Mục lục

1 Giới thiệu

1.1 Đặt vấn đề

Trong bối cảnh kinh tế hiện đại, việc phân tích và dự đoán giá cả hàng hóa, đặc biệt là các sản phẩm nông nghiệp, đóng vai trò quan trọng trong việc hỗ trợ ra quyết định của các bên liên quan, từ nhà sản xuất, nhà bán lẻ đến người tiêu dùng. Giá cả nông sản không chỉ phản ánh mối quan hệ giữa cung và cầu mà còn chịu ảnh hưởng của nhiều yếu tố phức tạp như mùa vụ, điều kiện thời tiết, chi phí vận chuyển, và xu hướng tiêu dùng.

Trong các sản phẩm nông nghiệp, những mặt hàng có tính thời vụ cao và nhu cầu tiêu thụ lớn thường có giá biến động mạnh. Điều này tạo ra thách thức lớn cho các nhà hoạch định chính sách và doanh nghiệp trong việc lập kế hoạch sản xuất, phân phối và kinh doanh. Một giải pháp quan trọng là áp dụng các kỹ thuật phân tích dữ liệu và dự đoán giá cả dựa trên dữ liệu lịch sử, qua đó nhận diện các xu hướng tiềm ẩn và dự báo các biến động trong tương lai.

Bài toán dự đoán giá cả không chỉ dừng lại ở việc đưa ra các con số chính xác mà còn giúp phát hiện các yếu tố chính ảnh hưởng đến giá, đồng thời xác định các giai đoạn hoặc khu vực có khả năng xảy ra biến động giá bất thường. Từ đó, các chiến lược như tối ưu hóa chuỗi cung ứng, điều chỉnh sản lượng, và định giá hợp lý có thể được triển khai để nâng cao hiệu quả hoạt động.

Câu hỏi đặt ra là: Liệu chúng ta có thể sử dụng dữ liệu lịch sử để phân tích và dự đoán chính xác giá cả của một sản phẩm nông nghiệp theo thời gian không? Đây chính là bài toán nhận diện mẫu và dự báo, có ý nghĩa thực tiễn cao không chỉ trong nông nghiệp mà còn trong các lĩnh vực khác như thương mại, vận tải, và thị trường tài chính.

1.2 Định nghĩa bài toán

Bài toán được đặt ra là dự đoán giá của một sản phẩm theo thời gian dựa trên các yếu tố ảnh hưởng. Trong nhiều lĩnh vực, từ nông nghiệp, công nghiệp cho đến tài chính, giá cả sản phẩm thường xuyên biến động do ảnh hưởng của nhiều yếu tố như:

- Thời gian: Các xu hướng dài hạn hoặc các biến động ngắn hạn theo mùa vụ.
- Cung và cầu: Sự thay đổi trong sản lượng hoặc nhu cầu tiêu thụ.
- Đặc tính sản phẩm: Loại sản phẩm, chất lượng, và các yếu tố khác liên quan trực tiếp đến giá trị của sản phẩm.
- Yếu tố địa lý: Khu vực phân phối, chi phí vận chuyển hoặc sự khác biệt về thị trường tiêu dùng.

Bài toán có thể được mô tả cụ thể như sau:

- Đầu vào: Dữ liệu lịch sử bao gồm các thông tin: thời gian, sản lượng, loại sản phẩm, và các yếu tố khác có ảnh hưởng đến giá.
- Đầu ra: Dự đoán giá sản phẩm tại một thời điểm cụ thể trong tương lai.

Mục tiêu của bài toán là xây dựng một mô hình dự đoán chính xác giá sản phẩm Xây dựng một mô hình dự đoán chính xác giá sản phẩm dựa trên các thông tin đầu vào.

Hiểu rõ các yếu tố chính ảnh hưởng đến giá để hỗ trợ các chiến lược quản lý, tối ưu hóa sản xuất, hoặc định giá phù hợp.

2 Chuẩn bị dữ liệu

2.1 Mô tả bộ dữ liệu

Bộ dữ liệu Avocado Prices cung cấp thông tin về giá cả, sản lượng và các yếu tố liên quan đến quả bơ (avocado) tại nhiều khu vực khác nhau ở Mỹ trong khoảng thời gian từ năm 2015 đến 2018. Dữ liệu này được sử dụng phổ biến để phân tích thị trường nông sản và xây dựng các mô hình dự đoán giá cả. Bộ dữ liệu được chia sẻ từ trang web Hass Avocado Board[1], và có thể tải về từ Kaggle[2].

Hình 2.1. Bộ dữ liệu được chia sẻ từ trang webhassavocadoboard.com.

Bộ dữ liệu có 18249 và 13 cột, thể hiện lịch sử của giá bơ, với các đặc trưng sau:

- Date (Ngày): Ngày thu thập dữ liệu (kiểu dữ liệu: datetime).
- AveragePrice (Giá trung bình): Giá trung bình của một quả bơ (đơn vị: USD).
- Total Volume (Tổng sản lượng): Số lượng quả bơ được bán ra trên toàn thị trường (đơn vị: số lượng quả).
- Type (Loại bơ): Phân loại sản phẩm Conventional (bơ thường) hoặc Organic (bơ hữu cơ).
- Region (Khu vực): Tên khu vực thu thập dữ liệu (ví dụ: California, New York).
- Year (Năm): Năm dữ liệu được thu thập.
- 4046, 4225, 4770 (Sản lượng dựa theo mã PLU): Số lượng bơ được bán dựa trên ba mã sản phẩm khác nhau tương ứng với kích thước (PLU4046, PLU4225, PLU4770).

	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region
0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conventional	2015	Albany
1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conventional	2015	Albany
2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	conventional	2015	Albany
3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	conventional	2015	Albany
4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	conventional	2015	Albany
...	...	...	...	...	...	...	...	...	...	...	...	...	...
18244	2018-02-04	1.63	17074.83	2046.96	1529.20	0.00	13498.67	13066.82	431.85	0.0	organic	2018	WestTexNewMexico
18245	2018-01-28	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8940.04	324.80	0.0	organic	2018	WestTexNewMexico
18246	2018-01-21	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9351.80	42.31	0.0	organic	2018	WestTexNewMexico
18247	2018-01-14	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10919.54	50.00	0.0	organic	2018	WestTexNewMexico
18248	2018-01-07	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11988.14	26.01	0.0	organic	2018	WestTexNewMexico

18249 rows × 13 columns

Hình 2.2. Bộ dữ liệu Avocado Price.

2.2 Tổng quan về bộ dữ liệu

```
df.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18249 entries, 0 to 18248
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Date          18249 non-null  object
 1   AveragePrice  18249 non-null  float64
 2   Total Volume  18249 non-null  float64
 3   4046          18249 non-null  float64
 4   4225          18249 non-null  float64
 5   4770          18249 non-null  float64
 6   Total Bags    18249 non-null  float64
 7   Small Bags    18249 non-null  float64
 8   Large Bags    18249 non-null  float64
 9   XLarge Bags   18249 non-null  float64
 10  type          18249 non-null  object
 11  year          18249 non-null  int64
 12  region        18249 non-null  object
dtypes: float64(9), int64(1), object(3)
memory usage: 1.8+ MB
```

Kết quả trả về từ hàm info() cho thấy bộ dữ liệu không có giá trị null. Các đặc trưng: Date, AveragePrice, Total, Volume, 4046, 4225, 4770, Total, Bags, Small, Bags, Large, Bags, XLarge, Bags, year thuộc kiểu dữ liệu số; còn các đặc trưng: type, region thuộc kiểu dữ liệu danh mục.

```
numeric_columns = ['AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags']
categorical_columns = ['type', 'region']
```

Thống kê mô tả về dữ liệu

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
AveragePrice	18249.0	1.405978	4.026766e-01	0.44	1.10	1.37	1.66	3.25
Total Volume	18249.0	850644.013009	3.453545e+06	84.56	10838.58	107376.76	432962.29	62505646.52
4046	18249.0	293008.424531	1.264989e+06	0.00	854.07	8645.30	111020.20	22743616.17
4225	18249.0	295154.568356	1.204120e+06	0.00	3008.78	29061.02	150206.86	20470572.61
4770	18249.0	22839.735993	1.074641e+05	0.00	0.00	184.99	6243.42	2546439.11
Total Bags	18249.0	239639.202060	9.862424e+05	0.00	5088.64	39743.83	110783.37	19373134.37
Small Bags	18249.0	182194.686696	7.461785e+05	0.00	2849.42	26362.82	83337.67	13384586.80
Large Bags	18249.0	54338.088145	2.439660e+05	0.00	127.47	2647.71	22029.25	5719096.61
XLarge Bags	18249.0	3106.426507	1.769289e+04	0.00	0.00	0.00	132.50	551693.65
year	18249.0	2016.147899	9.399385e-01	2015.00	2015.00	2016.00	2017.00	2018.00

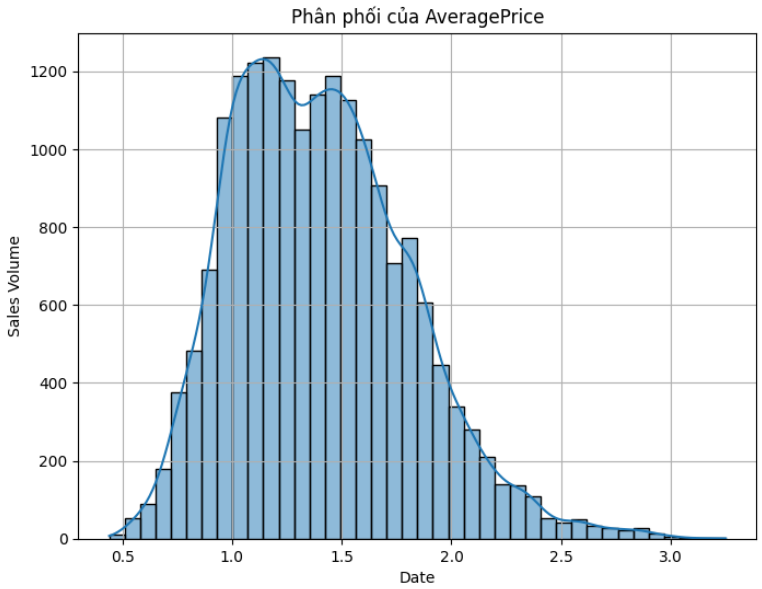
Hình 2.3. Bảng thống kê mô tả về dữ liệu.

3 Phân tích khám phá dữ liệu (EDA)

3.1 Các đặc trưng số học

Quan sát phân phối của thuộc tính AveragePrice (**Hình 3.1**, biểu đồ này không biểu diễn giá trị theo thời gian mà chỉ hiển thị tần suất của giá trị) ta thấy rằng thuộc tính này có phân phối gần chuẩn, hầu hết giá trị của AveragePrice tập trung trong khoảng từ 1.0 đến 2.0.

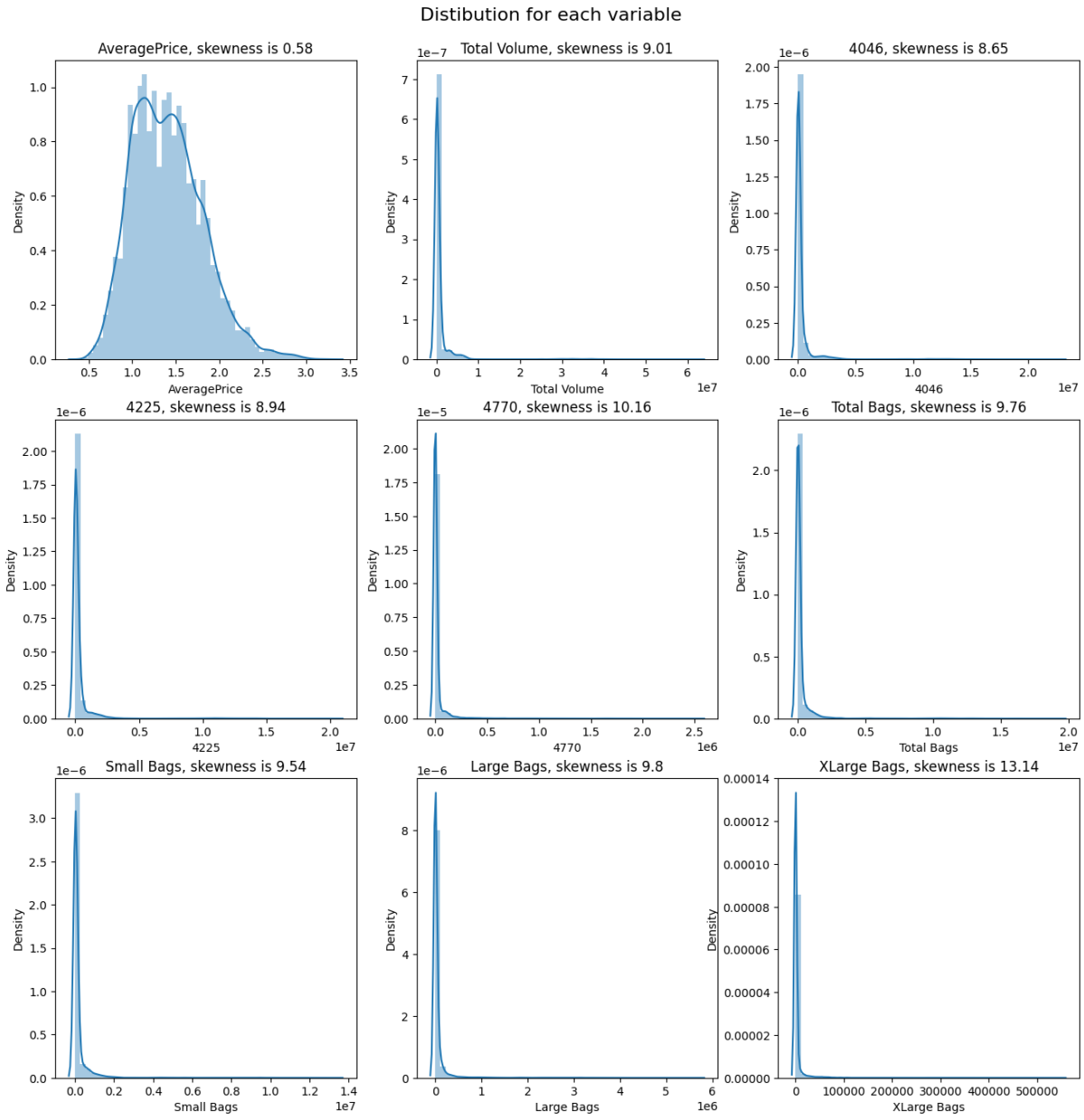
```
plt.figure(figsize=(8, 6))
sns.histplot(df1['AveragePrice'], kde=True, bins=40)
plt.title('Phân phối của AveragePrice')
plt.xlabel('Date')
plt.ylabel('Sales Volume')
plt.grid()
plt.show()
```



Hình 3.1. Phân phối của thuộc tính AveragePrice.

Phân phối của từng thuộc tính số học của dữ liệu (**Hình 3.2**), ngoài AveragePrice ra các thuộc tính còn lại có xu hướng lệch trái rất mạnh (positive skewness).

```
rows=3
cols=3
fig, axs = plt.subplots(rows, cols,figsize=(16,16))
fig.suptitle('Distibution for each variable',y=0.92, size=16)
axs = axs.flatten()
for i, data in enumerate(numeric_columns):
    sns.distplot(df[data], ax=axs[i])
    axs[i].set_title(data + ', skewness is '+str(round(df[data].skew(axis = 0, skipna =
True),2)))
```



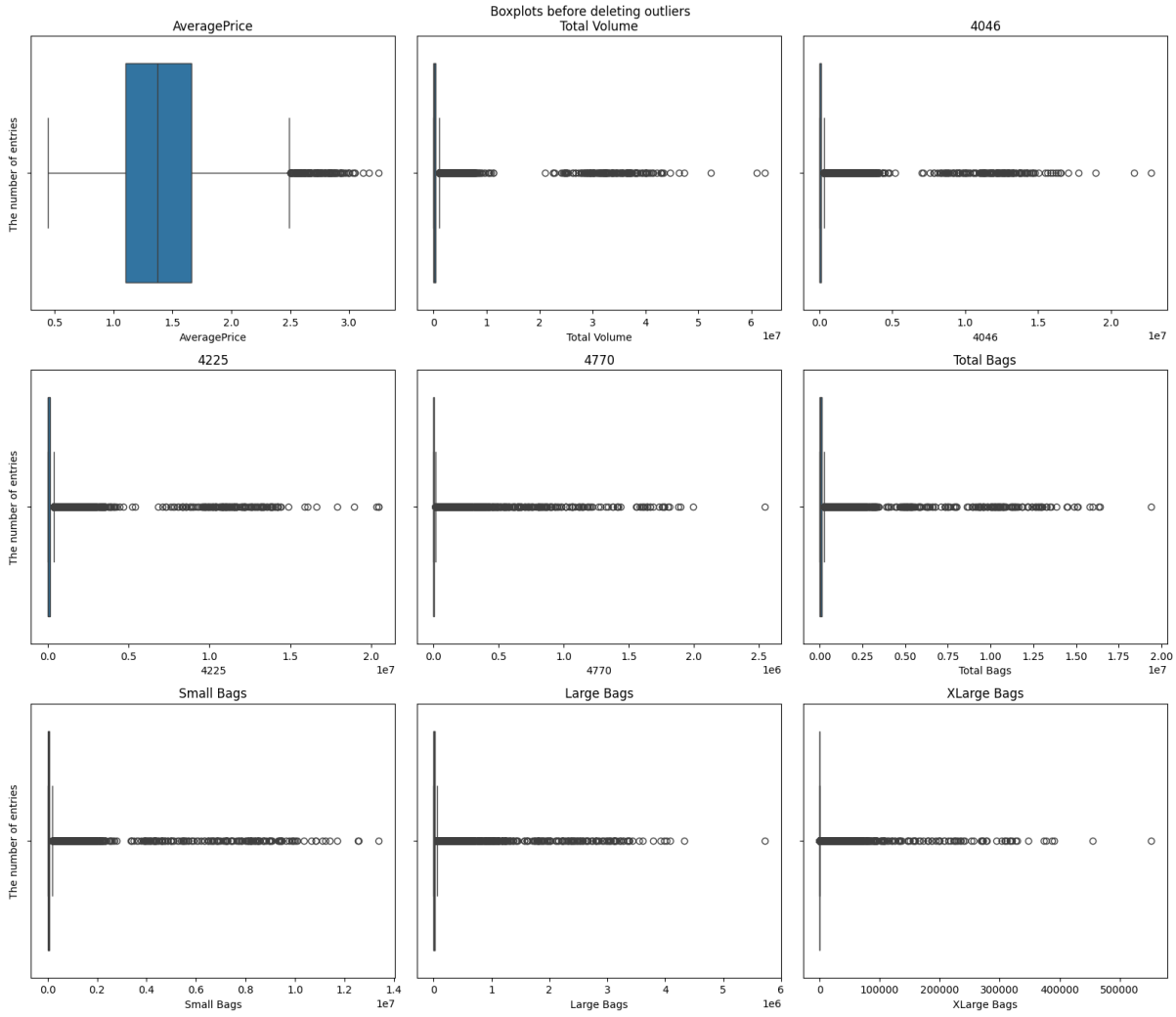
Hình 3.2. Phân phối của các thuộc tính số học.

Hầu hết các thuộc tính đều có skewness lớn (>8), đặc biệt là các thuộc tính liên quan đến sản lượng (volume). Điều này cho thấy cần xử lý các giá trị ngoại lai (outliers), áp

dụng biến đổi log hoặc chuẩn hoá dữ liệu để giúp giảm độ lệch. AveragePrice là biến mục tiêu và có phân phối tốt nhất.

Ta sẽ sử dụng biểu đồ boxplot để quan sát các giá trị ngoại lệ của dữ liệu:

```
rows=3
cols=3
fig, axs = plt.subplots(rows, cols, sharey=True, figsize=(16,14))
fig.suptitle('Boxplots before deleting outliers')
axs = axs.flatten()
for i, data in enumerate(numeric_columns):
    if i % 3 == 0:
        axs[i].set_ylabel('The number of entries')
    sns.boxplot( data=df[data], orient='h', ax=axs[i])
    axs[i].set_title(data)
plt.tight_layout()
```



Hình 3.3. Biểu đồ hộp các đặc trưng số học.

Dựa vào **Hình 3.3**, ta thấy rằng giống như đã phân tích ở phân phối của dữ liệu, các đặc trưng có ngoại lệ rất lớn.

3.2 Phân tích đặc trưng giá

Trước khi xử lý dữ liệu ta sẽ tiến hành đặt chỉ mục của DataFrame thành cột Date. Khi một DataFrame có cột datetime làm chỉ mục, chúng ta có thể dễ dàng biểu thị dữ liệu trên biểu đồ với hàm plot() của pandas.

```
df1 = df.copy()
df1 = df1.sort_values(by='Date')
df1['Date'] = pd.to_datetime(df['Date'])
df1.set_index('Date',inplace = True)
```

Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region
2015-01-04	1.75	27365.89	9307.34	3844.81	615.28	13598.46	13061.10	537.36	0.00	organic	2015	Southeast
2015-01-04	1.49	17723.17	1189.35	15628.27	0.00	905.55	905.55	0.00	0.00	organic	2015	Chicago
2015-01-04	1.68	2896.72	161.68	206.96	0.00	2528.08	2528.08	0.00	0.00	organic	2015	HarrisburgScranton
2015-01-04	1.52	54956.80	3013.04	35456.88	1561.70	14925.18	11264.80	3660.38	0.00	conventional	2015	Pittsburgh
2015-01-04	1.64	1505.12	1.27	1129.50	0.00	374.35	186.67	187.68	0.00	organic	2015	Boise
...	...	...	...	...	...	...	...	...	...	...	...	...
2018-03-25	1.36	908202.13	142681.06	463136.28	174975.75	127409.04	103579.41	22467.04	1362.59	conventional	2018	Chicago
2018-03-25	0.70	9010588.32	3999735.71	966589.50	30130.82	4014132.29	3398569.92	546409.74	69152.63	conventional	2018	SouthCentral
2018-03-25	1.42	163496.70	29253.30	5080.04	0.00	129163.36	109052.26	20111.10	0.00	organic	2018	SouthCentral
2018-03-25	1.70	190257.38	29644.09	70982.10	0.00	89631.19	89424.11	207.08	0.00	organic	2018	California
2018-03-25	1.34	1774776.77	63905.98	908653.71	843.45	801373.63	774634.09	23833.93	2905.61	conventional	2018	NewYork

18249 rows × 12 columns

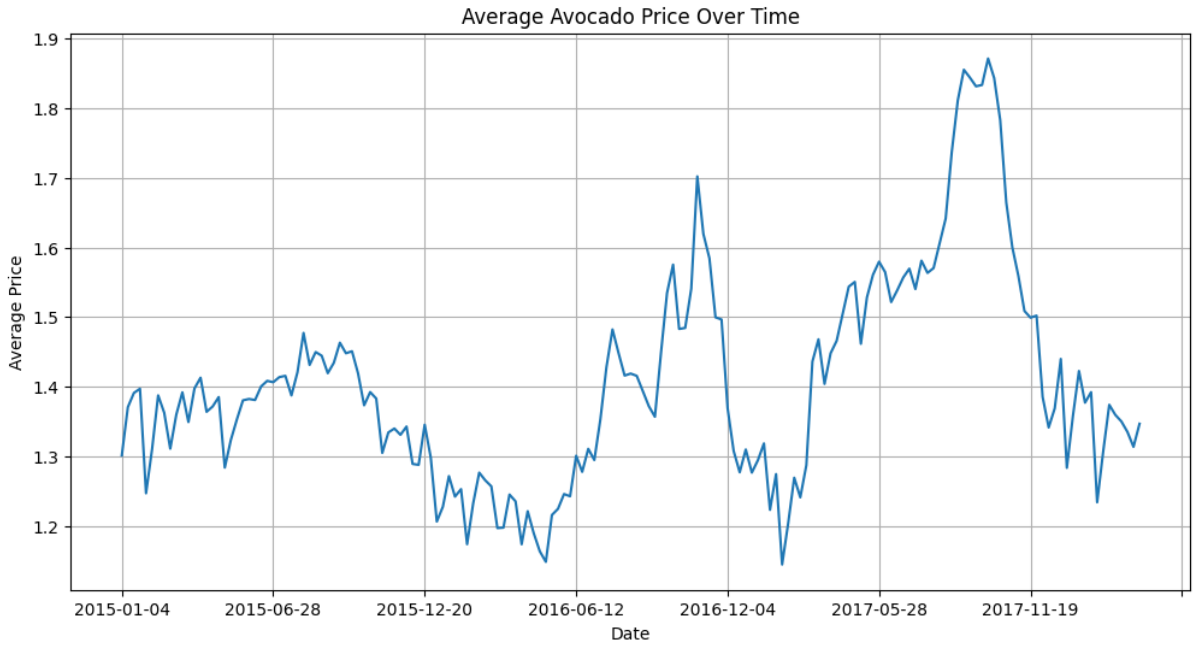
Hình 3.4. Dữ liệu sau khi thay đổi chỉ mục.

Việc thay đổi chỉ mục của dữ liệu giúp ta có thể trực quan thuộc tính AveragePrice theo thời gian:

```
avg_price = df.groupby('Date')['AveragePrice'].mean().reset_index()
avg_price.set_index('Date',inplace = True)
plt.figure(figsize=(12,6))
avg_price['AveragePrice'].plot()
plt.title('Average Avocado Price Over Time')
plt.xlabel('Date')
plt.ylabel('Average Price')
plt.grid()
plt.show()
```

Kết quả biến động của thuộc tính AveragePrice theo thời gian được thể hiện ở **Hình 3.5**, ta thấy xu hướng chung là từ năm 2015 đến cuối năm 2017, giá bơ có xu hướng dao động với sự gia tăng đáng kể vào năm 2017. Đỉnh giá cao nhất rơi vào khoảng giữa năm 2017, đạt gần 1.9 USD, sau đó giảm mạnh.

Biểu đồ cho thấy có thể tồn tại yếu tố mùa vụ (Seasonality) trong dữ liệu chuỗi thời gian, đặc biệt là sự tăng giá vào giữa năm và giảm vào cuối năm. Điều này có thể liên quan đến nguồn cung hoặc nhu cầu tăng cao trong các thời điểm nhất định (như mùa hè hoặc lễ hội), vì vậy các điểm giá tăng hoặc giảm đột ngột này có thể là ngoại lệ cần được xem xét trong mô hình dự đoán.

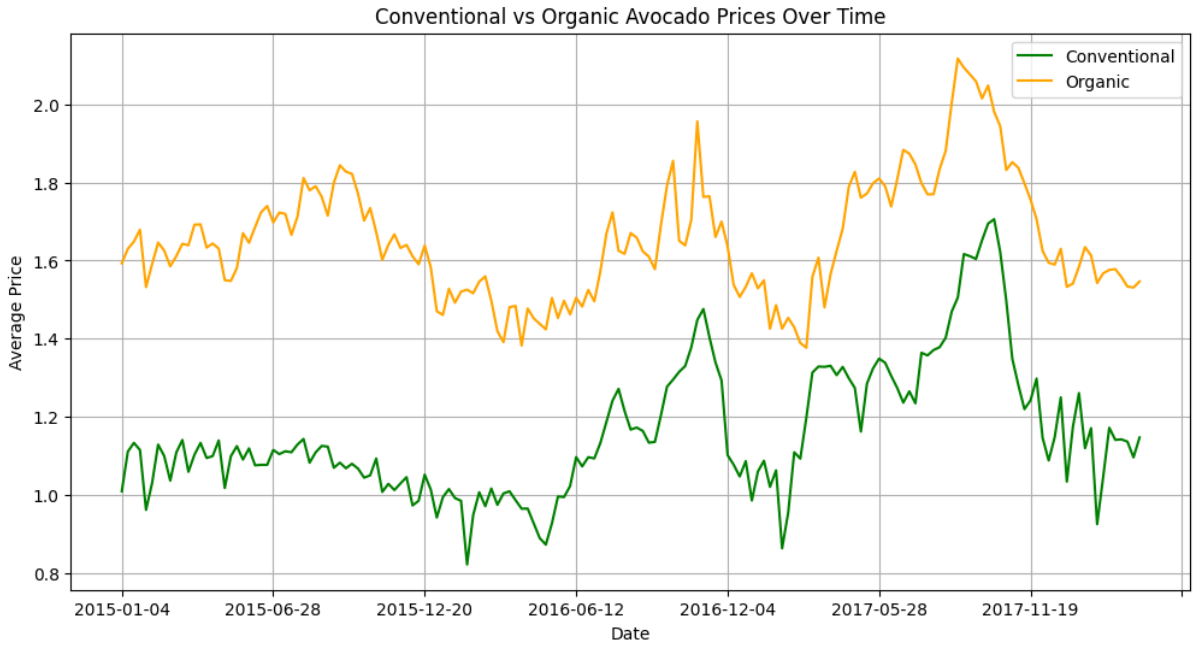


Hình 3.5. Biến động giá trung bình theo thời gian.

Dựa vào thuộc tính Type, ta thấy rằng dữ liệu được thu thập từ giá của hai loại bơ, đó là **Conventional** và **Organic** (tức là bơ thông thường và bơ hữu cơ). Biểu đồ ở **Hình 3.5** chỉ thể hiện biến động giá của cả hai loại bơ, vì vậy ta cần quan sát biến động giá theo từng loại bơ:

```
conv_price = df[df['type'] ==
'conventional'].groupby('Date')['AveragePrice'].mean().reset_index()
conv_price.set_index('Date',inplace = True)
org_price = df[df['type'] ==
'organic'].groupby('Date')['AveragePrice'].mean().reset_index()
org_price.set_index('Date',inplace = True)

# Plot price trends for conventional and organic avocados
plt.figure(figsize=(12, 6))
conv_price['AveragePrice'].plot(label='Conventional', color='green')
org_price['AveragePrice'].plot(label='Organic', color='orange')
plt.title('Conventional vs Organic Avocado Prices Over Time')
plt.xlabel('Date')
plt.ylabel('Average Price')
plt.legend()
plt.grid()
plt.show()
```

Kết quả biến động giá theo từng loại bơ được thể hiện ở **Hình 3.6**, ta thấy rằng xu hướng biến động giá và tính mùa vụ của cả hai loại bơ là như nhau. Ta thấy rằng giá trung bình của bơ thường (conventional) thấp hơn so với bơ hữu cơ (Organic), với giá bơ thường dao động trong khoảng 0.8 đến 1.6 còn bơ hữu cơ dao động trong khoảng 1.4 đến 2.0.



Hình 3.6. Biến động giá theo từng loại bơ theo thời gian.

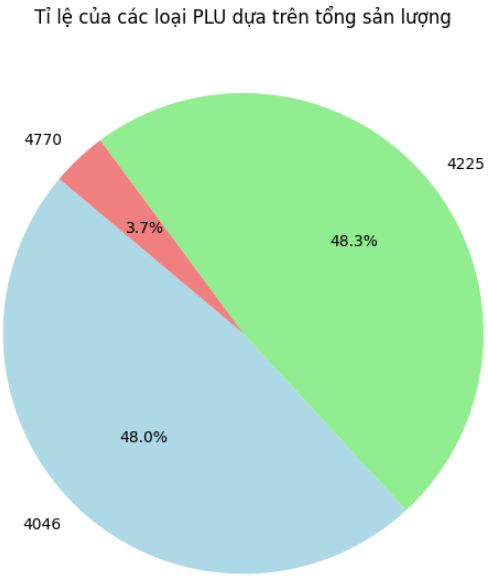
3.3 Phân tích các đặc trưng khối lượng giao dịch (volume).

Các cột dữ liệu 4046, 4225, 4770 là sản lượng của các mã sản phẩm PLU4046, PLU4225, PLU4770. PLU là mã tra cứu sản phẩm (Product Lookup codes). Dựa vào **Hình 3.7**, ta thấy rằng các mã sản phẩm này được phân loại dựa trên kích thước đóng gói.

Hình 3.7. Các loại sản phẩm.

Xem xét tỉ lệ của các loại mã sản phẩm dựa trên tổng sản lượng ở **Hình 3.8**, ta thấy rằng sản phẩm 4046 và 4225 chiếm đa số (48.0% và 48.3%), còn sản phẩm 4770 có sản lượng ít hơn rất nhiều so với hai loại sản phẩm trên.

```
total_sales = df[['4046', '4225', '4770']].sum()
# Plot pie chart for total sales volume
plt.figure(figsize=(8, 6))
plt.pie(total_sales, labels=total_sales.index, autopct='%1.1f%%', startangle=140,
colors=['lightblue', 'lightgreen', 'lightcoral'])
plt.title('Tỉ lệ của các loại PLU dựa trên tổng sản lượng')
plt.tight_layout()
plt.show()
```



Hình 3.8. Tỉ lệ các loại sản phẩm dựa trên sản lượng.

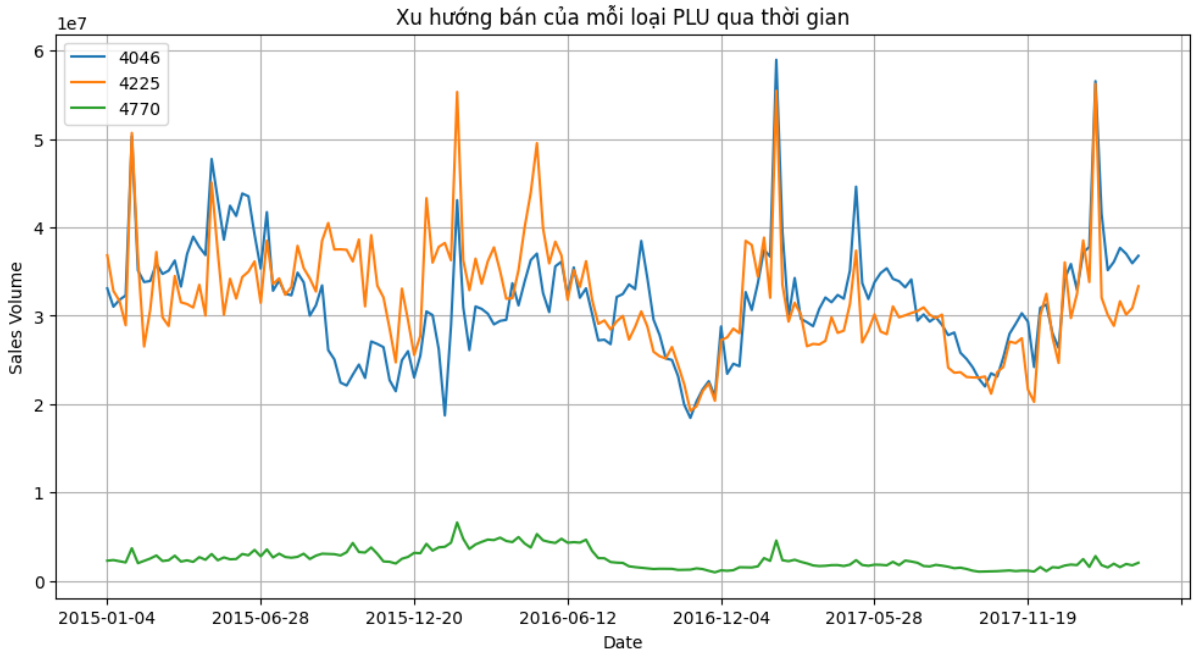
Điều này cho thấy PLU 4046 và 4225 là hai sản phẩm chủ lực, chúng chiếm phần lớn thị phần và có mức độ tiêu thụ tương đương nhau. Nguyên nhân dẫn đến sự khác biệt về sản lượng giữa các mã PLU có thể là do sở thích của khách hàng, vì sản phẩm 4770 có kích cỡ lớn hơn hai loại còn lại và khách có xu hướng mua loại vừa và nhỏ hơn.

Ta sẽ xem xét qua xu hướng bán của mỗi loại sản phẩm theo thời gian:

```
sales_trends = df.groupby('Date')[['4046', '4225', '4770']].sum()

plt.figure(figsize=(12, 6))
for column in sales_trends.columns:
    sales_trends[column].plot(label=column)

plt.title('Xu hướng bán của mỗi loại PLU qua thời gian')
plt.xlabel('Date')
plt.ylabel('Sales Volume')
plt.legend()
plt.grid()
plt.show()
```

Dựa vào **Hình 3.9**, ta thấy xu hướng chung là cả hai mã sản phẩm PLU 4046 và 4225 đều có xu hướng bán hàng tương đối ổn định trong suốt thời gian theo dõi, và có một số biến động theo mùa. Lượng bán hàng của hai mã này thường dao động trong một khoảng nhất định, không có sự tăng trưởng hoặc giảm mạnh. PLU 4770 có lượng bán hàng thấp hơn đáng kể so với hai mã còn lại và có xu hướng giảm dần theo thời gian.

So sánh với các biểu đồ xu hướng giá (**Hình 3.5** và **Hình 3.6**), sản lượng có thể có tương quan nghịch với giá trung bình. Vì quan sát ta thấy rằng vào các thời điểm các loại sản phẩm có sản lượng cao thì giá trung bình có xu hướng giảm ở thời điểm đó. Tương quan nghịch giữa sản lượng và giá sản phẩm có thể dễ dàng lý giải bằng kinh tế học cơ bản, rằng khi giá của một sản phẩm tăng lên, nhu cầu về sản phẩm đó thường giảm đi và ngược lại. Điều này có nghĩa là sản lượng bán ra của sản phẩm sẽ có xu hướng giảm khi giá tăng và tăng khi giá giảm.



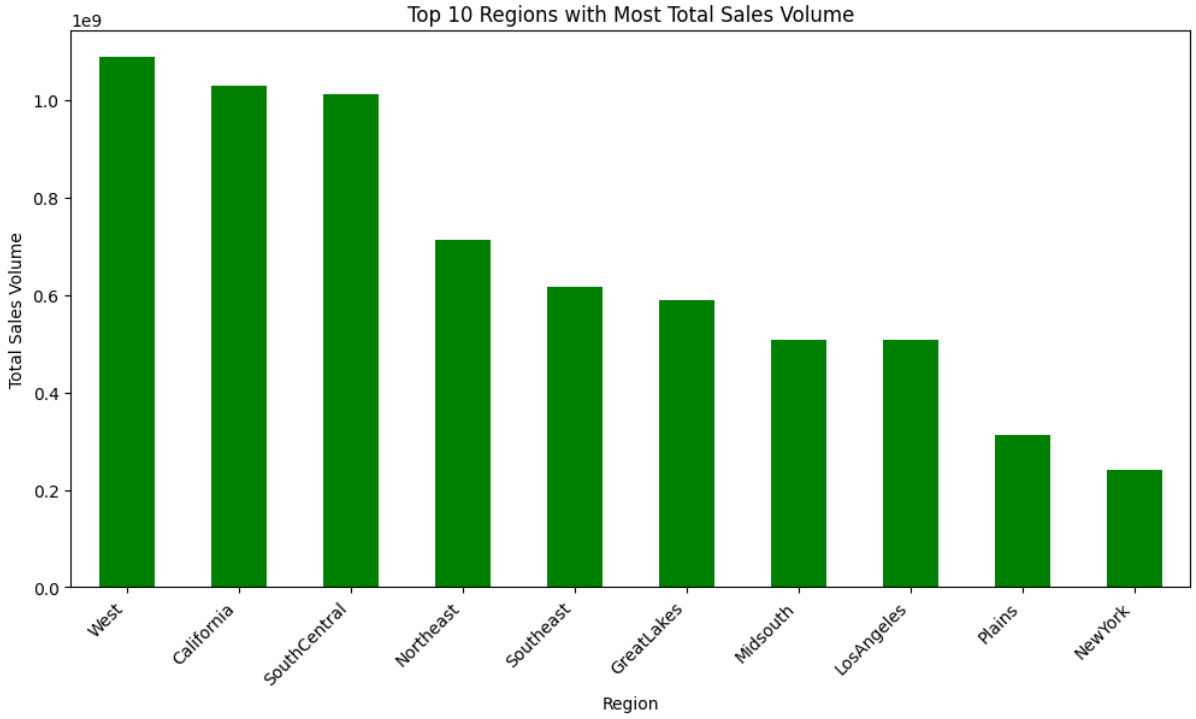
Hình 3.9. Sản lượng của các loại sản phẩm theo thời gian.

Thuộc tính Total Volume là tổng sản lượng của tất cả các mã sản phẩm, ta sẽ phân tích các khu vực có doanh số bán hàng tốt nhất:

```
region_sales = df[df['region'] != 'TotalUS'].groupby('region')['Total Volume'].sum()
top_10_regions = region_sales.sort_values(ascending=False).head(10)
plt.figure(figsize=(12, 6))
top_10_regions.plot(kind='bar', color='green')
plt.title('Top 10 Regions with Most Total Sales Volume')
plt.xlabel('Region')
plt.ylabel('Total Sales Volume')
plt.xticks(rotation=45, ha='right')
plt.show()
```

Kết quả (Hình 3.10) cho thấy vùng bờ Tây Hoa Kỳ (West) có tổng sản lượng cao nhất. Vùng này bao gồm các tiểu bang California, Oregon và Washington, với California là tiểu bang đông dân nhất Hoa Kỳ.

California cũng đã chiếm phần lớn sản lượng của vùng tây Hoa Kỳ, khi tiểu bang này đứng thứ hai trong tổng sản lượng (bộ dữ liệu này ghi nhận dữ liệu của cả vùng địa lý lẫn tiểu bang cho nên ta thấy rằng dù West đã bao gồm California nhưng bộ dữ liệu vẫn có dữ liệu sản lượng của California). Tiểu bang Los Angeles và New York cũng là những nơi có sản lượng cao nhất.



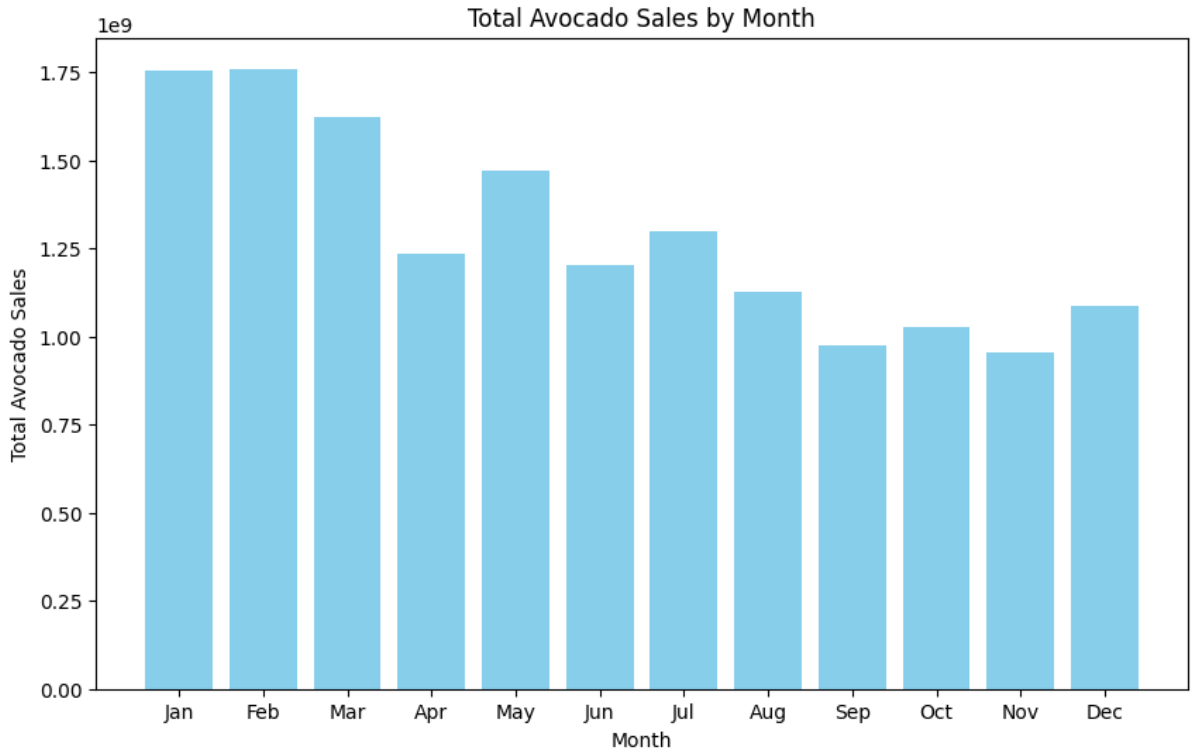
Hình 3.10. Những vùng có tổng sản lượng cao nhất.

Để có thể dễ dàng hình dung về mức độ sản tổng sản lượng của từng vùng, ta có thể trực quan hoá dữ liệu sử dụng biểu đồ Choropleth, phần này sẽ được trình bày ở mục **3.6**.

Phân tích tính mùa vụ đối với tổng sản lượng, ta sẽ xem xét tổng sản lượng theo các tháng trong năm:

```
avocado_df = df.copy()
avocado_df['Date'] = pd.to_datetime(avocado_df['Date'])
avocado_df['Month'] = avocado_df['Date'].dt.month
monthly_sales = avocado_df.groupby('Month')['Total Volume'].sum().reset_index()
monthly_sales = monthly_sales.sort_values(by='Total Volume', ascending=False)
monthly_sales = monthly_sales.sort_values(by='Month')
plt.figure(figsize=(10, 6))
plt.bar(monthly_sales['Month'], monthly_sales['Total Volume'], color='skyblue')
plt.xlabel('Month')
plt.ylabel('Total Avocado Sales')
plt.title('Total Avocado Sales by Month')
plt.xticks(monthly_sales['Month'], ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug',
'Sep', 'Oct', 'Nov', 'Dec'])
plt.show()
```

Biểu đồ cột **Hình 3.11** cho ta thấy doanh số bán bơ theo tháng trong tất cả các năm từ 2015 đến 2017. Doanh số bán hàng cao nhất vào tháng 2, tiếp theo là tháng 1 và 3, điều này cho thấy có một nhu cầu tiêu thụ bơ cao vào đầu năm. Doanh số giảm dần từ tháng 4 đến tháng 9, đạt mức thấp nhất vào khoảng tháng 9. Sau đó, doanh số bắt đầu tăng trở lại vào cuối năm. Dựa vào biểu đồ, có thể rút ra kết luận rằng doanh số có tính mùa vụ rõ rệt. Nhu cầu tiêu thụ bơ cao nhất vào đầu năm và giảm dần trong suốt mùa hè.



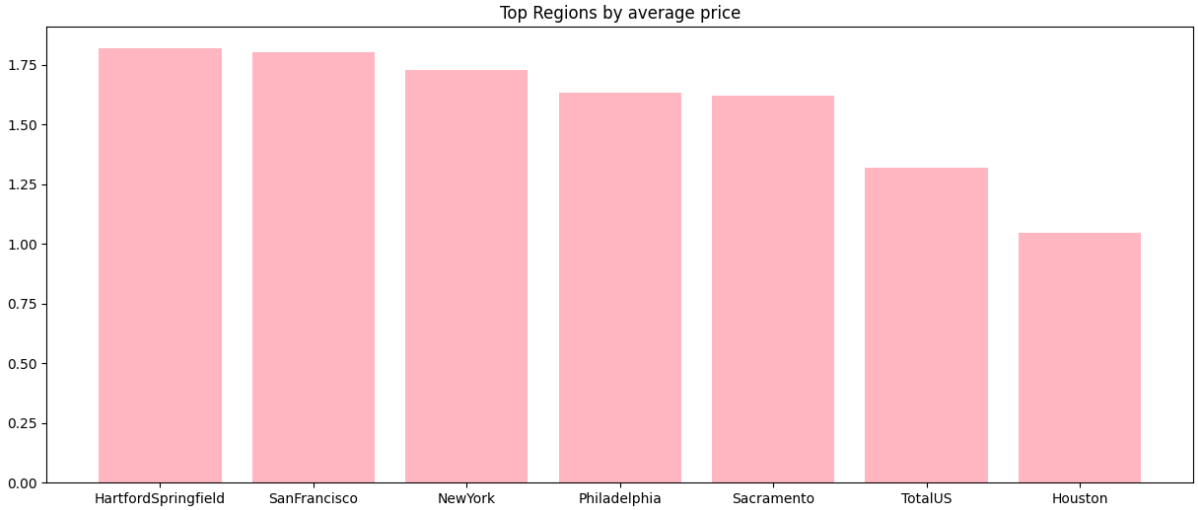
Hình 3.11. Doanh số bán hàng dựa theo tháng.

3.4 Phân tích đặc trưng khu vực

Phân tích khu vực trong dữ liệu chuỗi thời gian là quan trọng giúp chúng ta hiểu rõ hơn về sự khác biệt và tương đồng trong xu hướng, mùa vụ, và các biến động khác nhau giữa các địa điểm hoặc khu vực. Dựa vào thuộc tính Region của dữ liệu, ta có thể so sánh giá bơ giữa các khu vực với nhau:

```
top_5_avg_price = df.groupby('region')['AveragePrice'].mean().reset_index()
top_5_avg_price = top_5_avg_price.sort_values('AveragePrice', ascending=False)
cheapest_avg_price = top_5_avg_price.tail(1).reset_index()
avg_US_price = top_5_avg_price[top_5_avg_price.region=='TotalUS'][['region',
'AveragePrice']]
top_5_avg_price = top_5_avg_price.head()
combined_avg_price = pd.concat([top_5_avg_price, avg_US_price, cheapest_avg_price])
combined_avg_price = combined_avg_price.reset_index(drop=True)
combined_avg_price = combined_avg_price.drop('index', axis=1)
```

Các vùng Hartford–Springfield, San Francisco, New York, Philadenphia và Sacramento có giá bơ trung bình cao hơn so với các vùng còn lại. So với nơi có giá bơ thấp nhất là Houston thì có sự chênh lệch khá lớn. Điều này có thể là do những nơi này là những thành phố lớn và có mật độ dân cư cao và có chi phí sinh hoạt cao hơn. Các thành phố lớn thường nằm xa các vùng trồng bơ chính, có khả năng dẫn đến chi phí vận chuyển tăng cao.

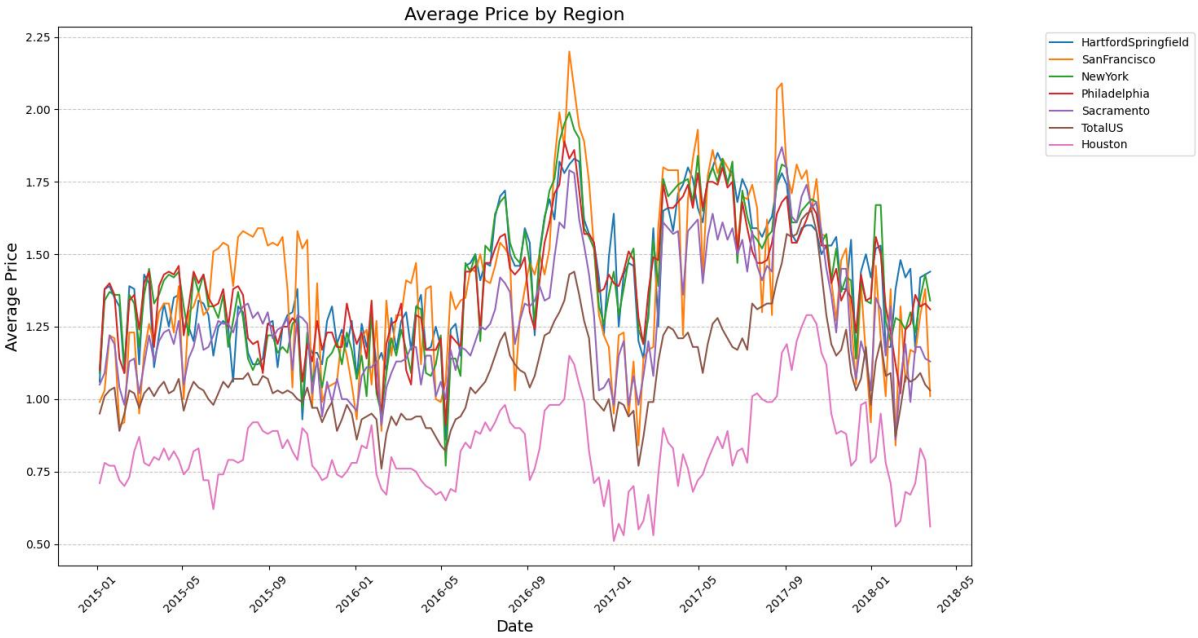


Hình 3.12. Những nơi có giá cao nhất, giá trung bình, và giá thấp nhất.

Biến động giá bơ theo thời gian của những vùng có giá trung bình cao nhất, so với giá trung bình (trên toàn bộ nước Mỹ) và giá thấp nhất:

```
regions = combined_avg_price['region'].unique()
plt.figure(figsize=(15, 8))

for region in regions:
    region_data = df1[df1['region'] == region]
    plt.plot(region_data['Date'], region_data['AveragePrice'], label=region)

plt.title('Average Price by Region', fontsize=16)
plt.xlabel('Date', fontsize=14)
plt.ylabel('Average Price', fontsize=14)

plt.legend(loc='upper right', bbox_to_anchor=(1.25, 1))
plt.xticks(rotation=45, fontsize=10)
plt.grid()
plt.tight_layout()
plt.show()
```

Tương tự như các phân tích trước, biến động giá bơ của các khu vực cũng có tính mùa vụ. Giá bơ có xu hướng tăng ở giữa năm và giảm ở đầu năm và cuối năm ở các khu vực.



Hình 3.13. Biến động giá bơ của các khu vực.

3.5 Phân tích đặc trưng doanh thu

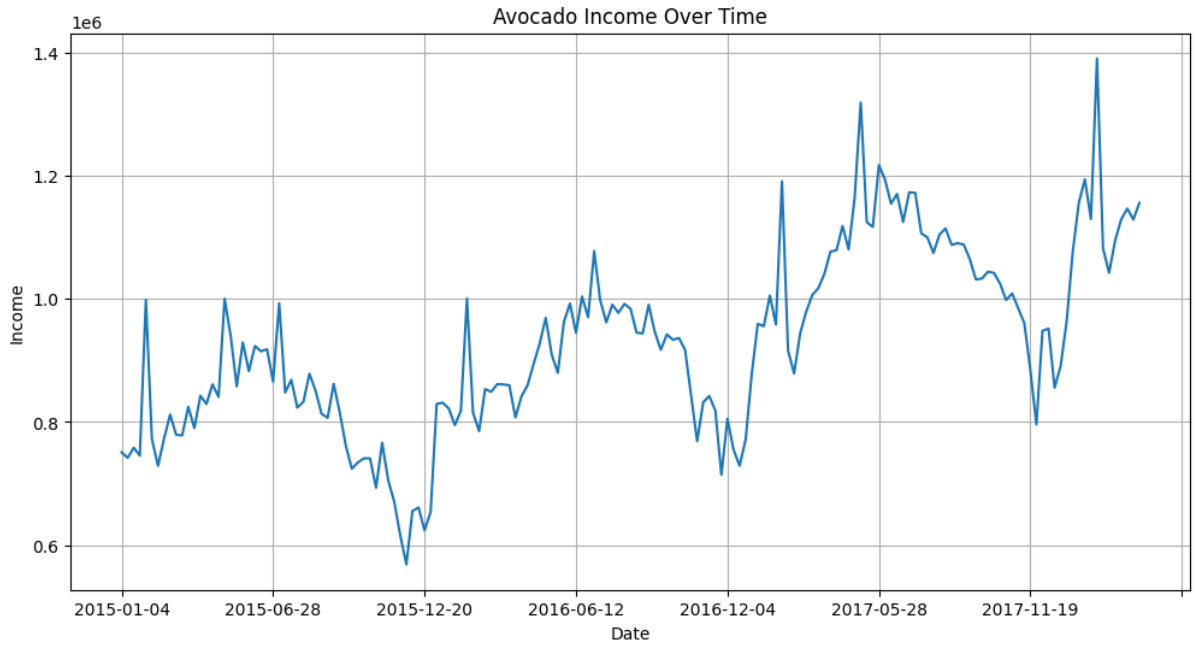
Bộ dữ liệu này chỉ có thuộc tính giá và doanh số bán hàng, để có thể hiểu rõ hơn về dữ liệu ta sẽ khởi tạo một thuộc tính mới là doanh thu bán hàng. Ta có công thức để tính doanh thu là: Doanh thu = Giá x Doanh số. Tạo một cột mới là Income, Income sẽ có đơn vị là USD giống như AveragePrice:

```
df['Income'] = df['Total Volume']*df['AveragePrice']
```

Trực quan biến động doanh thu theo thời gian:

```
avg_income = df.groupby('Date')['Income'].mean().reset_index()
avg_income.set_index('Date',inplace = True)
plt.figure(figsize=(12,6))
avg_income['Income'].plot()
plt.title('Avocado Income Over Time')
plt.xlabel('Date')
plt.ylabel('Income')
plt.grid()
plt.show()
```

Biểu đồ ở **Hình 3.14**, giai đoạn từ đầu năm đến khoảng tháng 9 là khoảng thời gian doanh thu bán hàng tốt nhất trong năm. Giai đoạn sau tháng 9 đến cuối năm là thời điểm doanh thu sụt giảm, tháng 9 cũng là thời điểm giá bơ cao nhất trong năm. Doanh thu tăng trở lại sau khi giá giảm dần từ giai đoạn cuối năm (**Hình 3.5**).



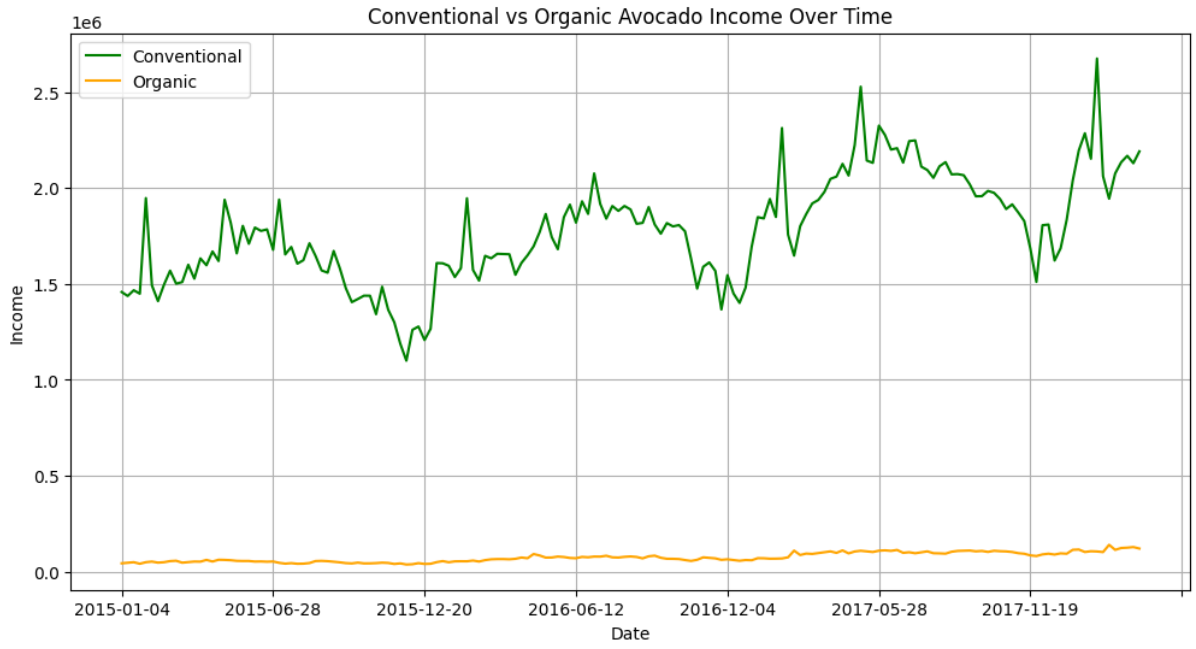
Hình 3.14. Doanh thu từ năm đầu năm 2015 đến cuối năm 2017.

Biến động doanh thu theo loại bơ (bơ thường và bơ hữu cơ):

```
conv_price = df[df['type'] ==
'conventional'].groupby('Date')['Income'].mean().reset_index()
conv_price.set_index('Date',inplace = True)
org_price = df[df['type'] == 'organic'].groupby('Date')['Income'].mean().reset_index()
org_price.set_index('Date',inplace = True)
plt.figure(figsize=(12, 6))
conv_price['Income'].plot(label='Conventional', color='green')
org_price['Income'].plot(label='Organic', color='orange')
plt.title('Conventional vs Organic Avocado Income Over Time')
plt.xlabel('Date'), plt.ylabel('Income'), plt.legend(), plt.grid(), plt.show()
```

So sánh doanh thu giữa hai loại bơ (**Hình 3.15**) ta thấy có sự chênh lệch rõ rệt. Bơ thông thường (Conventional) có doanh thu cao hơn rất nhiều so với bơ hữu cơ trong toàn bộ giai đoạn và có sự biến động rõ rệt theo thời gian, đặc biệt trong các khoảng thời gian giữa năm. Bơ hữu cơ (Organic) có doanh thu thấp hơn rất nhiều (thường dưới 0.2 triệu USD). So với bơ thường, bơ hữu cơ có doanh thu ổn định, ít dao động mạnh hơn. Bơ thường biến động lớn hơn, phản ánh sự phụ thuộc vào mùa vụ hoặc các yếu tố thị trường và có xu hướng giảm sau các giai đoạn cao điểm.

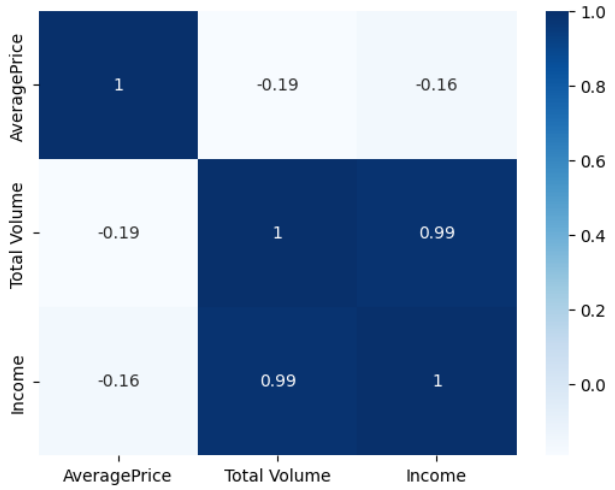
Có thể nhận xét rằng dù giá của cả hai loại bơ đều có yếu tố mùa vụ (**Hình 3.6**) nhưng đối với doanh thu bán hàng thì chỉ có bơ thường mang yếu tố mùa vụ. Có thể là do bơ thường có giá rẻ hơn so với bơ hữu cơ nên bơ thường sẽ có xu hướng được chọn mua nhiều hơn nên nó bị ảnh hưởng bởi yếu tố thị trường.



Hình 3.15. Biến động doanh thu theo thời gian của hai loại bơ.

Phân tích sự tương quan giữa AveragePrice, Total Volume và Income:

```
corr = df[['AveragePrice', 'Total Volume', 'Income']].corr(method = 'pearson')
sns.heatmap(corr, annot=True, cmap='Blues')
```



Hình 3.16. Tương quan giữa AveragePrice, Total Volume và Income.

3.6 Trực quan với Choropleth maps

Choropleth Map (Bản đồ Choropleth) là một loại bản đồ thể hiện thông tin dựa trên các khu vực địa lý. Nó sử dụng màu sắc hoặc mẫu để biểu diễn giá trị của một thuộc tính hoặc chỉ số trên các khu vực địa lý. Ví dụ, thể hiện thị phần của một sản phẩm tại một khu vực địa lí bằng cách căn cứ vào độ đậm hay nhạt của màu sắc trên bản đồ của khu vực đó.

Tạo danh sách các khu vực, giá trung bình, doanh thu và doanh số:

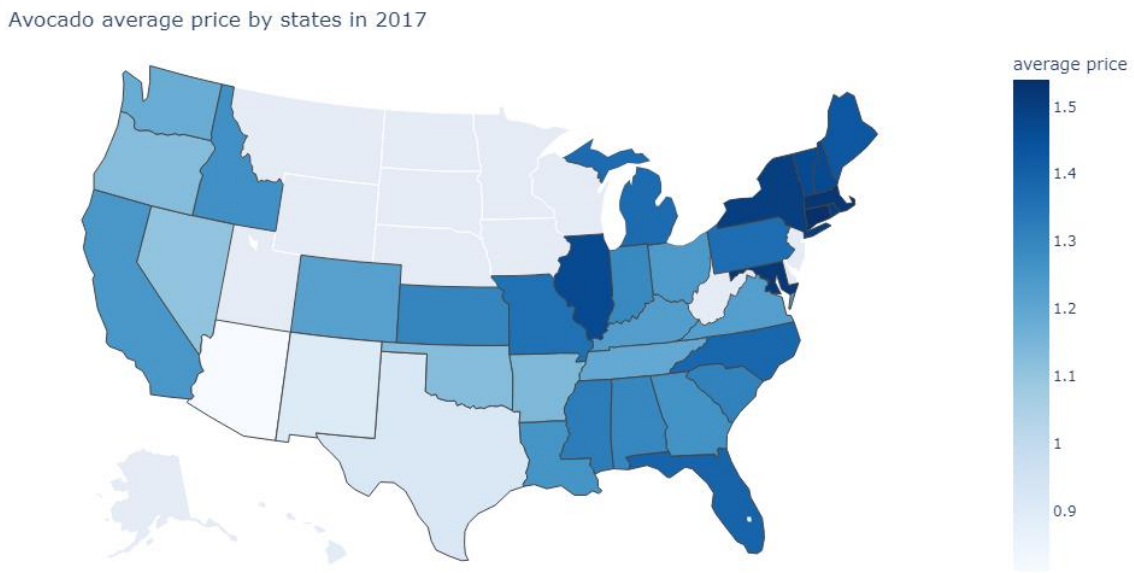
```
regions = {
```

```
    'Albany': ['NY'],
    'Atlanta': ['GA'],
    'BaltimoreWashington': ['MD', 'DC'],
    'Boise': ['ID'],
    'Boston': ['MA'],
    'BuffaloRochester': ['NY'],
    'California': ['CA'],
    'Charlotte': ['NC'],
    'Chicago': ['IL'],
    'CincinnatiDayton': ['OH'],
    'Columbus': ['OH'],
    'DallasFtWorth': ['TX'],
    'Denver': ['CO'],
    'Detroit': ['MI'],
    'GrandRapids': ['MI'],
    'GreatLakes': ['IL', 'IN', 'MI', 'OH', 'PA'],
    'HarrisburgScranton': ['PA'],
    'HartfordSpringfield': ['CT', 'MA'],
    'Houston': ['TX'],
    'Indianapolis': ['IN'],
    'Jacksonville': ['FL'],
    'LasVegas': ['NV'],
    'LosAngeles': ['CA'],
    'Louisville': ['KY'],
    'MiamiFtLauderdale': ['FL'],
    'Midsouth': ['TN', 'AR', 'MS'],
    'Nashville': ['TN'],
    'NewOrleansMobile': ['LA', 'AL'],
    'NewYork': ['NY'],
    'Northeast': ['CT', 'MA', 'NH', 'NY', 'RI', 'VT'],
    'NorthernNewEngland': ['ME', 'NH', 'VT'],
    'Orlando': ['FL'],
    'Philadelphia': ['PA'],
    'PhoenixTucson': ['AZ'],
    'Pittsburgh': ['PA'],
    'Plains': ['KS', 'OK'],
    'Portland': ['OR'],
    'RaleighGreensboro': ['NC'],
    'RichmondNorfolk': ['VA'],
    'Roanoke': ['VA'],
    'Sacramento': ['CA'],
    'SanDiego': ['CA'],
    'SanFrancisco': ['CA'],
    'Seattle': ['WA'],
    'SouthCarolina': ['SC'],
    'SouthCentral': ['TX', 'OK', 'AR'],
    'Southeast': ['AL', 'FL', 'GA', 'SC'],
    'Spokane': ['WA'],
    'StLouis': ['MO'],
    'Syracuse': ['NY'],
    'Tampa': ['FL'],
    'West': ['CA', 'WA', 'OR'],
    'WestTexNewMexico': ['TX', 'NM']
}
states_price = dict()
states_volume = dict()
states_income = dict()

for i in regions:
    avg_price = df[(df.region == i) & (df.year == 2017)]['AveragePrice'].mean()
    total_volume = df[(df.region == i) & (df.year == 2017)]['Total Volume'].mean()
    income = df[(df.region == i) & (df.year == 2017)]['Income'].mean()

    for state in regions.get(i):
        if states_price.get(state):
            states_price[state] = (avg_price + states_price[state]) / 2
```

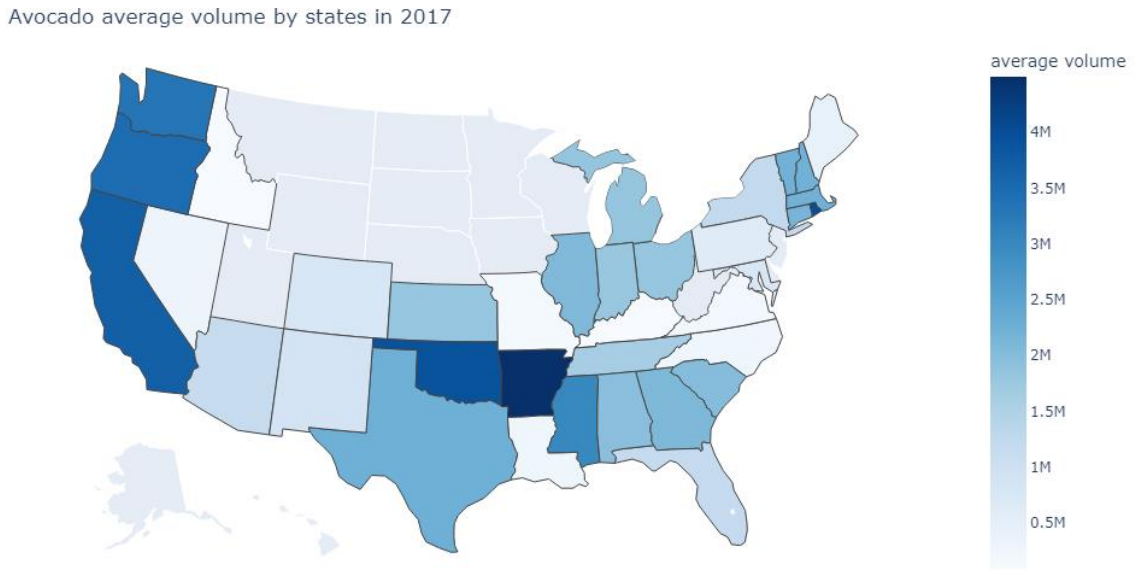
```
        else:
            states_price[state] = avg_price
        if states_volume.get(state):
            states_volume[state] = (total_volume + states_volume[state]) / 2
        else:
            states_volume[state] = total_volume
        if states_income.get(state):
            states_income[state] = (income + states_income[state]) / 2
        else:
            states_income[state] = income

locations = list(states_price.keys())
state_avg_price = list(states_price.values())
state_avg_volume = list(states_volume.values())
state_avg_income = list(states_income.values())
```

Trực quan biểu đồ choropleth với thư viện plotly:

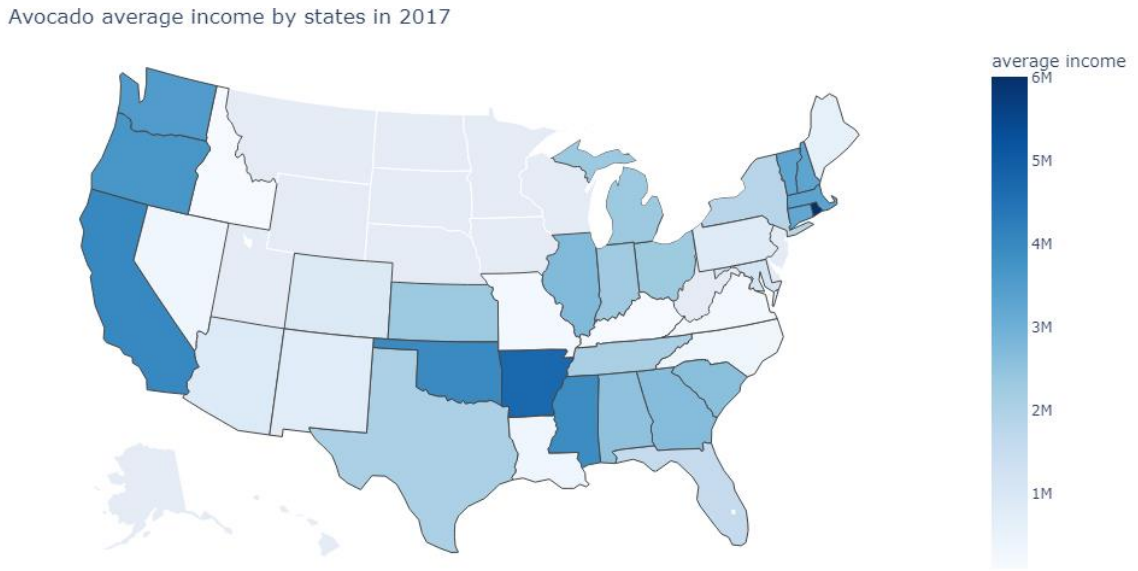
```
import plotly.express as px
fig = px.choropleth(locations=locations, locationmode="USA-states", color=state_avg_price,
scope="usa", color_continuous_scale='blues',
                    title='Avocado average price by states in 2017',
                    labels={'color':'average price'})
fig.update_layout(margin={"r":25,"t":25,"l":25,"b":25}, width=1000, height=500)
fig.show()
fig = px.choropleth(locations=locations, locationmode="USA-states", color=state_avg_volume,
scope="usa", color_continuous_scale='blues',
                    title='Avocado average volume by states in 2017',
                    labels={'color':'average volume'})
fig.show()
fig = px.choropleth(locations=locations, locationmode="USA-states", color=state_avg_income,
scope="usa", color_continuous_scale='blues',
                    title='Avocado average income by states in 2017',
                    labels={'color':'average income'})
fig.show()
```



Hình 3.17. Giá trung bình theo khu vực trong năm 2017.



Hình 3.18. Lượng bơ trung bình bán được theo khu vực năm 2017.



Hình 3.19. Doanh thu trung bình theo khư vực năm 2017

4 Phân tích chuỗi thời gian

Dữ liệu chuỗi thời gian có một số đặc điểm phân biệt nó với các loại dữ liệu khác:

- Xu hướng (Trend): Đây là sự di chuyển hoặc hướng đi dài hạn trong dữ liệu. Ví dụ, sự gia tăng chung trong doanh số bán hàng của một công ty qua nhiều năm.
- Tính mùa vụ (Seasonality): Đây là các tính lặp lại theo khoảng thời gian cố định, thành phần chỉ ra các xu hướng theo mùa, theo tháng, theo quý chẳng hạn như doanh số bán bơ thường tăng cao hơn vào đầu năm và giảm dần ở giữa năm.

- Mô hình chu kỳ (Cyclic Patterns): Khác với tính thời vụ, các mô hình chu kỳ không có chu kỳ cố định, nó có sự vận động trong khoảng thời gian dài hơn (nhiều năm). Chúng có thể bị ảnh hưởng bởi các chu kỳ kinh tế hoặc các yếu tố khác.
- Thành phần bất thường (Irregular Components): Hay còn gọi là nhiễu trắng (white noise) đây là các biến đổi ngẫu nhiên hoặc không thể dự đoán trong dữ liệu, nó chỉ ra sự bất thường của các điểm dữ liệu.

Để có thể hiểu được dữ liệu dạng chuỗi thời gian, ta có thể phân tách dữ liệu thành các thành phần đơn giản hơn, rồi từ đó có thể đi sâu vào phân tích xu hướng hay khuôn mẫu của các thành phần. Quá trình này gọi là phân rã chuỗi thời gian (**Decomposition**).

4.1 Resampling

Trước khi thực hiện decompose, ta sẽ thực hiện bước resample dữ liệu. Resampling dữ liệu trong chuỗi thời gian là một kỹ thuật quan trọng khi ta muốn thay đổi tần suất của dữ liệu, ví dụ từ ngày sang tuần. Bên cạnh đó đối với bộ dữ liệu này, giá và lượng hàng bán được được thu thập từ rất nhiều khu vực, ta có thể gom nhóm lại các giá trị AveragePrice và các thuộc tính Total Volume, 4046, 4225, 4770… để đơn giản hóa và làm rõ xu hướng tổng quát trong dữ liệu.

```
df1 = df.copy(deep = True)
df1['Date'] = pd.to_datetime(df1['Date'])
df1.set_index('Date',inplace = True)
df1 = df1.resample('W').agg({
    'AveragePrice': 'mean', 'Total Volume': 'sum', '4046': 'sum', '4225': 'sum', '4770': 
'sum', 'Total Bags': 'sum', 'Small Bags': 'sum', 'Large Bags': 'sum', 'XLarge Bags': 'sum'
})
```

Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags
2015-01-04	1.30	84674337.20	33098258.74	36851038.43	2278818.64	12446221.39	9910008.90	2485655.90	50556.59
2015-01-11	1.37	78555807.24	31024164.99	32808429.84	2349167.45	12374044.96	10352936.42	1977832.00	43276.54
2015-01-18	1.39	78388784.08	31802706.86	31662041.93	2208250.55	12715784.74	10548854.11	2145586.13	21344.50
2015-01-25	1.40	76466281.07	32305132.87	28929122.37	2078378.86	13153646.97	10877316.95	2252815.92	23514.10
2015-02-01	1.25	119453235.25	50292988.96	50696230.48	3687489.31	14776526.50	11576833.65	3073989.39	125703.46
...	...	...	...	...	...	...	...	...	...
2018-02-25	1.36	109231443.74	36101449.62	28851825.03	1937291.35	42338670.14	30234652.79	11314895.65	789121.70
2018-03-04	1.35	111844929.76	37680590.60	31640490.62	1546057.53	40975831.43	30843878.06	9433824.95	698128.42
2018-03-11	1.34	111465199.18	37014593.28	30127345.96	1906686.20	42415959.60	32760303.59	8948685.56	706970.45
2018-03-18	1.31	113673979.18	35938794.17	30868435.84	1754038.39	45111934.39	33306207.85	11097219.69	708506.85
2018-03-25	1.35	118268952.10	36785160.31	33340482.62	2044786.25	46095895.52	35511157.02	9779256.37	805482.13

169 rows × 9 columns

Hình 4.1. Dữ liệu sau khi resampling.

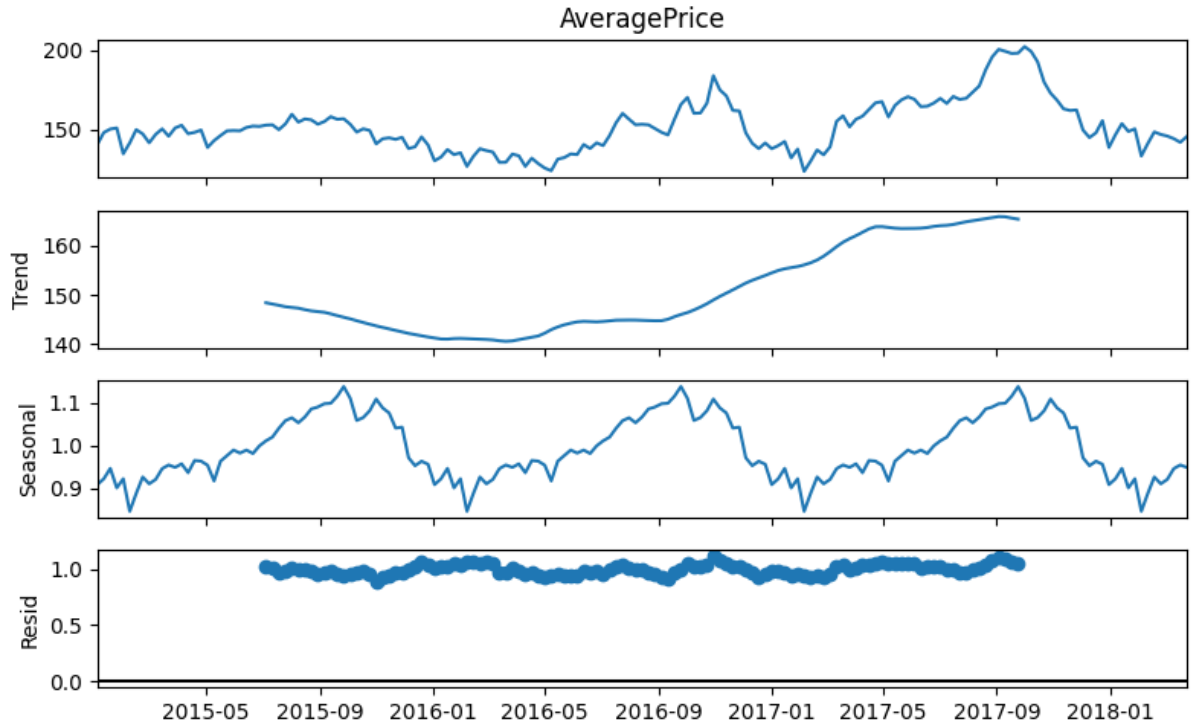
Các cột như Total Volume, Total Bags, hoặc các loại bơ như 4046, 4225, 4770 đại diện cho số lượng hoặc khối lượng, ta sẽ gom nhóm bằng tổng các giá trị của chúng. Cột AveragePrice thể hiện giá của sản phẩm nên việc gom nhóm bằng tổng là không hợp lý nên ta sẽ dùng giá trị trung bình.

4.2 Phân rã chuỗi thời gian

Statsmodels là một thư viện dành cho phân tích thống kê và chuỗi thời gian. Nó cung cấp các công cụ để phân tích dữ liệu và hỗ trợ mô hình hoá dữ liệu chuỗi thời gian với ARIMA, SARIMA, SARIMAX.

Thư viện statsmodel hỗ trợ phương thức seasonal_decompose để phân rã chuỗi thời gian thành các thành phần: xu hướng dài hạn, yếu tố mùa vụ và biến động ngẫu nhiên.

```
import statsmodels.api as sm
fig = sm.tsa.seasonal_decompose(df1['AveragePrice'], model = 'multiplicative').plot()
fig.set_size_inches((8, 5))
fig.tight_layout(), plt.show()
```



Hình 4.2. Phân rã chuỗi thời gian.

4.3 Kiểm tra tính dừng (Stationarity)

Một chuỗi thời gian là dừng khi giá trị trung bình, phương sai, hiệp phương sai (tại các độ trễ khác nhau) giữ nguyên không đổi cho dù chuỗi được xác định vào thời điểm nào đi nữa. Chuỗi dừng có xu hướng trở về giá trị trung bình và những dao động quanh giá trị trung bình sẽ là như nhau. Nói cách khác, một chuỗi thời gian không dừng sẽ có giá trị trung bình thay đổi theo thời gian, hoặc giá trị phương sai thay đổi theo thời gian hoặc cả hai.

Việc kiểm tra tính dừng của chuỗi thời gian là quan trọng vì nhiều mô hình chuỗi thời gian yêu cầu chuỗi thời gian phải là dừng (stationary). Nếu chuỗi không dừng, các ước lượng từ mô hình có thể không chính xác.

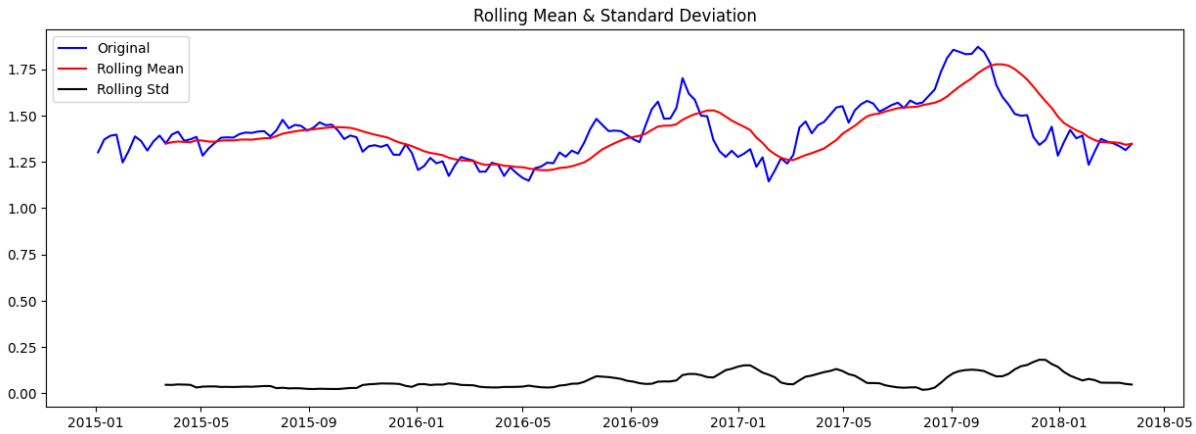
Có nhiều phương pháp kiểm tra tính dừng của chuỗi thời gian: kiểm định Dickey–Fuller (DF), kiểm định Phillip–Person (PP) và kiểm định Dickey và Fuller mở rộng (ADF), kiểm tra bằng giản đồ tự tương quan…

Ta sẽ thực hiện kiểm tra tính dừng của chuỗi thời gian với kiểm định Dickey và Fuller mở rộng (ADF), sử dụng thư viện statsmodels:

```
from statsmodels.tsa.stattools import adfuller
def test_stationarity(timeseries):
    MA = timeseries.rolling(window = 12).mean()
    MSTD = timeseries.rolling(window = 12).std()

    plt.figure(figsize=(15,5))
    orig = plt.plot(timeseries, color='blue',label='Original')
    mean = plt.plot(MA, color='red', label='Rolling Mean')
    std = plt.plot(MSTD, color='black', label = 'Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show(block=False)

    print('Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries, autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags
Used','Number of Observations Used'])
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    print(dfoutput)
test_stationarity(df1['AveragePrice'])
```

Output:

```
Results of Dickey-Fuller Test:
Test Statistic                  -2.36
p-value                          0.15
#Lags Used                       0.00
Number of Observations Used    168.00
Critical Value (1%)             -3.47
Critical Value (5%)             -2.88
Critical Value (10%)            -2.58
dtype: float64
```

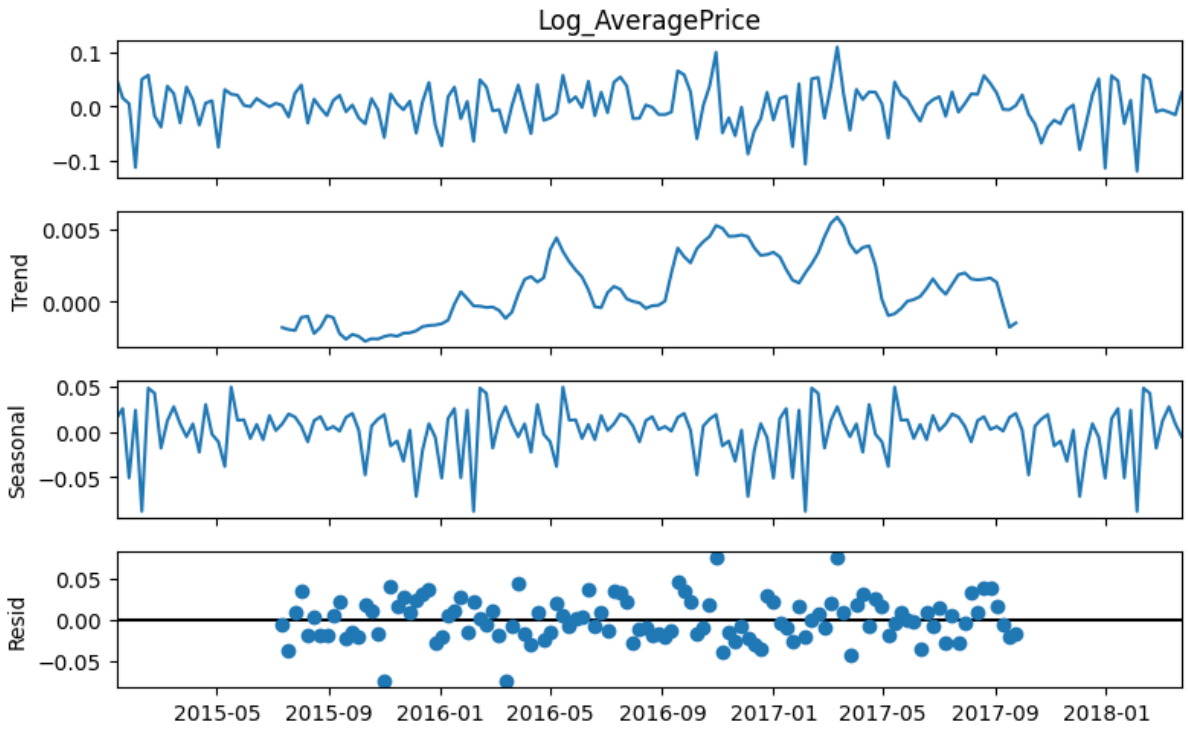


Kết quả kiểm định cho thấy giá trị thống kê kiểm định (-2.36) lớn hơn tất cả các giá trị tới hạn (critical values) và p-value > 0.05. Điều này có nghĩa là chuỗi thời gian không dừng (non-stationary) và có thể cần được xử lý thêm như lấy sai phân hoặc chuyển đổi logarit để làm cho chuỗi dừng.

Thực hiện chuyển đổi logarit và lấy sai phân:

```
df1['Log_AveragePrice'] = np.log(df1['AveragePrice'])
df1_log_diff = df1['Log_AveragePrice'].diff()
df1_log_diff = df1_log_diff.dropna()
```

```
fig = sm.tsa.seasonal_decompose(df1_log_diff,period = 52).plot()
fig.set_size_inches((8, 5))
fig.tight_layout(), plt.show()
```



Hình 4.3. Phân rã chuỗi thời gian (Log + Diff).

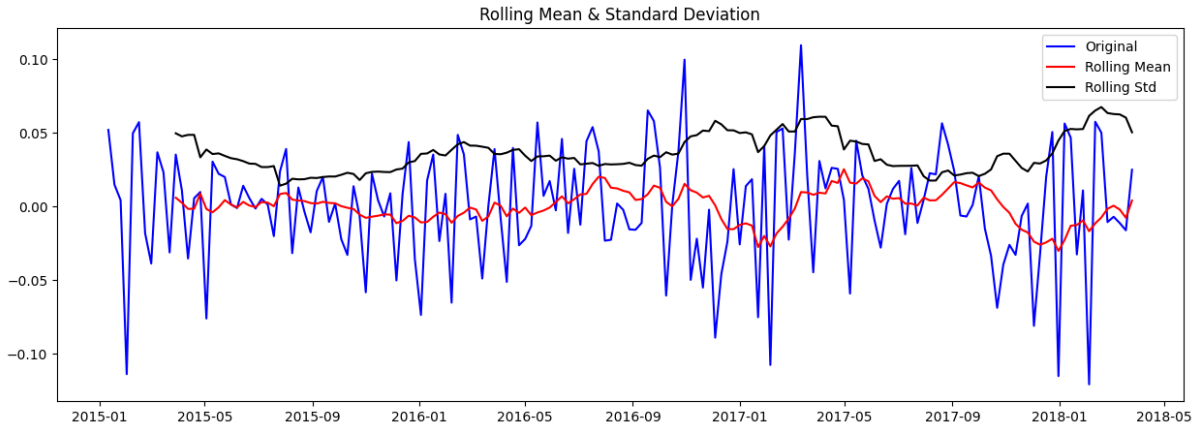
Sau khi chuyển đổi logarit và lấy sai phân, ta thấy rằng giá trị của Trend và Seasonal đã giảm đi.

Kiểm định lại tính dừng của chuỗi thời gian:

```
test_stationarity(df1_log_diff)
```

Output:

```
Results of Dickey-Fuller Test:
Test Statistic                -13.82
p-value                         0.00
#Lags Used                      0.00
Number of Observations Used   167.00
Critical Value (1%)            -3.47
Critical Value (5%)            -2.88
Critical Value (10%)           -2.58
dtype: float64
```



Từ kết quả kiểm định ta thấy Rolling Mean có giá trị gần 0 và có dao động nhỏ, Rolling Standard Deviation có giá trị dao động quanh 0.05. Vì giá trị thống kê kiểm định (-13.82) nhỏ hơn các giá trị tới hạn và p-value nhỏ hơn 0.05, chúng ta có thể kết luận chuỗi thời gian là dừng (stationary). Chuỗi thời gian của bạn đã đạt được tính ổn định và có thể được sử dụng trong các mô hình.

5 Huấn luyện mô hình

Khác với các mô hình dự báo thông thường trong machine learning, các mô hình trong dự báo chuỗi thời gian trong kinh tế lượng có những đặc trưng rất riêng. Đòi hỏi phải tuân thủ các điều kiện về chuỗi dừng, nhiễu trắng và tự tương quan. Có rất nhiều lớp mô hình chuỗi thời gian khác nhau và mỗi một lớp mô hình sẽ có một tiêu chuẩn áp dụng cụ thể. Chúng ta có thể liệt kê một số mô hình phổ biến[3]:

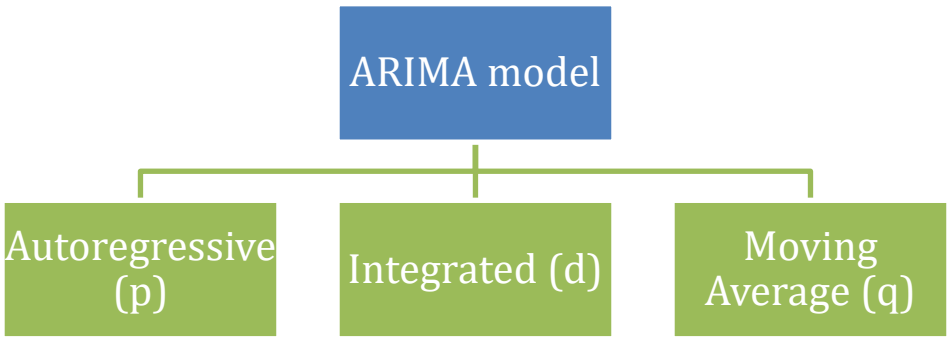
- Mô hình ARIMA: Dựa trên giả thuyết chuỗi dừng và phương sai sai số không đổi. Mô hình sử dụng đầu vào chính là những tín hiệu quá khứ của chuỗi được dự báo để dự báo nó. Các tín hiệu đó bao gồm: chuỗi tự hồi qui AR (auto regression) và chuỗi trung bình trượt MA (moving average). Hầu hết các chuỗi thời gian sẽ có xu hướng tăng hoặc giảm theo thời gian, do đó yếu tố chuỗi dừng thường không đạt được. Trong trường hợp chuỗi không dừng thì ta sẽ cần biến đổi sang chuỗi dừng bằng sai phân. Khi đó tham số đặc trưng của mô hình sẽ có thêm thành phần bậc của sai phân d và mô hình được đặc tả bởi 3 tham số ARIMA(p, d, q).
- Mô hình SARIMA: Về bản chất đây là mô hình ARIMA nhưng được điều chỉnh đặc biệt để áp dụng cho những chuỗi thời gian có yếu tố mùa vụ. Như chúng ta đã biết về bản chất ARIMA chính là mô hình hồi qui tuyến tính nhưng mối quan hệ tuyến tính thường không giải thích tốt chuỗi trong trường hợp chuỗi xuất hiện yếu tố mùa vụ. Chính vì thế, bằng cách tìm ra chu kì của qui luật mùa vụ và loại bỏ nó khỏi chuỗi ta sẽ dễ dàng hồi qui mô hình theo phương pháp ARIMA.
- Mô hình ARIMAX: Là một dạng mở rộng của model ARIMA. Mô hình cũng dựa trên giải định về mối quan hệ tuyến tính giữa giá trị và phương sai trong quá khứ với giá trị hiện tại và sử dụng phương trình hồi qui tuyến tính được suy ra từ mối quan hệ trong quá khứ nhằm dự báo tương lai. Mô hình sẽ có thêm một vài biến độc lập khác và cũng được xem như một mô hình hồi qui động (hoặc một số tài liệu tiếng việt gọi là mô hình hồi qui động thái). Về bản chất ARIMAX

tương ứng với một mô hình hồi qui đa biến nhưng chiếm lợi thế trong dự báo nhờ xem xét đến yếu tố tự tương quan được biểu diễn trong phần dư của mô hình. Nhờ đó cải thiện độ chính xác.

- Mô hình GARCH: Các giả thuyết về chuỗi dừng và phương sai sai số không đổi đều không dễ đạt được trong thực tế. Trái lại phương sai sai số biến đổi rất dễ xảy ra đối với các chuỗi tài chính, kinh tế bởi thường có những sự kiện không mong đợi và cú sốc kinh tế không lường trước khiến biến động phương sai của chuỗi thay đổi. Trong trường hợp đó nếu áp dụng ARIMA thì thường không mang lại hiệu quả cao cho mô hình. Các nhà kinh tế lượng và thống kê học đã nghĩ đến một lớp mô hình mà có thể dự báo được phương sai để kiểm soát các thay đổi không mong đợi. Dựa trên qui luật của phương sai, kết quả dự báo chuỗi sẽ tốt hơn so với trước đó.

5.1 Mô hình ARIMA (Autoregressive Intergrated Moving Average)

Chúng ta biết rằng hầu hết các chuỗi thời gian đều có sự tương quan giữa giá trị trong quá khứ đến giá trị hiện tại. Mức độ tương quan càng lớn khi chuỗi càng gần thời điểm hiện tại. Chính vì thế mô hình ARIMA sẽ tìm cách đưa vào các biến trễ nhằm tạo ra một mô hình dự báo fitting tốt hơn giá trị của chuỗi.



Hình 5.1. Mô hình ARIMA.

Các thành phần của ARIMA[3], [4], [5]:

- Auto regression: Kí hiệu là AR. Đây là thành phần tự hồi qui bao gồm tợp hợp các độ trễ của biến hiện tại. Độ trễ bậc p chính là giá trị lùi về quá khứ p bước thời gian của chuỗi. Độ trễ dài hoặc ngắn trong quá trình AR phụ thuộc vào tham số trễ p. Cụ thể, quá trình $AR(p)$ của chuỗi x_t được biểu diễn như bên dưới:

$$\mathrm{AR}(p) = \phi_0 + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p}$$

- Moving average: Quá trình trung bình trượt được hiểu là quá trình dịch chuyển hoặc thay đổi giá trị trung bình của chuỗi theo thời gian. Do chuỗi của chúng ta được giả định là dừng nên quá trình thay đổi trung bình dường như là một chuỗi nhiễu trắng. Quá trình moving average sẽ tìm mối liên hệ về mặt tuyến tính giữa các phần tử ngẫu nhiên ϵ_t (stochastic term). Chuỗi này phải là một chuỗi nhiễu trắng thỏa mãn các tính chất:

$$\begin{cases} \mathrm{E}(\epsilon_t) = 0 & (1) \\ \sigma(\epsilon_t) = \alpha & (2) \\ \rho(\epsilon_t, \epsilon_{t-s}) = 0, \forall s <= t & (3) \end{cases}$$

- Intergrated: Là quá trình đồng tích hợp hoặc lấy sai phân. Yêu cầu chung của các thuật toán trong time series là chuỗi phải đảm bảo tính dừng (stationary). Hầu hết các chuỗi đều tăng hoặc giảm theo thời gian. Do đó yếu tố tương quan giữa chúng chưa chắc là thực sự mà là do chúng cùng tương quan theo thời gian. Khi biến đổi sang chuỗi dừng, các nhân tố ảnh hưởng thời gian được loại bỏ và chuỗi sẽ dễ dự báo hơn. Để tạo thành chuỗi dừng, một phương pháp đơn giản nhất là chúng ta sẽ lấy sai phân. Một số chuỗi tài chính còn qui đổi sang logarit hoặc lợi suất. Bậc của sai phân để tạo thành chuỗi dừng còn gọi là bậc của quá trình đồng tích hợp (order of intergration). Quá trình sai phân bậc d của chuỗi được thực hiện như sau:
    - Sai phân bậc 1: $\mathrm{I}(1) = \Delta(x_t) = x_t - x_{t-1}$
    - Sai phân bậc d: $\mathrm{I}(d) = \Delta^d(x_t) = \underbrace{\Delta\left(\Delta(\dots\Delta(x_t))\right)}_{\text{d times}}$

Phương trình hồi qui ARIMA(p, d, q) có thể được biểu diễn dưới dạng:

$$\Delta x_t = \phi_1 \Delta x_{t-1} + \phi_2 \Delta x_{t-2} + \cdots + \phi_p \Delta x_{t-p} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}$$

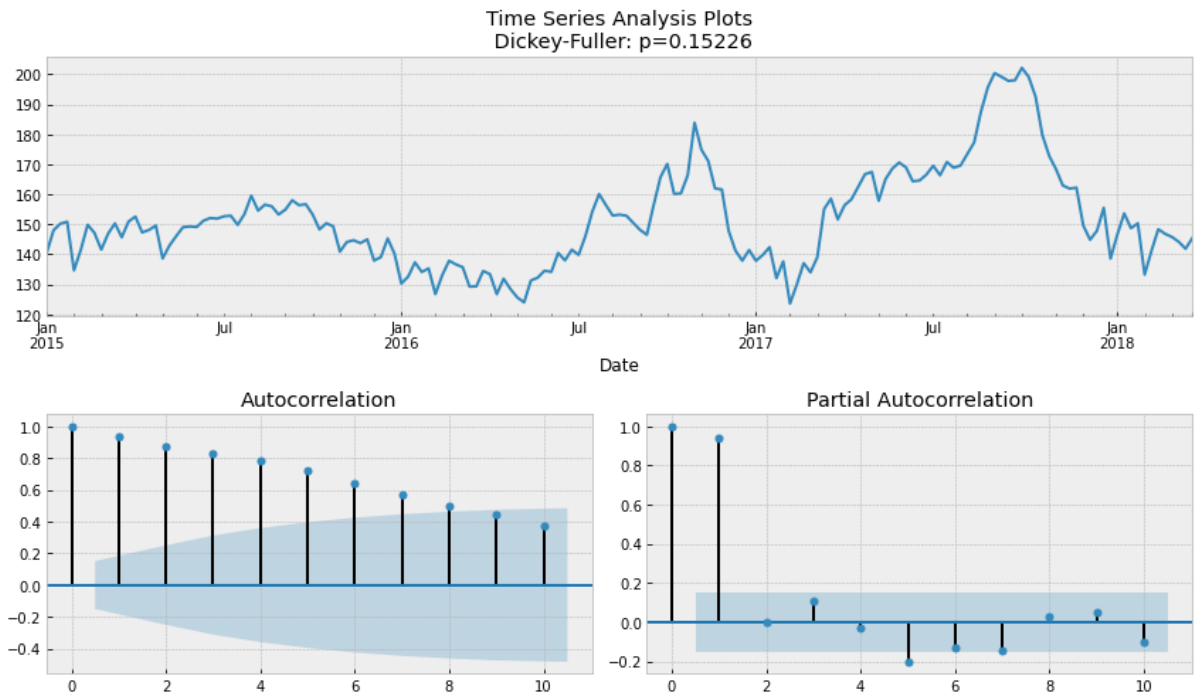
5.1.1 Tạo mô hình dự đoán

ACF (Autocorrelation Function)[3], [6], [7] là một công cụ trong phân tích chuỗi thời gian dùng để đo lường mối quan hệ tuyến tính giữa một giá trị trong chuỗi và các giá trị tại các độ trễ khác nhau. Nó cho biết mức độ tự tương quan của chuỗi tại mỗi độ trễ.

PACF (Partial Autocorrelation Function)[3], [6], [7] được sử dụng để xác định bậc p của thành phần tự hồi quy (AR) trong mô hình ARIMA. Nó đo lường mối tương quan trực tiếp giữa giá trị hiện tại và các giá trị tại các độ trễ khác nhau, sau khi loại bỏ ảnh hưởng của các độ trễ trung gian. Dựa vào đồ thị, nếu PACF cắt đứt nhanh sau một độ trễ cụ thể, đó có thể là bậc p của thành phần AR. Đồ thị này hỗ trợ phân tích mối quan hệ trong chuỗi thời gian và lựa chọn mô hình phù hợp.

```
def tsplot(y, lags=None, figsize=(12, 7), style='bmh'):
    if not isinstance(y, pd.Series):
        y = pd.Series(y)

    with plt.style.context(style):
        fig = plt.figure(figsize=figsize)
        layout = (2, 2)
        ts_ax = plt.subplot2grid(layout, (0, 0), colspan=2)
        acf_ax = plt.subplot2grid(layout, (1, 0))
        pacf_ax = plt.subplot2grid(layout, (1, 1))

        y.plot(ax=ts_ax)
        p_value = sm.tsa.stattools.adfuller(y)[1]
        ts_ax.set_title('Time Series Analysis Plots\n Dickey-Fuller:
p={0:.5f}'.format(p_value))
        smt.graphics.plot_acf(y, lags=lags, ax=acf_ax)
        smt.graphics.plot_pacf(y, lags=lags, ax=pacf_ax)
        plt.tight_layout()
    plt.show()
```

```
tsplot(df1['AveragePrice'],lags = 10)
```

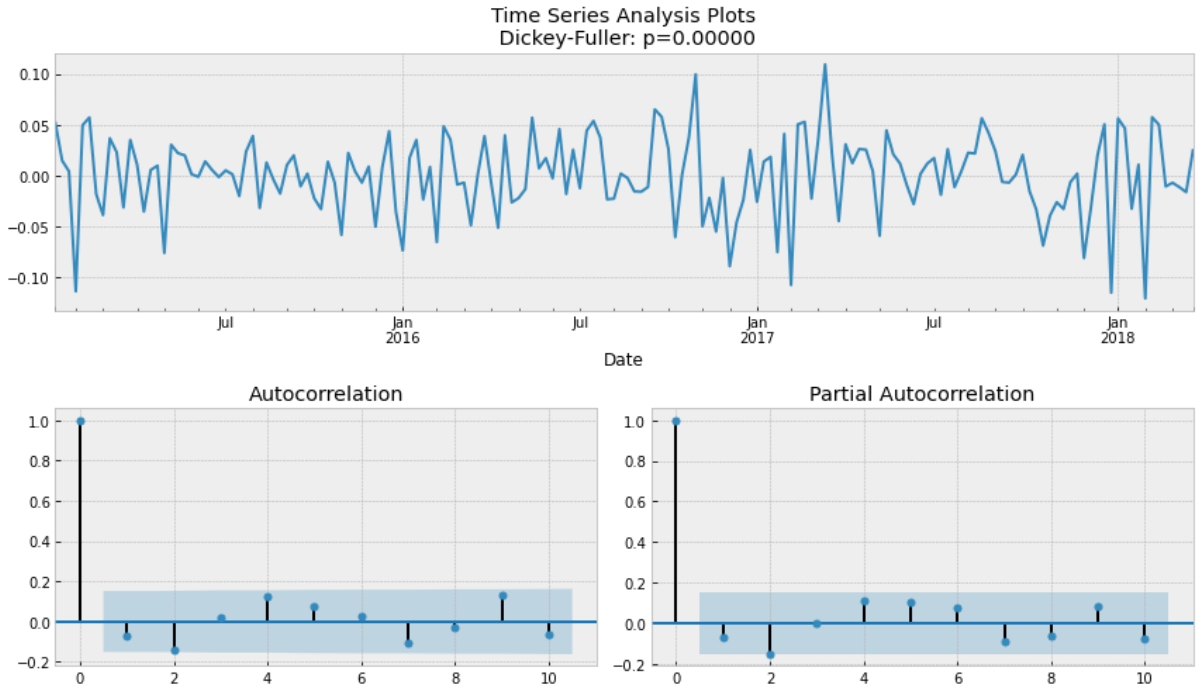


Hình 5.2. Đồ thị ACF và PACF.

Từ đồ thị ACF ta thấy mối tương quan giảm dần. Với đồ thị PACF ta thấy rằng có thể độ trễ đầu tiên là quan trọng nhất.

Quan sát đồ thị ACF và PACF ở **Hình 5.3** với chuỗi thời gian đã chuyển đổi logarit và lấy sai phân bậc 1 (ở mục **4.3**):

```
tsplot(df1_log_diff,lags = 10)
```



Hình 5.3. ACF và PACF với chuỗi chuyển đổi logarit và lấy sai phân.

Từ các biểu đồ Hình 5.2, Hình 5.3 ta chọn ra các thành phần của ARIMA model:

- p = 1, dựa vào partial autocorrelation, tìm lag đầu tiên giảm xuống 0 mà không ảnh hưởng bởi lag trước đó.
- d = 1, số lần thực hiện sai phân, ở mục **4.3** ta đã lấy sai phân một lần.
- q = 2, do ACF và PACF đều có các giá trị đáng kể ở lag 1 và 2.

Ta sử dụng ARIMA trong statsmodels:

```
from statsmodels.tsa.arima_model import ARIMA as ARIMA
model = ARIMA(df['Log_AveragePrice'],order = (1,1,2))
model_fit = model.fit()
print(model_fit.summary())
```

Output:

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:     D.Log_AveragePrice   No. Observations:                  168
Model:                 ARIMA(1, 1, 2)   Log Likelihood                 308.989
Method:                       css-mle   S.D. of innovations              0.038
Date:                Mon, 15 Dec 2024   AIC                           -607.978
Time:                        15:37:49   BIC                           -592.358
Sample:                    01-11-2015   HQIC                          -601.639
                         - 03-25-2018
==============================================================================
                                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
-
const                          0.0001      0.002      0.055      0.956      -0.005
0.005
ar.L1.D.Log_AveragePrice      -0.1490      0.388     -0.384      0.701      -0.909
0.611
```

```
ma.L1.D.Log_AveragePrice     0.0733      0.379      0.193      0.847      -0.670
0.817
ma.L2.D.Log_AveragePrice    -0.1238      0.074     -1.663      0.096      -0.270
0.022
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           -6.7128           +0.0000j            6.7128            0.5000
MA.1           -2.5609           +0.0000j            2.5609            0.5000
MA.2            3.1531           +0.0000j            3.1531            0.0000

-----------------------------------------------------------------------------
```

5.1.2 In-sample forecasting

In-sample forecasting (Dự đoán trên tập mẫu): Đây là việc sử dụng dữ liệu đã có để xây dựng mô hình và thực hiện dự báo, giống như việc sử dụng dữ liệu huấn luyện và dữ liệu kiểm thử để đánh giá mô hình trong bài toán hồi quy và phân loại.

Ta tiến hành chia tập dữ liệu thành train và test, sử dụng 30 giá trị cuối của tập dữ liệu để làm tập test và phần còn lại để làm tập train. Ở đây ta sử dụng phương pháp rolling forecast, tức là sử dụng giá trị đã dự đoán để dự đoán giá trị tiếp theo.

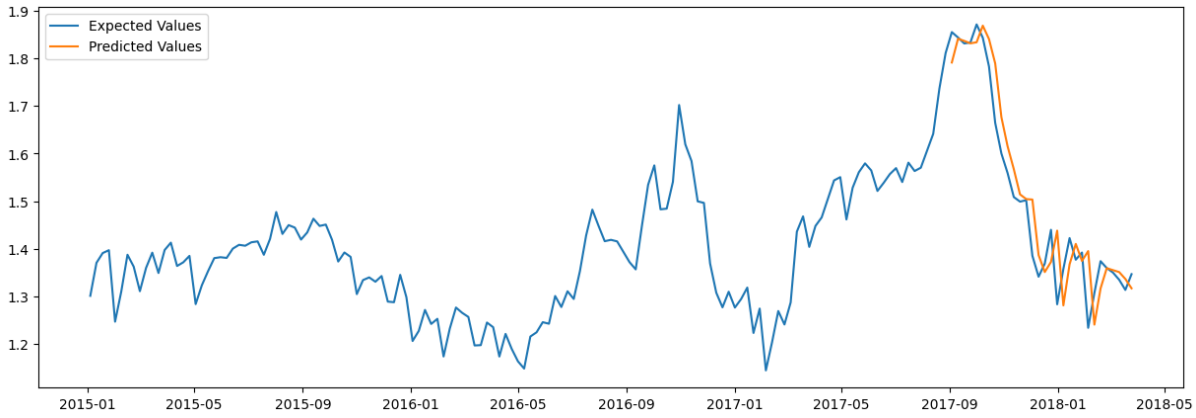
```
size = int(len(df1) - 30)
train, test = df1['Log_AveragePrice'][0:size], df1['Log_AveragePrice'][size:len(df1)]
print('\t ARIMA MODEL : In - Sample Forecasting \n')
history = [x for x in train]
predictions = []
for t in range(len(test)):
    model = ARIMA(history, order=(1,1,2))
    model_fit = model.fit(disp = 0)
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(float(yhat))
    obs = test[t]
    history.append(obs)
    print('predicted = %f, expected = %f' % (np.exp(yhat), np.exp(obs)))
```

Output:

```
        ARIMA MODEL : In - Sample Forecasting

predicted = 1.791331, expected = 1.855185
predicted = 1.841266, expected = 1.843889
predicted = 1.836749, expected = 1.831389
predicted = 1.831610, expected = 1.833333
predicted = 1.833854, expected = 1.871296
predicted = 1.868752, expected = 1.843333
predicted = 1.840565, expected = 1.782870
predicted = 1.789364, expected = 1.664537
predicted = 1.676536, expected = 1.600185
predicted = 1.615299, expected = 1.559074
predicted = 1.567359, expected = 1.508796
predicted = 1.514397, expected = 1.499167
predicted = 1.504528, expected = 1.502130
predicted = 1.503498, expected = 1.385370
predicted = 1.386997, expected = 1.341481
predicted = 1.351253, expected = 1.368981
predicted = 1.372485, expected = 1.439907
predicted = 1.438179, expected = 1.283519
predicted = 1.281181, expected = 1.357778
predicted = 1.367572, expected = 1.422593
predicted = 1.410333, expected = 1.377130
predicted = 1.374604, expected = 1.392222
```

```
predicted = 1.395309, expected = 1.234074
predicted = 1.240819, expected = 1.307037
predicted = 1.316857, expected = 1.374074
predicted = 1.359375, expected = 1.359630
predicted = 1.355281, expected = 1.350185
predicted = 1.351213, expected = 1.335093
predicted = 1.336742, expected = 1.313704
predicted = 1.317080, expected = 1.346852
```

```
predictions_series = pd.Series(predictions, index = test.index)
fig,ax = plt.subplots(nrows = 1,ncols = 1,figsize = (15,5))

plt.subplot(1,1,1)
plt.plot(df1['AveragePrice'],label = 'Expected Values')
plt.plot(np.exp(predictions_series),label = 'Predicted Values');
plt.legend(loc="upper left")
plt.show()
```



Hình 5.4. Dự đoán trên tập mẫu (ARIMA model).

Giá trị dự đoán rất gần với giá trị thực tế, điều này cho thấy mô hình có thể đã bị quá khớp.

```
error = np.sqrt(mean_squared_error(np.exp(test),np.exp(predictions)))
print('Test RMSE: %.4f' % error)
predictions_series = pd.Series(np.exp(predictions), index = test.index)
```

Output:

```
Test RMSE: 0.0654
```

5.1.3 Out-of-sample forecasts

Out-of-sample forecasting (dự đoán ngoài mẫu) là quá trình sử dụng một mô hình đã được huấn luyện trên một tập dữ liệu (training data) để dự đoán giá trị của một tập dữ liệu mới, chưa từng được sử dụng trong quá trình huấn luyện. Đây là một phương pháp đánh giá hiệu suất và khả năng tổng quát hóa của mô hình.

Thêm các điểm dữ liệu mới trong tương lai vào DataFrame:

```
from pandas.tseries.offsets import DateOffset
future_dates = [df1.index[-1] + DateOffset(weeks = x) for x in range(0,52)]
df2 = pd.DataFrame(index = future_dates[1:],columns = df1.columns)

forecast = pd.concat([df1,df2])
forecast['ARIMA_Forecast_Function'] = np.nan
```

```
forecast
```

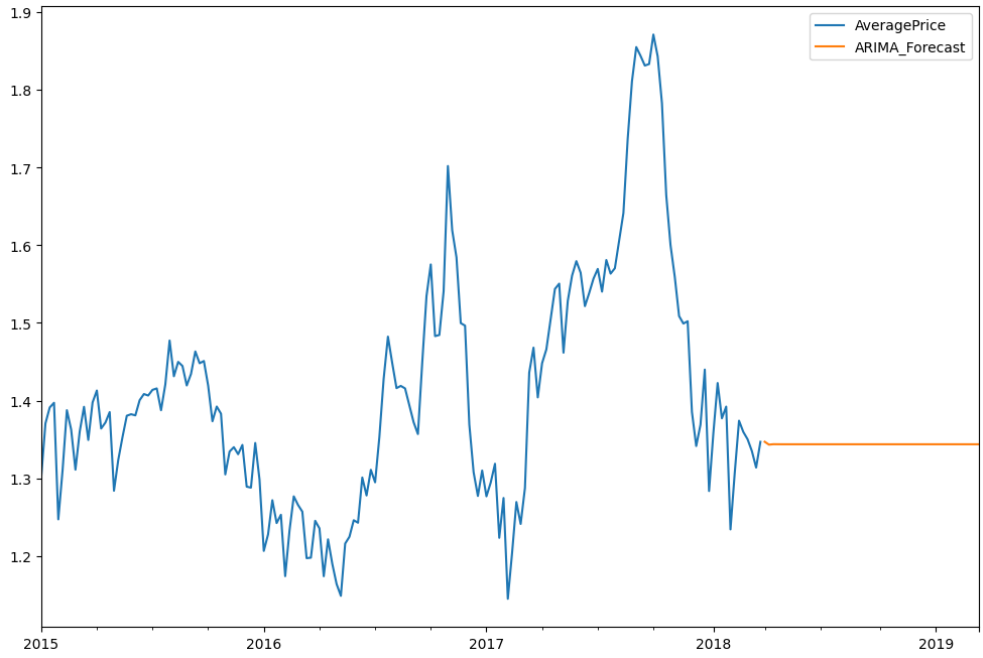
	AveragePrice	Log_AveragePrice	ARIMA_Forecast
2015-01-04	1.30	0.26	NaN
2015-01-11	1.37	0.32	NaN
2015-01-18	1.39	0.33	NaN
2015-01-25	1.40	0.33	NaN
2015-02-01	1.25	0.22	NaN
...	...	...	...
2019-02-17	NaN	NaN	NaN
2019-02-24	NaN	NaN	NaN
2019-03-03	NaN	NaN	NaN
2019-03-10	NaN	NaN	NaN
2019-03-17	NaN	NaN	NaN

220 rows × 3 columns

Hình 5.5. DataFrame được thêm các điểm dữ liệu tương lai.

Mô hình sẽ dự đoán các giá trị trong tương lai kể từ cuối năm 2018 đến tháng 3 năm 2019.

```
f1 = np.array(np.exp(model_fit.forecast(steps = 51)))
for i in range(len(f1)):
    forecast.iloc[169 + i,2] = f1[i]
forecast[['AveragePrice','ARIMA_Forecast']].plot(figsize = (12,8))
plt.show()
```



Hình 5.6. Kết quả dự đoán ngoài mẫu (ARIMA).

Kết quả cho thấy mô hình không thể dự đoán được các giá trị chưa biết trong tương lai, điều này có thể là do mô hình không nắm bắt được yếu tố mùa vụ của chuỗi thời gian.

5.2 Mô hình SARIMA (Seasonal Autoregressive Integrated Moving Average)

Mô hình SARIMA (Seasonal Autoregressive Integrated Moving Average) là một mô hình phổ biến trong phân tích chuỗi thời gian, đặc biệt khi dữ liệu có tính chất mùa vụ

(seasonality). SARIMA mở rộng mô hình ARIMA (AutoRegressive Integrated Moving Average) bằng cách bổ sung các thành phần mùa vụ, giúp mô hình hóa các chuỗi thời gian có xu hướng thay đổi theo mùa (seasonal patterns).

Cấu trúc của mô hình SARIMA gồm có các thành phần sau:

- AR (AutoRegressive): Phần này liên quan đến các giá trị trong quá khứ của chuỗi thời gian để dự báo giá trị tương lai.
- I (Integrated): Thành phần này xử lý sự không ổn định của chuỗi bằng cách lấy hiệu giữa các giá trị liên tiếp, giúp chuỗi thời gian trở thành chuỗi ổn định.
- MA (Moving Average): Phần này mô hình hóa sự ảnh hưởng của sai số trong các quan sát trước đó.
- Seasonal components:
    - Seasonal AR (SAR): Phần autoregressive mùa vụ, mô hình hóa sự ảnh hưởng của các giá trị trong quá khứ của chuỗi thời gian ở một chu kỳ mùa vụ cụ thể.
    - Seasonal I (SI): Phần integrated mùa vụ, giúp xử lý sự không ổn định của chuỗi theo mùa.
    - Seasonal MA (SMA): Phần moving average mùa vụ, mô hình hóa sai số ở các thời điểm theo mùa vụ.

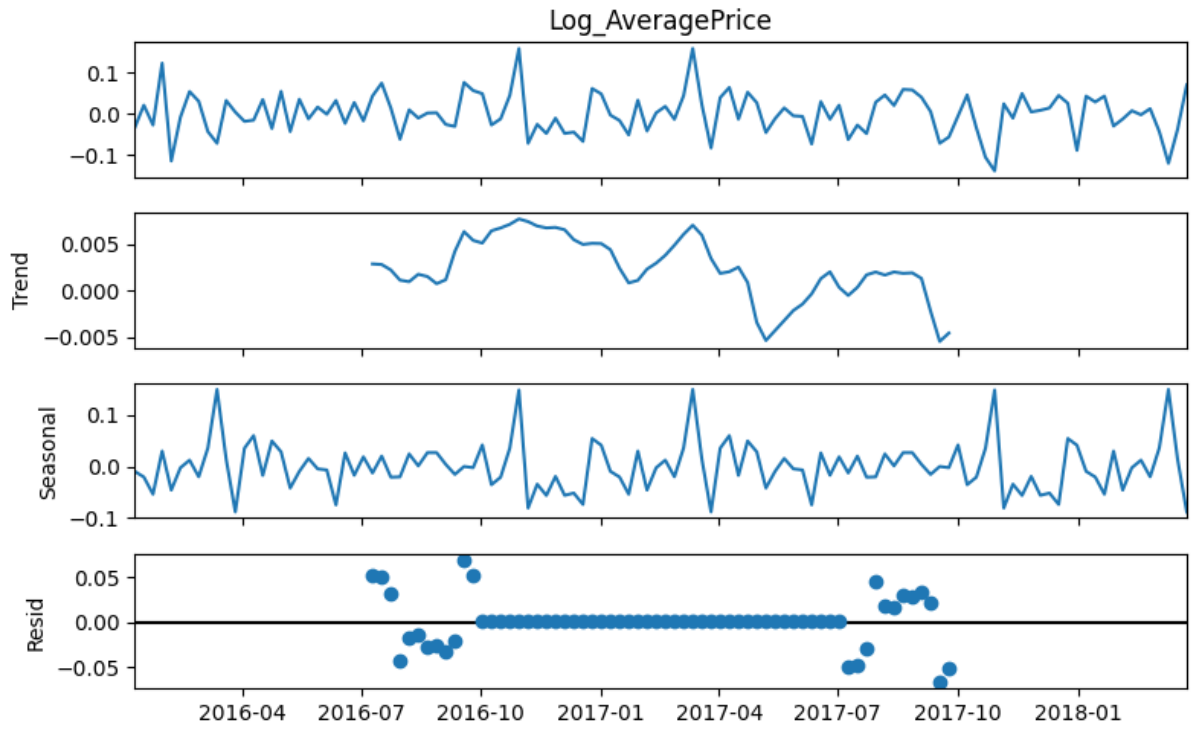
Mô hình SARIMA được ký hiệu là: SARIMA(p, d, q) x (P, D, Q, M), trong đó

- p, d, q là các tham số của mô hình ARIMA thông thường, với:
    - p: Số lượng độ trễ của phần AR (autoregressive).
    - d: Số lần khác biệt hóa chuỗi để đạt tính ổn định (differencing).
    - q: Số lượng độ trễ của phần MA (moving average).
- P, D, Q, M là các tham số của phần mùa vụ trong đó (P,D,Q) tương tự như (p,d,q) của mô hình ARIMA:
    - P: Số lượng độ trễ của phần AR mùa vụ.
    - D: Số lần khác biệt hóa chuỗi theo mùa vụ.
    - Q: Số lượng độ trễ của phần MA mùa vụ.
    - M là chu kỳ mùa vụ, ví dụ: m = 12 cho dữ liệu tháng (nếu dữ liệu có tính mùa vụ hàng năm).

5.2.1 Tạo mô hình dự đoán

Đối với bộ dữ liệu này, ta xác định một chu kỳ mùa vụ là một năm. Do đó, ta xác định giá trị M là 52. Thực hiện sai phân theo với giá trị M.

```
df1_log_diff_seas = df1_log_diff.diff(52)
df1_log_diff_seas = df1_log_diff_seas.dropna()
dec = sm.tsa.seasonal_decompose(df1_log_diff_seas).plot()
plt.show()
```



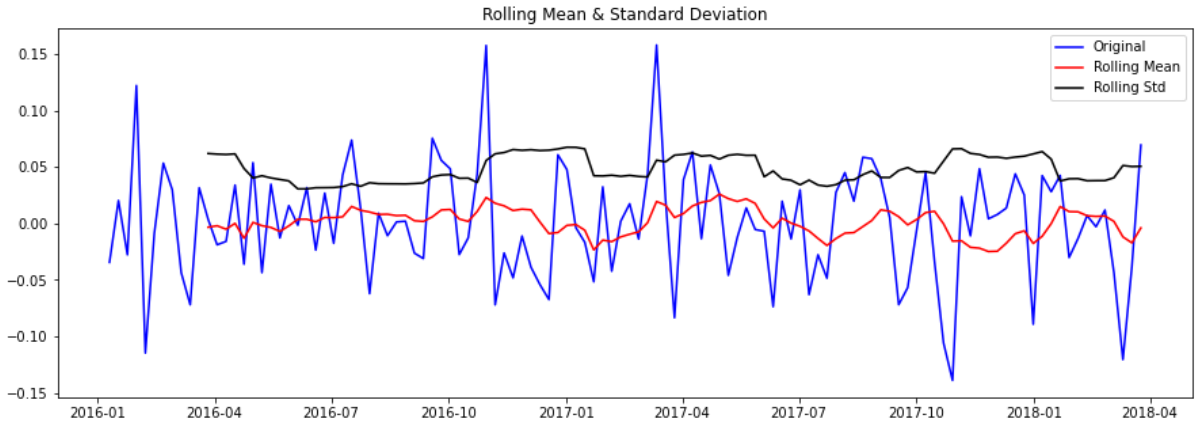
Hình 5.7. Kết quả phân rã chuỗi thời gian.

Kiểm tra tính dừng của dữ liệu:

```
test_stationarity(df1_log_diff_seas)
```

Output:

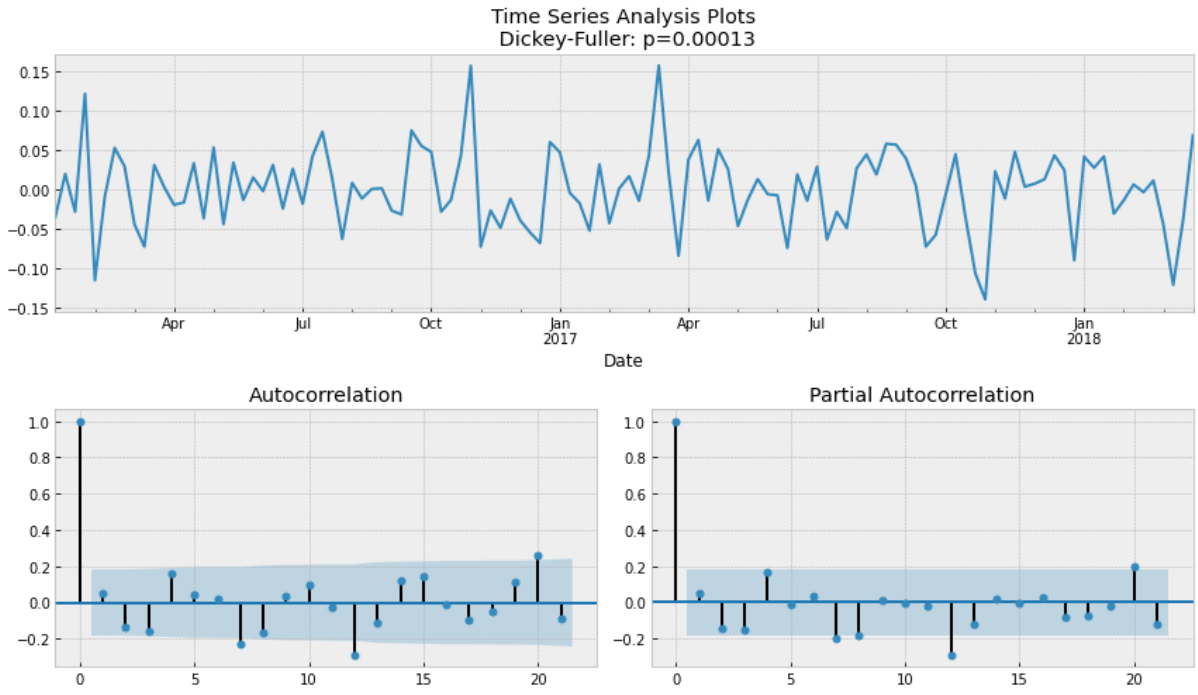
```
Results of Dickey-Fuller Test:
Test Statistic                  -4.57
p-value                          0.00
#Lags Used                      11.00
Number of Observations Used    104.00
Critical Value (1%)             -3.49
Critical Value (5%)             -2.89
Critical Value (10%)            -2.58
dtype: float64
```



Hình 5.8. Chuỗi sau khi biến đổi và các cửa sổ trượt trung bình, độ lệch chuẩn.

Giá trị thống kê kiểm định (-4.57) nhỏ hơn các giá trị tới hạn và p-value nhỏ hơn 0.05, cho nên chuỗi thời gian là dừng.

```
tsplot(df1_log_diff_seas)
```



Hình 5.9. ACF và PCF.

Đối với các tham số (p,d,q) ta sẽ chọn lại các tham số của ARIMA, với (P,D,Q,M) cách chọn tương tự với ARIMA, ta có:

- P = 0
- D = 1
- Q = 0
- M = 52

Sử dụng SARIMA với statsmodels

```
import statsmodels.api as sm
model = sm.tsa.statespace.SARIMAX(df1['Log_AveragePrice'],order = (1,1,2),seasonal_order =
(0,1,0,52))
model_fit = model.fit()
print(model_fit.summary())
```

Output:

```
                                SARIMAX Results
==========================================================================================
Dep. Variable:                  Log_AveragePrice   No. Observations:                  169
Model:             SARIMAX(1, 1, 2)x(0, 1, [], 52)   Log Likelihood                 184.923
Date:                             Mon, 15 Dec 2024   AIC                           -361.845
Time:                                     15:38:00   BIC                           -350.831
Sample:                                 01-04-2015   HQIC                          -357.374
                                      - 03-25-2018
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.7727      0.169      4.566      0.000       0.441       1.104
ma.L1         -0.7458      0.184     -4.047      0.000      -1.107      -0.385
```

```
ma.L2         -0.1546      0.109     -1.416      0.157      -0.369       0.059
sigma2         0.0024      0.000      9.163      0.000       0.002       0.003
===================================================================================
Ljung-Box (Q):                       77.68   Jarque-Bera (JB):                 7.83
Prob(Q):                              0.00   Prob(JB):                         0.02
Heteroskedasticity (H):               1.16   Skew:                             0.10
Prob(H) (two-sided):                  0.64   Kurtosis:                         4.26
===================================================================================

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

5.2.2 In-sample forecasting

Tương tự như với ARIMA, ta chia dữ liệu thành train và test với 30 điểm dữ liệu cuối làm tập test, và sử dụng phương pháp rolling forecast.

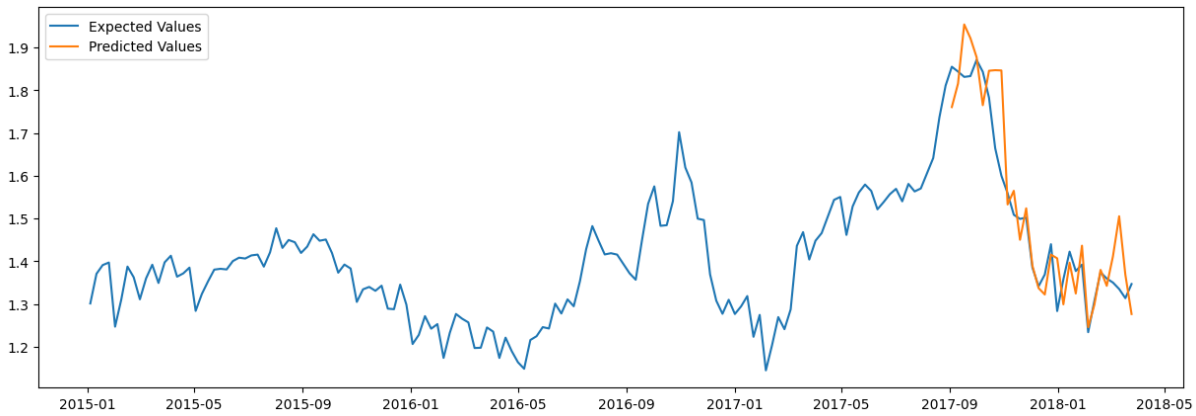
```
size = int(len(df1) - 30)
train, test = df1['Log_AveragePrice'][0:size], df1['Log_AveragePrice'][size:len(df1)]
print('\t SARIMA MODEL : In - Sample Forecasting \n')
history = [x for x in train]
predictions = []
for t in range(len(test)):
    model = sm.tsa.statespace.SARIMAX(history,order = (1,1,2),seasonal_order = (0,1,0,52))
    model_fit = model.fit(disp = 0)
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(float(yhat))
    obs = test[t]
    history.append(obs)
    print('predicted = %f, expected = %f' % (np.exp(yhat), np.exp(obs)))
```

Output:

```
         SARIMA MODEL : In - Sample Forecasting

predicted = 1.760839, expected = 1.855185
predicted = 1.815748, expected = 1.843889
predicted = 1.954110, expected = 1.831389
predicted = 1.922169, expected = 1.833333
predicted = 1.877496, expected = 1.871296
predicted = 1.765302, expected = 1.843333
predicted = 1.845840, expected = 1.782870
predicted = 1.847098, expected = 1.664537
predicted = 1.846475, expected = 1.600185
predicted = 1.532948, expected = 1.559074
predicted = 1.564911, expected = 1.508796
predicted = 1.450306, expected = 1.499167
predicted = 1.523822, expected = 1.502130
predicted = 1.389803, expected = 1.385370
predicted = 1.337151, expected = 1.341481
predicted = 1.322277, expected = 1.368981
predicted = 1.415474, expected = 1.439907
predicted = 1.406950, expected = 1.283519
predicted = 1.299212, expected = 1.357778
predicted = 1.396739, expected = 1.422593
predicted = 1.324639, expected = 1.377130
predicted = 1.436510, expected = 1.392222
predicted = 1.245638, expected = 1.234074
predicted = 1.298375, expected = 1.307037
predicted = 1.379873, expected = 1.374074
predicted = 1.342569, expected = 1.359630
predicted = 1.411688, expected = 1.350185
predicted = 1.505319, expected = 1.335093
```

```
predicted = 1.368537, expected = 1.313704
predicted = 1.276548, expected = 1.346852
```

```
predictions_series = pd.Series(predictions, index = test.index)
fig,ax = plt.subplots(nrows = 1,ncols = 1,figsize = (15,5))

plt.subplot(1,1,1)
plt.plot(df1['AveragePrice'],label = 'Expected Values')
plt.plot(np.exp(predictions_series),label = 'Predicted Values');
plt.legend(loc="upper left")
plt.show()
```



Hình 5.10. Kết quả dự đoán trên tập mẫu (SARIMA model).

Kiểm tra RMSE trên tập test:

```
error = np.sqrt(mean_squared_error(np.exp(test),np.exp(predictions)))
print('Test RMSE: %.4f' % error)
predictions_series = pd.Series(np.exp(predictions), index = test.index)
```

Output:

```
Test RMSE: 0.0840
```

5.2.3 Out-of-sample forecasts

Thêm các điểm dữ liệu mới trong tương lai vào DataFrame:

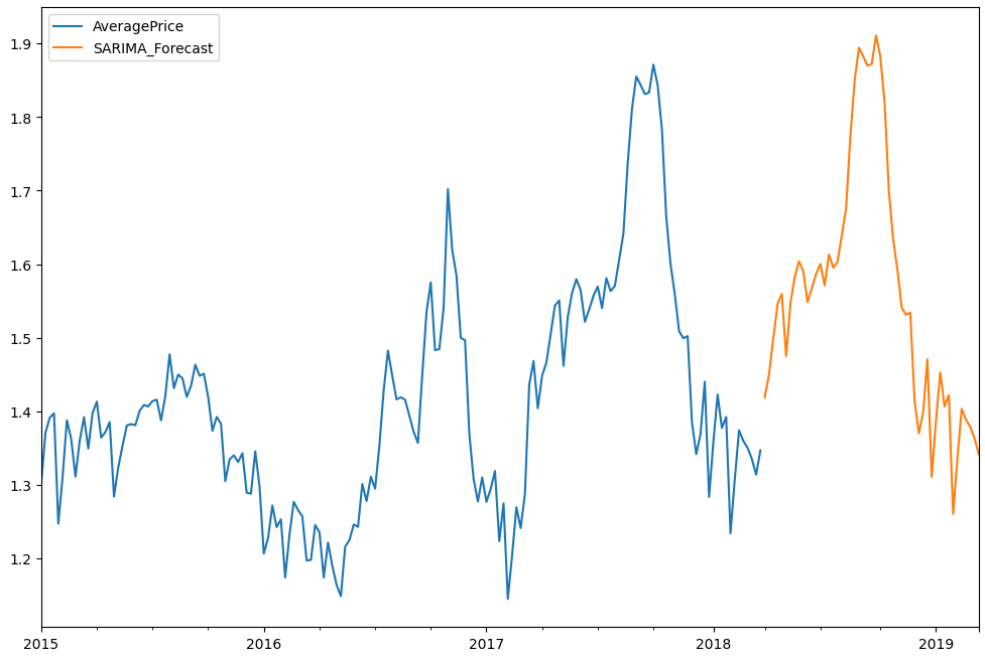
```
from pandas.tseries.offsets import DateOffset
future_dates = [df1.index[-1] + DateOffset(weeks = x) for x in range(0,52)]
df2 = pd.DataFrame(index = future_dates[1:],columns = df1.columns)

forecast = pd.concat([df1,df2])
forecast['ARIMA_Forecast_Function'] = np.nan
forecast
```

	AveragePrice	Log_AveragePrice	ARIMA_Forecast
2015-01-04	1.30	0.26	NaN
2015-01-11	1.37	0.32	NaN
2015-01-18	1.39	0.33	NaN
2015-01-25	1.40	0.33	NaN
2015-02-01	1.25	0.22	NaN
...	...	...	...
2019-02-17	NaN	NaN	NaN
2019-02-24	NaN	NaN	NaN
2019-03-03	NaN	NaN	NaN
2019-03-10	NaN	NaN	NaN
2019-03-17	NaN	NaN	NaN

220 rows × 3 columns

Hình 5.11. DataFrame được thêm các điểm dữ liệu tương lai.

```
f3 = np.array(np.exp(model_fit.forecast(steps = 51)))
for i in range(len(f3)):
    forecast.iloc[169 + i,2] = f3[i]
forecast[['AveragePrice','SARIMA_Forecast']].plot(figsize = (12,8))
plt.show()
```



Hình 5.12. Kết quả dự đoán ngoài mẫu (SARIMA model).

Dựa vào kết quả **Hình 5.12**, mô hình đã có thể nắm bắt được yếu tố mùa vụ và tạo ra những dự đoán cho thấy giá có xu hướng tăng nhẹ trong tương lai.

6 Kết luận

Qua quá trình phân tích dữ liệu bộ Avocado Prices, ta đã thực hiện các bước hiểu dữ liệu, phân tích khám phá dữ liệu, phân tích chuỗi thời gian, huấn luyện mô hình chuỗi thời gian với mô hình ARIMA và SARIMA. Quá trình phân tích khám phá dữ liệu đã cung cấp nhiều thông tin giá trị giúp ta có cái nhìn rõ hơn về những vấn đề liên quan

giá của một sản phẩm. Đó có thể là yếu tố thị trường, tính mùa vụ, sở thích của khách hàng.

Đây là một bộ dữ liệu tốt để hiểu về bài toán phân tích chuỗi thời gian. Trong phân tích chuỗi thời gian, các khái niệm như tính dừng, việc hiểu loại dữ liệu mà các mô hình sẽ huấn luyện, và các phép biến đổi được thực hiện có thể khá phức tạp vì nhiều yếu tố cần phải xử lý.

Kết quả huấn luyện mô hình cho thấy mô hình SARIMA(1,1,2)(0,1,0,52) có khả năng dự báo giá ngoài tập mẫu tốt hơn so với ARIMA, do khả năng nắm bắt yếu tố mùa vụ của SARIMA. Đánh giá chung mô hình ARIMA/SARIMA việc dự báo áp dụng thực tế đòi hỏi nghiên cứu phải tinh chỉnh nhiều lần các tham số mô hình và để có thể cải thiện độ chính xác của dự báo cần nghiên cứu thêm các phương pháp dự báo khác. Từ đó có thể so sánh kết quả dự báo này nhằm xác định một phương pháp dự báo tối ưu nhất.

Tài liệu tham khảo

[1] "Home - Hass Avocado Board." Accessed: Dec. 24, 2024. [Online]. Available: https://hassavocadoboard.com/

[2] "Avocado Prices." Accessed: Dec. 24, 2024. [Online]. Available: https://www.kaggle.com/datasets/neuromusic/avocado-prices

[3] "Khoa học dữ liệu." Accessed: Dec. 23, 2024. [Online]. Available: https://phamdinhkhanh.github.io/2019/12/12/ARIMAmodel.html#2-m%C3%B4-h%C3%ACnh-arima

[4] "Introduction to the Autoregressive Integrated Moving Average (ARIMA) model." Accessed: Dec. 23, 2024. [Online]. Available: https://www.projectguru.in/introduction-to-the-autoregressive-integrated-moving-average-arima-model/

[5] "Autoregressive integrated moving average - Wikipedia." Accessed: Dec. 23, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average

[6] "Choosing the best q and p from ACF and PACF plots in ARMA-type modeling | Baeldung on Computer Science." Accessed: Dec. 23, 2024. [Online]. Available: https://www.baeldung.com/cs/acf-pacf-plots-arma-modeling

[7] "Time Series: Interpreting ACF and PACF." Accessed: Dec. 23, 2024. [Online]. Available: https://www.kaggle.com/code/iamleonie/time-series-interpreting-acf-and-pacf