

GIỚI THIỆU VỀ TIME SERIES FORECASTING (DỰ BÁO CHUỖI THỜI GIAN)

- Time Series Forecasting (Dự đoán chuỗi thời gian) là một phương pháp dự đoán giá trị tương lai dựa trên dữ liệu thu thập theo thời gian, trong đó thời gian là yếu tố quan trọng trong quá trình dự đoán.
- Chuỗi thời gian là một tập các số liệu theo thời gian, thường được sắp xếp theo thứ tự từng đơn vị thời gian, ví dụ như các dữ liệu hàng ngày, hàng giờ, hàng tháng hoặc theo bất kỳ đơn vị thời gian nào khác.

Dự đoán chuỗi thời gian có thể phân chia thành hai loại:

- Nếu sử dụng các giá trị trước đó của chuỗi thời gian để dự đoán các giá trị tương lai của nó, thì đó được gọi là Dự báo chuỗi thời gian Univariate
- Nếu sử dụng các biến dự đoán khác ngoài chuỗi (như biến ảnh hưởng) để dự đoán, thì đó được gọi là Dự báo chuỗi thời gian Multivariate

Dự đoán chuỗi thời gian thường được sử dụng để:

- Dự đoán xu hướng tương lai: Phân tích và dự đoán xu hướng tăng, giảm hoặc ổn định của một biến số theo thời gian, ví dụ như doanh số bán hàng, giá cổ phiếu, nhiệt độ, hay lưu lượng mạng
- Xác định chu kỳ và theo mùa: xác định các chu kỳ lặp lại và mùa trong dữ liệu để dự đoán các biến số theo mùa như doanh số bán hàng trong mùa hè hoặc mùa đông
- Dự đoán sự biến đổi ngẫu nhiên: phân tích và sự đoán sự biến đổi ngẫu nhiên trong dữ liệu, giúp ước tính sai số trong dự đoán

Các phương pháp dự đoán chuỗi thời gian có thể sử dụng nhiều kỹ thuật khác nhau, bao gồm:

- Mô hình ARIMA (autoregressive Integrated Moving Average): là một mô hình thống kê phổ biến dựa trên sự kết hợp của autoregressive (AR) và moving average (MA) để dự đoán các biến số thay đổi theo thời gian
- Mô hình SARIMAX (Seasonal ARIMA with Exogenous Variables): là một mô hình mở rộng của ARIMA, cho phép tích hợp các biến exogenous (biến không phải là một phần của chuỗi thời gian nhưng có thể ảnh hưởng đến nó)

- Mạng nơ-ron học sâu (Deep Learning Neural Networks): sử dụng trong deep learning như Long Short-Term Memory (LSTM) và Gated Recurrent Unit (GRU) đã trở nên phổ biến trong sự dự đoán chuỗi thời gian
- Hot-Winters Exponential Smoothing: phương pháp này tập trung vào việc xác định xu hướng, mùa vụ và sai số trong chuỗi thời gian để thực hiện dự đoán
- Phương pháp thống kê và tự hồi quy

MỤC TIÊU BÁO CÁO

- Phát triển một khung linh hoạt và có khả năng mở rộng có thể áp dụng cho nhiều loại dữ liệu chuỗi thời gian, đồng thời tính đến các biến số có thể ảnh hưởng đến biến mục tiêu, và xem xét mối quan hệ giữa các kỹ thuật hợp nhất và sai số trong dự đoán. Bằng cách kết hợp các điểm mạnh của nhiều mô hình và kỹ thuật học máy, với mục tiêu đạt được hiệu suất dự đoán tốt hơn và có cái nhìn sâu hơn về các mẫu và xu hướng cơ bản trong dữ liệu.

PHƯƠNG PHÁP

1. Xác định dữ liệu (identify the data)
2. Tiền xử lý dữ liệu: chuẩn bị tập dữ liệu chuỗi thời gian để phân tích
3. Trực quan hóa dữ liệu
4. Kiểm tra tính dừng và xác định bậc sai phân: nếu chuỗi thời gian được xác định là không ổn định, bước tiếp theo là xác định bậc của sự khác biệt cần thiết để đạt được tính ổn định
5. Xác định bậc của các thuật ngữ autoregressive (AR) và moving average (MA) trong mô hình ARIMA, được thực hiện bằng cách phân tích các hàm tương quan riêng của chuỗi thời gian
6. Lập mô hình ARIMA, SARIMAX
7. Kiểm tra mô hình: đánh giá hiệu suất của mô hình bằng cách kiểm tra chuẩn đoán
8. Tìm mô hình ARIMA tối ưu bằng cách sử dụng xác thực chéo ngoài thời gian
9. Kết luận

MÔ TẢ TẬP DỮ LIỆU (DATASET DESCRIPTION)

- Với tập dữ liệu này, nhóm sẽ dự đoán doanh số bán hàng cho hàng nghìn các mặt hàng sản phẩm được bán tại các cửa hàng Favorita tại Ecuador.
- Dữ liệu huấn luyện bao gồm thông tin về ngày tháng, cửa hàng và sản phẩm, thông tin về việc sản phẩm đó có được khuyến mãi không, cùng với các con số về doanh số bán hàng.

UPLOAD CÁC TỆP DỮ LIỆU CẦN THIẾT

```
In [ ]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.rcParams.update({'figure.figsize':(7,3), 'figure.dpi':120})
```

```
In [ ]: holidays = open(r"C:\Users\Quynh Nhu\Downloads\holidays_events.csv")
holidays = pd.read_csv(holidays)
holidays
```

Out[]:

	date	type	locale	locale_name	description	transferred
0	2012-03-02	Holiday	Local	Manta	Fundacion de Manta	False
1	2012-04-01	Holiday	Regional	Cotopaxi	Provincializacion de Cotopaxi	False
2	2012-04-12	Holiday	Local	Cuenca	Fundacion de Cuenca	False
3	2012-04-14	Holiday	Local	Libertad	Canionizacion de Libertad	False
4	2012-04-21	Holiday	Local	Riobamba	Canionizacion de Riobamba	False
...
345	2017-12-22	Additional	National	Ecuador	Navidad-3	False
346	2017-12-23	Additional	National	Ecuador	Navidad-2	False
347	2017-12-24	Additional	National	Ecuador	Navidad-1	False
348	2017-12-25	Holiday	National	Ecuador	Navidad	False
349	2017-12-26	Additional	National	Ecuador	Navidad+1	False

350 rows × 6 columns

_Thông tin về holidays_events.csv

- Các ngày lễ và sự kiện, kèm theo thông tin bổ sung

```
In [ ]: oil = open(r"C:\Users\Quynh Nhu\Downloads\oil.csv")
oil = pd.read_csv(oil)
oil
```

Out[]:

	date	dcoilwtico
0	2013-01-01	NaN
1	2013-01-02	93.14
2	2013-01-03	92.97
3	2013-01-04	93.12
4	2013-01-07	93.20
...
1213	2017-08-25	47.65
1214	2017-08-28	46.40
1215	2017-08-29	46.46
1216	2017-08-30	45.96
1217	2017-08-31	47.26

1218 rows × 2 columns

-Thông tin về oil.csv

- Giá dầu hàng ngày. Bao gồm các giá trị trong cả khoảng thời gian của dữ liệu huấn luyện và kiểm tra. (Ecuador là một quốc gia dựa vào dầu và tình hình kinh tế của nó rất dễ bị ảnh hưởng bởi biến động trong giá dầu)

In []:

```
sample_submission = open(r"C:\Users\Quynh Nhu\Downloads\sample_submission.csv")
sample_submission = pd.read_csv(sample_submission)
sample_submission
```

Out[]:

	id	sales
0	3000888	0.0
1	3000889	0.0
2	3000890	0.0
3	3000891	0.0
4	3000892	0.0
...
28507	3029395	0.0
28508	3029396	0.0
28509	3029397	0.0
28510	3029398	0.0
28511	3029399	0.0

28512 rows × 2 columns

-Thông tin về sample_submission.csv

- Một tệp gửi mẫu ở đúng định dạng, gồm có id và sales

```
In [ ]: stores = open(r"C:\Users\Quynh Nhu\Downloads\stores.csv")
stores = pd.read_csv(stores)
stores
```

Out[]:	store_nbr	city	state	type	cluster
0	1	Quito	Pichincha	D	13
1	2	Quito	Pichincha	D	13
2	3	Quito	Pichincha	D	8
3	4	Quito	Pichincha	D	9
4	5	Santo Domingo	Santo Domingo de los Tsachilas	D	4
5	6	Quito	Pichincha	D	13
6	7	Quito	Pichincha	D	8
7	8	Quito	Pichincha	D	8
8	9	Quito	Pichincha	B	6
9	10	Quito	Pichincha	C	15
10	11	Cayambe	Pichincha	B	6
11	12	Latacunga	Cotopaxi	C	15
12	13	Latacunga	Cotopaxi	C	15
13	14	Riobamba	Chimborazo	C	7
14	15	Ibarra	Imbabura	C	15
15	16	Santo Domingo	Santo Domingo de los Tsachilas	C	3
16	17	Quito	Pichincha	C	12
17	18	Quito	Pichincha	B	16
18	19	Guaranda	Bolivar	C	15
19	20	Quito	Pichincha	B	6
20	21	Santo Domingo	Santo Domingo de los Tsachilas	B	6
21	22	Puyo	Pastaza	C	7
22	23	Ambato	Tungurahua	D	9
23	24	Guayaquil	Guayas	D	1
24	25	Salinas	Santa Elena	D	1
25	26	Guayaquil	Guayas	D	10
26	27	Daule	Guayas	D	1
27	28	Guayaquil	Guayas	E	10
28	29	Guayaquil	Guayas	E	10
29	30	Guayaquil	Guayas	C	3
30	31	Babahoyo	Los Rios	B	10
31	32	Guayaquil	Guayas	C	3
32	33	Quevedo	Los Rios	C	3
33	34	Guayaquil	Guayas	B	6

store_nbr	city	state	type	cluster
34	Playas	Guayas	C	3
35	Libertad	Guayas	E	10
36	Cuenca	Azuay	D	2
37	Loja	Loja	D	4
38	Cuenca	Azuay	B	6
39	Machala	El Oro	C	3
40	Machala	El Oro	D	4
41	Cuenca	Azuay	D	2
42	Esmeraldas	Esmeraldas	E	10
43	Quito	Pichincha	A	5
44	Quito	Pichincha	A	11
45	Quito	Pichincha	A	14
46	Quito	Pichincha	A	14
47	Quito	Pichincha	A	14
48	Quito	Pichincha	A	11
49	Ambato	Tungurahua	A	14
50	Guayaquil	Guayas	A	17
51	Manta	Manabi	A	11
52	Manta	Manabi	D	13
53	El Carmen	Manabi	C	3

_Thông tin về stores.csv

- Dữ liệu về cửa hàng, bao gồm thông tin về thành phố, bang, loại hình và nhóm (cluster) của cửa hàng
- Cluster là một nhóm các cửa hàng tương tự

```
In [ ]: test = open(r"C:\Users\Quynh Nhu\Downloads\test.csv")
test = pd.read_csv(test)
test
```

Out[]:

	id	date	store_nbr	family	onpromotion
0	3000888	2017-08-16	1	AUTOMOTIVE	0
1	3000889	2017-08-16	1	BABY CARE	0
2	3000890	2017-08-16	1	BEAUTY	2
3	3000891	2017-08-16	1	BEVERAGES	20
4	3000892	2017-08-16	1	BOOKS	0
...
28507	3029395	2017-08-31	9	POULTRY	1
28508	3029396	2017-08-31	9	PREPARED FOODS	0
28509	3029397	2017-08-31	9	PRODUCE	1
28510	3029398	2017-08-31	9	SCHOOL AND OFFICE SUPPLIES	9
28511	3029399	2017-08-31	9	SEAFOOD	0

28512 rows × 5 columns

_Thông tin về test.csv

- Dữ liệu kiểm tra, có các tính năng giống với dữ liệu huấn luyện. Nhóm sẽ dự đoán doanh số bán hàng mục tiêu cho các ngày trong tệp này
- Các ngày trong dữ liệu kiểm tra là cho 15 ngày sau ngày cuối cùng trong dữ liệu huấn luyện

In []:

```
train = open(r"C:\Users\Quynh Nhu\Downloads\train.csv\train.csv")
train = pd.read_csv(train)
train
```

Out[]:

	id	date	store_nbr	family	sales	onpromotion
0	0	2013-01-01	1	AUTOMOTIVE	0.000	0
1	1	2013-01-01	1	BABY CARE	0.000	0
2	2	2013-01-01	1	BEAUTY	0.000	0
3	3	2013-01-01	1	BEVERAGES	0.000	0
4	4	2013-01-01	1	BOOKS	0.000	0
...
3000883	3000883	2017-08-15	9	POULTRY	438.133	0
3000884	3000884	2017-08-15	9	PREPARED FOODS	154.553	1
3000885	3000885	2017-08-15	9	PRODUCE	2419.729	148
3000886	3000886	2017-08-15	9	SCHOOL AND OFFICE SUPPLIES	121.000	8
3000887	3000887	2017-08-15	9	SEAFOOD	16.000	0

3000888 rows × 6 columns

_Thông tin về train.csv

- Dữ liệu huấn luyện, bao gồm chuỗi thời gian về các thông tin cửa hàng (store_nbr), các mặt hàng cùng loại (family), và thông tin về việc sản phẩm đó có được khuyến mãi (onpromotion), cũng như mục tiêu là doanh số bán hàng (sales)
- store_nbr xác định loại sản phẩm được bán
- sales cho biết tổng doanh số bán hàng cho một họ hàng sản phẩm tại một cửa hàng cụ thể tại một ngày cụ thể. Giá trị có thể là số thập phân vì sản phẩm có thể đúp wjc bán theo đơn vị thập phân
- onpromotion cho biết tổng số sản phẩm trong family đã được khuyến mãi tại một cửa hàng vào một ngày cụ thể

In []:

```
transactions = open(r"C:\Users\Quynh Nhu\Downloads\transactions.csv\transactions.csv")
transactions = pd.read_csv(transactions)
transactions
```

Out[]:

	date	store_nbr	transactions
0	2013-01-01	25	770
1	2013-01-02	1	2111
2	2013-01-02	2	2358
3	2013-01-02	3	3487
4	2013-01-02	4	1922
...
83483	2017-08-15	50	2804
83484	2017-08-15	51	1573
83485	2017-08-15	52	2255
83486	2017-08-15	53	932
83487	2017-08-15	54	802

83488 rows × 3 columns

1. Xác định dữ liệu (Identify the data)

- Holidays_events

In []:

```
holidays.info()
holidays.isnull().sum()
columns_to_display = ['transferred', 'type', 'locale']

for col in columns_to_display:
    unique_counts = holidays[col].value_counts()

    print(f"Unique value counts for '{col}':")
    unique_counts
    print('\n')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 350 entries, 0 to 349
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   date        350 non-null    object  
 1   type         350 non-null    object  
 2   locale       350 non-null    object  
 3   locale_name  350 non-null    object  
 4   description   350 non-null    object  
 5   transferred   350 non-null    bool    
dtypes: bool(1), object(5)
memory usage: 14.1+ KB
Unique value counts for 'transferred':
```

Unique value counts for 'type':

Unique value counts for 'locale':

In []: `holidays.describe()`

Out[]:

	date	type	locale	locale_name	description	transferred
count	350	350	350	350	350	350
unique	312	6	3	24	103	2
top	2014-06-25	Holiday	National	Ecuador	Carnaval	False
freq	4	221	174	174	10	338

- oil

In []: `oil.info()`
`oil.isnull().sum()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1218 entries, 0 to 1217
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   date        1218 non-null    object  
 1   dcoilwtico  1175 non-null    float64 
dtypes: float64(1), object(1)
memory usage: 19.2+ KB
```

Out[]:

```
date          0
dcoilwtico  43
dtype: int64
```

In []: `oil.describe()`

Out[]:

dcoilwtico	
count	1175.000000
mean	67.714366
std	25.630476
min	26.190000
25%	46.405000
50%	53.190000
75%	95.660000
max	110.620000

- Stores

In []:

```
stores.info()
stores.isnull().sum()

columns_to_display = ['city', 'state', 'type']

for col in columns_to_display:
    unique_counts = stores[col].value_counts()

    print(f"Unique value counts for '{col}':")
    print(unique_counts)
    print('\n')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54 entries, 0 to 53
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   store_nbr   54 non-null    int64  
 1   city        54 non-null    object  
 2   state       54 non-null    object  
 3   type        54 non-null    object  
 4   cluster     54 non-null    int64  
dtypes: int64(2), object(3)
memory usage: 2.2+ KB
Unique value counts for 'city':
city
Quito          18
Guayaquil      8
Cuenca         3
Santo Domingo  3
Manta          2
Latacunga      2
Machala         2
Ambato          2
Quevedo         1
Esmeraldas     1
Loja            1
Libertad        1
Playas          1
Daule           1
Babahoyo        1
Salinas         1
Puyo            1
Guaranda        1
Ibarra          1
Riobamba        1
Cayambe         1
El Carmen       1
Name: count, dtype: int64
```

Unique value counts for 'state':

```
state
Pichincha      19
Guayas          11
Santo Domingo de los Tsachilas  3
Azuay           3
Manabi          3
Cotopaxi        2
Tungurahua      2
Los Rios         2
El Oro           2
Chimborazo      1
Imbabura         1
Bolivar          1
Pastaza          1
Santa Elena      1
Loja             1
Esmeraldas      1
Name: count, dtype: int64
```

```
Unique value counts for 'type':  
type  
D    18  
C    15  
A     9  
B     8  
E     4  
Name: count, dtype: int64
```

```
In [ ]: stores.describe()
```

```
Out[ ]:      store_nbr    cluster  
count  54.000000  54.000000  
mean   27.500000  8.481481  
std    15.732133  4.693395  
min    1.000000  1.000000  
25%   14.250000  4.000000  
50%   27.500000  8.500000  
75%   40.750000  13.000000  
max   54.000000  17.000000
```

- transactions

```
In [ ]: transactions.info()  
transactions.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 83488 entries, 0 to 83487  
Data columns (total 3 columns):  
 #   Column       Non-Null Count  Dtype    
---  --    
 0   date         83488 non-null   object  
 1   store_nbr    83488 non-null   int64  
 2   transactions 83488 non-null   int64  
dtypes: int64(2), object(1)  
memory usage: 1.9+ MB
```

```
Out[ ]: date          0  
store_nbr      0  
transactions    0  
dtype: int64
```

```
In [ ]: transactions.describe()
```

Out[]:

	store_nbr	transactions
count	83488.000000	83488.000000
mean	26.939237	1694.602158
std	15.608204	963.286644
min	1.000000	5.000000
25%	13.000000	1046.000000
50%	27.000000	1393.000000
75%	40.000000	2079.000000
max	54.000000	8359.000000

- test

In []:

```
test.info()
test.isnull().sum()

columns_to_display = ['family', 'store_nbr']

for col in columns_to_display:
    unique_counts = test[col].value_counts()

    print(f"Unique counts for '{col}':")
    print(unique_counts)
    print("\n")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28512 entries, 0 to 28511
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   id          28512 non-null   int64  
 1   date         28512 non-null   object  
 2   store_nbr    28512 non-null   int64  
 3   family       28512 non-null   object  
 4   onpromotion  28512 non-null   int64  
dtypes: int64(3), object(2)
memory usage: 1.1+ MB
Unique for counts for 'family':
family
AUTOMOTIVE           864
HOME APPLIANCES      864
SCHOOL AND OFFICE SUPPLIES 864
PRODUCE               864
PREPARED FOODS       864
POULTRY               864
PLAYERS AND ELECTRONICS 864
PET SUPPLIES          864
PERSONAL CARE         864
MEATS                 864
MAGAZINES              864
LIQUOR,WINE,BEER      864
LINGERIE               864
LAWN AND GARDEN        864
LADIESWEAR             864
HOME CARE               864
HOME AND KITCHEN II    864
BABY CARE               864
HOME AND KITCHEN I     864
HARDWARE               864
GROCERY II              864
GROCERY I               864
FROZEN FOODS            864
EGGS                   864
DELI                    864
DAIRY                   864
CLEANING                864
CELEBRATION              864
BREAD/BAKERY             864
BOOKS                   864
BEVERAGES               864
BEAUTY                  864
SEAFOOD                 864
Name: count, dtype: int64
```

```
Unique for counts for 'store_nbr':
store_nbr
1      528
46     528
36     528
37     528
38     528
39     528
4      528
40     528
```

```
41    528
42    528
43    528
44    528
45    528
47    528
10    528
48    528
49    528
5     528
50    528
51    528
52    528
53    528
54    528
6     528
7     528
8     528
35    528
34    528
33    528
32    528
11    528
12    528
13    528
14    528
15    528
16    528
17    528
18    528
19    528
2     528
20    528
21    528
22    528
23    528
24    528
25    528
26    528
27    528
28    528
29    528
3     528
30    528
31    528
9     528
Name: count, dtype: int64
```

```
In [ ]: test.describe()
```

Out[]:

	id	store_nbr	onpromotion
count	2.851200e+04	28512.000000	28512.000000
mean	3.015144e+06	27.500000	6.965383
std	8.230850e+03	15.586057	20.683952
min	3.000888e+06	1.000000	0.000000
25%	3.008016e+06	14.000000	0.000000
50%	3.015144e+06	27.500000	0.000000
75%	3.022271e+06	41.000000	6.000000
max	3.029399e+06	54.000000	646.000000

- train

In []:

```
train.info()
train.isnull().sum()

columns_to_display = ['family', 'store_nbr']

for col in columns_to_display:
    unique_counts = train[col].value_counts()
    print(f"Unique value counts for '{col}':")
    print(unique_counts)
    print('\n')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000888 entries, 0 to 3000887
Data columns (total 6 columns):
 #   Column      Dtype  
 --- 
 0   id          int64  
 1   date         object  
 2   store_nbr    int64  
 3   family        object  
 4   sales        float64 
 5   onpromotion  int64  
dtypes: float64(1), int64(3), object(2)
memory usage: 137.4+ MB
Unique value counts for 'family':
family
AUTOMOTIVE           90936
HOME APPLIANCES      90936
SCHOOL AND OFFICE SUPPLIES 90936
PRODUCE               90936
PREPARED FOODS       90936
POULTRY                90936
PLAYERS AND ELECTRONICS 90936
PET SUPPLIES          90936
PERSONAL CARE          90936
MEATS                  90936
MAGAZINES              90936
LIQUOR,WINE,BEER      90936
LINGERIE                90936
LAWN AND GARDEN        90936
LADIESWEAR              90936
HOME CARE                90936
HOME AND KITCHEN II     90936
BABY CARE                90936
HOME AND KITCHEN I      90936
HARDWARE                90936
GROCERY II              90936
GROCERY I                90936
FROZEN FOODS            90936
EGGS                     90936
DELI                      90936
DAIRY                      90936
CLEANING                 90936
CELEBRATION               90936
BREAD/BAKERY              90936
BOOKS                      90936
BEVERAGES                 90936
BEAUTY                      90936
SEAFOOD                    90936
Name: count, dtype: int64
```

```
Unique value counts for 'store_nbr':
store_nbr
1      55572
46     55572
36     55572
37     55572
38     55572
39     55572
4      55572
```

```
40    55572
41    55572
42    55572
43    55572
44    55572
45    55572
47    55572
10    55572
48    55572
49    55572
5     55572
50    55572
51    55572
52    55572
53    55572
54    55572
6     55572
7     55572
8     55572
35    55572
34    55572
33    55572
32    55572
11    55572
12    55572
13    55572
14    55572
15    55572
16    55572
17    55572
18    55572
19    55572
2     55572
20    55572
21    55572
22    55572
23    55572
24    55572
25    55572
26    55572
27    55572
28    55572
29    55572
3     55572
30    55572
31    55572
9     55572
Name: count, dtype: int64
```

```
In [ ]: train.describe()
```

Out[]:

	id	store_nbr	sales	onpromotion
count	3.000888e+06	3.000888e+06	3.000888e+06	3.000888e+06
mean	1.500444e+06	2.750000e+01	3.577757e+02	2.602770e+00
std	8.662819e+05	1.558579e+01	1.101998e+03	1.221888e+01
min	0.000000e+00	1.000000e+00	0.000000e+00	0.000000e+00
25%	7.502218e+05	1.400000e+01	0.000000e+00	0.000000e+00
50%	1.500444e+06	2.750000e+01	1.100000e+01	0.000000e+00
75%	2.250665e+06	4.100000e+01	1.958473e+02	0.000000e+00
max	3.000887e+06	5.400000e+01	1.247170e+05	7.410000e+02

2. Tiền xử lý dữ liệu (Data Preprocessing)

In []:

```
import numpy as np
import pandas as pd
import datetime
import math
from collections import defaultdict
import itertools
from scipy.stats import shapiro
import scipy.stats as stats
import matplotlib.pyplot as plt
import seaborn as sns

import os
from sklearn.compose import make_column_selector as selector

#Configurations
from warnings import simplefilter
simplefilter("ignore") # ignore warnings to clean up output cells
pd.set_option('display.max_columns', None)
pd.options.display.float_format = '{:.2f}'.format
```

In []:

```
#store_nbr ordered giống như file test và train
store_nbrs = train['store_nbr'].unique()
store_nbrs = sorted(map(str, store_nbrs))
store_nbrs = list(map(int, store_nbrs))

#family của products có cùng thứ tự trong tập train và test
family_values = train['family'].unique()
family_values = sorted(family_values)
```

In []:

```
# chuyển đổi các cột ngày tháng trong các DataFrame thành định dạng datetime (thời gian)
holidays['date'] = pd.to_datetime(holidays['date'], format="%Y-%m-%d")
oil['date'] = pd.to_datetime(oil['date'], format="%Y-%m-%d")
transactions['date'] = pd.to_datetime(transactions['date'], format="%Y-%m-%d")
train['date'] = pd.to_datetime(train['date'], format="%Y-%m-%d")
test['date'] = pd.to_datetime(test['date'], format="%Y-%m-%d")
```

```
In [ ]: # Xử lý dữ liệu ngày bị mất trong dữ liệu  
# xử lý ở train-data
```

```
train_start = train.date.min().date()  
train_end = train.date.max().date()  
  
missing_dates = pd.date_range(train_start, train_end).difference(train.date.unique())  
missing_dates = missing_dates.strftime("%Y-%m-%d").tolist()  
  
missing_dates
```

```
Out[ ]: ['2013-12-25', '2014-12-25', '2015-12-25', '2016-12-25']
```

```
In [ ]: # ta giả sử các cửa hàng đã đóng cửa vào ngày này và do đó ta có thể điền dữ liệu này
```

```
multi_index = pd.MultiIndex.from_product([pd.date_range(train_start, train_end),  
train.store_nbr.unique(),  
train.family.unique()],  
names=['date','store_nbr','family'])  
  
train = train.set_index(['date','store_nbr','family']).reindex(multi_index).reset_index()  
  
# điền dữ liệu bị thiếu với 0  
train[['sales','onpromotion']] = train[['sales','onpromotion']].fillna(0.)  
train.id = train.id.interpolate(method='linear')  
train
```

```
Out[ ]:
```

	date	store_nbr	family	id	sales	onpromotion
0	2013-01-01	1	AUTOMOTIVE	0.00	0.00	0.00
1	2013-01-01	1	BABY CARE	1.00	0.00	0.00
2	2013-01-01	1	BEAUTY	2.00	0.00	0.00
3	2013-01-01	1	BEVERAGES	3.00	0.00	0.00
4	2013-01-01	1	BOOKS	4.00	0.00	0.00
...
3008011	2017-08-15	9	POULTRY	3000883.00	438.13	0.00
3008012	2017-08-15	9	PREPARED FOODS	3000884.00	154.55	1.00
3008013	2017-08-15	9	PRODUCE	3000885.00	2419.73	148.00
3008014	2017-08-15	9	SCHOOL AND OFFICE SUPPLIES	3000886.00	121.00	8.00
3008015	2017-08-15	9	SEAFOOD	3000887.00	16.00	0.00

3008016 rows × 6 columns

```
In [ ]: # xử lý giá trị null ở oil
```

```
oil = oil.merge(pd.DataFrame({'date': pd.date_range(train_start, train_end)}), on='date')  
  
oil.dcoilwtico = oil.dcoilwtico.interpolate(method='linear', limit_direction='both')  
oil.isnull().sum()
```

```
Out[ ]: date      0  
dcoilwtico  0  
dtype: int64
```

```
In [ ]: oil
```

```
Out[ ]:    date  dcoilwtico  
0   2013-01-01     93.14  
1   2013-01-02     93.14  
2   2013-01-03     92.97  
3   2013-01-04     93.12  
4   2013-01-05     93.15  
...       ...      ...  
1695  2017-08-25     47.65  
1696  2017-08-28     46.40  
1697  2017-08-29     46.46  
1698  2017-08-30     45.96  
1699  2017-08-31     47.26
```

1700 rows × 2 columns

3. Trực quan hóa dữ liệu

```
In [ ]: print('Training Data Shape: ', train.shape)  
print('Testing Data Shape: ', test.shape)
```

```
Training Data Shape: (3008016, 6)  
Testing Data Shape: (28512, 5)
```

- Kết hợp dữ liệu về một DataFrame duy nhất

```
In [ ]: train1 = train.merge(oil, on = 'date', how = 'left')  
train1 = train1.merge(holidays, on='date', how='left')  
train1 = train1.merge(stores, on = 'store_nbr', how='left')  
train1 = train1.merge(transactions, on = ['date','store_nbr'], how='left')  
train1 = train1.rename(columns= {'type_x' : 'holiday_type','type_y' : 'store_type'})  
  
test1 = test.merge(oil, on = 'date', how = 'left')  
test1 = test1.merge(holidays, on='date', how='left')  
test1 = test1.merge(stores, on = 'store_nbr', how='left')  
test1 = test1.merge(transactions, on = ['date','store_nbr'], how='left')  
test1 = test1.rename(columns= {'type_x' : 'holiday_type','type_y' : 'store_type'})
```

```
In [ ]: train1
```

Out[]:

		date	store_nbr	family	id	sales	onpromotion	dcoilwtico	holiday_type
0		2013-01-01	1	AUTOMOTIVE	0.00	0.00	0.00	93.14	Holiday
1		2013-01-01	1	BABY CARE	1.00	0.00	0.00	93.14	Holiday
2		2013-01-01	1	BEAUTY	2.00	0.00	0.00	93.14	Holiday
3		2013-01-01	1	BEVERAGES	3.00	0.00	0.00	93.14	Holiday
4		2013-01-01	1	BOOKS	4.00	0.00	0.00	93.14	Holiday
...	
3061471		2017-08-15	9	POULTRY	3000883.00	438.13	0.00	47.57	Holiday
3061472		2017-08-15	9	PREPARED FOODS	3000884.00	154.55	1.00	47.57	Holiday
3061473		2017-08-15	9	PRODUCE	3000885.00	2419.73	148.00	47.57	Holiday
3061474		2017-08-15	9	SCHOOL AND OFFICE SUPPLIES	3000886.00	121.00	8.00	47.57	Holiday
3061475		2017-08-15	9	SEAFOOD	3000887.00	16.00	0.00	47.57	Holiday

3061476 rows × 17 columns

In []: plt.figure(figsize=(20,12), dpi= 120)

```

plt.subplot(2,3,1)
sales_city = train1.groupby(['city'])['sales'].sum()
sales_city.plot.bar()
plt.title('Sales Depends on city')

plt.subplot(2,3,2)
sales_state = train1.groupby(['state'])['sales'].sum()
sales_state.plot.bar()
plt.title('Sales depends on State')

plt.subplot(2,3,3)
sales_state = train1.groupby(['store_type'])['sales'].sum()
sales_state.plot.bar()
plt.title('Sales depends on Store_type')

plt.subplot(2,3,4)
sales_state = train1.groupby(['cluster'])['sales'].sum()

```

```

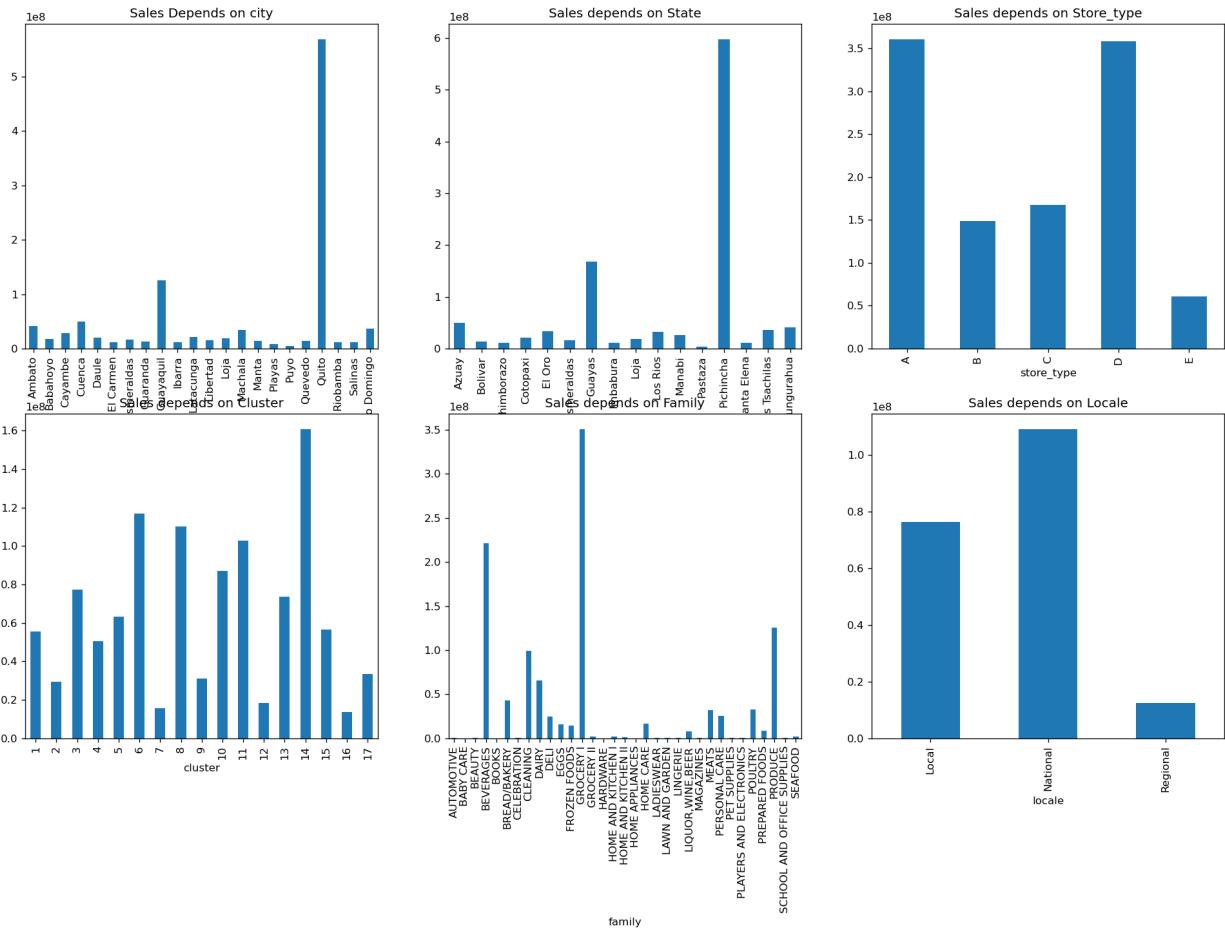
sales_state.plot.bar()
plt.title('Sales depends on Cluster')

plt.subplot(2,3,5)
sales_state = train1.groupby(['family'])['sales'].sum()
sales_state.plot.bar()
plt.title('Sales depends on Family')

plt.subplot(2,3,6)
sales_state = train1.groupby(['locale'])['sales'].sum()
sales_state.plot.bar()
plt.title('Sales depends on Locale')

```

Out[]: Text(0.5, 1.0, 'Sales depends on Locale')



In []: `import plotly.express as px`

```

temp = pd.merge(train1.groupby(["date", "store_nbr"]).sales.sum().reset_index(), trans
print("Spearman Correlation between Total Sales and Transactions: {:.4f}".format(temp
px.line(transactions.sort_values(["store_nbr", "date"]), x='date', y='transactions', c

```

Spearman Correlation between Total Sales and Transactions: 0.8095

In []: `a = train1.copy()`

```

a["year"] = a.date.dt.year
a["month"] = a.date.dt.month
px.box(a, x="year", y="transactions", color = "month", title = "Transactions")

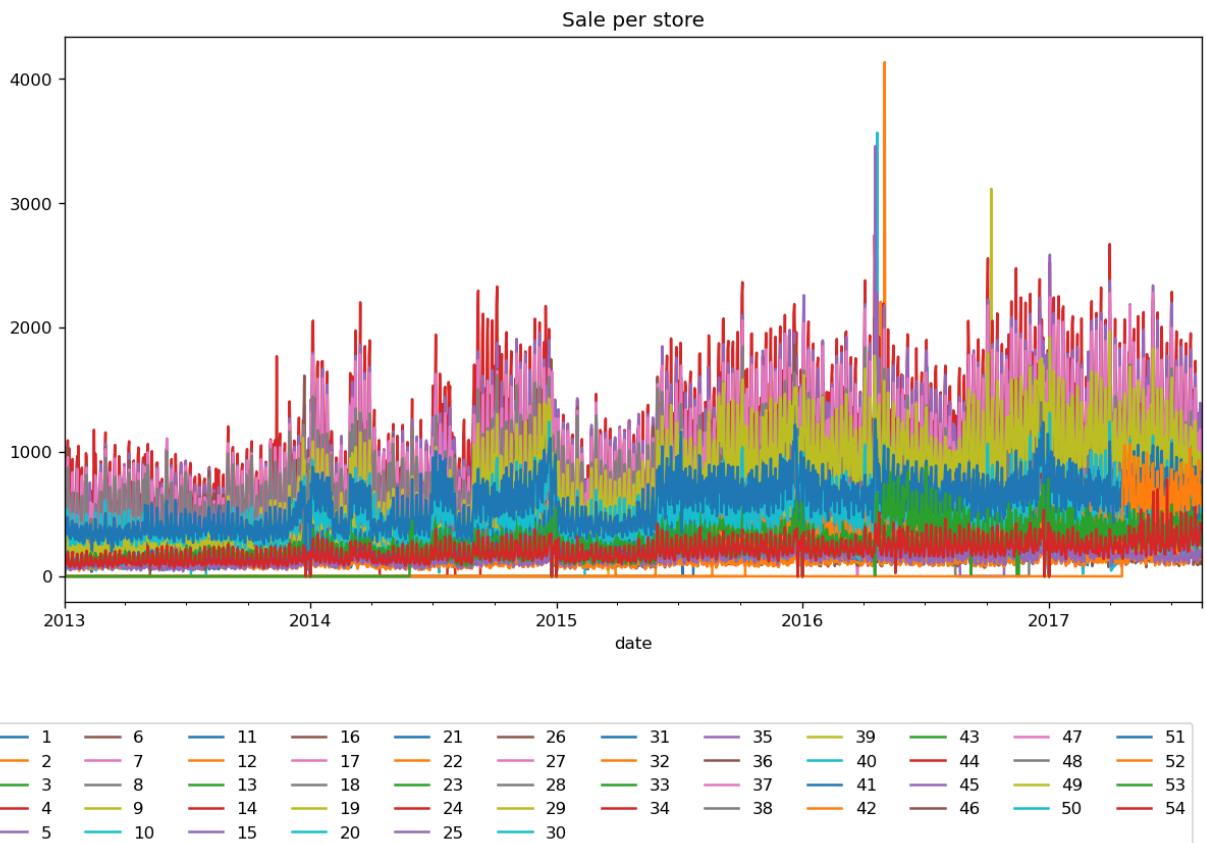
```

Có một mô hình ổn định trong Giao dịch. Tất cả các tháng đều giống nhau ngoại trừ tháng 12 từ năm 2013 đến năm 2017 theo boxplot. Ngoài ra, chúng tôi vừa thấy mô hình tương tự cho mỗi cửa hàng trong lô trước. Doanh số bán hàng tại cửa hàng luôn tăng vào dịp cuối năm.

```
In [ ]: a = transactions.set_index("date").resample("M").transactions.mean().reset_index()
        a["year"] = a.date.dt.year
        px.line(a, x='date', y='transactions', color='year', title = "Monthly Average Transacti
```

Khi nhìn vào mối quan hệ của chúng, chúng ta có thể thấy rằng cũng có mối tương quan cao giữa tổng doanh số bán hàng và giao dịch.

```
In [ ]: sales_store = train1.groupby(['date', 'store_nbr']).sales.mean().reset_index()
sales_store = sales_store.pivot(index='date', columns='store_nbr', values='sales')
sales_store.plot(figsize=(12,6))
plt.title('Sale per store', fontsize=12)
plt.legend(bbox_to_anchor=(1, -.2), ncol=12)
plt.show()
```



- Tôi nhận ra một số hàng không cần thiết trong dữ liệu khi tôi đang xem từng chuỗi thời gian của các cửa hàng. Nếu bạn chọn các cửa hàng ở trên, một số cửa hàng không có doanh số bán hàng vào đầu năm 2013. Bạn có thể thấy chúng, nếu bạn nhìn vào các cửa hàng 20, 21, 22, 29, 36, 42, 52 và 53. Tôi đã quyết định để xóa những hàng đó trước khi cửa hàng mở cửa. Trong các mã sau, chúng tôi sẽ loại bỏ chúng.

- Qua đây, có thể thấy sự bất thường của dữ liệu vào năm 2016. Bởi vì có một trận động đất mạnh 7,8 độ richter xảy ra ở Ecuador vào ngày 16 tháng 4 năm 2016. Người dân tập hợp lại trong nỗ lực cứu trợ quyên góp nước và các sản phẩm cần thiết đầu tiên khác, điều này đã ảnh hưởng lớn đến doanh số bán hàng của siêu thị trong vài tuần sau trận động đất.

```
In [ ]: print(train1.shape)
train1 = train1[~((train1.store_nbr == 52) & (train1.date < "2017-04-20"))]
train1 = train1[~((train1.store_nbr == 22) & (train1.date < "2015-10-09"))]
train1 = train1[~((train1.store_nbr == 42) & (train1.date < "2015-08-21"))]
train1 = train1[~((train1.store_nbr == 21) & (train1.date < "2015-07-24"))]
train1 = train1[~((train1.store_nbr == 29) & (train1.date < "2015-03-20"))]
train1 = train1[~((train1.store_nbr == 20) & (train1.date < "2015-02-13"))]
train1 = train1[~((train1.store_nbr == 53) & (train1.date < "2014-05-29"))]
train1 = train1[~((train1.store_nbr == 36) & (train1.date < "2013-05-09"))]
train1.shape

(3061476, 17)
Out[ ]: (2837241, 17)
```

```
In [ ]: agg = train1.groupby(['date', 'store_type']).agg({'transactions':'mean'}).reset_index()
fig = px.line(agg, x='date', y='transactions',color='store_type')
fig.update_layout(title = 'Average Transactions by Date and Store Type')
```

```
In [ ]: agg = train1.groupby(['date', 'family']).agg({'sales':'mean'}).reset_index()
fig = px.line(agg, x = 'date', y='sales',color='family')
fig.update_layout(title = 'Average Sales by Date and Family')
```

```
In [ ]: #xem xét các cột khác nhau và doanh thu trung bình có liên quan của chúng
from plotly.subplots import make_subplots
import plotly.graph_objects as go
def vbar(col):
    temp = train1.groupby(col).agg({"sales" : "mean"}).reset_index()
    temp = temp.sort_values('sales', ascending = False)
    c = {
        'x' : list(temp['sales'])[:15][::-1],
        'y' : list(temp[col])[:15][::-1],
        'title' : "Average sales by "+col
    }
    trace = go.Bar(y=[str(_) + " " for _ in c['y']], x=c['x'], orientation="h", marker=True)
    return trace

    layout = go.Layout(title=c['title'],
                        paper_bgcolor='rgba(0,0,0,0)', plot_bgcolor='rgba(0,0,0,0)'
                        xaxis_title="", yaxis_title="", width=650)
    fig = go.Figure([trace], layout=layout)
    fig.update_xaxes(tickangle=45, tickfont=dict(color='crimson'))
    fig.update_yaxes(tickangle=0, tickfont=dict(color='crimson'))
    fig.show()

trace1 = vbar('family')
trace2 = vbar('store_type')
trace3 = vbar('state')
trace4 = vbar('city')

titles = ['Store Family', 'Store Type', 'State', 'City']
```

```

titles = ['Top ' + _ + " by Average Sales" for _ in titles]
fig = make_subplots(rows=2, cols=2, subplot_titles = titles)

fig.add_trace(trace1, row=1, col=1).add_trace(trace2, row=1, col=2).add_trace(trace3,

```

```

In [ ]: trace1 = vbar('cluster')
trace2 = vbar('store_nbr')

titles = ['Cluster Number', 'Store Number']
titles = ['Top ' + _ + " by Average Sales" for _ in titles]
fig = make_subplots(rows=1, cols=2, subplot_titles=titles)

fig.add_trace(trace1, row=1, col=1).add_trace(trace2, row=1, col=2).update_layout(heig

```

```

In [ ]: def create_ts_feature(df):
    df['date'] = pd.to_datetime(df['date'])
    df['dayofweek'] = df['date'].dt.dayofweek
    df['quarter'] = df['date'].dt.quarter
    df['month'] = df['date'].dt.month
    df['year'] = df['date'].dt.year
    df['dayofyear'] = df['date'].dt.dayofyear
    df['dayofmonth'] = df['date'].dt.day
    return df

train1 = create_ts_feature(train1)
test1 = create_ts_feature(test1)
train1.head()

```

Out[]:

	date	store_nbr	family	id	sales	onpromotion	dcoilwtico	holiday_type	locale	locale
0	2013-01-01	1	AUTOMOTIVE	0.00	0.00	0.00	93.14	Holiday	National	E
1	2013-01-01	1	BABY CARE	1.00	0.00	0.00	93.14	Holiday	National	E
2	2013-01-01	1	BEAUTY	2.00	0.00	0.00	93.14	Holiday	National	E
3	2013-01-01	1	BEVERAGES	3.00	0.00	0.00	93.14	Holiday	National	E
4	2013-01-01	1	BOOKS	4.00	0.00	0.00	93.14	Holiday	National	E

In []:

```

def hbar(col) :
    temp = train1.groupby(col).agg({'sales':'mean'}).reset_index()
    temp = temp.sort_values(col, ascending = False)
    c = {
        'y' : list(temp['sales']),
        'x' : list(temp[col]),
        'title' : 'Average sales by ' + col
    }

    trace = go.Bar(y=c['y'], x=c['x'], orientation='v', marker=dict(color="#bbe707"))
    return trace

```

```

layout = go.Layout(title=c['title'],
                    paper_bgcolor='white',
                    plot_bgcolor= 'white',
                    xaxis_title = '',
                    yaxis_title = '',
                    width=650)

fig = go.Figure([trace], layout).update_xaxes(tickangle=45, tickfont=dict(color='black'))
fig.show()

trace1 = hbar('dayofweek')
trace2 = hbar('dayofmonth')
trace3 = hbar('dayofyear')
trace4 = hbar('month')
trace5 = hbar('quarter')
trace6 = hbar('year')

titles = ['Day of Week', 'Day of Month', 'Day of Year', 'Month', 'Quarter', 'Year']
titles = ['Average Sales by ' + _ for _ in titles]
fig = make_subplots(rows=3, cols=2, subplot_titles=titles)

fig.add_trace(trace1, row=1, col=1).add_trace(trace2, row=1, col=2).add_trace(trace3,

```

```

In [ ]: train1['holiday_type'] = train1['holiday_type'].fillna('No Holiday/Event')
train1['holiday_type'].value_counts()

def convert_t0_size(x):
    if x < 50:
        return 6
    elif x < 100:
        return 10
    elif x < 150:
        return 15
    elif x < 250:
        return 18
    elif x < 300:
        return 24
    elif x < 500:
        return 30
    else:
        return 40

def bubble(col1, col2):
    vc = train1.groupby([col1, col2]).agg({'sales':'mean'}).reset_index()
    vc = vc.sort_values(col2)
    fig = px.scatter(vc, x=col1, y=col2, size='sales', color='sales', size_max=40).update_traces(marker_line_color='black', marker_line_width=1)

bubble('month','holiday_type')
bubble('month','store_type')

```

```

In [ ]: # Mối tương quan giữa giá dầu và hoạt động bán hàng/khuyến mãi/giao dịch
data_sales_oil = train1.copy()
data_sales_oil = data_sales_oil.groupby(['dcoilwtico'], as_index=False)[['sales']].mean()

plt.figure(figsize=(10,6))

```

```

sns.regplot(x=data_sales_oil.dcoilwtico, y=data_sales_oil.sales)
plt.ylabel('Average Sales')
plt.title('Average Sales vs. Oil Prices')
plt.show()

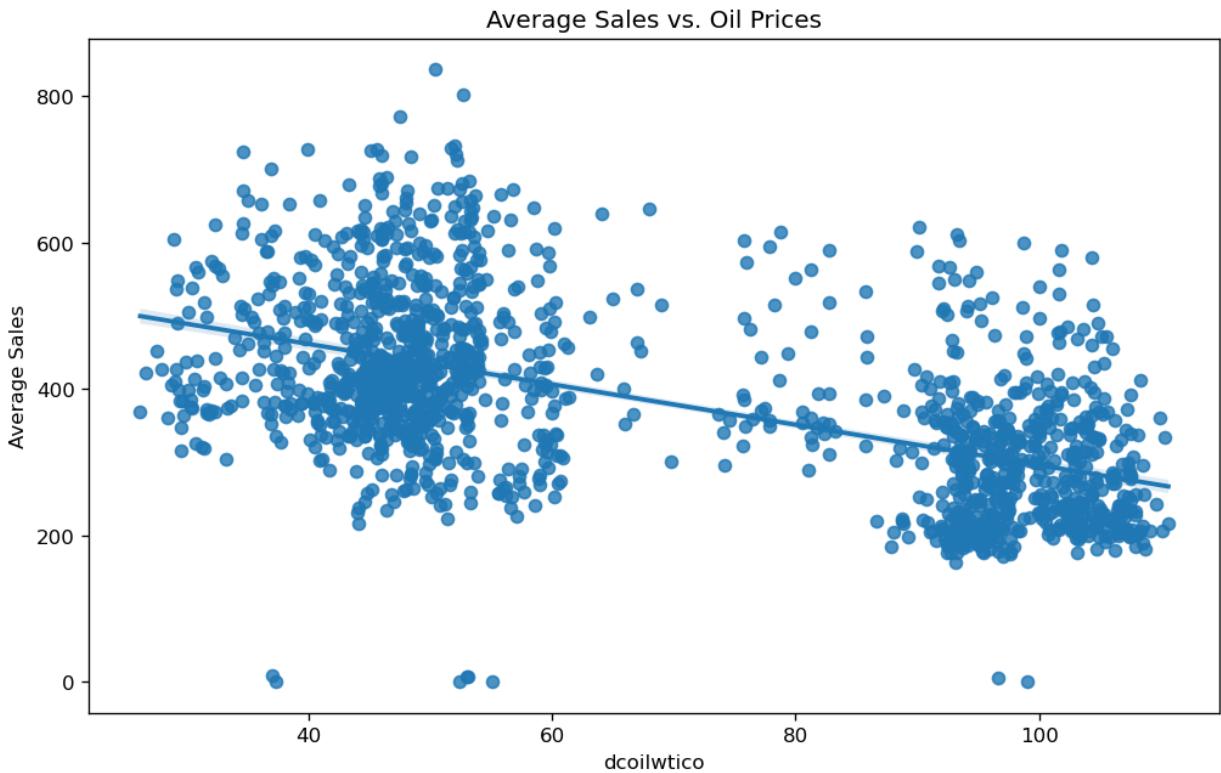
data_sales_transactions = train1.copy()
data_sales_transactions = data_sales_transactions.groupby(['transactions'], as_index=False).mean()

plt.figure(figsize=(10,6))
sns.regplot(x=data_sales_transactions.transactions, y=data_sales_transactions.sales)
plt.xlabel('Transactions')
plt.ylabel('Average Sales')
plt.title('Average Sales vs. Store Transactions')

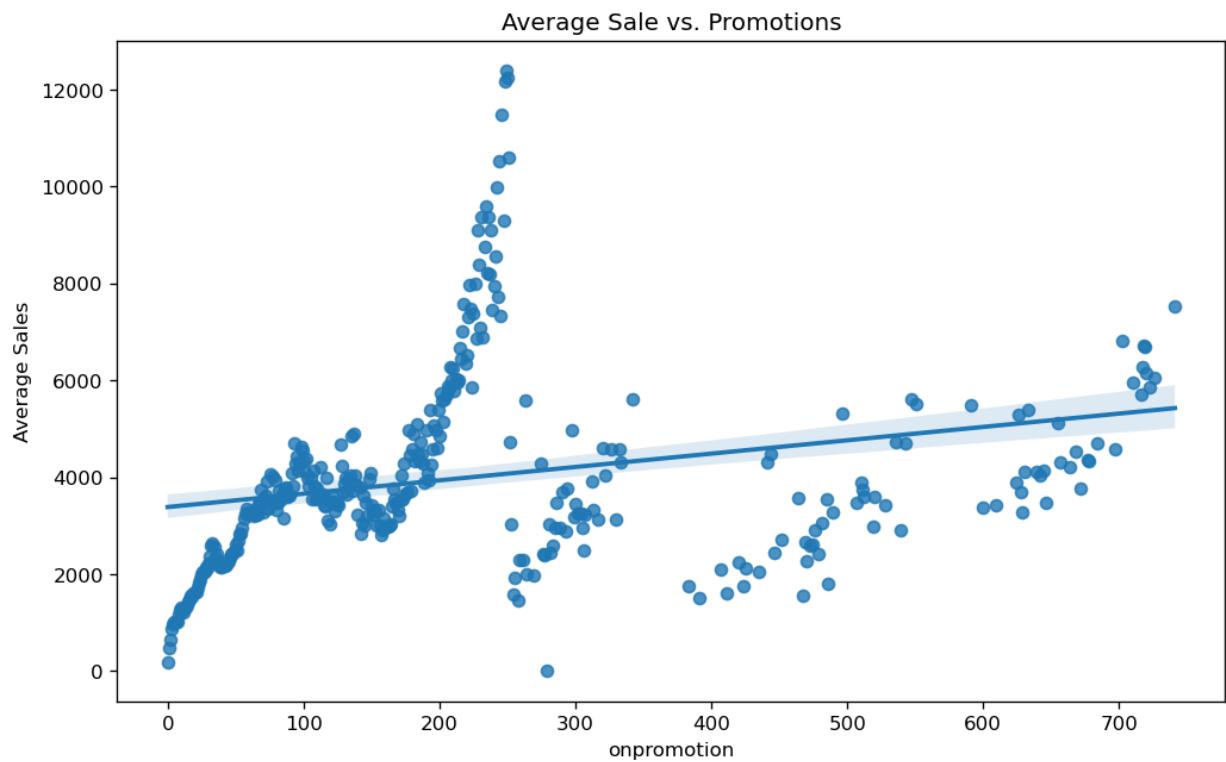
data_sales_prom = train1.copy()
data_sales_prom = data_sales_prom.groupby(['onpromotion'], as_index=False)['sales'].mean()

plt.figure(figsize=(10,6))
sns.regplot(x=data_sales_prom.onpromotion, y=data_sales_prom.sales)
plt.xlabel('onpromotion')
plt.ylabel('Average Sales')
plt.title('Average Sale vs. Promotions')

```



Out[]: Text(0.5, 1.0, 'Average Sale vs. Promotions')



- Chúng ta có thể nhận thấy rằng dầu càng đắt thì doanh số càng ít
- Có thể thấy rằng số lượng bán hàng tăng theo số lượng giao dịch, điều này khá được mong đợi
- Chúng ta có thể quan sát thấy doanh số bán hàng tăng lên khi số lượng sản phẩm được khuyến mại tăng lên

```
In [ ]: a = train1.sort_values(["store_nbr", "family", "date"])
for i in [20, 30, 45, 60, 90, 120, 365, 730]:
```

```

    a[ "SMA"+str(i)+"_sales_lag16"] = a.groupby(["store_nbr", "family"]).rolling(i).sales.agg("mean").shift(-i)
    a[ "SMA"+str(i)+"_sales_lag30"] = a.groupby(["store_nbr", "family"]).rolling(i).sales.agg("mean").shift(-i*3)
    a[ "SMA"+str(i)+"_sales_lag60"] = a.groupby(["store_nbr", "family"]).rolling(i).sales.agg("mean").shift(-i*6)
    print("Correlation")
    a[["sales"]+a.columns[a.columns.str.startswith("SMA")]].tolist().corr()

```

Correlation

Out[]:

	sales	SMA20_sales_lag16	SMA20_sales_lag30	SMA20_sales_lag60	SMA30_sales_lag16
sales	1.00	0.90	0.89	0.87	
SMA20_sales_lag16	0.90	1.00	0.99	0.95	
SMA20_sales_lag30	0.89	0.99	1.00	0.96	
SMA20_sales_lag60	0.87	0.95	0.96	1.00	
SMA30_sales_lag16	0.90	1.00	0.99	0.96	
SMA30_sales_lag30	0.89	0.98	1.00	0.98	
SMA30_sales_lag60	0.87	0.95	0.96	1.00	
SMA45_sales_lag16	0.91	0.99	1.00	0.98	
SMA45_sales_lag30	0.89	0.98	0.99	0.99	
SMA45_sales_lag60	0.87	0.95	0.97	0.99	
SMA60_sales_lag16	0.91	0.99	0.99	0.99	
SMA60_sales_lag30	0.89	0.98	0.99	0.99	
SMA60_sales_lag60	0.87	0.95	0.97	0.99	
SMA90_sales_lag16	0.90	0.99	0.99	0.99	
SMA90_sales_lag30	0.89	0.98	0.99	0.99	
SMA90_sales_lag60	0.86	0.95	0.96	0.99	
SMA120_sales_lag16	0.90	0.98	0.99	0.99	
SMA120_sales_lag30	0.89	0.97	0.98	0.99	
SMA120_sales_lag60	0.86	0.95	0.96	0.98	
SMA365_sales_lag16	0.90	0.97	0.97	0.98	
SMA365_sales_lag30	0.89	0.97	0.97	0.97	
SMA365_sales_lag60	0.87	0.95	0.96	0.97	
SMA730_sales_lag16	0.91	0.98	0.98	0.98	
SMA730_sales_lag30	0.89	0.98	0.98	0.98	
SMA730_sales_lag60	0.86	0.95	0.96	0.98	

- Trong vòng lặp, ta tính toán một số đặc trưng trung bình trượt cho mỗi kết hợp của "store_nbr" và "family" bằng cách sử dụng các kích thước cửa sổ đã chỉ định.

- Các đặc trưng này được thêm vào DataFrame a. Trung bình trượt được tính cho các khoảng lệch thời gian (lag) khác nhau, bao gồm 16, 30 và 60 chu kỳ.
- Ví dụ, với kích thước cửa sổ là 20, ba cột mới sẽ được tạo ra:
 - "SMA20_sales_lag16": Trung bình động đơn giản trong 20 chu kỳ của doanh số, lệch 16 chu kỳ.
 - "SMA20_sales_lag30": Trung bình động đơn giản trong 20 chu kỳ của doanh số, lệch 30 chu kỳ.
 - "SMA20_sales_lag60": Trung bình động đơn giản trong 20 chu kỳ của doanh số, lệch 60 chu kỳ.
- Kết quả của mã này là ma trận tương quan giữa cột "sales" (doanh số) và các đặc trưng mới được tạo, đó là trung bình động đơn giản (Simple Moving Average - SMA) của doanh số với các kích thước cửa sổ và lệch thời gian khác nhau.
- Nhìn chung, giá trị tương quan cao (gần 1), có thể thấy mối quan hệ mạnh mẽ giữa cột "sales" và đặc trưng tương ứng.

=> Việc sử dụng trung bình động đơn giản có thể giúp làm mịn và làm giảm nhiễu trong dữ liệu, có thể làm cho xu hướng hoặc biểu đồ phổ biến của doanh số dễ theo dõi hơn.

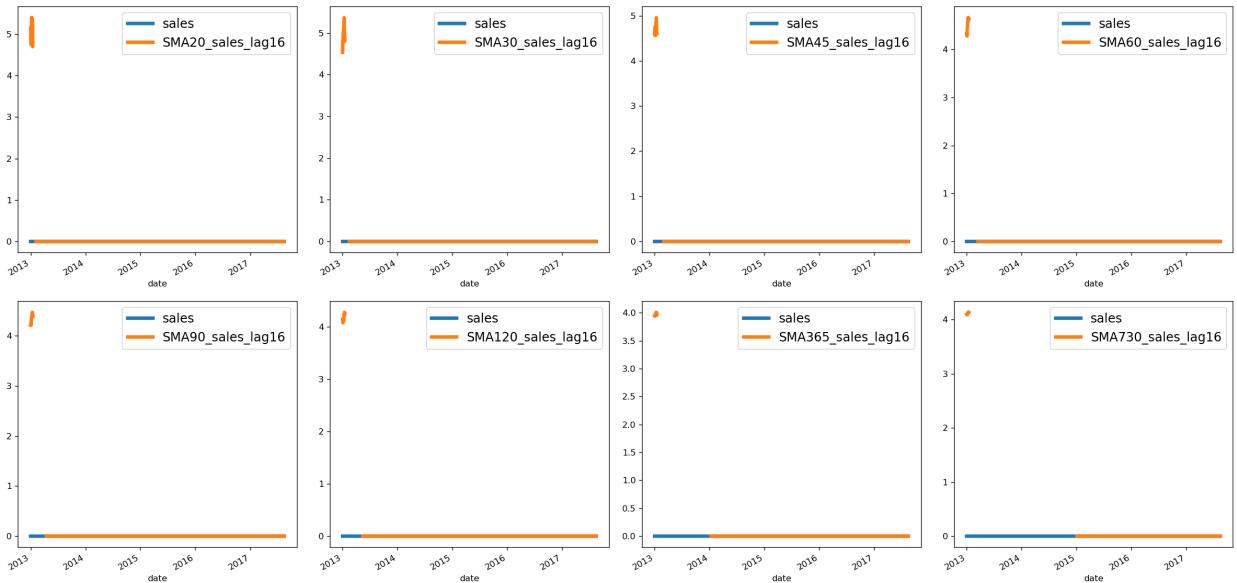
- Ta thấy mối quan hệ giữa SMA và sales mạnh, có thể sử dụng giá trị SMA để dự đoán hoặc hiểu hơn về xu hướng trong dữ liệu bán hàng.

```
In [ ]: b = a[(a.store_nbr == 1)].set_index("date")
for i in b.family.unique():
    fig, ax = plt.subplots(2,4,figsize=(20,10))
    b[b.family == i][["sales", "SMA20_sales_lag16"]].plot(legend = True, ax = ax[0,0])
    b[b.family == i][["sales", "SMA30_sales_lag16"]].plot(legend = True, ax = ax[0,1])
    b[b.family == i][["sales", "SMA45_sales_lag16"]].plot(legend = True, ax = ax[0,2])
    b[b.family == i][["sales", "SMA60_sales_lag16"]].plot(legend = True, ax = ax[0,3])
    b[b.family == i][["sales", "SMA90_sales_lag16"]].plot(legend = True, ax = ax[1,0])
    b[b.family == i][["sales", "SMA120_sales_lag16"]].plot(legend = True, ax = ax[1,1])
    b[b.family == i][["sales", "SMA365_sales_lag16"]].plot(legend = True, ax = ax[1,2])
    b[b.family == i][["sales", "SMA730_sales_lag16"]].plot(legend = True, ax = ax[1,3])
    plt.suptitle("STORE 1 - "+i, fontsize = 15)
    plt.tight_layout(pad = 1.5)
    for j in range(0,4):
        ax[0,j].legend(fontsize="x-large")
        ax[1,j].legend(fontsize="x-large")
plt.show()
```

STORE 1 - AUTOMOTIVE



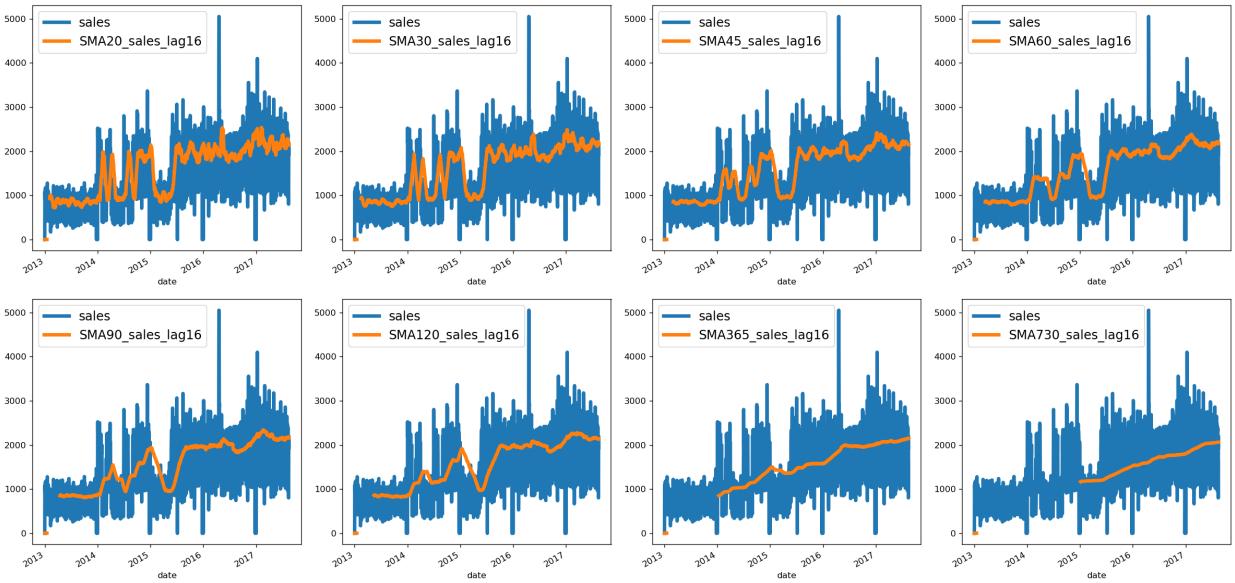
STORE 1 - BABY CARE



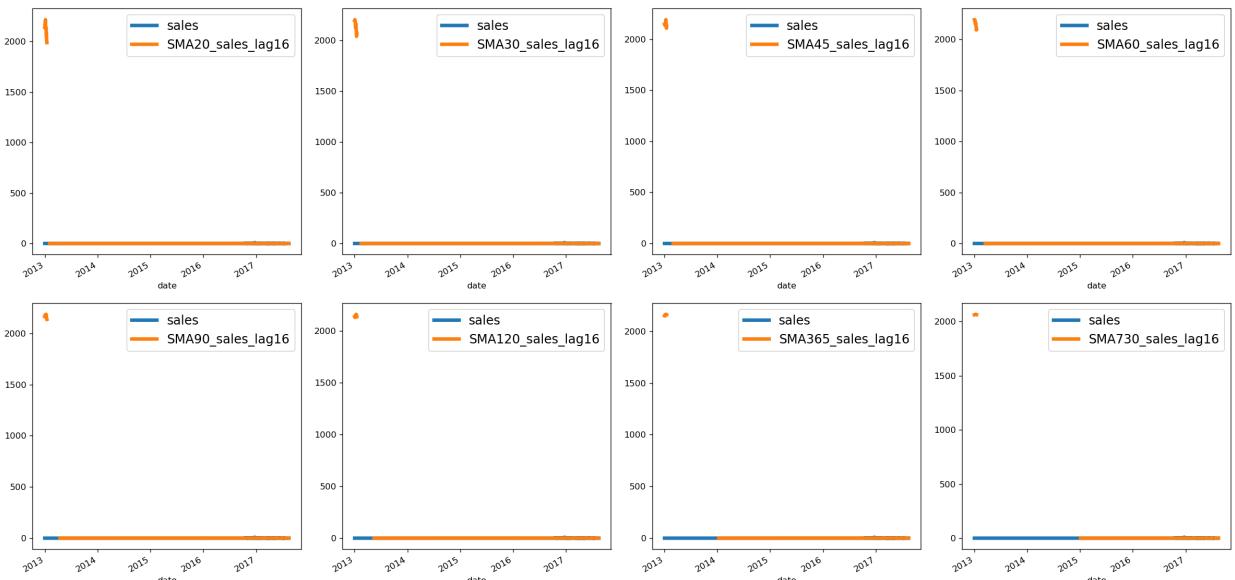
STORE 1 - BEAUTY



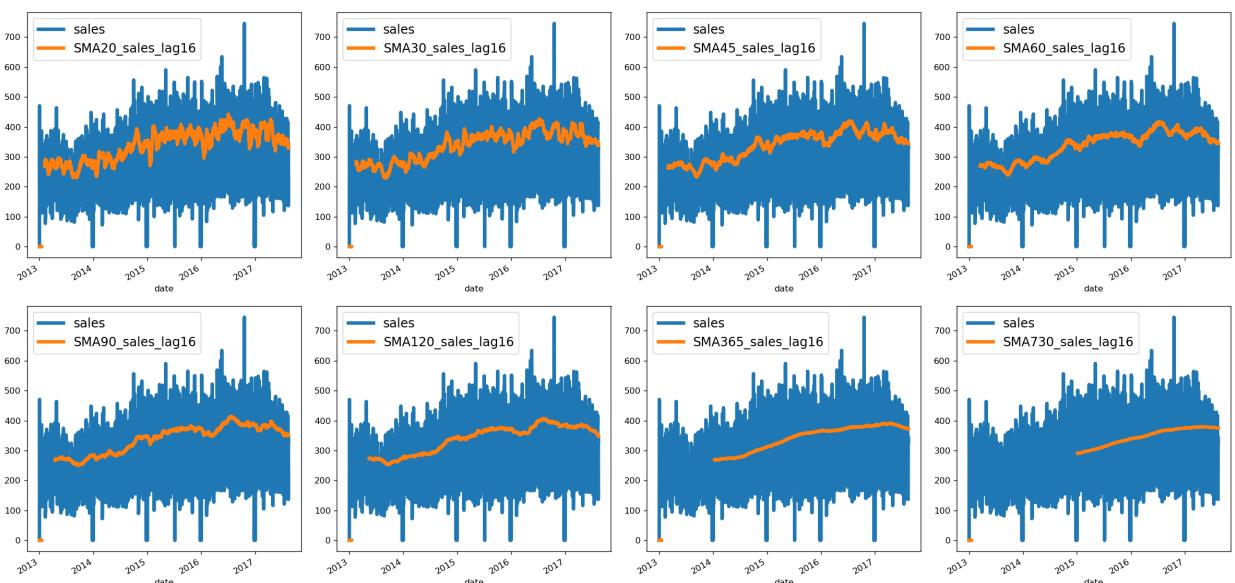
STORE 1 - BEVERAGES



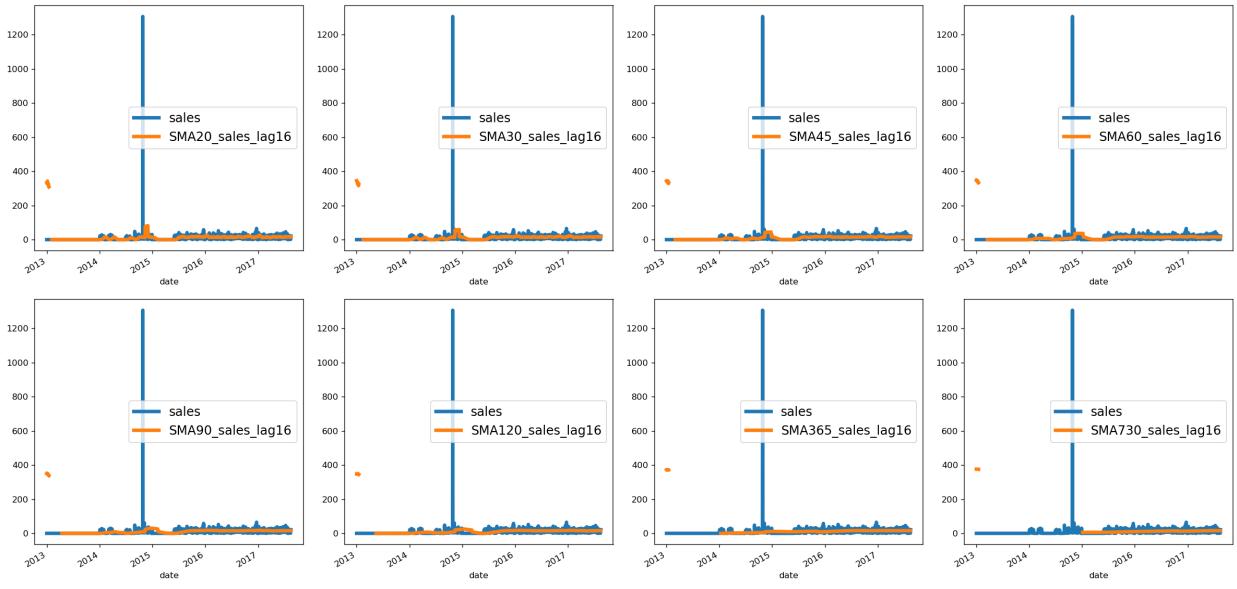
STORE 1 - BOOKS



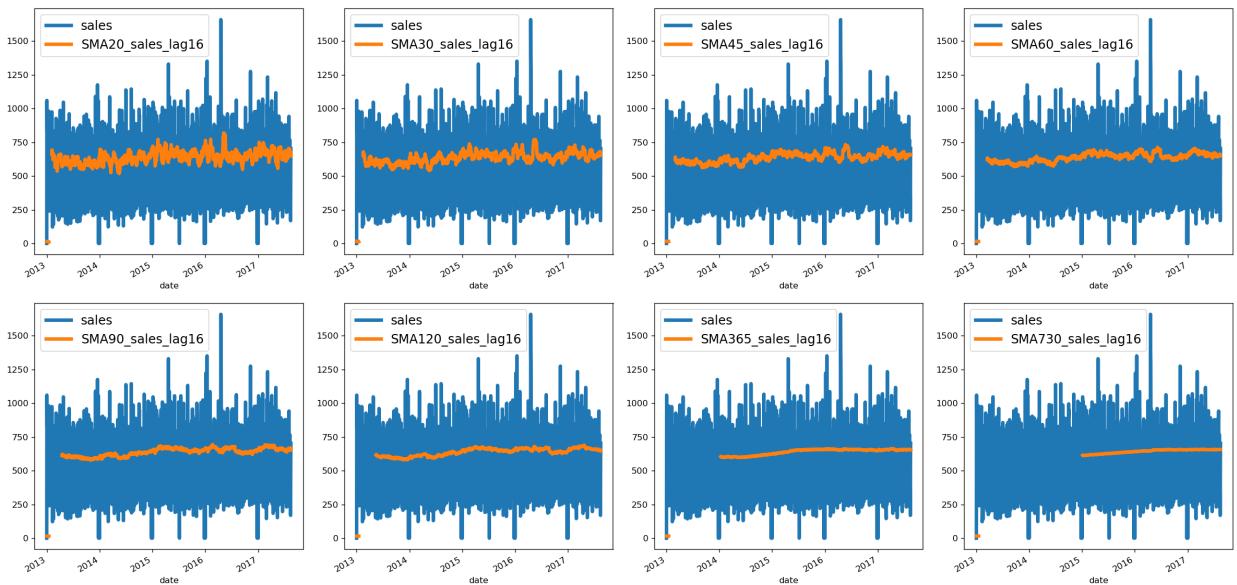
STORE 1 - BREAD/BAKERY



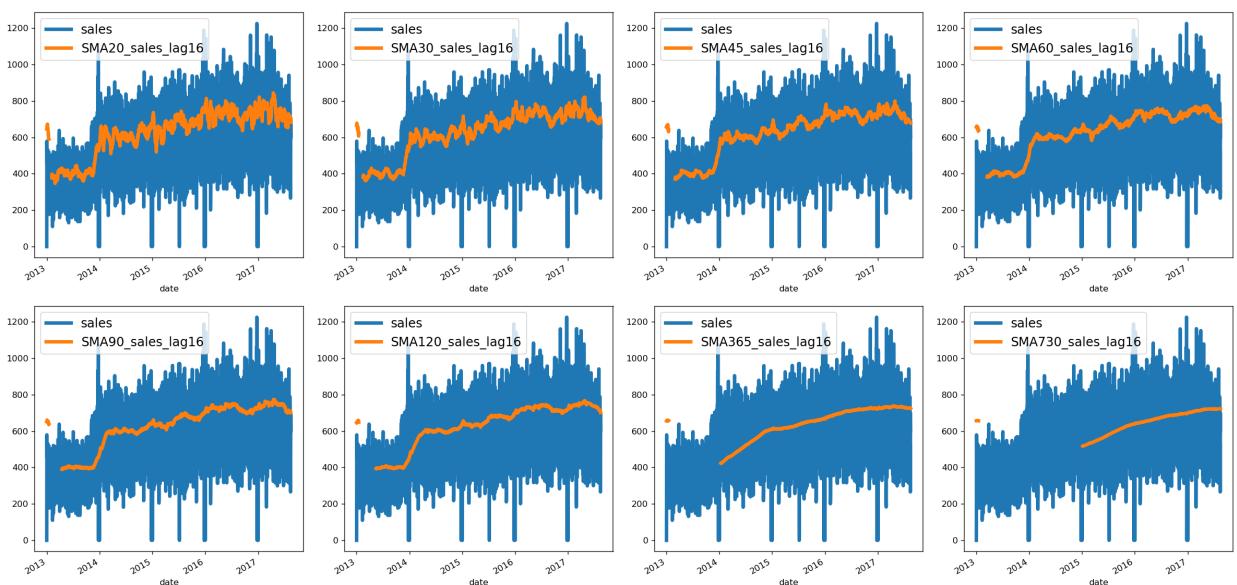
STORE 1 - CELEBRATION



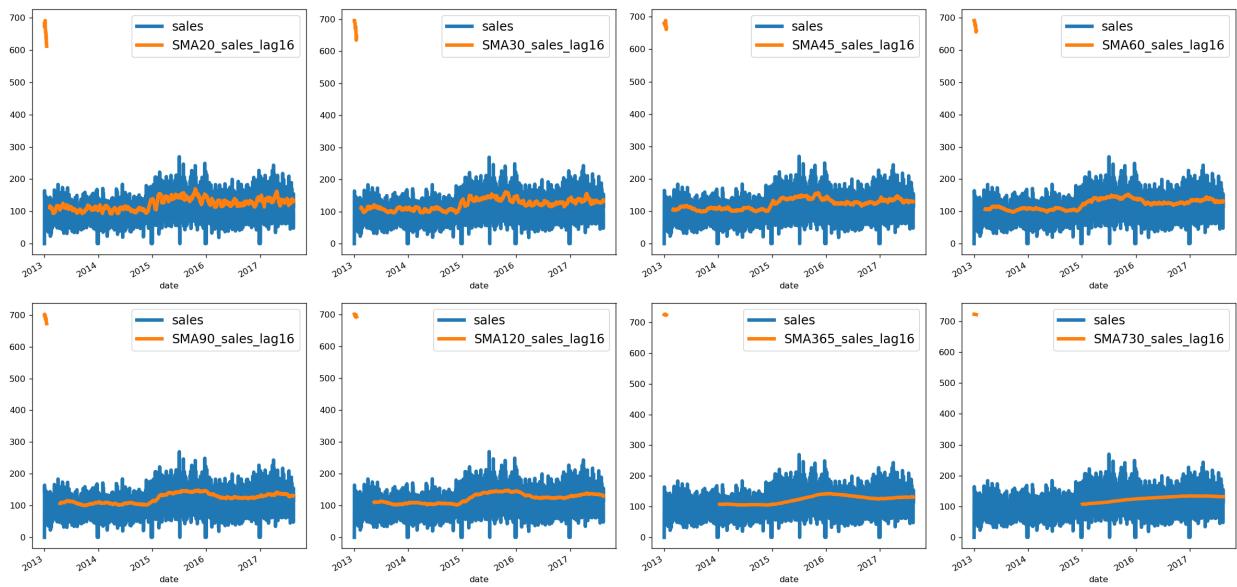
STORE 1 - CLEANING



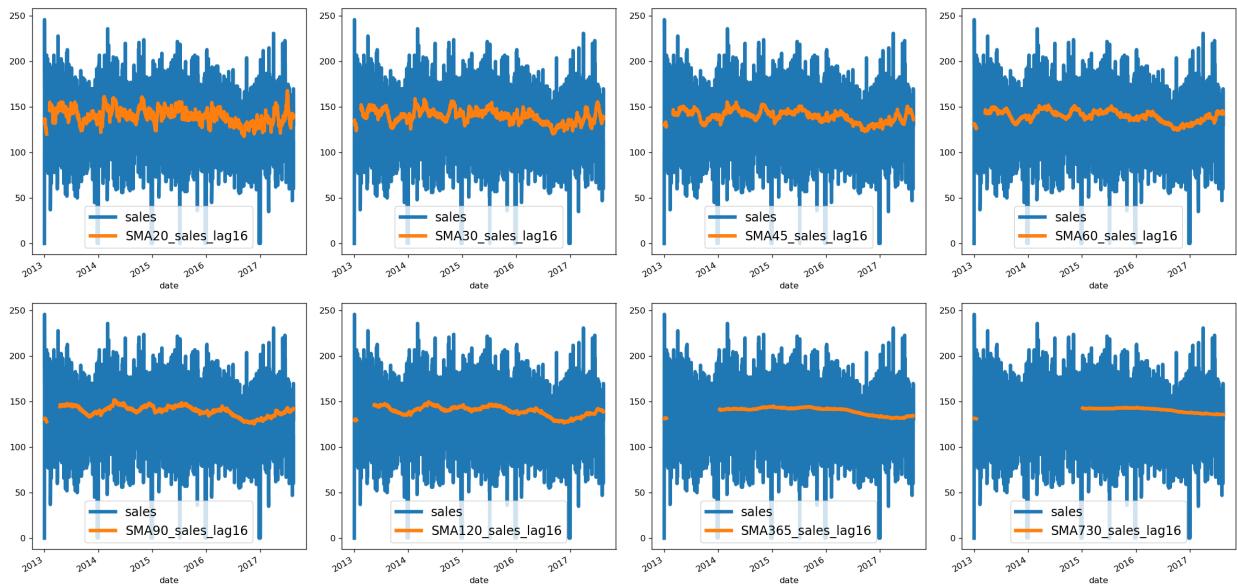
STORE 1 - DAIRY



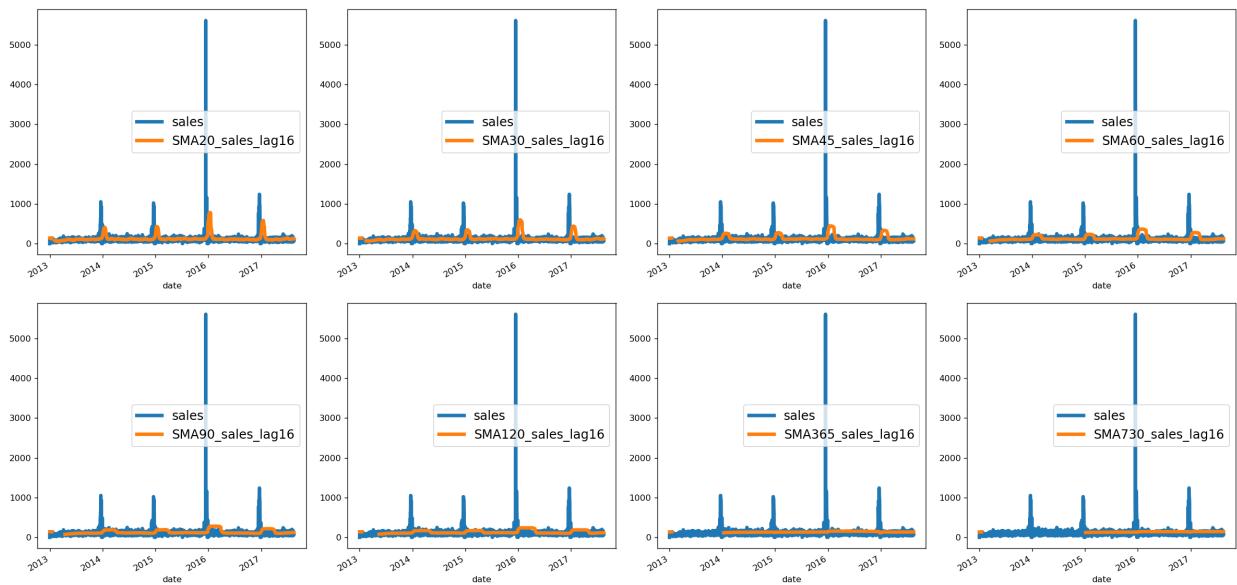
STORE 1 - DELI



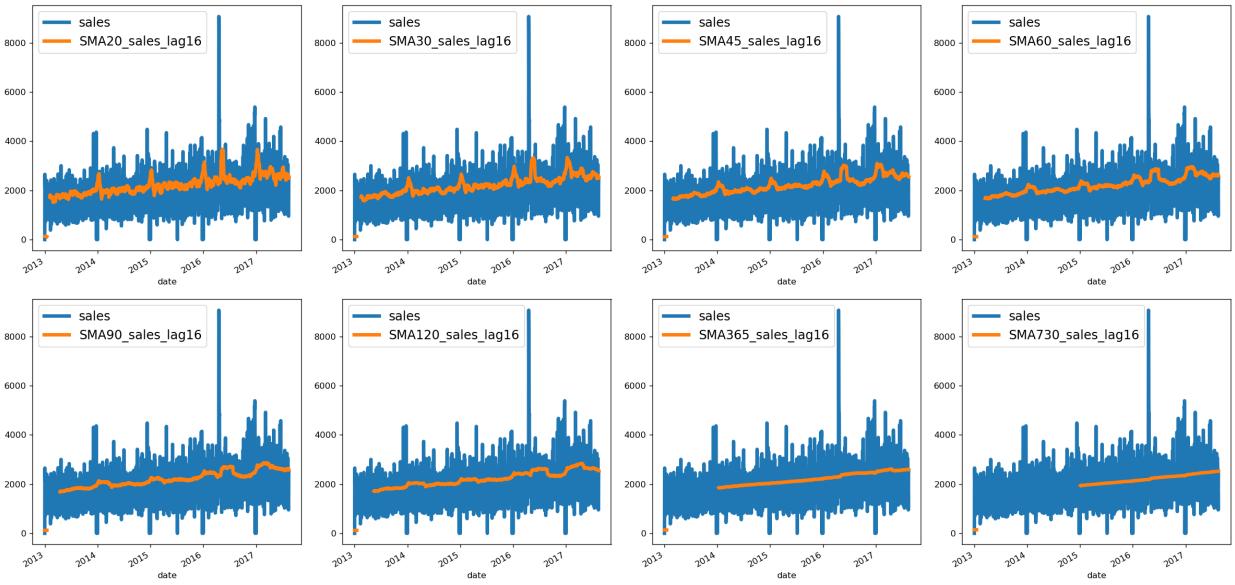
STORE 1 - EGGS



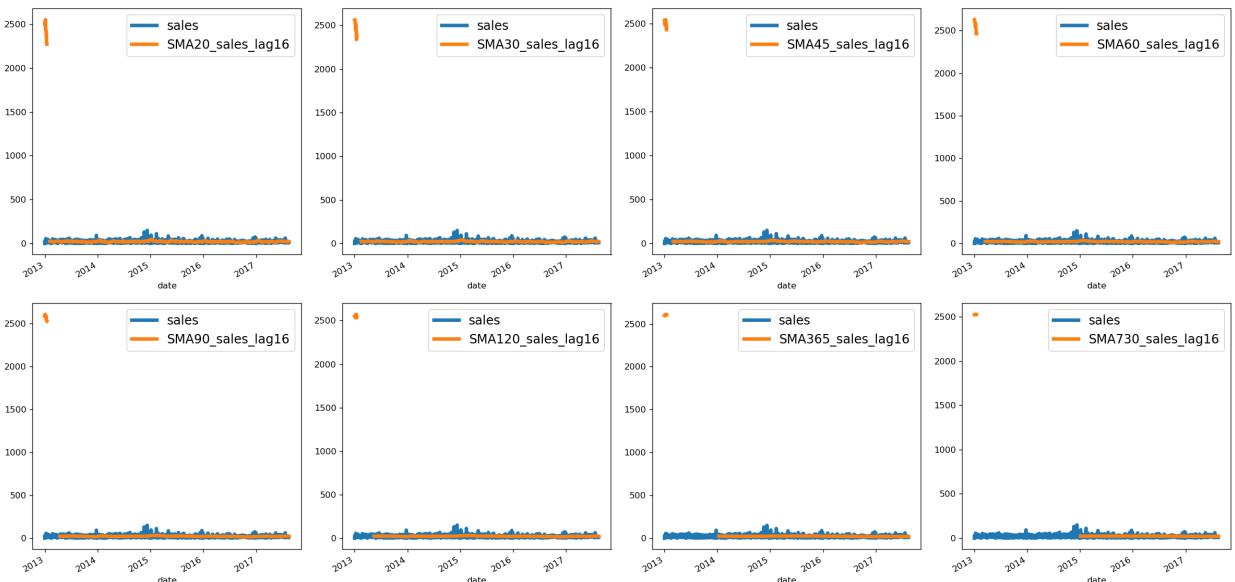
STORE 1 - FROZEN FOODS



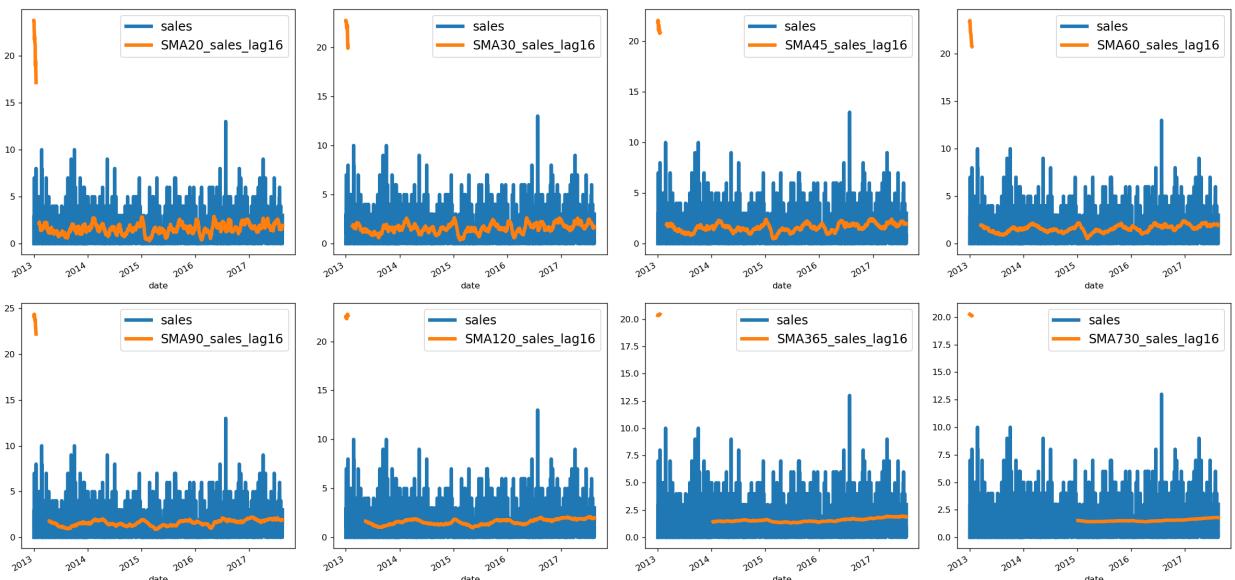
STORE 1 - GROCERY I



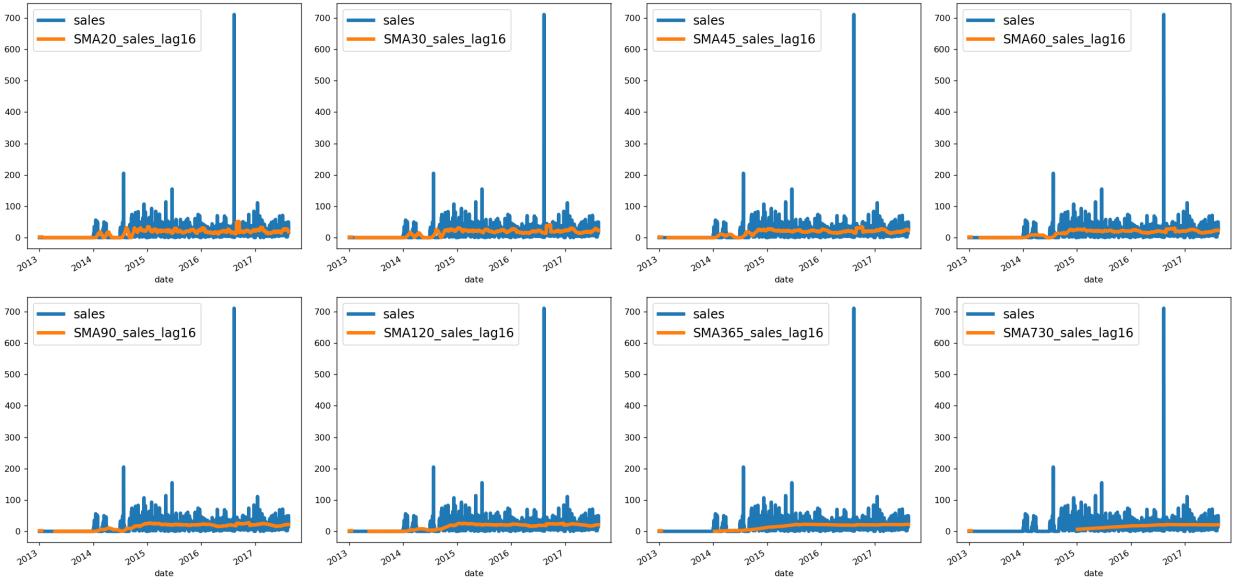
STORE 1 - GROCERY II



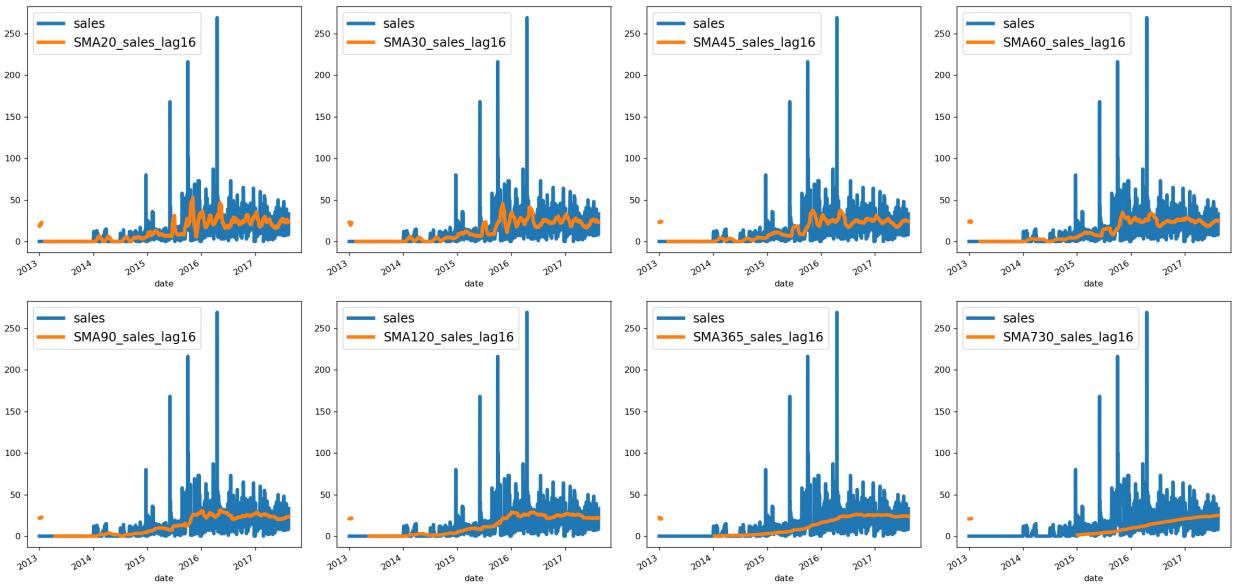
STORE 1 - HARDWARE



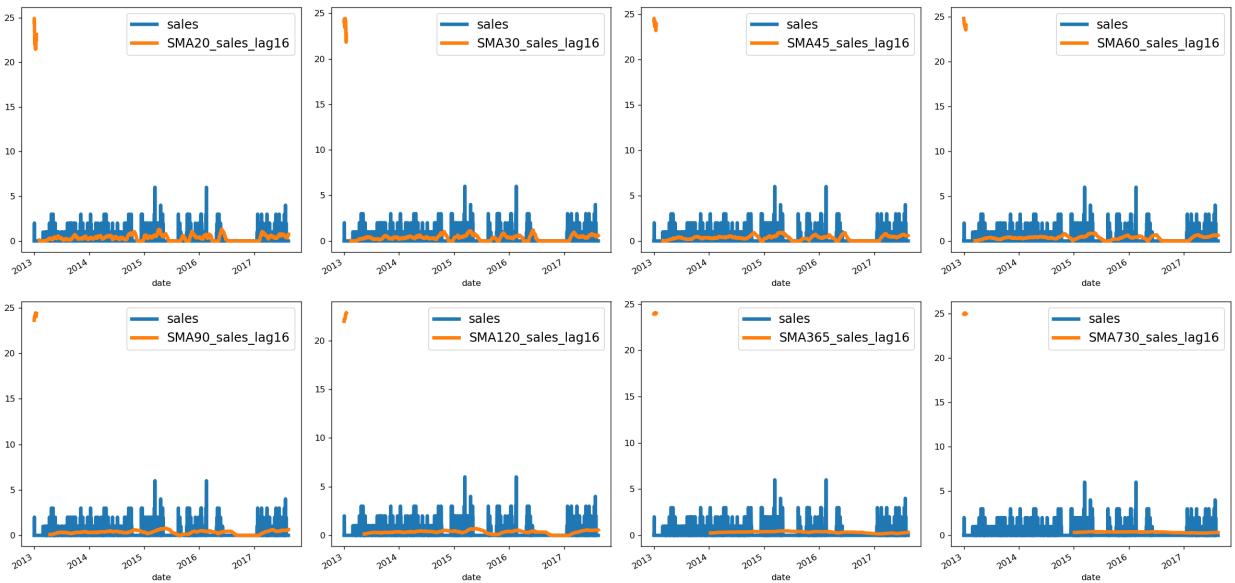
STORE 1 - HOME AND KITCHEN I



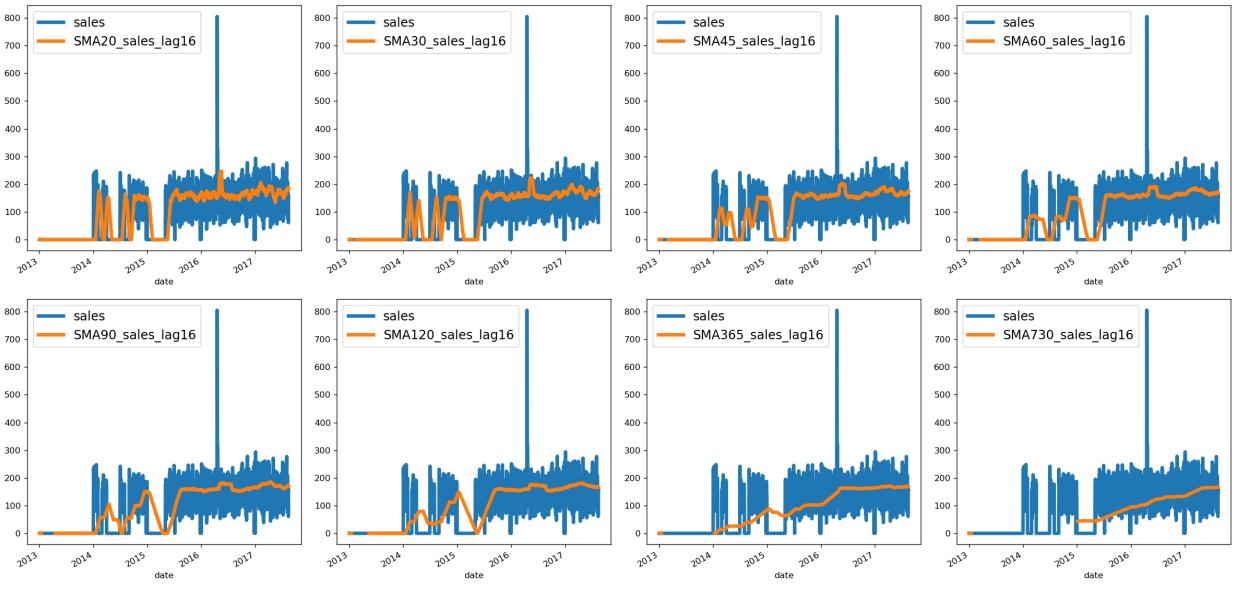
STORE 1 - HOME AND KITCHEN II



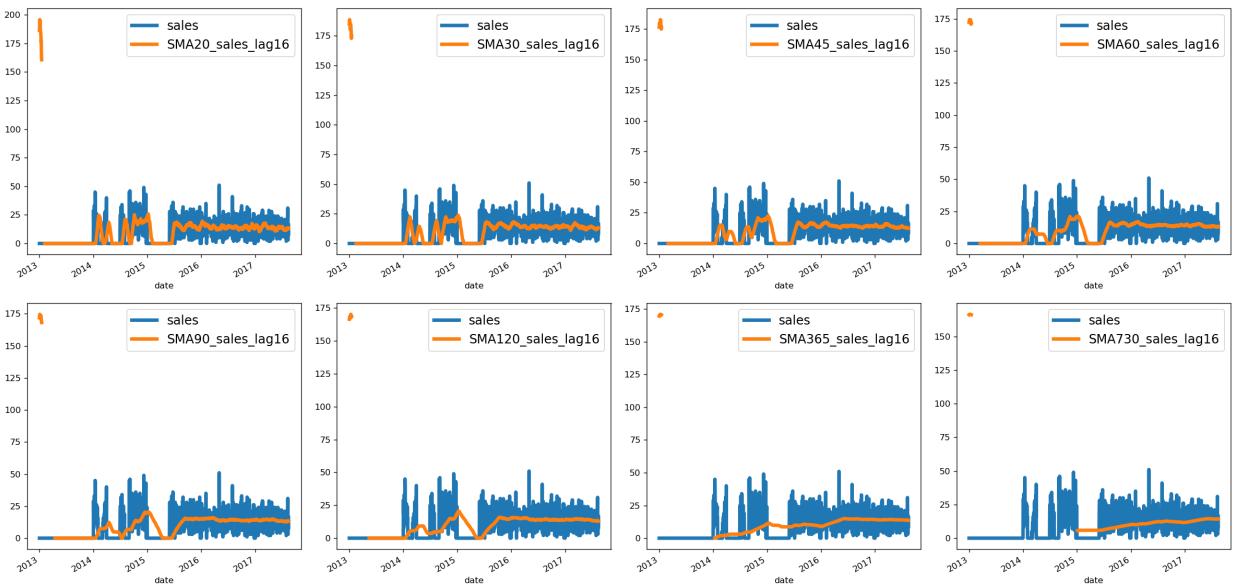
STORE 1 - HOME APPLIANCES



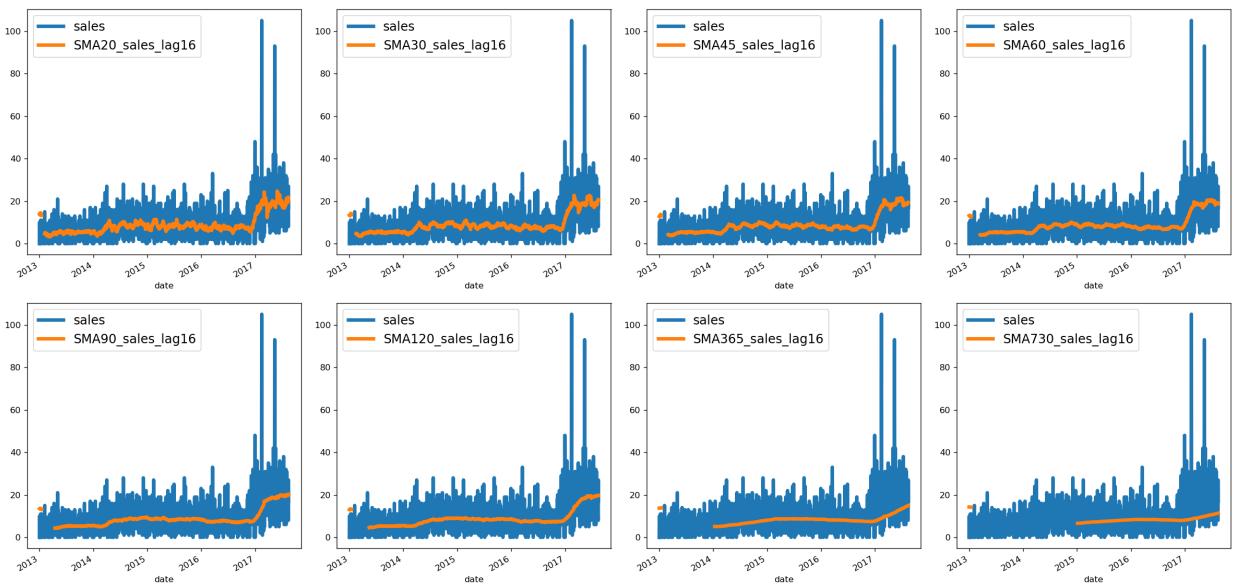
STORE 1 - HOME CARE



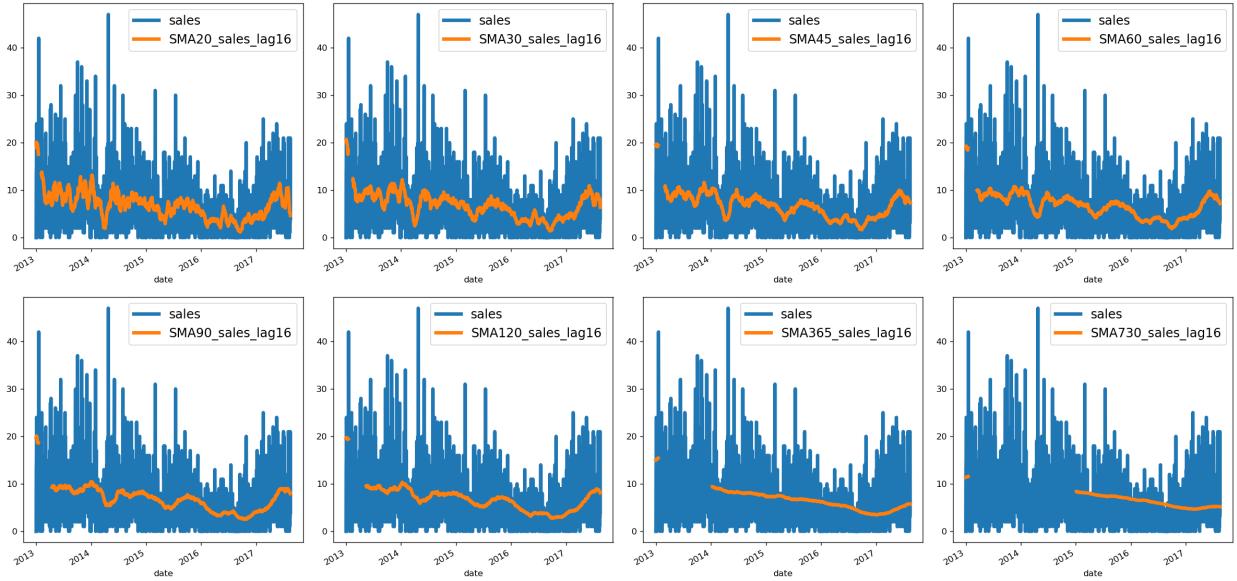
STORE 1 - LADIESWEAR



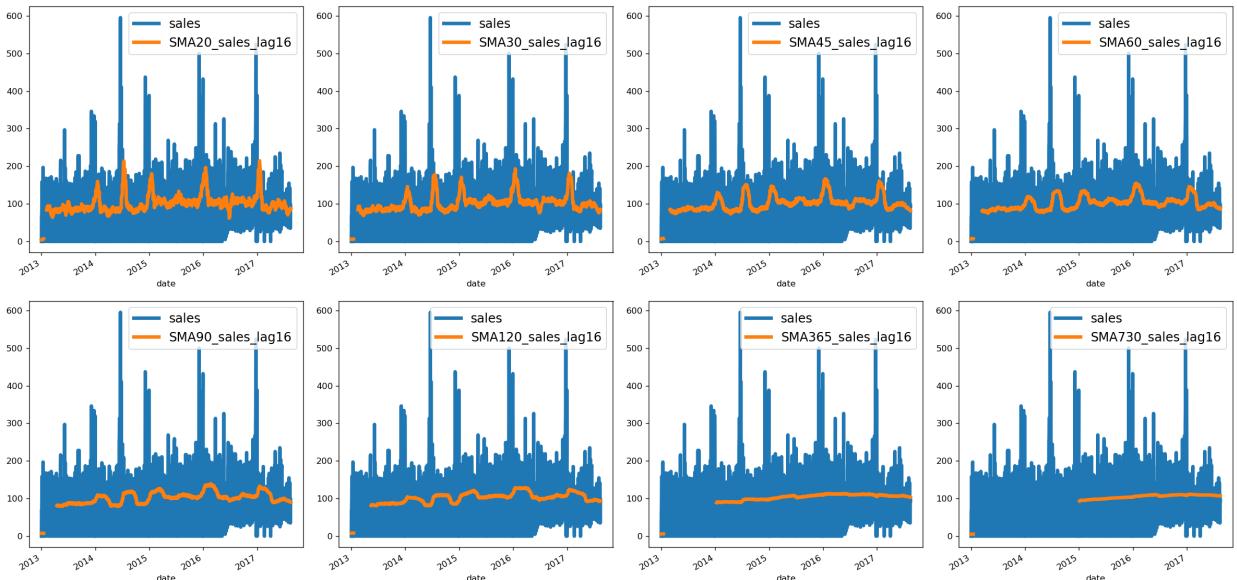
STORE 1 - LAWN AND GARDEN



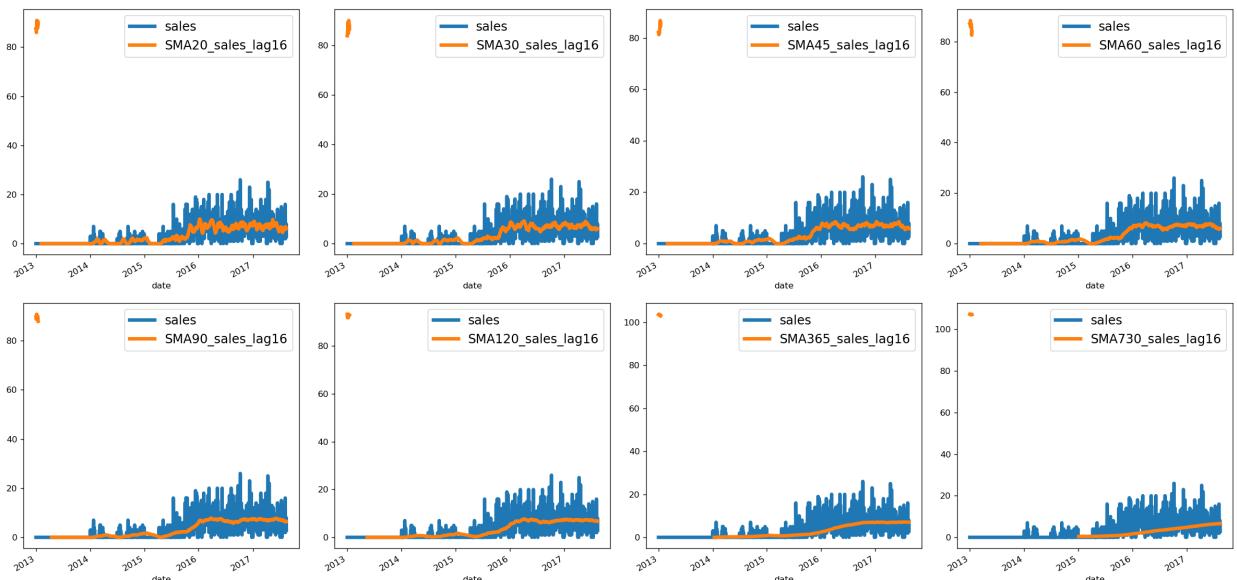
STORE 1 - LINGERIE



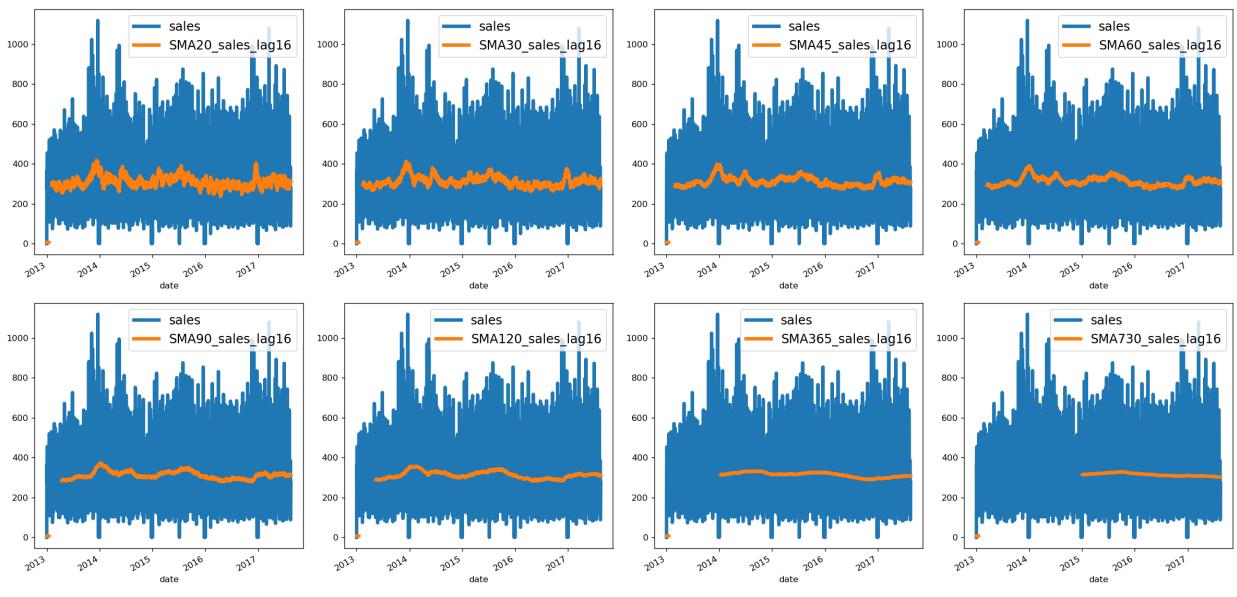
STORE 1 - LIQUOR,WINE,BEER



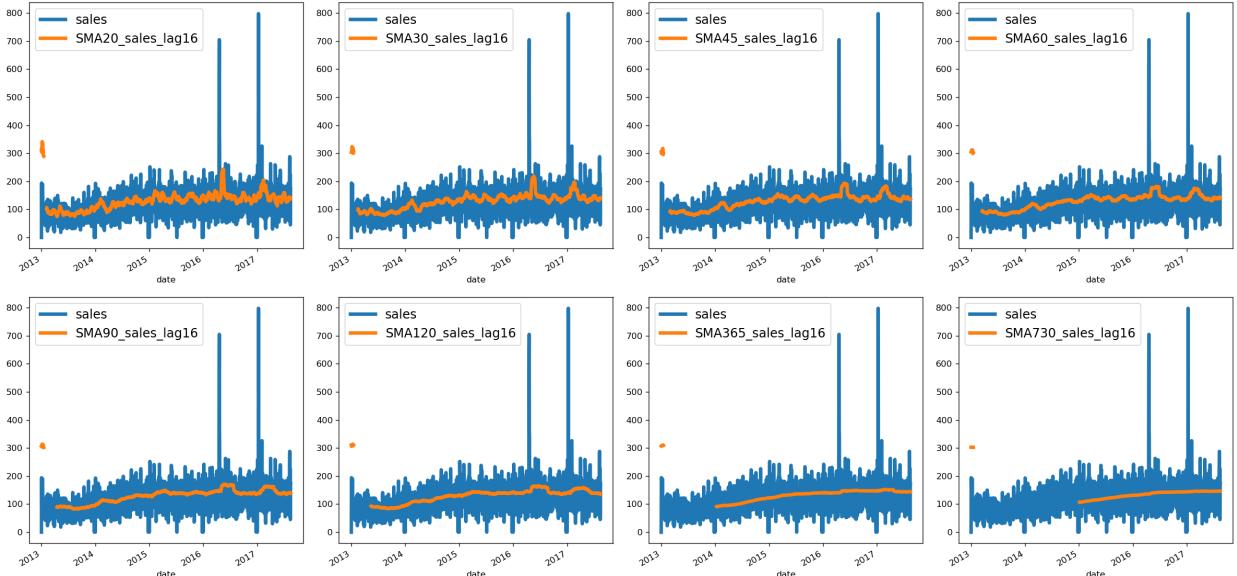
STORE 1 - MAGAZINES



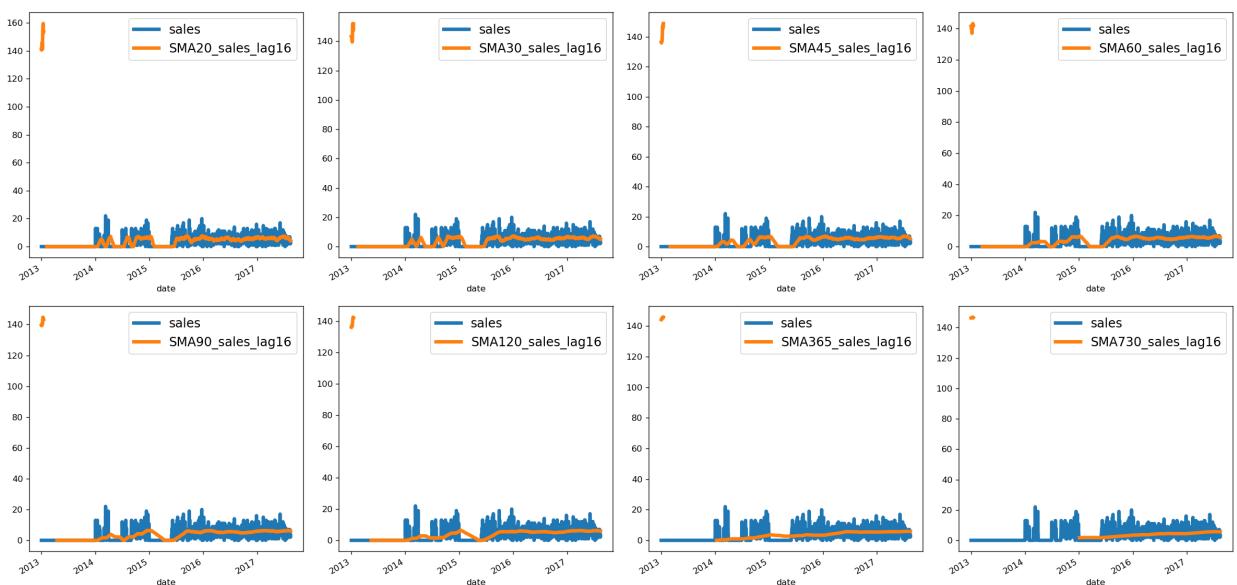
STORE 1 - MEATS



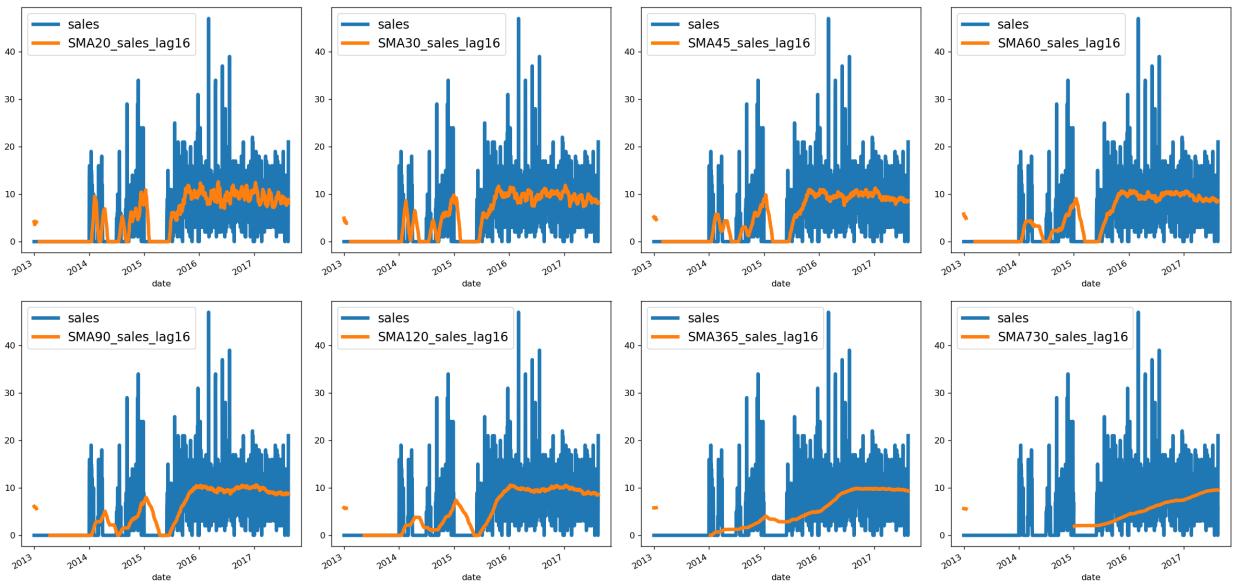
STORE 1 - PERSONAL CARE



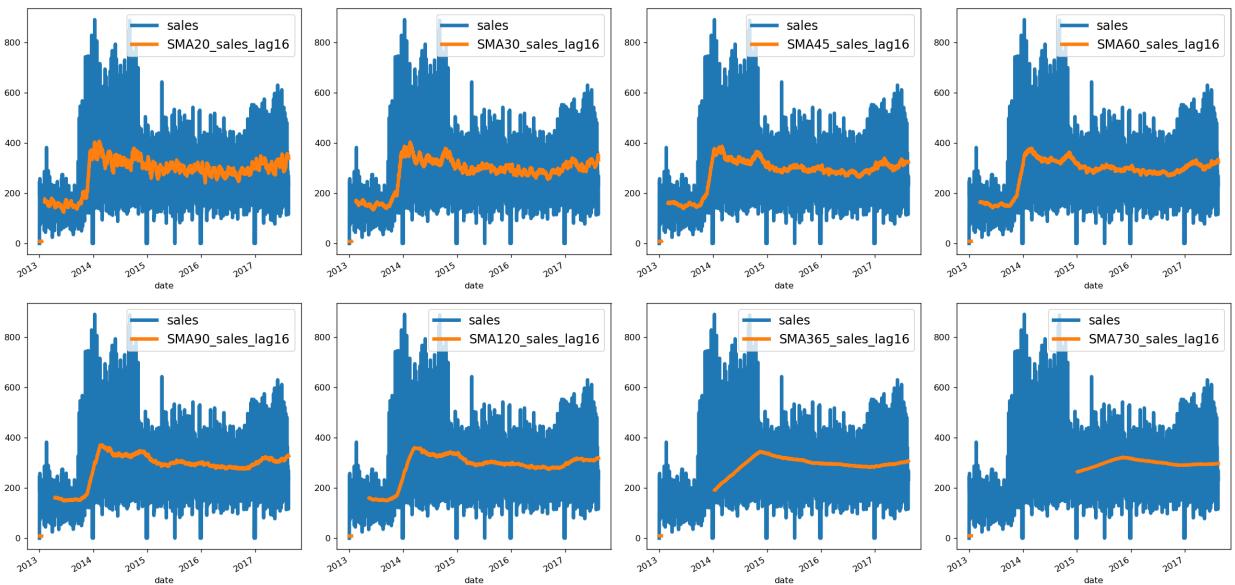
STORE 1 - PET SUPPLIES



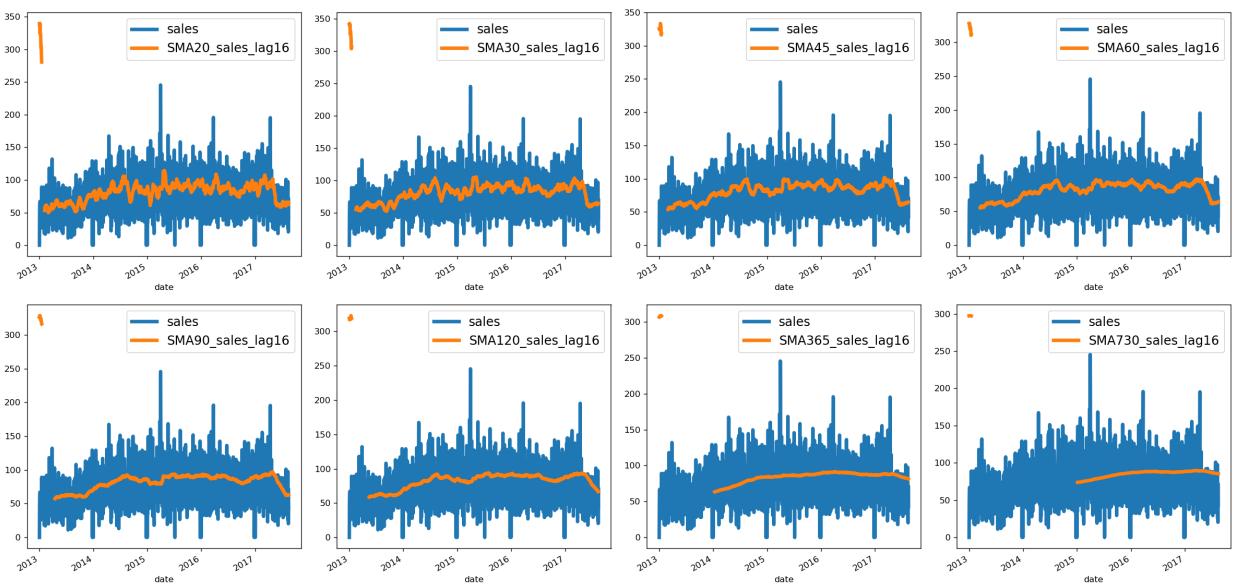
STORE 1 - PLAYERS AND ELECTRONICS



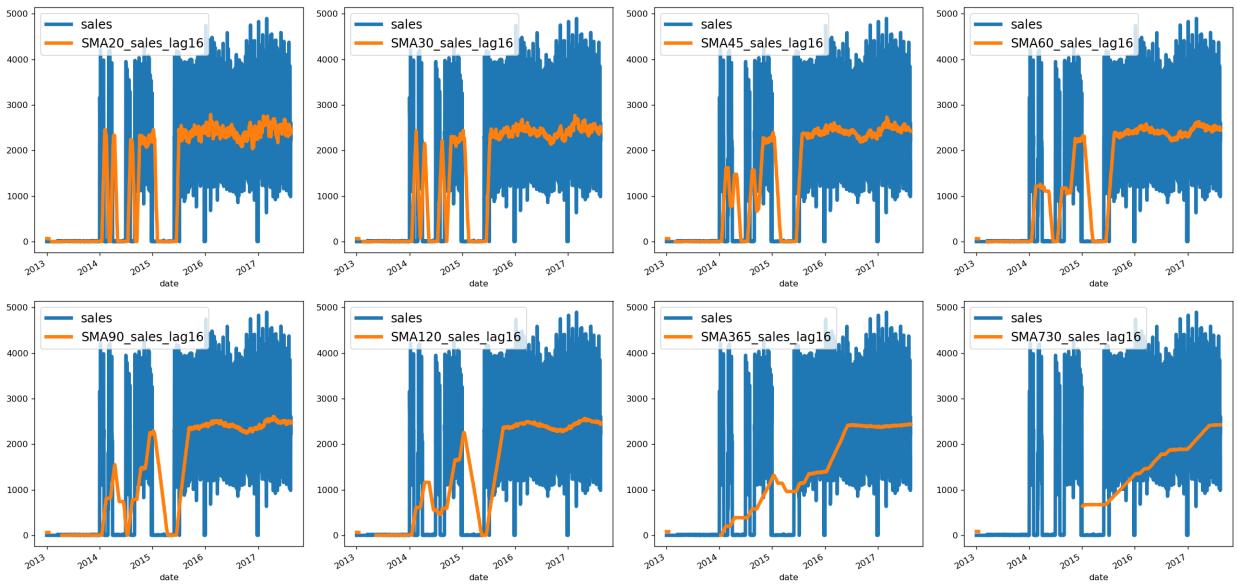
STORE 1 - POULTRY



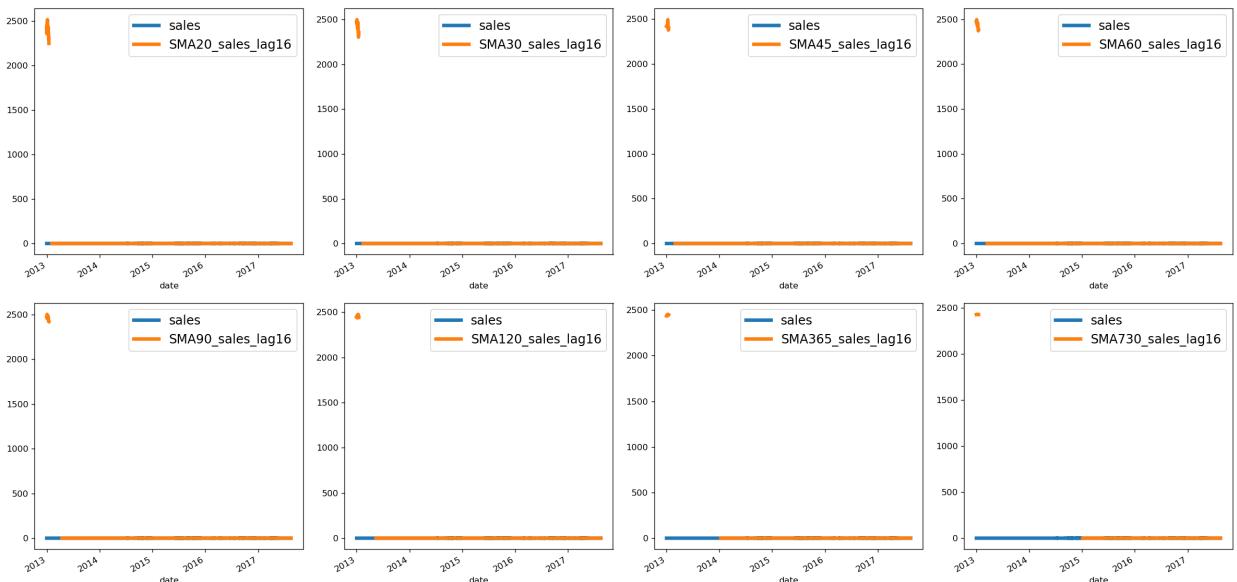
STORE 1 - PREPARED FOODS



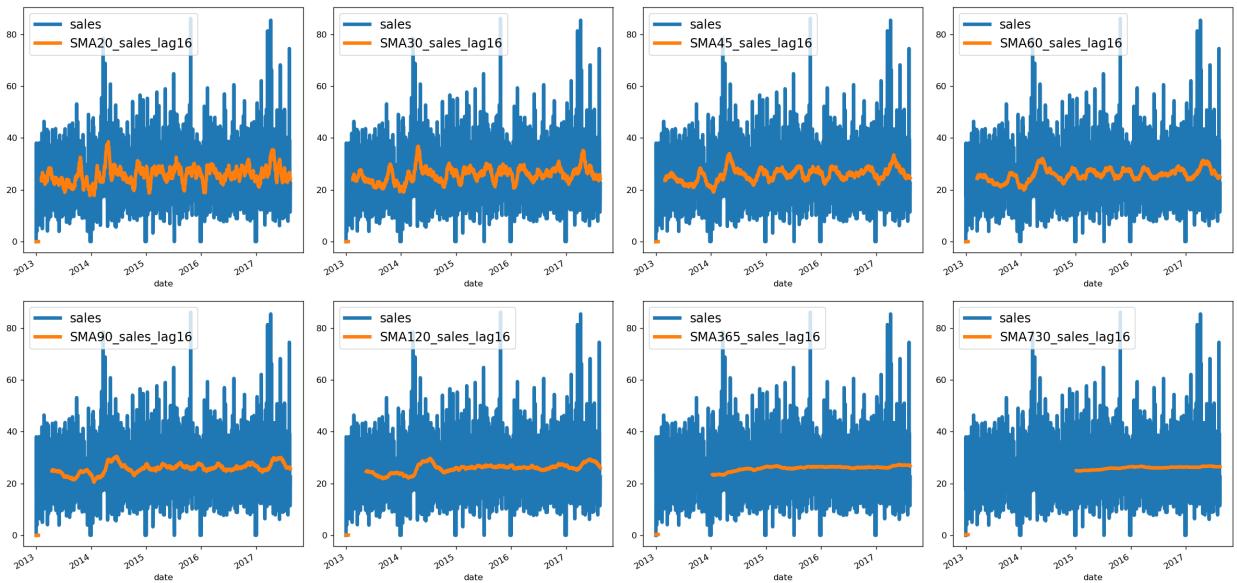
STORE 1 - PRODUCE



STORE 1 - SCHOOL AND OFFICE SUPPLIES



STORE 1 - SEAFOOD



=> Đoạn code này tạo ra một loạt các biểu đồ để so sánh doanh số thực tế của cửa hàng với các giá trị trung bình động đơn giản (SMA) tương ứng trong nhiều kịch thước cửa sổ và lệch thời gian khác nhau cho từng họ hàng (family) cụ thể.

4. Kiểm tra tính dừng và xác định bậc sai phân

- Kiểm tra tính dừng của chuỗi thời gian bằng cách sử dụng thử nghiệm Augmented Dickey-Fuller (ADF) từ thư viện mô hình thống kê

Giải thích về Augmented Dickey-Fuller (ADF) là một kiểm tra thống kê phổ biến được sử dụng trong phân tích dữ liệu chuỗi thời gian để xác định tính ổn định của chuỗi thời gian. Nó được sử dụng để kiểm tra xem một chuỗi thời gian có tính ổn định hay không, tức là nó có thể hoặc không thể dự đoán được trong tương lai

Một số điểm quan trọng về ADF:

- Mục tiêu: được sử dụng để kiểm tra giả thiết về tính ổn định của chuỗi thời gian. Giả thiết không ổn định cho thấy rằng chuỗi thời gian có tính ổn định và có thể dự đoán trong tương lai.
- Giả thuyết kiểm tra: Giả thiết gốc (null hypothesis) trong kiểm tra ADF là chuỗi thời gian không ổn định, tức là có tính hiệu suất (stationarity). Khi giả thiết này không bị bác bỏ, chuỗi thời gian được coi là không ổn định. Ngược lại, khi giả thiết bị bác bỏ, chuỗi thời gian được coi là có tính ổn định.
- Kết quả kiểm tra: kết quả của kiểm tra ADF thường gồm một số thống kê kiểm tra và trị p. Nếu giá trị p nhỏ hơn một ngưỡng xác định trước (thường là 0.05), thì ta có đủ bằng chứng để bác bỏ giả thuyết gốc và kết luận rằng chuỗi thời gian là ổn định. Ngược lại, nếu giá trị p lớn hơn ngưỡng, thì giả thiết không ổn định được chấp nhận.
- Đánh giá kết quả: khi kết quả kiểm tra ADF cho thấy chuỗi thời gian có tính ổn định, nó có thể được sử dụng để thực hiện các phân tích sự đoán và mô hình hóa chuỗi thời gian. Nếu kết quả cho thấy chuỗi thời gian không ổn định, có thể cần thực hiện các biện pháp biến đổi để biến nó thành ổn định trước khi áp dụng vào mô hình dự đoán.

=> Kiểm tra ADF là một phần quan trọng của quá trình phân tích dữ liệu chuỗi thời gian và giúp đảm bảo tính ổn định của chuỗi trước khi thực hiện dự đoán và mô hình hóa

```
In [ ]: train2 = train1.groupby('date')[['sales', 'onpromotion']].sum().reset_index()  
train2
```

Out[]:

	date	sales	onpromotion
0	2013-01-01	2511.62	0.00
1	2013-01-02	496092.42	0.00
2	2013-01-03	361461.23	0.00
3	2013-01-04	354459.68	0.00
4	2013-01-05	477350.12	0.00
...
1683	2017-08-11	826373.72	14179.00
1684	2017-08-12	792630.54	8312.00
1685	2017-08-13	865639.68	9283.00
1686	2017-08-14	760922.41	8043.00
1687	2017-08-15	762661.94	10605.00

1688 rows × 3 columns

In []:

```
ts = train2['sales']

from statsmodels.tsa.stattools import adfuller

result = adfuller(ts)

print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values: ')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
```

```
ADF Statistic: -2.723700
p-value: 0.070026
Critical Values:
    1%: -3.434
    5%: -2.863
    10%: -2.568
```

- Thống kê ADF là thống kê kiểm tra được sử dụng trong phân tích chuỗi thời gian để kiểm tra sự hiện diện của nghiệm đơn vị trong chuỗi thời gian. Một nghiệm đơn vị ngụ ý rằng chuỗi không cố định và các tính chất của nó thay đổi theo thời gian, gây khó khăn cho việc đưa ra các dự đoán đáng tin cậy.
- Trong trường hợp của bạn, số liệu thống kê ADF là -2.723700. Giá trị này cho thấy sức mạnh của bằng chứng chống lại giả thuyết không về nghiệm đơn vị hiện diện trong chuỗi thời gian. Thống kê ADF càng thấp thì bằng chứng chống lại giả thuyết không càng mạnh.
- Giá trị p của 0.070026 là xác suất quan sát được một thống kê ADF ở mức cực đại hoặc cực đoan hơn giá trị quan sát được theo giả thuyết khống của nghiệm đơn vị. Nói cách khác, nó cho bạn biết khả năng thống kê ADF là kết quả của cơ hội ngẫu nhiên chứ không phải là dấu hiệu thực sự của chuỗi thời gian không cố định.

- Trong trường hợp này, giá trị p nhỏ hơn 0.05, có nghĩa là chúng ta có thể bác bỏ giả thuyết không về nghiệm đơn vị với độ tin cậy 95% và kết luận rằng chuỗi thời gian là dừng.
- Các giá trị tới hạn của thử nghiệm ADF được cung cấp để so sánh với thống kê ADF quan sát được. Chúng chỉ ra các giá trị ngưỡng mà thống kê ADF cần vượt qua để bác bỏ giả thuyết không về nghiệm đơn vị ở các mức ý nghĩa khác nhau. Trong trường hợp của bạn, các giá trị tới hạn ở mức 1%, 5% và 10% lần lượt là -3.434, -2.863 và -2.568. Vì thống kê ADF được quan sát lớn hơn giá trị tới hạn ở mức 1% và 5%, kết quả là chúng ta chắc chắn rằng chuỗi thời gian của chúng ta không dừng.
- Vì giá trị p (0.070026) lớn hơn mức ý nghĩa (0.05), hãy chênh lệch chuỗi và xem biểu đồ tự tương quan trông như thế nào.
- vẽ biểu đồ hàm tự tương quan (ACF - Autocorrelation Function) và hàm tự tương quan riêng phần (PACF - Partial Autocorrelation Function) của một chuỗi dữ liệu thời gian.
 - Nhìn chung, biểu đồ ACF giúp xác định mức độ tương quan tuyến tính giữa giá trị tại một thời điểm và giá trị tại các thời điểm trước đó, trong khi biểu đồ PACF giúp xác định mức độ tương quan chỉ của thời điểm hiện tại và các thời điểm trước đó mà không có sự ảnh hưởng từ các thời điểm ở giữa.

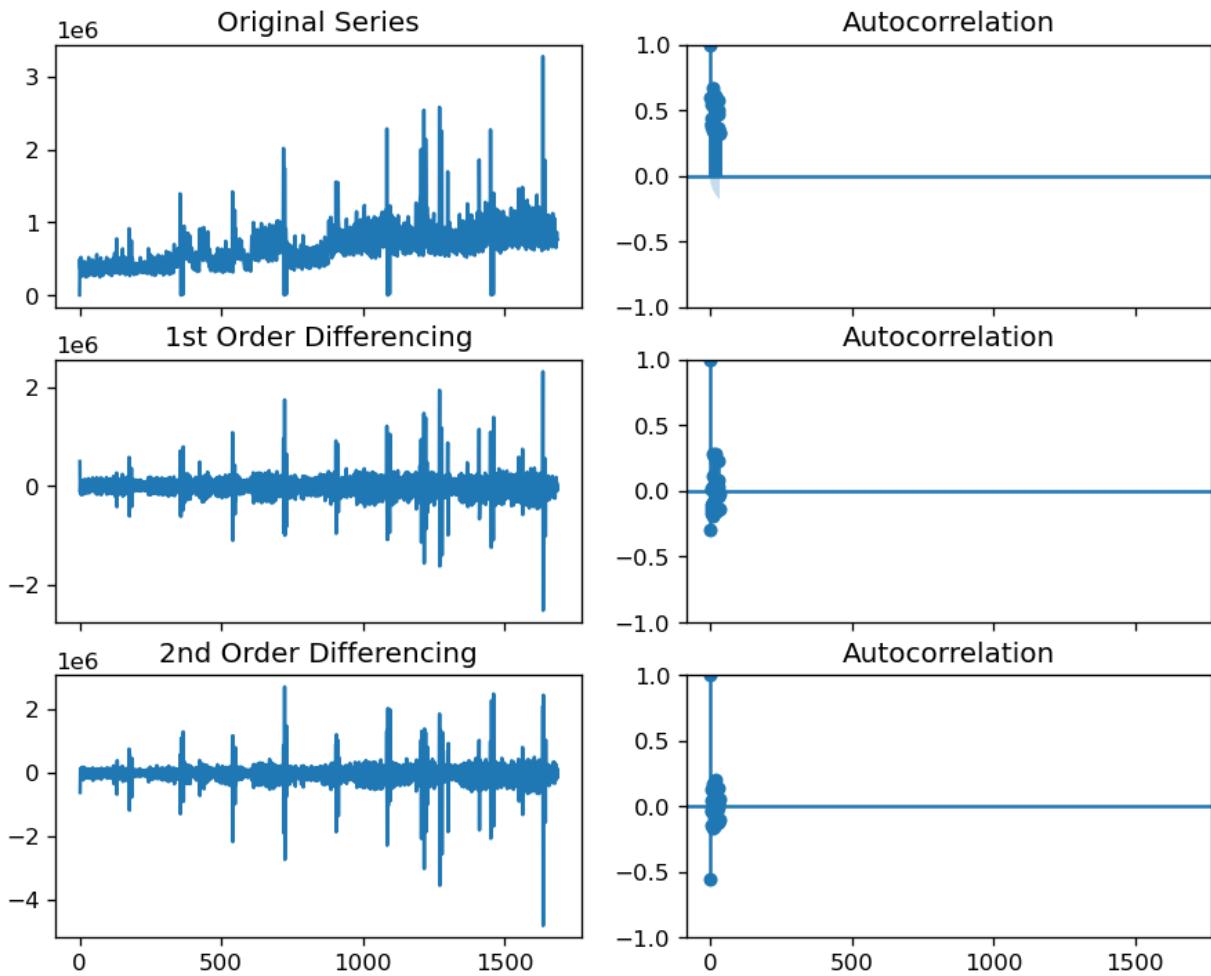
```
In [ ]: from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
plt.rcParams.update({'figure.figsize':(9,7), 'figure.dpi':120})

# Original Series
fig, axes = plt.subplots(3, 2, sharex=True)
axes[0, 0].plot(train2.sales); axes[0, 0].set_title('Original Series')
plot_acf(train2.sales, ax=axes[0, 1])

# 1st Differencing
axes[1, 0].plot(train2.sales.diff()); axes[1, 0].set_title('1st Order Differencing')
plot_acf(train2.sales.diff().dropna(), ax=axes[1, 1])

# 2nd Differencing
axes[2, 0].plot(train2.sales.diff().diff()); axes[2, 0].set_title('2nd Order Differencing')
plot_acf(train2.sales.diff().diff().dropna(), ax=axes[2, 1])

plt.show()
```



- Đoạn code nhằm giúp đánh giá mức độ tương quan qua các chênh lệch theo từng cấp độ (cấp 1,2) và biểu đồ của chuỗi thời gian sau khi áp dụng các chênh lệch
- Nhìn chung, sau khi biến đổi cấp 1, có sự khác biệt so với dữ liệu gốc, nhưng chênh lệch cấp 2 so với cấp 1 thay đổi lại không đáng kể => Sự thay đổi giữa dữ liệu gốc và sau khi chênh lệch cấp 1 có thể làm giảm sự tương quan tự (autocorrelation) trong chuỗi thời gian, và biểu đồ hàm tự tương quan sau chênh lệch cấp 1 thường giảm đáng kể từ mức độ cao ở lag = 1.

```
In [ ]: from statsmodels.tsa.seasonal import seasonal_decompose

# decompose the time series
decomposition = seasonal_decompose(train2['sales'], model='additive', period=12)

# create a seasonal plot
fig, axes = plt.subplots(nrows=4, ncols=1, figsize=(10,20))
fig.subplots_adjust(hspace=0.5) # added space between subplots

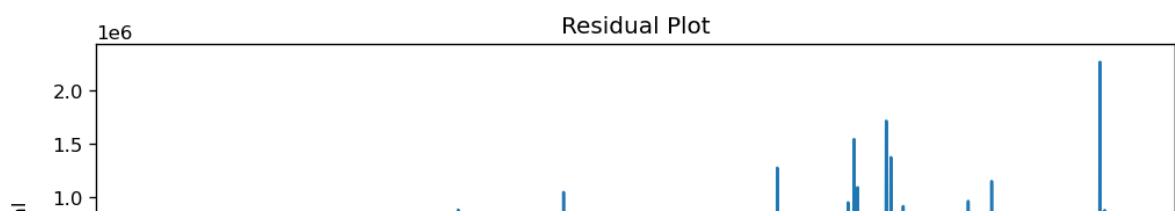
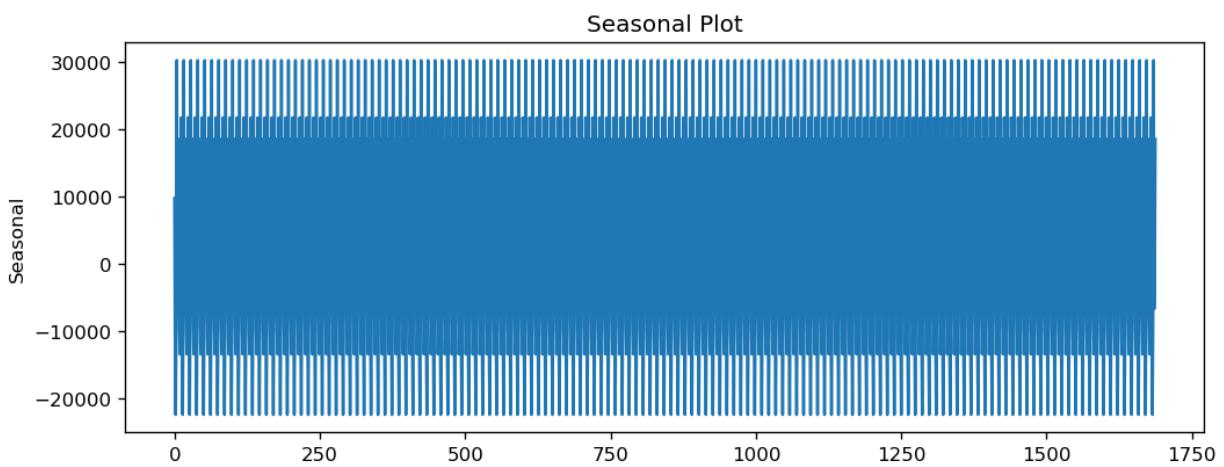
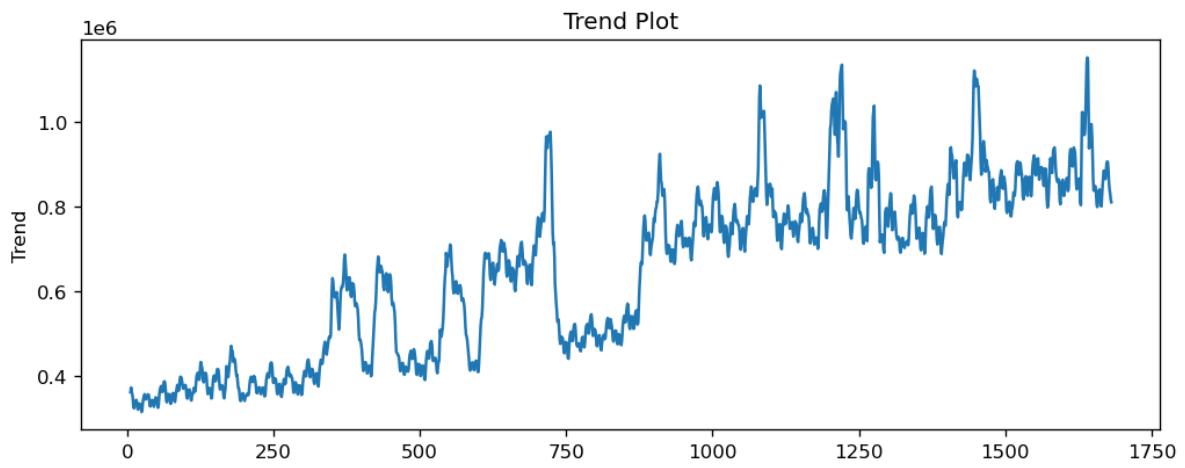
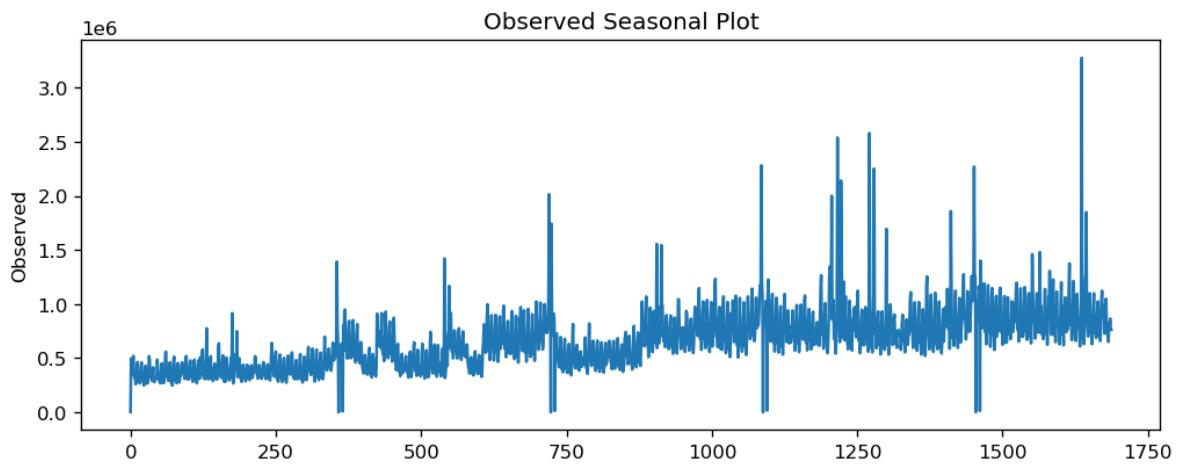
decomposition.observed.plot(ax=axes[0], legend=False)
axes[0].set_ylabel('Observed')
axes[0].set_title('Observed Seasonal Plot')

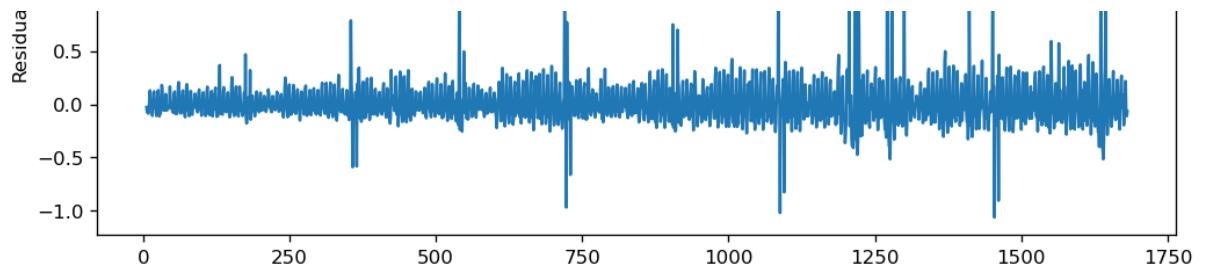
decomposition.trend.plot(ax=axes[1], legend=False)
axes[1].set_ylabel('Trend')
axes[1].set_title('Trend Plot')
```

```
decomposition.seasonal.plot(ax=axes[2], legend=False)
axes[2].set_ylabel('Seasonal')
axes[2].set_title('Seasonal Plot')

decomposition.resid.plot(ax=axes[3], legend=False)
axes[3].set_ylabel('Residual')
axes[3].set_title('Residual Plot')

Out[ ]: Text(0.5, 1.0, 'Residual Plot')
```





- Như chúng ta có thể thấy, chúng ta có một số mô hình theo mùa trong chuỗi và mối tương quan cao giữa giá trị hiện tại và quá khứ và chúng ta không có nhiễu trắng trong chuỗi vì chúng ta có mối tương quan này. Nhưng trong hiện tượng tự tương quan, nếu một giá trị tương quan với hiện tại thì giá trị tiếp theo cũng là hiện tại. Để làm điều này, chúng ta cần xem xét Tự tương quan một phần, bởi vì bằng cách này, chúng ta loại bỏ ảnh hưởng của các giá trị trong quá khứ đối với giá trị tiếp theo.

5. MÔ HÌNH ARIMA

- Mô hình ARIMA (AutoRegressive Integrated Moving Average) là một mô hình thống kê phổ biến được sử dụng để dự báo chuỗi thời gian. Mô hình ARIMA kết hợp cả các yếu tố tự hồi quy (AR - AutoRegressive) và yếu tố trung bình động trượt (MA - Moving Average), cùng với một thành phần tích hợp (I - Integrated) để xử lý xu hướng phi tuyến và không đồng đều trong dữ liệu.
- Dưới đây là giải thích về các thành phần chính của mô hình ARIMA:
 - Thành phần Tự hồi quy (AR - AutoRegressive): Thể hiện mối quan hệ giữa giá trị hiện tại và các giá trị trước đó trong chuỗi thời gian. Nếu mô hình ARIMA có thành phần AR, thì nó chứa các tham số AR (p) để xác định số lượng các giá trị trước đó cần được sử dụng trong dự đoán.
 - Thành phần Trung bình động trượt (MA - Moving Average): Thể hiện mối quan hệ giữa giá trị hiện tại và các giá trị nhiễu trước đó trong chuỗi thời gian. Nếu mô hình ARIMA có thành phần MA, thì nó chứa các tham số MA (q) để xác định số lượng các giá trị nhiễu trước đó cần được sử dụng trong dự đoán.
 - Thành phần Tích hợp (I - Integrated): Thể hiện số lượng lần tích hợp cần thiết để làm cho chuỗi thời gian trở nên dừng. Nếu mô hình ARIMA có thành phần I, thì nó chứa tham số I (d) xác định số lần tích hợp cần thiết.
- Công thức chính của mô hình ARIMA có thể được biểu diễn như sau:

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \dots + \phi_p Y_{t-p} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \dots + \theta_q \epsilon_{t-q} + \epsilon_t$$

- Ở đây:

- Y_t là giá trị của chuỗi thời gian tại thời điểm t ,
- c là hằng số
- $\phi_1, \phi_2, \dots, \phi_p$ là các hệ số tự hồi quy,
- $\theta_1, \theta_2, \dots, \theta_q$ là các hệ số trung bình động trượt
- ϵ_t là nhiễu ngẫu nhiên tại thời điểm t

Cách tìm thứ tự của số hạng AR (p)

- Bước tiếp theo là xác định xem mô hình có cần bất kỳ thuật ngữ AR nào không. Chúng ta sẽ tìm ra số lượng số hạng AR cần thiết bằng cách kiểm tra biểu đồ Tự tương quan một phần (PACF). Tự tương quan một phần có thể được hình dung như mối tương quan giữa chuỗi và độ trễ của nó, sau khi loại trừ những đóng góp từ độ trễ trung gian. Vì vậy, PACF dường như truyền tải mối tương quan thuần túy giữa độ trễ và chuỗi. Bằng cách này, chúng ta sẽ biết liệu độ trễ đó có cần thiết trong thuật ngữ AR hay không.
- Tự tương quan từng phần của độ trễ (k) của một chuỗi là hệ số của độ trễ đó trong phương trình tự hồi quy của Y .

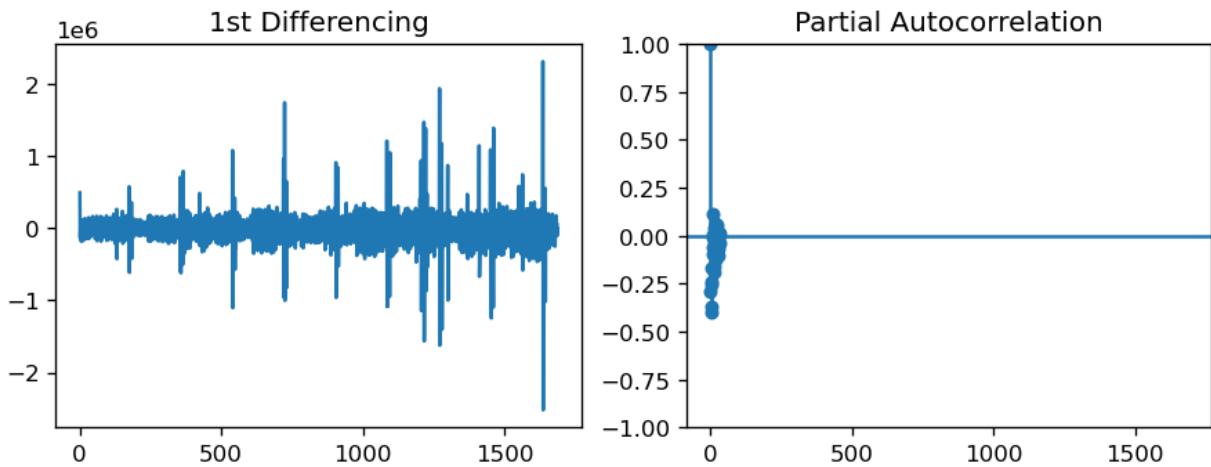
$$Y_t = \alpha_0 + \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + \alpha_3 Y_{t-3}$$

- Nghĩa là, giả sử nếu Y_t là chuỗi hiện tại và Y_{t-1} là độ trễ 1 của Y , thì tương quan một phần của độ trễ 3 (Y_{t-3}) là hệ số α_3 của Y_{t-3} trong phương trình trên.
- Bây giờ, chúng ta sẽ tìm thấy số lượng thuật ngữ AR. Bất kì mối tương quan tự động nào trong một chuỗi đứng yên đều có thể được khắc phục bằng cách thêm đủ các thuật ngữ AR. Vì vậy, ban đầu chúng ta lấy thứ tự của thuật ngữ AR bằng số độ trễ vượt qua giới hạn ý nghĩa trong biểu đồ PACF.

```
In [ ]: # PACF plot of 1st differenced series
plt.rcParams.update({'figure.figsize':(9,3), 'figure.dpi':120})

fig, axes = plt.subplots(1, 2, sharex=True)
axes[0].plot(train2.sales.diff()); axes[0].set_title('1st Differencing')
axes[1].set(ylim=(0,5))
plot_pacf(train2.sales.diff().dropna(), ax=axes[1])

plt.show()
```



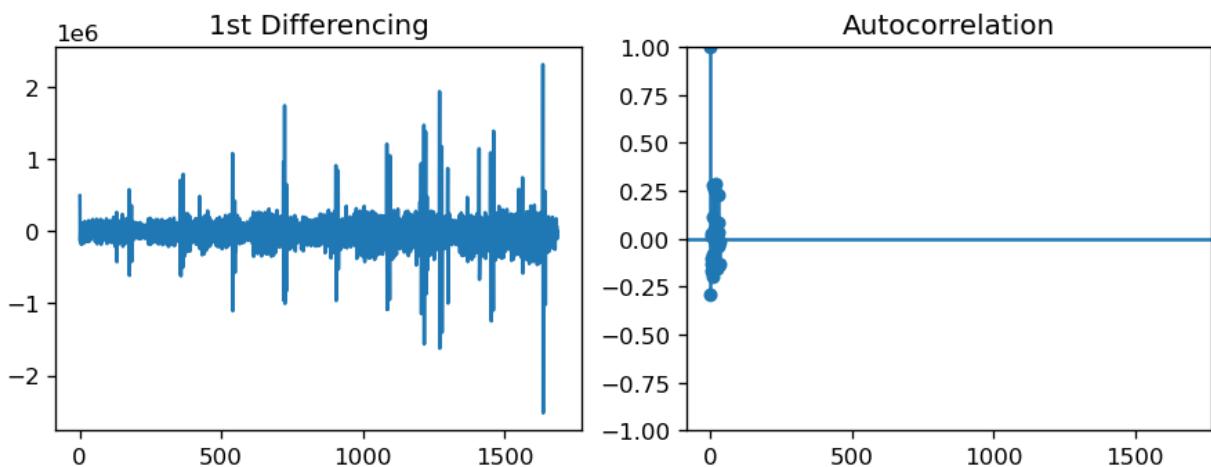
Cách tìm thứ tự của số hạng MA (q)

- Giống như cách chúng ta xem xét biểu đồ PACF cho số lượng thuật ngữ AR, chúng ta sẽ xem xét biểu đồ ACF cho số lượng thuật ngữ MA. Thuật ngữ MA về mặt kỹ thuật là lỗi của dự báo bị trễ.
- ACF cho biết cần bao nhiêu số hạng MA để loại bỏ bất kỳ hiện tượng tự tương quan nào trong chuỗi cố định.
- Hãy xem đồ thị tự tương quan của chuỗi sai phân.

```
In [ ]: from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
plt.rcParams.update({'figure.figsize':(9,3), 'figure.dpi':120})

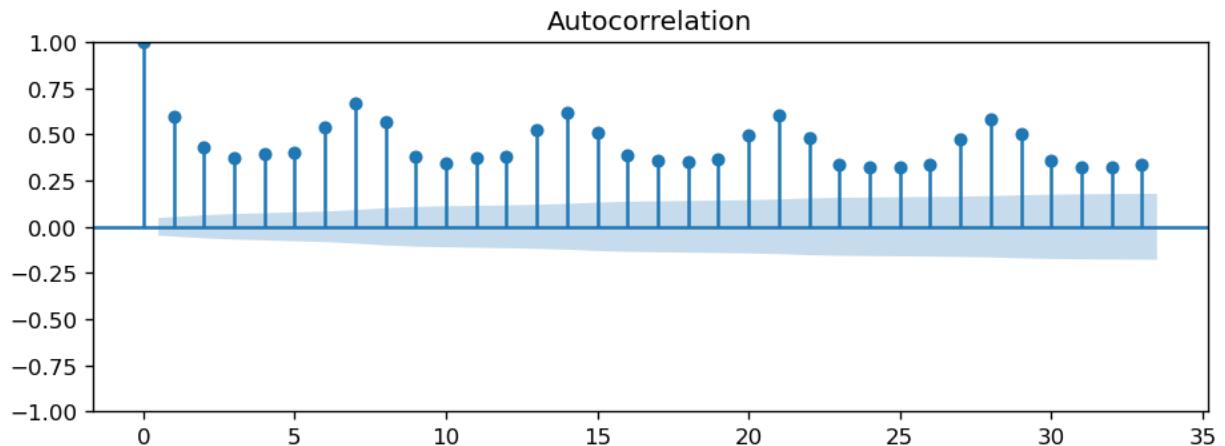
fig, axes = plt.subplots(1, 2, sharex=True)
axes[0].plot(train2.sales.diff()); axes[0].set_title('1st Differencing')
axes[1].set(ylim=(0,1.2))
plot_acf(train2.sales.diff().dropna(), ax=axes[1])

plt.show()
```



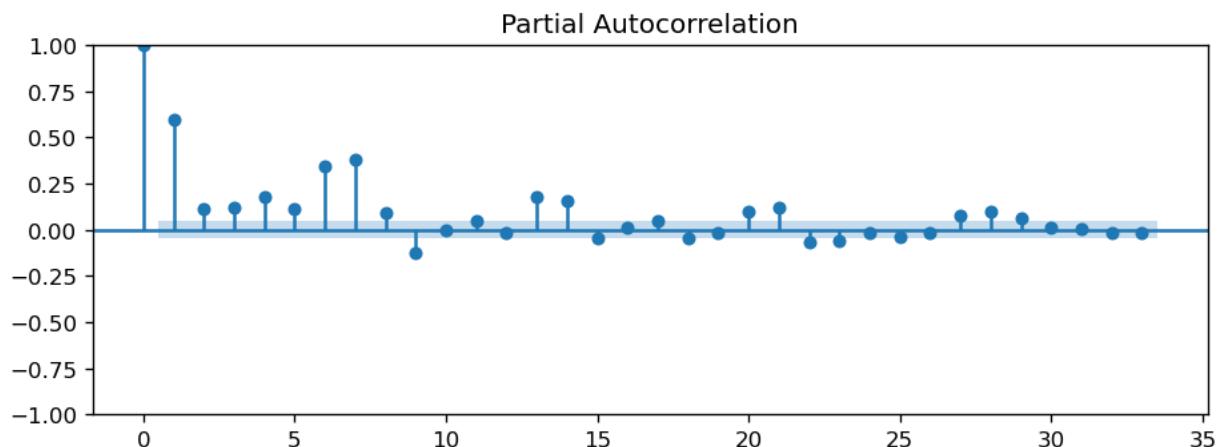
```
In [ ]: # Plot the autocorrelation function
plot_acf(train2['sales'], alpha=0.05)
```

```
# Show the plot  
plt.show()
```



- Nhìn vào kết quả, chúng ta có thể thấy được hệ số tương quan gần như là cao, có khi chạm đến 1 => có mối tương quan với doanh số bán hàng trong 1 và 7 ngày trước đây

```
In [ ]: from statsmodels.graphics.tsaplots import plot_pacf  
  
# Plot the autocorrelation function  
plot_pacf(train2['sales'], alpha=0.05)  
  
# Show the plot  
plt.show()
```



- Như chúng ta có thể thấy, chúng ta có mô hình theo mùa vào ngày 6,7 và 13, 14, tức là cứ 6-7 ngày chúng ta lại có doanh số tăng.

6. XÂY DỰNG MÔ HÌNH ARIMA

```
In [ ]: import pandas as pd  
from statsmodels.tsa.arima.model import ARIMA  
  
model = ARIMA(train2.sales, order=(1, 1, 2))
```

```
model_fit = model.fit()
print(model_fit.summary())
```

```
SARIMAX Results
=====
Dep. Variable:           sales    No. Observations:             1688
Model:                 ARIMA(1, 1, 2)    Log Likelihood:        -22998.405
Date: Mon, 25 Dec 2023   AIC:                   46004.811
Time: 03:51:11           BIC:                   46026.533
Sample:                0 - HQIC:                  46012.856
                           - 1688
Covariance Type: opg
=====
            coef      std err      z      P>|z|      [0.025      0.975]
-----
ar.L1      0.0615      0.093     0.662      0.508     -0.120      0.243
ma.L1     -0.7261      0.093    -7.813      0.000     -0.908     -0.544
ma.L2     -0.2186      0.087    -2.501      0.012     -0.390     -0.047
sigma2    4.296e+10  4.85e-12  8.85e+21      0.000     4.3e+10    4.3e+10
=====
Ljung-Box (L1) (Q):      0.00  Jarque-Bera (JB):       51195.29
Prob(Q):                  0.96  Prob(JB):                  0.00
Heteroskedasticity (H):    4.29  Skew:                      2.88
Prob(H) (two-sided):      0.00  Kurtosis:                 29.36
=====
```

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 1.76e+37. Standard errors may be unstable.

```
In [ ]: import statsmodels.api as sm

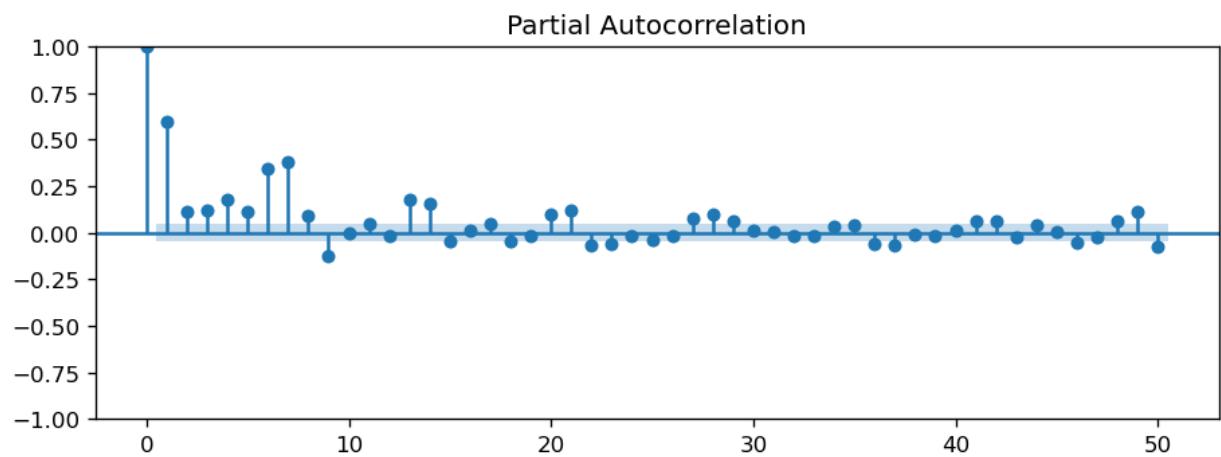
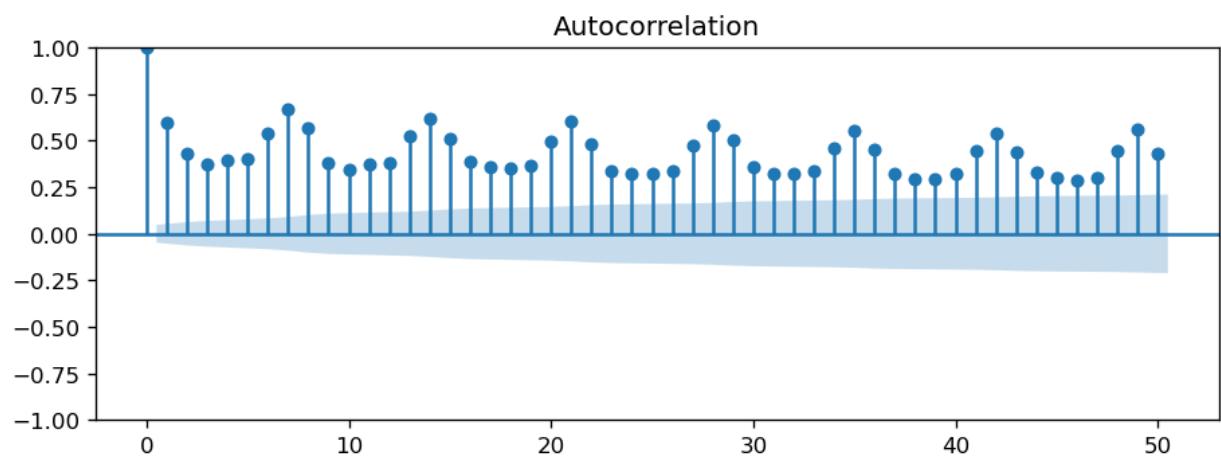
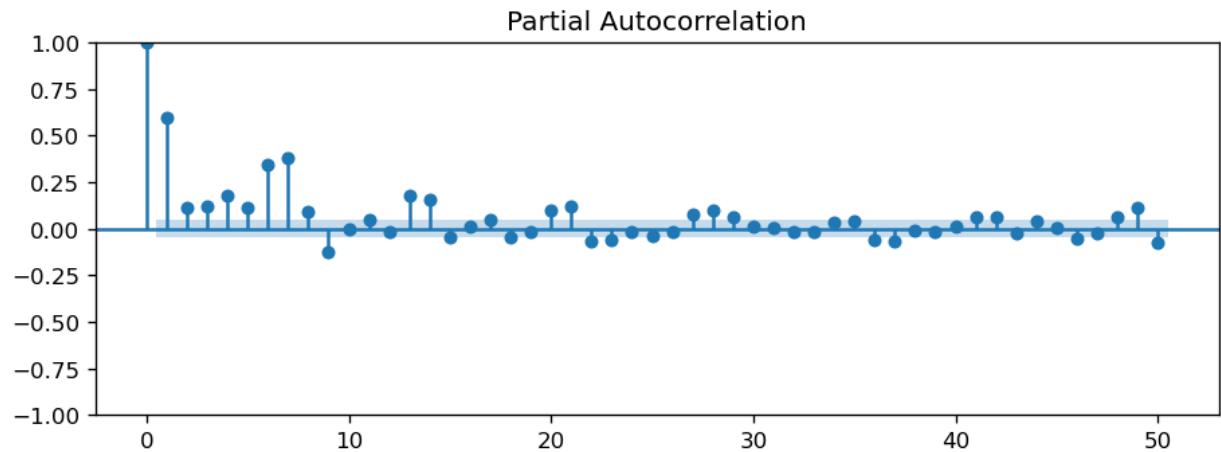
p = 1
d = 1
q = 1

train_np = train2['sales'].values.astype('float64')
model = sm.tsa.ARIMA(train_np, order=(p, d, q))

# Define the order of differencing, AR, and MA terms
model_fit = model.fit()

# Plot ACF and PACF
plot_acf(train_np, lags=50)
plot_pacf(train_np, lags=50)
```

Out[]:



- Như chúng ta có thể thấy, chỉ có diff_onpromotion có mức độ tương quan thấp với doanh số bán hàng. Chúng tôi đã tạo ra sự khác biệt bởi vì đôi khi mối tương quan nhất thiết có nghĩa là chúng có mối tương quan với nhau.

7. CHUẨN ĐOÁN MÔ HÌNH

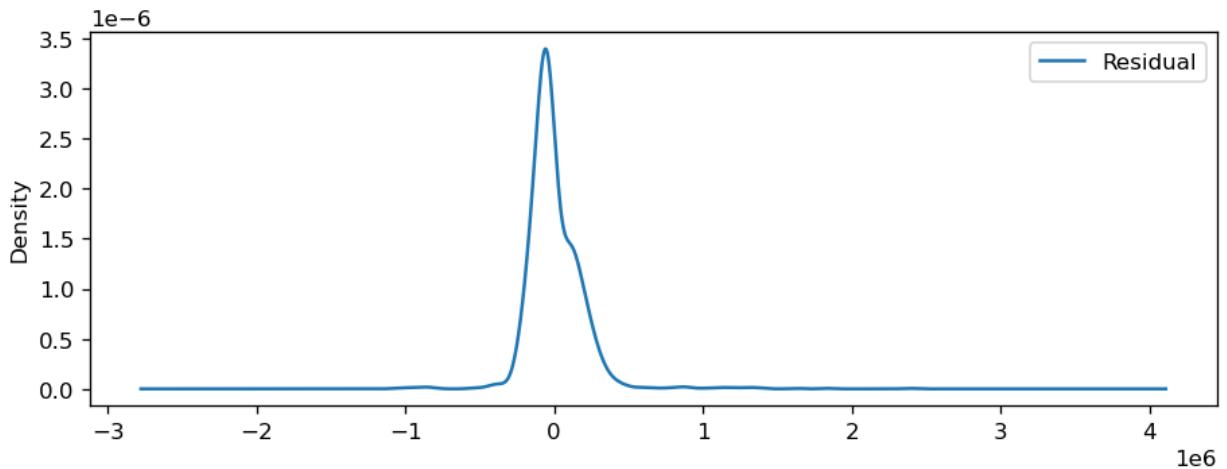
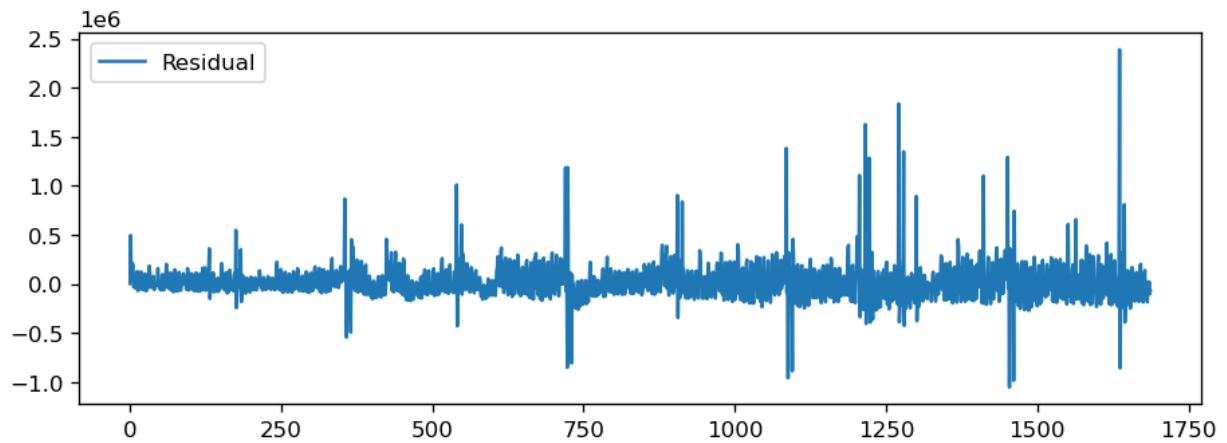
In []: `train2['date'] = pd.to_datetime(train2['date'])
train = train2.set_index('date')`

- Vẽ biểu đồ sai số theo thời gian
 - Biểu đồ này giúp xem xét xem có xu hướng nào trong các sai số và xác định xem chúng có giữ mức độ ngẫu nhiên không hay không.
 - Nếu có xu hướng hay các chu kỳ trong biểu đồ, có thể đề xuất rằng mô hình chưa hoàn toàn chứa đựng các thông tin có thể dự đoán được.

```
In [ ]: # Model diagnostics
residuals = pd.DataFrame(model_fit.resid, columns=['Residual'])
residuals.plot()
plt.show()

residuals.plot(kind='kde')
plt.show()

print(residuals.describe())
```



	Residual
count	1688.00
mean	6402.81
std	202042.69
min	-1055059.20
25%	-93456.86
50%	-32630.40
75%	95644.34
max	2388968.99

Biểu đồ KDE giúp xác định hình dạng của phân phối của các sai số. Ta thấy, phân phối gần với phân phối chuẩn, đồng thời không có đỉnh cao hay đuôi dài, ta nhận thấy mô hình có thể được coi là hiệu quả.

```
In [ ]: # define the order of differencing, AR, and MA terms
p = 1
d = 1
q = 1

# extract the target variable as a numpy array
train_np = train['sales'].values.astype('float64')

# fit the ARIMA model
model = sm.tsa.ARIMA(train_np, order=(p, d, q))

# train the ARIMA model
results = model.fit()

# print the summary of the trained model
print(results.summary())
```

```
SARIMAX Results
=====
Dep. Variable:                      y    No. Observations:                 1688
Model:                ARIMA(1, 1, 1)    Log Likelihood:            -23018.660
Date:          Mon, 25 Dec 2023    AIC:                         46043.321
Time:                  03:51:13    BIC:                         46059.613
Sample:                           0    HQIC:                        46049.354
                                - 1688
Covariance Type:                   opg
=====
            coef      std err       z     P>|z|      [0.025      0.975]
-----
ar.L1      0.2721      0.017   16.443      0.000      0.240      0.305
ma.L1     -0.9597      0.006  -150.131      0.000     -0.972     -0.947
sigma2    4.971e+10  1.24e-15   4.01e+25      0.000  4.97e+10  4.97e+10
=====
Ljung-Box (L1) (Q):                  1.59    Jarque-Bera (JB):        49785.13
Prob(Q):                               0.21    Prob(JB):                  0.00
Heteroskedasticity (H):                  4.27    Skew:                      2.89
Prob(H) (two-sided):                  0.00    Kurtosis:                 28.98
=====
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 8.03e+41. Standard errors may be unstable.
```

8. Tìm mô hình ARIMA tối ưu bằng cách sử dụng xác thực chéo ngoài thời gian

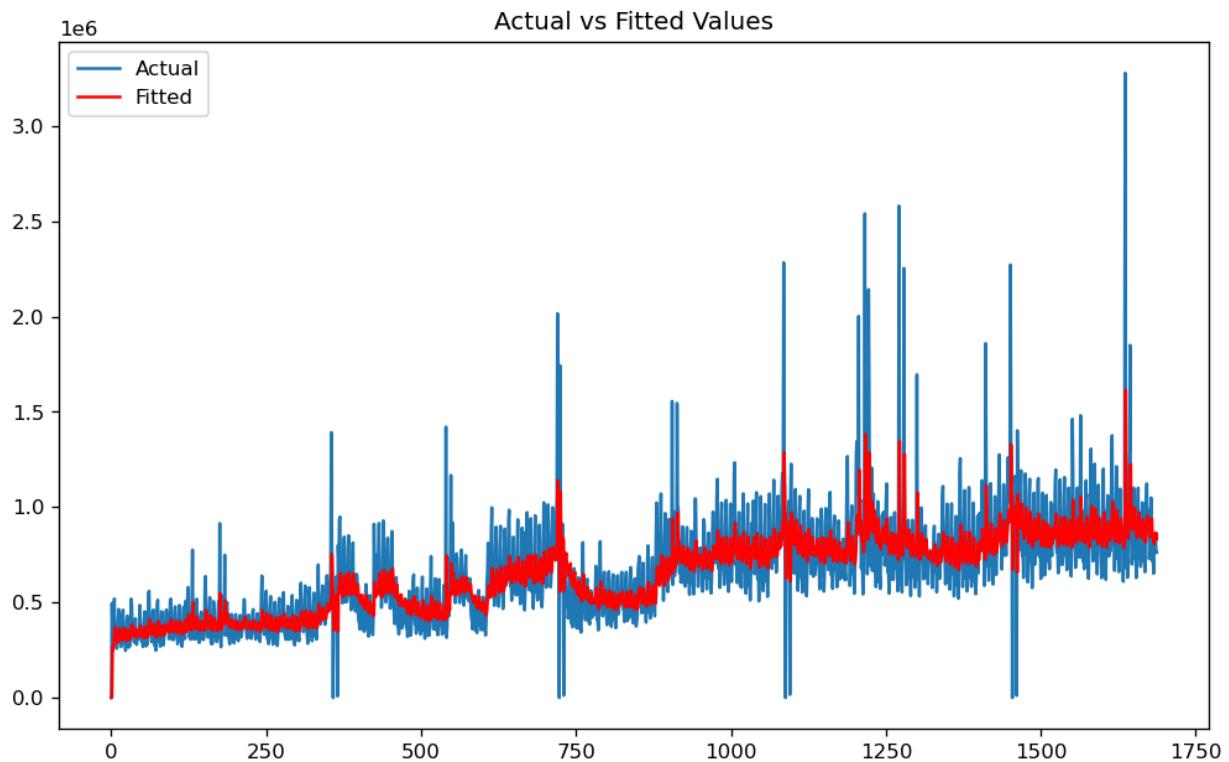
- Trong xác thực chéo ngoài thời gian, chúng tôi quay ngược thời gian và dự báo tương lai về số bước chúng tôi đã lùi lại. Sau đó, chúng tôi so sánh dự báo với thực tế.

- Để làm như vậy, chúng ta sẽ tạo tập dữ liệu huấn luyện và kiểm tra bằng cách chia chuỗi thời gian thành 2 phần liền kề theo tỷ lệ hợp lý dựa trên tần suất thời gian của chuỗi.
- Vẽ biểu đồ so sánh giữa dữ liệu thực tế và dữ liệu dự đoán:
 - Biểu đồ này so sánh giữa dữ liệu thực tế (Actual) và giá trị dự đoán của mô hình (Fitted). Nếu mô hình hoạt động tốt, các đường này sẽ khá gần nhau.
 - Nếu có sự chênh lệch lớn giữa dữ liệu thực tế và dự đoán, đặc biệt là ở những điểm quan trọng, có thể là dấu hiệu của việc mô hình cần được điều chỉnh hoặc cải thiện.

In []: `from statsmodels.tsa.arima.model import ARIMA`

```
# 1,1,1 ARIMA Model
model = ARIMA(train2['sales'], order=(1, 1, 1))
model_fit = model.fit()

# Plotting actual vs. fitted
plt.figure(figsize=(10, 6))
plt.plot(train2['sales'], label='Actual')
plt.plot(model_fit.fittedvalues, color='red', label='Fitted')
plt.legend()
plt.title('Actual vs Fitted Values')
plt.show()
```



=> Kết quả cho ta thấy dự đoán và thực tế không có sự khác biệt lớn, rõ ràng mô hình của chúng ta hoạt động tốt

In []: `from statsmodels.tsa.stattools import acf`

```
# Create Training and Test
```

```
train_train = train.sales[:85]
test = train.sales[85:]
```

- Bây giờ, chúng ta sẽ xây dựng mô hình ARIMA trên tập dữ liệu huấn luyện, dự báo và vẽ biểu đồ.

```
In [ ]: # Build Model
model = ARIMA(train_train, order=(1, 1, 1))
fitted = model.fit()

# Forecast
fc = fitted.forecast(steps=1603, alpha=0.05) # 95% conf

# Make as pandas series
fc_series = pd.Series(fc, index=test.index)

# Plot
plt.figure(figsize=(12,5), dpi=100)
plt.plot(train_train, label='training')
plt.plot(test, label='actual')
plt.plot(fc_series, label='forecast')
plt.fill_between(test.index, fc_series - fc_series.std(), fc_series + fc_series.std(),
                 color='k', alpha=.15)
plt.title('Forecast vs Actuals')
plt.legend(loc='upper left', fontsize=8)
plt.show()
```

c:\Users\Quynh Nhu\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473:
ValueWarning:

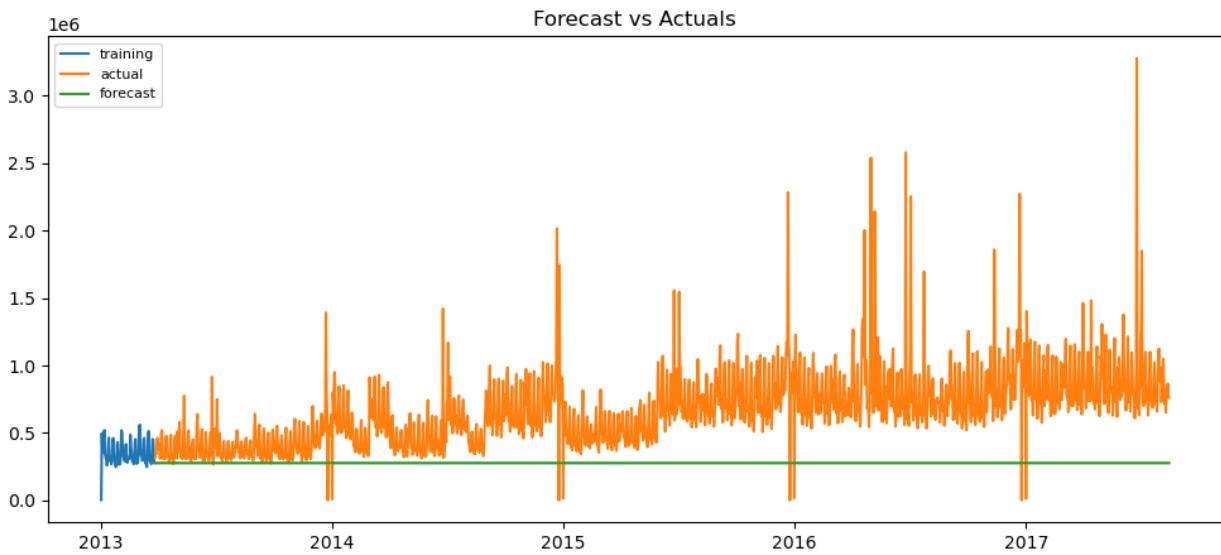
No frequency information was provided, so inferred frequency D will be used.

c:\Users\Quynh Nhu\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473:
ValueWarning:

No frequency information was provided, so inferred frequency D will be used.

c:\Users\Quynh Nhu\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473:
ValueWarning:

No frequency information was provided, so inferred frequency D will be used.



- Từ biểu đồ trên, mô hình ARIMA(1,1,1) dường như dự đoán chính xác. Các giá trị quan sát thực tế nằm trong khoảng tin cậy 95%.
- Tuy nhiên, chúng ta có thể thấy rằng dự báo được dự đoán luôn thấp hơn thực tế. Điều đó có nghĩa là bằng cách thêm một hằng số nhỏ vào dự báo của chúng tôi, độ chính xác chắc chắn sẽ được cải thiện. Vì vậy, trong trường hợp này, chúng ta nên tăng lặp lại p và q lên 3 để xem mô hình nào mang lại AIC ít nhất, đồng thời tìm kiếm biểu đồ cung cấp số liệu thực tế và dự báo gần hơn. Trong khi thực hiện việc này, tôi để ý đến giá trị P của các số hạng AR và MA trong bản tóm tắt mô hình. Lý tưởng nhất là chúng phải gần bằng 0, nhỏ hơn 0.05.

```
In [ ]: # Build Model
model = ARIMA(train_train, order=(3, 1, 1))
fitted = model.fit()
print(fitted.summary())

# Forecast
forecast_result = fitted.get_forecast(steps=1603, alpha=0.05) # 95% conf

# Get forecasted values and confidence intervals
fc_series = forecast_result.predicted_mean
conf = forecast_result.conf_int()

# Make as pandas series
lower_series = pd.Series(conf.iloc[:, 0].values, index=test.index)
upper_series = pd.Series(conf.iloc[:, 1].values, index=test.index)

# Plot
plt.figure(figsize=(12,5), dpi=100)
plt.plot(train_train, label='training')
plt.plot(test, label='actual')
plt.plot(fc_series, label='forecast')
plt.fill_between(test.index, lower_series, upper_series,
                 color='k', alpha=.15)
plt.title('Forecast vs Actuals')
plt.legend(loc='upper left', fontsize=8)
plt.show()
```

```
c:\Users\Quynh Nhu\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473:
ValueWarning:

No frequency information was provided, so inferred frequency D will be used.

c:\Users\Quynh Nhu\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473:
ValueWarning:

No frequency information was provided, so inferred frequency D will be used.

c:\Users\Quynh Nhu\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473:
ValueWarning:

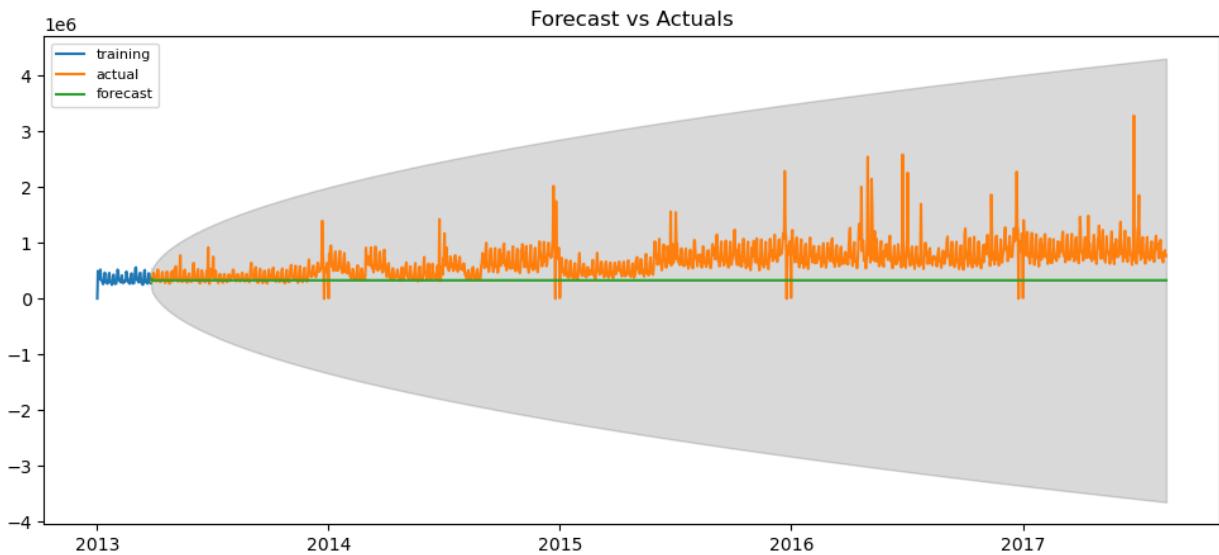
No frequency information was provided, so inferred frequency D will be used.
```

SARIMAX Results

Dep. Variable:	sales	No. Observations:	85			
Model:	ARIMA(3, 1, 1)	Log Likelihood	-1083.160			
Date:	Mon, 25 Dec 2023	AIC	2176.320			
Time:	03:51:13	BIC	2188.474			
Sample:	01-01-2013 - 03-26-2013	HQIC	2181.205			
Covariance Type:	opg					
	coef	std err	z	P> z	[0.025	0.975]
ar.L1	-0.1898	0.830	-0.229	0.819	-1.816	1.436
ar.L2	-0.5263	0.035	-14.853	0.000	-0.596	-0.457
ar.L3	-0.0233	0.478	-0.049	0.961	-0.960	0.913
ma.L1	0.2379	0.822	0.289	0.772	-1.374	1.850
sigma2	5.054e+09	8.92e-11	5.67e+19	0.000	5.05e+09	5.05e+09
Ljung-Box (L1) (Q):		0.02	Jarque-Bera (JB):		82.75	
Prob(Q):		0.88	Prob(JB):		0.00	
Heteroskedasticity (H):		0.44	Skew:		1.41	
Prob(H) (two-sided):		0.03	Kurtosis:		6.96	

Warnings:

- [1] Covariance matrix calculated using the outer product of gradients (complex-step).
- [2] Covariance matrix is singular or near-singular, with condition number 3.47e+36. Standard errors may be unstable.



Thông số của mô hình:

1. ARIMA(3, 1, 1):

- **AR (AutoRegressive):** Có 3 hệ số AR (ar.L1, ar.L2, ar.L3). Chúng đại diện cho mức độ ảnh hưởng của các giá trị trước đó đối với giá trị hiện tại.
- **I (Integrated):** $d=1$, đại diện cho việc chuỗi thời gian đã được chia đạo hàm một lần để đạt được tính chất dừng.
- **MA (Moving Average):** Có 1 hệ số MA (ma.L1), đại diện cho ảnh hưởng của sai số dự đoán trước đó đối với giá trị hiện tại.

2. Covariance Type (Loại hiệp biến):

- **opg (robust standard errors using the outer product of the gradient):** Sử dụng phương pháp robust để tính toán độ lệch chuẩn.

3. Sigma2 (Độ biến động của sai số):

- **sigma2:** Độ biến động của sai số, đo lường sự biến động của sai số trong mô hình.

Đo lường sự biến động của sai số:

1. Skewness (Chêch) và Kurtosis (Độ béo):

- **Skew (Chêch):** 1.41 - Đo lường mức độ chêch của phân phối sai số. Nếu giá trị là 0, phân phối là đối xứng.
- **Kurtosis (Độ béo):** 6.96 - Đo lường độ "đuôi" của phân phối, tức là sự tập trung của dữ liệu tại đuôi.

Kiểm định thống kê:

1. Ljung-Box (Q):

- **Ljung-Box (Q):** 0.02 - Kiểm tra tính tự tương quan của các sai số. Giá trị thấp (đặc biệt là giá trị p thấp) có thể chỉ ra sự tự tương quan có ý nghĩa.

2. Jarque-Bera (JB):

- **Jarque-Bera (JB):** 82.75 - Kiểm tra xem phân phối của sai số có tuân theo phân phối chuẩn hay không. Nếu giá trị p thấp, có thể phân phối không tuân theo phân phối chuẩn.

Kiểm định về không đồng nhất của sai số:

1. Heteroskedasticity (H):

- **Heteroskedasticity (H):** 0.44 - Kiểm tra sự không đồng nhất của sai số. Giá trị thấp có thể chỉ ra sự không đồng nhất.

2. Prob(H) (two-sided):

- **Prob(H) (two-sided):** 0.03 - Giá trị p của kiểm định về không đồng nhất của sai số. Nếu giá trị p thấp, có thể có sự không đồng nhất trong sai số.

Tổng quan về kết quả:

- **Log Likelihood (LL):** -1083.160 - Giá trị cao hơn cho thấy mô hình phù hợp tốt với dữ liệu.
 - **AIC (Akaike Information Criterion):** 2176.320 - Giá trị thấp là tốt, giúp so sánh giữa các mô hình khác nhau.
 - **BIC (Bayesian Information Criterion):** 2188.474 - Tương tự như AIC, giá trị thấp là tốt.

Kết quả này cung cấp một cái nhìn tổng quan về hiệu suất của mô hình SARIMAX, với sự đánh giá về tính phù hợp của mô hình, phân phối của sai số, và các thông số đặc trưng khác. Cần lưu ý rằng việc đánh giá mô hình không chỉ dựa trên các giá trị p mà còn dựa trên sự hiểu biết sâu rộng về dữ liệu và ngữ cảnh cụ thể của vấn đề.

Performing stepwise search to minimize aic

ARIMA(1,0,1)(0,0,0)[0]	: AIC=46184.565, Time=0.24 sec
ARIMA(0,0,0)(0,0,0)[0]	: AIC=50260.822, Time=0.02 sec
ARIMA(1,0,0)(0,0,0)[0]	: AIC=46680.877, Time=0.04 sec
ARIMA(0,0,1)(0,0,0)[0]	: AIC=49252.571, Time=0.07 sec
ARIMA(2,0,1)(0,0,0)[0]	: AIC=46093.138, Time=0.46 sec
ARIMA(2,0,0)(0,0,0)[0]	: AIC=46556.937, Time=0.12 sec
ARIMA(3,0,1)(0,0,0)[0]	: AIC=46075.761, Time=0.46 sec
ARIMA(3,0,0)(0,0,0)[0]	: AIC=46470.144, Time=0.14 sec
ARIMA(3,0,2)(0,0,0)[0]	: AIC=inf, Time=1.08 sec
ARIMA(2,0,2)(0,0,0)[0]	: AIC=46048.087, Time=0.46 sec
ARIMA(1,0,2)(0,0,0)[0]	: AIC=46047.534, Time=0.40 sec
ARIMA(0,0,2)(0,0,0)[0]	: AIC=48975.717, Time=0.16 sec
ARIMA(1,0,3)(0,0,0)[0]	: AIC=46037.531, Time=0.61 sec
ARIMA(0,0,3)(0,0,0)[0]	: AIC=48949.910, Time=0.24 sec
ARIMA(2,0,3)(0,0,0)[0]	: AIC=46028.273, Time=1.18 sec
ARIMA(3,0,3)(0,0,0)[0]	: AIC=45979.308, Time=1.20 sec
ARIMA(3,0,3)(0,0,0)[0] intercept	: AIC=46053.551, Time=1.19 sec

Best model: ARIMA(3,0,3)(0,0,0)[0]

Total fit time: 8.076 seconds

SARIMAX Results

Dep. Variable:	y	No. Observations:	1688			
Model:	SARIMAX(3, 0, 3)	Log Likelihood	-22982.654			
Date:	Mon, 25 Dec 2023	AIC	45979.308			
Time:	03:51:23	BIC	46017.327			
Sample:	0	HQIC	45993.388			
	- 1688					
Covariance Type:	opg					
coef	std err	z	P> z	[0.025	0.975]	
ar.L1	0.4938	0.028	17.369	0.000	0.438	0.550
ar.L2	-0.2660	0.038	-7.033	0.000	-0.340	-0.192
ar.L3	0.7716	0.023	33.035	0.000	0.726	0.817
ma.L1	-0.2095	0.026	-8.101	0.000	-0.260	-0.159
ma.L2	0.1696	0.023	7.303	0.000	0.124	0.215
ma.L3	-0.7861	0.017	-45.313	0.000	-0.820	-0.752
sigma2	3.613e+10	2.06e-13	1.75e+23	0.000	3.61e+10	3.61e+10
Ljung-Box (L1) (Q):	8.98	Jarque-Bera (JB):	51532.43			
Prob(Q):	0.00	Prob(JB):	0.00			
Heteroskedasticity (H):	4.41	Skew:	2.95			
Prob(H) (two-sided):	0.00	Kurtosis:	29.42			

Warnings:

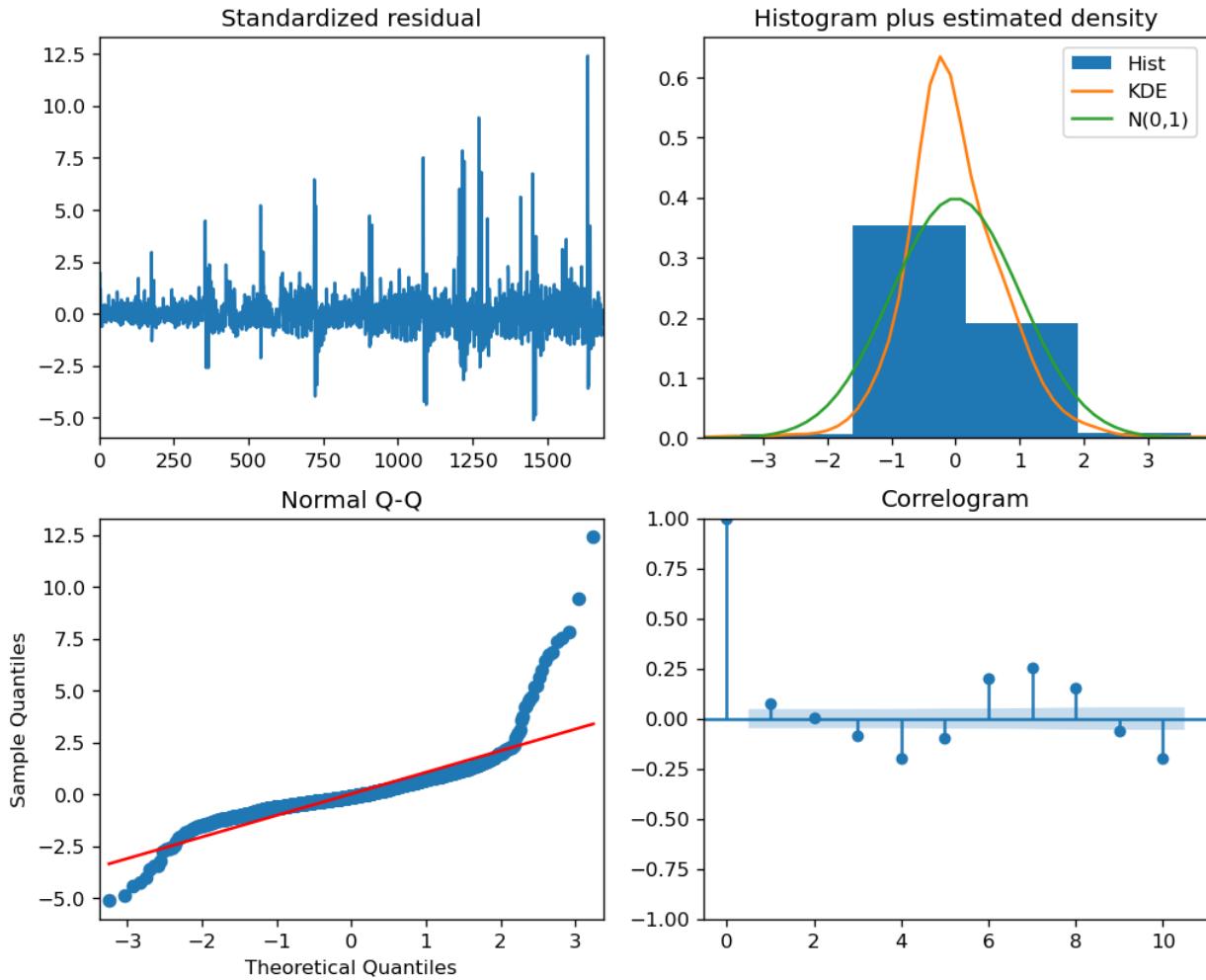
- [1] Covariance matrix calculated using the outer product of gradients (complex-step).
- [2] Covariance matrix is singular or near-singular, with condition number 2.7e+39. Standard errors may be unstable.

- Biểu đồ kiểm tra chất lượng mô hình:

- Biểu đồ này thường bao gồm các phần như histogram của sai số, biểu đồ QQ để so sánh phân phối sai số với phân phối chuẩn, biểu đồ tự tương quan của sai số, và biểu đồ năng suất đơn giản.

- Các biểu đồ này giúp kiểm tra giả định về phân phối và tính tương quan của sai số, đảm bảo rằng mô hình làm tốt trên dữ liệu.

```
In [ ]: model.plot_diagnostics(figsize=(10,8))
plt.show()
```



- Dự đoán và vẽ biểu đồ dự báo sử dụng mô hình ARIMA đã được xây dựng
 - Biểu đồ kết quả sẽ hiển thị dữ liệu gốc, giá trị dự đoán, và khoảng tin cậy.
 - Điều này giúp hiểu rõ hơn về khả năng dự đoán của mô hình và cung cấp thông tin về mức độ không chắc chắn của dự đoán trong tương lai.

```
In [ ]: # Forecast
n_periods = 1000
fc, confint = model.predict(n_periods=n_periods, return_conf_int=True)
index_of_fc = np.arange(len(train2.sales), len(train2.sales)+n_periods)

# make series for plotting purpose
fc_series = pd.Series(fc, index=index_of_fc)
lower_series = pd.Series(confint[:, 0], index=index_of_fc)
upper_series = pd.Series(confint[:, 1], index=index_of_fc)

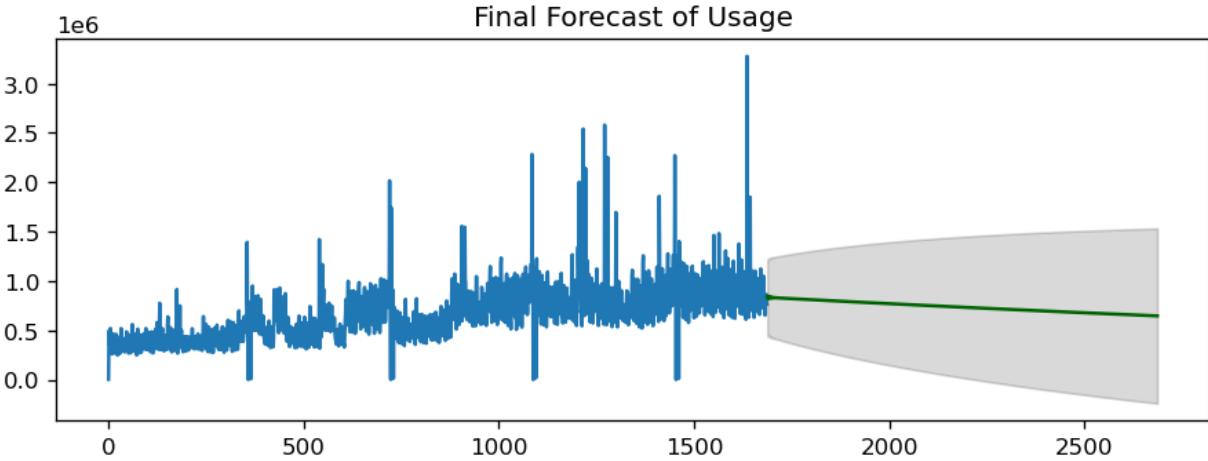
# Plot
plt.plot(train2.sales)
plt.plot(fc_series, color='darkgreen')
```

```

plt.fill_between(lower_series.index,
                 lower_series,
                 upper_series,
                 color='k', alpha=.15)

plt.title("Final Forecast of Usage")
plt.show()

```



In []: `from statsmodels.tsa.statespace.sarimax import SARIMAX`

```

# Train ARIMA model on sales data
model = SARIMAX(train['sales'], order=(2,1,2))
model_fit = model.fit()

# Predict sales for the next year
predicted_sales = model_fit.forecast(steps=365)

# Create a list of dates for the year 2018
dates_2018 = pd.date_range(start='2018-01-01', end='2018-12-31')

# Plot the predicted sales data for the year 2019
plt.plot(dates_2018, predicted_sales, label='Predicted Sales')

# Set the plot title and axis labels
plt.title('Predicted Sales for 2018')
plt.xlabel('Date')
plt.ylabel('Sales')

# Add a legend to the plot
plt.legend()

# Show the plot
plt.show()

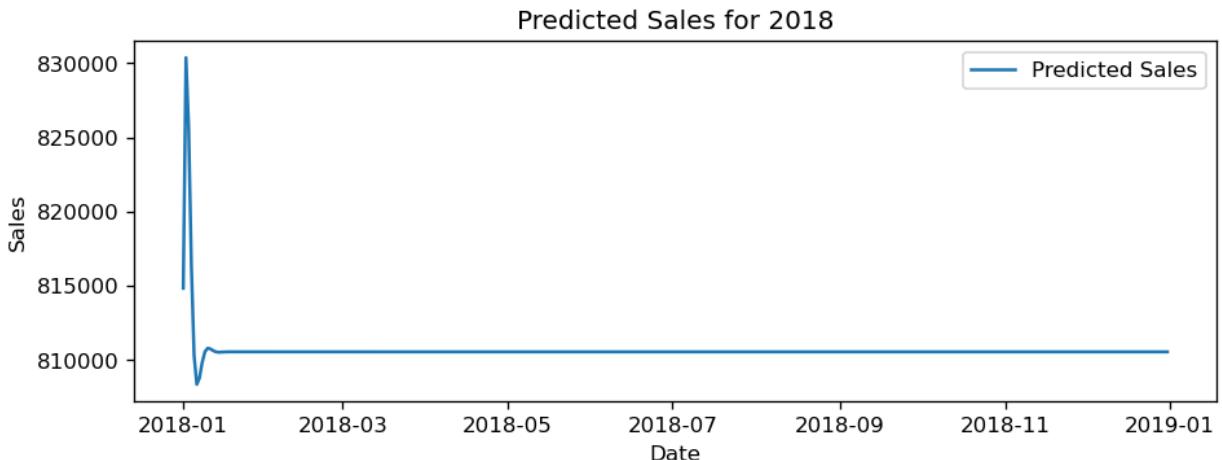
```

c:\Users\Quynh Nhu\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning:

No frequency information was provided, so inferred frequency D will be used.

c:\Users\Quynh Nhu\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning:

No frequency information was provided, so inferred frequency D will be used.



=> Kết quả dự đoán gần như là khá giống với những biểu đồ đã thể hiện trước đó, doanh số thường sẽ tăng vào tháng 1 và những tháng sau đó không có điều gì nổi bật. Dự đoán này chưa bao gồm những tác động không thể biết trước xảy ra, nếu như không có gì bất thường so với những năm trước, doanh số có thể sẽ được tiếp tục như vậy. Muốn nâng cao doanh số cần có những chiến lược phát triển khác, tùy từng thời điểm có những sự kiện nào xảy ra mà doanh số có thể tăng hoặc giảm khác nhau

```
In [ ]: # assuming your time series data is stored in a DataFrame called "test_data"
# create an empty DataFrame for the submission file
submission = pd.DataFrame()

# add the required columns to the submission DataFrame
submission['id'] = test.index
submission['sales'] = np.zeros(len(test))

# save the submission file as a CSV file
submission.to_csv('submission.csv', index=False)
```

9. Kết Luận về Phương Pháp Time Series Forecasting

Trong quá trình thực hiện Time Series Forecasting, chúng tôi đã thực hiện một chuỗi các bước phân tích và mô hình hóa với mục tiêu dự báo xu hướng và biến động của chuỗi thời gian. Dưới đây là một kết luận tổng quan về phương pháp đã sử dụng và những điều quan trọng đã học được:

1. Xác Định Dữ Liệu:

- Bắt đầu bằng việc xác định và thu thập dữ liệu phù hợp là quan trọng để đảm bảo tính chính xác và đầy đủ của mô hình.
- Chúng tôi cũng đã đưa ra các dữ liệu rõ ràng và giải thích cho từng loại dữ liệu, giúp hiểu hơn về tập dữ liệu

2. Tiền Xử Lý Dữ Liệu:

- Quá trình tiền xử lý dữ liệu giúp chuẩn bị tập dữ liệu chuỗi thời gian, bao gồm việc điều chỉnh các giá trị thiếu và chuẩn hóa dữ liệu.
- Việc tiền xử lý dữ liệu (loại bỏ null hay các giá trị bất thường) nhằm giúp dữ liệu của chúng tôi có tính logic hơn cho các dự đoán

3. Trực Quan Hóa Dữ Liệu:

- Sử dụng trực quan hóa dữ liệu, chúng tôi có cái nhìn tổng quan về xu hướng và mô hình các đặc điểm quan trọng của chuỗi thời gian.
- Trực quan hóa dữ liệu để giúp có cái nhìn rõ ràng hơn về doanh số so với những hoạt động khác.

4. Kiểm Tra Tính Dừng và Xác Định Bậc Sai Phân:

- Phân tích tính dừng và xác định bậc sai phân là bước quan trọng để đảm bảo tính ổn định của chuỗi thời gian.
- Chúng tôi đã xác định được dữ liệu không có tính dừng do đó chúng tôi đã ổn định dữ liệu để các dự báo và mức tương quan được chính xác hơn.

5. Xác Định Bậc Của Các Thuật Ngữ AR và MA:

- Bằng cách phân tích hàm tương quan riêng, chúng tôi xác định bậc của các thuật ngữ AR và MA trong mô hình ARIMA.

6. Lập Mô Hình ARIMA, SARIMAX:

- Áp dụng các mô hình ARIMA và SARIMAX để dự đoán xu hướng và biểu diễn dữ liệu chuỗi thời gian.

7. Kiểm Tra Mô Hình:

- Đánh giá mô hình thông qua kiểm tra chuẩn đoán, độ chệch, và các độ đo đánh giá hiệu suất.

8. Tìm Mô Hình ARIMA Tối Ưu:

- Sử dụng xác thực chéo ngoài thời gian để tối ưu hóa mô hình ARIMA và đảm bảo tính linh hoạt và hiệu suất tốt nhất.

9. Kết Luận và Chiến Lược Tương Lai:

- Tổng hợp kết quả và trí tuệ từ quá trình dự báo để đưa ra kết luận và đề xuất chiến lược cho quản lý và ra quyết định trong tương lai.
- Phương pháp này không chỉ cung cấp dự đoán mà còn mang lại hiểu biết sâu rộng về cấu trúc và đặc điểm của dữ liệu theo thời gian, hỗ trợ quyết định chiến lược và kế hoạch hành động.
- Thông qua quá trình phân tích, đối với một đất nước sản xuất dầu nên sự ảnh hưởng của giá dầu sẽ làm ảnh hưởng đến doanh số bán hàng. Các mối quan hệ đều có mối tương quan so với doanh số, tuy nhiên rõ ràng doanh số không có sự khác biệt to lớn như khi nhóm dự đoán trong năm 2018 . Rõ ràng vào năm 2022-2023, giá xăng dầu có sự biến động rất lớn trên toàn thế giới do các cuộc chiến tranh liên tục nổ ra, nếu như không có biện pháp thích hợp sẽ dễ dàng dẫn đến làm cho kinh tế đi xuống.

- Do đó, dựa vào sự trực quan hóa dữ liệu, khi xác định được city, state, store_type, cluster,, family, locale qua thời gian như tháng, năm, mùa, ngày trong tuần,... xác định được doanh số cao thì sẽ tập trung phát triển nhiều hơn ở những nơi đó. Đồng thời thông qua đó sẽ biết được nơi nào doanh số thấp thì sẽ đưa ra được các quyết định thích hợp để góp phần nâng cao doanh số.
- Việc sử dụng SMA - trung bình động đơn giản giữa sales và family - giúp chúng ta có cái nhìn rõ hơn về xu hướng tăng và giảm trong doanh số bán hàng đối với chênh lệch thời gian khác nhau. Rõ ràng đối với chênh lệch thời gian không có sự khác biệt to lớn. Điều này phản ánh và dự đoán xu hướng trong dữ liệu bán hàng của store với family, nhìn vào kết quả cho thấy đường đi với doanh số thực tế tương đương nhau, dựa vào điều này giúp chúng ta có cái nhìn tổng quan hơn về xu hướng doanh số trong tương lai.
- Việc sử dụng mô hình ARIMA đối với tập dữ liệu để dự đoán kết quả, chúng tôi đã kiểm tra xem mô hình đã hoạt động tốt chưa, đồng thời thay đổi q,p,d để có cái nhìn tốt nhất về đường dự đoán tương lai. nhìn chung không có gì khác biệt so với những năm trước đó (2013-2017), rõ ràng nếu cứ duy trì những phương pháp phát triển kinh tế như hiện tại thì doanh số sẽ không có gì thay đổi. Dựa vào những điều này giúp chúng ta có cái nhìn khái quát và cơ bản về dự đoán doanh số cho tương lai. Góp phần giúp phát triển doanh số, nâng cao nền kinh tế.