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("cities.csv")
a
```

Out[2]:

	id	name	state_id	state_code	state_name	country_id	country_code	country_nar
0	52	Ashkāsham	3901	BDS	Badakhshan	1	AF	Afghanist
1	68	Fayzabad	3901	BDS	Badakhshan	1	AF	Afghanist
2	78	Jurm	3901	BDS	Badakhshan	1	AF	Afghanist
3	84	Khandūd	3901	BDS	Badakhshan	1	AF	Afghanist
4	115	Rāghistān	3901	BDS	Badakhshan	1	AF	Afghanist
...	...	...	...	...	...	...	...	
150449	131496	Redcliff	1957	MI	Midlands Province	247	ZW	Zimbab
150450	131502	Shangani	1957	MI	Midlands Province	247	ZW	Zimbab
150451	131503	Shurugwi	1957	MI	Midlands Province	247	ZW	Zimbab
150452	131504	Shurugwi District	1957	MI	Midlands Province	247	ZW	Zimbab
150453	131508	Zvishavane District	1957	MI	Midlands Province	247	ZW	Zimbab

150454 rows × 11 columns

To display top 10 rows

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

Out[3]:

	id	name	state_id	state_code	state_name	country_id	country_code	country_name	lat
0	52	Ashkāsham	3901	BDS	Badakhshan	1	AF	Afghanistan	36.(

	id	name	state_id	state_code	state_name	country_id	country_code	country_name	lat
1	68	Fayzabad	3901	BDS	Badakhshan	1	AF	Afghanistan	37.
2	78	Jurm	3901	BDS	Badakhshan	1	AF	Afghanistan	36.
3	84	Khandūd	3901	BDS	Badakhshan	1	AF	Afghanistan	36.
4	115	Rāghistān	3901	BDS	Badakhshan	1	AF	Afghanistan	37.
5	131	Wākhān	3901	BDS	Badakhshan	1	AF	Afghanistan	37.
6	72	Ghormach	3871	BDG	Badghis	1	AF	Afghanistan	35.
7	108	Qala i Naw	3871	BDG	Badghis	1	AF	Afghanistan	34.
8	54	Baghlān	3875	BGL	Baghlan	1	AF	Afghanistan	36.
9	140	Ḩukūmatī Dahanah-ye Ghōrī	3875	BGL	Baghlan	1	AF	Afghanistan	35.
10	101	Nahrīn	3875	BGL	Baghlan	1	AF	Afghanistan	36.
11	105	Pul-e Khumrī	3875	BGL	Baghlan	1	AF	Afghanistan	35.
12	55	Balkh	3884	BAL	Balkh	1	AF	Afghanistan	36.
13	65	Dowlatābād	3884	BAL	Balkh	1	AF	Afghanistan	36.
14	85	Khulm	3884	BAL	Balkh	1	AF	Afghanistan	36.

To find Missing values

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15 entries, 0 to 14
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   id            15 non-null     int64  
 1   name          15 non-null     object 
 2   state_id      15 non-null     int64  
 3   state_code    15 non-null     object 
 4   state_name    15 non-null     object 
 5   country_id    15 non-null     int64  
 6   country_code  15 non-null     object 
 7   country_name  15 non-null     object 
 8   latitude      15 non-null     float64
 9   longitude     15 non-null     float64
 10  wikiDataId    15 non-null     object 
dtypes: float64(2), int64(3), object(6)
memory usage: 1.4+ KB
```

To display summary of statistics

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

Out[5]:

	id	state_id	country_id	latitude	longitude
count	150454.000000	150454.000000	150454.000000	150454.000000	150454.000000
mean	76407.091689	2678.377677	140.658460	31.556175	2.369557
std	44357.755335	1363.513591	70.666123	22.813220	68.012770
min	1.000000	1.000000	1.000000	-75.000000	-179.121980
25%	38160.250000	1451.000000	82.000000	19.000000	-58.468150
50%	75975.500000	2174.000000	142.000000	40.684720	8.669980
75%	115204.750000	3905.000000	207.000000	47.239220	27.750000
max	153528.000000	5116.000000	247.000000	73.508190	179.466000

To display column heading

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

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

Pairplot

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

Out[7]:

	id	name	state_id	state_name	country_id	country_name	latitude	longitude
0	52	Ashkāsham	3901	Badakhshan	1	Afghanistan	36.68333	71.53333
1	68	Fayzabad	3901	Badakhshan	1	Afghanistan	37.11664	70.58002
2	78	Jurm	3901	Badakhshan	1	Afghanistan	36.86477	70.83421
3	84	Khandūd	3901	Badakhshan	1	Afghanistan	36.95127	72.31800
4	115	Rāghistān	3901	Badakhshan	1	Afghanistan	37.66079	70.67346
...	...	...	...	...	...	...	...	...
150449	131496	Redcliff	1957	Midlands Province	247	Zimbabwe	-19.03333	29.78333
150450	131502	Shangani	1957	Midlands Province	247	Zimbabwe	-19.78333	29.36667
150451	131503	Shurugwi	1957	Midlands Province	247	Zimbabwe	-19.67016	30.00589
150452	131504	Shurugwi District	1957	Midlands Province	247	Zimbabwe	-19.75000	30.16667
150453	131508	Zvishavane District	1957	Midlands Province	247	Zimbabwe	-20.30345	30.07514

150454 rows × 8 columns

In [8]:
```
s.columns
```

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

To train the Model

In [10]:
```
g=c[['id','state_id','country_id','latitude']]
h=c['longitude']
```

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_)
```

```
1208.1067273022823
```

Coeffecient

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

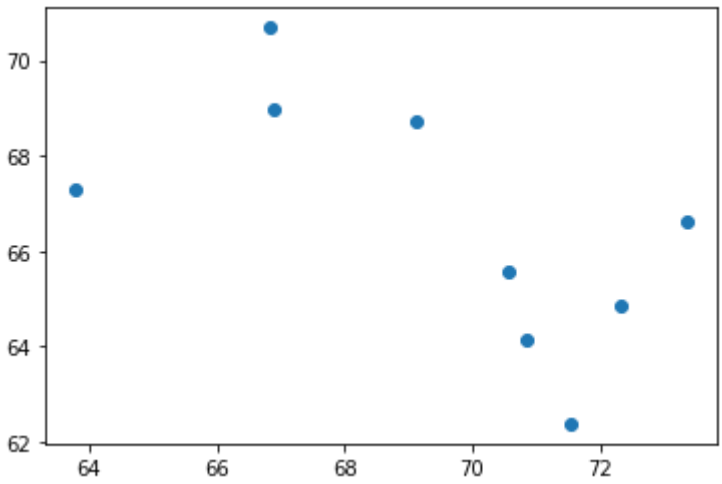
Out[15]:

	Co-effecient
id	0.022887
state_id	-0.355000
country_id	0.000000
latitude	6.486225

Best Fit line

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

Out[16]:
```
<matplotlib.collections.PathCollection at 0x205d3b7e700>
```



To find score

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

```
-2.742446383544979
```

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]:
```
0.5777712432152431
```

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

Out[21]:
```
0.4750148660068506
```

Lasso

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

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

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

Out[23]:
```
0.43928643672616297
```

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

Out[24]:
```
0.4750148660068506
```

ElasticNet

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

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

Coeffecient,intercept

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

```
[-0.00627202  0.14205907  0.          0.        ]
```

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

```
-482.6762226335557
```

Prediction

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

Out[28]:
```
array([70.67457048, 71.06970743, 66.78284731, 71.00698728, 70.96935519,
       68.67351931, 68.73623946, 67.16919515, 71.17005967])
```

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

```
0.6208089928338496
```

Evaluation

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

```
Mean Absolute error: 1.5228222557728168
```

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

```
Mean Squared error: 3.2276699983653265
```

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

```
Mean Squared error: 1.796571734823112
```

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=[[15,10,65,8],[19,54,30,90]]
result=model.predict(eral)
result
```

Out[36]:
```
array([1256.78983121, 1773.13184947])
```

In [ ]:
```

```