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\countries.csv")
data
```

Out[2]:

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

250 rows × 19 columns

info

In [3]:

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

describe

In [4]:

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

Out[4]:

	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

columns

In [5]:

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

Out[5]:

```
Index(['id', 'name', 'iso3', 'iso2', 'numeric_code', 'phone_code', 'capital',
       'currency', 'currency_name', 'currency_symbol', 'tld', 'native',
```

```
       'region', 'subregion', 'timezones', 'latitude', 'longitude', 'emoji',
       'emojiU'],
      dtype='object')
```

In [6]:

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

Out[6]:

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

250 rows × 14 columns

In [7]:

```
a.columns
```

Out[7]:

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

To train the model-Model Building

In [8]:

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

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_)
```

63.25062759177405

In [12]:

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

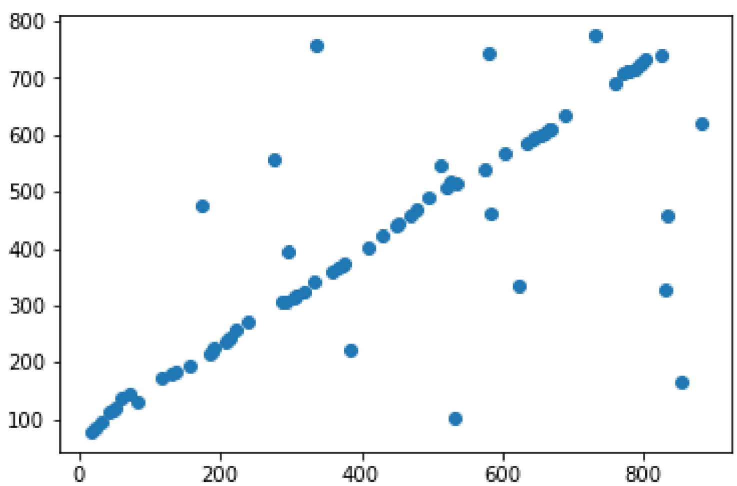
Out[12]:

	Co-efficient
id	2.91627

In [13]:

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

Out[13]: <matplotlib.collections.PathCollection at 0x1b5e3163880>



In [14]:

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

0.6372361378282494

In [15]:

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

Out[15]: 0.6961084435082606

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]: 0.6372364813086897

In [18]:
```
rr.score(x_train,y_train)
```

Out[18]: 0.6961084434262881

Lasso regression

In [19]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
la.score(x_train,y_train)
```

Out[19]: 0.6961081483205123

In [20]:
```
la.score(x_test,y_test)
```

Out[20]: 0.6372564707576569

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_)
```

[2.91589853]

In [23]:
```
print(en.intercept_)
```

63.29812785145532

In [24]:
```
predict=en.predict(x_test)
```

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

```
0.6372401635756959
```

In [26]:
```
from sklearn import metrics
```

In [27]:
```
print("Mean Absolute error:",metrics.mean_absolute_error(y_test,predict))
```

```
Mean Absolute error: 86.85101971005366
```

In [28]:
```
print("Mean Squared error:",metrics.mean_squared_error(y_test,predict))
```

```
Mean Squared error: 23480.69637831175
```

In [29]:
```
print("Root squared error:",np.sqrt(metrics.mean_squared_error(y_test,predict)))
```

```
Root squared error: 153.23412276092995
```

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([92.41333119, 83.66452011])
```

In [ ]: