Problem Statement

Linear Regression

Import Libraries

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

In [2]:
```
a=pd.read_csv("states.csv")
a
```

Out[2]:

	id	name	country_id	country_code	country_name	state_code	type	latitude	lo
0	3901	Badakhshan	1	AF	Afghanistan	BDS	NaN	36.734772	70
1	3871	Badghis	1	AF	Afghanistan	BDG	NaN	35.167134	63
2	3875	Baghlan	1	AF	Afghanistan	BGL	NaN	36.178903	68
3	3884	Balkh	1	AF	Afghanistan	BAL	NaN	36.755060	66
4	3872	Bamyan	1	AF	Afghanistan	BAM	NaN	34.810007	67
...	...	...	...	...	...	...	...	...	
5072	1953	Mashonaland West Province	247	ZW	Zimbabwe	MW	NaN	-17.485103	29
5073	1960	Masvingo Province	247	ZW	Zimbabwe	MV	NaN	-20.624151	31
5074	1954	Matabeleland North Province	247	ZW	Zimbabwe	MN	NaN	-18.533157	27
5075	1952	Matabeleland South Province	247	ZW	Zimbabwe	MS	NaN	-21.052337	29
5076	1957	Midlands Province	247	ZW	Zimbabwe	MI	NaN	-19.055201	29

5077 rows × 9 columns

To display top 10 rows

In [3]:
```
c=a.head(15)
c
```

Out[3]:

	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
5	3892	Daykundi	1	AF	Afghanistan	DAY	NaN	33.669495	66.046
6	3899	Farah	1	AF	Afghanistan	FRA	NaN	32.495328	62.262
7	3889	Faryab	1	AF	Afghanistan	FYB	NaN	36.079561	64.905
8	3870	Ghazni	1	AF	Afghanistan	GHA	NaN	33.545059	68.417
9	3888	Ghōr	1	AF	Afghanistan	GHO	NaN	34.099578	64.905
10	3873	Helmand	1	AF	Afghanistan	HEL	NaN	39.298936	-76.616
11	3887	Herat	1	AF	Afghanistan	HER	NaN	34.352865	62.204
12	3886	Jowzjan	1	AF	Afghanistan	JOW	NaN	36.896969	65.665
13	3902	Kabul	1	AF	Afghanistan	KAB	NaN	34.555349	69.207
14	3890	Kandahar	1	AF	Afghanistan	KAN	NaN	31.628871	65.737

To find Missing values

In [4]:
```
c.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15 entries, 0 to 14
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   id            15 non-null     int64  
 1   name          15 non-null     object 
 2   country_id    15 non-null     int64  
 3   country_code  15 non-null     object 
 4   country_name  15 non-null     object 
 5   state_code    15 non-null     object 
 6   type          0 non-null      object 
 7   latitude      15 non-null     float64
 8   longitude     15 non-null     float64
dtypes: float64(2), int64(2), object(5)
memory usage: 1.2+ KB
```

To display summary of statistics

In [5]:
```
a.describe()
```

Out[5]:

	id	country_id	latitude	longitude
count	5077.000000	5077.000000	5008.000000	5008.000000
mean	2609.765413	133.467599	27.576415	17.178713

	id	country_id	latitude	longitude
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

To display column heading

In [6]:
```
a.columns
```

Out[6]:
```
Index(['id', 'name', 'country_id', 'country_code', 'country_name',
       'state_code', 'type', 'latitude', 'longitude'],
      dtype='object')
```

Pairplot

In [7]:
```
s=a.dropna(axis=1)
s
```

Out[7]:

	id	name	country_id	country_name
0	3901	Badakhshan	1	Afghanistan
1	3871	Badghis	1	Afghanistan
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 [8]:
```
s.columns
```

Out[8]:
```
Index(['id', 'name', 'country_id', 'country_name'], dtype='object')
```

To train the Model

In [10]:
```
g=c[['id']]
h=c['country_id']
```

To split dataset into training end test

In [11]:
```
from sklearn.model_selection import train_test_split
g_train,g_test,h_train,h_test=train_test_split(g,h,test_size=0.6)
```

To run the model

In [12]:
```
from sklearn.linear_model import LinearRegression
```

In [13]:
```
lr=LinearRegression()
lr.fit(g_train,h_train)
```

Out[13]:
```
LinearRegression()
```

In [14]:
```
print(lr.intercept_)
```

```
1.0
```

Coeffecient

In [15]:
```
coeff=pd.DataFrame(lr.coef_,g.columns,columns=['Co-effecient'])
coeff
```

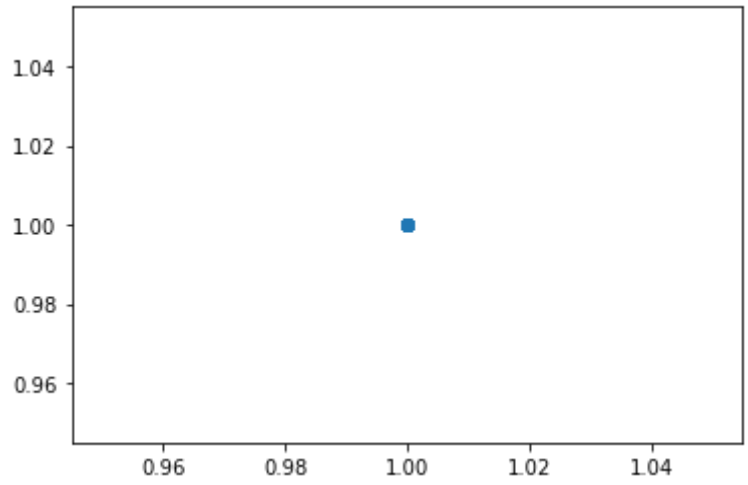
Out[15]:

	Co-effecient
id	-0.0

Best Fit line

In [16]:
```
prediction=lr.predict(g_test)
plt.scatter(h_test,prediction)
```

Out[16]:
```
<matplotlib.collections.PathCollection at 0x268751da580>
```



To find score

In [17]:
```
print(lr.score(g_test,h_test))
```

```
1.0
```

Import Lasso and ridge

In [18]:
```
from sklearn.linear_model import Ridge,Lasso
```

Ridge

In [19]:
```
ri=Ridge(alpha=5)
ri.fit(g_train,h_train)
```

Out[19]:
```
Ridge(alpha=5)
```

In [20]:
```
ri.score(g_test,h_test)
```

Out[20]:
```
1.0
```

In [21]:
```
ri.score(g_train,h_train)
```

Out[21]:
```
1.0
```

Lasso

In [22]:
```
l=Lasso(alpha=6)
l.fit(g_train,h_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.
py:530: ConvergenceWarning: Objective did not converge. You might want to increase t
he number of iterations. Duality gap: 0.0, tolerance: 0.0
  model = cd_fast.enet_coordinate_descent(
```

Out[22]:
```
Lasso(alpha=6)
```

In [23]:
```
l.score(g_test,h_test)
```

Out[23]:
```
1.0
```

In [24]:
```
ri.score(g_train,h_train)
```

Out[24]:
```
1.0
```

ElasticNet

In [25]:
```
from sklearn.linear_model import ElasticNet
e=ElasticNet()
e.fit(g_train,h_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.
py:530: ConvergenceWarning: Objective did not converge. You might want to increase t
he number of iterations. Duality gap: 0.0, tolerance: 0.0
  model = cd_fast.enet_coordinate_descent(
```

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

Coeffecient,intercept

In [26]:
```
print(e.coef_)
```

```
[0.]
```

In [27]:
```
print(e.intercept_)
```

```
1.0
```

Prediction

In [28]:
```
d=e.predict(g_test)
d
```

Out[28]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

In [29]:
```
print(e.score(g_test,h_test))
```

```
1.0
```

Evaluation

In [30]:
```
from sklearn import metrics
print("Mean Absolute error:",metrics.mean_absolute_error(h_test,d))
```

```
Mean Absolute error: 0.0
```

In [31]:
```
print("Mean Squared error:",metrics.mean_squared_error(h_test,d))
```

```
Mean Squared error: 0.0
```

In [32]:
```
print("Mean Squared error:",np.sqrt(metrics.mean_squared_error(h_test,d)))
```

```
Mean Squared error: 0.0
```

Model Saving

In [33]:
```
import pickle
filename="pre"
pickle.dump(lr,open(filename,"wb"))
```

In [34]:
```
filename='pre'
model = pickle.load(open(filename,'rb'))
```

In [36]:
```
eral=[[15],[54]]
result=model.predict(eral)
result
```

Out[36]:
```
array([1., 1.])
```

In [ ]:

In [ ]: