Data collection

Importing libraries

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

Importing dataset

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

Out[2]:

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

head

In [3]:
```
# to display first 8 dataset values
da=data.head(8)
da
```

Out[3]:

	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Calling Code	Capital
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	An
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	St Sa
6	Argentina	17	AR	54.30%	2,780,400	105,000	17.02	54.0	Buen
7	Armenia	104	AM	58.90%	29,743	49,000	13.99	374.0	

8 rows × 35 columns

info

In [4]:
```
# to identify missing values
data.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
```

describe

In [5]:
```
# to display summary of the dataset
data.describe()
```

Out[5]:

	Birth Rate	Calling Code	Fertility Rate	Infant mortality	Life expectancy	Maternal mortality ratio	Physicians per thousand	Latitude
count	189.000000	194.000000	188.000000	189.000000	187.000000	181.000000	188.000000	194.000000
mean	20.214974	360.546392	2.698138	21.332804	72.279679	160.392265	1.839840	19.092351
std	9.945774	323.236419	1.282267	19.548058	7.483661	233.502024	1.684261	23.961779
min	5.900000	1.000000	0.980000	1.400000	52.800000	2.000000	0.010000	-40.900557
25%	11.300000	82.500000	1.705000	6.000000	67.000000	13.000000	0.332500	4.544175
50%	17.950000	255.500000	2.245000	14.000000	73.200000	53.000000	1.460000	17.273849
75%	28.750000	506.750000	3.597500	32.700000	77.500000	186.000000	2.935000	40.124603
max	46.080000	1876.000000	6.910000	84.500000	85.400000	1150.000000	8.420000	64.963051

columns

In [6]:
```
# to display headings of the dataset
data.columns
```

Out[6]:
```
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')
```

In [7]:
```
a=data.dropna(axis=1,how='any')
a
```

Out[7]:

	Country	Density\n(P/Km2)
0	Afghanistan	60
1	Albania	105
2	Algeria	18
3	Andorra	164
4	Angola	26
...	...	...
190	Venezuela	32
191	Vietnam	314
192	Yemen	56
193	Zambia	25
194	Zimbabwe	38

195 rows × 2 columns

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

Out[8]:
```
Index(['Country', 'Density\n(P/Km2)'], dtype='object')
```

In [9]:
```
b=data.head(8)
b
```

Out[9]:

	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Calling Code	Capital
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	An
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	St Sa
6	Argentina	17	AR	54.30%	2,780,400	105,000	17.02	54.0	Buen

	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Calling Code	Capital
7	Armenia	104	AM	58.90%	29,743	49,000	13.99	374.0	

8 rows × 35 columns

In [10]:
```
c=b.fillna(value=6)
c
```

Out[10]:

	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Calling Code	Capital
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	6	7.20	376.0	An
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	St Sa
6	Argentina	17	AR	54.30%	2,780,400	105,000	17.02	54.0	Buen
7	Armenia	104	AM	58.90%	29,743	49,000	13.99	374.0	

8 rows × 35 columns

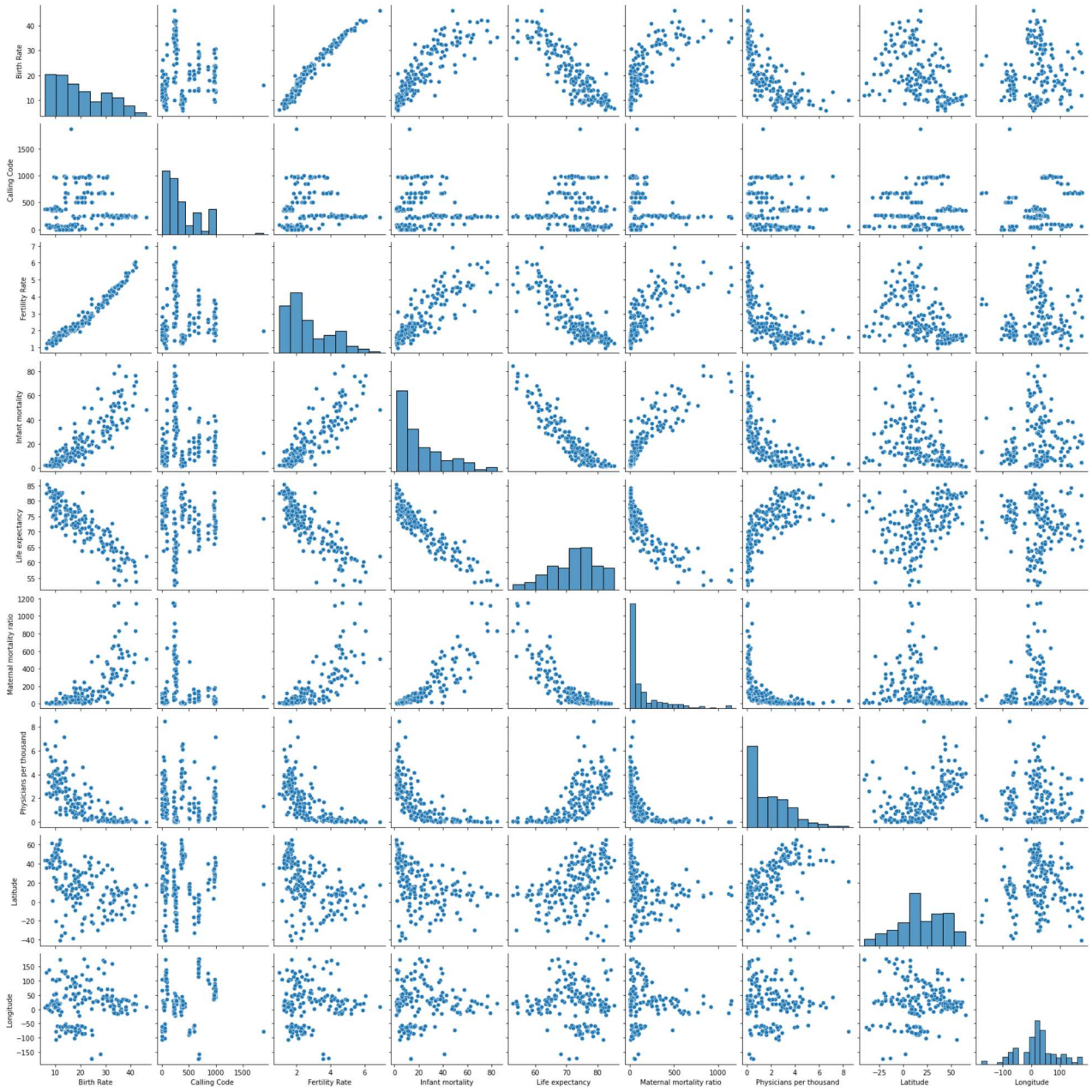
In [11]:
```
c.columns
```

Out[11]:
```
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 Visualization

In [12]:
```
sns.pairplot(data)
```

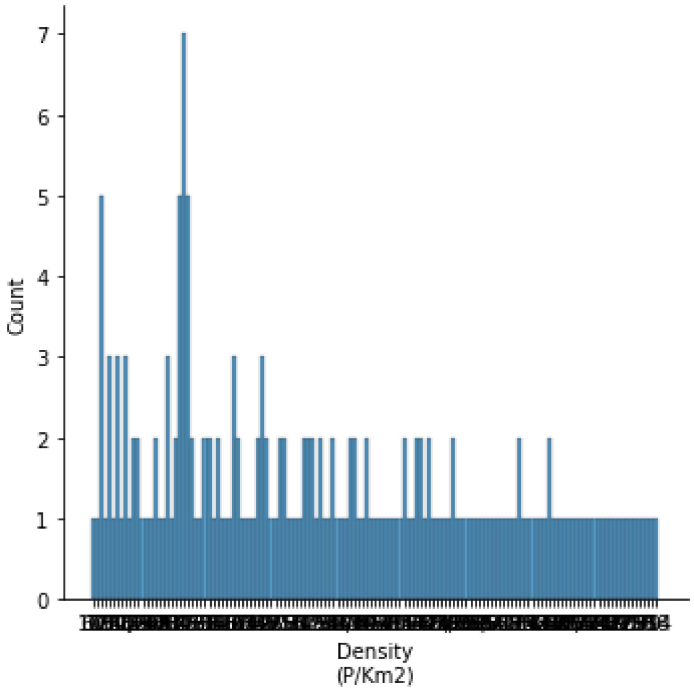
Out[12]:
```
<seaborn.axisgrid.PairGrid at 0x1c226bbf670>
```



distribution plot

In [13]:
```
sns.displot(a["Density\n(P/Km2)"])
```

Out[13]:
```
<seaborn.axisgrid.FacetGrid at 0x1c228f22fd0>
```



correlation

In [14]:
```
dat=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']]
sns.heatmap(dat.corr())
```

Out[14]: <AxesSubplot:>

To train the model-Model Building

In [15]:
```
x=c[['Density\n(P/Km2)']]
y=c['Density\n(P/Km2)']
```

In [16]:
```
# to split my dataset into training and test data
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]:
```
print(lr.intercept_)
```

```
-5.684341886080802e-14
```

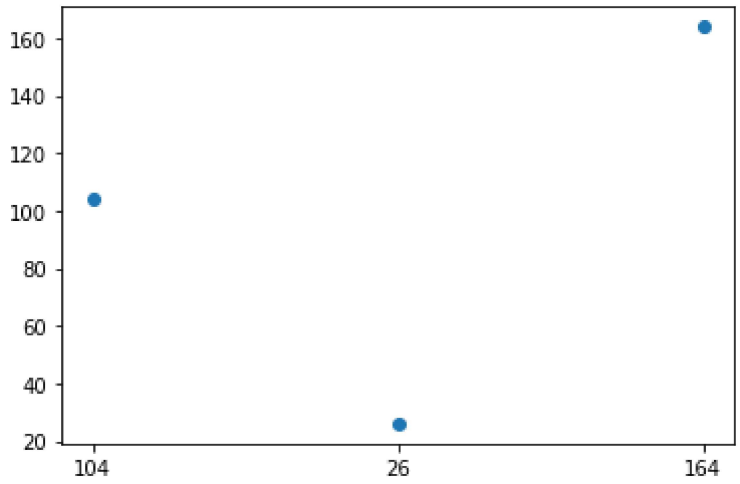
In [19]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[19]:

	Co-efficient
Density\n(P/Km2)	1.0

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

Out[20]: <matplotlib.collections.PathCollection at 0x1c22bdc3340>



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

1.0

In [22]:
```
lr.score(x_train,y_train)
```

Out[22]: 1.0

Ridge regression

In [23]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [24]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[24]: 0.9999998760449819

In [25]:
```
rr.score(x_train,y_train)
```

Out[25]: 0.9999998826464848

Lasso regression

In [26]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
la.score(x_train,y_train)
```

Out[26]: 0.999997064151002

In [27]:
```
la.score(x_test,y_test)
```

Out[27]:
```
0.9999968990002983
```

In [28]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[28]:
```
ElasticNet()
```

In [29]:
```
print(en.coef_)
```

```
[0.99982867]
```

In [30]:
```
print(en.intercept_)
```

```
0.014494392417674362
```

In [31]:
```
predict=en.predict(x_test)
```

In [32]:
```
print(en.score(x_test,y_test))
```

```
0.9999999689953156
```

In [33]:
```
from sklearn import metrics
```

In [34]:
```
print("Mean Absolute error:",metrics.mean_absolute_error(y_test,predict))
```

```
Mean Absolute error: 0.008989036117179458
```

In [35]:
```
print("Mean Squared error:",metrics.mean_squared_error(y_test,predict))
```

```
Mean Squared error: 9.896695243781906e-05
```

In [36]:
```
print("Root squared error:",np.sqrt(metrics.mean_squared_error(y_test,predict)))
```

```
Root squared error: 0.00994821352996703
```