Type *Markdown* and LaTeX: α^2

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

In [2]:
```
#import dataset
df=pd.read_csv(r"E:\154\18_world-data-2023.csv").dropna()
df
```

Out[2]:

	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Ca C
0	Afghanistan	60	AF	58.10%	652,230	323,000	32.49	
1	Albania	105	AL	43.10%	28,748	9,000	11.78	3
2	Algeria	18	DZ	17.40%	2,381,741	317,000	24.28	2
4	Angola	26	AO	47.50%	1,246,700	117,000	40.73	2
6	Argentina	17	AR	54.30%	2,780,400	105,000	17.02	
...	...	...	...	...	...	...	...	
185	United Kingdom	281	GB	71.70%	243,610	148,000	11.00	
186	United States	36	US	44.40%	9,833,517	1,359,000	11.60	

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 110 entries, 0 to 193
Data columns (total 35 columns):
 #   Column                                   Non-Null Count  Dtype
---  ------                                   --------------  -----
 0   Country                                  110 non-null    object
 1   Density
(P/Km2)                          110 non-null    object
 2   Abbreviation                             110 non-null    object
 3   Agricultural Land( %)                    110 non-null    object
 4   Land Area(Km2)                           110 non-null    object
 5   Armed Forces size                        110 non-null    object
 6   Birth Rate                               110 non-null    float64
 7   Calling Code                             110 non-null    float64
 8   Capital/Major City                       110 non-null    object
 9   Co2-Emissions                            110 non-null    object
 10  CPI                                      110 non-null    object
 11  CPI Change (%)                           110 non-null    object
 12  Currency-Code                            110 non-null    object
```

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

Out[4]:

	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
4	Angola	26	AO	47.50%	1,246,700	117,000	40.73	244.0
6	Argentina	17	AR	54.30%	2,780,400	105,000	17.02	54.0

5 rows × 35 columns

Data cleaning and Pre-Processing

In [5]:
```
#To find null values
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 110 entries, 0 to 193
Data columns (total 35 columns):
 #   Column                                    Non-Null Count  Dtype  
---  ------                                    --------------  -----  
 0   Country                                   110 non-null    object 
 1   Density
(P/Km2)                          110 non-null    object 
 2   Abbreviation                              110 non-null    object 
 3   Agricultural Land( %)                     110 non-null    object 
 4   Land Area(Km2)                            110 non-null    object 
 5   Armed Forces size                         110 non-null    object 
 6   Birth Rate                                110 non-null    float64
 7   Calling Code                              110 non-null    float64
 8   Capital/Major City                        110 non-null    object 
 9   Co2-Emissions                             110 non-null    object 
 10  CPI                                       110 non-null    object 
 11  CPI Change (%)                            110 non-null    object 
 12  Currency-Code                             110 non-null    object 
 13  Fertility Rate                            110 non-null    float64
 14  Forested Area (%)                         110 non-null    object 
 15  Gasoline Price                            110 non-null    object 
 16  GDP                                       110 non-null    object 
 17  Gross primary education enrollment (%)    110 non-null    object 
 18  Gross tertiary education enrollment (%)   110 non-null    object 
 19  Infant mortality                          110 non-null    float64
 20  Largest city                              110 non-null    object 
 21  Life expectancy                           110 non-null    float64
 22  Maternal mortality ratio                  110 non-null    float64
 23  Minimum wage                              110 non-null    object 
 24  Official language                         110 non-null    object 
 25  Out of pocket health expenditure          110 non-null    object 
 26  Physicians per thousand                   110 non-null    float64
 27  Population                                110 non-null    object 
 28  Population: Labor force participation (%) 110 non-null    object 
 29  Tax revenue (%)                           110 non-null    object 
 30  Total tax rate                            110 non-null    object 
 31  Unemployment rate                         110 non-null    object 
 32  Urban_population                          110 non-null    object 
 33  Latitude                                  110 non-null    float64
 34  Longitude                                 110 non-null    float64
dtypes: float64(9), object(26)
memory usage: 30.9+ KB
```

In [6]:
```
# To display summary of statistics
df.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

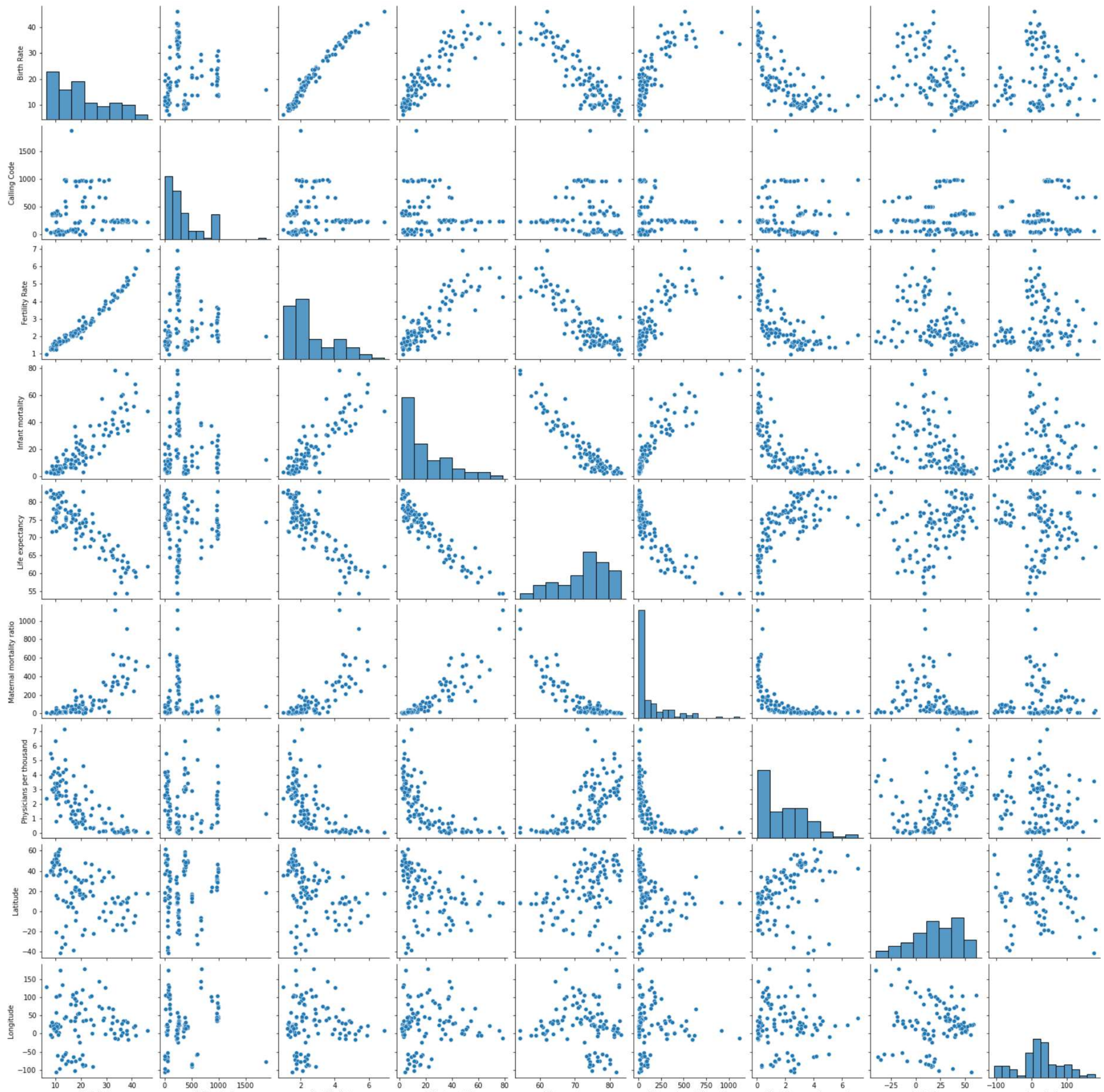
In [7]:
```
#To Display column heading
df.columns
```

Out[7]:
```
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')
```

EDA and VISUALIZATION

In [8]:
```
sns.pairplot(df)
```

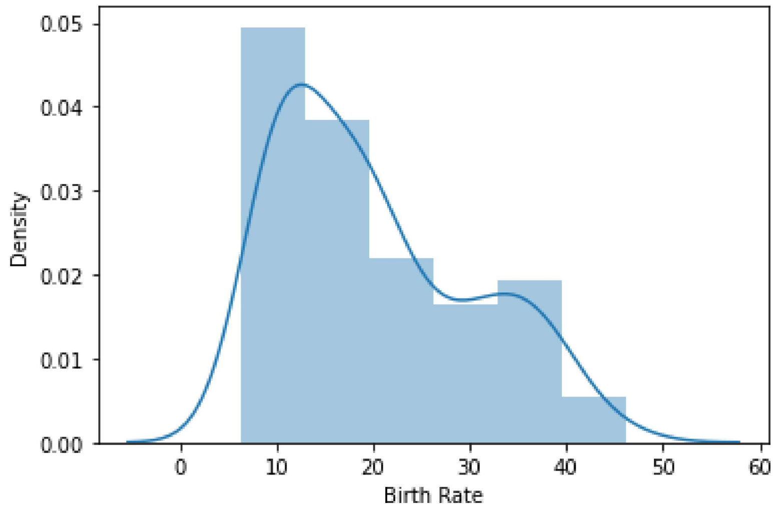
Out[8]: <seaborn.axisgrid.PairGrid at 0x1b59f578a00>



In [9]:
```
sns.distplot(df['Birth Rate'])
```

```
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[9]:
```
<AxesSubplot:xlabel='Birth Rate', ylabel='Density'>
```



In [10]:
```
df1=df[['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']]
```

Plot Using Heat Map

In [11]:
```
sns.heatmap(df1.corr())
```

Out[11]:
```
<AxesSubplot:>
```

To Train The Model-Model Building

we are going to train Linera Regression Model;We need to split out data into two variables x and y where x is independent variable(input) and y is dependent on x(output) we could ignore address column as it required for our model

In [12]:
```
x=df1[['Birth Rate', 'Calling Code',
        'Fertility Rate', 'Infant mortality',
       'Life expectancy', 'Maternal mortality ratio','Latitude','Physicians pe
       'Longitude'
       ]]
y=df1['Physicians per thousand']
```

To Split my dataset into training and test data

In [13]:
```

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

Out[14]:
```
LinearRegression()
```

In [15]:
```
lr.intercept_
```

Out[15]:
```
-2.531308496145357e-14
```

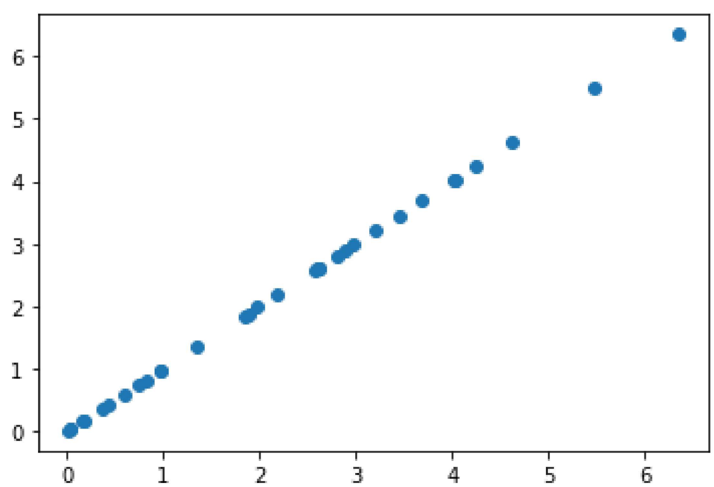
In [16]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
Birth Rate	-1.762563e-15
Calling Code	1.752775e-17
Fertility Rate	1.289564e-14
Infant mortality	1.704309e-16
Life expectancy	1.996199e-16
Maternal mortality ratio	9.188721e-18
Latitude	-4.783119e-17
Physicians per thousand	1.000000e+00
Longitude	-4.271240e-18

In [17]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[17]:
```
<matplotlib.collections.PathCollection at 0x1b5a44a7160>
```



Accuracy

In [18]:
```
lr.score(x_test,y_test)
```

Out[18]:
```
1.0
```

In [19]:
```
lr.score(x_train,y_train)
```

Out[19]:
```
1.0
```

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]:
```
0.9947727105233788
```

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]:
```
0.3593921547432203
```

ElasticNet

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

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

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

```
[-3.27421153e-02  6.00299313e-05 -0.00000000e+00 -1.68899798e-02
  0.00000000e+00  3.06774698e-06  1.52597188e-03  3.87425791e-01
 -7.30657009e-04]
```

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

```
2.1209864282200077
```

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

```
[ 0.90933623  1.25937394  3.25945677  1.72304163  2.91248742  4.28695572
  2.94917222  3.35942739  3.47754088  2.33725144  2.87836436  2.89229255
  2.87829374  1.23539902  0.39479642  2.78442559  0.61122873  1.74481716
 -0.1468399   2.76140535  2.19833086  0.71336975  2.6189586   0.42998079
  3.21240503  1.67219904  1.35043644  2.76117253  2.61395252  3.96355486
  0.26986623  2.39005891  1.55907402]
```

In [29]:
```
print(en.score(x_test,y_test))
```

```
0.831506599003479
```

Evaluation Metrics

In [30]:
```
from sklearn import metrics
```

In [31]:
```
print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error 5.374436165466448e-15
```

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

```
Mean Squared Error: 4.0151772251877673e-29
```

In [33]:
```
print("Root Mean Absolute Error:",np.sqrt(metrics.mean_squared_error(y_test,pr
```

```
Root Mean Absolute Error: 6.336542610278706e-15
```