kaviyadevi 20106064

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

In [11]:
```
#to import dataset
data1=pd.read_csv(r"C:\Users\user\Downloads\20_states - 20_states.csv")
data1
```

Out[11]:

	id	name	country_id	country_code	country_name	state_code	type	latitude	l
0	3901	Badakhshan	1	AF	Afghanistan	BDS	NaN	36.734772	7
1	3871	Badghis	1	AF	Afghanistan	BDG	NaN	35.167134	6
2	3875	Baghlan	1	AF	Afghanistan	BGL	NaN	36.178903	6
3	3884	Balkh	1	AF	Afghanistan	BAL	NaN	36.755060	6
4	3872	Bamyan	1	AF	Afghanistan	BAM	NaN	34.810007	6
...	...	...	...	...	...	...	...	...	
5072	1953	Mashonaland West Province	247	ZW	Zimbabwe	MW	NaN	-17.485103	2
5073	1960	Masvingo Province	247	ZW	Zimbabwe	MV	NaN	-20.624151	3
5074	1954	Matabeleland North Province	247	ZW	Zimbabwe	MN	NaN	-18.533157	2
5075	1952	Matabeleland South Province	247	ZW	Zimbabwe	MS	NaN	-21.052337	2
5076	1957	Midlands Province	247	ZW	Zimbabwe	MI	NaN	-19.055201	2

5077 rows × 9 columns

In [14]:
```
#to display top 5 rows
data=data1.head(100)
data
```

Out[14]:

	id	name	country_id	country_code	country_name	state_code	type	latitude	long
0	3901	Badakhshan	1	AF	Afghanistan	BDS	NaN	36.734772	70.81
1	3871	Badghis	1	AF	Afghanistan	BDG	NaN	35.167134	63.76
2	3875	Baghlan	1	AF	Afghanistan	BGL	NaN	36.178903	68.74
3	3884	Balkh	1	AF	Afghanistan	BAL	NaN	36.755060	66.89
4	3872	Bamyan	1	AF	Afghanistan	BAM	NaN	34.810007	67.82
...	...	...	...	...	...	...	...	...	
95	1105	Chlef	4	DZ	Algeria	2	NaN	36.169351	1.28
96	1121	Constantine	4	DZ	Algeria	25	NaN	36.337391	6.66
97	4912	Djanet	4	DZ	Algeria	56	NaN	23.831087	8.70
98	1098	Djelfa	4	DZ	Algeria	17	NaN	34.670396	3.25
99	1129	El Bayadh	4	DZ	Algeria	32	NaN	32.714882	0.90

100 rows × 9 columns

DATA CLEANING AND PREPROCESSING

In [15]:
```
data.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 [16]:
```
#to display summary of statistics
data.describe()
```

Out[16]:

	id	country_id	latitude	longitude
count	100.000000	100.000000	100.000000	100.000000
mean	1909.260000	2.540000	37.558787	28.310982
std	1573.838213	1.149616	4.105491	31.635217
min	593.000000	1.000000	22.966335	-83.612201
25%	617.750000	1.000000	34.846178	19.516644
50%	1104.500000	3.000000	36.849510	20.070278
75%	3880.250000	3.000000	40.968260	65.095930
max	4912.000000	4.000000	42.790134	71.097317

In [17]:
```
data.isnull()
```

Out[17]:

	id	name	country_id	country_code	country_name	state_code	type	latitude	longitude
0	False	False	False	False	False	False	True	False	False
1	False	False	False	False	False	False	True	False	False
2	False	False	False	False	False	False	True	False	False
3	False	False	False	False	False	False	True	False	False
4	False	False	False	False	False	False	True	False	False
...	...	...	...	...	...	...	...	...	...
95	False	False	False	False	False	False	True	False	False
96	False	False	False	False	False	False	True	False	False
97	False	False	False	False	False	False	True	False	False
98	False	False	False	False	False	False	True	False	False
99	False	False	False	False	False	False	True	False	False

100 rows × 9 columns

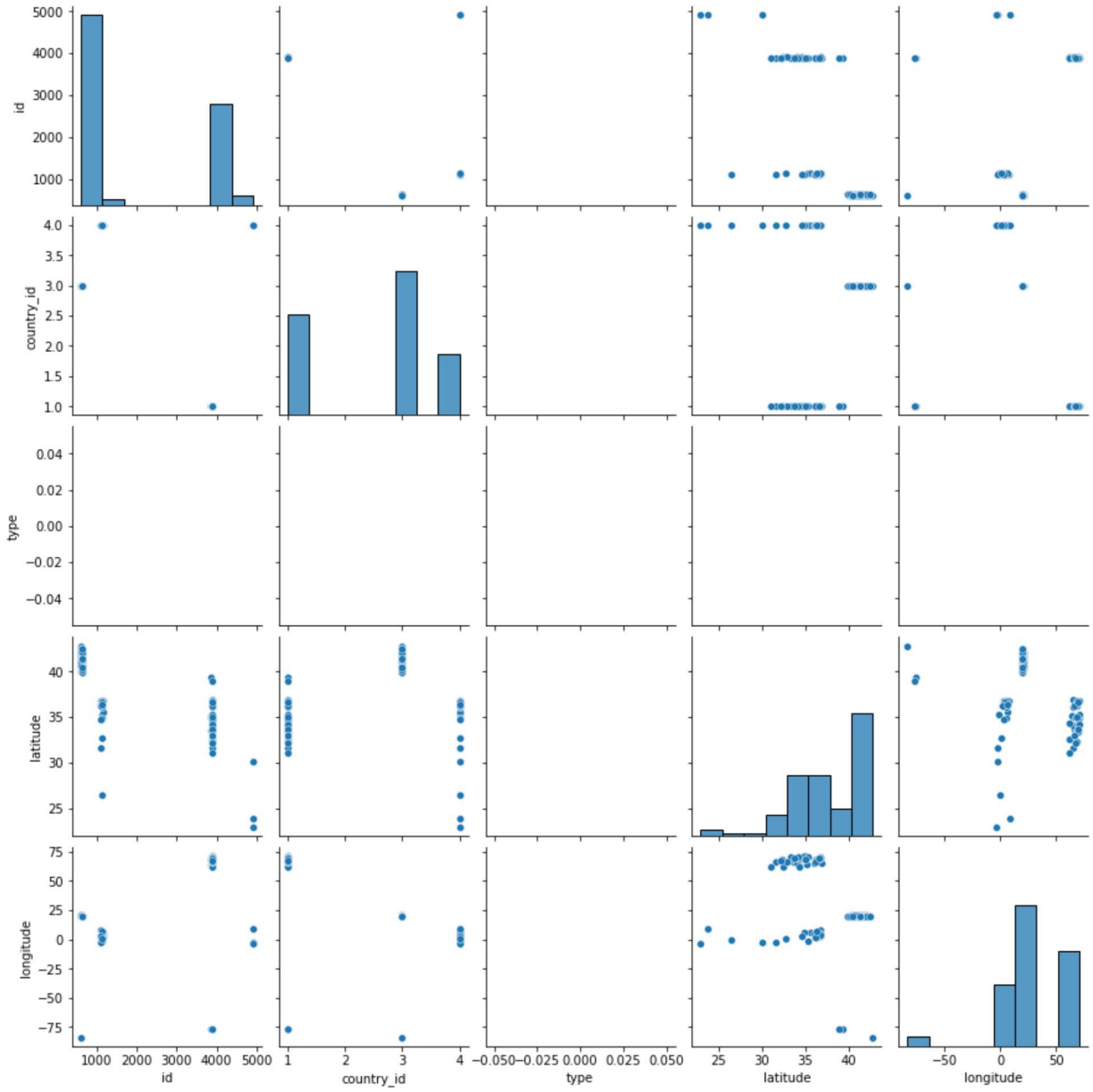
In [18]:
```
#to display the column heading
data.columns
```

Out[18]:
```
Index(['id', 'name', 'country_id', 'country_code', 'country_name',
       'state_code', 'type', 'latitude', 'longitude'],
      dtype='object')
```

EDA and DATA VISUALIZATION

In [19]:
```
sns.pairplot(data)
```

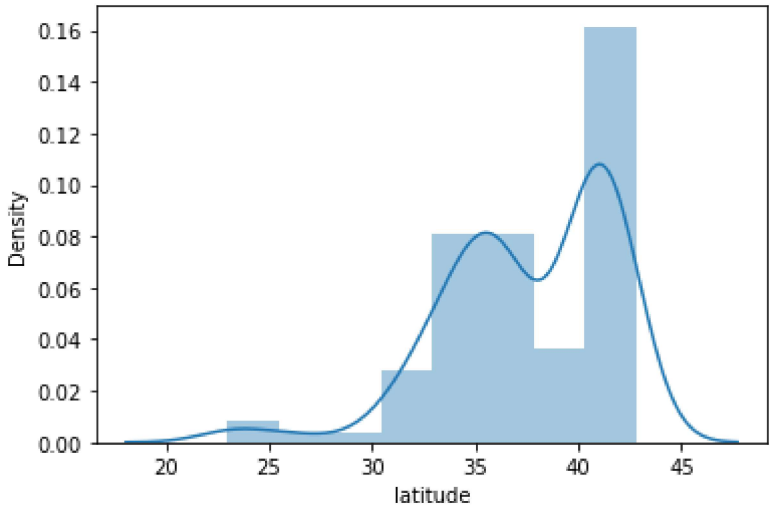
Out[19]: <seaborn.axisgrid.PairGrid at 0x1eae26d0f70>



In [22]:
```
sns.distplot(data['latitude'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

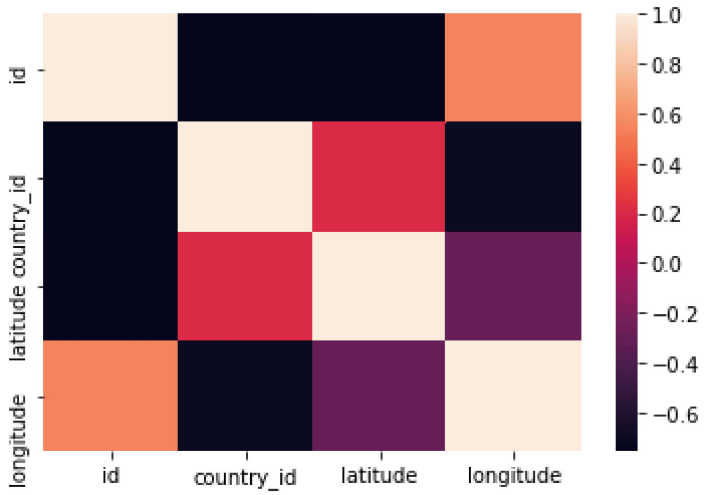
Out[22]: <AxesSubplot:xlabel='latitude', ylabel='Density'>



In [23]:
```
df=data[['id', 'name', 'country_id', 'country_code', 'country_name',
       'state_code', 'type', 'latitude', 'longitude']]
```

In [24]:
```
sns.heatmap(df.corr())
```

Out[24]: <AxesSubplot:>



TO TRAIN MODEL

In [35]:
```
x=df[['id','country_id',  'longitude']]
y=df['latitude']
```

In [36]:
```
#to split my dataset into trainning and test

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 [37]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[37]: LinearRegression()

In [38]:
```
#to find intercept
print(lr.intercept_)
```

```
51.906589553705004
```

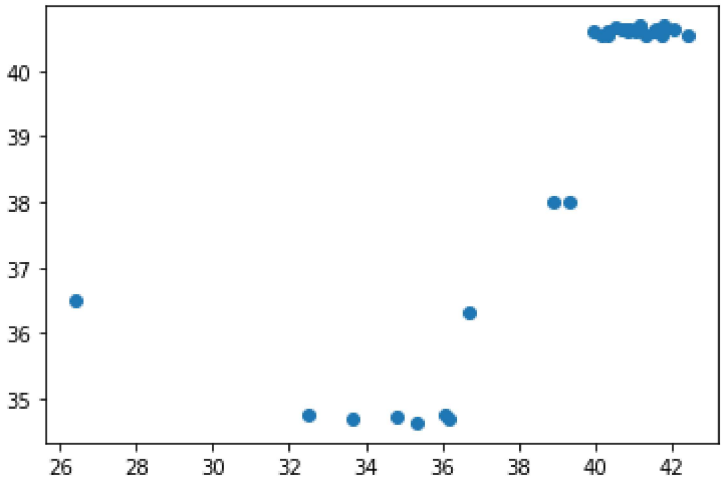
In [39]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[39]:

	Co-efficient
id	-0.003279
country_id	-2.941356
longitude	-0.022710

In [40]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[40]:
```
<matplotlib.collections.PathCollection at 0x1eae5bbc2b0>
```



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

```
0.6621197196566639
```

RIDGE AND LASSO REGRESSION

In [42]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [43]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[43]:
```
Ridge(alpha=10)
```

In [44]:
```
rr.score(x_test,y_test)
```

Out[44]:
```
0.4457213596569043
```

In [45]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[45]:
```
Lasso(alpha=10)
```

In [46]:
```
la.score(x_test,y_test)
```

Out[46]:
```
-0.24688676135583032
```

In [47]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[47]:
```
ElasticNet()
```

In [48]:
```
print(en.coef_)
```

```
[-0.00297107 -0.34669284  0.05919049]
```

In [49]:
```
print(en.predict(x_test))
```

```
[40.47334187 40.43313172 40.47868135 40.43673332 40.37081685 34.05913446
 40.36714102 34.32798255 40.36089518 40.50057408 40.44003155 40.51715503
 37.37447499 34.28219806 40.38225909 40.47349991 34.44108622 40.43662156
 40.5102694  37.51381059 25.72991461 40.42444824 40.40835966 34.11772199
 33.87296599 40.47153946 40.43177221 40.5011446  40.44141778 25.67325528]
```

In [50]:
```
print(en.score(x_test,y_test))
```

```
-0.33770478686566907
```

EVALUATION METRICS

In [51]:
```
from sklearn import metrics
```

In [52]:
```
print("Mean Absolute error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute error 1.0941135596142322
```

In [53]:
```
print("Mean Squared error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared error 4.27265356067959
```

In [54]:
```
print("Root Mean Absolute error",np.sqrt(metrics.mean_squared_error(y_test,predic
```

```
Root Mean Absolute error 2.0670398062639213
```

MODEL SAVING

In [55]:
```
import pickle
```

In [57]:
```
filename='predict2'
pickle.dump(lr,open(filename,'wb'))
```