kaviyadevi 20106064

In [9]:
```
#to import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [10]:
```
#to import dataset
data1=pd.read_csv(r"C:\Users\user\Downloads\18_world-data-2023 - 18_world-data-20
data1
```

Out[10]:
```
                                                                                    Armed
                                                      Agricultural      Land       Forces  Birth  Calling
         Country  Density\n(P/Km2)  Abbreviation       Land( %)   Area(Km2)        size   Rate     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 [11]:

```
#to display top 5 rows
data=data1.head()
data
```

Out[11]:

```
                                                                                   Armed
                                                  Agricultural       Land         Forces   Birth   Calling
       Country  Density\n(P/Km2)  Abbreviation       Land( %)   Area(Km2)          size    Rate      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

5 rows × 35 columns
```

DATA CLEANING AND PREPROCESSING

In [12]:

```
#
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 35 columns):
 #   Column                                     Non-Null Count  Dtype  
---  ------                                     --------------  -----  
 0   Country                                    5 non-null      object 
 1   Density
(P/Km2)                            5 non-null      object 
 2   Abbreviation                               5 non-null      object 
 3   Agricultural Land( %)                      5 non-null      object 
 4   Land Area(Km2)                             5 non-null      object 
 5   Armed Forces size                          4 non-null      object 
 6   Birth Rate                                 5 non-null      float64
 7   Calling Code                               5 non-null      float64
 8   Capital/Major City                         5 non-null      object 
 9   Co2-Emissions                              5 non-null      object 
 10  CPI                                        4 non-null      object 
 11  CPI Change (%)                             4 non-null      object 
 12  Currency-Code                              5 non-null      object 
 13  Fertility Rate                             5 non-null      float64
 14  Forested Area (%)                          5 non-null      object 
 15  Gasoline Price                             5 non-null      object 
 16  GDP                                        5 non-null      object 
 17  Gross primary education enrollment (%)     5 non-null      object 
 18  Gross tertiary education enrollment (%)    4 non-null      object 
 19  Infant mortality                           5 non-null      float64
 20  Largest city                               5 non-null      object 
 21  Life expectancy                            4 non-null      float64
 22  Maternal mortality ratio                   4 non-null      float64
 23  Minimum wage                               5 non-null      object 
 24  Official language                          5 non-null      object 
 25  Out of pocket health expenditure           5 non-null      object 
 26  Physicians per thousand                    5 non-null      float64
 27  Population                                 5 non-null      object 
 28  Population: Labor force participation (%)  4 non-null      object 
 29  Tax revenue (%)                            4 non-null      object 
 30  Total tax rate                             4 non-null      object 
 31  Unemployment rate                          4 non-null      object 
 32  Urban_population                           5 non-null      object 
 33  Latitude                                   5 non-null      float64
 34  Longitude                                  5 non-null      float64
dtypes: float64(9), object(26)
memory usage: 1.5+ KB
```

In [13]:

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

Out[13]:

	Birth Rate	Calling Code	Fertility Rate	Infant mortality	Life expectancy	Maternal mortality ratio	Physicians per thousand	Latitude
count	5.000000	5.000000	5.000000	5.0000	4.000000	4.000000	5.000000	5.000000
mean	23.296000	256.200000	3.180000	26.0200	70.125000	251.500000	1.348000	26.885984
std	13.974456	114.850773	1.819821	22.6048	8.793321	273.791283	1.277134	22.075793
min	7.200000	93.000000	1.270000	2.7000	60.800000	15.000000	0.210000	-11.202692
25%	11.780000	213.000000	1.620000	7.8000	63.575000	87.750000	0.280000	28.033886
50%	24.280000	244.000000	3.020000	20.1000	70.600000	176.500000	1.200000	33.939110
75%	32.490000	355.000000	4.470000	47.9000	77.150000	340.250000	1.720000	41.153332
max	40.730000	376.000000	5.520000	51.6000	78.500000	638.000000	3.330000	42.506285

In [14]:

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

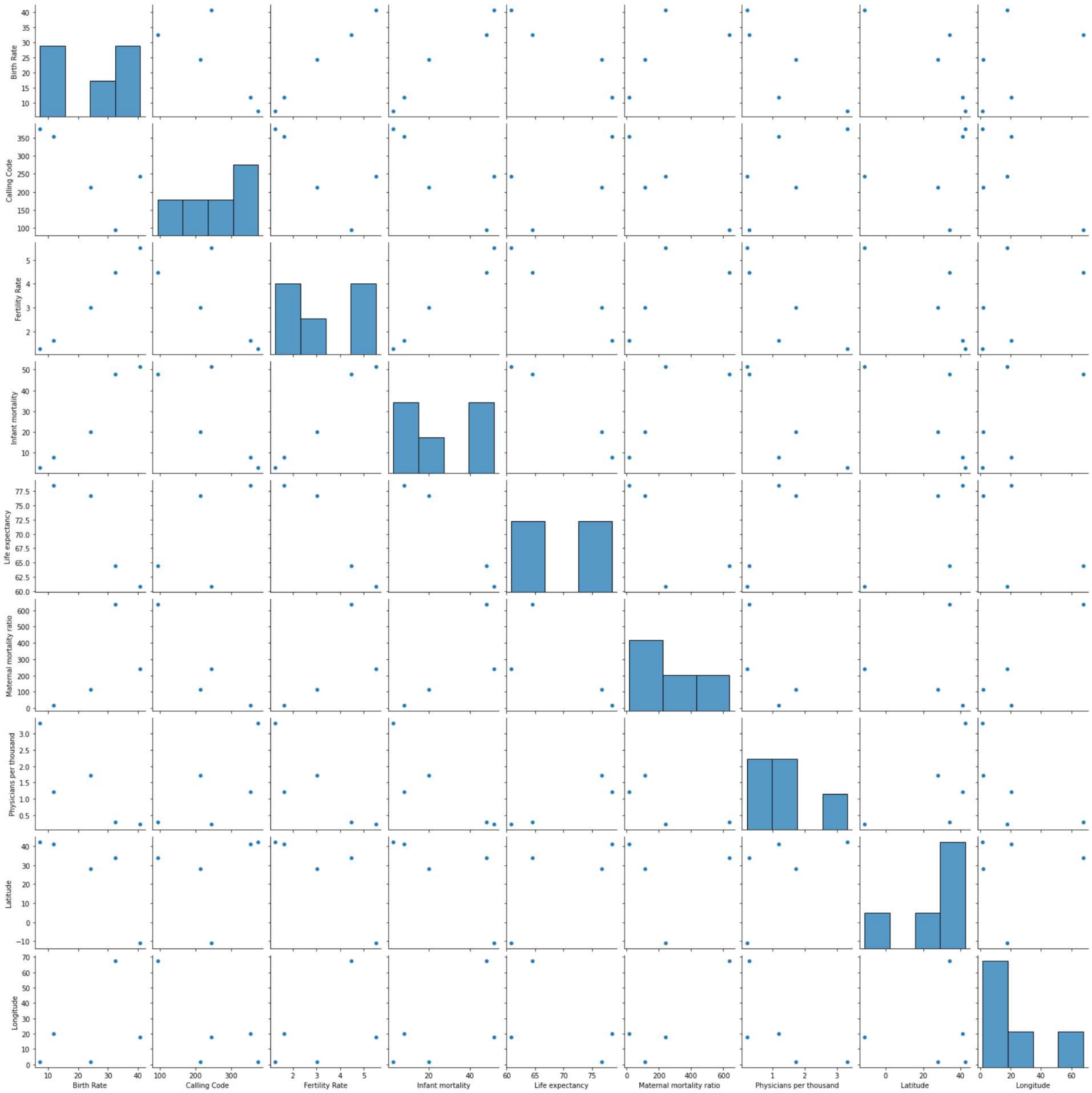
Out[14]:

```
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 expenditure',
       'Physicians per thousand', 'Population',
       'Population: Labor force participation (%)', 'Tax revenue (%)',
       'Total tax rate', 'Unemployment rate', 'Urban_population', 'Latitude',
       'Longitude'],
      dtype='object')
```

EDA and DATA VISUALIZATION

In [15]:
```
sns.pairplot(data)
```

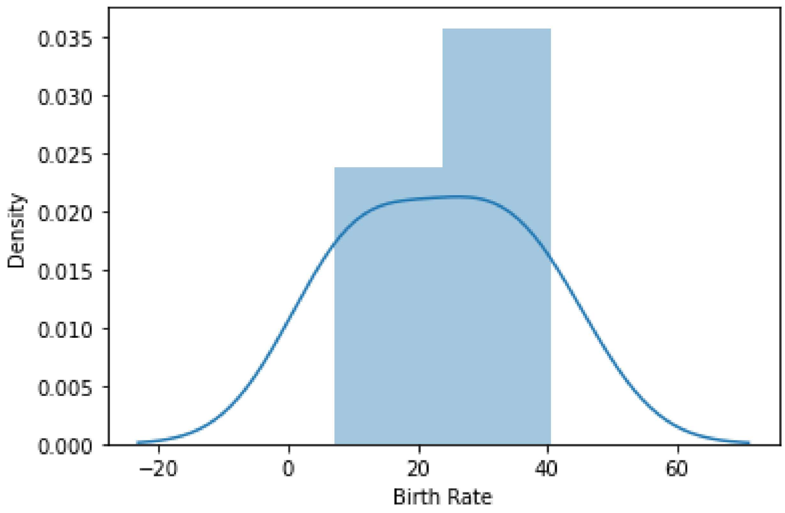
Out[15]:
```
<seaborn.axisgrid.PairGrid at 0x1839b94c8e0>
```



In [16]:
```
sns.distplot(data['Birth Rate'])
```

```
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)
```

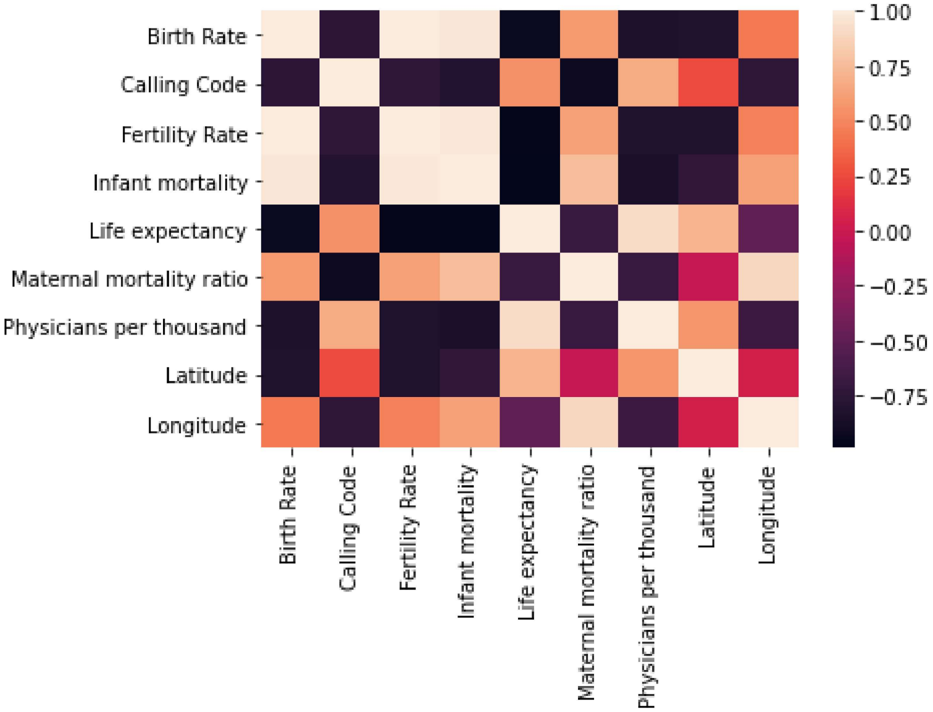
Out[16]:
```
<AxesSubplot:xlabel='Birth Rate', ylabel='Density'>
```



In [17]:
```
df=data[['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 expenditure',
       'Physicians per thousand', 'Population',
       'Population: Labor force participation (%)', 'Tax revenue (%)',
       'Total tax rate', 'Unemployment rate', 'Urban_population', 'Latitude',
       'Longitude']]
```

In [18]:
```
sns.heatmap(df.corr())
```

Out[18]:
```
<AxesSubplot:>
```



TRAINING MODEL

In [19]:
```
x=df[['Density\n(P/Km2)', 'Calling Code','Physicians per thousand','Latitude','L
y=df['Birth Rate']
```

In [20]:
```
#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 [21]:
```
from sklearn.linear_model import LinearRegression

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

Out[21]:
```
LinearRegression()
```

In [22]:
```
#to find intercept
print(lr.intercept_)
```

```
40.53209455498641
```

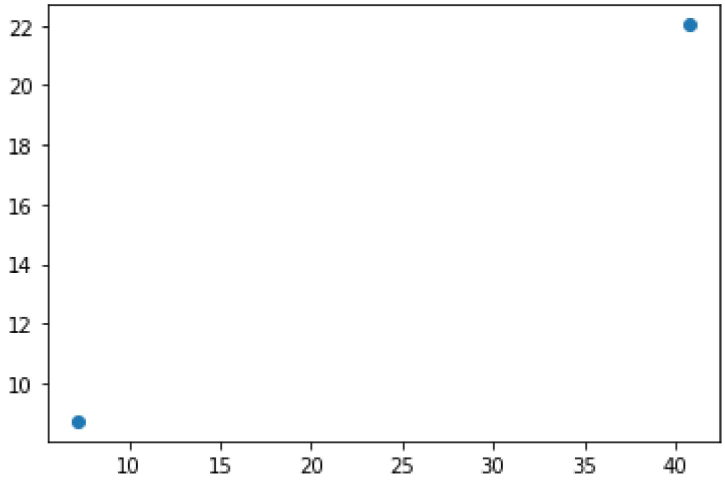
In [23]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[23]:

	Co-efficient
Density\n(P/Km2)	-0.023768
Calling Code	-0.073853
Physicians per thousand	-0.000079
Latitude	-0.003664
Longitude	0.005563

In [24]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[24]:
```
<matplotlib.collections.PathCollection at 0x183a0057b50>
```



In [25]:
```
print(lr.score(x_test,y_test))
```

```
0.3741205537460398
```

RIDGE AND LASSO REGRESSION

In [26]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [27]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[27]:
```
Ridge(alpha=10)
```

In [28]:
```
rr.score(x_test,y_test)
```

Out[28]:
```
0.3740314910611825
```

In [29]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[29]:
```
Lasso(alpha=10)
```

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

Out[30]:
```
0.3248845431918559
```

In [31]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[31]:
```
ElasticNet()
```

In [32]:
```
print(en.coef_)
```

```
[-0.01990074 -0.07558713  0.          -0.          -0.         ]
```

In [33]:
```
print(en.predict(x_test))
```

```
[ 9.03382805 21.75763045]
```

In [34]:
```
print(en.score(x_test,y_test))
```

```
0.35368430614607294
```

In [35]:
```
from sklearn import metrics
```

In [36]:
```
print("Mean Absolute error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute error 10.10670295627709
```

In [37]:
```
print("Mean Squared error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared error 175.9129473842447
```

In [38]:
```
print("Root Mean Absolute error",np.sqrt(metrics.mean_squared_error(y_test,predic
```

```
Root Mean Absolute error 13.263217836718384
```