```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline


df_index = pd.read_csv('/content/economic_index.csv')


df_index.head()
```

	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

Next steps: Generate code with df_index View recommended plots

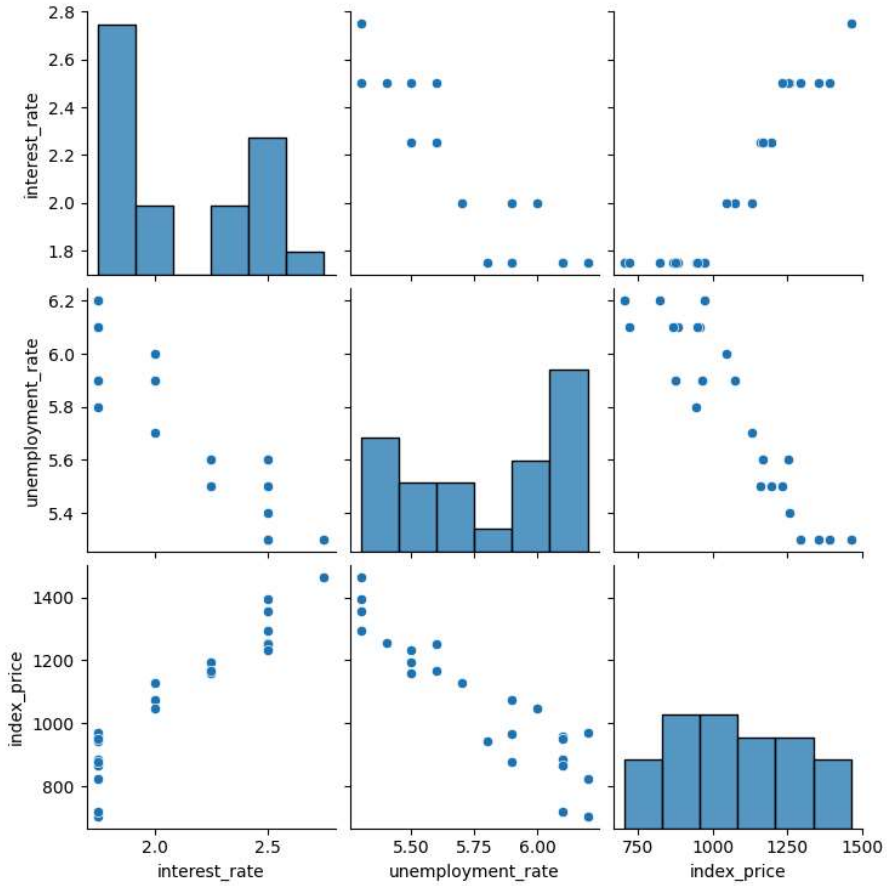
```
#drop unnecessary columns
df_index.drop(columns = ["Unnamed: 0", "year", "month"], axis=1, inplace=True)


df_index.head()
```

	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

Next steps: Generate code with df_index View recommended plots

```
#let do some visulaization
import seaborn as sns
sns.pairplot(df_index)
```

```
<seaborn.axisgrid.PairGrid at 0x7f08c9a7bd00>
```

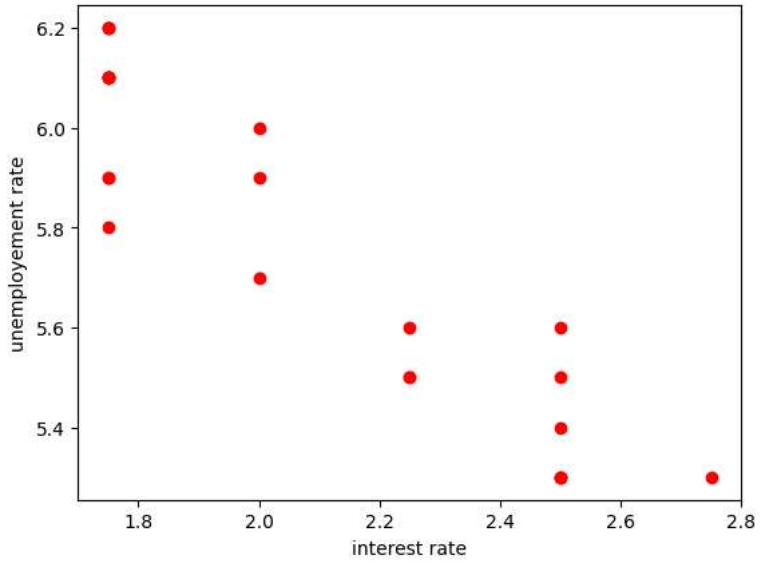



```
df_index.corr()
```

	interest_rate	unemployment_rate	index_price
interest_rate	1.000000	-0.925814	0.935793
unemployment_rate	-0.925814	1.000000	-0.922338
index_price	0.935793	-0.922338	1.000000

```
plt.scatter(df_index['interest_rate'], df_index['unemployment_rate'], color='r')
plt.xlabel("interest rate")
plt.ylabel("unemployement rate")
```

```
Text(0, 0.5, 'unemployement rate')
```




```
#independent and depend features
X  = df_index.iloc[:,:-1]
y = df_index.iloc[:,-1]
```

```
# train test split
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=42)
```

```
import seaborn as sns
```

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

```
X_train
```

```
array([[-0.90115511,  0.37908503],
       [ 1.31077107, -1.48187786],
       [-0.90115511,  1.30956648],
       [ 1.31077107, -0.55139641],
       [ 1.31077107, -1.48187786],
       [-0.16384638,  0.68924552],
       [-0.90115511,  0.999406  ],
       [ 1.31077107, -1.48187786],
       [ 1.31077107, -1.17171738],
       [-0.90115511,  1.30956648],
       [-0.90115511,  0.999406  ],
       [-0.90115511,  0.37908503],
       [-0.90115511,  0.999406  ],
       [ 0.57346234, -0.8615569 ],
       [-0.16384638, -0.24123593],
       [-0.90115511,  0.06892455],
       [-0.90115511,  0.999406  ],
       [ 1.31077107, -0.8615569 ]])
```

```
from sklearn.linear_model import LinearRegression
regression=LinearRegression()
```

```
regression.fit(X_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
## cross validation
from sklearn.model_selection import cross_val_score
validation_score=cross_val_score(regression,X_train,y_train,scoring='neg_mean_squared_error',
                                 cv=3)
```

```
np.mean(validation_score)
```

```
-5914.828180162386
```

```
#prediction
y_pred = regression.predict(X_test)
```

```
y_pred
```

```
array([1180.7466813 ,  802.74279699, 1379.83457045,  838.52599602,
        973.85313963, 1144.96348227])
```

```
## Performance Metrics
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)
```

```
8108.567426306604
73.80444932337097
90.04758423359621
```

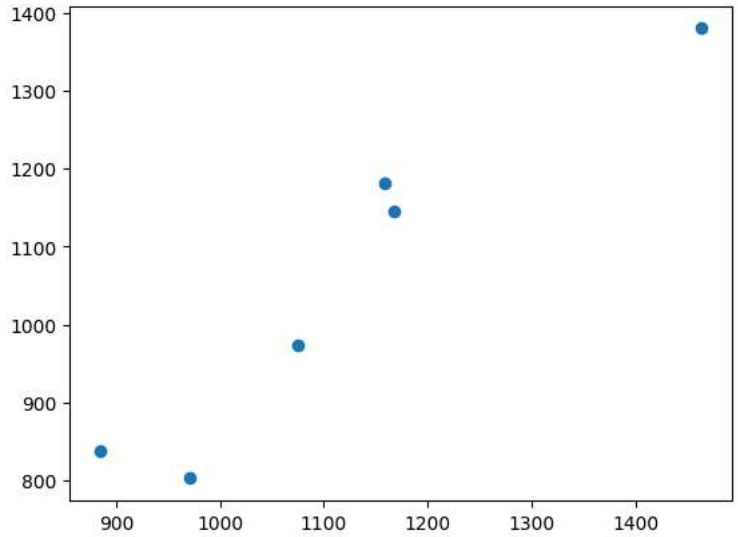
```
from sklearn.metrics import r2_score
score=r2_score(y_test,y_pred)
print(score)
#display adjusted R-squared
print(1 - (1-score)*(len(y_test)-1)/(len(y_test)-X_test.shape[1]-1))
```

```
0.7591371539010257
0.5985619231683761
```

Assumptions

```
plt.scatter(y_test,y_pred)
```

```
<matplotlib.collections.PathCollection at 0x7f08c80f5720>
```



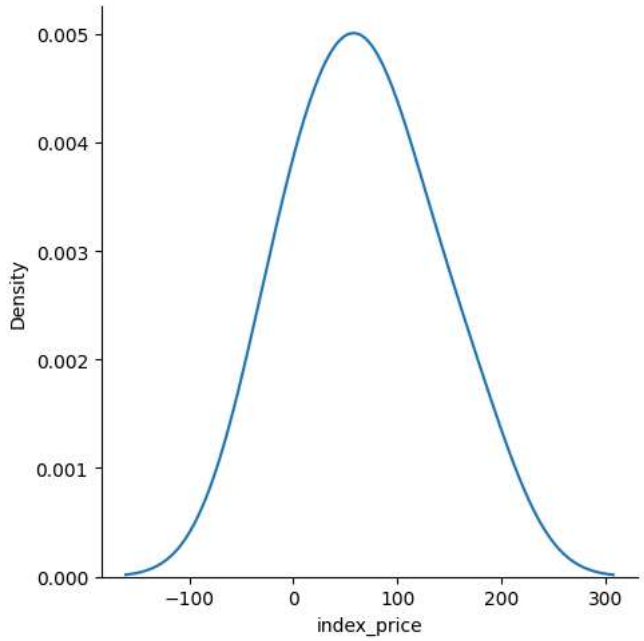

```
residuals=y_test-y_pred
print(residuals)
```

```
8     -21.746681
16    168.257203
0      84.165430
```

```
18     45.474004
11    101.146860
9      22.036518
Name: index_price, dtype: float64
```

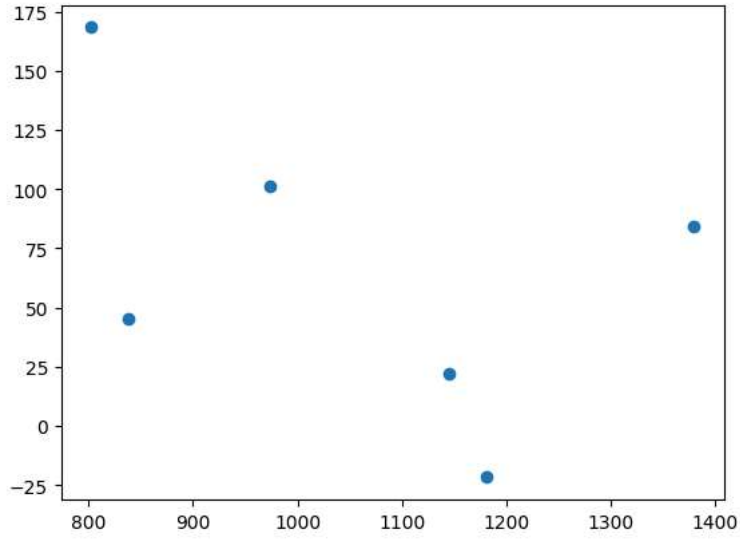
```
## Plot this residuals
sns.displot(residuals,kind='kde')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f08cf671db0>
```




```
## scatter plot with respect to prediction and residuals
plt.scatter(y_pred,residuals)
```

```
<matplotlib.collections.PathCollection at 0x7f08c83377f0>
```




```
## OLS Linear Regression
import statsmodels.api as sm
model=sm.OLS(y_train,X_train).fit()
```

```
model.summary()
```

```
/usr/local/lib/python3.10/dist-packages/scipy/stats/_stats_py.py:1806: UserWarning: kurtosistest only valid for n>=20 ... continuing any
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
```

OLS Regression Results

Dep. Variable:	index_price	**R-squared (uncentered):**	0.035
Model:	OLS	**Adj. R-squared (uncentered):**	-0.086
Method:	Least Squares	**F-statistic:**	0.2880