```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression,Lasso,Ridge


from sklearn.linear_model import ElasticNet
from sklearn import metrics


df=pd.read_csv("/content/14_Iris.csv")
df
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

```
df=df.drop(["Species"],axis=1)
df
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	1	5.1	3.5	1.4	0.2
1	2	4.9	3.0	1.4	0.2
2	3	4.7	3.2	1.3	0.2
3	4	4.6	3.1	1.5	0.2
4	5	5.0	3.6	1.4	0.2
...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3
146	147	6.3	2.5	5.0	1.9
147	148	6.5	3.0	5.2	2.0
148	149	6.2	3.4	5.4	2.3
149	150	5.9	3.0	5.1	1.8

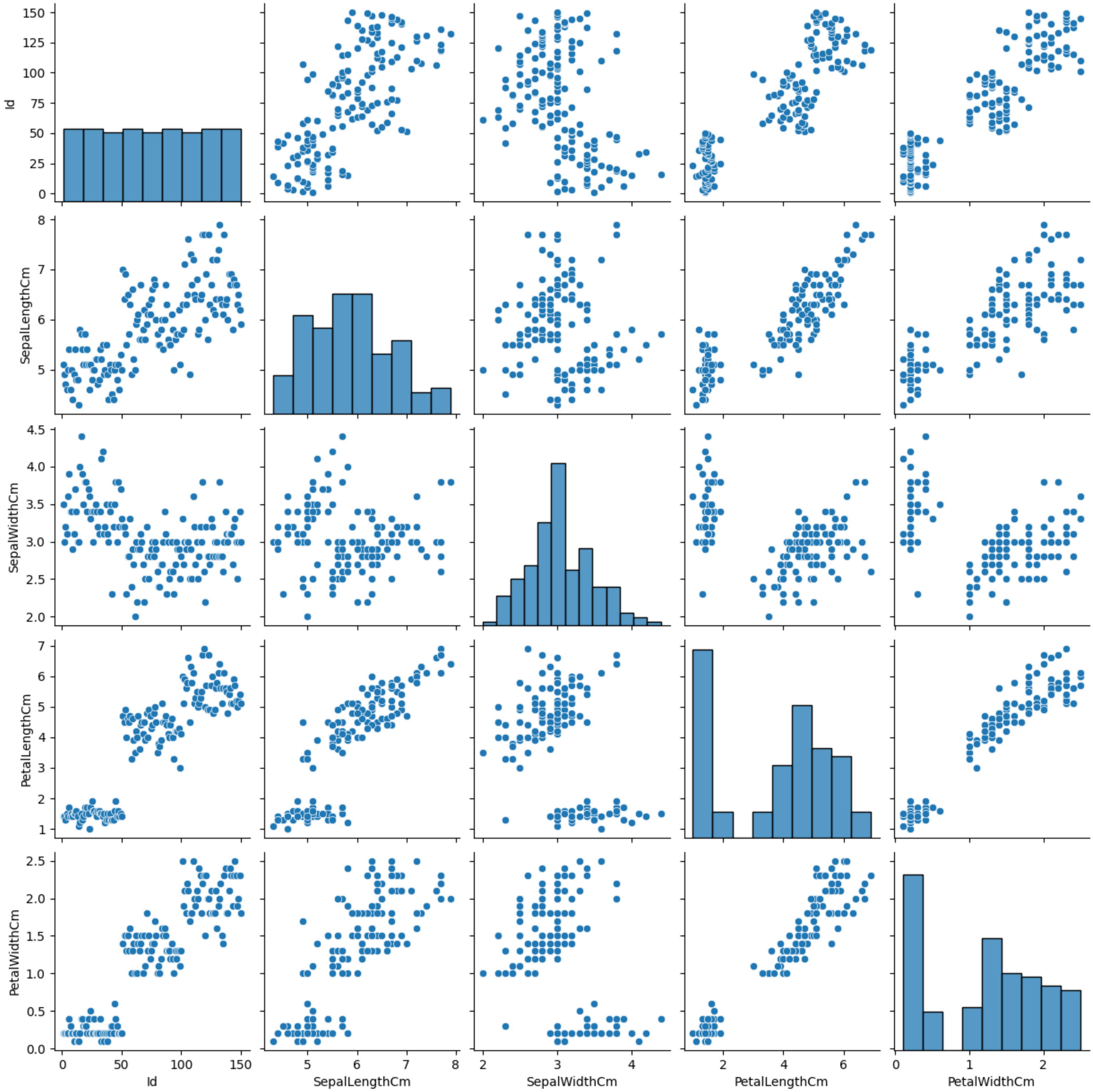
150 rows × 5 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
dtypes: float64(4), int64(1)
memory usage: 6.0 KB
```

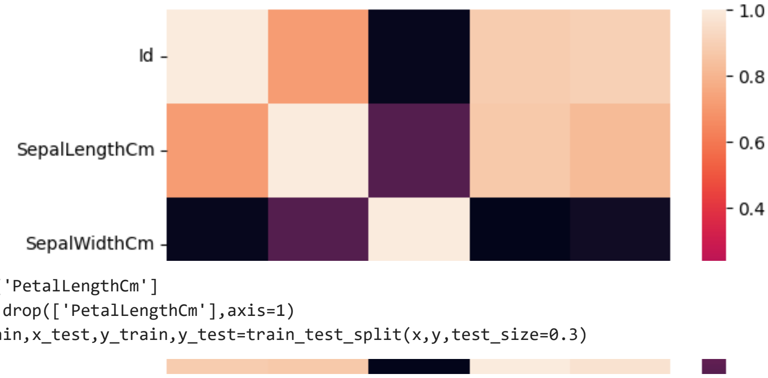
```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7bbcbb81fb50>
```




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

```
<Axes: >
```




```
y=df['PetalLengthCm']
x=df.drop(['PetalLengthCm'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
model=LinearRegression()
model.fit(x_train,y_train)
model.intercept_
```

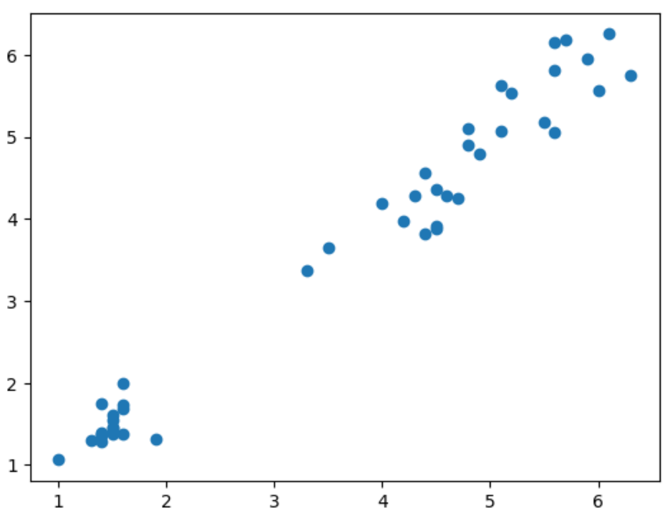
```
2.3117333487898146
```

```
coeff=pd.DataFrame(model.coef_,x.columns,columns=["Coefficient"])
coeff
```

	Coefficient
Id	0.002886
SepalLengthCm	0.685540
SepalWidthCm	-0.567305
PetalWidthCm	1.373958

```
prediction=model.predict(x_test)
plt.scatter(y_test,prediction)
```

```
<matplotlib.collections.PathCollection at 0x7bbcb660ccd0>
```




```
model.score(x_test,y_test)
```

```
0.9703186643149012
```

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
la=Lasso(alpha=10)
la.fit(x_train,y_train)
print(rr.score(x_test,y_test))
la.score(x_test,y_test)
```

```
0.9556103438378818
0.7848568895304222
```

```
en=ElasticNet()
en.fit(x_train,y_train)
print(en.coef_)
print(en.intercept_)
print(en.predict(x_test))
```

```
print(en.score(x_test,y_test))
```

```
[ 0.03574257  0.         -0.          0.        ]
1.082777008588335
[6.12247995 6.1939651  2.44099482 3.65624234 1.90485621 6.0509948
 5.83653936 4.80000471 5.62208391 4.58554926 3.90644036 1.65465819
 4.94297501 6.26545024 4.37109382 3.4417869  1.97634136 4.47832154
 2.72693541 2.29802453 2.65545027 5.26465817 3.37030175 5.19317302
 1.40446017 4.76426214 2.97713343 5.01446015 4.33535125 2.04782651
 4.1208958  3.08436115 1.54743047 2.08356908 2.3337671  2.01208393
 1.19000473 1.44020275 6.37267797 3.94218293 5.58634134 2.15505423
 4.44257897 1.3687176  1.11851958]
0.8022268907230128
```

```
print("MAE",metrics.mean_absolute_error(y_test,prediction))
print("MSE",metrics.mean_squared_error(y_test,prediction))
print("RMSE",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
MAE 0.2378939307972533
MSE 0.09547657543987052
RMSE 0.30899284043464587
```

```
df1=pd.read_csv("/content/16_Sleep_health_and_lifestyle_dataset.csv")
df1
```

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blood Pressure	Heart Rate	Daily Steps	Sleep Disorder
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	126/83	77	4200	None
1	2	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	None
2	3	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	None
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3000	Sleep Apnea
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3000	Sleep Apnea
...	...	...	...	...	...	...	...	...	...	...	...	...	...
369	370	Female	59	Nurse	8.1	9	75	3	Overweight	140/95	68	7000	Sleep Apnea
370	371	Female	59	Nurse	8.0	9	75	3	Overweight	140/95	68	7000	Sleep Apnea
371	372	Female	59	Nurse	8.1	9	75	3	Overweight	140/95	68	7000	Sleep Apnea
372	373	Female	59	Nurse	8.1	9	75	3	Overweight	140/95	68	7000	Sleep Apnea
373	374	Female	59	Nurse	8.1	9	75	3	Overweight	140/95	68	7000	Sleep Apnea

```
df2=df1.drop(["Gender","Occupation","BMI Category","Sleep Disorder","Blood Pressure"],axis=1)
df2
```

	Person ID	Age	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	Heart Rate	Daily Steps
0	1	27	6.1	6	42	6	77	4200
1	2	28	6.2	6	60	8	75	10000
2	3	28	6.2	6	60	8	75	10000
3	4	28	5.9	4	30	8	85	3000
4	5	28	5.9	4	30	8	85	3000
...	...	...	...	...	...	...	...	...
369	370	59	8.1	9	75	3	68	7000
370	371	59	8.0	9	75	3	68	7000
371	372	59	8.1	9	75	3	68	7000
372	373	59	8.1	9	75	3	68	7000
373	374	59	8.1	9	75	3	68	7000

374 rows × 8 columns

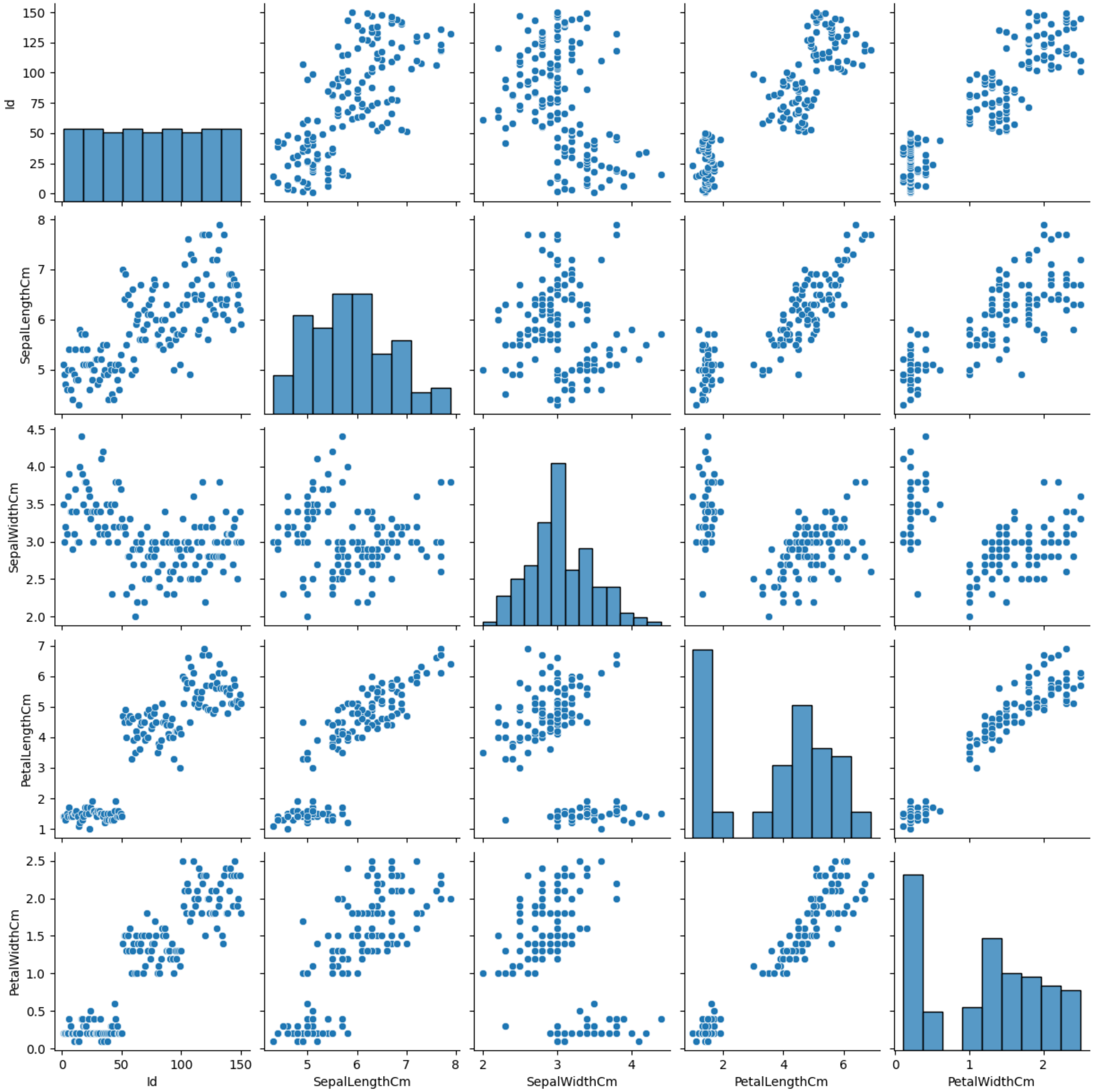
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 374 entries, 0 to 373
Data columns (total 8 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Person ID                374 non-null    int64  
 1   Age                      374 non-null    int64  
 2   Sleep Duration           374 non-null    float64
 3   Quality of Sleep         374 non-null    int64  
```

```
 4   Physical Activity Level  374 non-null    int64
 5   Stress Level             374 non-null    int64
 6   Heart Rate               374 non-null    int64
 7   Daily Steps              374 non-null    int64
dtypes: float64(1), int64(7)
memory usage: 23.5 KB
```

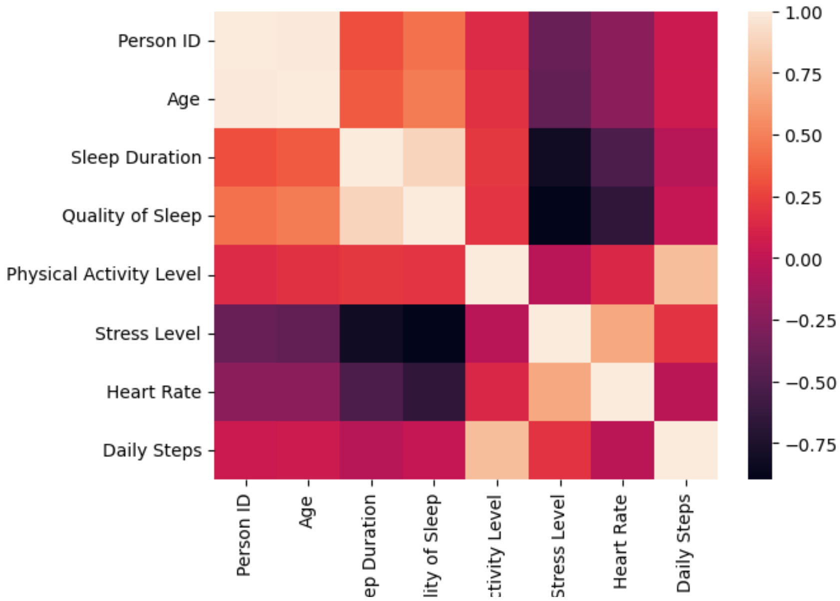
```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7bbcb6659ae0>
```




```
sns.heatmap(df1.corr())
```

```
<ipython-input-21-3ed1a1a51dc0>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to Fa
  sns.heatmap(df1.corr())
<Axes: >
```




```
y=df2['Age']
x=df2.drop(['Age'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
model=LinearRegression()
model.fit(x_train,y_train)
model.intercept_
```

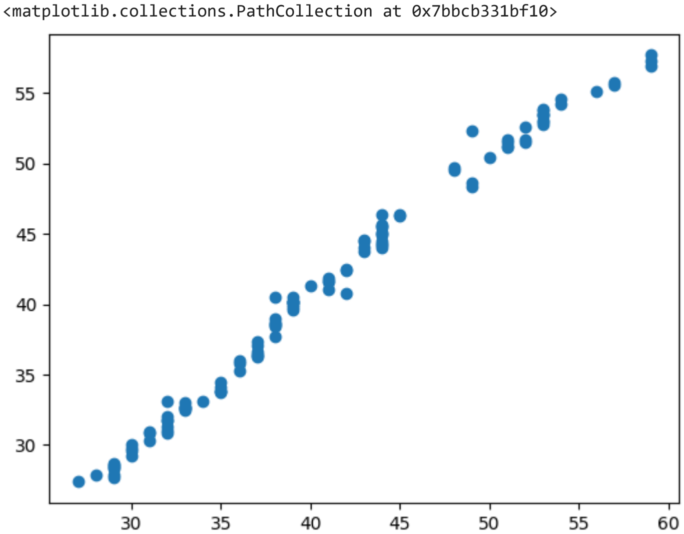
```
10.035480555924565
```

```
coeff=pd.DataFrame(model.coef_,x.columns,columns=["Coefficient"])
coeff
```

	Coefficient
Person ID	0.077793
Sleep Duration	0.257935
Quality of Sleep	0.663167
Physical Activity Level	-0.006079
Stress Level	0.021597
Heart Rate	0.147178
Daily Steps	0.000123

```
prediction=model.predict(x_test)
plt.scatter(y_test,prediction)
```

```
<matplotlib.collections.PathCollection at 0x7bbcb331bf10>
```



```
model.score(x_test,y_test)
```

```
0.9874371634120617
```

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
la=Lasso(alpha=10)
la.fit(x_train,y_train)
print(rr.score(x_test,y_test))
la.score(x_test,y_test)
```

```
0.987513778843579
0.9855327459608795
```

```
en=ElasticNet()
en.fit(x_train,y_train)
print(en.coef_)
print(en.intercept_)
print(en.predict(x_test))
print(en.score(x_test,y_test))
```

```
[ 0.07933532  0.          0.          0.02173581 -0.00039562 -0.
 -0.00015546]
27.09716628433617
[39.64852614 52.67539577 52.19938387 39.92864189 49.14643485 55.08796387
 31.33221752 41.77074584 53.7860902  32.00341045 53.46874894 51.80270729
 39.21462404 52.04071324 51.61355933 39.61130062 38.49779126 33.48665067
 45.70421182 50.89954148 37.46643215 34.52460433 28.27465058 56.51599957
 44.38881129 39.46927453 31.57022347 36.09708388 30.73404539 50.34420337
 33.75821058 46.33889435 52.59606045 47.37144033 33.07759449 27.53375398
 34.92770202 57.30935274 45.54554119 29.62335096 45.46739273 43.91279939
 39.69063594 37.86592366 37.94525898 46.97357688 46.02273995 34.13434885
 50.69201286 38.33912063 32.40882853 52.75473109 31.7654045  33.31560044
 32.80068929 53.07207235 36.2764024  36.11773177 36.1764192  45.3087221
 32.76025322 56.91267615 29.46468032 38.42127088 27.97506614 30.25803349
 35.64171987 44.99138083 55.32596982 41.53273989 39.45262999 31.17354689
 39.14751461 49.0748292  37.06975557 48.83682325 47.45077564 51.39610235
 45.15005147 40.08705993 46.18022372 48.98776421 35.48304923 32.99825917
 54.70961478 52.12004855 32.83958854 41.26143409 40.97739268 50.85068349
 28.8493599  41.45340457 51.49568993 51.73369588 27.37508335 33.67887526
 46.49875185 43.93238064 28.19531526 28.69068926 29.14733906 44.54748192
 27.43409942 34.76261028 29.86135691 36.7524143  52.99273704 32.04623537
 27.08155485 42.40542837 42.24675774 30.89271602 39.56919083]
0.9854908342571518
```

```
print("MAE",metrics.mean_absolute_error(y_test,prediction))
print("MSE",metrics.mean_squared_error(y_test,prediction))
print("RMSE",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
MAE 0.7555237315881547
MSE 0.8988552653896111
RMSE 0.9480797779668181
```

```
df3=pd.read_csv("/content/17_student_marks.csv")
df3
```

	Student_ID	Test_1	Test_2	Test_3	Test_4	Test_5	Test_6	Test_7	Test_8	Test_9	Test_10	Test_11	Test_12
0	22000	78	87	91	91	88	98	94	100	100	100	100	93
1	22001	79	71	81	72	73	68	59	69	59	60	61	67
2	22002	66	65	70	74	78	86	87	96	88	82	90	86
3	22003	60	58	54	61	54	57	64	62	72	63	72	76
4	22004	99	95	96	93	97	89	92	98	91	98	95	88
5	22005	41	36	35	28	35	36	27	26	19	22	27	31
6	22006	47	50	47	57	62	64	71	75	85	87	85	89
7	22007	84	74	70	68	58	59	56	56	64	70	67	59
8	22008	74	64	58	57	53	51	47	45	42	43	34	24
9	22009	87	81	73	74	71	63	53	45	39	43	46	38
10	22010	40	34	37	33	31	35	39	38	40	48	44	50
11	22011	91	84	78	74	76	80	80	73	75	71	79	70
12	22012	81	83	93	88	89	90	99	99	95	85	75	84
13	22013	52	50	42	38	33	30	28	22	12	20	19	20
14	22014	63	67	65	74	80	86	95	96	92	83	75	81
15	22015	76	82	88	94	85	76	70	60	50	58	49	59
16	22016	83	78	71	71	77	72	66	75	66	61	61	66
17	22017	55	45	43	38	43	35	44	37	45	37	45	54
18	22018	71	67	76	74	64	61	57	64	61	51	51	58
19	22019	62	61	53	49	54	59	68	74	65	55	60	61
20	22020	44	38	36	34	26	34	39	44	36	45	35	44
21	22021	50	56	53	46	41	38	47	39	44	36	43	46
22	22022	57	48	40	45	43	36	26	19	9	12	22	27
23	22023	59	56	52	44	50	40	45	46	54	57	52	47
24	22024	84	92	89	80	90	80	84	74	68	73	81	74
25	22025	74	80	86	87	90	100	95	87	85	79	85	88
26	22026	92	84	74	83	93	83	75	82	81	73	70	73
27	22027	63	70	74	65	64	55	61	58	48	46	46	51
28	22028	78	77	69	76	78	74	67	69	78	68	65	68
29	22029	55	58	59	67	71	62	53	61	67	76	75	70
30	22030	54	54	48	38	35	45	46	47	41	37	30	25
31	22031	84	93	97	89	86	95	100	100	100	99	100	100
32	22032	95	100	94	100	98	99	100	90	80	84	75	80
33	22033	64	61	63	73	63	68	64	58	50	51	56	64
34	22034	76	79	73	77	83	86	95	89	90	95	100	100
35	22035	78	71	61	55	54	48	41	32	41	40	48	38
36	22036	95	89	91	84	89	94	85	91	100	100	100	92
37	22037	99	89	79	87	87	81	82	74	64	54	51	50
38	22038	82	83	85	86	89	80	88	95	87	93	90	89
39	22039	65	56	64	62	58	51	61	68	70	70	63	73

```
df3.info()
```

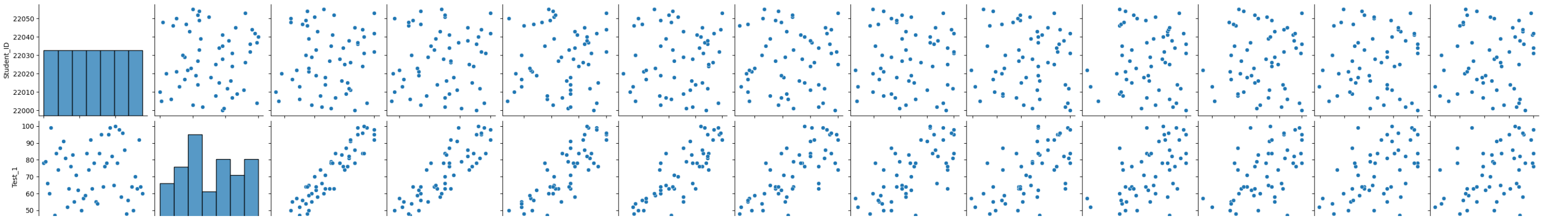
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56 entries, 0 to 55
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Student_ID  56 non-null     int64
 1   Test_1      56 non-null     int64
 2   Test_2      56 non-null     int64
 3   Test_3      56 non-null     int64
 4   Test_4      56 non-null     int64
 5   Test_5      56 non-null     int64
 6   Test_6      56 non-null     int64
 7   Test_7      56 non-null     int64
 8   Test_8      56 non-null     int64
 9   Test_9      56 non-null     int64
```

```
 10  Test_10     56 non-null     int64
 11  Test_11     56 non-null     int64
 12  Test_12     56 non-null     int64
dtypes: int64(13)
memory usage: 5.8 KB
```

53	22053	92	100	100	100	100	100	92	87	94	100	94	98

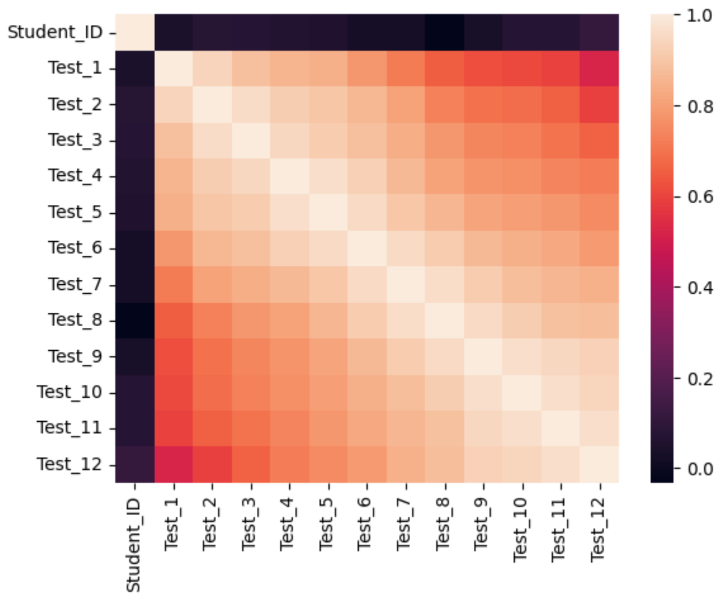
```
sns.pairplot(df3)
```

<seaborn.axisgrid.PairGrid at 0x7bbcb318f580>



```
sns.heatmap(df3.corr())
```

<Axes: >



```
y=df3['Test_2']
x=df3.drop(['Student_ID',"Test_2"],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
model=LinearRegression()

model.fit(x_train,y_train)

model.intercept_
```

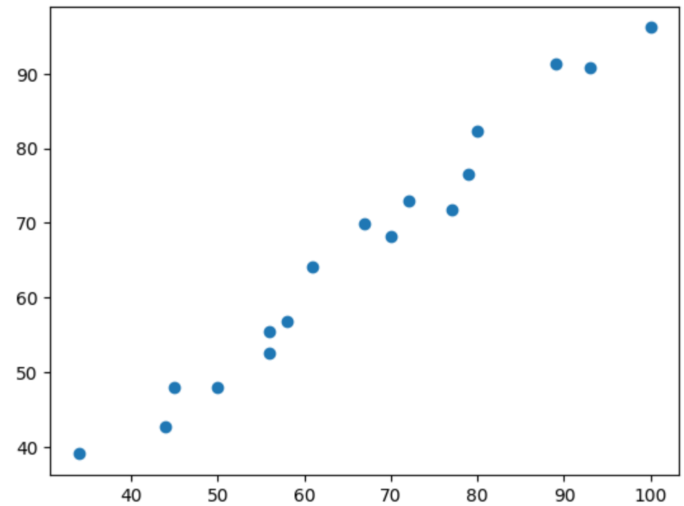
2.781188182681163

```
coeff=pd.DataFrame(model.coef_,x.columns,columns=["Coefficient"])

coeff
```

	Coefficient
Test_1	0.403898
Test_3	0.471293
Test_4	0.031125
Test_5	-0.022681
Test_6	0.028182
Test_7	0.266769
Test_8	-0.193957
Test_9	-0.028587
Test_10	0.062280
Test_11	0.117660
Test_12	-0.174370

```
prediction=model.predict(x_test)

plt.scatter(y_test,prediction)
```

```
<matplotlib.collections.PathCollection at 0x7bbca9b9d720>
```




```
model.score(x_test,y_test)
```

```
0.9744617636019209
```

```
rr=Ridge(alpha=10)

rr.fit(x_train,y_train)

la=Lasso(alpha=10)

la.fit(x_train,y_train)

print(rr.score(x_test,y_test))

la.score(x_test,y_test)
```

```
0.9743772738013958
0.9584347988676005
```

```
en=ElasticNet()
en.fit(x_train,y_train)
print(en.coef_)
print(en.intercept_)
print(en.predict(x_test))
print(en.score(x_test,y_test))
```

```
[ 0.40581572  0.4816498   0.01445122  0.          0.03200866  0.20893728
 -0.16127177 -0.          0.03691159  0.08789689 -0.14693703]
2.8692407771719957
[56.90726619 82.0548     38.83270048 75.83560142 63.83588307 72.32625095
 47.7967069  55.52659308 91.52116233 68.41633544 48.18217069 95.6279422
 70.74037726 72.95330083 90.52155834 42.63902831 52.44828504]
0.9729222267237271
```

```
print("MAE",metrics.mean_absolute_error(y_test,prediction))
print("MSE",metrics.mean_squared_error(y_test,prediction))
print("RMSE",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
MAE 2.5561088631343813
MSE 8.175593616185031
RMSE 2.8592994974617527
```

```
df4=pd.read_csv("/content/18_world-data-2023.csv")
df4
```

	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Calling Code	Capital/Major City	Co2-Emissions	...	Out of pocket health expenditure	Physicians per thousand	Popula
0	Afghanistan	60	AF	58.10%	652,230	323,000	32.49	93.0	Kabul	8,672	...	78.40%	0.28	38,04
1	Albania	105	AL	43.10%	28,748	9,000	11.78	355.0	Tirana	4,536	...	56.90%	1.20	2,85
2	Algeria	18	DZ	17.40%	2,381,741	317,000	24.28	213.0	Algiers	150,006	...	28.10%	1.72	43,05
3	Andorra	164	AD	40.00%	468	NaN	7.20	376.0	Andorra la Vella	469	...	36.40%	3.33	7
4	Angola	26	AO	47.50%	1,246,700	117,000	40.73	244.0	Luanda	34,693	...	33.40%	0.21	31,82
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
190	Venezuela	32	VE	24.50%	912,050	343,000	17.88	58.0	Caracas	164,175	...	45.80%	1.92	28,51
191	Vietnam	314	VN	39.30%	331,210	522,000	16.75	84.0	Hanoi	192,668	...	43.50%	0.82	96,46
192	Yemen	56	YE	44.60%	527,968	40,000	30.45	967.0	Sanaa	10,609	...	81.00%	0.31	29,16
193	Zambia	25	ZM	32.10%	752,618	16,000	36.19	260.0	Lusaka	5,141	...	27.50%	1.19	17,86
194	Zimbabwe	38	ZW	41.90%	390,757	51,000	30.68	263.0	Harare	10,983	...	25.80%	0.21	14,64

195 rows × 35 columns

```
df4.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
```

```
df4=df4.dropna()
df4
```

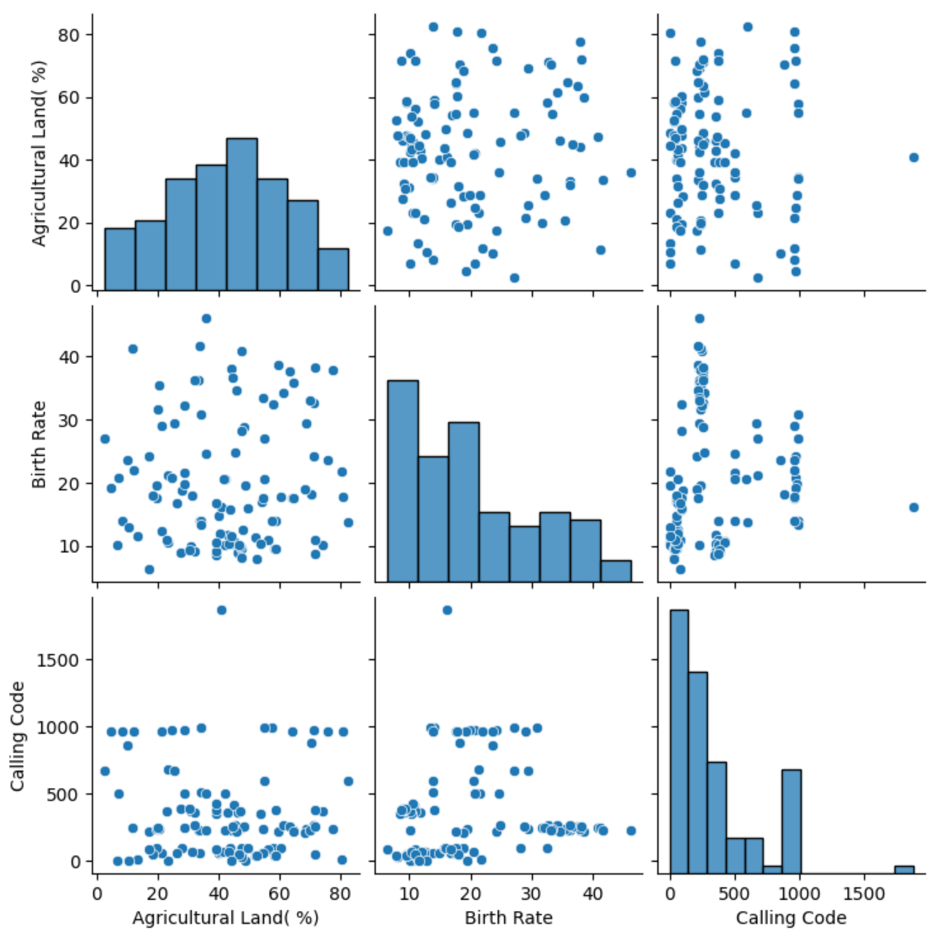
	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Calling Code	Capital/Major City	Co2-Emissions	...	Out of pocket health expenditure	Physicians per thousand	Popu
0	Afghanistan	60	AF	58.10%	652,230	323,000	32.49	93.0	Kabul	8,672	...	78.40%	0.28	38,0
1	Albania	105	AL	43.10%	28,748	9,000	11.78	355.0	Tirana	4,536	...	56.90%	1.20	2,8
2	Algeria	18	DZ	17.40%	2,381,741	317,000	24.28	213.0	Algiers	150,006	...	28.10%	1.72	43,0
4	Angola	26	AO	47.50%	1,246,700	117,000	40.73	244.0	Luanda	34,693	...	33.40%	0.21	31,8
6	Argentina	17	AR	54.30%	2,780,400	105,000	17.02	54.0	Buenos Aires	201,348	...	17.60%	3.96	44,9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
185	United Kingdom	281	GB	71.70%	243,610	148,000	11.00	44.0	London	379,025	...	14.80%	2.81	66,8

```
df4=df4.drop(["Country","Abbreviation","Capital/Major City","Currency-Code","Largest city","Official language","Minimum wage","Gasoline Price"],axis=1)
df4["Agricultural Land( %)"]=df4["Agricultural Land( %)"].replace("%","",regex=True).astype(float)
df4["CPI Change (%)"]=df4["CPI Change (%)"].replace("%","",regex=True).astype(float)
df4["Forested Area (%)"]=df4["Forested Area (%)"].replace("%","",regex=True).astype(float)
df4["Gross primary education enrollment (%)"]=df4["Gross primary education enrollment (%)"].replace("%","",regex=True).astype(float)
df4["Gross tertiary education enrollment (%)"]=df4["Gross tertiary education enrollment (%)"].replace("%","",regex=True).astype(float)
df4["Out of pocket health expenditure"]=df4["Out of pocket health expenditure"].replace("%","",regex=True).astype(float)
df4["Population: Labor force participation (%)"]=df4["Population: Labor force participation (%)"].replace("%","",regex=True).astype(float)
df4["Tax revenue (%)"]=df4["Tax revenue (%)"].replace("%","",regex=True).astype(float)
df4["Total tax rate"]=df4["Total tax rate"].replace("%","",regex=True).astype(float)
df4["Unemployment rate"]=df4["Unemployment rate"].replace("%","",regex=True).astype(float)
```

```
df4=df4.drop(["GDP"],axis=1)
```

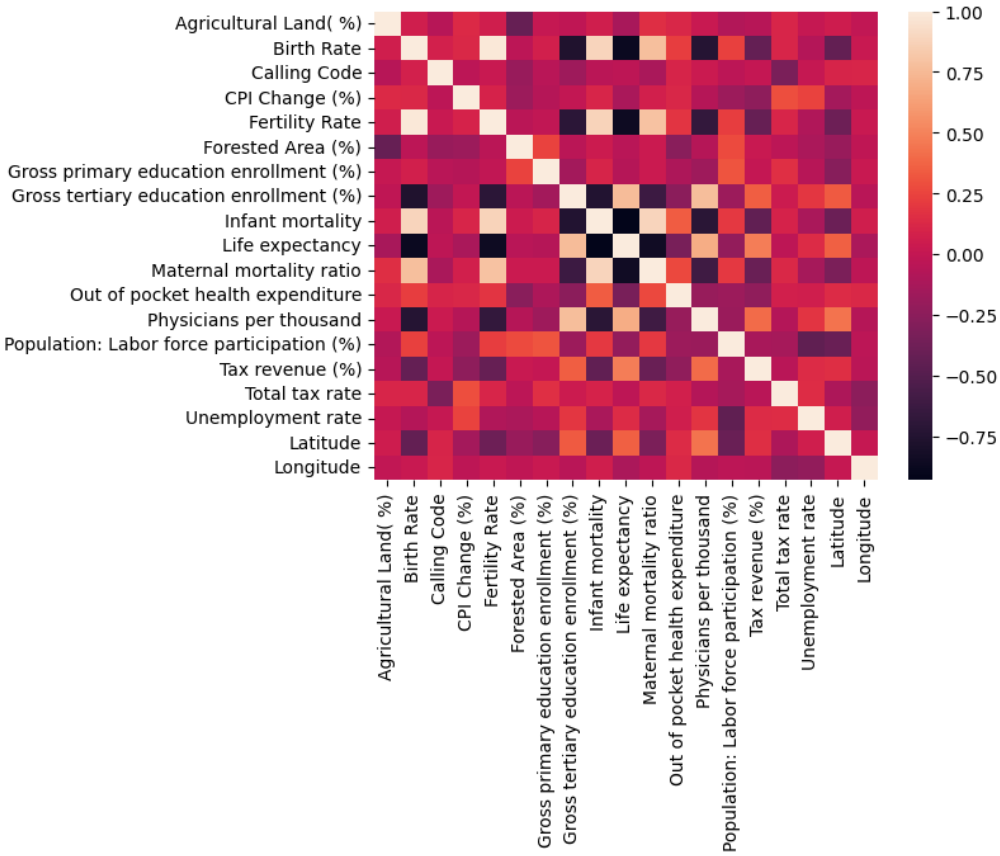
```
sns.pairplot(df4.iloc[:,:8])
```

```
<seaborn.axisgrid.PairGrid at 0x7bbca9c50670>
```




```
sns.heatmap(df4.corr())
```

```
<ipython-input-48-16ec28ac65e5>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to Fa
  sns.heatmap(df4.corr())
<Axes: >
```




```
df4=df4.replace(",","",regex=True)
df4=df4.astype(float)
```

```
y=df4['Fertility Rate']

x=df4.drop(['Fertility Rate'],axis=1)

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
model=LinearRegression()

model.fit(x_train,y_train)

model.intercept_
```

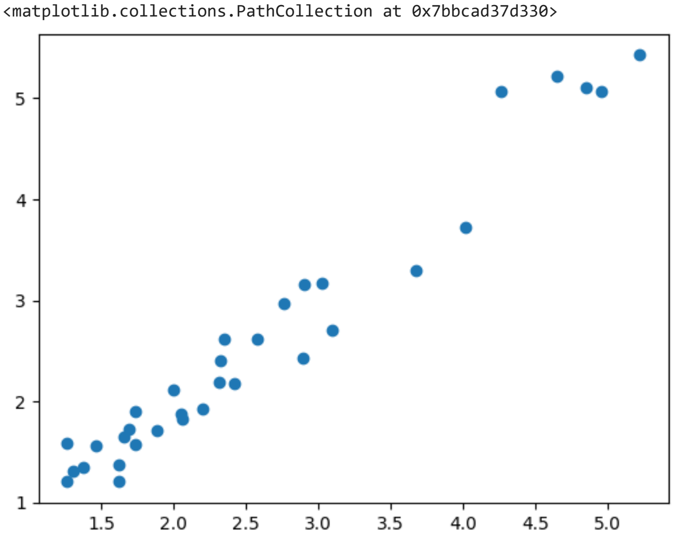
```
2.3117333487898146
```

```
coeff=pd.DataFrame(model.coef_,x.columns,columns=["Coefficient"])

coeff
```

	Coefficient
Density\n(P/Km2)	7.923212e-05
Agricultural Land(%)	-2.085825e-04
Land Area(Km2)	-6.687924e-09
Armed Forces size	-1.589795e-07
Birth Rate	1.279550e-01
Calling Code	-1.825059e-04
Co2-Emissions	5.772030e-08
CPI	2.872235e-04
CPI Change (%)	-6.395494e-03
Forested Area (%)	1.207074e-04
Gross primary education enrollment (%)	-7.833269e-03
Gross tertiary education enrollment (%)	3.047277e-03
Infant mortality	2.992270e-03
Life expectancy	-1.676885e-02
Maternal mortality ratio	5.786498e-04
Out of pocket health expenditure	-6.060383e-03
Physicians per thousand	8.232721e-02

```
prediction=model.predict(x_test)

plt.scatter(y_test,prediction)
```

```
<matplotlib.collections.PathCollection at 0x7bbcad37d330>
```




```
model.score(x_test,y_test)
```

```
0.9405957515632362
```

```
rr=Ridge(alpha=10)

rr.fit(x_train,y_train)

la=Lasso(alpha=10)

la.fit(x_train,y_train)

print(rr.score(x_test,y_test))

la.score(x_test,y_test)
```

```
0.9382009713694189
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: Ill-conditioned matrix (rcond=6.59549e-18): result may not be accura
```