction)))
```

```
Root Mean Squared Error: 1.1322097734007351e-15
```

In [162]:

```
import pickle
```

In [163]:

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

In [164]:

```
filename="prediction"
model=pickle.load(open(filename,'rb'))
```

In [165]:

```
real=[[5,8],[4,7]]
result=model.predict(real)
```

In [166]:

```
result
```

Out[166]:

```
array([8., 7.])
```

In [ ]:

```

```