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

In [2]:
```
d=pd.read_csv(r"C:\Users\user\Downloads\22_countries - 22_countries.csv")
d
```

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

In [3]:
```
d.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 [4]:
```
d.columns
```

Out[4]:
```
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 [5]:
```
d1=d.head(100)
d1
```

Out[5]:

	id	name	iso3	iso2	numeric_code	phone_code	capital	currency	currency_name	curr
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	
...	...	...	...	...	...	...	...	...	...	
95	95	Haiti	HTI	HT	332	509	Port-au-Prince	HTG	Haitian gourde	
96	96	Heard Island and McDonald Islands	HMD	HM	334	672	NaN	AUD	Australian dollar	
97	97	Honduras	HND	HN	340	504	Tegucigalpa	HNL	Honduran lempira	
98	98	Hong Kong S.A.R.	HKG	HK	344	852	Hong Kong	HKD	Hong Kong dollar	
99	99	Hungary	HUN	HU	348	36	Budapest	HUF	Hungarian	

id	name	iso3	iso2	numeric_code	phone_code	capital	currency	currency_name	curre
								forint	

100 rows × 19 columns

In [6]:
```
d1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   id               100 non-null    int64  
 1   name             100 non-null    object 
 2   iso3             100 non-null    object 
 3   iso2             100 non-null    object 
 4   numeric_code     100 non-null    int64  
 5   phone_code       100 non-null    object 
 6   capital          97 non-null     object 
 7   currency         100 non-null    object 
 8   currency_name    100 non-null    object 
 9   currency_symbol  100 non-null    object 
 10  tld              100 non-null    object 
 11  native           99 non-null     object 
 12  region           98 non-null     object 
 13  subregion        97 non-null     object 
 14  timezones        100 non-null    object 
 15  latitude         100 non-null    float64
 16  longitude        100 non-null    float64
 17  emoji            100 non-null    object 
 18  emojiU           100 non-null    object 
dtypes: float64(2), int64(2), object(15)
memory usage: 15.0+ KB
```

In [22]:
```
x=d1[['id','latitude']]
y=d1[['longitude']]
```

In [23]:
```
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)
```

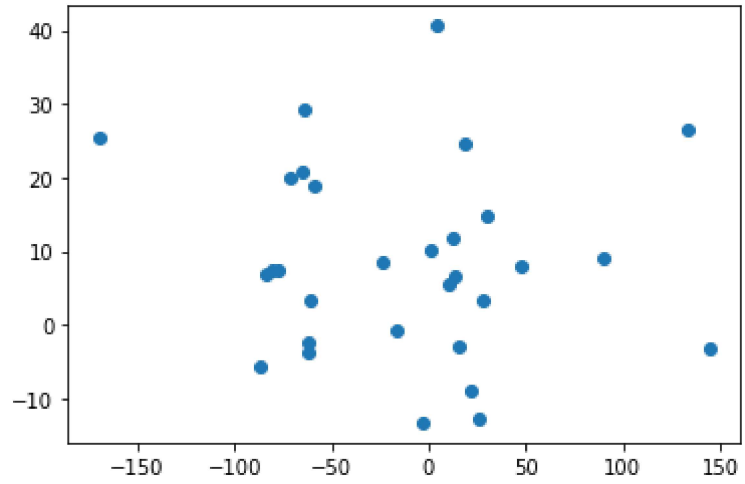
In [24]:
```
from sklearn.linear_model import LinearRegression
```

In [25]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[25]:
```
LinearRegression()
```

In [26]:
```
prediction =lr.predict(x_test)
py.scatter(y_test,prediction)
```

Out[26]:
```
<matplotlib.collections.PathCollection at 0x270e822dd90>
```



In [27]:
```
print(lr.score(x_test,y_test))
```

```
-0.1833756162667035
```

In [28]:
```
print(lr.score(x_train,y_train))
```

```
0.02999786784254388
```

In [29]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [30]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[30]:
```
Ridge(alpha=10)
```

In [31]:
```
rr.score(x_test,y_test)
```

Out[31]:
```
-0.1833451348091586
```

In [32]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[32]:
```
Lasso(alpha=10)
```

In [33]:
```
la.score(x_test,y_test)
```

Out[33]:
```
-0.1754070143332227
```

In [34]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[34]:
```
ElasticNet()
```

In [35]:
```
print(en.coef_)
```

```
[-0.25213369 -0.26651761]
```

In [36]:
```
print(en.intercept_)
```

```
[22.88851508]
```

In [37]:
```
print(en.predict(x_test))
```

```
[ 19.7901612    7.28499429  29.17664335   3.46827416   9.00380484
  14.74451397  -5.56621682  -8.93703528  25.44793242  18.77930971
 -12.82650503  -3.01654992   3.39954693  -3.63016065  24.45504942
  -2.27642083 -13.23938842   7.35394043   5.46252277  -3.14048698
  20.61170493  -0.87128379   6.8602535   26.554619     6.49134276
  11.70832927  10.04871441   8.06041276  40.51485169   8.53888576]
```

In [38]:
```
print(en.score(x_test,y_test))
```

```
-0.1828668015727275
```

In [39]:
```
from sklearn import metrics
```

In [40]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 56.47679051895102
```

In [41]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 5243.481470106939
```

In [42]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
Root Mean Squared Error: 72.41188762977346
```

In [43]:
```
import pickle
```

In [44]:
```
filename="countries"
pickle.dump(lr,open(filename,'wb'))
```

In [ ]: