ECONOMIC DATA SET

In [8]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
```

In [10]:
```
df_index=pd.read_excel("economic_index.xlsx")
```

In [12]:
```
df_index.head()
```

Out[12]:

	Unnamed: 0	year	month	interest_rate	unemployment_rate	index_price
0	0.0	2017	12	2.75	5.3	1464
1	1.0	2017	11	2.50	5.3	1394
2	2.0	2017	10	2.50	5.3	1357
3	3.0	2017	9	2.50	5.3	1293
4	4.0	2017	8	2.50	5.4	1256

In [14]:
```
##drop unneccessary columns
df_index.drop(columns=["Unnamed: 0","year","month"], errors='ignore', inplace=True)
```

In [16]:
```
df_index.head()
```

Out[16]:

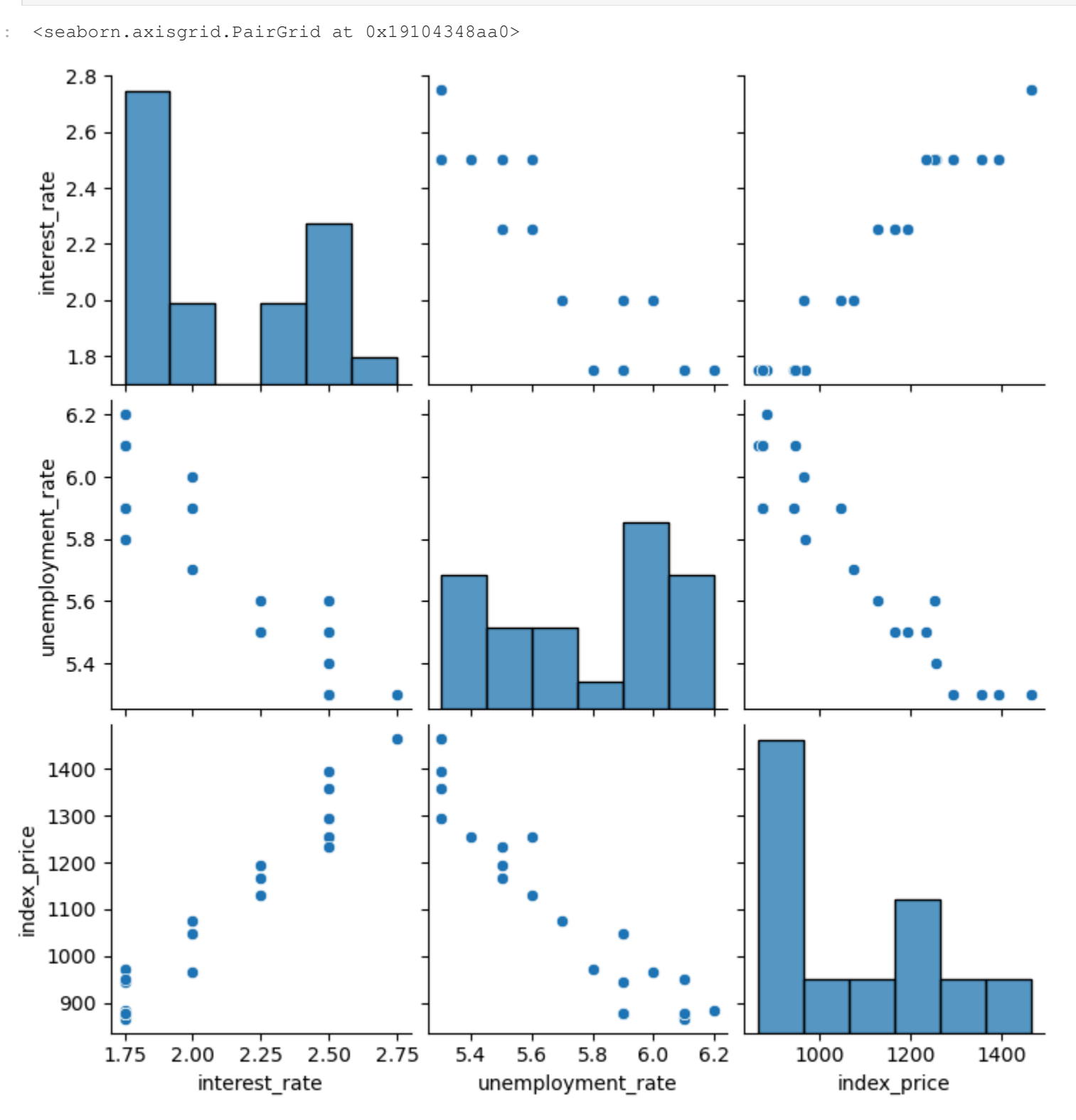
	interest_rate	unemployment_rate	index_price
0	2.75	5.3	1464
1	2.50	5.3	1394
2	2.50	5.3	1357
3	2.50	5.3	1293
4	2.50	5.4	1256

In [18]:
```
df_index.isnull().sum()
```

Out[18]:
```
interest_rate        0
unemployment_rate    0
index_price          0
dtype: int64
```

In [20]:
```
import seaborn as sns
sns.pairplot(df_index)
```

Out[20]: <seaborn.axisgrid.PairGrid at 0x19104348aa0>



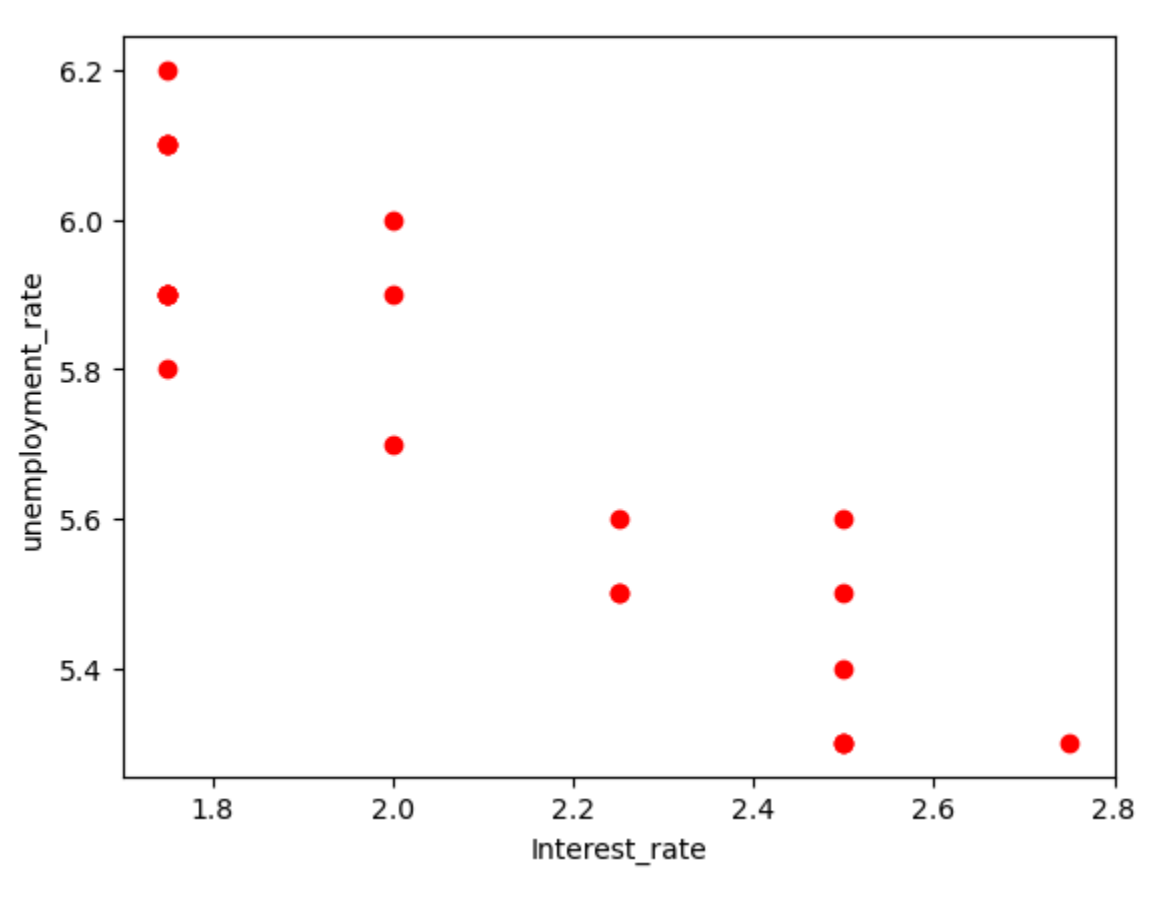
In [21]:
```
df_index.corr()
```

Out[21]:

	interest_rate	unemployment_rate	index_price
interest_rate	1.000000	-0.913920	0.974334
unemployment_rate	-0.913920	1.000000	-0.937018
index_price	0.974334	-0.937018	1.000000

In [22]:
```
plt.scatter(df_index['interest_rate'],df_index['unemployment_rate'],color='r')
plt.xlabel("Interest_rate")
plt.ylabel("unemployment_rate")
```

Out[22]: Text(0, 0.5, 'unemployment_rate')



In [23]:
```
x=df_index.iloc[:,:-1]
y=df_index.iloc[:,-1]
```

In [24]:
```
x.head()
```

Out[24]:

	interest_rate	unemployment_rate
0	2.75	5.3
1	2.50	5.3
2	2.50	5.3
3	2.50	5.3
4	2.50	5.4

In [25]:
```
y
```

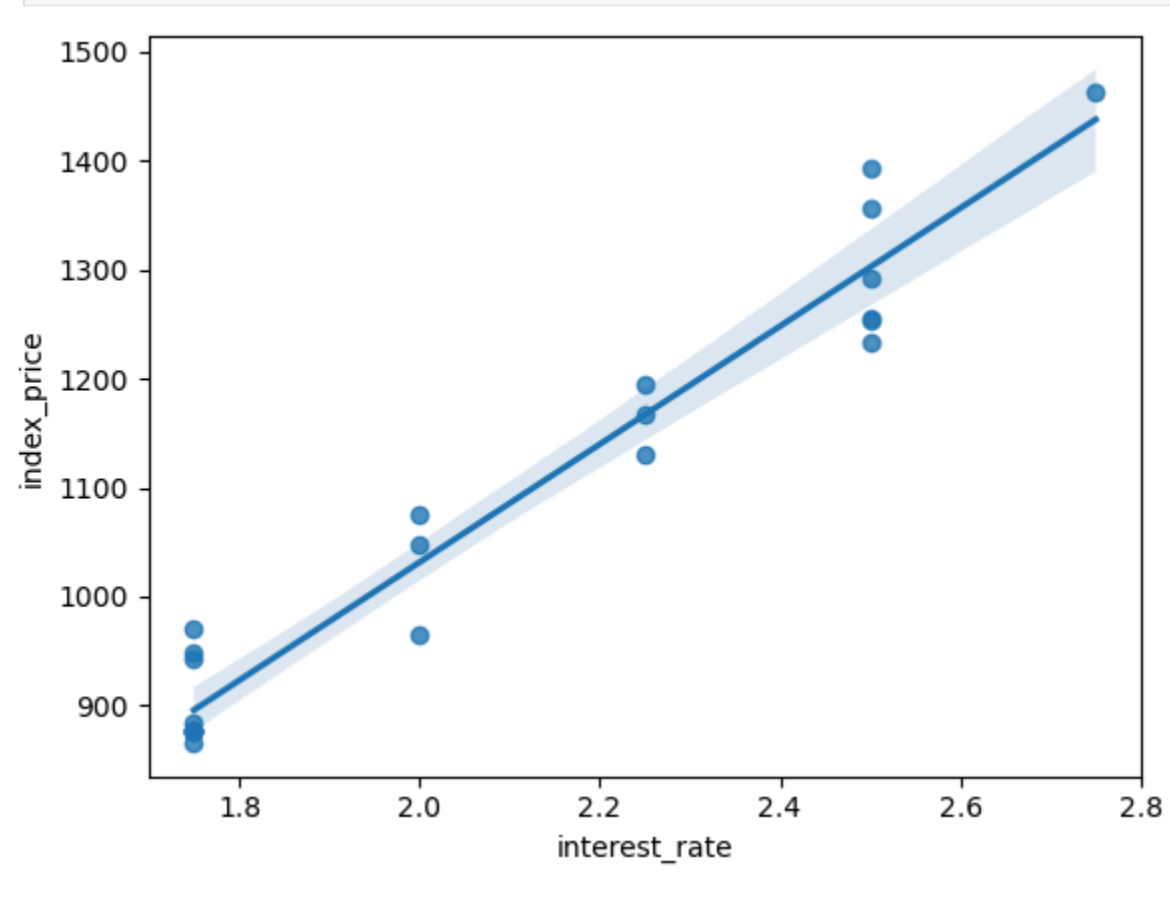
Out[25]:
```
0     1464
1     1394
2     1357
3     1293
4     1256
5     1254
6     1234
7     1195
8     1167
9     1130
10    1075
11    1047
12     965
13     943
14     971
15     949
16     884
17     866
18     876
19     876
20     876
21     876
22     876
23     876
Name: index_price, dtype: int64
```

In [26]:
```
from sklearn.model_selection import train_test_split
```

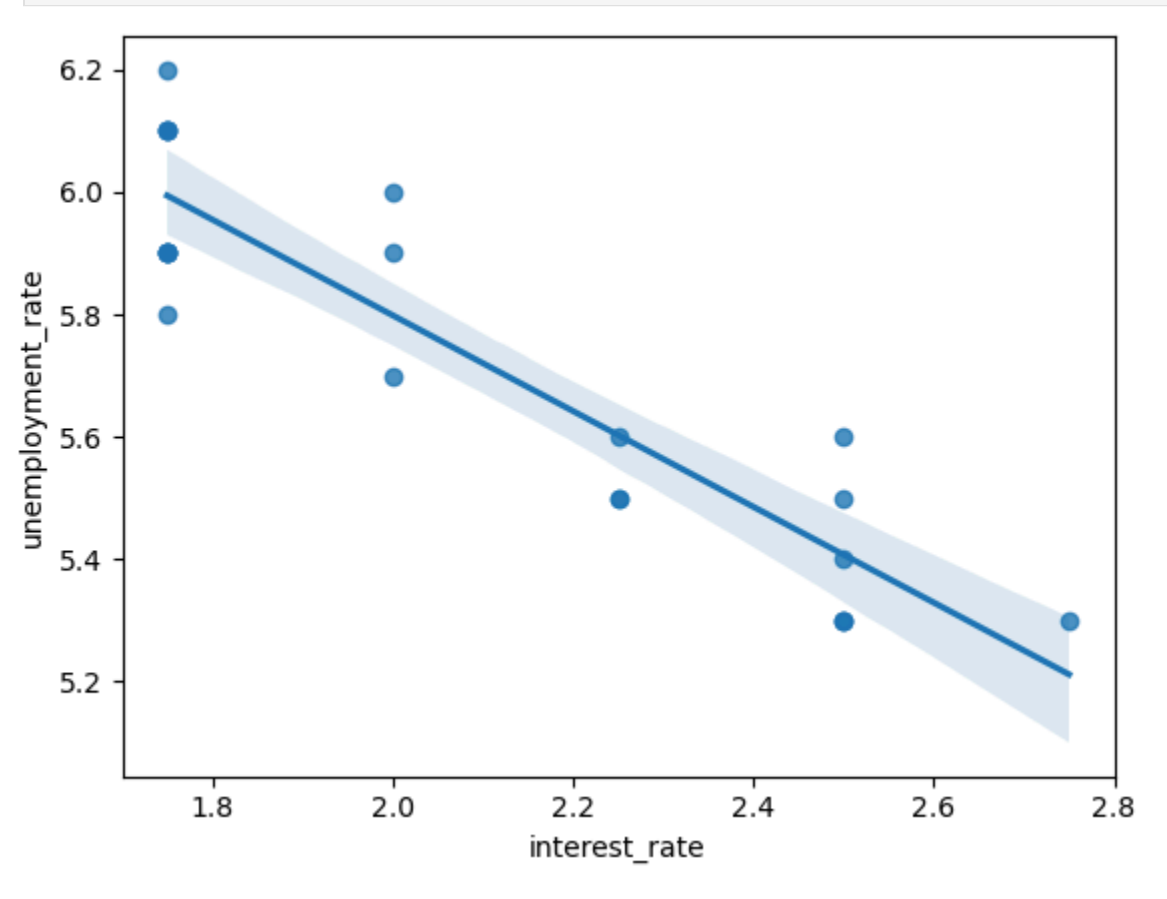
In [27]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=45)
```

In [28]:
```
import seaborn as sns
```

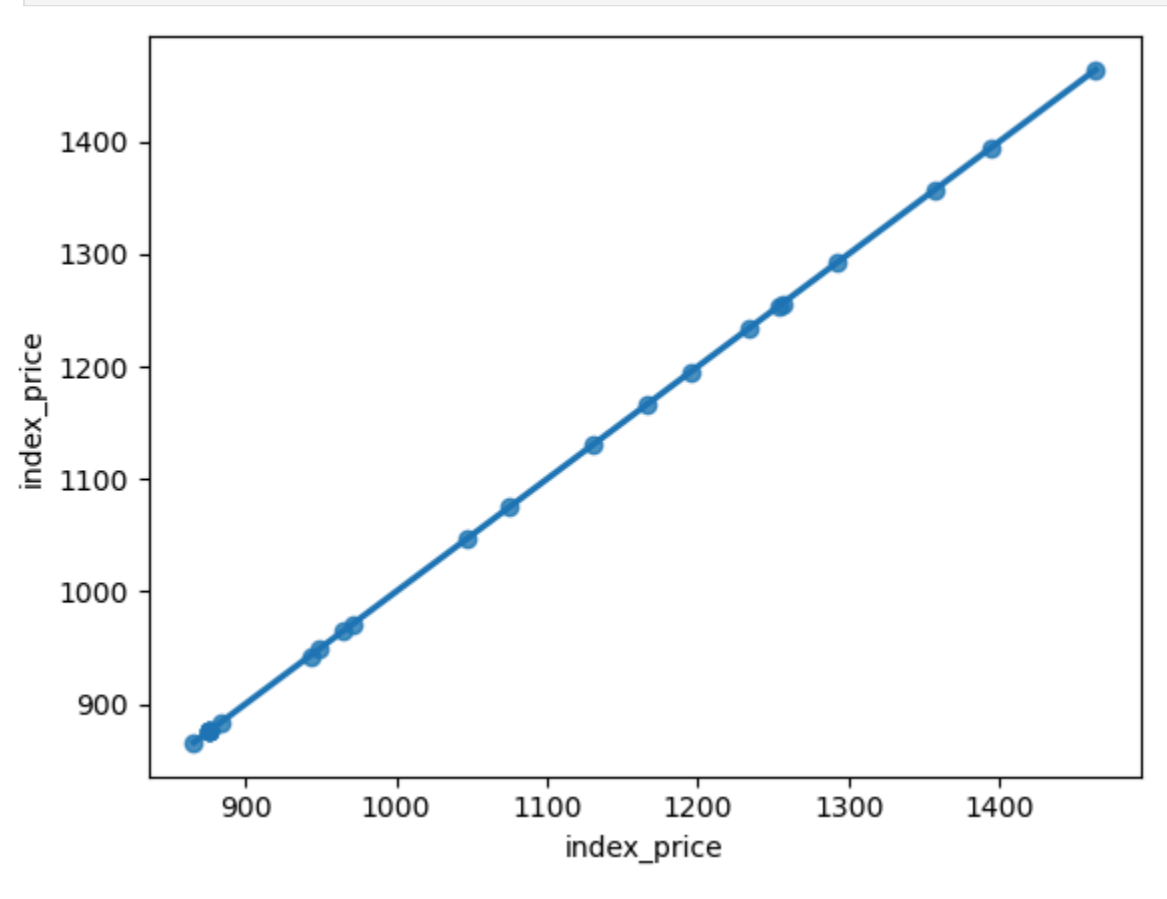
In [29]:
```
sns.regplot(x=df_index['interest_rate'],y=df_index['index_price'])
plt.show()
```



In [36]:
```
sns.regplot(x=df_index['interest_rate'],y=df_index['unemployment_rate'])
plt.show()
```



In [37]:
```
sns.regplot(x=df_index['index_price'],y=df_index['index_price'])
plt.show()
```



In [42]:
```
from sklearn.preprocessing import StandardScaler
```

In [46]:
```
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.fit_transform(x_test)
```

In [48]:
```
x_train
```

Out[48]:
```
array([[-1.21267813,  0.77468884],
       [-1.21267813,  1.45490344],
       [ 0.9701425 , -1.26595494],
       [-1.21267813,  1.45490344],
       [-0.48507125,  1.11479614],
       [ 0.24253563, -0.58574035],
       [ 0.24253563, -0.58574035],
       [ 0.9701425 , -0.58574035],
       [ 0.24253563, -0.24563305],
       [ 0.9701425 , -0.24563305],
       [-1.21267813,  0.43458155],
       [ 0.9701425 , -1.26595494],
       [-1.21267813,  1.45490344],
       [ 0.9701425 , -0.92584764],
       [-1.21267813,  0.77468884],
       [ 1.69774938, -1.26595494],
       [ 0.9701425 , -1.26595494],
       [-0.48507125,  0.77468884]])
```

In [50]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
```

In [52]:
```
regression = LinearRegression()
```

In [54]:
```
regression.fit(x_train,y_train)
```

Out[54]:

▾ LinearRegression

LinearRegression()

In [56]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.datasets import make_regression
X, y = make_regression(n_samples=100, n_features=2, noise=0.1)
```

In [58]:
```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [60]:
```
regression = LinearRegression()
```

In [62]:
```
validation_score = cross_val_score(regression, x_train, y_train, scoring='neg_mean_squared_error', cv=3)
```

In [64]:
```
validation_score
```

Out[64]: array([-0.01490603, -0.01366399, -0.00869088])

In [73]:
```
regression.fit(x_train, y_train)
y_pred = regression.predict(x_test)
print(y_pred)
```

```
[  58.39962381 -124.55987225   82.00619686 -108.97703659   56.35773024
   69.21490594   58.92850072   66.96758284    1.18834145  -92.29059094
  -56.60245735    6.88552724  106.98471526  106.72905768  124.21110479
   98.21192578   35.24625365  -96.55430246   34.38509682  -56.02021913]
```

In [75]:
```
y_pred = regression.predict(x_test)
print(y_pred)
```

```
[  58.39962381 -124.55987225   82.00619686 -108.97703659   56.35773024
   69.21490594   58.92850072   66.96758284    1.18834145  -92.29059094
  -56.60245735    6.88552724  106.98471526  106.72905768  124.21110479
   98.21192578   35.24625365  -96.55430246   34.38509682  -56.02021913]
```

In [77]:
```
import numpy as np
from sklearn.metrics import mean_absolute_error,mean_squared_error
mse=mean_squared_error(y_test,y_pred)
mae=mean_absolute_error(y_test,y_pred)
rmse=np.sqrt(mse)
print(mse)
print(mae)
print(rmse)
```

```
0.010805936091722545
0.08103085046303785
0.10395160456540603
```

In [79]:
```
from sklearn.metrics import r2_score
score = r2_score(y_test,y_pred)
print(score)
```

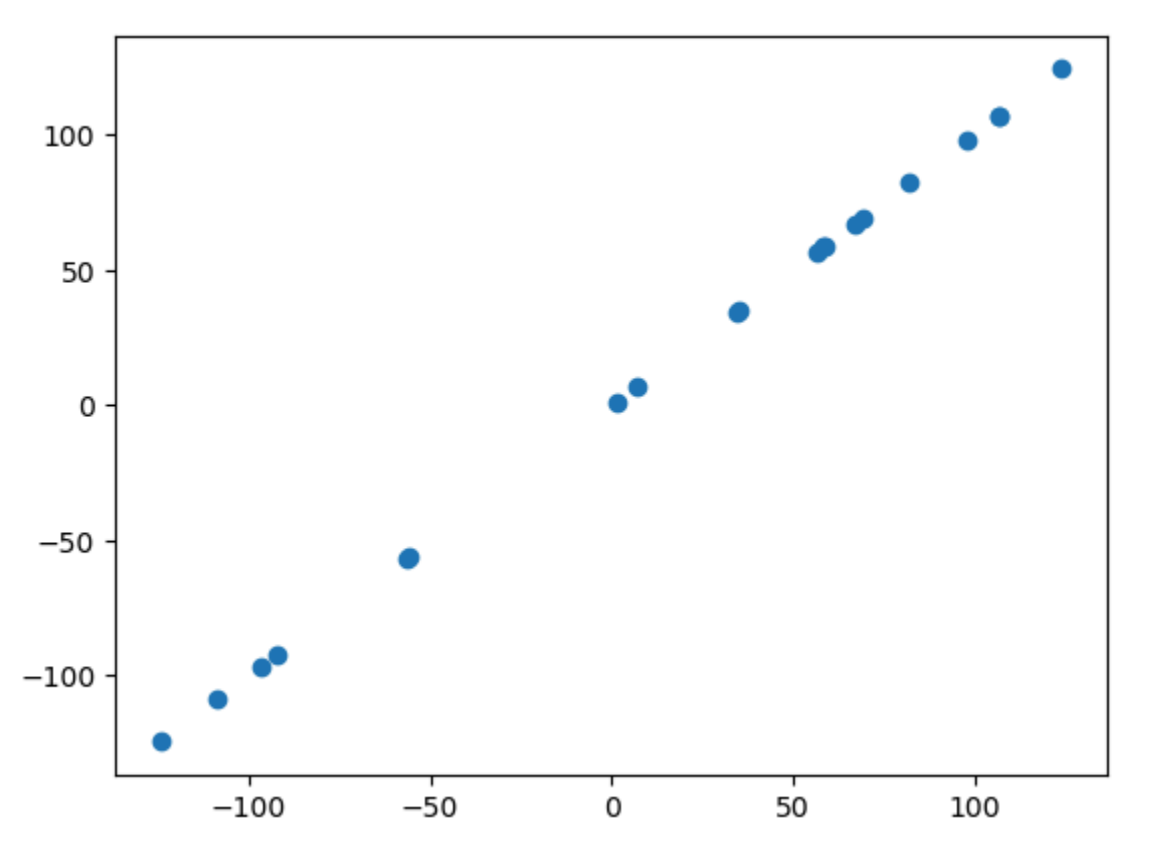
```
0.9999982151808697
```

In [81]:
```
print(1-(1-score)*(len(y_test)-1)/(len(y_test)-x_test.shape[1]-1))
```

```
0.9999980052021485
```

In [83]:
```
import matplotlib.pyplot as plt
plt.scatter(y_test,y_pred)
```

Out[83]: <matplotlib.collections.PathCollection at 0x19108537c50>

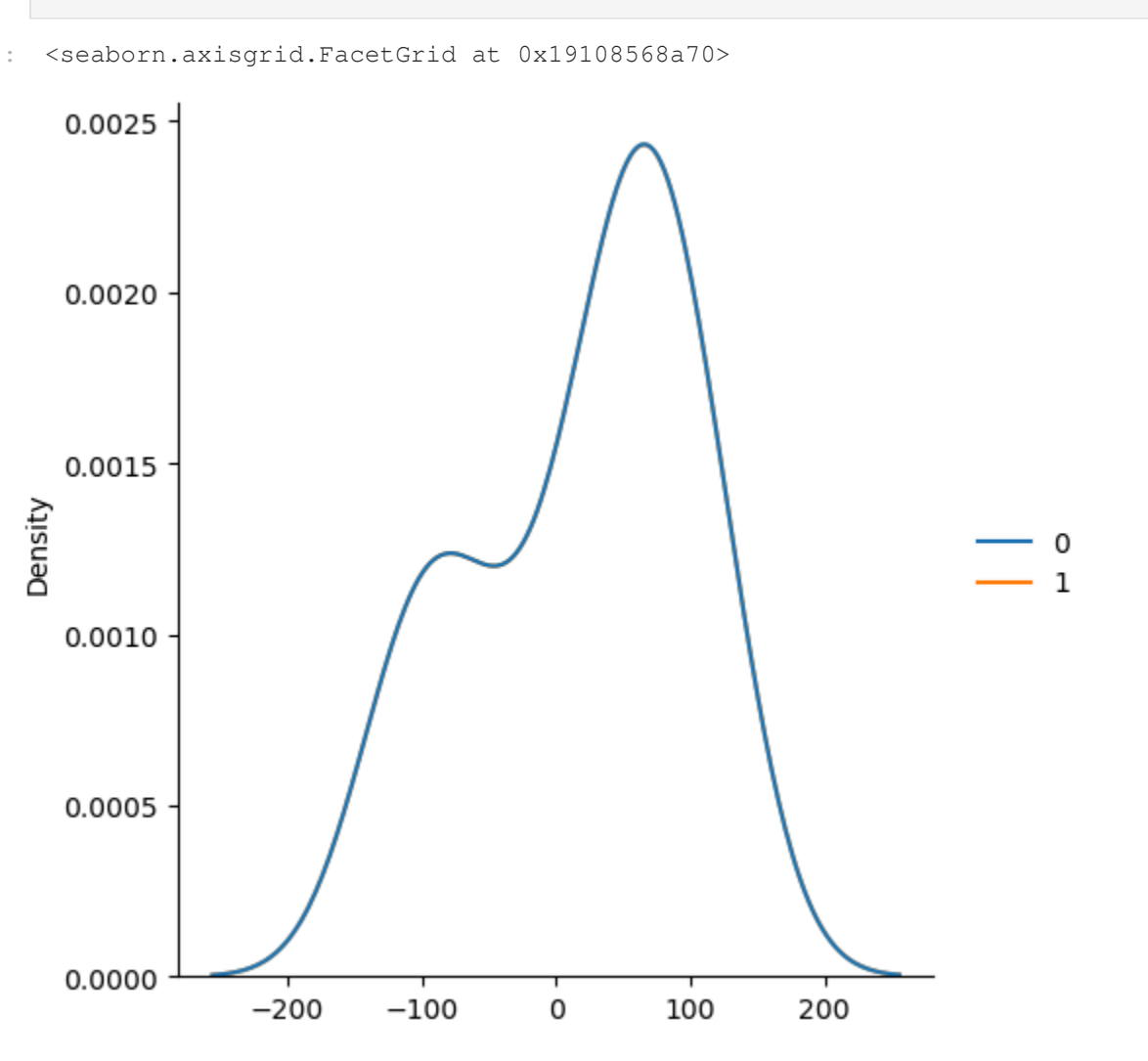


In [85]:
```
residuals=y_test,y_pred
print(residuals)
```

```
(array([  58.44931239, -124.56584963,   81.99892615, -108.9799827 ,
         56.44902012,   69.27654305,   59.00655704,   67.17926929,
          1.37399372,  -92.19783283,  -56.60279729,    6.74948589,
        106.93023863,  106.67536448,  124.22510866,   98.04445237,
         35.04982726,  -96.45770567,   34.42894205,  -55.94946186]), array([  58.39962381, -124.55987225,   82.00619686, -108.97703659,
         56.35773024,   69.21490594,   58.92850072,   66.96758284,
          1.18834145,  -92.29059094,  -56.60245735,    6.88552724,
        106.98471526,  106.72905768,  124.21110479,   98.21192578,
         35.24625365,  -96.55430246,   34.38509682,  -56.02021913]))
```

In [87]:
```
import seaborn as sns
sns.displot(residuals,kind='kde')
```

Out[87]: <seaborn.axisgrid.FacetGrid at 0x19108568a70>



In [89]:
```
import statsmodels.api as sm
model=sm.OLS(y_train,x_train).fit()
```

In [90]:
```
model.summary()
```

Out[90]:

OLS Regression Results

Dep. Variable:	y	R-squared (uncentered):	1.000
Model:	OLS	Adj. R-squared (uncentered):	1.000
Method:	Least Squares	F-statistic:	4.766e+07
Date:	Sun, 20 Oct 2024	Prob (F-statistic):	4.01e-238
Time:	15:28:39	Log-Likelihood:	63.019
No. Observations:	80	AIC:	-122.0
Df Residuals:	78	BIC:	-117.3
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
x1	62.3414	0.012	5018.787	0.000	62.317	62.366
x2	75.4742	0.012	6548.278	0.000	75.451	75.497

Omnibus:	0.092	Durbin-Watson:	1.925
Prob(Omnibus):	0.955	Jarque-Bera (JB):	0.262
Skew:	0.045	Prob(JB):	0.877
Kurtosis:	2.735	Cond. No.	1.36

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [91]:
```
print(regression.coef_)
```

```
[62.34266153 75.47254323]
```

In [ ]: