kaviyadevi 20106064

In [2]:
```
#to import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
#to import dataset
data=pd.read_csv(r"C:\Users\user\Downloads\22_countries - 22_countries.csv")
data
```

Out[3]:

numeric_code	phone_code	capital	currency	currency_name	currency_symbol	tld	native
4	93	Kabul	AFN	Afghan afghani	؋	.af	افغانستان
248	+358-18	Mariehamn	EUR	Euro	€	.ax	Åland
8	355	Tirana	ALL	Albanian lek	Lek	.al	Shqipëria
12	213	Algiers	DZD	Algerian dinar	دج	.dz	الجزائر
16	+1-684	Pago Pago	USD	US Dollar	$	.as	American Samoa
...	...	...	...	...	...	...	...
876	681	Mata Utu	XPF	CFP franc	₣	.wf	Wallis et Futuna
732	212	El-Aaiun	MAD	Moroccan Dirham	MAD	.eh	الصحراء الغربية
887	967	Sanaa	YER	Yemeni rial	ريال	.ye	اليَمَن
894	260	Lusaka	ZMW	Zambian kwacha	ZK	.zm	Zambia
716	263	Harare	ZWL	Zimbabwe Dollar	$	.zw	Zimbabwe

In [4]:
```
#to display top 5 rows
data.head()
```

Out[4]:

	id	name	iso3	iso2	numeric_code	phone_code	capital	currency	currency_name	c
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	

DATA CLEANING AND PREPROCESSING

In [5]:
```
#
data.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 [6]:

```
#to display summary of statistics(here to know min max value)
data.describe()
```

Out[6]:

	id	numeric_code	latitude	longitude
count	250.000000	250.00000	250.000000	250.00000
mean	125.500000	435.80400	16.402597	13.52387
std	72.312977	254.38354	26.757204	73.45152
min	1.000000	4.00000	-74.650000	-176.20000
25%	63.250000	219.00000	1.000000	-49.75000
50%	125.500000	436.00000	16.083333	17.00000
75%	187.750000	653.50000	39.000000	48.75000
max	250.000000	926.00000	78.000000	178.00000

In [7]:

```
#to display the column heading
data.columns
```

Out[7]:

```
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')
```

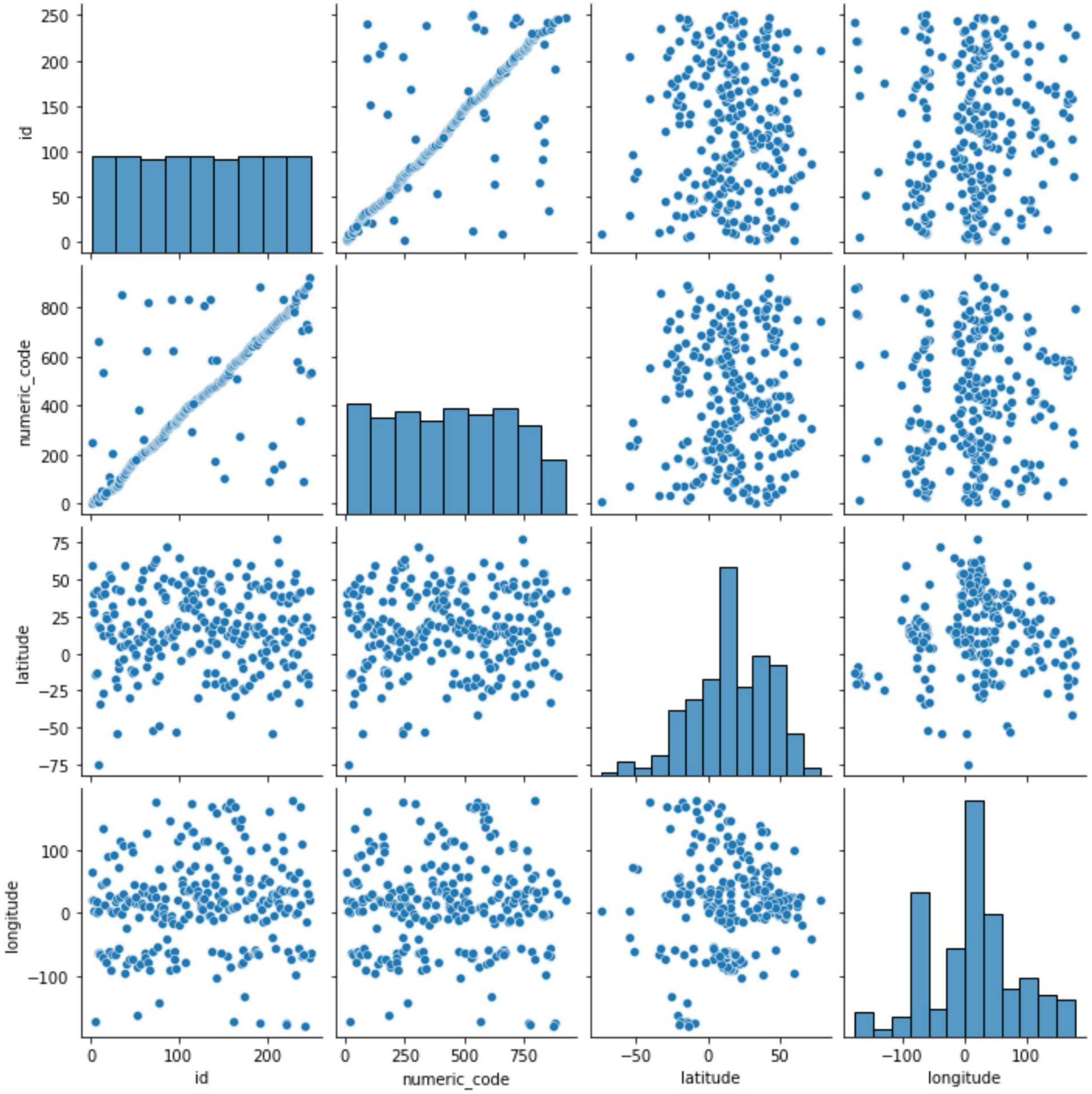
In [8]:

```
#here there is no missing values (identified through info() 5000 data are describ
```

EDA and DATA VISUALIZATION

```
In [9]: sns.pairplot(data)
```

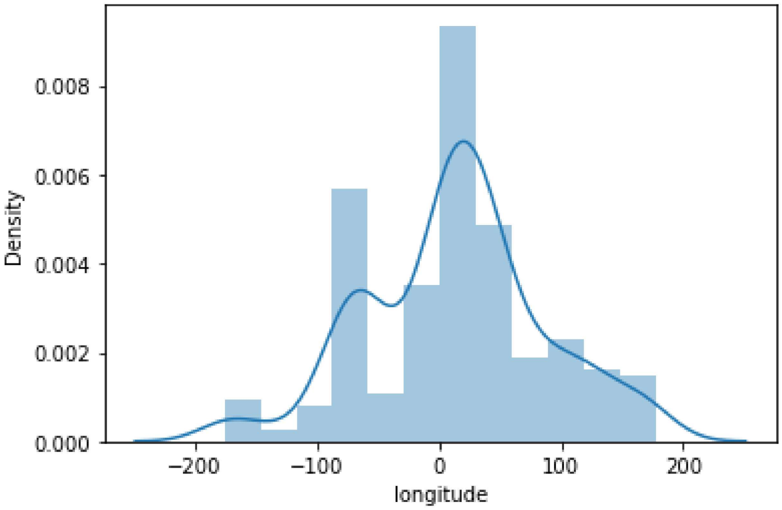
```
Out[9]: <seaborn.axisgrid.PairGrid at 0x1a8cd4fcd90>
```



In [11]:
```
sns.distplot(data['longitude'])
```

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

Out[11]:
```
<AxesSubplot:xlabel='longitude', ylabel='Density'>
```



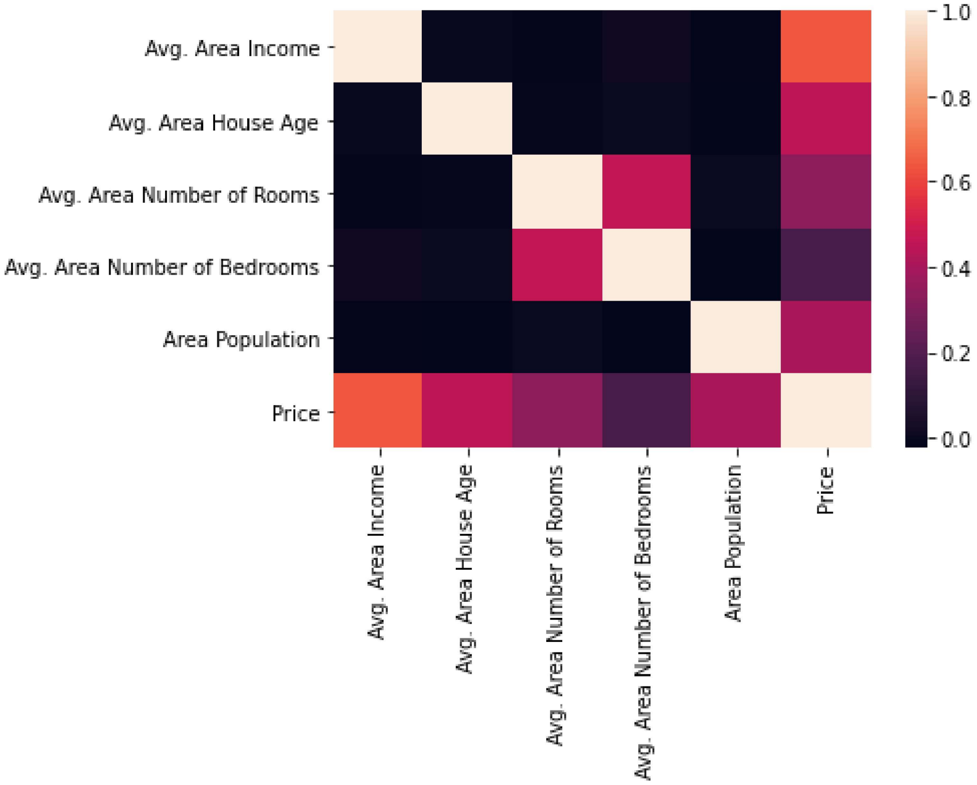
In [10]:
```
df=data[['id', 'name', 'iso3', 'iso2', 'numeric_code', 'phone_code', 'capital',
       'currency', 'currency_name', 'currency_symbol', 'tld', 'native',
       'region', 'subregion', 'timezones', 'latitude', 'longitude', 'emoji',
       'emojiU']]
```

In [13]:

```
sns.heatmap(df.corr())
```

Out[13]:

```
<AxesSubplot:>
```



TO TRAIN MODEL

In [15]:

```
x=df[[ 'numeric_code', 'latitude']]
y=df[ 'longitude']
```

In [16]:
```
#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 [17]:
```
from sklearn.linear_model import LinearRegression

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

Out[17]:
```
LinearRegression()
```

In [18]:
```
#to find intercept
print(lr.intercept_)
```

```
6.9106175018100195
```

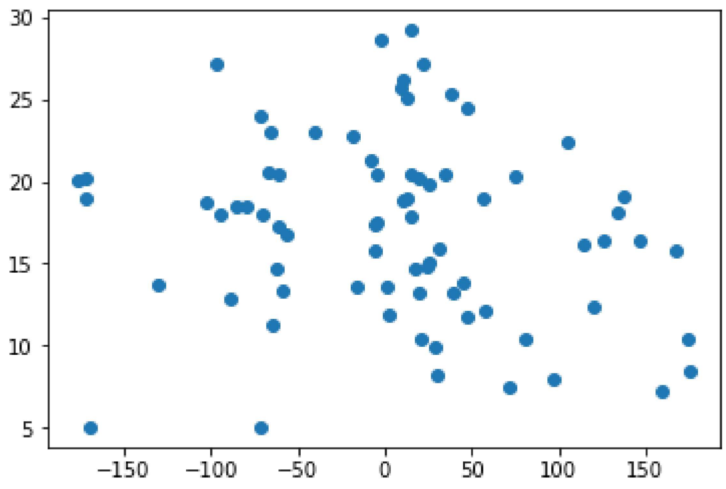
In [19]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[19]:

	Co-efficient
numeric_code	0.017294
latitude	0.149752

In [20]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[20]:
```
<matplotlib.collections.PathCollection at 0x1a8cf151580>
```



In [21]:
```
print(lr.score(x_test,y_test))
```

```
-0.04765080309190939
```

RIDGE AND LASSO REGRESSION

In [22]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [23]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[23]:
```
Ridge(alpha=10)
```

In [24]:
```
rr.score(x_test,y_test)
```

Out[24]:
```
-0.04765044867109092
```

In [25]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[25]:
```
Lasso(alpha=10)
```

In [26]:
```
la.score(x_test,y_test)
```

Out[26]:
```
-0.04705597236351888
```

In [27]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[27]:
```
ElasticNet()
```

In [28]:
```
print(en.coef_)
```

```
[0.01730174 0.14889432]
```

In [29]:
```
print(en.predict(x_test))
```

```
[ 5.06377847 16.18580013 29.16345978 12.40525781 23.02635811  8.26856046
 20.40579836 28.64919652 11.72899975  7.51568636 20.53852088 13.16418351
 12.20586526 25.0990842  13.75926407  8.42802618 11.25719093 20.0971335
 20.16296455 10.40160015 20.15875723 15.86157697 21.24874919 23.93174006
 20.45738889 13.34054527 25.2817271  14.7387785  17.5149174  16.85248408
 13.16852129 15.9171056  16.43675945 27.11235345 20.44200475 20.24059573
 20.38830071  9.89926193 18.1593286  24.50116188 26.15295678 22.35190489
 18.93799797  5.08413763 18.90637478  7.93201259 19.0635806  13.59514589
 13.77748397 11.86515378 27.11254936 14.68357443  7.28710468 22.90122284
 13.59768284 17.84084837 10.38560653 15.05373235 15.83859771 25.61728154
 16.37417793 18.71971466 18.00017757 17.30591131 18.97315956 14.88740363
 12.82179384 18.51110033 18.85794657 17.35325904 19.73717306 22.68944619
 18.00410964 18.48648052 10.4548135 ]
```

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

```
-0.047616873319522135
```

Evaluation metrics

In [31]:
```
from sklearn import metrics
```

In [32]:
```
print("Mean Absolute error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute error 64.69322764303017
```

In [33]:
```
print("Mean Squared error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared error 7140.384138915294
```

In [34]:
```
print("Root Mean Absolute error",np.sqrt(metrics.mean_squared_error(y_test,predic
```

```
Root Mean Absolute error 84.50079371766454
```

MODEL SAVING

In [35]:
```
import pickle
```

In [36]:
```
filename='predict3'
pickle.dump(lr,open(filename,'wb'))
```