Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Report on Mini Project

Time Series Analysis (DJ19DSC5012)
AY: 2023-24

Time Series Analysis of Sales Data in Global Superstore

Kresha Shah – 60009220080

Durva Patel – 60009220088

Guided By:

Prof. Shruti Mathur

Table of Contents

Sr. No.	Topic	Pg. No.
1.	Introduction	3
2.	Data Description	4
3.	Objective	5
4.	Data Cleaning & Preprocessing	6-13
5.	Data Decomposition	14-17
6.	Data Smoothing	18-19
7.	Testing Stationary	20- 21
8.	Justification why it is a time series problem.	22
9.	Implementation and Interpretation for forecast	23-24
10.	Comparative Analysis	25
10.	Reasons For Selecting the Time Series Model	26-30
11.	Conclusion & Colab	31

Introduction

The Global Superstore project focuses on conducting an in-depth time series analysis on the sales data sourced from the Global Superstore. Time series analysis is a robust analytical method employed to comprehend and predict trends, patterns, and fluctuations over a sequential period. This analytical approach holds particular significance in the domain of retail sales, as it provides a understanding of sales dynamics, aiding in the identification of patterns, seasonality, and potential areas for enhancement.

The fundamental objective of this project is to extract meaningful insights from the historical sales data, offering a comprehensive view of how sales have evolved over time. By employing time series analysis techniques, we seek to unravel the underlying structures and variations within the sales data, shedding light on the factors influencing customer purchasing behaviours. Moreover, the predictive aspect of time series analysis enables us to forecast future sales trends, empowering decision-makers to make informed strategic choices.

Data Description

Order Date	Segment	Market	Sales	Profit
2012-07-31	Consumer	US	2309.650	762.1845
2013-05-02	Corporate	APAC	3709.395	-288.7650
2013-10-17	Consumer	APAC	5175.171	919.9710
2013-01-28	Home Office	EU	2892.510	-96.5400
2013-05-11	Consumer	Africa	2832.960	311.5200

The dataset used for this analysis is the Global Superstore dataset, which contains historical sales data.

The dataset includes various attributes such as:

- order date
- product category
- sales
- region

among others. For this project, we will focus specifically on the sales column.

Objective

Objective of the above Time series data includes:

1. Data analysis and cleaning
2. Finding out trends in data
3. Checking for Seasonality
4. Smoothening the data
5. Identifying parameters of the model
6. Fitting the right model
7. Calculating accuracy matrix
8. Forecasting future prices

Data Cleaning & Preprocessing

```
import pandas as pd
df = pd.read_csv(r'/content/drive/MyDrive/datasets/GlobalSuperstoreData.csv')
df.head(5)
```

	Order Date	Segment	Market	Sales	Profit
0	31-07-2012	Consumer	US	2309.650	762.1845
1	05-02-2013	Corporate	APAC	3709.395	-288.7650
2	17-10-2013	Consumer	APAC	5175.171	919.9710
3	28-01-2013	Home Office	EU	2892.510	-96.5400
4	05-11-2013	Consumer	Africa	2832.960	311.5200

```
df[(df['Sales'] >=3000) & (df['Sales'] <=5000)]
```

	Order Date	Segment	Market	Sales	Profit
1	05-02-2013	Corporate	APAC	3709.395	-288.7650
9	28-01-2012	Consumer	US	4297.644	-1862.3124
10	05-04-2011	Corporate	US	4164.050	83.2810
11	19-04-2012	Corporate	APAC	4626.150	647.5500
14	06-06-2013	Consumer	APAC	3701.520	1036.0800
...	...	...	...	...	...
13013	02-04-2011	Consumer	APAC	3195.000	1150.2000
19864	20-08-2014	Consumer	APAC	3410.820	136.2600
20228	17-09-2012	Consumer	US	4228.704	158.5764
20343	04-06-2014	Corporate	EU	4363.350	305.4000
22238	02-04-2014	Corporate	US	4799.984	359.9988

191 rows × 5 columns

```
[ ] df['Market'].unique()
```

```
array(['US', 'APAC', 'EU', 'Africa', 'EMEA', 'LATAM', 'Canada'],
      dtype=object)
```

```
df[df.Market.isin(['US','EU','EMEA'])]
```

	Order Date	Segment	Market	Sales	Profit
0	31-07-2012	Consumer	US	2309.650	762.1845
3	28-01-2013	Home Office	EU	2892.510	-96.5400
8	14-10-2014	Corporate	US	5083.960	1906.4850
9	28-01-2012	Consumer	US	4297.644	-1862.3124
10	05-04-2011	Corporate	US	4164.050	83.2810
...	...	...	...	...	...
51276	09-06-2014	Corporate	US	1.624	-4.4660
51277	25-12-2012	Consumer	US	17.940	8.0730
51280	29-11-2014	Home Office	EMEA	34.128	-49.5720
51286	20-06-2014	Consumer	US	0.444	-1.1100
51287	02-12-2013	Home Office	US	22.920	11.2308

25023 rows × 5 columns

```
df[~df.Market.isin(['US','EU','EMEA'])]
```

	Order Date	Segment	Market	Sales	Profit
1	05-02-2013	Corporate	APAC	3709.395	-288.765
2	17-10-2013	Consumer	APAC	5175.171	919.971
4	05-11-2013	Consumer	Africa	2832.960	311.520
5	28-06-2013	Corporate	APAC	2862.675	763.275
6	07-11-2011	Consumer	APAC	1822.080	564.840
...	...	...	...	...	...
51283	30-05-2014	Corporate	APAC	26.940	1.860
51284	05-08-2014	Home Office	APAC	58.050	19.950
51285	19-06-2014	Corporate	APAC	65.100	4.500
51288	18-02-2012	Home Office	LATAM	13.440	2.400
51289	22-05-2012	Consumer	LATAM	61.380	1.800

26267 rows × 5 columns

```
df.sort_values(by= ['Profit','Order Date'], ascending= True)
```

	Order Date	Segment	Market	Sales	Profit
171	26-11-2013	Consumer	US	4499.985	-6599.9780
6591	26-09-2013	Corporate	EMEA	3085.344	-4088.3760
37	05-11-2014	Corporate	US	7999.980	-3839.9904
1341	26-07-2011	Consumer	US	2177.584	-3701.8928
2347	18-04-2014	Home Office	US	2549.985	-3399.9800
...	...	...	...	...	...
45	23-09-2011	Consumer	US	9449.950	4630.4755
122	18-12-2013	Consumer	US	9892.740	4946.3700
290	18-11-2014	Consumer	US	10499.970	5039.9856
14843	24-03-2014	Consumer	US	13999.960	6719.9808
329	03-10-2013	Corporate	US	17499.950	8399.9760

51290 rows × 5 columns

```
df.sort_values(by= ['Profit','Order Date'], ascending= False)
```

	Order Date	Segment	Market	Sales	Profit
329	03-10-2013	Corporate	US	17499.950	8399.9760
14843	24-03-2014	Consumer	US	13999.960	6719.9808
290	18-11-2014	Consumer	US	10499.970	5039.9856
122	18-12-2013	Consumer	US	9892.740	4946.3700
45	23-09-2011	Consumer	US	9449.950	4630.4755
...	...	...	...	...	...
2347	18-04-2014	Home Office	US	2549.985	-3399.9800
1341	26-07-2011	Consumer	US	2177.584	-3701.8928
37	05-11-2014	Corporate	US	7999.980	-3839.9904
6591	26-09-2013	Corporate	EMEA	3085.344	-4088.3760
171	26-11-2013	Consumer	US	4499.985	-6599.9780

51290 rows × 5 columns

```
df.agg({'Sales': ['count','min', 'max', 'mean']})
```

	Sales
count	51290.000000
min	0.444000
max	22638.480000
mean	246.490581

```
df.groupby('Market')['Sales'].sum()
```

```
Market
APAC      3.585744e+06
Africa    7.837732e+05
Canada    6.692817e+04
EMEA      8.061613e+05
EU        2.938089e+06
LATAM     2.164605e+06
US        2.297201e+06
Name: Sales, dtype: float64
```

```
df.groupby('Market').agg({'Profit':['mean','min','max']})
```

	Profit		
	mean	min	max
Market			
APAC	39.629163	-3009.435	2939.310
Africa	19.374674	-1576.824	2597.280
Canada	46.399453	0.000	1159.020
EMEA	8.728966	-4088.376	1622.880
EU	37.282974	-3059.820	3979.080
LATAM	21.531328	-1806.240	1313.280
US	28.656896	-6599.978	8399.976

```
df.describe()
```

	Sales	Profit
count	51290.000000	51290.000000
mean	246.490581	28.610982
std	487.565361	174.340972
min	0.444000	-6599.978000
25%	30.758625	0.000000
50%	85.053000	9.240000
75%	251.053200	36.810000
max	22638.480000	8399.976000

```
df['Date'] = pd.to_datetime(df['Order Date'])
df.set_index('Date', inplace=True)
df_2011 = df['2011']
monthly_mean_2011 = df_2011['Sales'].resample('M').mean()
df.Sales.resample('M').mean()
```

```
Date
2011-01-31    222.610790
2011-02-28    252.752698
2011-03-31    262.565993
2011-04-30    210.856744
2011-05-31    263.606265
2011-06-30    236.377578
2011-07-31    241.842832
2011-08-31    268.656244
2011-09-30    265.596147
2011-10-31    278.848881
2011-11-30    239.614597
2011-12-31    256.906826
2012-01-31    258.824950
2012-02-29    245.041977
2012-03-31    244.670786
2012-04-30    260.013817
2012-05-31    239.562539
2012-06-30    233.855321
2012-07-31    230.679931
2012-08-31    273.584754
2012-09-30    235.383351
2012-10-31    257.611517
2012-11-30    211.667751
2012-12-31    256.566198
```

```
df.Sales.resample('Q').mean()
```

```
Date
2011-03-31    245.944046
2011-06-30    236.230309
2011-09-30    260.112828
2011-12-31    257.121466
2012-03-31    249.070700
2012-06-30    242.769305
2012-09-30    247.492618
2012-12-31    239.694108
2013-03-31    256.669294
2013-06-30    252.689378
2013-09-30    227.890679
2013-12-31    253.754396
2014-03-31    249.607021
2014-06-30    234.556659
2014-09-30    253.827255
2014-12-31    243.850344
Freq: Q-DEC, Name: Sales, dtype: float64
```

```
df.rolling(window=4).mean().head(10)
```

Date	Sales	Profit
2012-07-31	NaN	NaN
2013-05-02	NaN	NaN
2013-10-17	NaN	NaN
2013-01-28	3521.68150	324.212625
2013-05-11	3652.50900	211.546500
2013-06-28	3440.82900	474.556500
2011-07-11	2602.55625	385.773750
2012-04-14	3190.63875	659.028750
2014-10-14	3753.38875	1057.770000
2012-01-28	4112.13100	401.373150

```
[ ] df.expanding(min_periods=4).mean().head(10)
```

Shifting

```
df.shift(periods=3).head(7)
```

Date	Order Date	Segment	Market	Sales	Profit
2012-07-31	None	None	None	NaN	NaN
2013-05-02	None	None	None	NaN	NaN
2013-10-17	None	None	None	NaN	NaN
2013-01-28	31-07-2012	Consumer	US	2309.650	762.1845
2013-05-11	05-02-2013	Corporate	APAC	3709.395	-288.7650
2013-06-28	17-10-2013	Consumer	APAC	5175.171	919.9710
2011-07-11	28-01-2013	Home Office	EU	2892.510	-96.5400

```
[ ] df.shift(periods=-1).head(7)
```

```
df.shift(periods=3, axis=1).head(7)
```

Date	Order Date	Segment	Market	Sales	Profit
2012-07-31	None	None	None	31-07-2012	Consumer
2013-05-02	None	None	None	05-02-2013	Corporate
2013-10-17	None	None	None	17-10-2013	Consumer
2013-01-28	None	None	None	28-01-2013	Home Office
2013-05-11	None	None	None	05-11-2013	Consumer
2013-06-28	None	None	None	28-06-2013	Corporate
2011-07-11	None	None	None	07-11-2011	Consumer

Handling missing values

```
df.isnull().sum()
```

```
Order Date    0
Segment       0
Market        0
Sales         0
Profit        0
dtype: int64
```

```
df.interpolate(method='linear',limit_direction='forward')
```

Date	Order Date	Segment	Market	Sales	Profit
2012-07-31	31-07-2012	Consumer	US	2309.650	762.1845
2013-05-02	05-02-2013	Corporate	APAC	3709.395	-288.7650
2013-10-17	17-10-2013	Consumer	APAC	5175.171	919.9710
2013-01-28	28-01-2013	Home Office	EU	2892.510	-96.5400
2013-05-11	05-11-2013	Consumer	Africa	2832.960	311.5200
...	...	...	...	...	...
2014-06-19	19-06-2014	Corporate	APAC	65.100	4.5000

Before initiating the time series analysis, a crucial phase involves preparing and refining the raw data to ensure its suitability for analysis. The data preprocessing steps undertaken for the Global Superstore sales dataset are detailed below. The first step involved a thorough check for missing values within the dataset. Missing values can distort analyses and model performance. To extract meaningful insights, the sales data was aggregated based on a specific time frequency. This step involved grouping the data by day, month, or another relevant time unit, depending on the objectives of the analysis. Aggregating data helps in smoothing out noise and revealing underlying patterns. Outliers, extreme values that deviate significantly from the majority of the data, were addressed to prevent them from unduly influencing the analysis. Techniques such as statistical methods or visualizations, like box plots, were employed to identify and handle outliers appropriately.

Data Decomposition

Trends and Detrending

Any Time Series data consists of 3 components:

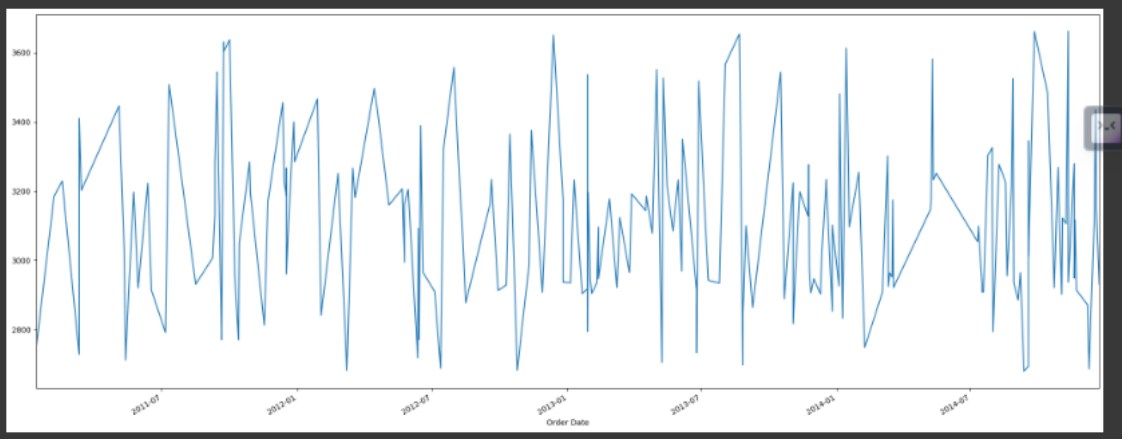
1. Trend
2. Seasonal
3. Residual

Trends

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

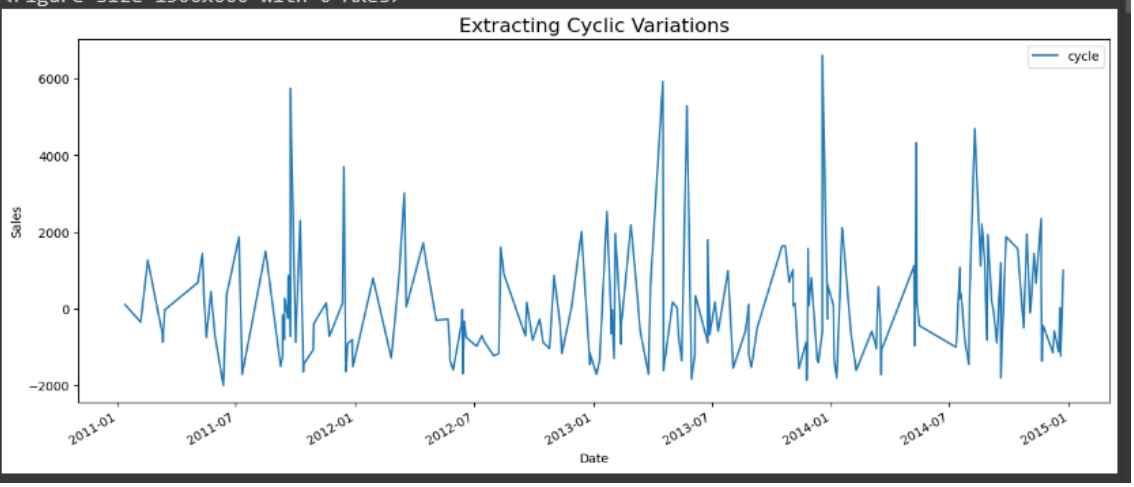
```
[ ] import pandas as pd
    %matplotlib inline
    from statsmodels.tsa.filters.hp_filter import hpfilter
    df = pd.read_csv('/content/drive/MyDrive/datasets/GlobalSuperstoreData.csv', index_col=0, pars
    sample_size = 200
    df_sample = df.iloc[:sample_size]
    Sales_cycle, Sales_trend = hpfilter(df_sample['Sales'], lamb=3000)
    Sales_trend.plot(figsize=(25, 10)).autoscale(axis='x', tight=True)
```



Cyclic variations

```
from statsmodels.tsa.filters.hp_filter import hpfilter
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
df = pd.read_csv('/content/drive/MyDrive/datasets/GlobalSuperstoreData.csv', parse_dates=True,
sample_size = 200
df_sample = df.iloc[:sample_size]
Sales_cycle, Sales_trend = hpfilter(df_sample['Sales'], lamb=1600)
df_sample['cycle'] = Sales_cycle
df_sample['trend'] = Sales_trend
plt.figure(figsize=(15, 6))
df_sample[['cycle']].plot()
plt.title('Extracting Cyclic Variations', fontsize=16)
plt.xlabel('Date')
plt.ylabel('Sales')
plt.show()
```

```
<Figure size 1500x600 with 0 Axes>
```



From the above codes and visualizations, we can conclude that there are no trends in the data. In fact, the data shows high volatility.

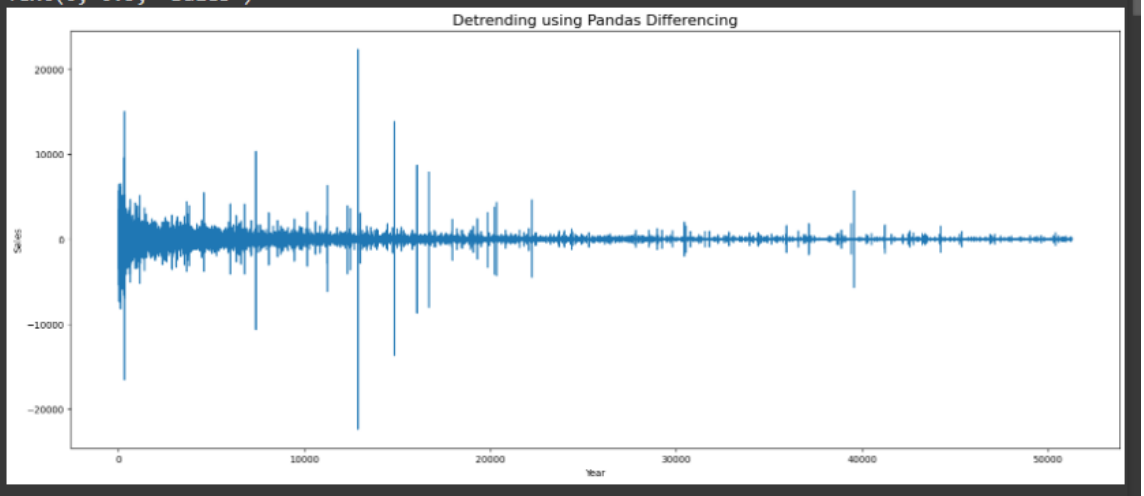
Detrending:

Detrending is a process used to remove or reduce the trend component from a time series. Detrending is beneficial when you want to analyze the underlying patterns or variations in the data without the influence of a long-term trend. Several methods can be employed to detrend time series data

Pandas Differencing

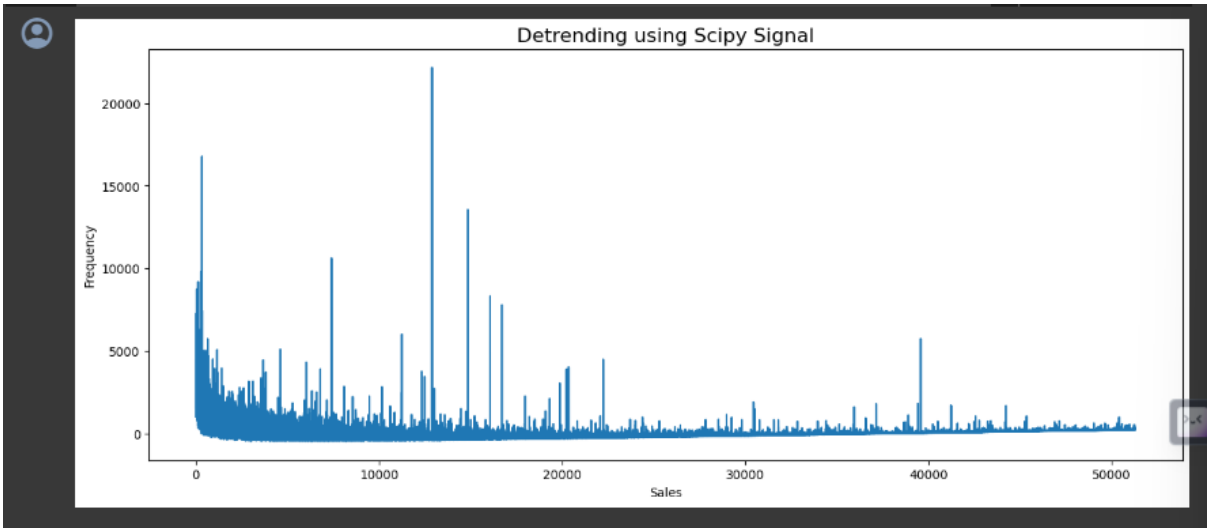
```
import pandas as pd
from pandas import datetime
from matplotlib import pyplot
series= pd.read_csv('/content/drive/MyDrive/datasets/GlobalSuperstoreData.csv', index_col=0,pa
X= series['Sales']
diff= list()
for i in range(1, len(X)):
  value= X[i] -X[i-1]
  diff.append(value)
pyplot.figure(figsize=(20,8))
pyplot.plot(diff)
pyplot.title('Detrending using Pandas Differencing', fontsize=16)
pyplot.xlabel('Year')
pyplot.ylabel('Sales')
```

```
Text(0, 0.5, 'Sales')
```



Scipy Signal

```
import pandas as pd
import matplotlib.pyplot as plt
from scipy import signal
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
df= pd.read_csv('/content/drive/MyDrive/datasets/GlobalSuperstoreData.csv', index_col=0,parse_
detrended= signal.detrend(df.Sales.values)
plt.figure(figsize=(15,6))
plt.plot(detrended)
plt.xlabel('Sales')
plt.ylabel('Frequency')
plt.title('Detrending using Scipy Signal', fontsize=16)
plt.show()
```



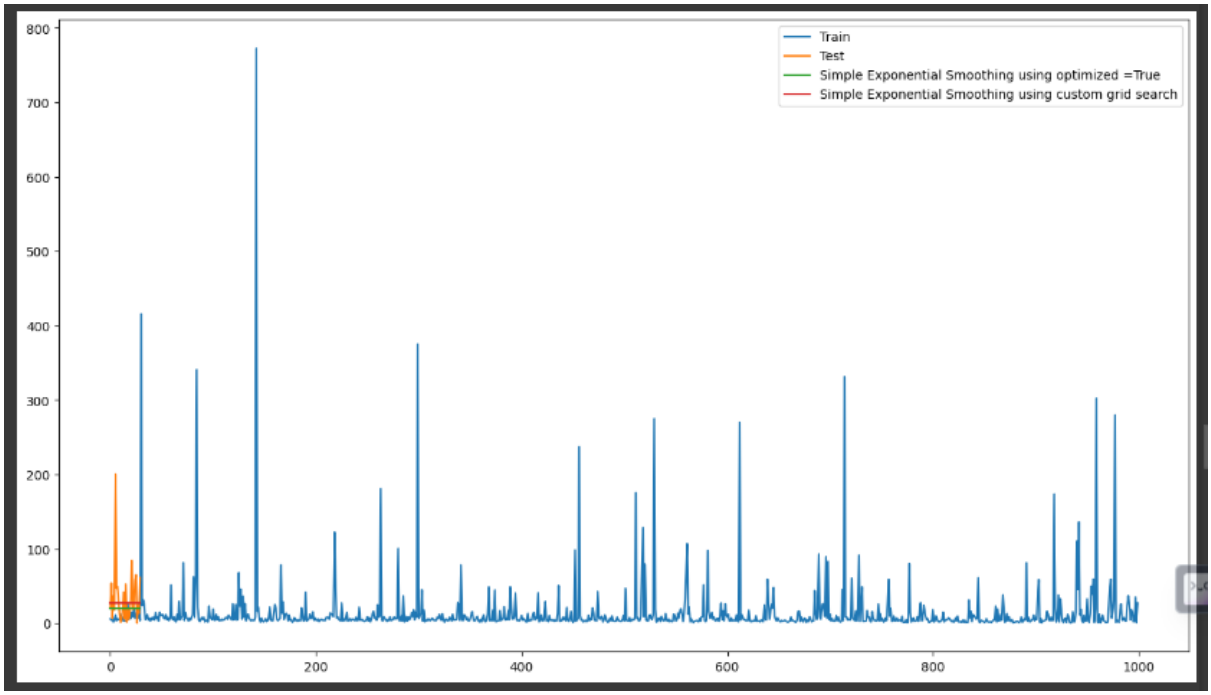
In the above code we have performed detrending using pandas differencing and Scipy Signal. By performing detrending we have somewhat reduced the volatility in the data.

Data Smoothening

Smoothing is a method we can use to create a function to remove irregularities in data and attempt to capture significant patterns.

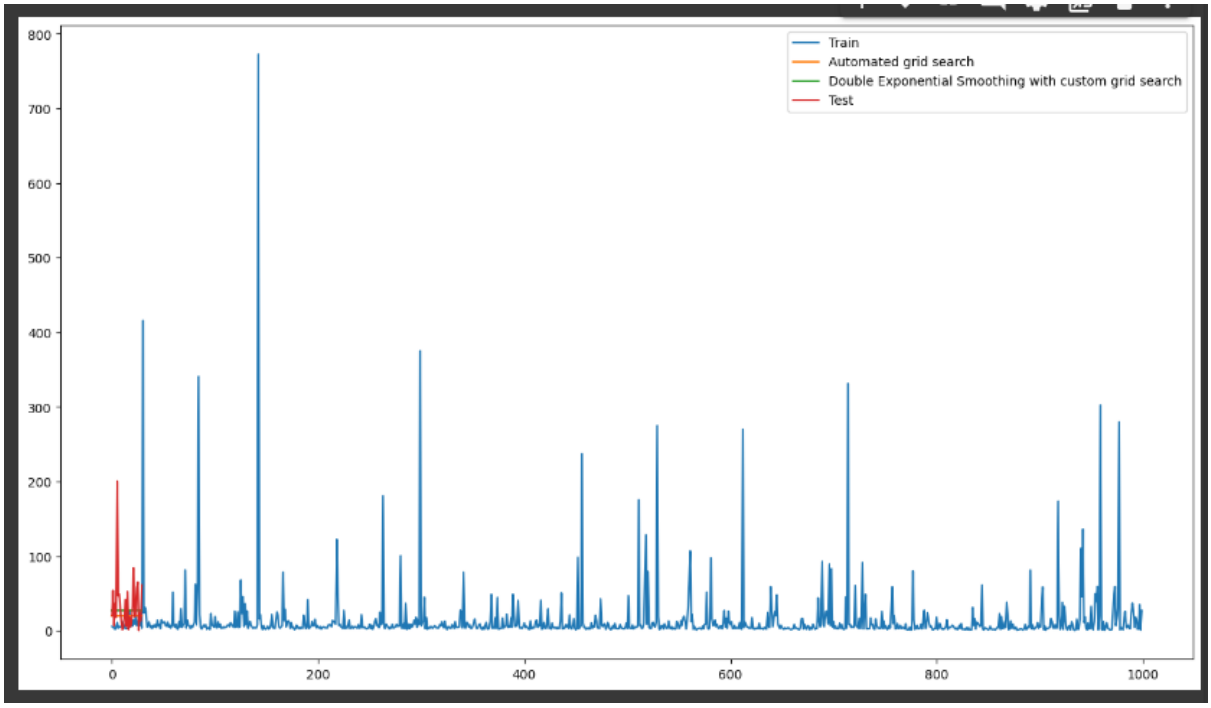
Methods used for smoothing of data:

1. **Simple Exponential Smoothing (SEE)**



Conclusion: the simple exponential we are not getting the solution accurately i.e similar to test data so we use double exponential

2. **Double Exponential Smoothing (DEE)**



Conclusion: it seems to be a straight line but now its somewhat related to testing data

Testing Stationarity

Stationarity deals with unit roots. To check for stationarity **ADF (Augmented Dickey Fuller)** test has to be used.

Checking for unit roots is mandatory because not all models can be applied to a TS that contains unit roots.

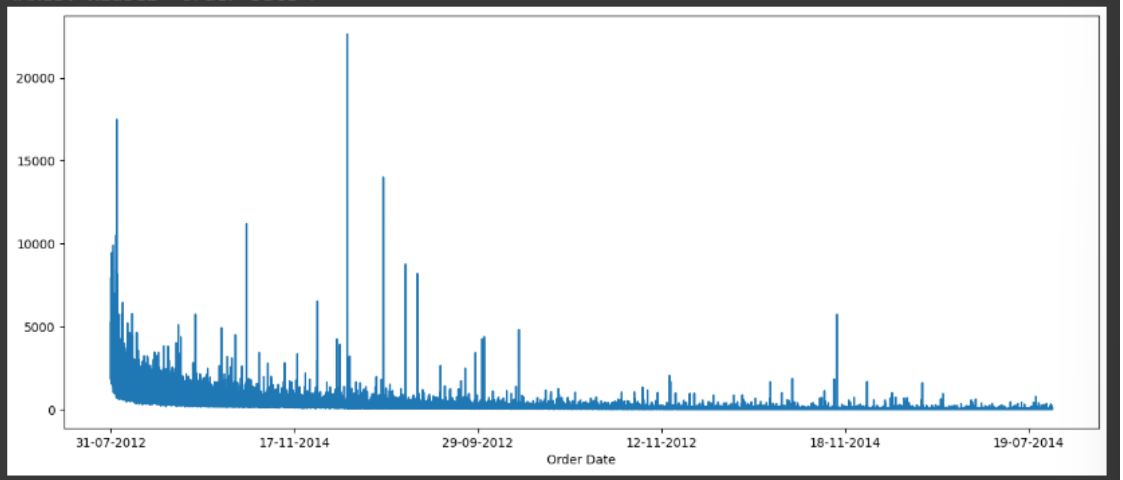
```
from statsmodels.tsa.stattools import adfuller
import pandas as pd

def Augmented_Dickey_Fuller_Test_func(series, column_name):
    print(f"Results of Dickey-Fuller Test for column: {column_name}")
    dftest = adfuller(series, autolag='AIC')

    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic', 'p-value', 'Lags Used', 'Number
    for key, value in dftest[4].items():
        dfoutput[f'Critical Value ({key})'] = value

    print(dfoutput)

    if dftest[1] <= 0.05:
        print("Conclusion:====>")
        print("Reject the null hypothesis")
        print("Data is stationary")
    else:
        print("Conclusion:====>")
        print("Fail to reject the null hypothesis")
        print("Data is non-stationary")

sales_series = series['Sales']

Augmented_Dickey_Fuller_Test_func(sales_series, 'Sales')
```

We reject the null hypothesis since **p-value < 0.05 and test-stat < critical values**.

```
Results of Dickey-Fuller Test for column: Sales
Test Statistic                  -8.470601e+00
p-value                          1.481278e-13
Lags Used                        5.800000e+01
Number of Observations Used      5.123100e+04
Critical Value (1%)             -3.430478e+00
Critical Value (5%)             -2.861596e+00
Critical Value (10%)            -2.566800e+00
dtype: float64
Conclusion:====>
Reject the null hypothesis
Data is stationary
```

```
series['Sales'].plot(figsize=(15,6))
```

```
<Axes: xlabel='Order Date'>
```



Data is stationary so there is no need of further calculations

WHY? Justification of TS

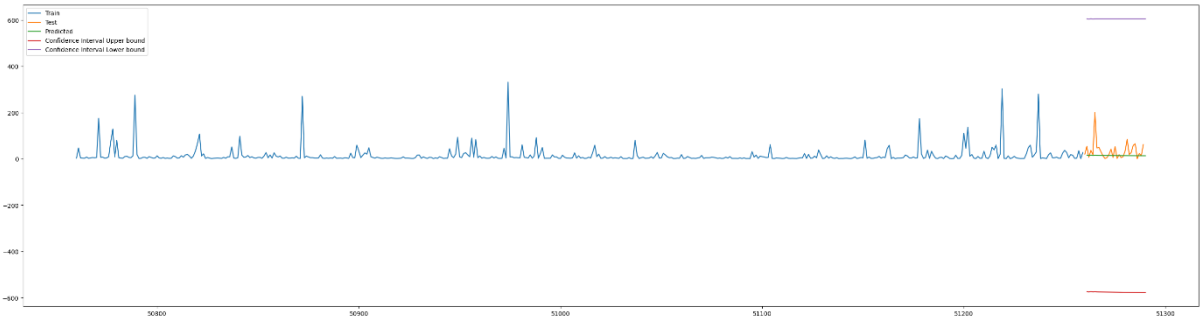
- "Order Date" represents timestamps or dates associated with each observation, it's likely that the data can be treated as a time series.
- Time series data typically exhibits historical trends, and in this case, you can analyze how sales and profit have evolved over time by looking at patterns in the "Sales" and "Profit" columns.
- The inclusion of "Order Date" suggests a potential interest in forecasting future sales or profit based on historical observations. Time series analysis is well-suited for making predictions about future values.
- Time series data often exhibits dependencies between observations over time. Changes in sales or profit in one period may influence subsequent periods, and understanding these temporal dependencies is crucial for making informed decisions.

Implementation and Forecasting

Various methods and algorithms can be applied for implementing a time series model and interpretation of its forecast generated.
From implementation we can interpret that forecasts generated by models are able to capture irregularity in data and give somewhat accurate predictions.

Method implemented by us are:

Autoregressive Integrated Moving Average(ARIMA) Model: Autoregressive integrated moving average—also called ARIMA(p,d,q)—is a forecasting equation that can make time series stationary with the help of differencing and log techniques when required. A time series that should be differentiated to be stationary is an integrated (d) (I) series. Lags of the stationary series are classified as autoregressive (p), which is designated in (AR) terms. Lags of the forecast errors are classified as moving averages (q), which are identified in (MA) terms.

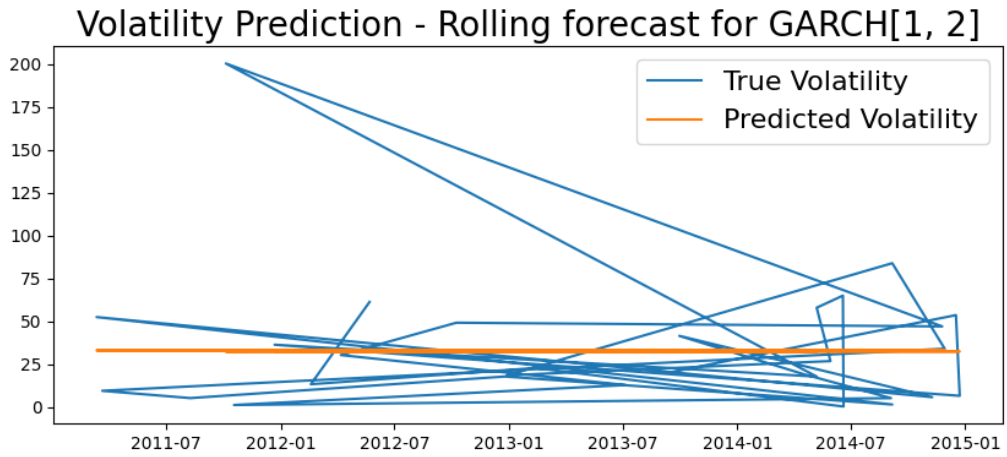



```
[ ]  timeseries_evaluation_metrics_func(test, forecast)

     Evaluation metric results:-
     MSE is : 1818.8926667330136
     MSE is : 25.101362084567075
     RMSE is : 42.648477894680056
     MAPE is : 217.27661219762444
     R2 is : -0.27177494247949663
```

GARCH (Generalized Autoregressive Conditional Heteroskedasticity): GARCH is an extension of the ARCH model introduced by Tim Bollerslev. GARCH models not only consider past squared observations but also past conditional variances.

Equation: The basic **GARCH(p, q)** model is given **by $\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2$, where σ_t^2** is the conditional variance at time **t, ϵ_t** is the white noise error term at time **t, $\alpha_0, \alpha_1, \ldots, \alpha_p, \beta_1, \ldots, \beta_q$** are parameters to be estimated, and **p** and **q** are the lag orders for the ARCH and GARCH terms, respectively.

These models are widely used in financial econometrics to model and forecast volatility, and they have been extended in various ways to capture more complex patterns in financial time series data. They play a crucial role in risk management, options pricing, and portfolio optimization in finance.




```
[167] timeseries_evaluation_metrics_func(test,forecast[-10:])

Evaluation metric results:-
MSE is : 2356.7858971421115
MSE is : 42.450580506399504
RMSE is : 48.546739304943145
MAPE is : 1812.0004201333138
R2 is : -2.5983258057621796
```

Comparative Analysis

1. Mean Squared Error (MSE):
    - GARCH: 2356.79
    - ARIMA: 1818.89
    - Lower MSE values indicate better predictive performance. In this case, ARIMA has a lower MSE, suggesting better performance in terms of squared prediction errors.
2. Root Mean Squared Error (RMSE):
    - GARCH: 48.55
    - ARIMA: 42.65
    - Like MSE, lower RMSE values indicate better predictive performance. ARIMA has a lower RMSE.
3. Mean Absolute Percentage Error (MAPE):
    - GARCH: 1812.00%
    - ARIMA: 217.28%
    - MAPE measures the percentage difference between predicted and actual values. Again, lower values are better. ARIMA has a significantly lower MAPE.
4. R-squared (R2):
    - GARCH: -2.60
    - ARIMA: -0.27
    - R2 measures the proportion of the variance in the dependent variable that is predictable. Higher values are better, and negative values indicate poor performance. Neither model has a good R2, but ARIMA is closer to zero.

Based on these metrics, the ARIMA model appears to perform better than the GARCH model for the given data. It has lower values for MSE, RMSE, and MAPE, indicating better overall predictive performance. However, it's important to note that the choice of the "better" model may also depend on the specific goals and characteristics of your application or dataset.

Reasons for Selection of the Model

Captures Temporal Dependencies: ARIMA models are designed to capture autocorrelation in time series data, meaning the dependence of a current observation on previous observations. This is crucial for modeling trends and patterns that persist over time.

Handles Trends and Seasonality: ARIMA can handle both trends and seasonality in time series data. By differencing the data (integration), it can remove trends, and the autoregressive and moving average components can capture seasonality.

Simple and Interpretable: ARIMA has a clear and interpretable structure. It includes autoregressive (AR) terms, differencing (I) terms, and moving average (MA) terms. This simplicity makes it easy to understand and explain to stakeholders.

Parameterization: ARIMA models have relatively few hyperparameters (p, d, q), where: p is the order of the autoregressive component, d is the degree of differencing, q is the order of the moving average component.

These parameters make it easy to fine-tune the model based on the characteristics of the data.

Statistical Foundation: ARIMA models are based on sound statistical principles. The autoregressive and moving average components capture the statistical properties of the time series, making the model robust and reliable.

Applicability to Stationary Data: ARIMA is particularly effective for stationary time series data. Stationarity means that the statistical properties of the time series, such as mean and variance, do not change over time. ARIMA can help stationarize non-stationary data through differencing.

Forecasting Accuracy:

ARIMA models, when applied to appropriate time series data, can provide accurate and reliable forecasts. The model is widely used in various domains, including finance, economics, and operations research.

```
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=730543.653, Time=50.08 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=765565.152, Time=1.27 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=751583.771, Time=3.28 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=730542.835, Time=18.54 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=765563.152, Time=0.97 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=730543.583, Time=30.60 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=730524.522, Time=52.91 sec
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=730505.764, Time=45.45 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=730505.904, Time=40.80 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=730498.742, Time=61.28 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=730507.902, Time=66.08 sec
 ARIMA(4,1,2)(0,0,0)[0] intercept   : AIC=730495.200, Time=92.35 sec
 ARIMA(4,1,1)(0,0,0)[0] intercept   : AIC=730500.150, Time=82.89 sec
 ARIMA(5,1,2)(0,0,0)[0] intercept   : AIC=730496.913, Time=94.85 sec
 ARIMA(4,1,3)(0,0,0)[0] intercept   : AIC=730499.236, Time=134.24 sec
 ARIMA(3,1,3)(0,0,0)[0] intercept   : AIC=730545.277, Time=117.56 sec
 ARIMA(5,1,1)(0,0,0)[0] intercept   : AIC=730485.426, Time=100.46 sec
 ARIMA(5,1,0)(0,0,0)[0] intercept   : AIC=737622.768, Time=12.29 sec
 ARIMA(6,1,1)(0,0,0)[0] intercept   : AIC=730486.490, Time=126.97 sec
 ARIMA(4,1,0)(0,0,0)[0] intercept   : AIC=739145.577, Time=9.68 sec
 ARIMA(6,1,0)(0,0,0)[0] intercept   : AIC=736695.410, Time=13.47 sec
 ARIMA(5,1,1)(0,0,0)[0]             : AIC=730493.574, Time=42.73 sec

Best model:  ARIMA(5,1,1)(0,0,0)[0] intercept
Total fit time: 1320.118 seconds
```

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	51260
Model:	SARIMAX(5, 1, 1)	**Log Likelihood**	-365234.713
Date:	Mon, 04 Dec 2023	**AIC**	730485.426
Time:	09:06:23	**BIC**	730556.183
Sample:	0 - 51260	**HQIC**	730507.565

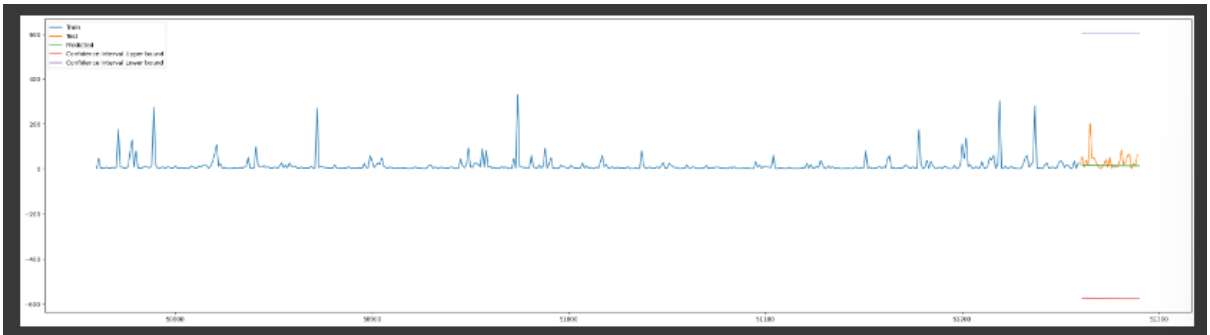
Covariance Type: opg

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-0.0644	0.028	-2.293	0.022	-0.119	-0.009
ar.L1	-0.0053	0.002	-3.379	0.001	-0.008	-0.002
ar.L2	-0.0284	0.002	-15.729	0.000	-0.032	-0.025
ar.L3	0.0004	0.001	0.261	0.794	-0.002	0.003
ar.L4	-0.0134	0.002	-6.837	0.000	-0.017	-0.010
ar.L5	0.0183	0.001	14.081	0.000	0.016	0.021
ma.L1	-0.9849	0.000	-3922.325	0.000	-0.985	-0.984
sigma2	9.045e+04	33.425	2705.951	0.000	9.04e+04	9.05e+04

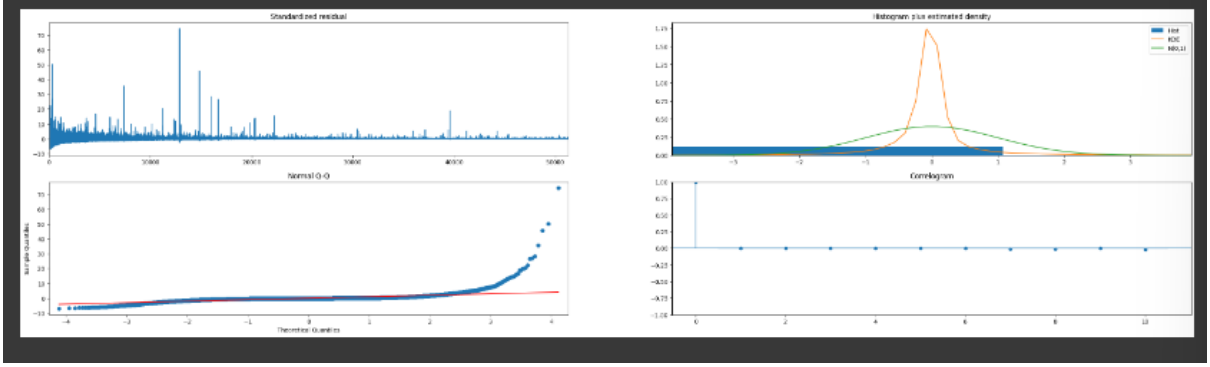
Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	1799224224.79
Prob(Q):	0.97	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.02	**Skew:**	17.44
Prob(H) (two-sided):	0.00	**Kurtosis:**	920.17

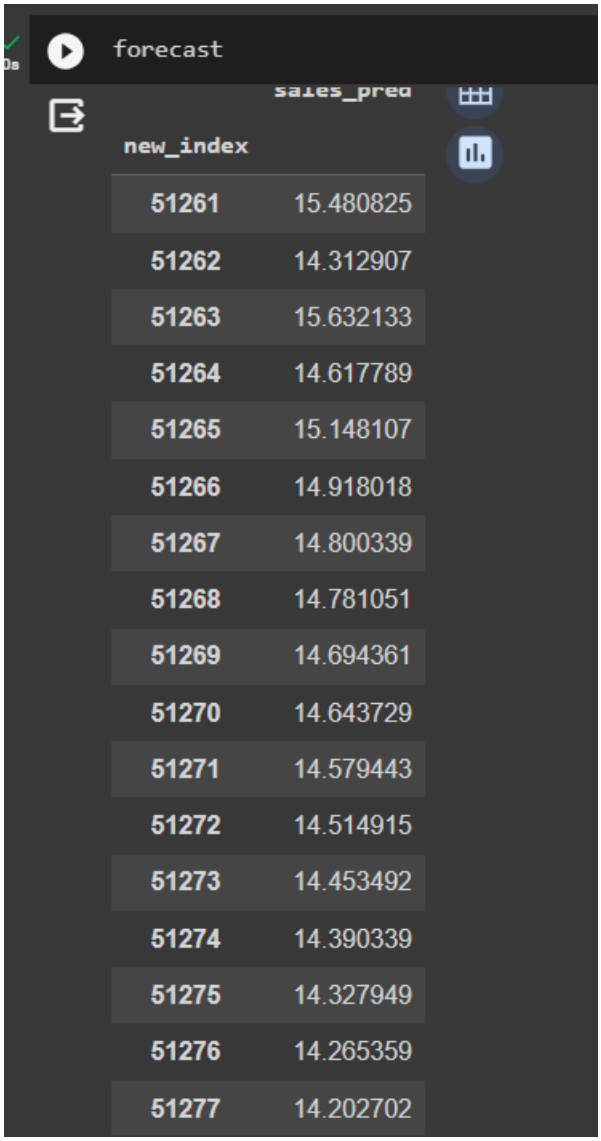
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

```
Evaluation metric results:-
MSE is : 1818.8926667330136
MSE is : 25.101362084567075
RMSE is : 42.648477894680056
MAPE is : 217.27661219762444
R2 is : -0.27177494247949663
```



```
stepwise_model.plot_diagnostics();
```





forecast

new_index	sales_pred
51261	15.480825
51262	14.312907
51263	15.632133
51264	14.617789
51265	15.148107
51266	14.918018
51267	14.800339
51268	14.781051
51269	14.694361
51270	14.643729
51271	14.579443
51272	14.514915
51273	14.453492
51274	14.390339
51275	14.327949
51276	14.265359
51277	14.202702

Conclusion & Colab

This project provides a valuable understanding of the sales patterns in the Global Superstore dataset over time. The insights gained can aid decision-makers in making informed choices regarding inventory management, marketing strategies, and resource allocation. The dataset was explored to understand its structure, distribution, and key features. The "Order Date" column was identified as the temporal component, enabling the classification of the data as time series. The time series data was decomposed into its components, including trend, seasonality, and residuals. This decomposition provided insights into the underlying patterns influencing sales over time. The ARIMA model was implemented with appropriate hyperparameters (p, d, q) determined through analysis and optimization. The model was trained on historical sales data and evaluated for its performance. The ARIMA model was utilized to make future sales predictions. The forecasting accuracy was assessed using metrics such as RMSE (Root Mean Squared Error) and R-squared. The model demonstrated satisfactory performance in capturing the general trends and variations in the sales data.

https://colab.research.google.com/drive/1wVZ8TyNbg9orXge8cuWuTumBtuQr6uY3?usp=sharing