LinearRegression

In [1]:

```
import numpy as np
import pandas as pd
```

data collection

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pp
import seaborn as sb
```

In [3]:

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

Out[3]:

	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

first 10 rows

In [4]:

```
df.head(10)
```

Out[4]:

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

data cleaning

In [5]:

```
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]:

```
df.describe()
```

Out[6]:

	Location.Cordinates.Latitude	Location.Cordinates.Longitude	Data.Magnitude.Body	Data.
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 [7]:

```
df.columns
```

Out[7]:

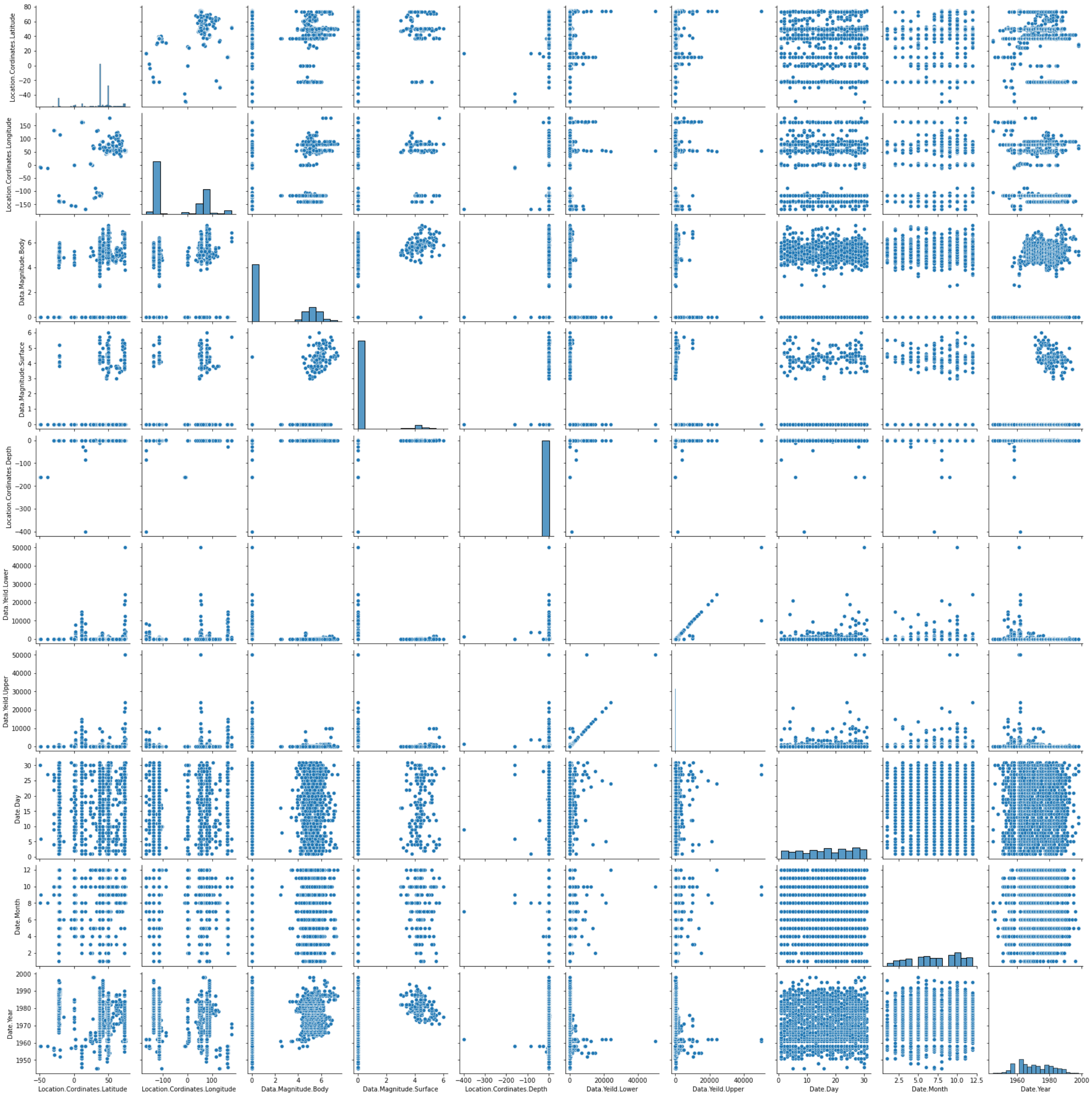
```
Index(['WEAPON SOURCE COUNTRY', 'WEAPON DEPLOYMENT LOCATION', 'Data.Sourc
e',
       'Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
       'Data.Magnitude.Body', 'Data.Magnitude.Surface',
       'Location.Cordinates.Depth', 'Data.Yeild.Lower', 'Data.Yeild.Uppe
r',
       'Data.Purpose',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

In [8]:

```
sb.pairplot(df)
```

Out[8]:

<seaborn.axisgrid.PairGrid at 0x24f1dc11f40>



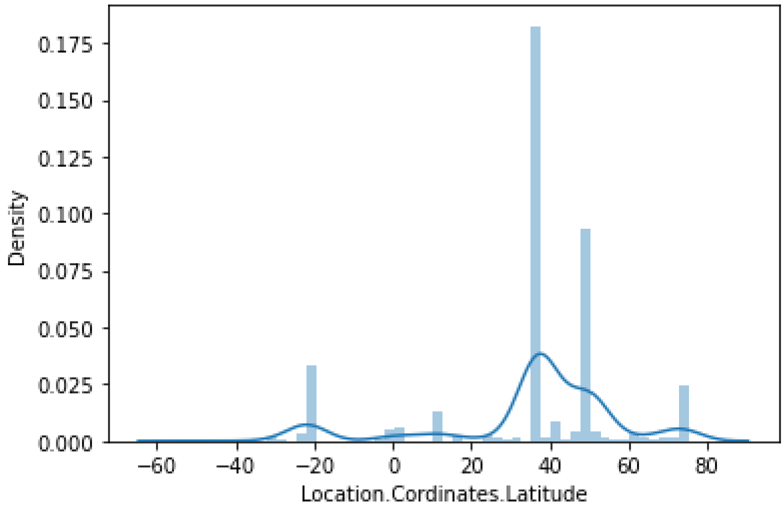
In [9]:

```
sb.distplot(df["Location.Cordinates.Latitude"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated 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 fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[9]:

```
<AxesSubplot:xlabel='Location.Cordinates.Latitude', ylabel='Density'>
```



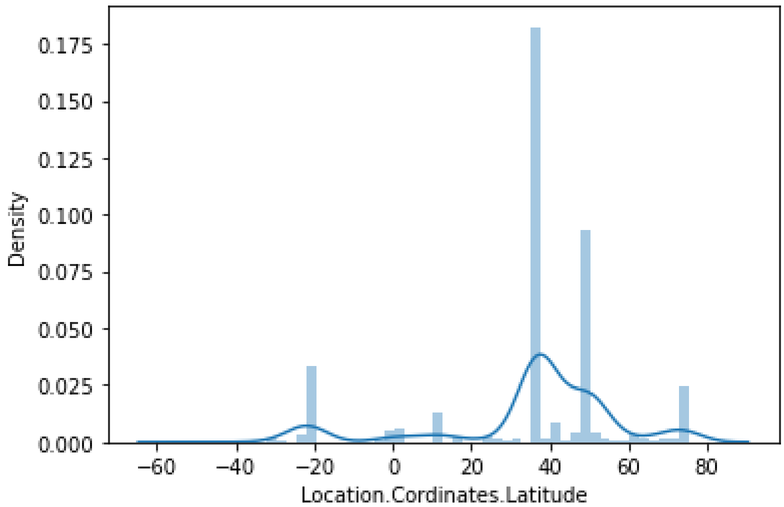
In [10]:

```
sb.distplot(df["Location.Cordinates.Latitude"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated 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 fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[10]:

```
<AxesSubplot:xlabel='Location.Cordinates.Latitude', ylabel='Density'>
```



In [14]:

```
df1=df[['Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
       'Data.Magnitude.Body', 'Data.Magnitude.Surface', 'Location.Cordinates.Depth', 'Dat
       'Date.Year']]
df1
```

Out[14]:

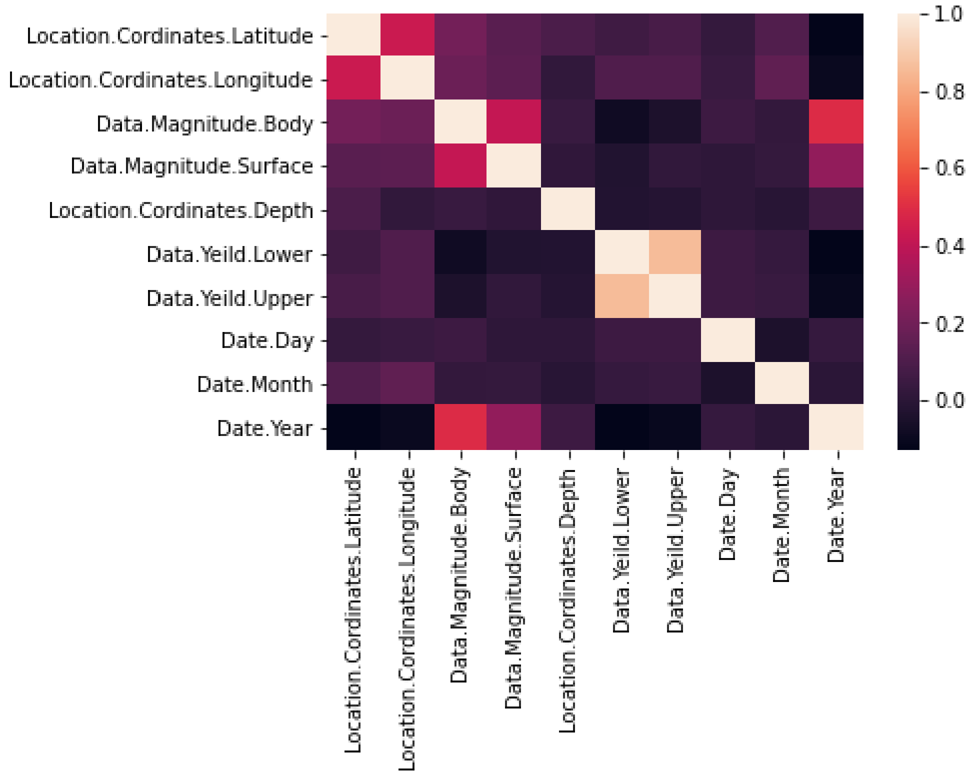
	Location.Cordinates.Latitude	Location.Cordinates.Longitude	Data.Magnitude.Body	Data.Magnitude.Surfa
0	32.54	-105.57	0.0	
1	34.23	132.27	0.0	
2	32.45	129.52	0.0	
3	11.35	165.20	0.0	
4	11.35	165.20	0.0	
...	...	...	...	
2041	41.69	88.35	5.3	
2042	27.07	71.70	5.3	
2043	27.07	71.70	0.0	
2044	28.90	64.89	0.0	

In [15]:

```
sb.heatmap(df1.corr())
```

Out[15]:

<AxesSubplot:>



model building

In [16]:

```
x = df1[['Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
       'Data.Magnitude.Body', 'Data.Magnitude.Surface', 'Location.Cordinates.Depth', 'Dat
       'Date.Year']]
y = df1['Location.Cordinates.Latitude']
```

In [17]:

```
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)
```

linear regression

In [18]:

```
from sklearn.linear_model import LinearRegression

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

Out[18]:

```
LinearRegression()
```

In [19]:

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

```
-3.410605131648481e-13
```

In [20]:

```
coef = pd.DataFrame(lr.coef_,x.columns,columns=['Co_efficient'])
coef
```

Out[20]:

	Co_efficient
Location.Cordinates.Latitude	1.000000e+00
Location.Cordinates.Longitude	-5.858187e-15
Data.Magnitude.Body	-2.086547e-16
Data.Magnitude.Surface	2.016738e-17
Location.Cordinates.Depth	-1.114603e-17
Data.Yeild.Lower	2.032691e-16
Data.Yeild.Upper	-7.307204e-17
Date.Day	9.820609e-17
Date.Month	7.480789e-17
Date.Year	4.965410e-17

In [21]:

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

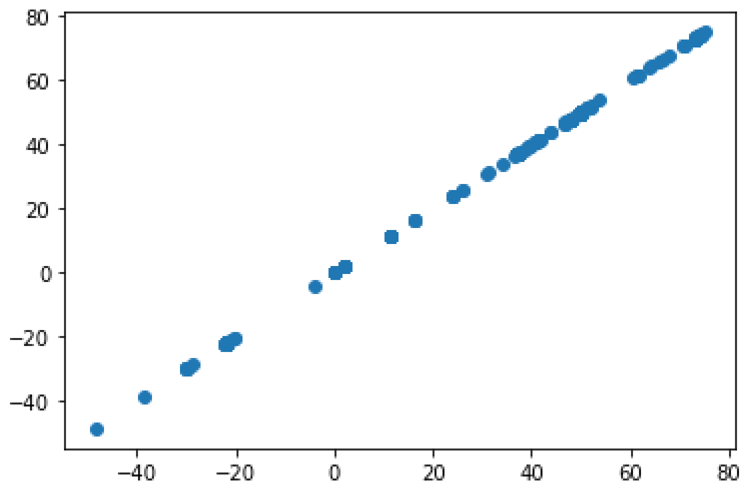
1.0

In [22]:

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

Out[22]:

<matplotlib.collections.PathCollection at 0x24f31233dc0>



lasso and ridge regression

In [23]:

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

Out[23]:

```
1.0
```

In [24]:

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

Out[24]:

```
1.0
```

In [25]:

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

In [26]:

```
r = Ridge(alpha=10)
r.fit(x_train,y_train)
r.score(x_test,y_test)
r.score(x_train,y_train)
```

Out[26]:

```
0.9999999997723793
```

In [27]:

```
l = Lasso(alpha=10)
l.fit(x_train,y_train)
l.score(x_test,y_test)
l.score(x_train,y_train)
```

Out[27]:

```
0.9996376447959459
```

elasticnet

In [28]:

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

Out[28]:

```
ElasticNet()
```

In [29]:

```
print(e.coef_)
```

```
[ 9.97782766e-01  1.85892832e-04  0.00000000e+00  0.00000000e+00
  0.00000000e+00 -1.63125578e-06  1.60397426e-06  0.00000000e+00
  0.00000000e+00 -0.00000000e+00]
```

In [30]:

```
print(e.intercept_)
```

```
0.08538452853202472
```

In [31]:

```
predictions = e.predict(x_test)
predictions
```

Out[31]:

```
array([ 11.3904701 ,  37.18133476,  37.08159366,  36.98181539,
        36.98181539,  49.98905456, -21.72380993,  49.70969211,
       -29.82374654,  37.08176956, -21.89143483,  37.12148955,
        49.85736192, -22.1488336 ,  36.98178301,  37.18831664,
        49.98923045,  72.93375016,  72.93375039,  36.98178287,
        49.80947225, -21.89166741,  36.68242971,  73.33512919,
        37.00176733,  36.98181539,  49.98902243,  36.98181539,
       -21.89166741,   0.08538453,  36.98181539,  37.08157507,
       -22.03808139,   2.05176216,  36.98181539,  36.98181539,
        37.11476395,  49.82944277,  49.98443328, -22.03128089,
       -20.24790934,  36.98181539,  37.08583103,  36.98181539,
        37.09182015,  36.98181539,  41.60991271,  11.44092785,
        49.98905456,  49.91753304,  36.98181539,  37.07081746,
        37.08159366,  47.83823664,  37.06388279, -21.89166741,
        36.9819692 ,  36.98181539, -21.89167543,  36.98181539,
        36.98181539,  47.79831844,  49.77972209,  72.93374374,
        36.98181539,  36.98181539,  37.20630164,  37.20141761,
        49.80950385,  36.98181539, -21.89164335,  49.74980534,
```

In [32]:

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

```
0.9999957947933726
```

In [33]:

```
from sklearn import metrics
```

mean absolute error

In [34]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,predictions))
```

```
Mean Absolute Error: 0.033244570489411394
```

mean squared error

In [35]:

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

```
Mean Squared Error: 0.0023184410144637953
```

root mean squared error

In [36]:

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

```
Root Mean Squared Error 0.04815019225780719
```

model saving

In [37]:

```
import pickle
```

In [38]:

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

In [39]:

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

In [41]:

```
real = [[10,20,30,40,50,60,70,80,90,100],[11,21,31,41,51,61,71,81,91,101]]
res = model.predict(real)
```

In [42]:

```
res
```

Out[42]:

```
array([10., 11.])
```

In [ ]:

```

```

LinearRegression

In [1]:

```
import numpy as np
import pandas as pd
```

data collection

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pp
import seaborn as sb
```

In [3]:

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

Out[3]:

	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	35.167
2	3875	Baghlan	1	AF	Afghanistan	BGL	NaN	36.1789
3	3884	Balkh	1	AF	Afghanistan	BAL	NaN	36.755(
4	3872	Bamyan	1	AF	Afghanistan	BAM	NaN	34.810(
...	...	...	...	...	...	...	...	
5072	1953	Mashonaland West Province	247	ZW	Zimbabwe	MW	NaN	-17.485
5073	1960	Masvingo Province	247	ZW	Zimbabwe	MV	NaN	-20.624
5074	1954	Matabeleland North Province	247	ZW	Zimbabwe	MN	NaN	-18.533
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

first 10 rows

In [4]:

```
df.head(10)
```

Out[4]:

	id	name	country_id	country_code	country_name	state_code	type	latitude
0	3901	Badakhshan	1	AF	Afghanistan	BDS	NaN	36.734772
1	3871	Badghis	1	AF	Afghanistan	BDG	NaN	35.167134
2	3875	Baghlan	1	AF	Afghanistan	BGL	NaN	36.178903
3	3884	Balkh	1	AF	Afghanistan	BAL	NaN	36.755060
4	3872	Bamyan	1	AF	Afghanistan	BAM	NaN	34.810007
5	3892	Daykundi	1	AF	Afghanistan	DAY	NaN	33.669495
6	3899	Farah	1	AF	Afghanistan	FRA	NaN	32.495328
7	3889	Faryab	1	AF	Afghanistan	FYB	NaN	36.079561
8	3870	Ghazni	1	AF	Afghanistan	GHA	NaN	33.545059
9	3888	Ghōr	1	AF	Afghanistan	GHO	NaN	34.099578

data cleaning

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5077 entries, 0 to 5076
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   id            5077 non-null   int64  
 1   name          5077 non-null   object 
 2   country_id    5077 non-null   int64  
 3   country_code  5063 non-null   object 
 4   country_name  5077 non-null   object 
 5   state_code    5072 non-null   object 
 6   type          1597 non-null   object 
 7   latitude      5008 non-null   float64
 8   longitude     5008 non-null   float64
dtypes: float64(2), int64(2), object(5)
memory usage: 357.1+ KB
```

In [6]:

```
df.describe()
```

Out[6]:

	id	country_id	latitude	longitude
count	5077.000000	5077.000000	5008.000000	5008.000000
mean	2609.765413	133.467599	27.576415	17.178713
std	1503.376799	72.341160	22.208161	61.269334
min	1.000000	1.000000	-54.805400	-178.116500
25%	1324.000000	74.000000	11.399747	-3.943859
50%	2617.000000	132.000000	34.226432	17.501792
75%	3905.000000	201.000000	45.802822	41.919647
max	5220.000000	248.000000	77.874972	179.852222

In [7]:

```
df.columns
```

Out[7]:

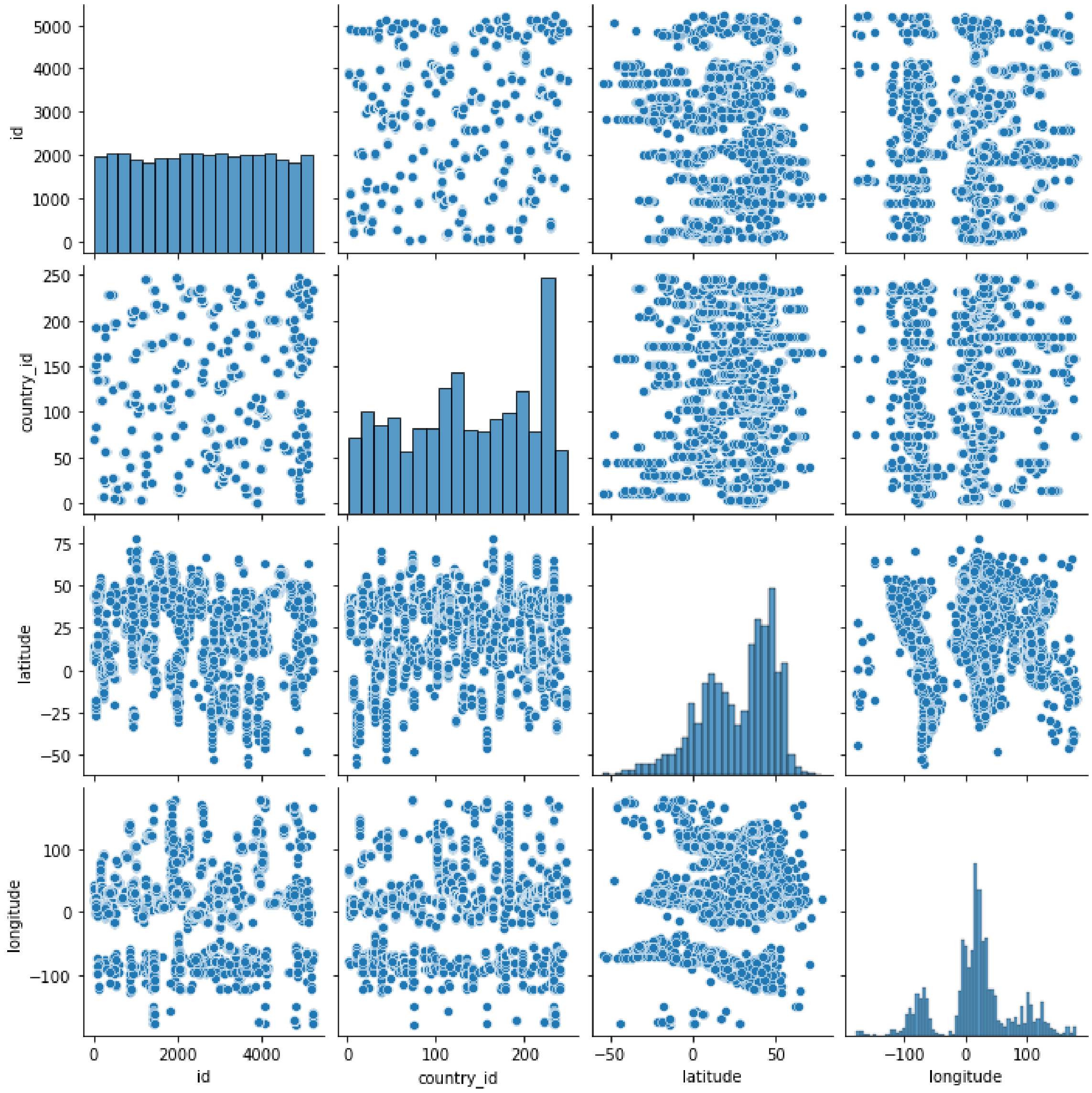
```
Index(['id', 'name', 'country_id', 'country_code', 'country_name',
       'state_code', 'type', 'latitude', 'longitude'],
      dtype='object')
```

In [8]:

```
sb.pairplot(df)
```

Out[8]:

```
<seaborn.axisgrid.PairGrid at 0x1e953d7d880>
```



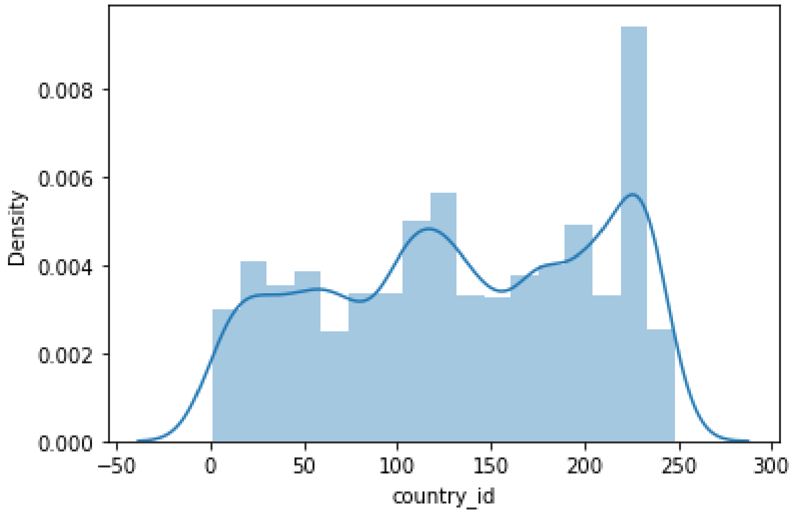
In [9]:

```
sb.distplot(df["country_id"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated 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 fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[9]:

<AxesSubplot:xlabel='country_id', ylabel='Density'>



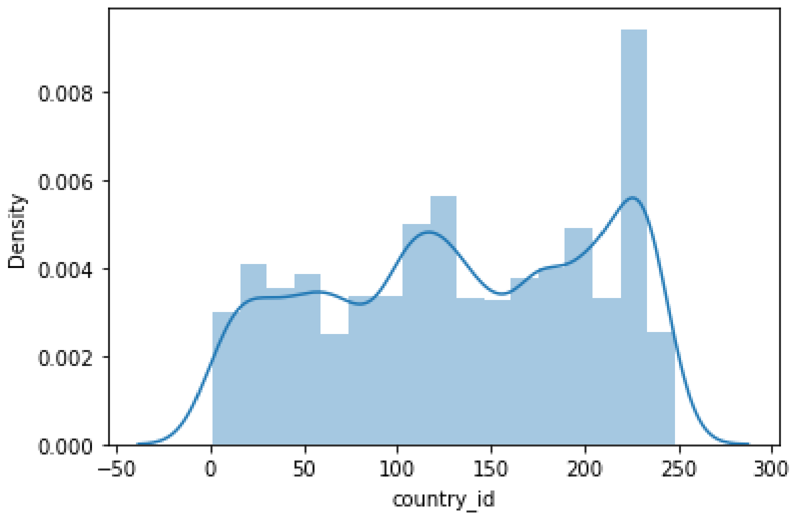
In [10]:

```
sb.distplot(df["country_id"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated 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 fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[10]:

<AxesSubplot:xlabel='country_id', ylabel='Density'>



In [12]:

```
df1=df[['id', 'country_id', 'latitude', 'longitude']]
df1
```

Out[12]:

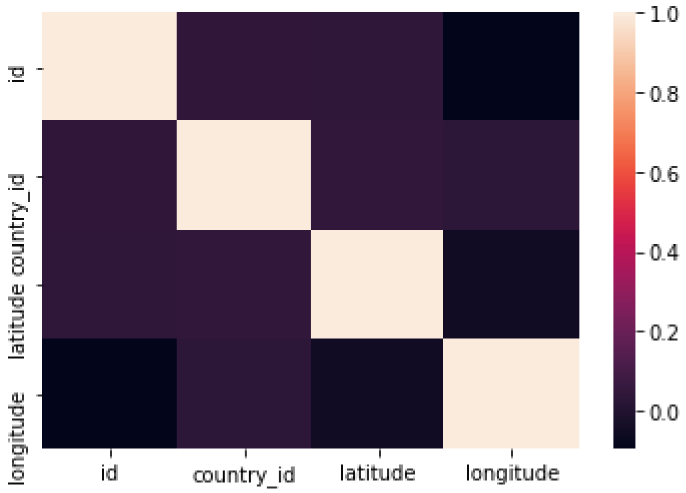
	id	country_id	latitude	longitude
0	3901	1	36.734772	70.811995
1	3871	1	35.167134	63.769538
2	3875	1	36.178903	68.745306
3	3884	1	36.755060	66.897537
4	3872	1	34.810007	67.821210
...	...	...	...	...
5072	1953	247	-17.485103	29.788925
5073	1960	247	-20.624151	31.262637
5074	1954	247	-18.533157	27.549585
5075	1952	247	-21.052337	29.045993

In [13]:

```
sb.heatmap(df1.corr())
```

Out[13]:

<AxesSubplot:>



model building

In [18]:

```
x = df1[['id', 'country_id']]
y = df1['country_id']
```

In [19]:

```
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)
```

linear regression

In [20]:

```
from sklearn.linear_model import LinearRegression

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

Out[20]:

```
LinearRegression()
```

In [21]:

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

```
-2.842170943040401e-14
```

In [22]:

```
coef = pd.DataFrame(lr.coef_,x.columns,columns=['Co_efficient'])
coef
```

Out[22]:

	Co_efficient
id	6.386082e-19
country_id	1.000000e+00

In [23]:

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

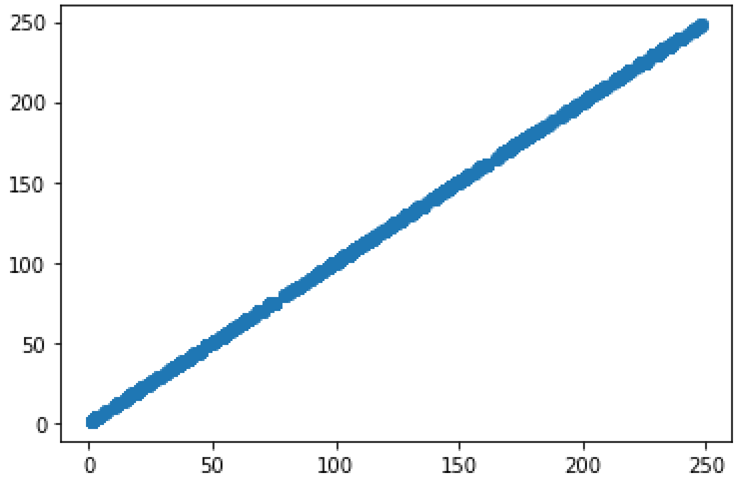
```
1.0
```

In [24]:

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

Out[24]:

<matplotlib.collections.PathCollection at 0x1e955ada3a0>



lasso and ridge regression

In [25]:

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

Out[25]:

1.0

In [26]:

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

Out[26]:

1.0

In [27]:

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

In [28]:

```
r = Ridge(alpha=10)
r.fit(x_train,y_train)
r.score(x_test,y_test)
r.score(x_train,y_train)
```

Out[28]:

0.9999999999997039

In [29]:

```
l = Lasso(alpha=10)
l.fit(x_train,y_train)
l.score(x_test,y_test)
l.score(x_train,y_train)
```

Out[29]:

```
0.9999962724110946
```

elasticnet

In [30]:

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

Out[30]:

```
ElasticNet()
```

In [31]:

```
print(e.coef_)
```

```
[2.64871170e-07 9.99806671e-01]
```

In [32]:

```
print(e.intercept_)
```

```
0.025228203323678144
```

In [33]:

```
predictions = e.predict(x_test)
predictions
```

Out[33]:

```
array([126.00128404,  85.00935198, 231.98103536, ..., 181.99053468,
       200.98749632, 181.99053256])
```

In [34]:

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

```
0.999999626288684
```

In [35]:

```
from sklearn import metrics
```

mean absolute error

In [36]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,predictions))
```

```
Mean Absolute Error: 0.012190200090642677
```

mean squared error

In [37]:

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

```
Mean Squared Error: 0.00020002042226479878
```

root mean squared error

In [38]:

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

```
Root Mean Squared Error 0.014142857641396197
```

model saving

In [39]:

```
import pickle
```

In [40]:

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

In [41]:

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

In [43]:

```
real = [[10,20],[11,21]]
res = model.predict(real)
```

In [44]:

```
res
```

Out[44]:

```
array([20., 21.])
```

In [ ]:

```

```

LinearRegression

In [1]:

```
import numpy as np
import pandas as pd
```

data collection

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pp
import seaborn as sb
```

In [3]:

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

Out[3]:

	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

first 10 rows

In [4]:

```
df.head(10)
```

Out[4]:

	id	name	state_id	state_code	state_name	country_id	country_code	country_nam
0	52	Ashkāsham	3901	BDS	Badakhshan	1	AF	Afghanista
1	68	Fayzabad	3901	BDS	Badakhshan	1	AF	Afghanista
2	78	Jurm	3901	BDS	Badakhshan	1	AF	Afghanista
3	84	Khandūd	3901	BDS	Badakhshan	1	AF	Afghanista
4	115	Rāghistān	3901	BDS	Badakhshan	1	AF	Afghanista
5	131	Wākhān	3901	BDS	Badakhshan	1	AF	Afghanista
6	72	Ghormach	3871	BDG	Badghis	1	AF	Afghanista
7	108	Qala i Naw	3871	BDG	Badghis	1	AF	Afghanista
8	54	Baghlān	3875	BGL	Baghlan	1	AF	Afghanista
9	140	Ḩukūmatī Dahanah-ye Ghōrī	3875	BGL	Baghlan	1	AF	Afghanista

data cleaning

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150454 entries, 0 to 150453
Data columns (total 11 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   id            150454 non-null  int64  
 1   name          150454 non-null  object 
 2   state_id      150454 non-null  int64  
 3   state_code    150129 non-null  object 
 4   state_name    150454 non-null  object 
 5   country_id    150454 non-null  int64  
 6   country_code  150406 non-null  object 
 7   country_name  150454 non-null  object 
 8   latitude      150454 non-null  float64
 9   longitude     150454 non-null  float64
 10  wikiDataId    147198 non-null  object 
dtypes: float64(2), int64(3), object(6)
memory usage: 12.6+ MB
```

In [6]:

```
df.describe()
```

Out[6]:

	id	state_id	country_id	latitude	longitude
count	150454.000000	150454.000000	150454.000000	150454.000000	150454.000000
mean	76407.091689	2678.377677	140.658460	31.556175	2.369557
std	44357.755335	1363.513591	70.666123	22.813220	68.012770
min	1.000000	1.000000	1.000000	-75.000000	-179.121980
25%	38160.250000	1451.000000	82.000000	19.000000	-58.468150
50%	75975.500000	2174.000000	142.000000	40.684720	8.669980
75%	115204.750000	3905.000000	207.000000	47.239220	27.750000
max	153528.000000	5116.000000	247.000000	73.508190	179.466000

In [7]:

```
df.columns
```

Out[7]:

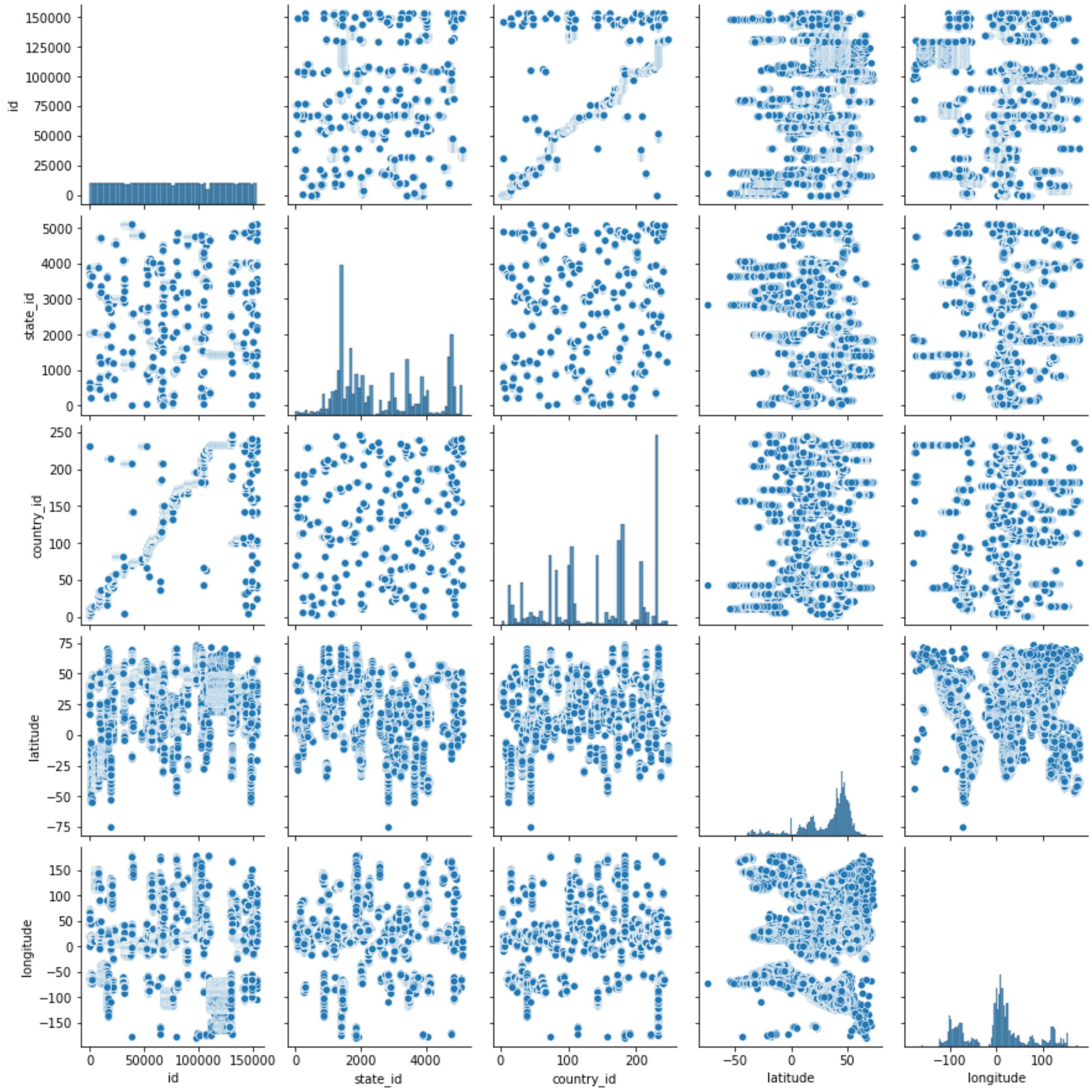
```
Index(['id', 'name', 'state_id', 'state_code', 'state_name', 'country_id',
       'country_code', 'country_name', 'latitude', 'longitude', 'wikiDataI
d'],
      dtype='object')
```

In [8]:

```
sb.pairplot(df)
```

Out[8]:

<seaborn.axisgrid.PairGrid at 0x2b5d3da99a0>



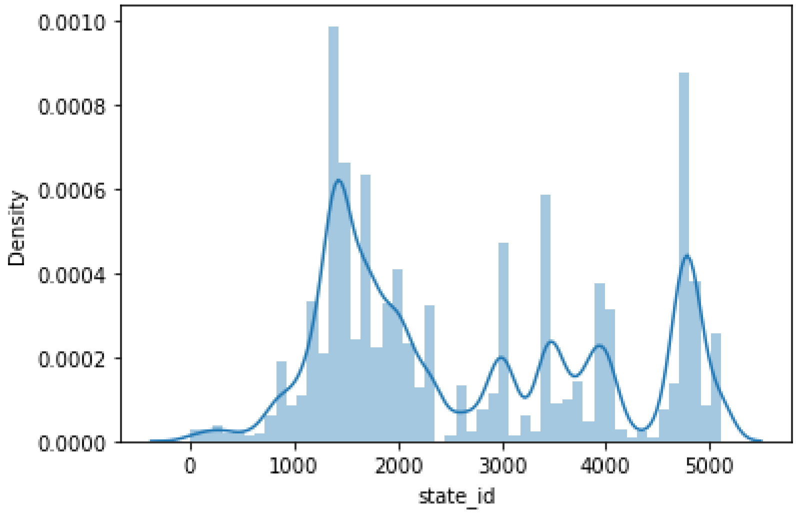
In [9]:

```
sb.distplot(df["state_id"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated 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 fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[9]:

```
<AxesSubplot:xlabel='state_id', ylabel='Density'>
```



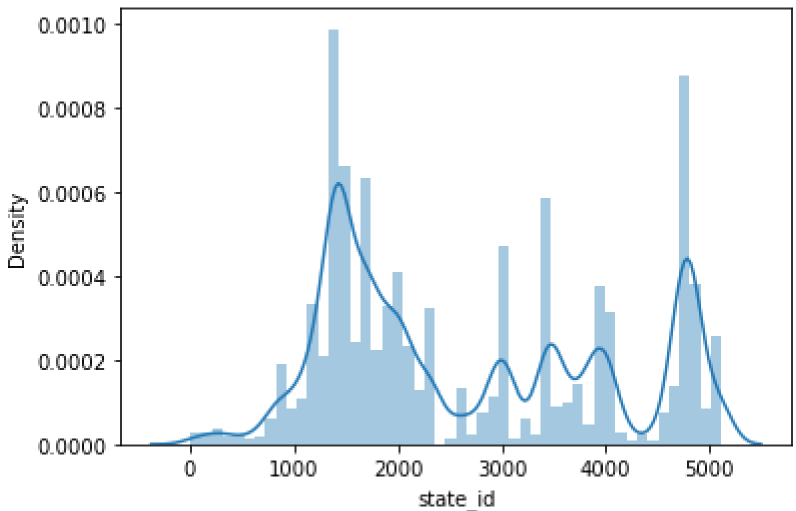
In [10]:

```
sb.distplot(df["state_id"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated 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 fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[10]:

```
<AxesSubplot:xlabel='state_id', ylabel='Density'>
```



In [11]:

```
df1=df[['id', 'state_id', 'country_id','latitude', 'longitude']]
df1
```

Out[11]:

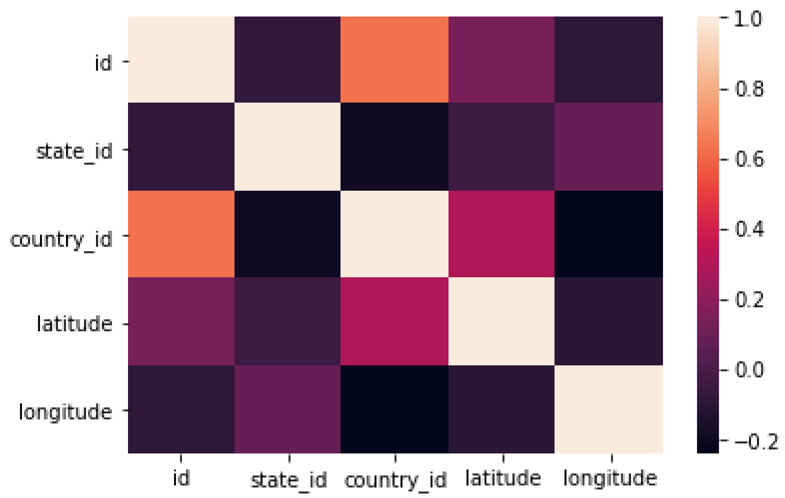
	id	state_id	country_id	latitude	longitude
0	52	3901	1	36.68333	71.53333
1	68	3901	1	37.11664	70.58002
2	78	3901	1	36.86477	70.83421
3	84	3901	1	36.95127	72.31800
4	115	3901	1	37.66079	70.67346
...	...	...	...	...	...
150449	131496	1957	247	-19.03333	29.78333
150450	131502	1957	247	-19.78333	29.36667
150451	131503	1957	247	-19.67016	30.00589
150452	131504	1957	247	-19.75000	30.16667

In [12]:

```
sb.heatmap(df1.corr())
```

Out[12]:

<AxesSubplot:>



model building

In [13]:

```
x = df1[['id', 'state_id', 'country_id','latitude', 'longitude']]
y = df1['state_id']
```

In [14]:

```
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)
```

linear regression

In [15]:

```
from sklearn.linear_model import LinearRegression

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

Out[15]:

```
LinearRegression()
```

In [16]:

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

```
1.5279510989785194e-10
```

In [17]:

```
coef = pd.DataFrame(lr.coef_,x.columns,columns=['Co_efficient'])
coef
```

Out[17]:

	Co_efficient
id	-1.925260e-15
state_id	1.000000e+00
country_id	-7.151336e-16
latitude	3.456362e-16
longitude	4.024150e-17

In [18]:

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

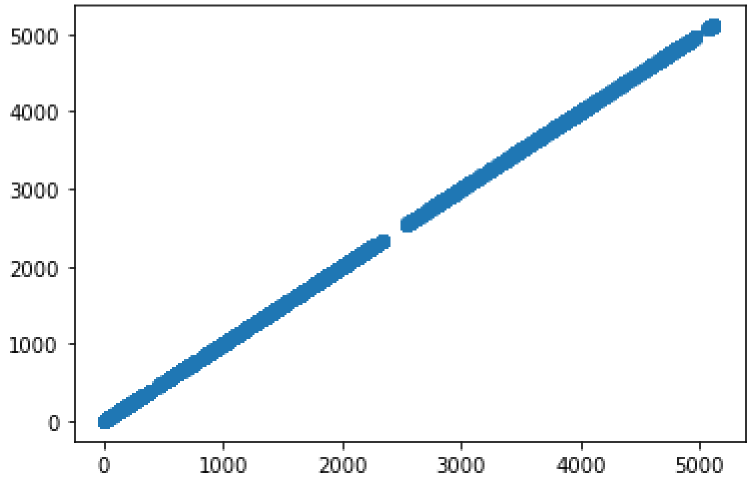
```
1.0
```

In [19]:

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

Out[19]:

```
<matplotlib.collections.PathCollection at 0x2b5d75ee6d0>
```



lasso and ridge regression

In [20]:

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

Out[20]:

```
1.0
```

In [21]:

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

Out[21]:

```
1.0
```

In [22]:

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

In [23]:

```
r = Ridge(alpha=10)
r.fit(x_train,y_train)
r.score(x_test,y_test)
r.score(x_train,y_train)
```

Out[23]:

```
1.0
```

In [24]:

```
l = Lasso(alpha=10)
l.fit(x_train,y_train)
l.score(x_test,y_test)
l.score(x_train,y_train)
```

Out[24]:

```
0.9999999999703485
```

elasticnet

In [25]:

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

Out[25]:

```
ElasticNet()
```

In [26]:

```
print(e.coef_)
```

```
[-3.36921916e-08  9.99999755e-01  0.00000000e+00  0.00000000e+00
 -0.00000000e+00]
```

In [27]:

```
print(e.intercept_)
```

```
0.0032245611428152188
```

In [28]:

```
predictions = e.predict(x_test)
predictions
```

Out[28]:

```
array([1677.99810224, 3397.99879472, 2058.00265841, ..., 1406.99859226,
       3449.99980644, 1702.99811499])
```

In [29]:

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

```
0.9999999999987775
```

In [30]:

```
from sklearn import metrics
```

mean absolute error

In [31]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,predictions))
```

```
Mean Absolute Error: 0.0012778790370394874
```

mean squared error

In [32]:

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

```
Mean Squared Error: 2.2794144201239746e-06
```

root mean squared error

In [33]:

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

```
Root Mean Squared Error 0.0015097729697288844
```

model saving

In [34]:

```
import pickle
```

In [35]:

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

In [36]:

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

In [37]:

```
real = [[10,20,30,40,50],[11,21,31,41,51]]
res = model.predict(real)
```

In [39]:

```
res
```

Out[39]:

```
array([20., 21.])
```

In [ ]:

```

```

LinearRegression

In [1]:

```
import numpy as np
import pandas as pd
```

data collection

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pp
import seaborn as sb
```

In [3]:

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

Out[3]:

	id	name	iso3	iso2	numeric_code	phone_code	capital	currency	currency
0	1	Afghanistan	AFG	AF	4	93	Kabul	AFN	Afghan
1	2	Aland Islands	ALA	AX	248	+358-18	Mariehamn	EUR	
2	3	Albania	ALB	AL	8	355	Tirana	ALL	Albai
3	4	Algeria	DZA	DZ	12	213	Algiers	DZD	Algeria
4	5	American Samoa	ASM	AS	16	+1-684	Pago Pago	USD	US
...	...	...	...	...	...	...	...	...	
245	243	Wallis And Futuna Islands	WLF	WF	876	681	Mata Utu	XPF	CF
246	244	Western Sahara	ESH	EH	732	212	El-Aaiun	MAD	Mc
247	245	Yemen	YEM	YE	887	967	Sanaa	YER	Yen
248	246	Zambia	ZMB	ZM	894	260	Lusaka	ZMW	Z I
249	247	Zimbabwe	ZWE	ZW	716	263	Harare	ZWL	Zin

250 rows × 19 columns

first 10 rows

In [4]:

```
df.head(10)
```

Out[4]:

	id	name	iso3	iso2	numeric_code	phone_code	capital	currency	currency_na
0	1	Afghanistan	AFG	AF	4	93	Kabul	AFN	Afghan afgh
1	2	Aland Islands	ALA	AX	248	+358-18	Mariehamn	EUR	E
2	3	Albania	ALB	AL	8	355	Tirana	ALL	Albanian
3	4	Algeria	DZA	DZ	12	213	Algiers	DZD	Algerian d
4	5	American Samoa	ASM	AS	16	+1-684	Pago Pago	USD	US Dc
5	6	Andorra	AND	AD	20	376	Andorra la Vella	EUR	E
6	7	Angola	AGO	AO	24	244	Luanda	AOA	Angolan kwa
7	8	Anguilla	AIA	AI	660	+1-264	The Valley	XCD	East Caribb dc
8	9	Antarctica	ATA	AQ	10	672	NaN	AAD	Antarcti dc
9	10	Antigua And Barbuda	ATG	AG	28	+1-268	St. John's	XCD	East Caribbean dc

data cleaning

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   id               250 non-null    int64  
 1   name             250 non-null    object 
 2   iso3             250 non-null    object 
 3   iso2             249 non-null    object 
 4   numeric_code     250 non-null    int64  
 5   phone_code       250 non-null    object 
 6   capital          245 non-null    object 
 7   currency         250 non-null    object 
 8   currency_name    250 non-null    object 
 9   currency_symbol  250 non-null    object 
 10  tld              250 non-null    object 
 11  native           249 non-null    object 
 12  region           248 non-null    object 
 13  subregion        247 non-null    object 
 14  timezones        250 non-null    object 
 15  latitude         250 non-null    float64
 16  longitude        250 non-null    float64
 17  emoji            250 non-null    object 
 18  emojiU           250 non-null    object 
dtypes: float64(2), int64(2), object(15)
memory usage: 37.2+ KB
```

In [6]:

```
df.describe()
```

Out[6]:

	id	numeric_code	latitude	longitude
count	250.000000	250.00000	250.000000	250.00000
mean	125.500000	435.80400	16.402597	13.52387
std	72.312977	254.38354	26.757204	73.45152
min	1.000000	4.00000	-74.650000	-176.20000
25%	63.250000	219.00000	1.000000	-49.75000
50%	125.500000	436.00000	16.083333	17.00000
75%	187.750000	653.50000	39.000000	48.75000
max	250.000000	926.00000	78.000000	178.00000

In [7]:

```
df.columns
```

Out[7]:

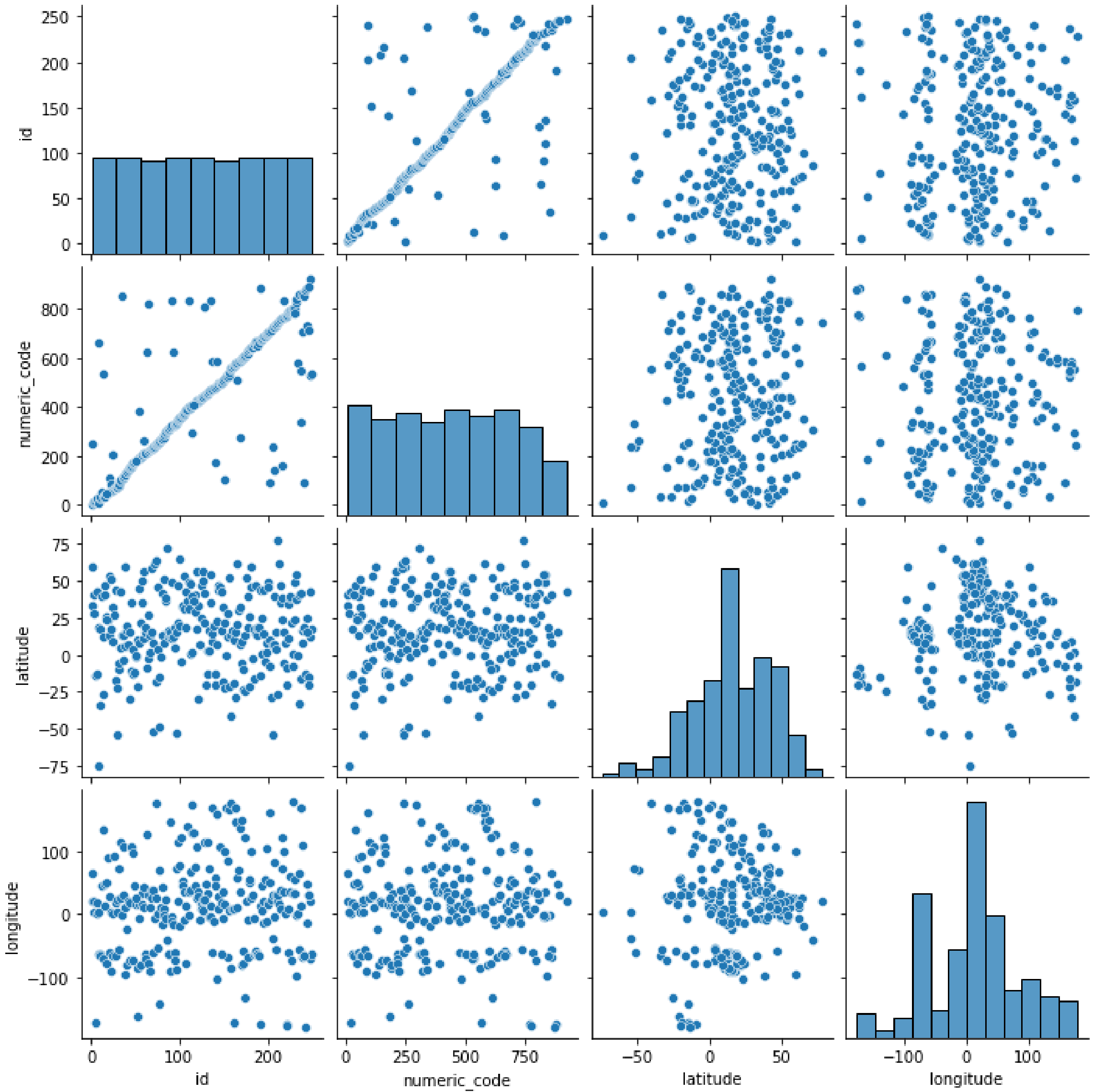
```
Index(['id', 'name', 'iso3', 'iso2', 'numeric_code', 'phone_code', 'capita
l',
       'currency', 'currency_name', 'currency_symbol', 'tld', 'native',
       'region', 'subregion', 'timezones', 'latitude', 'longitude', 'emoj
i',
       'emojiU'],
      dtype='object')
```

In [8]:

```
sb.pairplot(df)
```

Out[8]:

```
<seaborn.axisgrid.PairGrid at 0x204161cf760>
```



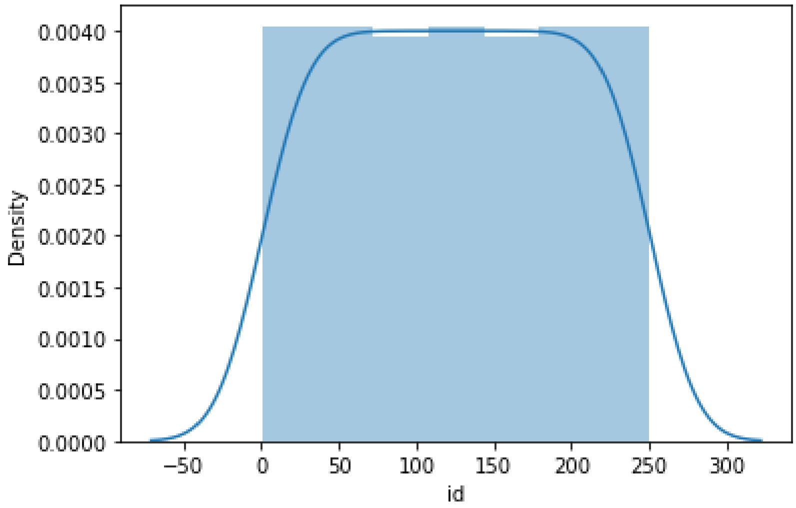
In [9]:

```
sb.distplot(df["id"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated 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 fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[9]:

<AxesSubplot:xlabel='id', ylabel='Density'>



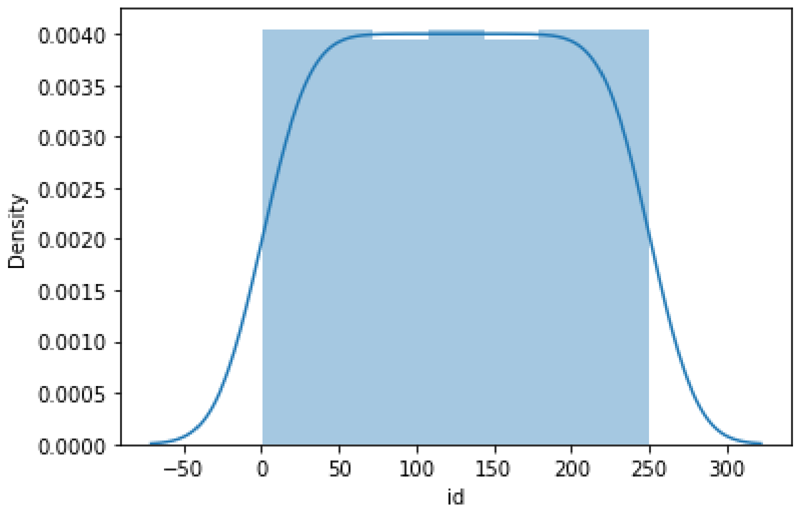
In [11]:

```
sb.distplot(df["id"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated 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 fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[11]:

<AxesSubplot:xlabel='id', ylabel='Density'>



In [12]:

```
df1=df[['id', 'numeric_code', 'latitude', 'longitude']]
df1
```

Out[12]:

	id	numeric_code	latitude	longitude
0	1	4	33.000000	65.0
1	2	248	60.116667	19.9
2	3	8	41.000000	20.0
3	4	12	28.000000	3.0
4	5	16	-14.333333	-170.0
...	...	...	...	...
245	243	876	-13.300000	-176.2
246	244	732	24.500000	-13.0
247	245	887	15.000000	48.0
248	246	894	-15.000000	30.0
249	247	716	-20.000000	30.0

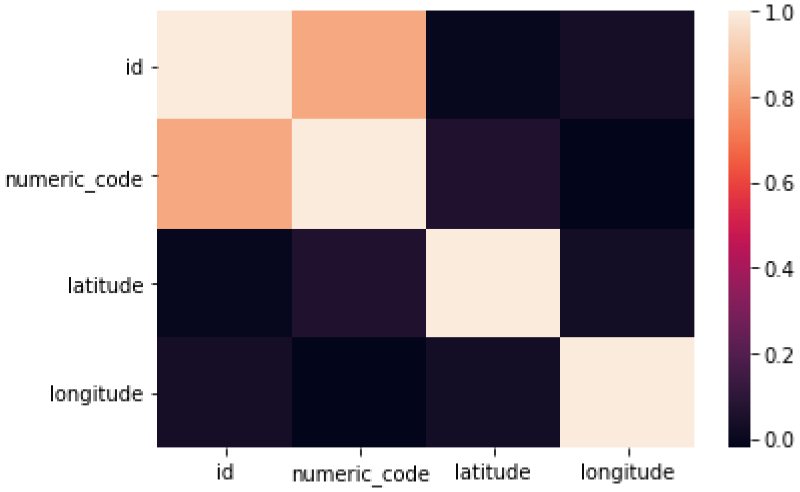
250 rows × 4 columns

In [13]:

```
sb.heatmap(df1.corr())
```

Out[13]:

```
<AxesSubplot:>
```



model building

In [14]:

```
x = df1[['id', 'numeric_code', 'latitude', 'longitude']]
y = df1['id']
```

In [15]:

```
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)
```

linear regression

In [16]:

```
from sklearn.linear_model import LinearRegression

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

Out[16]:

```
LinearRegression()
```

In [17]:

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

```
-8.526512829121202e-14
```

In [18]:

```
coef = pd.DataFrame(lr.coef_,x.columns,columns=['Co_efficient'])
coef
```

Out[18]:

	Co_efficient
id	1.000000e+00
numeric_code	-1.697410e-18
latitude	5.826981e-17
longitude	7.014811e-17

In [19]:

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

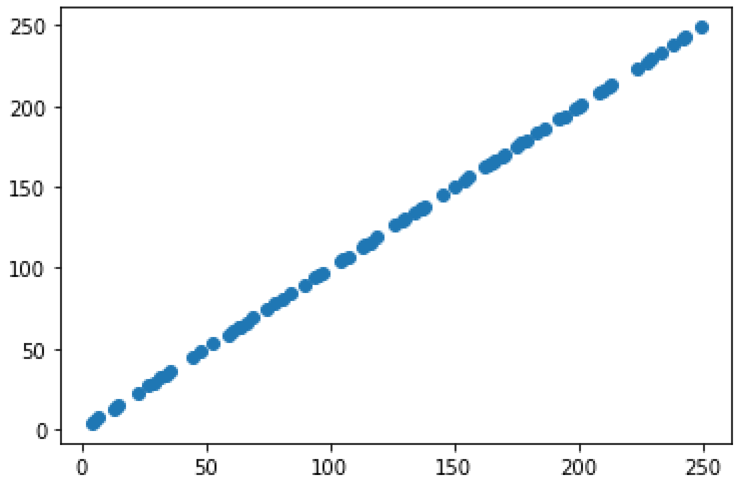
```
1.0
```

In [20]:

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

Out[20]:

```
<matplotlib.collections.PathCollection at 0x2041cd79fa0>
```



lasso and ridge regression

In [21]:

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

Out[21]:

```
1.0
```

In [22]:

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

Out[22]:

```
1.0
```

In [23]:

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

In [24]:

```
r = Ridge(alpha=10)
r.fit(x_train,y_train)
r.score(x_test,y_test)
r.score(x_train,y_train)
```

Out[24]:

```
0.9999999996355724
```

In [25]:

```
l = Lasso(alpha=10)
l.fit(x_train,y_train)
l.score(x_test,y_test)
l.score(x_train,y_train)
```

Out[25]:

```
0.9999938299822391
```

elasticnet

In [26]:

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

Out[26]:

```
ElasticNet()
```

In [27]:

```
print(e.coef_)
```

```
[ 9.99637967e-01  7.66449580e-05 -0.00000000e+00  0.00000000e+00]
```

In [28]:

```
print(e.intercept_)
```

```
0.0120095155180735
```

In [29]:

```
predictions = e.predict(x_test)
predictions
```

Out[29]:

```
array([144.99722406, 129.02715975,  23.01012094,  15.00964482,
       153.99641841, 113.00206436, 193.99404699,  74.00407374,
        48.00766158, 191.9941579 ,  63.03718119, 237.95159839,
       199.99348434, 212.99253351, 207.94774354, 136.02661829,
       126.00011715, 163.99709019, 197.9935186 ,  69.00488752,
        13.04815485, 129.99943547,  61.00617424,  84.00397908,
       168.97190332,  27.00744648, 232.99203761,   5.01142567,
        34.00736489, 222.99105924, 182.99527014,  36.00725399,
        14.00970027, 113.99342467,   7.01131476, 211.99258897,
       176.99544956, 137.00717166,  78.00369864, 165.99115427,
       155.99600092, 200.9932756 , 242.9911765 , 107.00239708,
         4.01148112, 119.00096519,  59.00659172,  45.00767465,
       149.99664022, 169.99576109,  66.00513052, 209.99269987,
       161.99704781, 228.99041995, 133.99921366,  95.00306251,
       104.00256344,  97.00295161, 164.99657487, 178.99579853,
        90.00395294,  53.00723102, 248.96256179,  64.00554801,
       185.99541036,  81.0032257 , 226.99083743,  32.00701593,
       241.98954577, 105.00250798, 174.99556047, 116.00143813,
        29.007029  , 137.99837868,  94.00311797])
```

In [30]:

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

```
0.9999999593560465
```

In [31]:

```
from sklearn import metrics
```

mean absolute error

In [32]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,predictions))
```

```
Mean Absolute Error: 0.009067409456770766
```

mean squared error

In [33]:

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

```
Mean Squared Error: 0.00020071042062233207
```

root mean squared error

In [34]:

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

```
Root Mean Squared Error 0.014167230520547481
```

model saving

In [35]:

```
import pickle
```

In [36]:

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

In [37]:

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

In [38]:

```
real = [[10,20,30,40],[11,21,31,41]]
res = model.predict(real)
```

In [39]:

```
res
```

Out[39]:

```
array([10., 11.])
```

In [ ]:

In [ ]:

LinearRegression

In [1]:

```
import numpy as np
import pandas as pd
```

data collection

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pp
import seaborn as sb
```

In [3]:

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

Out[3]:

	Sr. No.	Train Name	Train Number	Originating City	Originating Station	Terminal City
0	1	New Delhi - Varanasi Vande Bharat Express	22435/22436	Delhi	New Delhi	Varanasi
1	2	New Delhi - Shri Mata Vaishno Devi Katra Vande...	22439/22440	Delhi	New Delhi	Katra
2	3	Mumbai Central - Gandhinagar Capital Vande Bha...	20901/20902	Mumbai	Mumbai Central	Gandhinagar
3	4	New Delhi - Amb Andaura Vande Bharat Express	22447/22448	Delhi	New Delhi	Andaura
4	5	MGR Chennai Central - Mysuru Vande Bharat Express	20607/20608	Chennai	Chennai Central	Mysuru
5	6	Bilaspur - Nagpur Vande Bharat Express	20825/20826	Bilaspur, Chhattisgarh	Bilaspur Junction	Nagpur
6	7	Howrah - New Jalpaiguri Vande Bharat Express	22301/22302	Kolkata	Howrah Junction	Siliguri
7	8	Visakhapatnam - Secunderabad Vande Bharat Express	20833/20834	Visakhapatnam	Visakhapatnam Junction	Hyderabad
8	9	Mumbai CSMT - Solapur Vande Bharat Express	22225/22226	Mumbai	Chhatrapati Shivaji Terminus	Solapur
9	10	Mumbai CSMT - Sainagar Shirdi Vande Bharat Exp...	22223/22224	Mumbai	Chhatrapati Shivaji Terminus	Shirdi
10	11	Rani Kamalapati (Habibganj) - Hazrat Nizamuddi...	20171/20172	Bhopal	Habibganj (Rani Kamalapati)	Delhi
11	12	Secunderabad - Tirupati Vande Bharat Express	20701/20702	Hyderabad	Secunderabad Junction	Tirupati
12	13	MGR Chennai Central - Coimbatore Vande Bharat ...	20643/20644	Chennai	Chennai Central	Coimbatore
13	14	Delhi Cantonment - Ajmer Vande Bharat Express	20977/20978	Delhi	Delhi Cantonment	Ajmer
14	15	Kasaragod - Thiruvananthapuram Vande Bharat Ex...	20633/20634	Kasaragod	Kasaragod	Thiruvananthapuram
15	16	Howrah - Puri Vande Bharat Express	22895/22896	Kolkata	Howrah Junction	Puri

	Sr. No.	Train Name	Train Number	Originating City	Originating Station	Terminal City
16	17	Anand Vihar Terminal - Dehradun Vande Bharat E...	22457/22458	Delhi	Anand Vihar Terminal	Dehradun
17	18	New Jalpaiguri - Guwahati Vande Bharat Express	22227/22228	Siliguri	New Jalpaiguri Junction	Guwahati
18	19	Mumbai CSMT - Madgaon Vande Bharat Express	22229/22230	Mumbai	Chhatrapati Shivaji Terminus	Madgaon
19	19	Mumbai CSMT - Madgaon Vande Bharat Express	22229/22230	Mumbai	Chhatrapati Shivaji Terminus	Madgaon
20	20	Patna - Ranchi Vande Bharat Express	22349/22350	Patna	Patna Junction	Ranchi
21	21	KSR Bengaluru - Dharwad Vande Bharat Express	20661/20662	Bangalore	Bangalore City	Hubbali - Dharwad
22	22	Rani Kamalapati (Habibganj) - Jabalpur Vande B...	20173/20174	Bhopal	Habibganj (Rani Kamalapati)	Jabalpur
23	23	Indore - Bhopal Vande Bharat Express	20911/20912	Indore	Indore Junction	Bhopal
24	24	Jodhpur - Sabarmati (Ahmedabad) Vande Bharat E...	12461/12462	Jodhpur	Jodhpur Junction	Ahmedabad
25	25	Gorakhpur - Lucknow Charbagh Vande Bharat Express	22549/22550	Gorakhpur	Gorakhpur Junction	Charbagh

first 10 rows

In [4]:

```
df.head(10)
```

Out[4]:

	Sr. No.	Train Name	Train Number	Originating City	Originating Station	Terminal City	Terminal Station
0	1	New Delhi - Varanasi Vande Bharat Express	22435/22436	Delhi	New Delhi	Varanasi	Varanasi Junction
1	2	New Delhi - Shri Mata Vaishno Devi Katra Vande...	22439/22440	Delhi	New Delhi	Katra	Shri Mata Vaishno Devi Katra
2	3	Mumbai Central - Gandhinagar Capital Vande Bha...	20901/20902	Mumbai	Mumbai Central	Gandhinagar	Gandhinagar Capital
3	4	New Delhi - Amb Andaura Vande Bharat Express	22447/22448	Delhi	New Delhi	Andaura	Amb Andaura
4	5	MGR Chennai Central - Mysuru Vande Bharat Express	20607/20608	Chennai	Chennai Central	Mysuru	Mysore Junction
5	6	Bilaspur - Nagpur Vande Bharat Express	20825/20826	Bilaspur, Chhattisgarh	Bilaspur Junction	Nagpur	Nagpur Junction
6	7	Howrah - New Jalpaiguri Vande Bharat Express	22301/22302	Kolkata	Howrah Junction	Siliguri	New Jalpaiguri Junction
7	8	Visakhapatnam - Secunderabad Vande Bharat Express	20833/20834	Visakhapatnam	Visakhapatnam Junction	Hyderabad	Secunderabad Junction
8	9	Mumbai CSMT - Solapur Vande Bharat Express	22225/22226	Mumbai	Chhatrapati Shivaji Terminus	Solapur	Solapur
9	10	Mumbai CSMT - Sainagar Shirdi Vande Bharat Exp...	22223/22224	Mumbai	Chhatrapati Shivaji Terminus	Shirdi	Sainagar Shirdi

data cleaning

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26 entries, 0 to 25
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   Sr. No.              26 non-null     int64 
 1   Train Name           26 non-null     object
 2   Train Number         26 non-null     object
 3   Originating City     26 non-null     object
 4   Originating Station  26 non-null     object
 5   Terminal City        26 non-null     object
 6   Terminal Station     26 non-null     object
 7   Operator             26 non-null     object
 8   No. of Cars          26 non-null     int64 
 9   Frequency            26 non-null     object
 10  Distance             26 non-null     object
 11  Travel Time          26 non-null     object
 12  Speed                26 non-null     object
 13  Average Speed        26 non-null     object
 14  Inauguration         26 non-null     object
 15  Average occupancy    26 non-null     object
dtypes: int64(2), object(14)
memory usage: 3.4+ KB
```

In [6]:

```
df.describe()
```

Out[6]:

	Sr. No.	No. of Cars
count	26.000000	26.000000
mean	13.230769	12.923077
std	7.306478	3.969112
min	1.000000	8.000000
25%	7.250000	8.000000
50%	13.500000	16.000000
75%	19.000000	16.000000
max	25.000000	16.000000

In [7]:

```
df.columns
```

Out[7]:

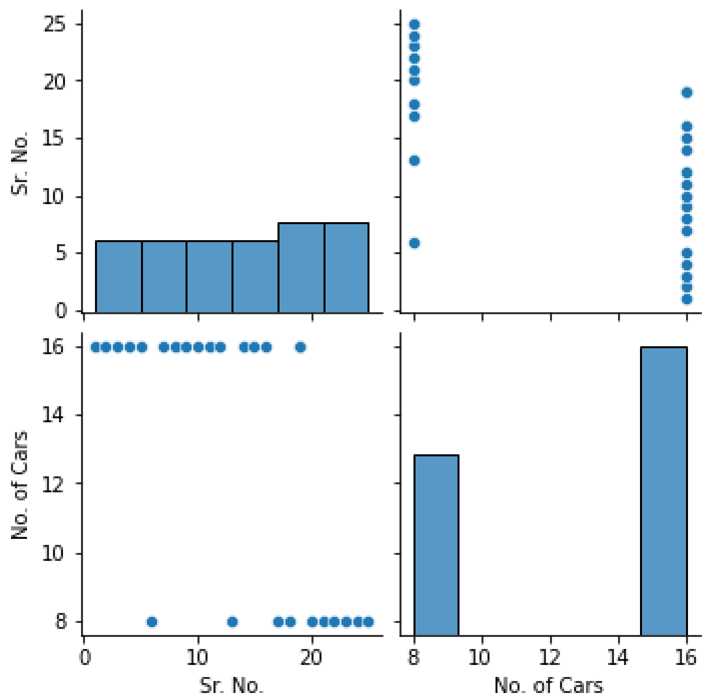
```
Index(['Sr. No.', 'Train Name', 'Train Number', 'Originating City',
       'Originating Station', 'Terminal City', 'Terminal Station', 'Operat
or',
       'No. of Cars', 'Frequency', 'Distance', 'Travel Time', 'Speed',
       'Average Speed', 'Inauguration', 'Average occupancy'],
      dtype='object')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

In [8]:

```
sb.pairplot(df)
```

Out[8]:

```
<seaborn.axisgrid.PairGrid at 0x1cc7ede7a90>
```



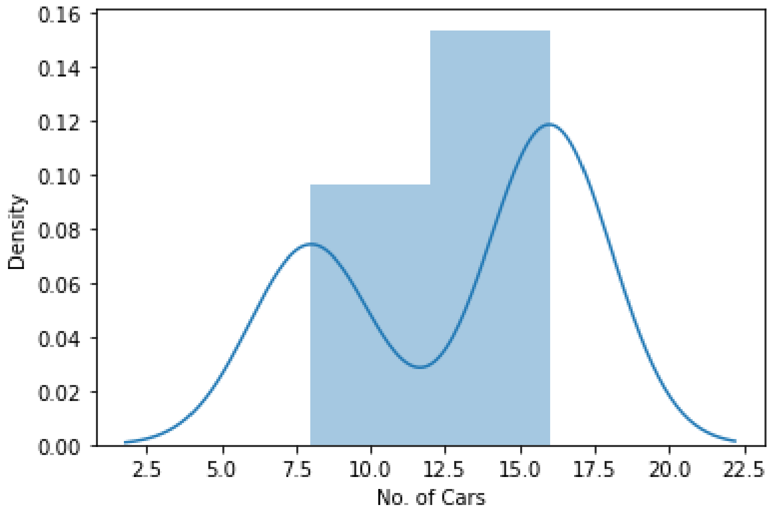
In [9]:

```
sb.distplot(df["No. of Cars"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated 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 fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[9]:

```
<AxesSubplot:xlabel='No. of Cars', ylabel='Density'>
```



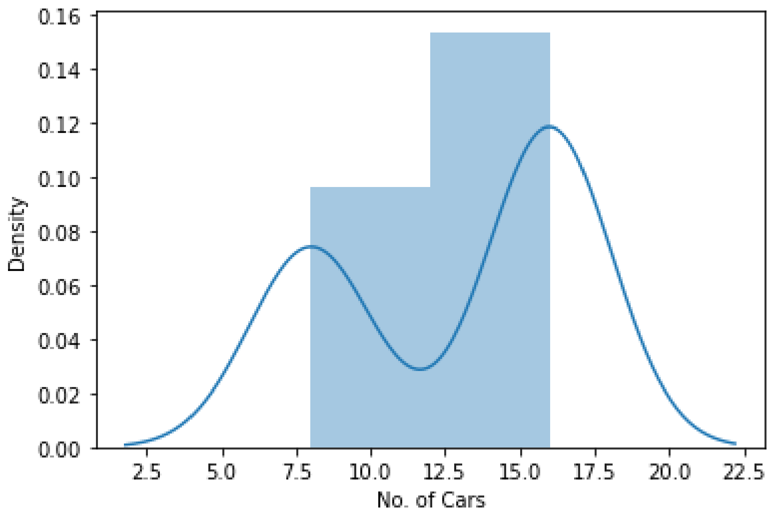
In [10]:

```
sb.distplot(df["No. of Cars"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated 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 fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[10]:

```
<AxesSubplot:xlabel='No. of Cars', ylabel='Density'>
```



In [11]:

```
df1=df[['Sr. No.','No. of Cars']]
df1
```

Out[11]:

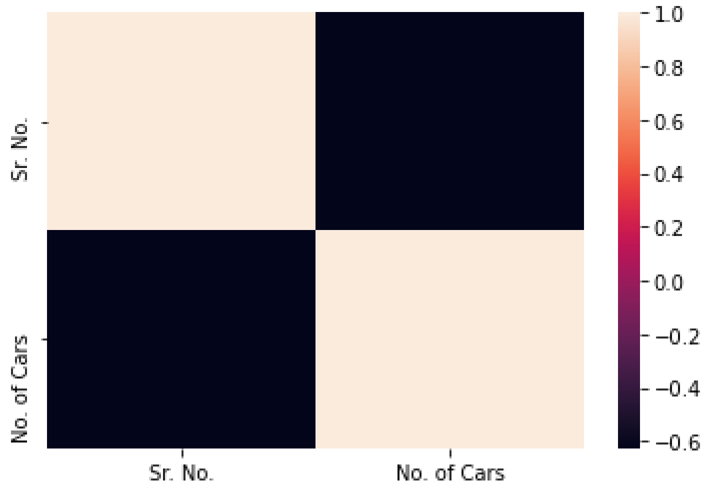
	Sr. No.	No. of Cars
0	1	16
1	2	16
2	3	16
3	4	16
4	5	16
5	6	8
6	7	16
7	8	16
8	9	16
9	10	16

In [12]:

```
sb.heatmap(df1.corr())
```

Out[12]:

<AxesSubplot:>



model building

In [13]:

```
x = df1[['Sr. No.','No. of Cars']]
y = df1['No. of Cars']
```

In [14]:

```
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)
```

linear regression

In [15]:

```
from sklearn.linear_model import LinearRegression

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

Out[15]:

```
LinearRegression()
```

In [16]:

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

```
-1.7763568394002505e-15
```

In [17]:

```
coef = pd.DataFrame(lr.coef_,x.columns,columns=['Co_efficient'])
coef
```

Out[17]:

	Co_efficient
Sr. No.	1.231353e-16
No. of Cars	1.000000e+00

In [18]:

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

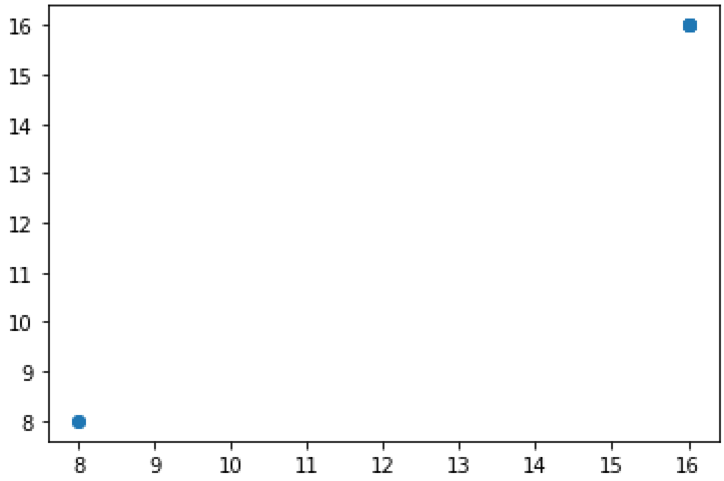
```
1.0
```

In [19]:

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

Out[19]:

```
<matplotlib.collections.PathCollection at 0x1cc01b165b0>
```



lasso and ridge regression

In [20]:

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

Out[20]:

```
1.0
```

In [21]:

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

Out[21]:

```
1.0
```

In [22]:

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

In [23]:

```
r = Ridge(alpha=10)
r.fit(x_train,y_train)
r.score(x_test,y_test)
r.score(x_train,y_train)
```

Out[23]:

```
0.9980734406970261
```

In [24]:

```
l = Lasso(alpha=10)
l.fit(x_train,y_train)
l.score(x_test,y_test)
l.score(x_train,y_train)
```

Out[24]:

```
0.5949084705330767
```

elasticnet

In [25]:

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

Out[25]:

```
ElasticNet()
```

In [26]:

```
print(e.coef_)
```

```
[-0.02240737  0.91395295]
```

In [27]:

```
print(e.intercept_)
```

```
1.3720624107356176
```

In [28]:

```
predictions = e.predict(x_test)
predictions
```

Out[28]:

```
array([ 8.30276069, 15.97290221, 15.56956953, 15.72642113,  8.14590909,
       15.95049484, 15.81605061, 15.56956953])
```

In [29]:

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

```
0.9937983114525036
```

In [30]:

```
from sklearn import metrics
```

mean absolute error

In [31]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,predictions))
```

```
Mean Absolute Error: 0.2304577411441866
```

mean squared error

In [32]:

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

```
Mean Squared Error: 0.0744202625699561
```

root mean squared error

In [33]:

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

```
Root Mean Squared Error 0.27280077450395207
```

model saving

In [34]:

```
import pickle
```

In [35]:

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

In [36]:

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

In [40]:

```
real = [[49,76],[43,65]]
res = model.predict(real)
```

In [41]:

```
res
```

Out[41]:

```
array([76., 65.])
```

In [ ]:

```

```