Importing libraries

In [1]:

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

Importing dataset

In [2]:

```
data=pd.read_csv(r"C:\Users\user\Downloads\states.csv")
data
```

Out[2]:

	id	name	country_id	country_code	country_name	state_code	type	latitude	longit
0	3901	Badakhshan	1	AF	Afghanistan	BDS	NaN	36.734772	70.811
1	3871	Badghis	1	AF	Afghanistan	BDG	NaN	35.167134	63.769
2	3875	Baghlan	1	AF	Afghanistan	BGL	NaN	36.178903	68.745
3	3884	Balkh	1	AF	Afghanistan	BAL	NaN	36.755060	66.897
4	3872	Bamyan	1	AF	Afghanistan	BAM	NaN	34.810007	67.821
...	...	...	...	...	...	...	...	...	
5072	1953	Mashonaland West Province	247	ZW	Zimbabwe	MW	NaN	-17.485103	29.788
5073	1960	Masvingo Province	247	ZW	Zimbabwe	MV	NaN	-20.624151	31.262
5074	1954	Matabeleland North Province	247	ZW	Zimbabwe	MN	NaN	-18.533157	27.549
5075	1952	Matabeleland South Province	247	ZW	Zimbabwe	MS	NaN	-21.052337	29.045
5076	1957	Midlands Province	247	ZW	Zimbabwe	MI	NaN	-19.055201	29.603

5077 rows × 9 columns

info

In [3]:

```
# to identify missing values
data.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
```

describe

In [4]:

```
# to display summary of the dataset
data.describe()
```

Out[4]:

	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

columns

In [5]:

```
# to display headings of the dataset
data.columns
```

Out[5]:

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

In [6]:

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

Out[6]:

	id	name	country_id	country_name
0	3901	Badakhshan	1	Afghanistan
1	3871	Badghis	1	Afghanistan

	id	name	country_id	country_name
2	3875	Baghlan	1	Afghanistan
3	3884	Balkh	1	Afghanistan
4	3872	Bamyan	1	Afghanistan
...	...	...	...	...
5072	1953	Mashonaland West Province	247	Zimbabwe
5073	1960	Masvingo Province	247	Zimbabwe
5074	1954	Matabeleland North Province	247	Zimbabwe
5075	1952	Matabeleland South Province	247	Zimbabwe
5076	1957	Midlands Province	247	Zimbabwe

5077 rows × 4 columns

In [7]:

```
a.columns
```

Out[7]:

```
Index(['id', 'name', 'country_id', 'country_name'], dtype='object')
```

To train the model-Model Building

In [8]:

```
x=a[[ 'id']]
y=a['country_id']
```

In [9]:

```
# to split my dataset into training and test data
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 [10]:

```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

Out[10]:

```
LinearRegression()
```

In [11]:

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

```
129.2348128470479
```

In [12]:

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

Out[12]:

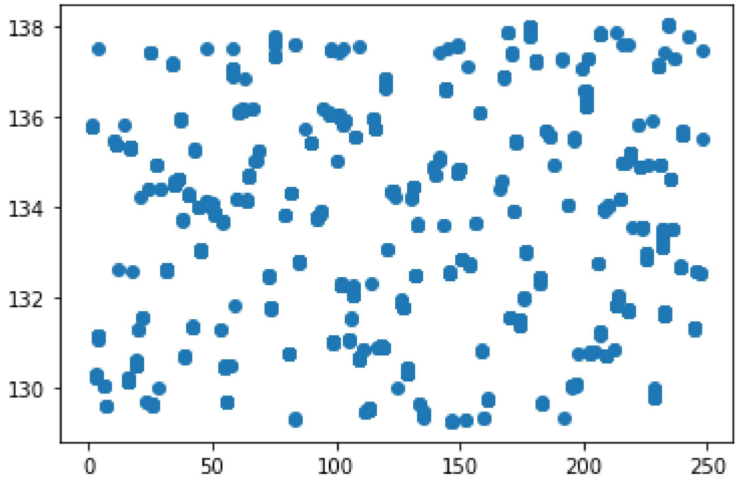
	Co-efficient
id	0.001689

In [13]:

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

Out[13]:

```
<matplotlib.collections.PathCollection at 0x23e1de5dfa0>
```



In [14]:

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

```
0.0031123102570940198
```

In [15]:

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

Out[15]:

```
0.0012766012503980795
```

Ridge regression

In [16]:

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

In [17]:

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

Out[17]:

```
0.0031123102545747017
```

In [18]:

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

Out[18]:

```
0.0012766012503983015
```

Lasso regression

In [19]:

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

Out[19]:

```
0.0012765926958366869
```

In [20]:

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

Out[20]:

```
0.0031070024342493285
```

Elastic net regression

In [21]:

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

Out[21]:

```
ElasticNet()
```

In [22]:

```
print(en.coef_)
```

```
[0.00168872]
```

In [23]:

```
print(en.intercept_)
```

```
129.23538595495046
```

In [24]:

```
predict=en.predict(x_test)
```

In [25]:

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

```
0.0031120447798368422
```

In [26]:

```
from sklearn import metrics
```

In [27]:

```
print("Mean Absolute error:",metrics.mean_absolute_error(y_test,predict))
```

```
Mean Absolute error: 64.41393523458736
```

In [28]:

```
print("Mean Squared error:",metrics.mean_squared_error(y_test,predict))
```

```
Mean Squared error: 5497.840089572804
```

In [29]:

```
print("Root squared error:",np.sqrt(metrics.mean_squared_error(y_test,predict)))
```

```
Root squared error: 74.1474213278709
```

Model saving

In [30]:

```
import pickle
filename="prediction"
pickle.dump(lr,open(filename,'wb'))
filename='prediction'
model=pickle.load(open(filename,'rb'))
```

In [31]:

```
real=[[10],[7]]
result=model.predict(real)
result
```

Out[31]:

```
array([129.25170228, 129.24663545])
```

In [ ]: