In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
indexprice_df = pd.read_csv('economic_index.csv')
```

In [3]:
```
indexprice_df.info()#there are no null values as all columns have 24 records
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24 entries, 0 to 23
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         24 non-null     int64  
 1   year               24 non-null     int64  
 2   month              24 non-null     int64  
 3   interest_rate      24 non-null     float64
 4   unemployment_rate  24 non-null     float64
 5   index_price        24 non-null     int64  
dtypes: float64(2), int64(4)
memory usage: 1.2 KB
```

In [4]:
```
indexprice_df.head()#we require interest_rate and unemployement_rate to predict the inde
#We can remove unwanted columns
```

Out[4]:

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

In [6]:
```
indexprice_df.drop(columns= ['Unnamed: 0', 'year', 'month'], axis= 1, inplace= True)
```

In [7]:
```
indexprice_df.head()
```

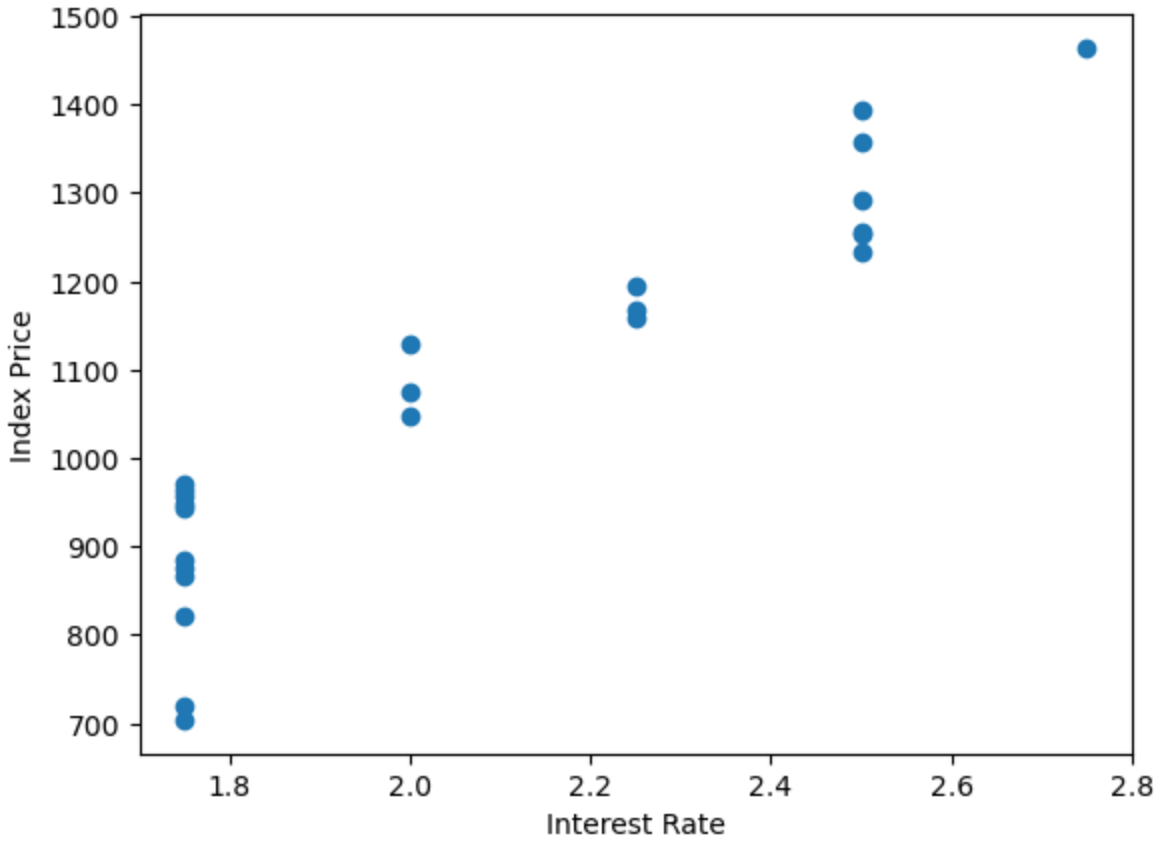
Out[7]:

	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 [8]:
```
import matplotlib.pyplot as plt
import seaborn as sn
%matplotlib inline
```

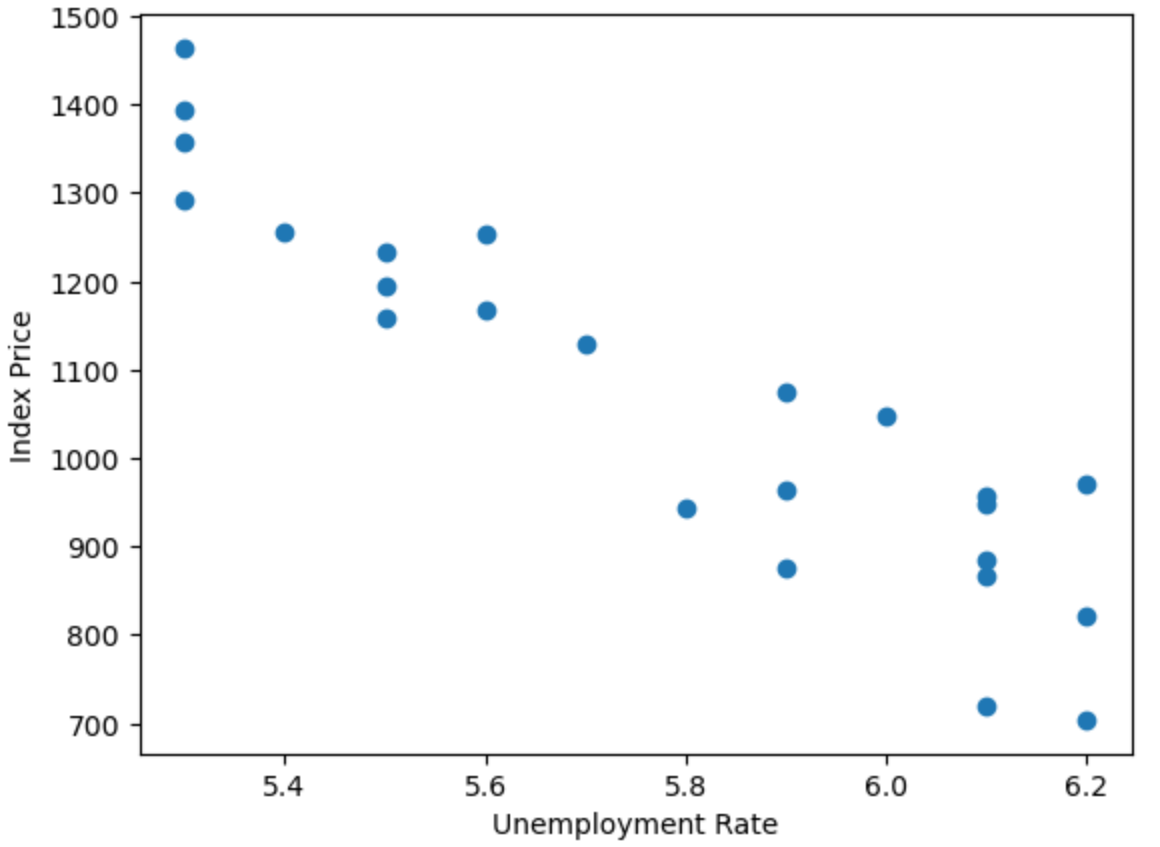
In [9]:
```
plt.scatter(indexprice_df.interest_rate, indexprice_df.index_price)
plt.xlabel('Interest Rate')
plt.ylabel('Index Price')
#looks like Interest Rate and Index Price are linearly related
```

Out[9]:
```
Text(0, 0.5, 'Index Price')
```



In [10]:
```
plt.scatter(indexprice_df.unemployment_rate, indexprice_df.index_price)
plt.xlabel('Unemployment Rate')
plt.ylabel('Index Price')
#looks like unemployment and index price are inversely related
```

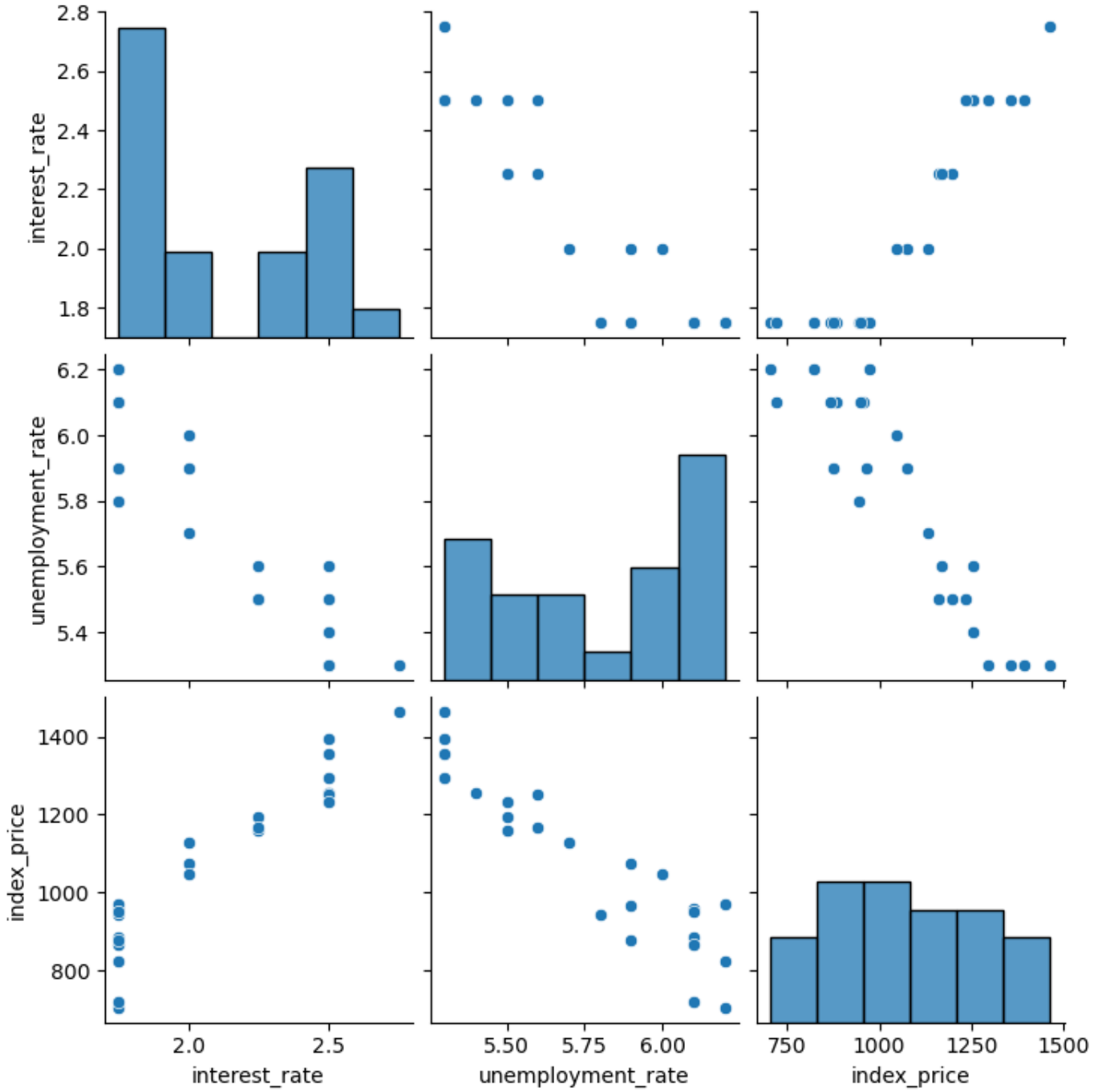
Out[10]:
```
Text(0, 0.5, 'Index Price')
```



In [11]:
```
sn.pairplot(indexprice_df)
#looks like interest rate and unemployment rate are also inversely related
```

Out[11]:

```
<seaborn.axisgrid.PairGrid at 0x7fb20340ed60>
```



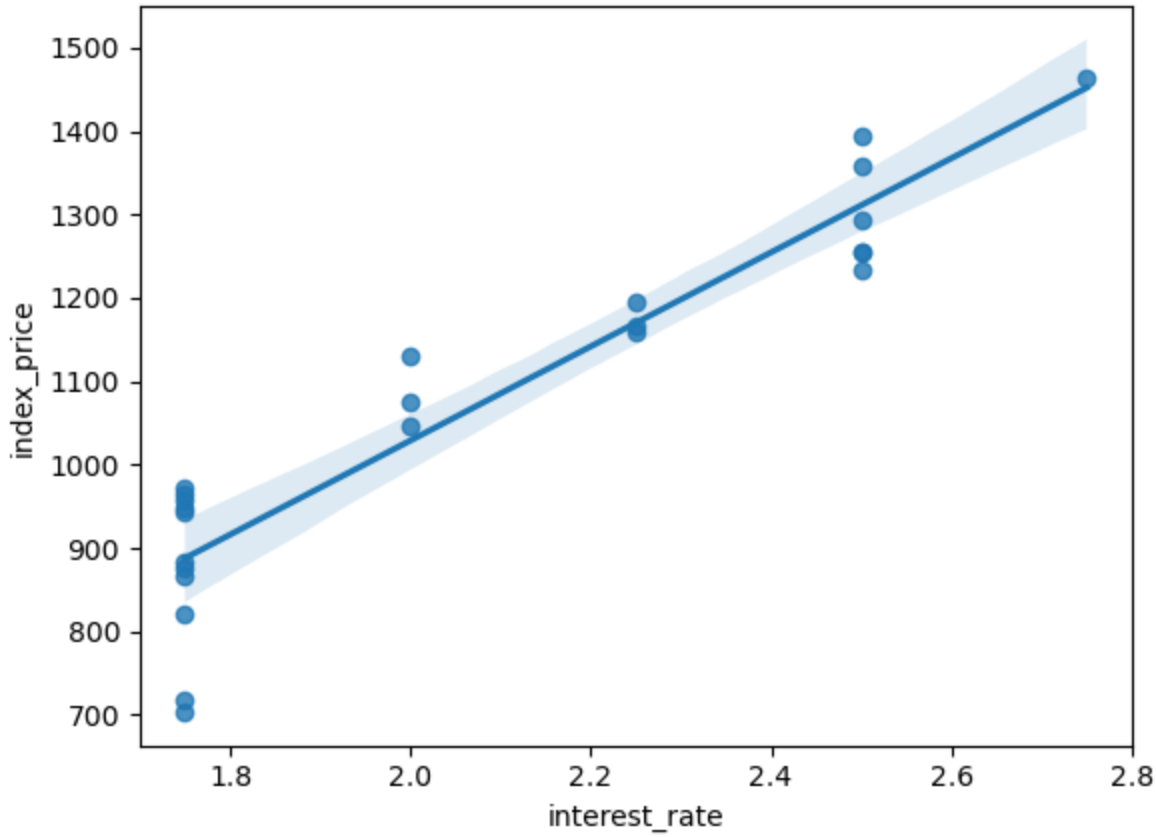
In [27]:

```
sn.regplot(indexprice_df.interest_rate, indexprice_df.index_price)
```

```
/Users/ishutejwani/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[27]:

```
<AxesSubplot:xlabel='interest_rate', ylabel='index_price'>
```



In [12]:
```
X = indexprice_df.iloc[:,:-1]
X
```

Out[12]:

	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
5	2.50	5.6
6	2.50	5.5
7	2.25	5.5
8	2.25	5.5
9	2.25	5.6
10	2.00	5.7
11	2.00	5.9
12	2.00	6.0
13	1.75	5.9
14	1.75	5.8
15	1.75	6.1
16	1.75	6.2
17	1.75	6.1
18	1.75	6.1

19	1.75	6.1
20	1.75	5.9
21	1.75	6.2
22	1.75	6.2
23	1.75	6.1

In [13]:
```
type(X)#independant variable is in dataframe format
```

Out[13]:
```
pandas.core.frame.DataFrame
```

In [14]:
```
Y = indexprice_df.iloc[:,-1]
Y
```

Out[14]:
```
0     1464
1     1394
2     1357
3     1293
4     1256
5     1254
6     1234
7     1195
8     1159
9     1167
10    1130
11    1075
12    1047
13     965
14     943
15     958
16     971
17     949
18     884
19     866
20     876
21     822
22     704
23     719
Name: index_price, dtype: int64
```

In [15]:
```
type(Y)#dependatn variable is in Series format
```

Out[15]:
```
pandas.core.series.Series
```

In [16]:
```
from sklearn.model_selection import train_test_split
```

In [17]:
```
#splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size= 0.3, random_state=
```

In [18]:
```
from sklearn.preprocessing import StandardScaler
```

In [19]:
```
scaler = StandardScaler()
```

In [20]:
```
#standardizing the data
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [21]:
```
from sklearn.linear_model import LinearRegression
```

In [22]:
```
regression = LinearRegression()
```

In [23]:
```
model = regression.fit(X_train, y_train)
```

In [24]:
```
model.coef_
```

Out[24]:
```
array([  96.28689501, -101.57024663])
```

In [25]:
```
print('Coefficients or Slope', model.coef_)
```

```
Coefficients or Slope [  96.28689501 -101.57024663]
```

In [26]:
```
print('Intercept', model.intercept_)
```

```
Intercept 1037.6875
```

In [30]:
```
#performing cross validation
from sklearn.model_selection import cross_val_score
validation_score = cross_val_score(regression,X_train, y_train, scoring='neg_mean_square
```

In [31]:
```
validation_score
```

Out[31]:
```
array([-5495.88007512, -7496.90768619, -9203.37178948, -9743.00656351,
        -523.25201601])
```

In [32]:
```
np.mean(validation_score)
```

Out[32]:
```
-6492.48362606083
```

In [34]:
```
y_pred = model.predict(X_test)
```

In [35]:
```
y_pred
```

Out[35]:
```
array([1192.13083729,  824.23971817, 1400.41971162,  856.16016713,
        992.22505325, 1160.21038833,  920.00106505, 1328.19572341])
```

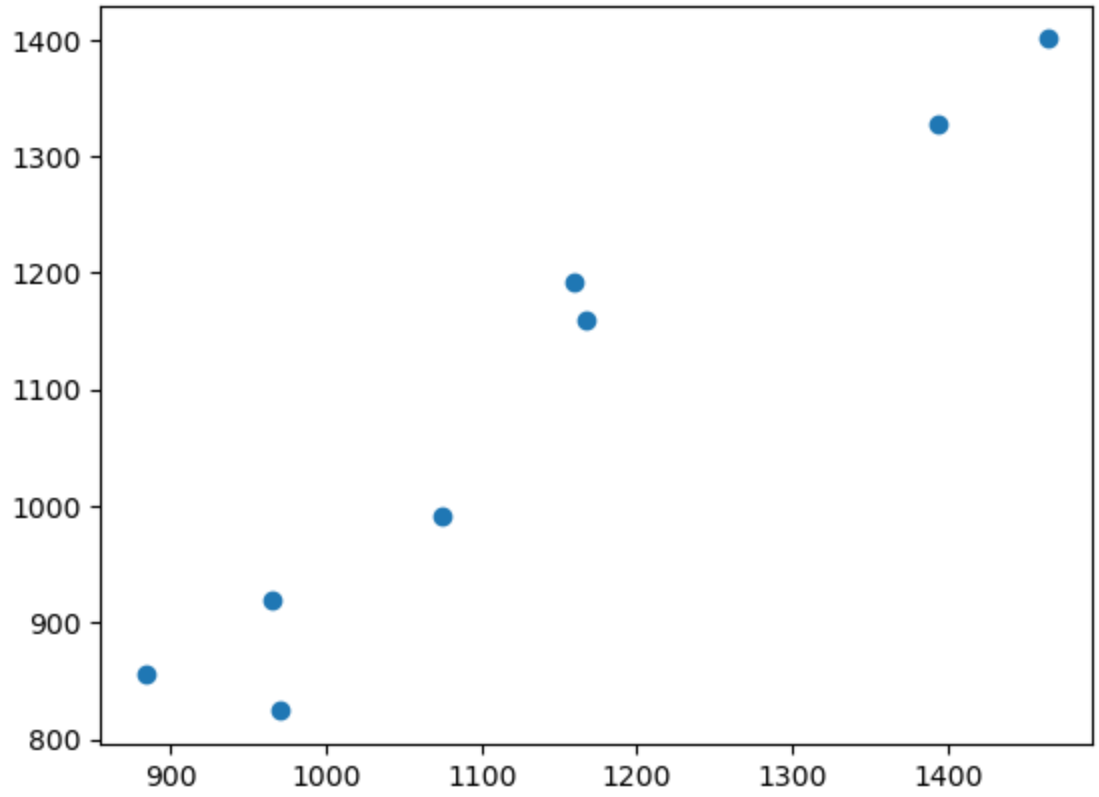
In [37]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

In [38]:
```
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mse)
score = r2_score(y_test, y_pred)
print('Mean Squared Error:', mse)
print('Mean Absolute Error:', mae)
print('Root Mean Squared Error:', rmse)
print('R squared:', score)
```

```
Mean Squared Error: 5088.329958294
Mean Absolute Error: 58.95987629034906
Root Mean Squared Error: 71.33253085580239
R squared: 0.8640024299625207
```

In [39]:
```
plt.scatter(y_test,y_pred)#the data shows that the predicition is linear with absolute d
#the model has performed fairly well
```

Out[39]:
```
<matplotlib.collections.PathCollection at 0x7fb207a56a00>
```

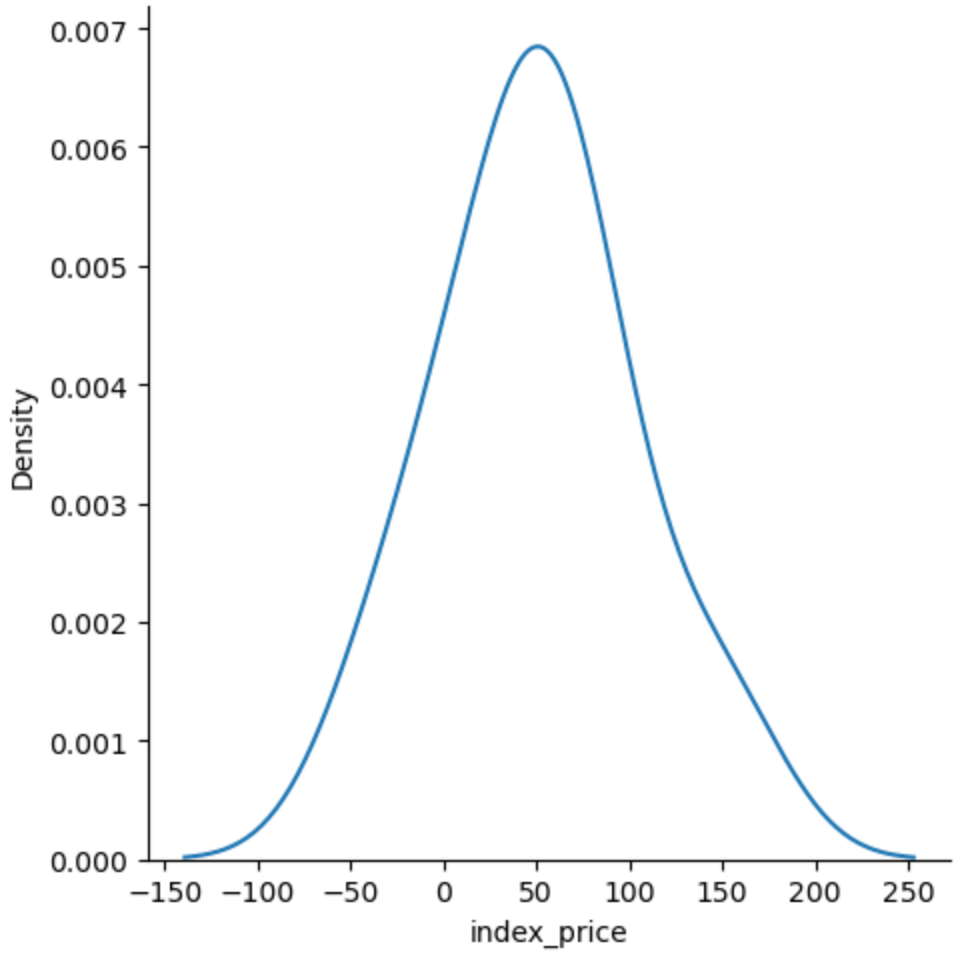


In [40]:
```
residuals = y_test - y_pred
residuals
```

Out[40]:
```
8     -33.130837
16    146.760282
0      63.580288
18     27.839833
11     82.774947
9       6.789612
13     44.998935
1      65.804277
Name: index_price, dtype: float64
```

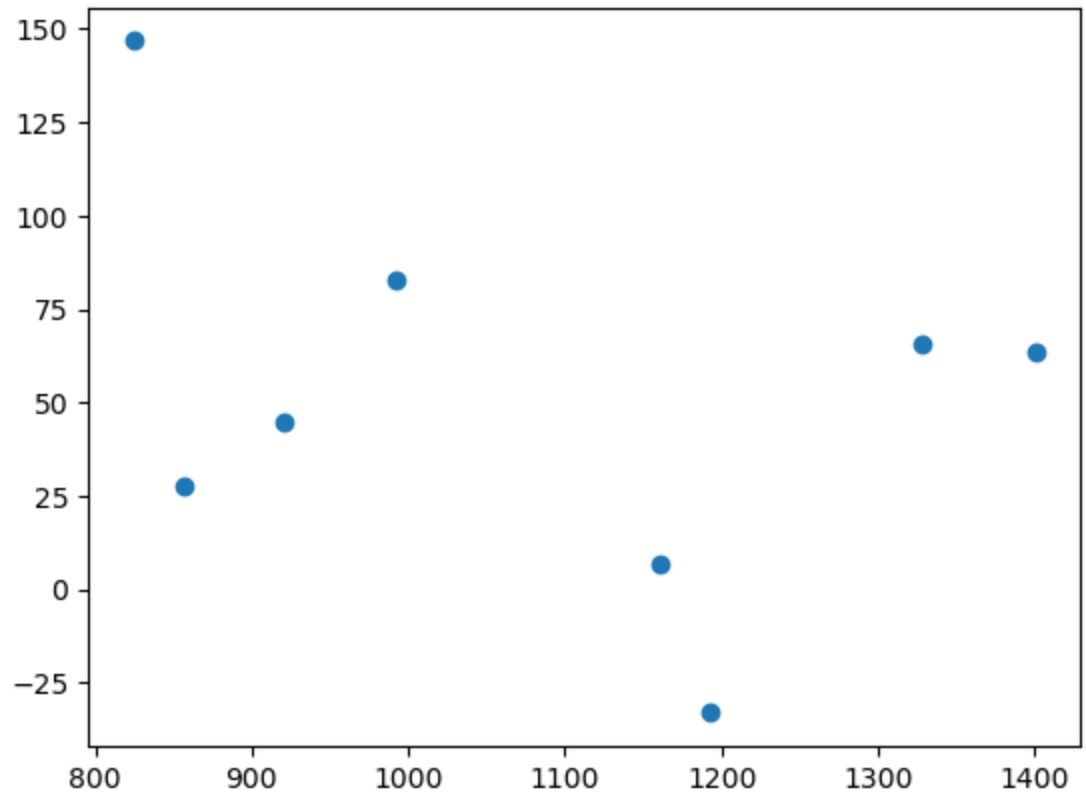
In [41]:
```
sn.displot(residuals, kind= 'kde')#residuals are normally distributed. this also means t
#performed well
```

Out[41]:
```
<seaborn.axisgrid.FacetGrid at 0x7fb2046e7850>
```



In [42]:
```
plt.scatter(y_pred, residuals)#the data is uniformly distributed which means there is no
#is performing well
```

Out[42]:
```
<matplotlib.collections.PathCollection at 0x7fb20661fee0>
```



In [44]:
```
#performing prediction by entering interest rate as 3.5 and unemployment rate as 7.5.
model.predict(scaler.transform([[3.5, 7.5]]))#the predicted index price is 914.84
```

```
/Users/ishutejwani/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:465: UserWa
rning: X does not have valid feature names, but StandardScaler was fitted with feature n
ames
  warnings.warn(
```

Out[44]:
```
array([914.84179912])
```