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\20_states - 20_states.csv")
d
```

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

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

Out[4]:
```
Index(['id', 'name', 'country_id', 'country_code', 'country_name',
       'state_code', 'type', 'latitude', 'longitude'],
      dtype='object')
```

In [7]:
```
d1=d.head(100)
d1
```

Out[7]:

	id	name	country_id	country_code	country_name	state_code	type	latitude	longitude
0	3901	Badakhshan	1	AF	Afghanistan	BDS	NaN	36.734772	70.811995
1	3871	Badghis	1	AF	Afghanistan	BDG	NaN	35.167134	63.769538
2	3875	Baghlan	1	AF	Afghanistan	BGL	NaN	36.178903	68.745306
3	3884	Balkh	1	AF	Afghanistan	BAL	NaN	36.755060	66.897537
4	3872	Bamyan	1	AF	Afghanistan	BAM	NaN	34.810007	67.821210
...	...	...	...	...	...	...	...	...	...
95	1105	Chlef	4	DZ	Algeria	2	NaN	36.169351	1.289104
96	1121	Constantine	4	DZ	Algeria	25	NaN	36.337391	6.663812
97	4912	Djanet	4	DZ	Algeria	56	NaN	23.831087	8.700467
98	1098	Djelfa	4	DZ	Algeria	17	NaN	34.670396	3.250376
99	1129	El Bayadh	4	DZ	Algeria	32	NaN	32.714882	0.905662

100 rows × 9 columns

In [8]:
```
d1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   id            100 non-null    int64  
 1   name          100 non-null    object 
 2   country_id    100 non-null    int64  
 3   country_code  100 non-null    object 
 4   country_name  100 non-null    object 
 5   state_code    100 non-null    object 
 6   type          0 non-null      object 
 7   latitude      100 non-null    float64
 8   longitude     100 non-null    float64
dtypes: float64(2), int64(2), object(5)
memory usage: 7.2+ KB
```

In [25]:
```
x=d1[[ 'country_id','longitude']]
y=d1['latitude']
```

In [26]:
```
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)
```

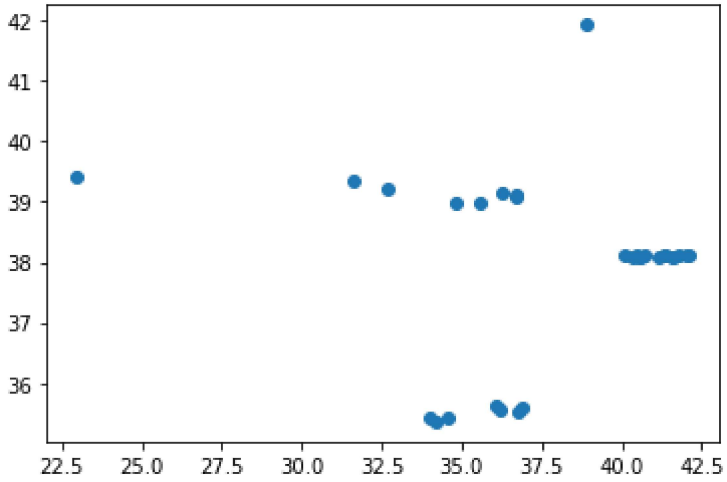
In [27]:
```
from sklearn.linear_model import LinearRegression
```

In [28]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[28]:
```
LinearRegression()
```

In [29]:
```
prediction =lr.predict(x_test)
py.scatter(y_test,prediction)
```

Out[29]:
```
<matplotlib.collections.PathCollection at 0x20bdbf0db20>
```



In [30]:
```
print(lr.score(x_test,y_test))
```

```
-0.1225486922846808
```

In [31]:
```
print(lr.score(x_train,y_train))
```

```
0.16028188571278168
```

In [32]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [33]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[33]:
```
Ridge(alpha=10)
```

In [34]:
```
rr.score(x_test,y_test)
```

Out[34]: -0.12089083337393514

In [35]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[35]: Lasso(alpha=10)

In [36]:
```
la.score(x_test,y_test)
```

Out[36]: -0.07069481980708736

In [37]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[37]: ElasticNet()

In [38]:
```
print(en.coef_)
```

```
[ 0.         -0.05058641]
```

In [39]:
```
print(en.intercept_)
```

```
39.050648395429235
```

In [40]:
```
print(en.predict(x_test))
```

```
[42.95448871 38.76104846 39.25032677 38.86743554 35.71531654 35.76728927
 38.02286022 38.04417632 39.16276023 38.06314575 35.4781513  38.02700386
 38.06404873 38.06103866 38.75225365 39.00483419 38.00771388 38.06465071
 35.54969027 38.02700386 35.66654228 38.00454841 35.55045384 38.95254205
 38.02459307 38.89597439 38.04809164 38.05682399 35.72884857 38.0336328 ]
```

In [41]:
```
print(en.score(x_test,y_test))
```

```
-0.11414001731345125
```

In [42]:
```
from sklearn import metrics
```

In [43]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 3.274963238298584
```

In [44]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 19.034453816258743
```

In [45]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
Root Mean Squared Error: 4.3628492772795555
```

In [46]:
```
import pickle
```

In [47]:
```
filename="states"
pickle.dump(lr,open(filename,'wb'))
```

In [ ]:
```

```