D14

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

In [2]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\18_world-data-2023.csv")
df
```

Out[2]:

	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Calling Code
0	Afghanistan	60	AF	58.10%	652,230	323,000	32.49	93.0
1	Albania	105	AL	43.10%	28,748	9,000	11.78	355.0
2	Algeria	18	DZ	17.40%	2,381,741	317,000	24.28	213.0
3	Andorra	164	AD	40.00%	468	NaN	7.20	376.0
4	Angola	26	AO	47.50%	1,246,700	117,000	40.73	244.0
...	...	...	...	...	...	...	...	...
190	Venezuela	32	VE	24.50%	912,050	343,000	17.88	58.0
191	Vietnam	314	VN	39.30%	331,210	522,000	16.75	84.0
192	Yemen	56	YE	44.60%	527,968	40,000	30.45	967.0
193	Zambia	25	ZM	32.10%	752,618	16,000	36.19	260.0
194	Zimbabwe	38	ZW	41.90%	390,757	51,000	30.68	263.0

195 rows × 35 columns

In [3]:
```
df.head(10)
```

Out[3]:

	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Calling Code	C
0	Afghanistan	60	AF	58.10%	652,230	323,000	32.49	93.0	
1	Albania	105	AL	43.10%	28,748	9,000	11.78	355.0	
2	Algeria	18	DZ	17.40%	2,381,741	317,000	24.28	213.0	
3	Andorra	164	AD	40.00%	468	NaN	7.20	376.0	
4	Angola	26	AO	47.50%	1,246,700	117,000	40.73	244.0	
5	Antigua and Barbuda	223	AG	20.50%	443	0	15.33	1.0	
6	Argentina	17	AR	54.30%	2,780,400	105,000	17.02	54.0	
7	Armenia	104	AM	58.90%	29,743	49,000	13.99	374.0	
8	Australia	3	AU	48.20%	7,741,220	58,000	12.60	61.0	
9	Austria	109	AT	32.40%	83,871	21,000	9.70	43.0	

10 rows × 35 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 35 columns):
 #   Column                                   Non-Null Count  Dtype  
---  ------                                   --------------  -----  
 0   Country                                  195 non-null    object 
 1   Density
(P/Km2)                          195 non-null    object 
 2   Abbreviation                             188 non-null    object 
 3   Agricultural Land( %)                    188 non-null    object 
 4   Land Area(Km2)                           194 non-null    object 
 5   Armed Forces size                        171 non-null    object 
 6   Birth Rate                               189 non-null    float64
 7   Calling Code                             194 non-null    float64
 8   Capital/Major City                       192 non-null    object 
 9   Co2-Emissions                            188 non-null    object 
 10  CPI                                      178 non-null    object 
 11  CPI Change (%)                           179 non-null    object 
 12  Currency-Code                            180 non-null    object 
 13  Fertility Rate                           188 non-null    float64
 14  Forested Area (%)                        188 non-null    object 
 15  Gasoline Price                           175 non-null    object 
 16  GDP                                      193 non-null    object 
 17  Gross primary education enrollment (%)   188 non-null    object 
 18  Gross tertiary education enrollment (%)  183 non-null    object 
 19  Infant mortality                         189 non-null    float64
 20  Largest city                             189 non-null    object 
 21  Life expectancy                          187 non-null    float64
 22  Maternal mortality ratio                 181 non-null    float64
 23  Minimum wage                             150 non-null    object 
 24  Official language                        194 non-null    object 
 25  Out of pocket health expenditure         188 non-null    object 
 26  Physicians per thousand                  188 non-null    float64
 27  Population                               194 non-null    object 
 28  Population: Labor force participation (%) 176 non-null    object 
 29  Tax revenue (%)                          169 non-null    object 
 30  Total tax rate                           183 non-null    object 
 31  Unemployment rate                        176 non-null    object 
 32  Urban_population                         190 non-null    object 
 33  Latitude                                 194 non-null    float64
 34  Longitude                                194 non-null    float64
dtypes: float64(9), object(26)
memory usage: 53.4+ KB
```

In [5]:
```
dff=df.dropna()
```

In [6]:
```
dff.describe()
```

Out[6]:

	Birth Rate	Calling Code	Fertility Rate	Infant mortality	Life expectancy	Maternal mortality ratio	Physicians per thousand	
count	110.000000	110.000000	110.000000	110.000000	110.000000	110.000000	110.000000	11
mean	20.196455	344.290909	2.672182	20.271818	72.671818	137.227273	1.919182	2
std	10.039056	341.231562	1.308142	18.453214	7.000788	201.171462	1.598116	2
min	6.400000	1.000000	0.980000	1.700000	54.300000	2.000000	0.010000	-4
25%	11.075000	70.000000	1.682500	6.100000	67.625000	15.250000	0.467500	
50%	17.830000	239.500000	2.200000	13.600000	74.400000	41.000000	1.640000	2
75%	27.962500	420.750000	3.505000	31.500000	77.350000	176.000000	3.007500	4
max	46.080000	1876.000000	6.910000	78.500000	83.300000	1120.000000	7.120000	6

In [7]:
```
dff.isnull().sum()
```

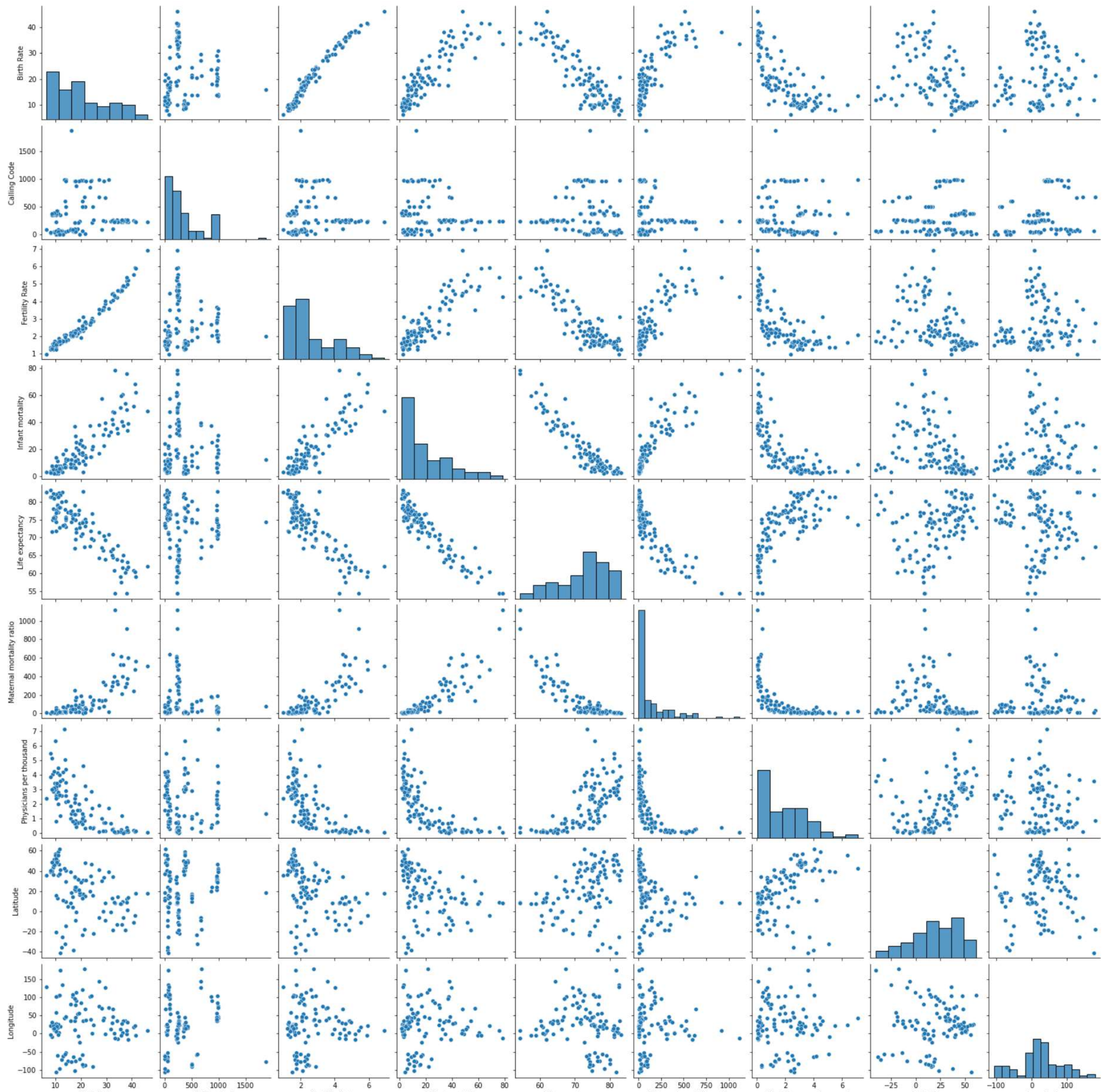
Out[7]:
```
Country                                    0
Density\n(P/Km2)                           0
Abbreviation                               0
Agricultural Land( %)                      0
Land Area(Km2)                             0
Armed Forces size                          0
Birth Rate                                 0
Calling Code                               0
Capital/Major City                         0
Co2-Emissions                              0
CPI                                        0
CPI Change (%)                             0
Currency-Code                              0
Fertility Rate                             0
Forested Area (%)                          0
Gasoline Price                             0
GDP                                        0
Gross primary education enrollment (%)     0
Gross tertiary education enrollment (%)    0
Infant mortality                           0
Largest city                               0
Life expectancy                            0
Maternal mortality ratio                   0
Minimum wage                               0
Official language                          0
Out of pocket health expenditure           0
Physicians per thousand                    0
Population                                 0
Population: Labor force participation (%)  0
Tax revenue (%)                            0
Total tax rate                             0
Unemployment rate                          0
Urban_population                           0
Latitude                                   0
Longitude                                  0
dtype: int64
```

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

Out[8]:
```
Index(['Country', 'Density\n(P/Km2)', 'Abbreviation', 'Agricultural Land(
%)',
       'Land Area(Km2)', 'Armed Forces size', 'Birth Rate', 'Calling Code',
       'Capital/Major City', 'Co2-Emissions', 'CPI', 'CPI Change (%)',
       'Currency-Code', 'Fertility Rate', 'Forested Area (%)',
       'Gasoline Price', 'GDP', 'Gross primary education enrollment (%)',
       'Gross tertiary education enrollment (%)', 'Infant mortality',
       'Largest city', 'Life expectancy', 'Maternal mortality ratio',
       'Minimum wage', 'Official language', 'Out of pocket health expenditur
e',
       'Physicians per thousand', 'Population',
       'Population: Labor force participation (%)', 'Tax revenue (%)',
       'Total tax rate', 'Unemployment rate', 'Urban_population', 'Latitude',
       'Longitude'],
      dtype='object')
```

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

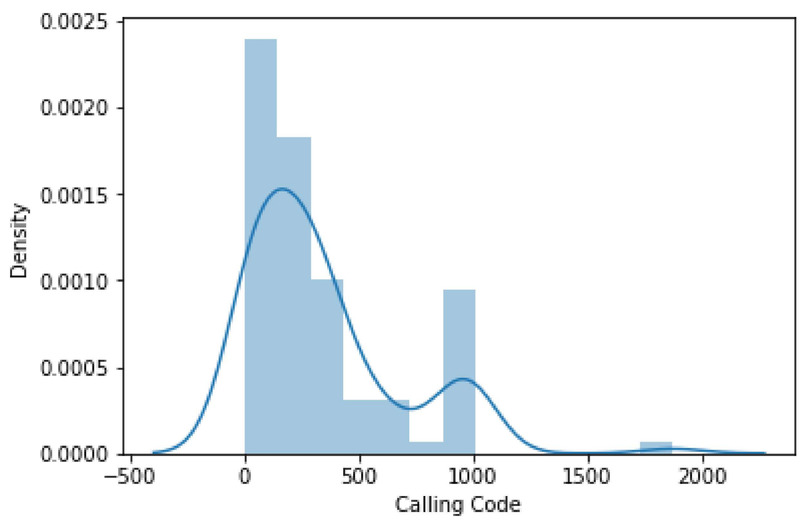
Out[9]: <seaborn.axisgrid.PairGrid at 0x19cc6314790>



In [10]:
```
sns.distplot(dff['Calling Code'])
```

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

Out[10]: <AxesSubplot:xlabel='Calling Code', ylabel='Density'>



In [11]:
```
df1=dff[['Birth Rate','Calling Code','Fertility Rate','Infant mortality','Life
'Physicians per thousand','Latitude','Longitude']]
```

In [12]:
```
sns.heatmap(df1.corr())
```

Out[12]: <AxesSubplot:>

In [13]:
```
x=df1[['Birth Rate','Fertility Rate','Infant mortality','Life expectancy','Mat
'Physicians per thousand','Latitude','Longitude']]
y=df1['Calling Code']
```

In [14]:
```
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 [15]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[15]:
```
LinearRegression()
```

In [16]:
```
print(lr.intercept_)
```

```
416.7127740323258
```

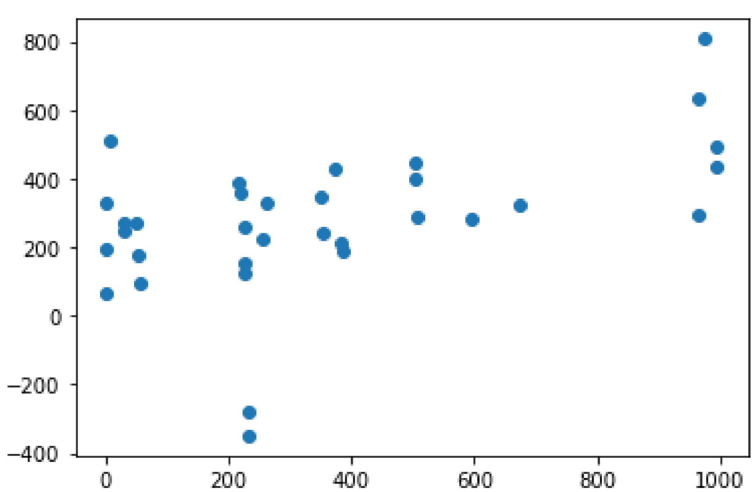
In [17]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[17]:

	Co-efficient
Birth Rate	77.700147
Fertility Rate	-406.574257
Infant mortality	-6.694317
Life expectancy	-6.924270
Maternal mortality ratio	-0.668265
Physicians per thousand	44.234071
Latitude	2.766625
Longitude	0.438515

In [18]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[18]:
```
<matplotlib.collections.PathCollection at 0x19ccb2b2460>
```



In [19]:
```
print(lr.score(x_test,y_test))
```

```
0.16036525309516636
```

In [20]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [21]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[21]:
```
Ridge(alpha=10)
```

In [22]:
```
rr.score(x_test,y_test)
```

Out[22]:
```
-0.014122169269146445
```

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

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

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

Out[24]:
```
-0.025944609713617783
```

In [25]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

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

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

```
[ 28.40978173 -25.44399664  -7.31087925  -9.60131452  -1.04809307
   0.           2.37356641   0.29012118]
```

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

```
747.6681420446432
```

In [28]:
```
print(en.predict(x_train))
```

```
[248.91473958 349.99911134 186.33973855 477.37456994  19.00882903
 384.72457823 482.32402718 472.61735015 506.63112586 235.68620279
 649.64887253 722.3255186  351.27923183 235.31923213 168.02439998
 349.34097557 618.89950169 224.34756526 202.5159874  233.47043982
 395.0938667  361.63708706 426.88263513 296.89467701 333.63377231
  98.02212061 411.07669424 298.31526548 523.70494767 347.47717453
 459.74701384 354.14878981 286.40501475 288.65100231 451.6463785
 449.9048207  309.1283756  368.83887792 305.88730825 312.77009813
 304.73609526 394.33116291 305.88040396 381.7042052  403.14569234
 343.98884965 332.144002   425.62741333 449.11465353 371.9431973
 357.77623348 307.81418194 267.86340245 270.55127196 212.7423844
 217.19838116 529.35122912 314.59675227 681.12729886 351.95990324
 207.16884578 167.9989962  226.13955704 233.99299929 210.92536441
 184.03122182  92.27125529 380.51787088 407.35698665 323.43015694
 346.96334938 189.4125089  197.20028783 227.40380025 520.82786006
 233.53535481 440.56895213]
```

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


```
[248.91473958 349.99911134 186.33973855 477.37456994  19.00882903
 384.72457823 482.32402718 472.61735015 506.63112586 235.68620279
 649.64887253 722.3255186  351.27923183 235.31923213 168.02439998
 349.34097557 618.89950169 224.34756526 202.5159874  233.47043982
 395.0938667  361.63708706 426.88263513 296.89467701 333.63377231
  98.02212061 411.07669424 298.31526548 523.70494767 347.47717453
 459.74701384 354.14878981 286.40501475 288.65100231 451.6463785
 449.9048207  309.1283756  368.83887792 305.88730825 312.77009813
 304.73609526 394.33116291 305.88040396 381.7042052  403.14569234
 343.98884965 332.144002   425.62741333 449.11465353 371.9431973
 357.77623348 307.81418194 267.86340245 270.55127196 212.7423844
 217.19838116 529.35122912 314.59675227 681.12729886 351.95990324
 207.16884578 167.9989962  226.13955704 233.99299929 210.92536441
 184.03122182  92.27125529 380.51787088 407.35698665 323.43015694
 346.96334938 189.4125089  197.20028783 227.40380025 520.82786006
 233.53535481 440.56895213]
```


In [30]:
```
print(en.score(x_train,y_train))
```

```
0.15370669879740695
```

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

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

```
Mean Absolytre Error: 225.6887110890833
```

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

```
Mean Square Error: 83732.08862496029
```

In [34]:
```
print("Root Mean Square Error:",np.sqrt(metrics.mean_absolute_error(y_test,pre
```

```
Root Mean Square Error: 15.022939495620799
```

In [ ]:
```
```