```
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
```

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

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Specie
0	1	5.1	3.5	1.4	0.2	Iri setos
1	2	4.9	3.0	1.4	0.2	Iri setos
2	3	4.7	3.2	1.3	0.2	Iri setos
3	4	4.6	3.1	1.5	0.2	Iri setos
4	5	5.0	3.6	1.4	0.2	Iri setos
...	...	...	...	...	...	
145	146	6.7	3.0	5.2	2.3	Iri virginic

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

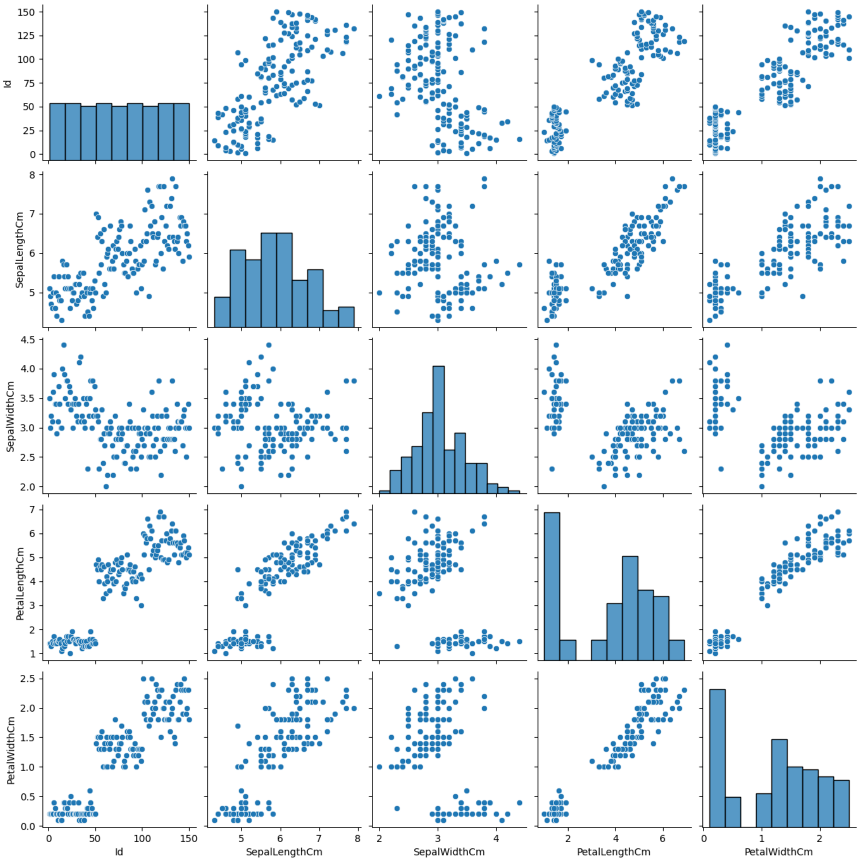
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	1	5.1	3.5	1.4	0.2
1	2	4.9	3.0	1.4	0.2
2	3	4.7	3.2	1.3	0.2
3	4	4.6	3.1	1.5	0.2
4	5	5.0	3.6	1.4	0.2
...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3
146	147	6.3	2.5	5.0	1.9
147	148	6.5	3.0	5.2	2.0
148	149	6.2	3.4	5.4	2.3
149	150	5.9	3.0	5.1	1.8

```
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
```

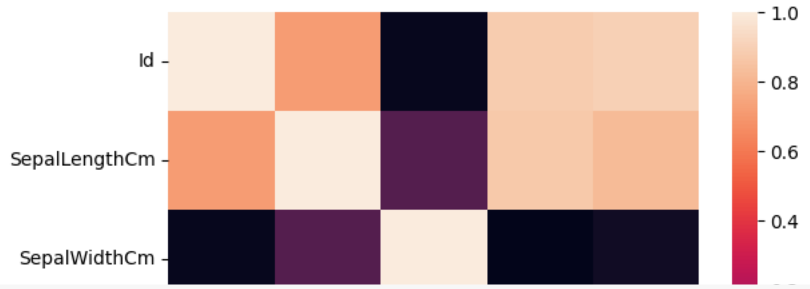
```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7a3a9fbbf430>
```




```
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_
```

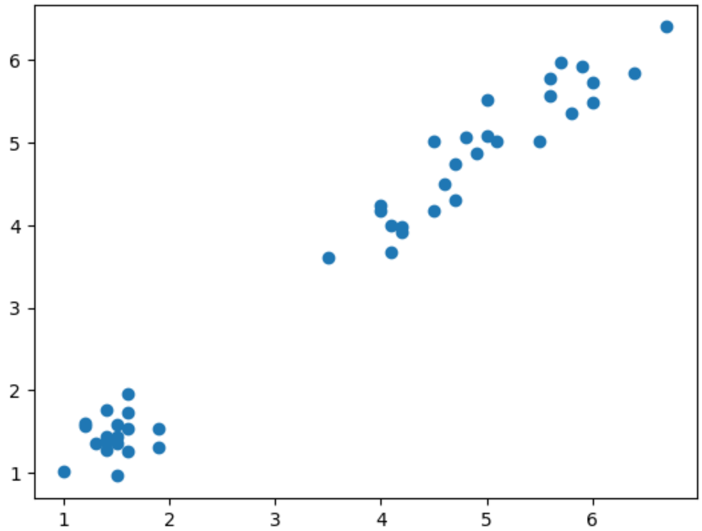
```
0.05270943475182621
```

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

	Coefficient
Id	0.001202
SepalLengthCm	0.704042
SepalWidthCm	-0.714965
PetalWidthCm	1.391341

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

```
<matplotlib.collections.PathCollection at 0x7a3a9efbd3c0>
```




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

```
0.9732057688319714
```

```
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)
```

```
0.9679230025891463
0.8006190313275552
```

```
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

```
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

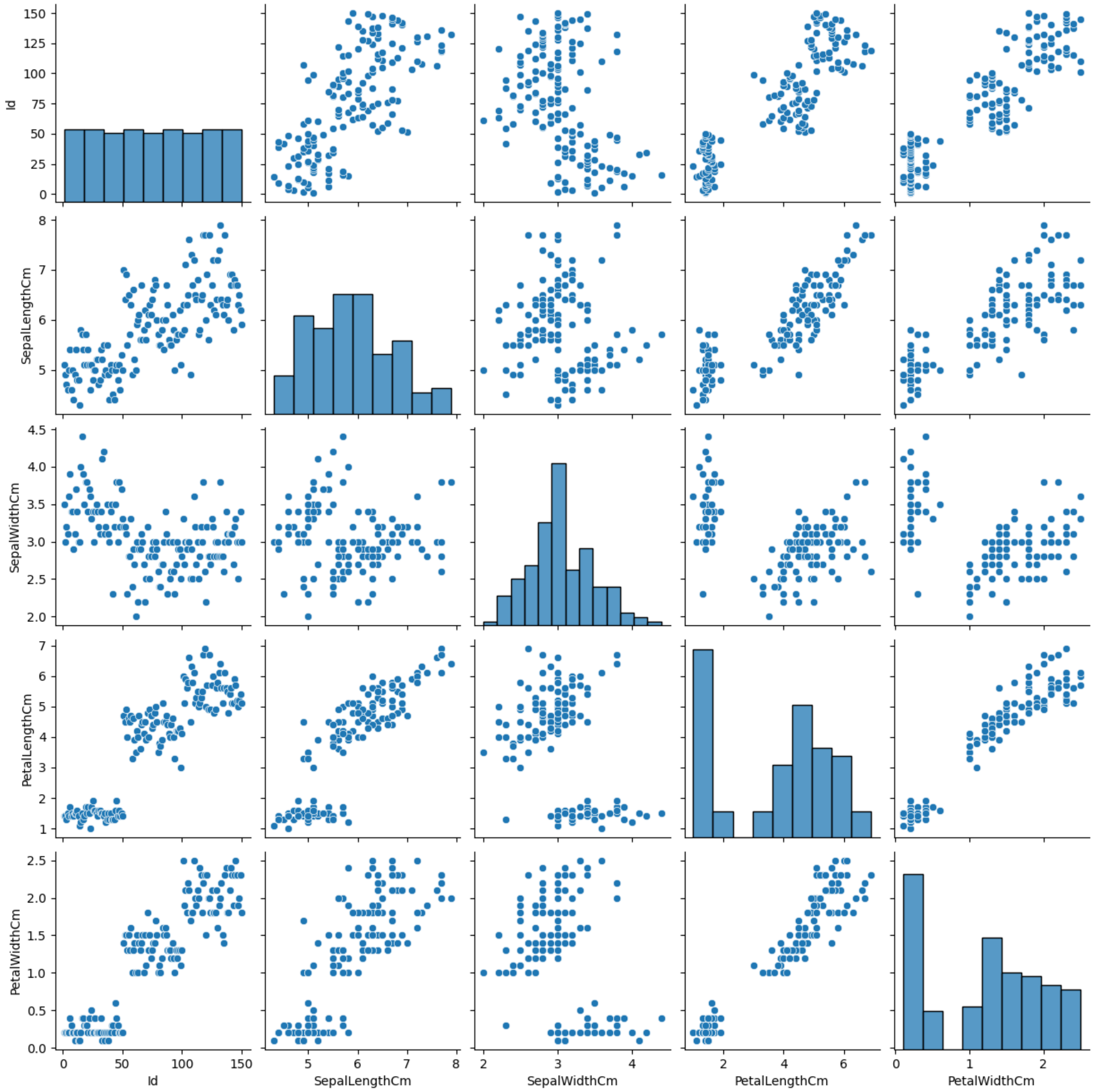
374 rows × 8 columns

```
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 0x7a3a9ede7fd0>
```




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

```
<ipython-input-104-3ed1a1a51dc0>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future v
  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)
```

−0.25

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

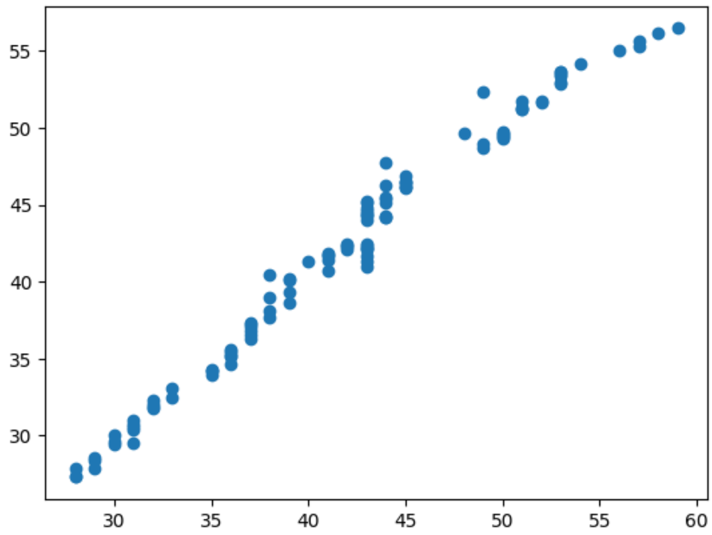
```
9.70952758771648
```

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

	Coefficient
Person ID	0.077839
Sleep Duration	0.271183
Quality of Sleep	0.716281
Physical Activity Level	-0.004037
Stress Level	0.081725
Heart Rate	0.141143
Daily Steps	0.000094

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

```
<matplotlib.collections.PathCollection at 0x7a3a9e2b8a30>
```




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

```
0.9797045985318592
```

```
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)
```

```
0.9799564320958067
0.9769271400204332
```

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

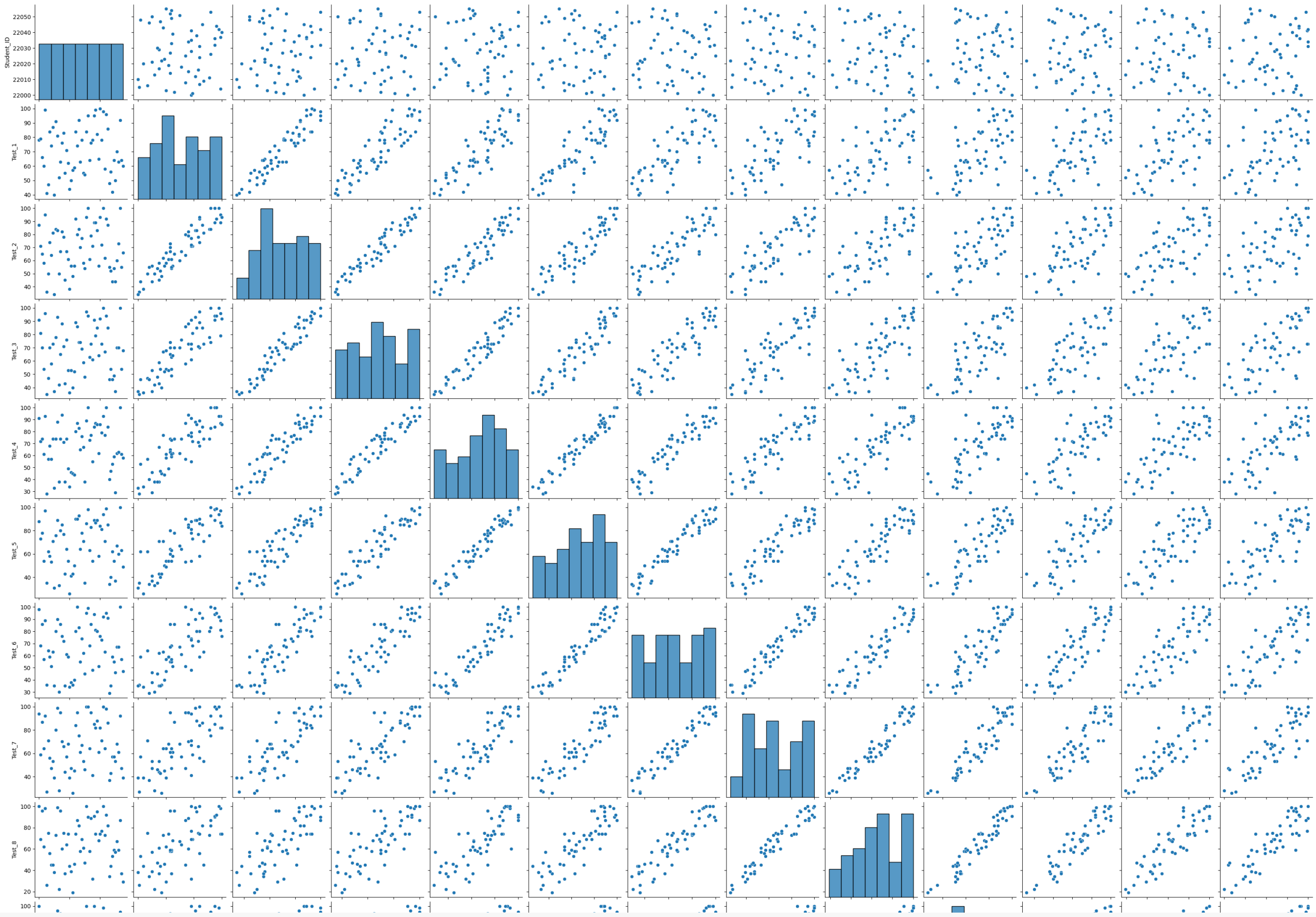
	Student_ID	Test_1	Test_2	Test_3	Test_4	Test_5	Test_6	Test_7	Test_8	Test_9	Test_10	Test_11	Test_12
0	22000	78	87	91	91	88	98	94	100	100	100	100	93
1	22001	79	71	81	72	73	68	59	69	59	60	61	67
2	22002	66	65	70	74	78	86	87	96	88	82	90	86
3	22003	60	58	54	61	54	57	64	62	72	63	72	76
4	22004	99	95	96	93	97	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

```
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
```

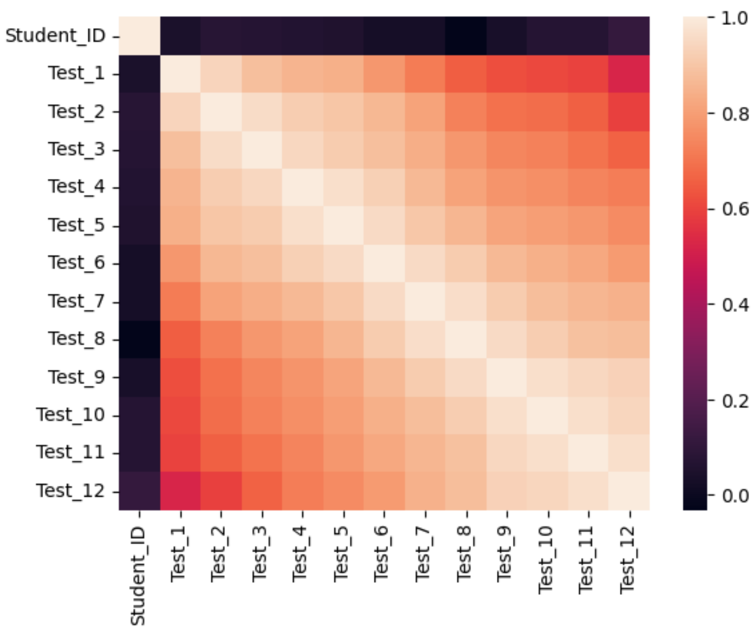
```
sns.pairplot(df3)
```

```
<seaborn.axisgrid.PairGrid at 0x7a3a9e0dea70>
```



```
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_
```

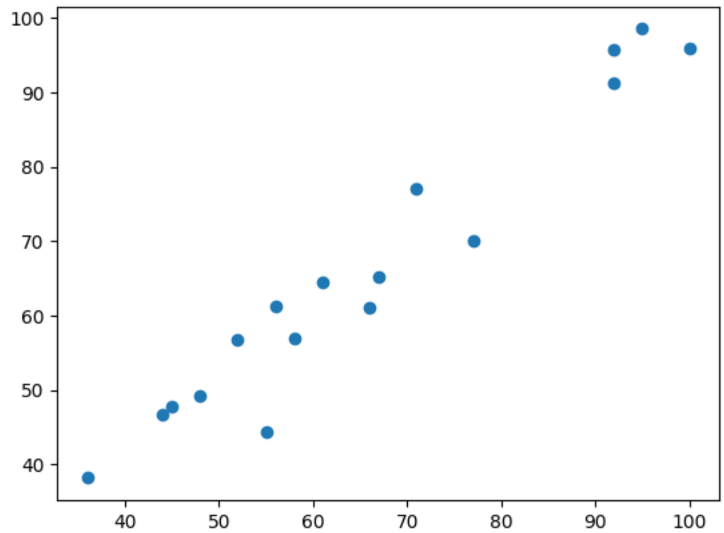
```
1.82330302617234
```

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

coeff
```

	Coefficient
Test_1	0.407266
Test_3	0.470980
Test_4	0.022131
Test_5	0.045375
Test_6	-0.079965
Test_7	0.252799
Test_8	-0.003530
Test_9	-0.183875
Test_10	-0.035321
Test_11	0.282963
Test_12	-0.202148

```
prediction=model.predict(x_test)

plt.scatter(y_test,prediction)
```

```
<matplotlib.collections.PathCollection at 0x7a3a94a32860>
```




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

```
0.9426166099503436
```

```
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)
```

```
0.9432522722880003
0.9420727436814061
```

```
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	...	exp
0	Afghanistan	60	AF	58.10%	652,230	323,000	32.49	93.0	Kabul	8,672	...	
1	Albania	105	AL	43.10%	28,748	9,000	11.78	355.0	Tirana	4,536	...	
2	Algeria	18	DZ	17.40%	2,381,741	317,000	24.28	213.0	Algiers	150,006	...	
3	Andorra	164	AD	40.00%	468	NaN	7.20	376.0	Andorra la Vella	469	...	
4	Angola	26	AO	47.50%	1,246,700	117,000	40.73	244.0	Luanda	34,693	...	
...	...	...	...	...	...	...	...	...	...	...	...	
190	Venezuela	32	VE	24.50%	912,050	343,000	17.88	58.0	Caracas	164,175	...	
191	Vietnam	314	VN	39.30%	331,210	522,000	16.75	84.0	Hanoi	192,668	...	
192	Yemen	56	YE	44.60%	527,968	40,000	30.45	967.0	Sanaa	10,609	...	
193	Zambia	25	ZM	32.10%	752,618	16,000	36.19	260.0	Lusaka	5,141	...	
194	Zimbabwe	38	ZW	41.90%	390,757	51,000	30.68	263.0	Harare	10,983	...	

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	...
0	Afghanistan	60	AF	58.10%	652,230	323,000	32.49	93.0	Kabul	8,672	...
1	Albania	105	AL	43.10%	28,748	9,000	11.78	355.0	Tirana	4,536	...
2	Algeria	18	DZ	17.40%	2,381,741	317,000	24.28	213.0	Algiers	150,006	...
4	Angola	26	AO	47.50%	1,246,700	117,000	40.73	244.0	Luanda	34,693	...
6	Argentina	17	AR	54.30%	2,780,400	105,000	17.02	54.0	Buenos Aires	201,348	...
...	...	...	...	...	...	...	...	...	...	...	...
185	United Kingdom	281	GB	71.70%	243,610	148,000	11.00	44.0	London	379,025	...
186	United States	36	US	44.40%	9,833,517	1,359,000	11.60	1.0	Washington, D.C.	5,006,302	...
187	Uruguay	20	UY	82.60%	176,215	22,000	13.86	598.0	Montevideo	6,766	...
191	Vietnam	314	VN	39.30%	331,210	522,000	16.75	84.0	Hanoi	192,668	...
193	Zambia	25	ZM	32.10%	752,618	16,000	36.19	260.0	Lusaka	5,141	...

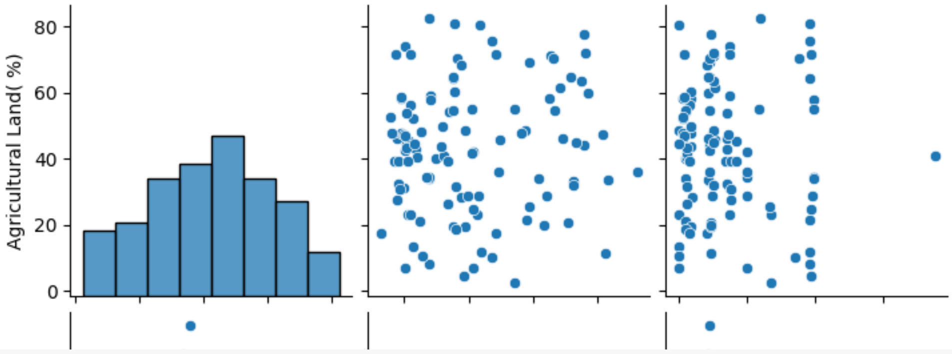
110 rows × 35 columns

```
df4=df4.drop(["Country","Abbreviation","Capital/Major City","Currency-Code","Largest city","Official language","Minimum wage","Gasoline F
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)
```

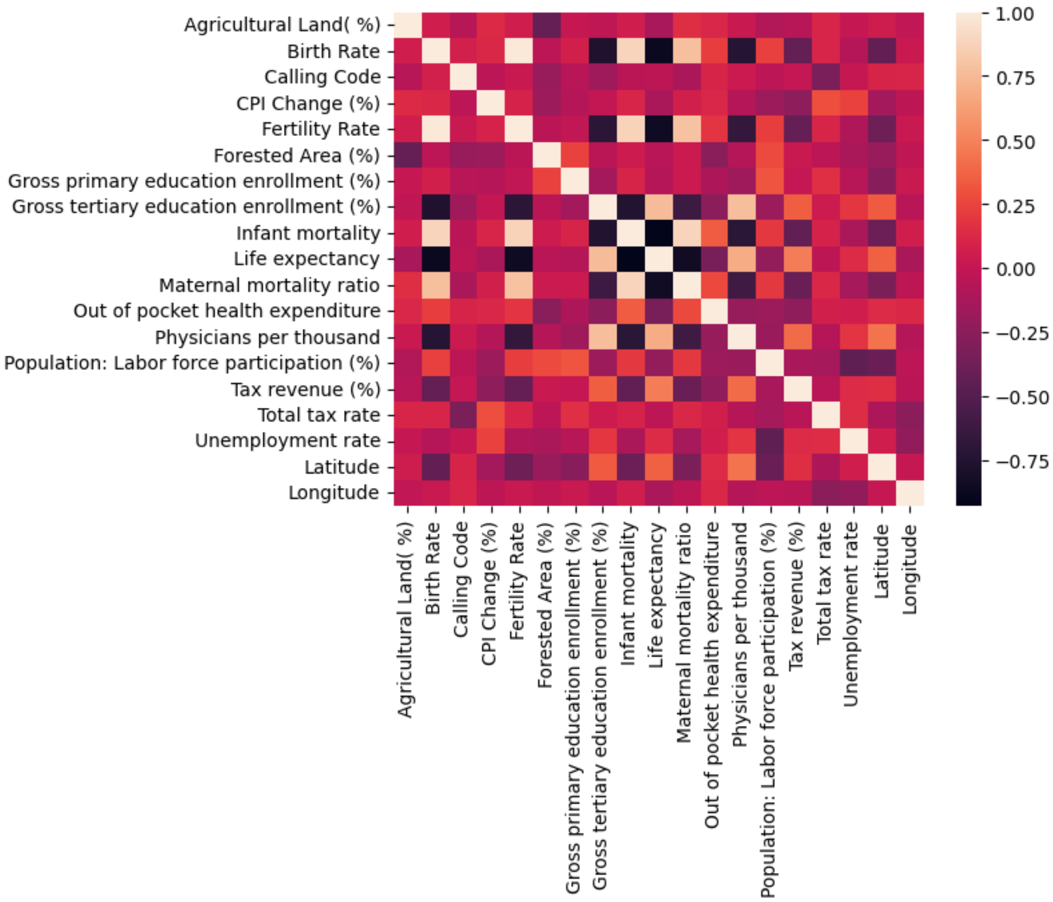
```
sns.pairplot(df4.iloc[:,:8])
```

```
<seaborn.axisgrid.PairGrid at 0x7a3a948f3a60>
```




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

```
<ipython-input-127-16ec28ac65e5>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future v
  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_
```

```
1.2208407111776067
```

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

coeff
```

	Coefficient
Density\n(P/Km2)	1.166185e-05
Agricultural Land(%)	-1.259609e-03
Land Area(Km2)	-4.361556e-09
Armed Forces size	-9.837214e-08
Birth Rate	1.376233e-01
Calling Code	-1.867638e-04
Co2-Emissions	-1.275433e-09
CPI	2.296431e-04
CPI Change (%)	-5.367021e-03
Forested Area (%)	-1.410413e-03
Gross primary education enrollment (%)	-6.752963e-03
Gross tertiary education enrollment (%)	1.980932e-03
Infant mortality	-3.247107e-03
Life expectancy	-9.387256e-03
Maternal mortality ratio	4.303126e-04
Out of pocket health expenditure	-3.783321e-03
Physicians per thousand	7.280566e-02
Population	2.246167e-10
Population: Labor force participation (%)	2.028631e-03
Tax revenue (%)	-2.334029e-03
Total tax rate	5.861105e-04
Unemployment rate	-6.325363e-03
Urban_population	1.683495e-10
Latitude	2.627209e-03
Longitude	4.304176e-04

```
prediction=model.predict(x_test)

plt.scatter(y_test,prediction)
```

```
<matplotlib.collections.PathCollection at 0x7a3a942b3070>
```


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

```
0.9780878147242772
```

```python
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)
```

```
0.9780750355129515
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: Ill-conditioned matrix (rcond=6.72644e-1
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did not conv
  model = cd_fast.enet_coordinate_descent(
0.7252163062614683
```