

# Chapter 9 - Ex4: Economy

- Cho dữ liệu economy.xlsx. Bộ dữ liệu chứa thông tin về các tỷ lệ Interest\_Rate, Unemployment\_Rate và Stock\_Index\_Price theo thời gian (năm, tháng). Người ta dự đoán Stock\_Index\_Price dựa trên Interest\_Rate, hoặc Unemployment\_Rate hoặc cả 2.

## Yêu cầu:

- Đọc dữ liệu, tiền xử lý dữ liệu, tổng quan ban đầu về dữ liệu.

### 1. Single variable

- Thực hiện Simple Linear Regression để dự đoán Stock\_Index\_Price từ Interest\_Rate, với các đánh giá mô hình: Training vs Testing 80:20
- Nhận xét kết quả. Trực quan hóa kết quả.
- Kiểm chứng overfitting, underfitting của model vừa chọn
- Đánh giá mô hình vừa xây dựng: có cần phải cải tiến gì không? Nếu cần thì thay đổi mô hình.
- Nhận xét kết quả. Trực quan hóa kết quả với mô hình mới thay đổi. So sánh với mô hình ban đầu. Mô hình sau có tốt hơn không? Quyết định chọn mô hình nào? Lý do?

### 2. Multiple variables

- Thực hiện Multiple Linear Resgression để dự đoán Stock\_Index\_Price từ Interest\_Rate, Unemployment\_Rate. với các đánh giá mô hình: Training vs Testing 80:20
- Nhận xét kết quả. Trực quan hóa kết quả.
- Kiểm chứng overfitting, underfitting của model vừa chọn
- Đánh giá mô hình vừa xây dựng: có cần phải cải tiến gì không? Nếu cần thì thay đổi mô hình.
- Nhận xét kết quả. Trực quan hóa kết quả với mô hình mới thay đổi. So sánh với mô hình ban đầu. Mô hình sau có tốt hơn không? Quyết định chọn mô hình nào? Lý do?

```
In [1]: # from google.colab import drive  
# drive.mount("/content/gdrive", force_remount=True)  
# %cd '/content/gdrive/My Drive/MDS5_2022/Practice_2022/Chapter9/'
```

```
In [2]: import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns
```

```
In [3]: # Đọc dữ Liệu, tiền xử Lý dữ Liệu, tổng quan ban đầu về dữ Liệu  
data = pd.read_excel("economy.xlsx")
```

```
In [4]: data.head()
```

Out[4]:

	Year	Month	Interest_Rate	Unemployment_Rate	Stock_Index_Price
0	2017	12	2.75	5.3	1464
1	2017	11	2.50	5.3	1394
2	2017	10	2.50	5.3	1357
3	2017	9	2.50	5.3	1293
4	2017	8	2.50	5.4	1256

```
In [5]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24 entries, 0 to 23
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Year            24 non-null      int64  
 1   Month           24 non-null      int64  
 2   Interest_Rate   24 non-null      float64 
 3   Unemployment_Rate 24 non-null      float64 
 4   Stock_Index_Price 24 non-null      int64  
dtypes: float64(2), int64(3)
memory usage: 1.1 KB
```

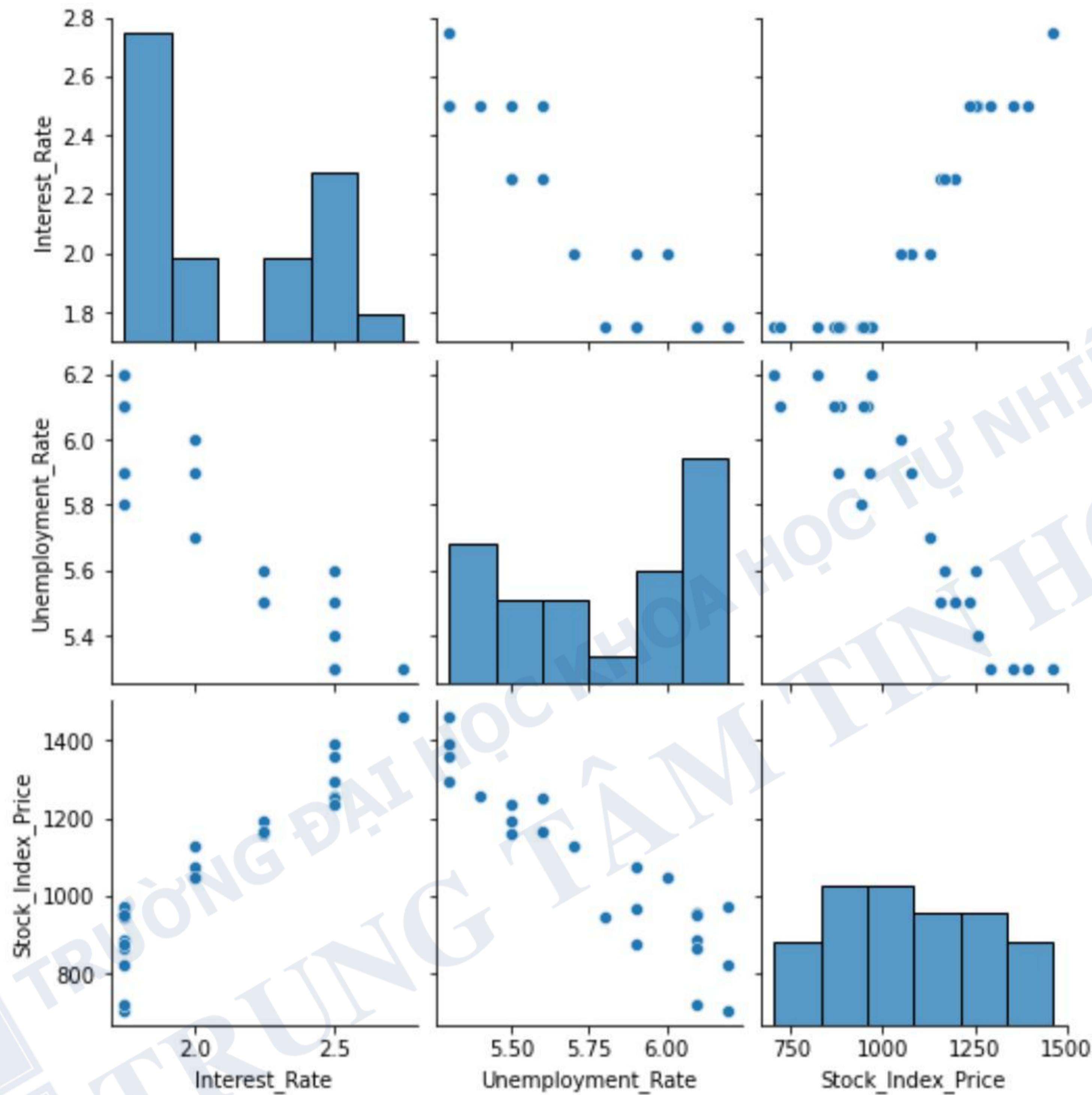
```
In [6]: data.describe()
```

Out[6]:

	Year	Month	Interest_Rate	Unemployment_Rate	Stock_Index_Price
count	24.000000	24.000000	24.000000	24.000000	24.000000
mean	2016.500000	6.500000	2.072917	5.77500	1070.083333
std	0.510754	3.526299	0.349527	0.33002	210.735341
min	2016.000000	1.000000	1.750000	5.30000	704.000000
25%	2016.000000	3.750000	1.750000	5.50000	928.250000
50%	2016.500000	6.500000	2.000000	5.85000	1061.000000
75%	2017.000000	9.250000	2.500000	6.10000	1239.000000
max	2017.000000	12.000000	2.750000	6.20000	1464.000000

```
In [7]: # không có dữ liệu null
```

```
In [8]: sns.pairplot(data[["Interest_Rate", "Unemployment_Rate", "Stock_Index_Price"]])  
plt.show()
```



```
In [9]: # Nhận xét: Stock Index Price gần như tương quan thuận với Interest Rate  
# và gần như tương quan nghịch với Unemployment Rate
```

## Single Variable

### Simple Linear Regression

#### Training & Testing

```
In [10]: from sklearn.model_selection import train_test_split  
from sklearn.linear_model import LinearRegression
```

```
In [11]: y = data.Stock_Index_Price  
X = data[["Interest_Rate"]]
```

```
In [12]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,  
random_state = 42)
```

```
In [13]: print("number of test samples :", X_test.shape[0])  
print("number of training samples:", X_train.shape[0])
```

number of test samples : 5  
number of training samples: 19

```
In [14]: lre = LinearRegression()
```

```
In [15]: lre.fit(X_train, y_train)
```

```
Out[15]: LinearRegression()
```

```
In [16]: # R^2, with training data  
lre.score(X_train, y_train)
```

```
Out[16]: 0.8563765019490677
```

```
In [17]: # R^2, with testing data  
lre.score(X_test, y_test)
```

```
Out[17]: 0.9412459051773162
```

```
In [18]: # R^2, with dataset  
lre.score(X, y)
```

```
Out[18]: 0.8743972896481225
```

```
In [19]: # Có sự chênh lệch score ~0.1 giữa training dataset và testing dataset  
# Với cả dataset: có thể nói rằng ~ 87.5% variation của Stock Index Price  
# được giải thích bằng simple Linear model này.
```

```
In [20]: yhat_LR = lre.predict(X)
```

```
In [21]: df_compare_LR = pd.DataFrame({"Actual": y.values, "Predicted": yhat_LR})
df_compare_LR.head()
```

Out[21]:

	Actual	Predicted
0	1464	1452.581250
1	1394	1308.926563
2	1357	1308.926563
3	1293	1308.926563
4	1256	1308.926563

## Overfitting, Underfitting & Model Selection

```
In [22]: yhat_train = lre.predict(X_train)
```

```
In [23]: yhat_test = lre.predict(X_test)
```

```
In [24]: y_train_mean = y_train.mean()
yhat_train_mean = yhat_train.mean()
display(y_train_mean, yhat_train_mean)
```

1059.421052631579

1059.4210526315792

```
In [25]: y_train_std = y_train.std()
yhat_train_std = yhat_train.std()
display(y_train_std, yhat_train_std)
```

212.35591606479744

191.27410907777676

```
In [26]: y_test_mean = y_test.mean()
yhat_test_mean = yhat_test.mean()
display(y_test_mean, yhat_test_mean)
```

1110.6

1079.0790625

```
In [27]: y_test_std = y_test.std()
yhat_test_std = yhat_test.std()
display(y_test_std, yhat_test_std)
```

223.23597380350685

215.00264905163095

```
In [28]: plt.figure(figsize=(14,6))
plt.subplot(1,2,1)
ax1 = sns.distplot(y_train, hist=False, color="r",
                    label="Actual Train Values")
sns.distplot(yhat_train, hist=False, color="b",
              label="Predicted Train Values", ax=ax1)

plt.subplot(1,2,2)
ax2 = sns.distplot(y_test, hist=False, color="r",
                    label="Actual Test Values")
sns.distplot(yhat_test, hist=False, color="b",
              label="Predicted Test Values", ax=ax2)

plt.show()
```

c:\program files\python36\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

warnings.warn(msg, FutureWarning)

c:\program files\python36\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

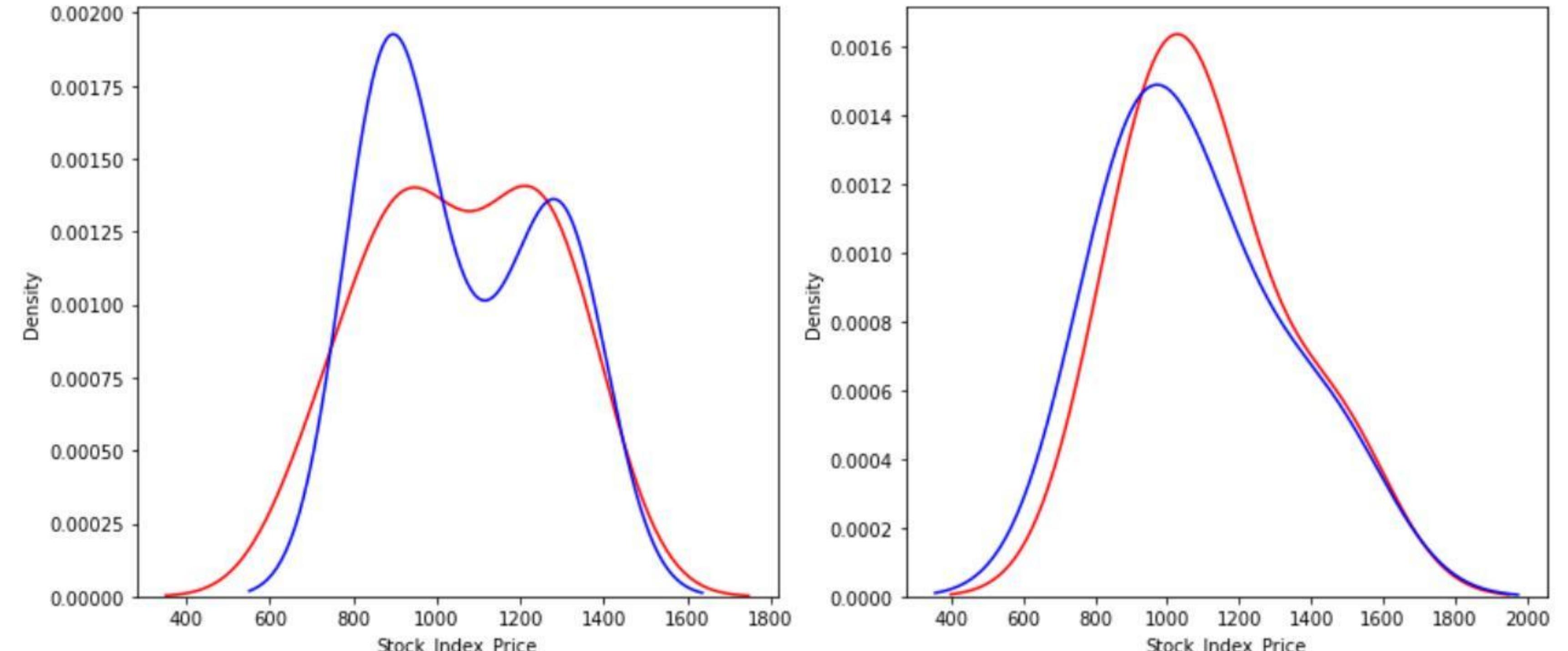
warnings.warn(msg, FutureWarning)

c:\program files\python36\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

warnings.warn(msg, FutureWarning)

c:\program files\python36\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

warnings.warn(msg, FutureWarning)



Với kết quả trên ta thấy:

- Training Data: thì trung bình dự đoán bằng thực tế, độ lệch chuẩn của dự đoán thấp hơn thực tế, do đó phân phối ở dự đoán có hình vẽ cao hơn thực tế
- Testing Data: thì trung bình dự đoán thấp hơn thực tế, độ lệch chuẩn của dự đoán thấp thực tế, do đó phân phối ở dự đoán có hình vẽ thấp hơn thực tế
- So sánh Hình 1 và Hình 2, chúng ta thấy là sự phân phối dữ liệu thử nghiệm ở trong Hình 1 không tốt hơn trong việc fit dữ liệu.

## Áp dụng Polinorminal Regression

```
In [29]: from sklearn.preprocessing import PolynomialFeatures
```

```
In [30]: # Sử dụng 45% dataset làm testing data
```

```
In [31]: X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                       test_size=0.2,
                                                       random_state= 42)
```

```
In [32]: pr = PolynomialFeatures(degree=3)
X_train_pr = pr.fit_transform(X_train)
X_test_pr = pr.fit_transform(X_test)
pr
```

```
Out[32]: PolynomialFeatures(degree=3)
```

```
In [33]: X_train.shape, X_train_pr.shape
```

```
Out[33]: ((19, 1), (19, 4))
```

```
In [34]: poly = LinearRegression()
poly.fit(X_train_pr, y_train)
```

```
Out[34]: LinearRegression()
```

```
In [35]: yhat = poly.predict(X_test_pr)
yhat[0:5]
```

```
Out[35]: array([1181.          , 866.88888889, 1593.11111111, 866.88888889,
               1088.5        ])
```

```
In [36]: df_compare_pr= pd.DataFrame({"Actual": y_test.values, "Predicted": yhat})  
df_compare_pr.head()
```

Out[36]:

	Actual	Predicted
0	1159	1181.000000
1	971	866.888889
2	1464	1593.111111
3	884	866.888889
4	1075	1088.500000

```
In [37]: yhat_test_pr= poly.predict(X_test_pr)  
yhat_train_pr = poly.predict(X_train_pr)
```

```
In [38]: # R^2 of the training data:  
poly.score(X_train_pr, y_train)
```

Out[38]: 0.8702500807658259

```
In [39]: # R^2 of the test data:  
poly.score(X_test_pr, y_test)
```

Out[39]: 0.8571875064334733

```
In [40]: # R^2 of the dataset:  
X_pr = pr.fit_transform(X)  
poly.score(X_pr, y)
```

Out[40]: 0.8690178601942646

```
In [41]: # Với cả dataset: có thể nói rằng ~ 87% variation của Stock Index Price  
# được giải thích bằng polinomial linear model này.
```

```
In [42]: plt.figure(figsize=(14,6))
plt.subplot(1,2,1)
ax1 = sns.distplot(y_train, hist=False, color="r",
                    label="Actual Train Values")
sns.distplot(yhat_train_pr, hist=False, color="b",
              label="Predicted Train Values", ax=ax1)

plt.subplot(1,2,2)
ax2 = sns.distplot(y_test, hist=False, color="r",
                    label="Actual Test Values")
sns.distplot(yhat_test_pr, hist=False, color="b",
              label="Predicted Test Values", ax=ax2)

plt.show()
```

c:\program files\python36\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

warnings.warn(msg, FutureWarning)

c:\program files\python36\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

warnings.warn(msg, FutureWarning)

c:\program files\python36\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

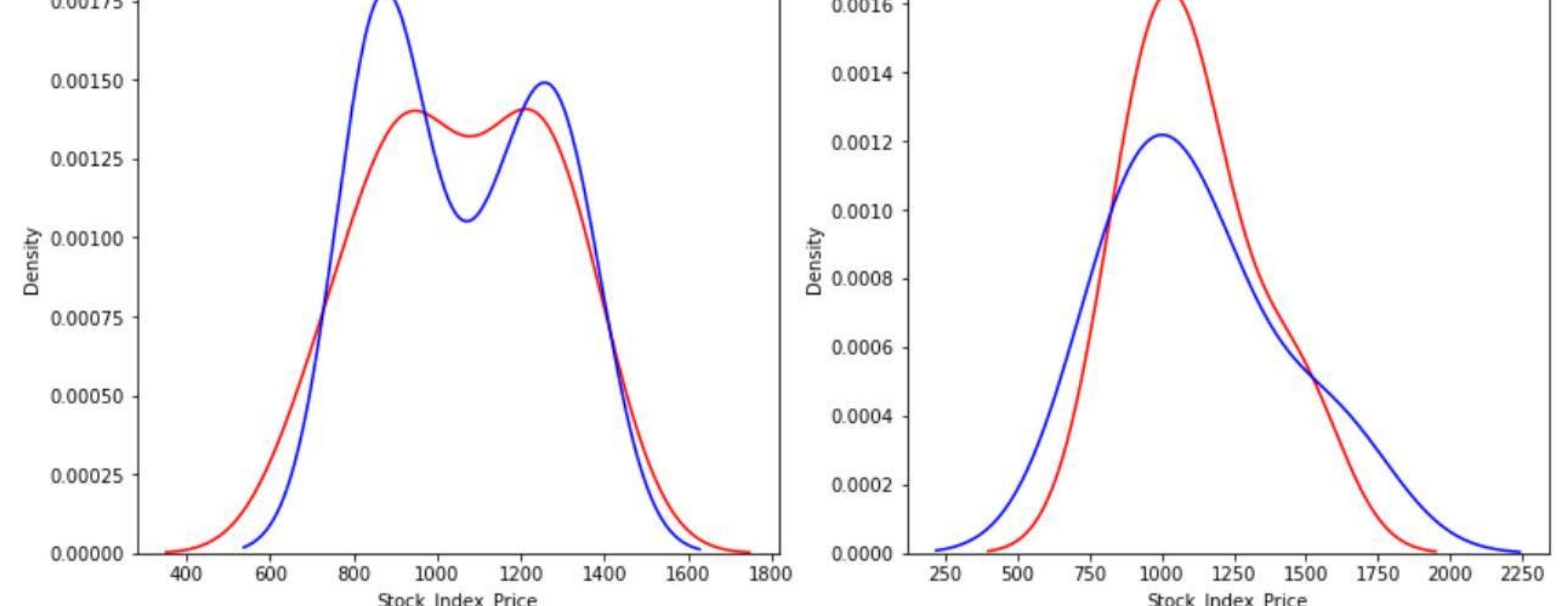
warnings.warn(msg, FutureWarning)

c:\program files\python36\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

warnings.warn(msg, FutureWarning)

c:\program files\python36\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

warnings.warn(msg, FutureWarning)



```
In [43]: # Quan sát hình ta thấy training dataset fit hơn testing dataset.
```

```
In [44]: # Nhận xét: Kết quả Polynomial không tốt hơn Simple Linear Regression
```

## Multiple Variables

### Training & Testing

```
In [45]: X_m = data[["Interest_Rate", "Unemployment_Rate"]]
y_m = data.Stock_Index_Price
```

```
In [46]: X_m.head()
```

Out[46]:

	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 [47]: X_train_m, X_test_m, y_train_m, y_test_m = train_test_split(X_m, y_m,
                                                               test_size=0.2,
                                                               random_state=42)
```

```
In [48]: mlr = LinearRegression()
```

```
In [49]: mlr.fit(X_train_m, y_train_m)
```

Out[49]: LinearRegression()

```
In [50]: # R^2, with training data  
mlr.score(X_train_m, y_train_m)
```

```
Out[50]: 0.9045353261246192
```

```
In [51]: # R^2, with testing data  
mlr.score(X_test_m, y_test_m)
```

```
Out[51]: 0.8254940547158579
```

```
In [52]: mlr.score(X_m, y_m)
```

```
Out[52]: 0.8900788360850509
```

```
In [53]: # Có sự chênh lệch score ~0.1 giữa training dataset và testing dataset  
# Với cả dataset: có thể nói rằng ~ 89% variation của Stock Index Price
```

## Overfitting, Underfitting & Model Selection

```
In [54]: yhat_train_m = mlr.predict(X_train_m)
```

```
In [55]: yhat_test_m = mlr.predict(X_test_m)
```

```
In [56]: y_train_mean_m = y_train_m.mean()  
yhat_train_mean_m = yhat_train_m.mean()  
display(y_train_mean_m, yhat_train_mean_m)
```

```
1059.421052631579
```

```
1059.4210526315787
```

```
In [57]: y_train_std_m = y_train_m.std()  
yhat_train_std_m = yhat_train_m.std()  
display(y_train_std_m, yhat_train_std_m)
```

```
212.35591606479744
```

```
196.5787562616388
```

```
In [58]: y_test_mean_m = y_test_m.mean()  
yhat_test_mean_m = yhat_test_m.mean()  
display(y_test_mean_m, yhat_test_mean_m)
```

```
1110.6
```

```
1056.9388308651596
```

```
In [59]: y_test_std_m = y_test_m.std()  
yhat_test_std_m = yhat_test_m.std()  
display(y_test_std_m, yhat_test_std_m)
```

223.23597380350685

220.32339715165364



```
In [60]: plt.figure(figsize=(14,6))
plt.subplot(1,2,1)
ax1 = sns.distplot(y_train_m, hist=False, color="r",
                    label="Actual Train Values")
sns.distplot(yhat_train_m, hist=False, color="b",
              label="Predicted Train Values", ax=ax1)

plt.subplot(1,2,2)
ax2 = sns.distplot(y_test_m, hist=False, color="r",
                    label="Actual Test Values")
sns.distplot(yhat_test_m, hist=False, color="b",
              label="Predicted Test Values", ax=ax2)

plt.show()
```

c:\program files\python36\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

warnings.warn(msg, FutureWarning)

c:\program files\python36\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

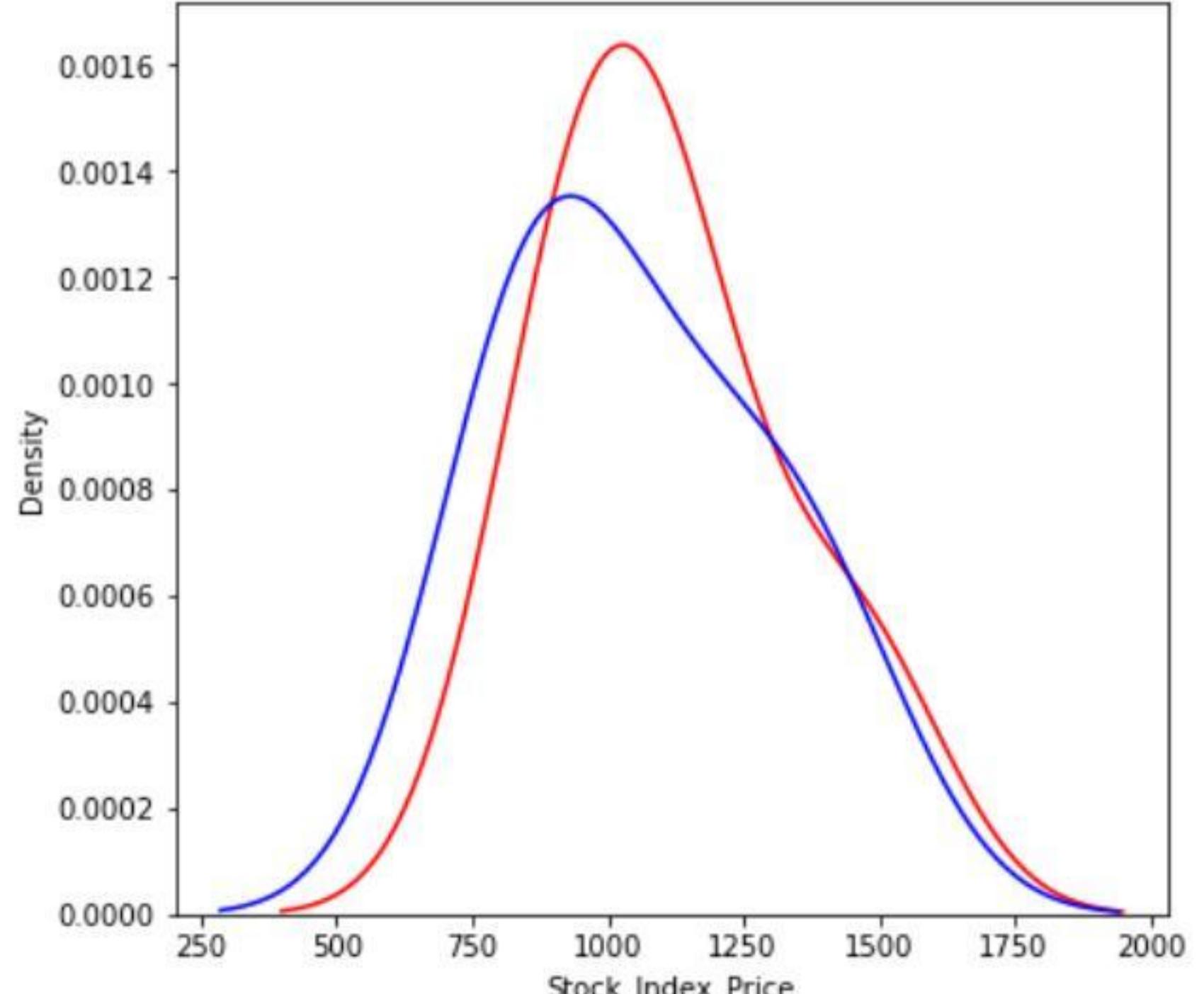
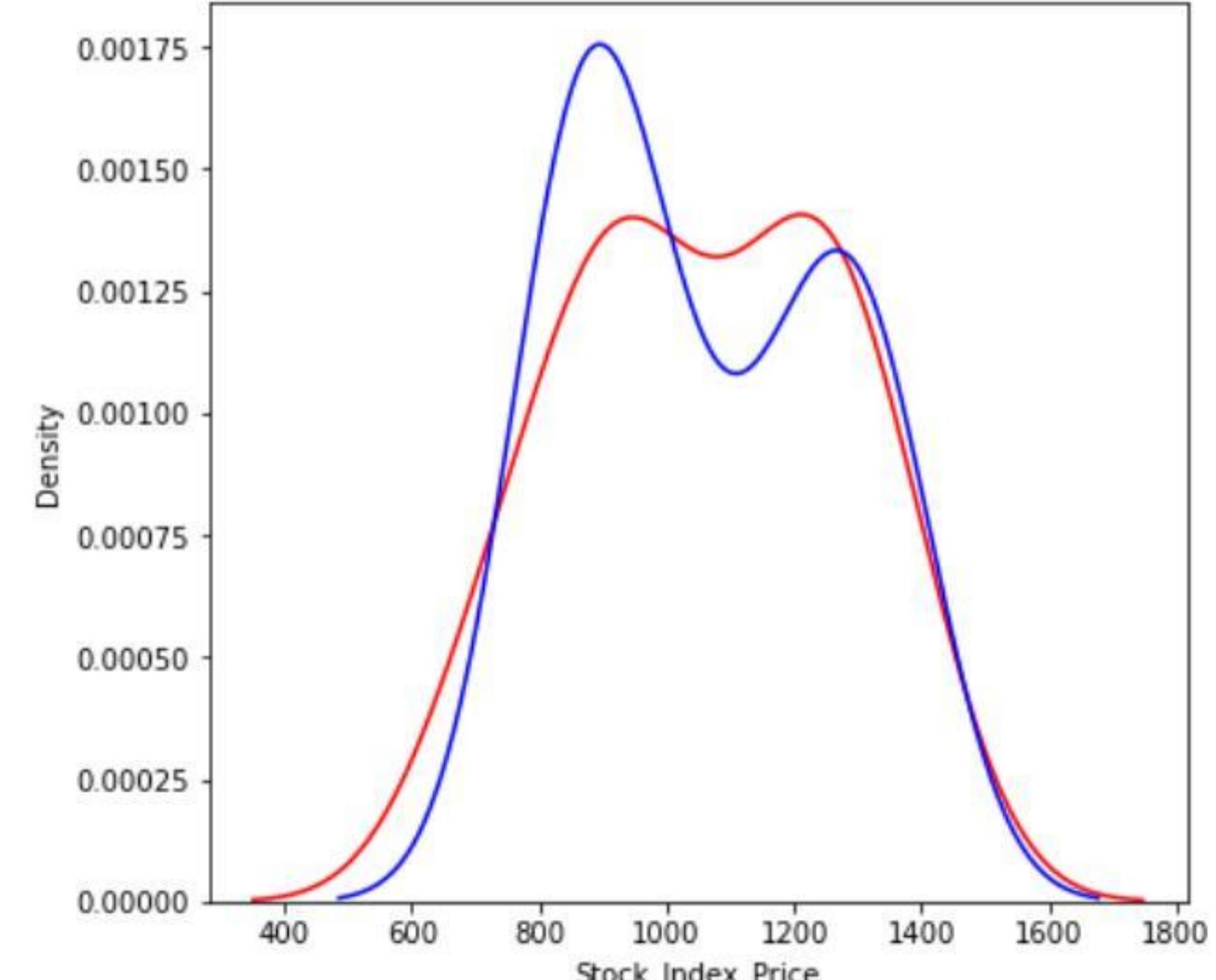
warnings.warn(msg, FutureWarning)

c:\program files\python36\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

warnings.warn(msg, FutureWarning)

c:\program files\python36\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

warnings.warn(msg, FutureWarning)



Với kết quả trên ta thấy:

- Training Data: thì trung bình dự đoán bằng thực tế, độ lệch chuẩn của dự đoán thấp hơn thực tế, do đó phân phối ở dự đoán có hình vẽ cao hơn thực tế.
- Testing Data: thì trung bình dự đoán thấp hơn thực tế, độ lệch chuẩn của dự đoán thấp thực tế, do đó phân phối ở dự đoán có hình vẽ thấp hơn thực tế.
- So sánh Hình 1 và Hình 2, chúng ta thấy là sự phân phối dữ liệu thử nghiệm ở trong Hình 1 tốt hơn trong việc fit dữ liệu.
- Sự khác biệt này trong Hình 2 là rõ ràng để nhận biết

## Áp dụng Polinomial Regression

```
In [61]: from sklearn.preprocessing import PolynomialFeatures
```

```
In [62]: X_train_m, X_test_m, y_train_m, y_test_m = train_test_split(X_m, y_m,  
test_size=0.2,  
random_state=0)
```

```
In [63]: pr = PolynomialFeatures(degree=3)  
pr.fit(X_m)  
X_pr_m = pr.transform(X_m)  
X_train_pr_m = pr.transform(X_train_m)  
X_test_pr_m = pr.transform(X_test_m)  
pr
```

```
Out[63]: PolynomialFeatures(degree=3)
```

```
In [64]: X_train_m.shape, X_train_pr_m.shape
```

```
Out[64]: ((19, 2), (19, 10))
```

```
In [65]: poly = LinearRegression()  
poly.fit(X_train_pr_m, y_train_m)
```

```
Out[65]: LinearRegression()
```

```
In [66]: yhat_m = poly.predict(X_test_pr_m)  
yhat_m[0:5]
```

```
Out[66]: array([1018.53654567, 1085.26443226, 893.17037487, 931.54557971,  
962.66926241])
```

```
In [67]: df_compare_pr_m = pd.DataFrame({"Actual": y_test_m.values, "Predicted": yhat_m})
df_compare_pr_m.head()
```

Out[67]:

	Actual	Predicted
0	1075	1018.536546
1	1130	1085.264432
2	704	893.170375
3	943	931.545580
4	876	962.669262

```
In [68]: # R^2 of the training data:
poly.score(X_train_pr_m, y_train_m)
```

Out[68]: 0.9317399808627672

```
In [69]: # R^2 of the test data:
poly.score(X_test_pr_m, y_test_m)
```

Out[69]: 0.5734146750502889

```
In [70]: poly.score(X_pr_m, y_m)
```

Out[70]: 0.8982987202613253

```
In [71]: # Có sự chênh lệch score cao giữa training dataset và testing dataset => Overfitting
# Với cả dataset: có thể nói rằng ~ 89% variation của Stock Index Price
```

```
In [72]: yhat_test_m_p = poly.predict(X_test_pr_m)
yhat_train_m_p = poly.predict(X_train_pr_m)
```

```
In [73]: plt.figure(figsize=(14,6))
plt.subplot(1,2,1)
ax1 = sns.distplot(y_train_m, hist=False, color="r",
                    label="Actual Train Values")
sns.distplot(yhat_train_m_p, hist=False, color="b",
              label="Predicted Train Values", ax=ax1)

plt.subplot(1,2,2)
ax2 = sns.distplot(y_test_m, hist=False, color="r",
                    label="Actual Test Values")
sns.distplot(yhat_test_m_p , hist=False, color="b",
              label="Predicted Test Values" , ax=ax2)

plt.show()
```

c:\program files\python36\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

warnings.warn(msg, FutureWarning)

c:\program files\python36\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

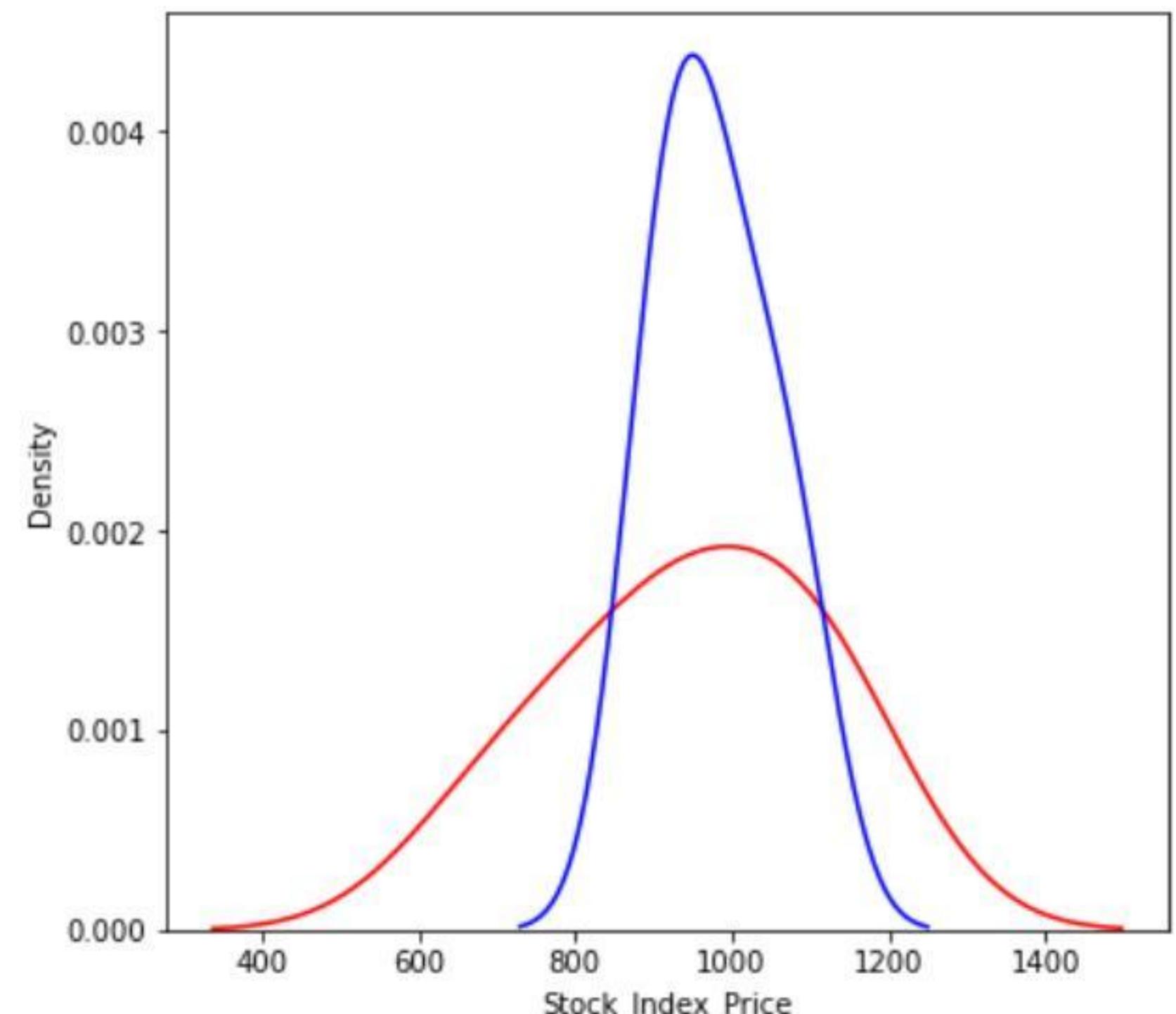
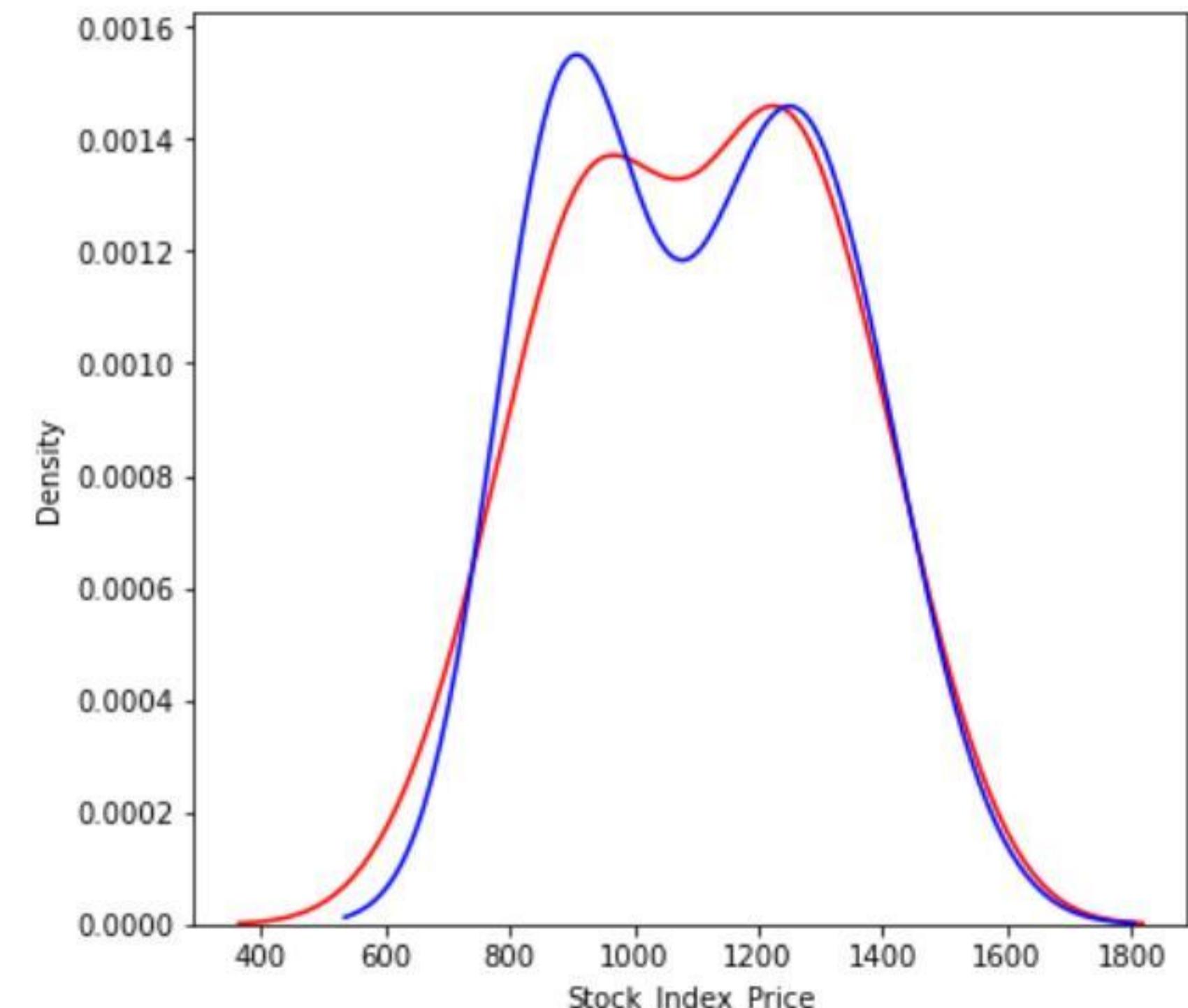
warnings.warn(msg, FutureWarning)

c:\program files\python36\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

warnings.warn(msg, FutureWarning)

c:\program files\python36\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

warnings.warn(msg, FutureWarning)



In [74]: # Quan sát hình ta thấy training dataset fit hơn rất nhiều so với testing dataset.  
# Kết quả: Multiple Polynominal không tốt so với Multiple Regression.

In [75]: # Vấn đề còn Lại:  
# Bộ dữ liệu quá ít, để chính xác hơn thì cần thu thập nhiều dữ liệu hơn

