ASSIGNMENT: Retail Gaint sales and Quality forecasting

SUBMITTED BY : Nidhi Sharma

Overview

Problem Statement:

Global Mart is having worldwide operations.

Takes orders and delivers across the globe and deals with all the major product categories - consumer, corporate & home office.

The store caters to 7 different market segments and in 3 major categories.

Our Job

Wants to forecast the sales and the demand for the next 6 months to help manage the revenue and inventory accordingly.

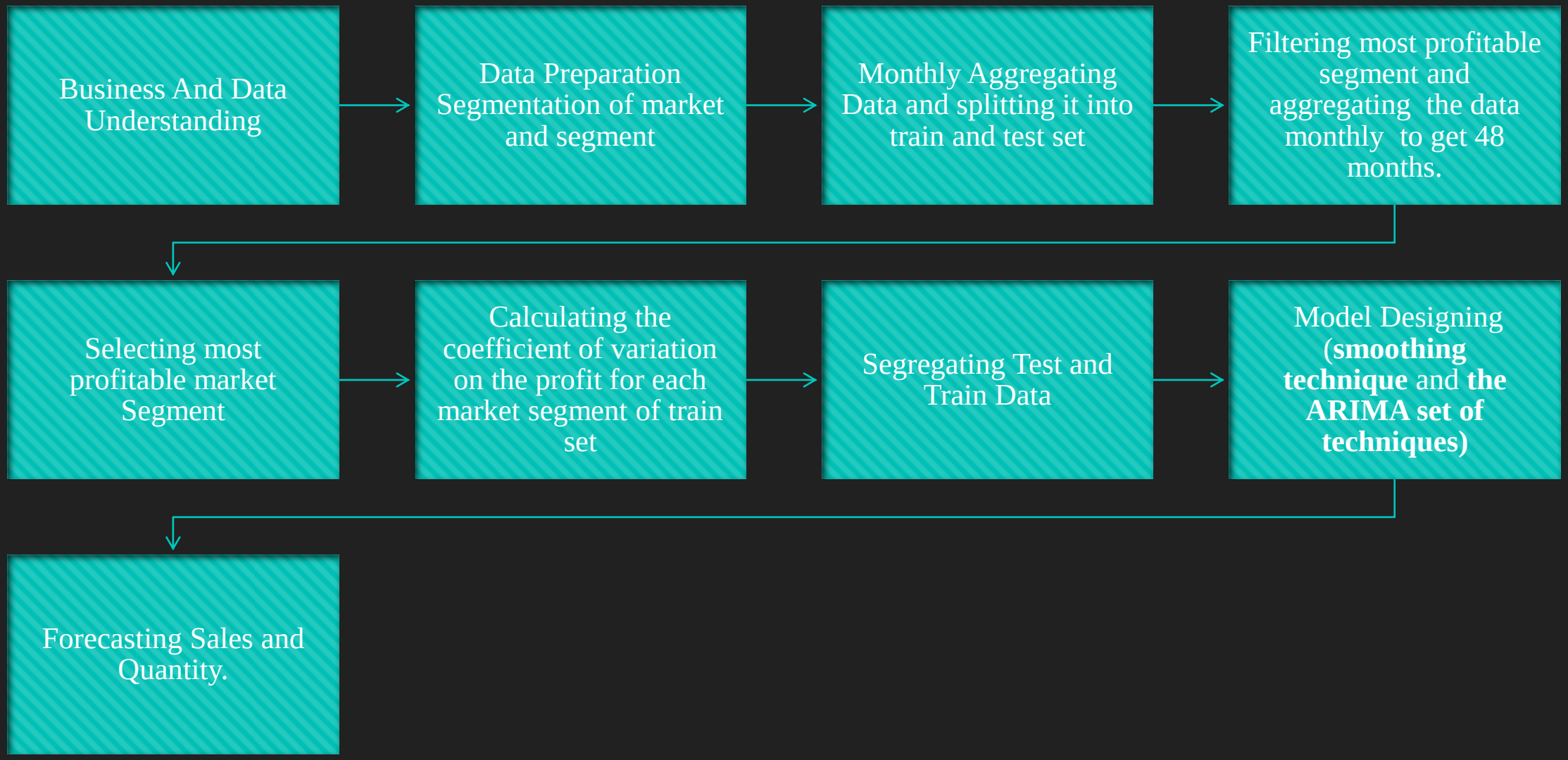
Find out the most profitable and consistent segments and forecast the sales and demand for these segments.

specify the models whose MAPE values is less

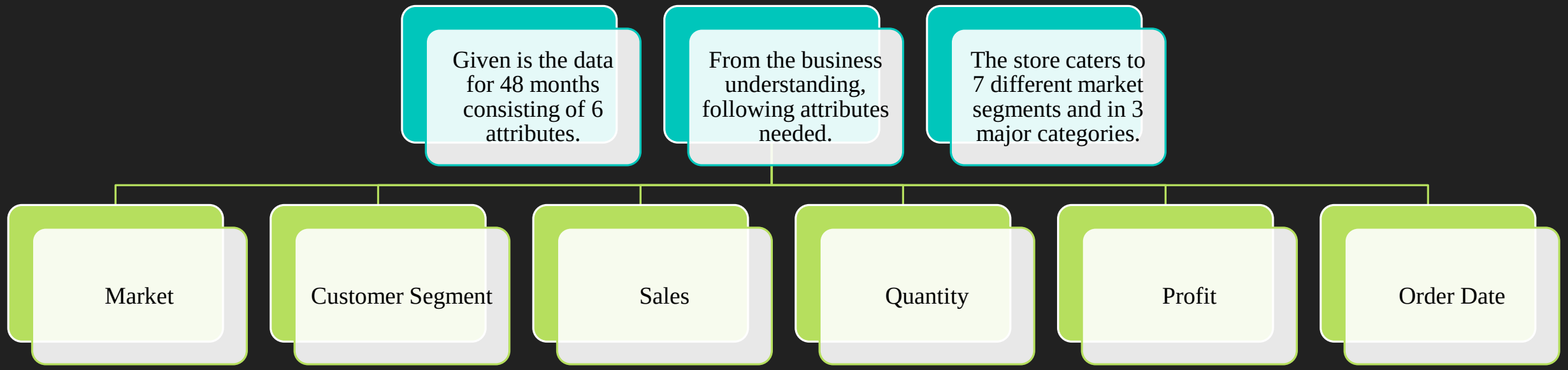
Define the problem Statement

Global Mart is an online supergiant store that has worldwide operations. This store takes orders and delivers across the globe and deals with all the major product categories — consumer, corporate & home office. As a sales manager for this store, you have to **forecast the sales and quantity of the products for the next 6 months**, so that you have an estimate of the demand and the sales of the products in those months and can plan your inventory and business processes accordingly.

Flow chart


Business And Data Understanding
Data Preparation Segmentation of market and segment
Monthly Aggregating Data and splitting it into train and test set
Filtering most profitable segment and aggregating the data monthly to get 48 months.
Selecting most profitable market Segment
Calculating the coefficient of variation on the profit for each market segment of train set
Segregating Test and Train Data
Model Designing (smoothing technique and the ARIMA set of techniques)
Forecasting Sales and Quantity.

Flow chart of Data Understanding



Collect the Data(Time Series data)

Collect the Data(Time Series data)

1. Import required packages and Import and read the data
2. Convert the "Order Date" column into date- time format

Import time series data: Global2

[4]:
```
data1 = pd.read_csv('Global2.csv')
data1.head()
```

[4]:

	Order Date	Segment	Market	Sales	Quantity	Profit
0	31-07-2012	Consumer	US	2309.650	7	762.1845
1	05-02-2013	Corporate	APAC	3709.395	9	-288.7650
2	17-10-2013	Consumer	APAC	5175.171	9	919.9710
3	28-01-2013	Home Office	EU	2892.510	5	-96.5400
4	05-11-2013	Consumer	Africa	2832.960	8	311.5200

- convert the "Order Date" column into date- time format

[348]:
```
data1['Order Date']= pd.to_datetime(data1['Order Date'], format='%d-%m-%Y')
```

[349]:
```
data1.head()
```

[349]:

	Order Date	Segment	Market	Sales	Quantity	Profit
0	2012-07-31	Consumer	US	2309.650	7	762.1845
1	2013-02-05	Corporate	APAC	3709.395	9	-288.7650
2	2013-10-17	Consumer	APAC	5175.171	9	919.9710
3	2013-01-28	Home Office	EU	2892.510	5	-96.5400
4	2013-11-05	Consumer	Africa	2832.960	8	311.5200

1. Inspect the dataframe

[350]:
```
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Order Date  51290 non-null  datetime64[ns]
 1   Segment     51290 non-null  object
 2   Market      51290 non-null  object
 3   Sales       51290 non-null  float64
 4   Quantity    51290 non-null  int64
 5   Profit      51290 non-null  float64
dtypes: datetime64[ns](1), float64(2), int64(1), object(2)
memory usage: 2.3+ MB
```

[351]:
```
data1.describe()
```

[351]:

	Sales	Quantity	Profit
count	51290.000000	51290.000000	51290.000000
mean	246.490581	3.476545	28.610982
std	487.565361	2.278766	174.340972
min	0.444000	1.000000	-6599.978000
25%	30.758625	2.000000	0.000000
50%	85.053000	3.000000	9.240000
75%	251.053200	5.000000	36.810000
max	22638.480000	14.000000	8399.976000

[352]:
```
data1.shape
```

[352]:
```
(51290, 6)
```

[353]:
```
data1.head()
```

[353]:

	Order Date	Segment	Market	Sales	Quantity	Profit
0	2012-07-31	Consumer	US	2309.650	7	762.1845
1	2013-02-05	Corporate	APAC	3709.395	9	-288.7650
2	2013-10-17	Consumer	APAC	5175.171	9	919.9710
3	2013-01-28	Home Office	EU	2892.510	5	-96.5400
4	2013-11-05	Consumer	Africa	2832.960	8	311.5200

[354]:
```
# Write your code for column-wise null percentages here
print(round(100*(data1.isnull().sum()/len(data1.index)), 2))
```

```
Order Date    0.0
Segment       0.0
Market        0.0
Sales         0.0
Quantity      0.0
Profit        0.0
dtype: float64
```

Data Preparation

Cleaning data

1. Find the 21 unique Market Segments
2. Creating a Pivot table for getting the month wise total sum of profit by each market segment
3. Calculate the CoV for each market segment of train data.
4. Find the most profitable market segment

1.

```
[355]: data1['Market_Segment']= data1[['Market', 'Segment']].apply(lambda x: '_'.join(x), axis=1)
       data1
```

[355]:

	Order Date	Segment	Market	Sales	Quantity	Profit	Market_Segment
0	2012-07-31	Consumer	US	2309.650	7	762.1845	US_Consumer
1	2013-02-05	Corporate	APAC	3709.395	9	-288.7650	APAC_Corporate
2	2013-10-17	Consumer	APAC	5175.171	9	919.9710	APAC_Consumer
3	2013-01-28	Home Office	EU	2892.510	5	-96.5400	EU_Home Office
4	2013-11-05	Consumer	Africa	2832.960	8	311.5200	Africa_Consumer
...	...	...	...	...	...	...	...
51285	2014-06-19	Corporate	APAC	65.100	5	4.5000	APAC_Corporate
51286	2014-06-20	Consumer	US	0.444	1	-1.1100	US_Consumer
51287	2013-12-02	Home Office	US	22.920	3	11.2308	US_Home Office
51288	2012-02-18	Home Office	LATAM	13.440	2	2.4000	LATAM_Home Office
51289	2012-05-22	Consumer	LATAM	61.380	3	1.8000	LATAM_Consumer

51290 rows × 7 columns

2.

Creating a Pivot table for getting the month wise total sum of profit by each market segment

```
[15]: df= pd.pivot_table(data=data1, values='Profit', index=pd.Grouper(key='Order Date', freq='1M'), columns='Market_Segment', aggfunc='sum')
      df.head()
```

[15]:

Market_Segment	APAC_Consumer	APAC_Corporate	APAC_Home Office	Africa_Consumer	Africa_Corporate	Africa_Home Office	Canada_Consumer	Canada_Corporate	Canada_Home Office
Order Date									
2011-01-31	991.2825	11.5998	86.4423	475.683	219.096	856.710	3.12	5.7	NaN
2011-02-28	1338.8688	4358.8254	-417.4128	1441.926	-490.551	820.302	23.31	NaN	87.99
2011-03-31	3747.1632	1213.3386	923.7492	322.140	-586.716	67.320	335.55	NaN	84.03
2011-04-30	3846.4746	71.0265	657.1080	292.122	776.691	500.136	55.08	NaN	NaN
2011-05-31	3639.9423	2534.1672	-272.1717	110.004	241.338	34.926	77.97	NaN	NaN

5 rows × 21 columns

3.

```
Cov_values=mean_std.iloc[1]/mean_std.iloc[0]
Cov_values
```

```
APAC_Consumer          0.603633
APAC_Corporate         0.740799
APAC_Home Office       1.061530
Africa_Consumer        1.446661
Africa_Corporate       1.685008
Africa_Home Office     2.013987
Canada_Consumer        1.497032
Canada_Corporate       1.219189
Canada_Home Office     2.245148
EMEA_Consumer          2.749927
EMEA_Corporate         6.861820
EMEA_Home Office       6.140222
EU_Consumer            0.655334
EU_Corporate           0.697702
EU_Home Office         1.128192
LATAM_Consumer         0.688935
LATAM_Corporate        0.890930
LATAM_Home Office      1.359984
US_Consumer            1.108571
US_Corporate           1.039660
US_Home Office         1.231887
dtype: float64
```

4. Most Profitable Marget-Segment (CoV value is least) : APAC_Consumer

Take out the Sale and Quantity of APAC_Consumer

1. Take out the sales data of APAC_Consumer
2. Checknull values

```
[19]: APAC_Consumer= data1[(data1.Market_Segment== 'APAC_Consumer')]
      APAC_Consumer= APAC_Consumer.sort_values(by='Order Date')
      APAC_Consumer.head()
```

[19]:

	Order Date	Segment	Market	Sales	Quantity	Profit	Market_Segment
42055	2011-01-01	Consumer	APAC	55.242	2	15.342	APAC_Consumer
22951	2011-01-01	Consumer	APAC	120.366	3	36.036	APAC_Consumer
31869	2011-01-01	Consumer	APAC	113.670	5	37.770	APAC_Consumer
2709	2011-01-03	Consumer	APAC	912.456	4	-319.464	APAC_Consumer
47553	2011-01-03	Consumer	APAC	6.006	1	0.546	APAC_Consumer

```
[20]: APAC_Consumer= APAC_Consumer.drop(['Segment','Market'],axis=1)
      APAC_Consumer.head()
```

[20]:

	Order Date	Sales	Quantity	Profit	Market_Segment
42055	2011-01-01	55.242	2	15.342	APAC_Consumer
22951	2011-01-01	120.366	3	36.036	APAC_Consumer
31869	2011-01-01	113.670	5	37.770	APAC_Consumer
2709	2011-01-03	912.456	4	-319.464	APAC_Consumer
47553	2011-01-03	6.006	1	0.546	APAC_Consumer

```
[22]: monthly_data_APAC= APAC_Consumer.groupby(pd.Grouper(key='Order Date', freq='1
      Sales_forecast= monthly_data_APAC[['Sales']]
      Sales_forecast.head()
```

[22]:

Order Date	Sales
2011-01-31	15711.7125
2011-02-28	12910.8588
2011-03-31	19472.5632
2011-04-30	15440.3046
2011-05-31	24348.9723

```
:  Sales_forecast.isnull().sum()
:  Sales    0
   dtype: int64
:  Sales_forecast.shape
:  (48, 1)
```

1. Take out the Quantity data of APAC_Consumer
2. Checknull values

```
[22]: APAC_Consumer= data1[(data1.Market_Segment== 'APAC_Consumer')]
      APAC_Consumer= APAC_Consumer.sort_values(by='Order Date')
      APAC_Consumer.head()
```

[22]:

	Order Date	Segment	Market	Sales	Quantity	Profit	Market_Segment
42055	2011-01-01	Consumer	APAC	55.242	2	15.342	APAC_Consumer
22951	2011-01-01	Consumer	APAC	120.366	3	36.036	APAC_Consumer
31869	2011-01-01	Consumer	APAC	113.670	5	37.770	APAC_Consumer
2709	2011-01-03	Consumer	APAC	912.456	4	-319.464	APAC_Consumer
47553	2011-01-03	Consumer	APAC	6.006	1	0.546	APAC_Consumer

```
[23]: APAC_Consumer= APAC_Consumer.drop(['Segment','Market'],axis=1)
      APAC_Consumer.head()
```

[23]:

	Order Date	Sales	Quantity	Profit	Market_Segment
42055	2011-01-01	55.242	2	15.342	APAC_Consumer
22951	2011-01-01	120.366	3	36.036	APAC_Consumer
31869	2011-01-01	113.670	5	37.770	APAC_Consumer
2709	2011-01-03	912.456	4	-319.464	APAC_Consumer
47553	2011-01-03	6.006	1	0.546	APAC_Consumer

```
[25]: monthly_data_APAC= APAC_Consumer.groupby(pd.Grouper(key='Order Date', freq='1M')).sum()
      Quantity_forecast= monthly_data_APAC[['Quantity']]
      Quantity_forecast.head()
```

[25]:

Order Date	Quantity
2011-01-31	214
2011-02-28	151
2011-03-31	283
2011-04-30	148
2011-05-31	244

```
[27]:  Quantity_forecast.isnull().sum()
[27]:  Quantity    0
       dtype: int64
[28]:  Quantity_forecast.shape
[28]:  (48, 1)
```

Analyze the Data

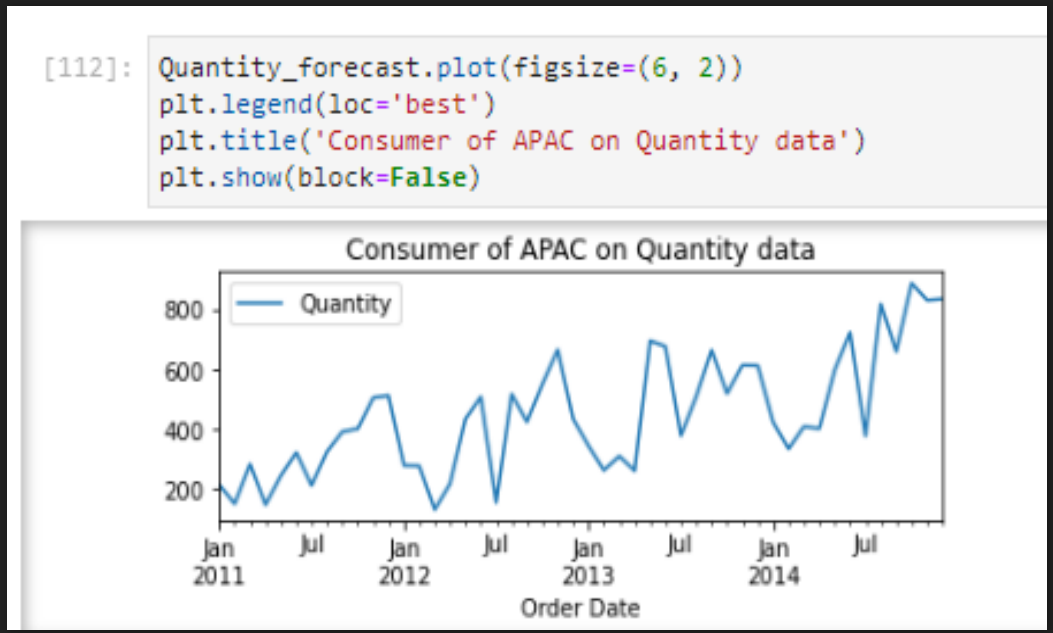
Steps to follow for Analyzing the Data

- Plot time series data "Quantity of APAC for Consumer segment“
- Outlier detection and treatment
- Time series Decomposition

Plot time series data “Quantity and Sales of APAC for Consumer segment“

Plot time series data "Quantity of APAC for Consumer segment“

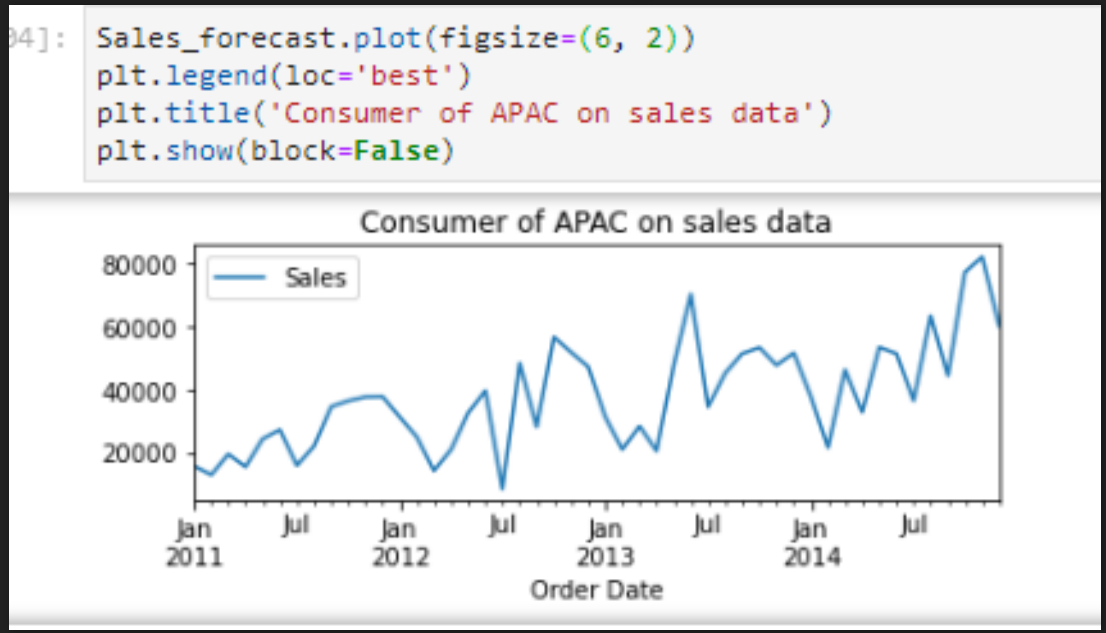
```
[112]: Quantity_forecast.plot(figsize=(6, 2))
       plt.legend(loc='best')
       plt.title('Consumer of APAC on Quantity data')
       plt.show(block=False)
```



The graph shows a upward trend but seasonality is not clear , need more insight into it.

Plot time series data “Sales of APAC for Consumer segment“

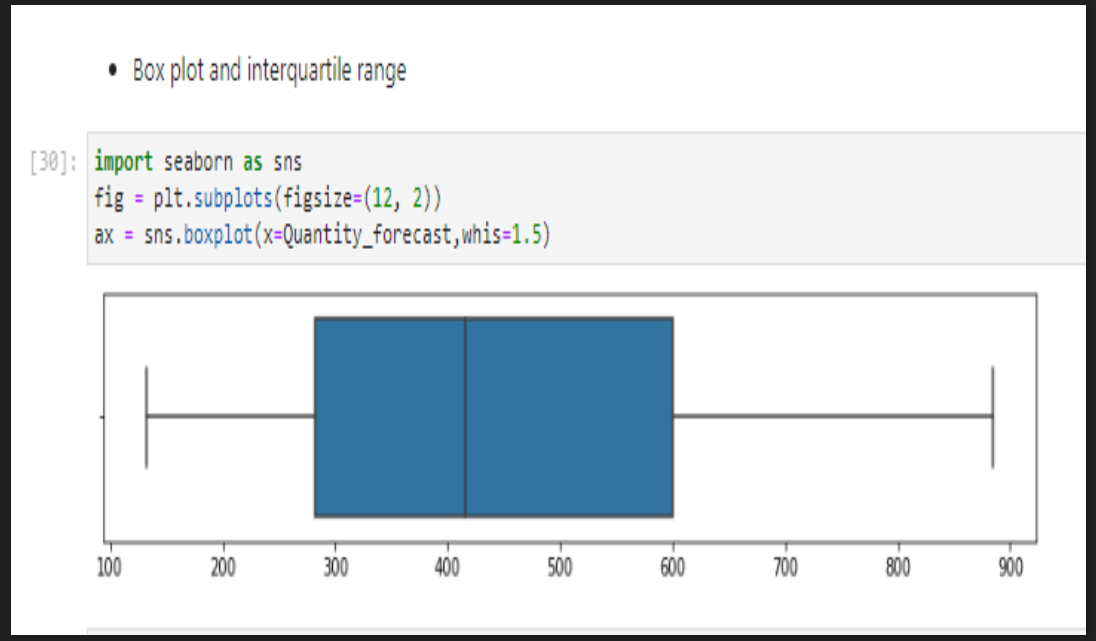
```
[114]: Sales_forecast.plot(figsize=(6, 2))
       plt.legend(loc='best')
       plt.title('Consumer of APAC on sales data')
       plt.show(block=False)
```



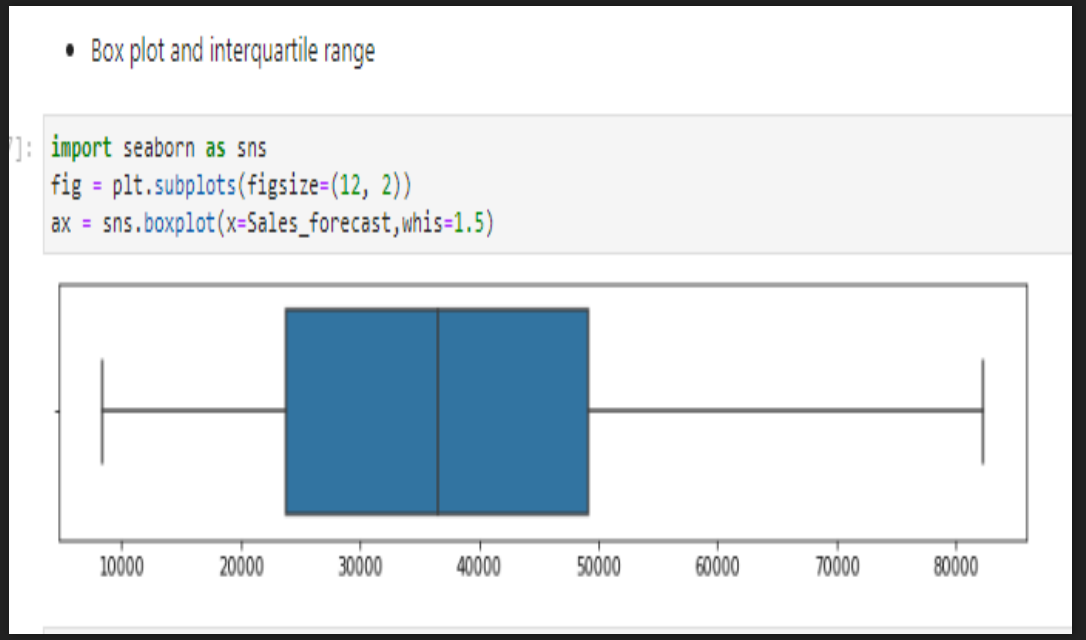
The graph shows a upward trend but seasonality is not clear , need more insight into it.

Outlier detection

Boxplot for Quantity_Forecast



Boxplot for Sales_Forecast

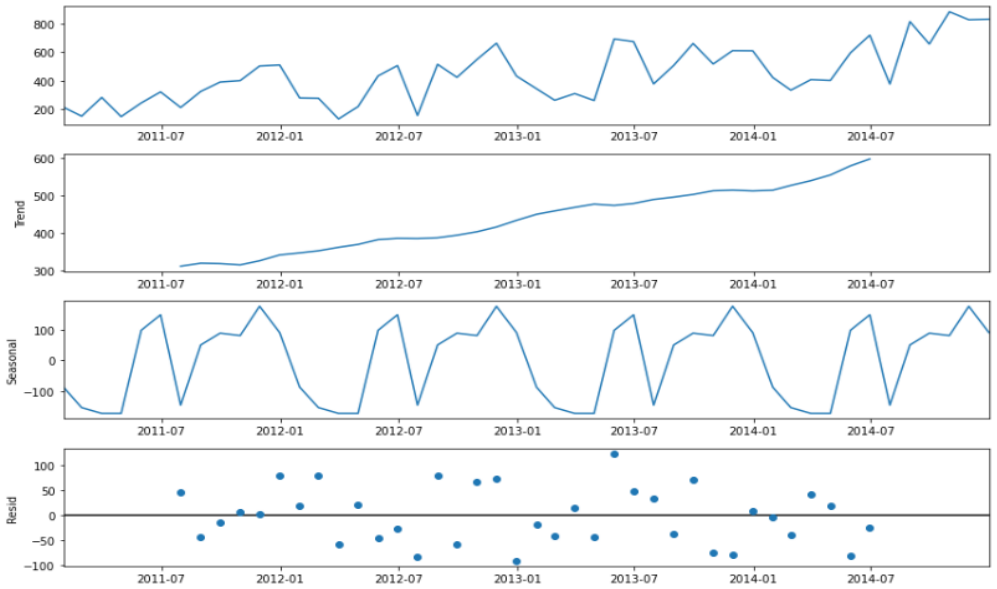


Whis : Proportion of the IQR past the low and high quartiles to extend the plot whiskers. Points outside this range will be identified as outliers

Time series Decomposition

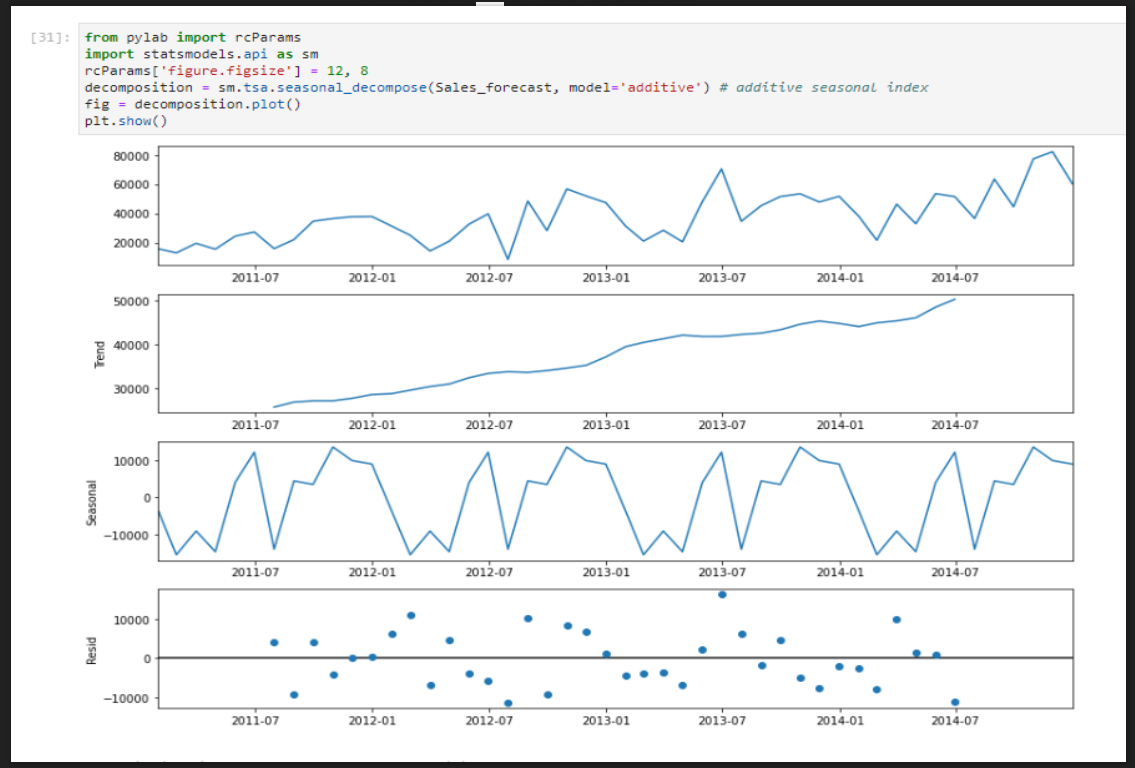
Additive seasonal decomposition of Quantity_Forecast

```
[34]: from pylab import rcParams
import statsmodels.api as sm
rcParams['figure.figsize'] = 12, 8
decomposition = sm.tsa.seasonal_decompose(Quantity_forecast, model='additive') # additive seasonal index
fig = decomposition.plot()
plt.show()
```



Additive seasonal decomposition of Sales_Forecast

```
[31]: from pylab import rcParams
import statsmodels.api as sm
rcParams['figure.figsize'] = 12, 8
decomposition = sm.tsa.seasonal_decompose(Sales_forecast, model='additive') # additive seasonal index
fig = decomposition.plot()
plt.show()
```

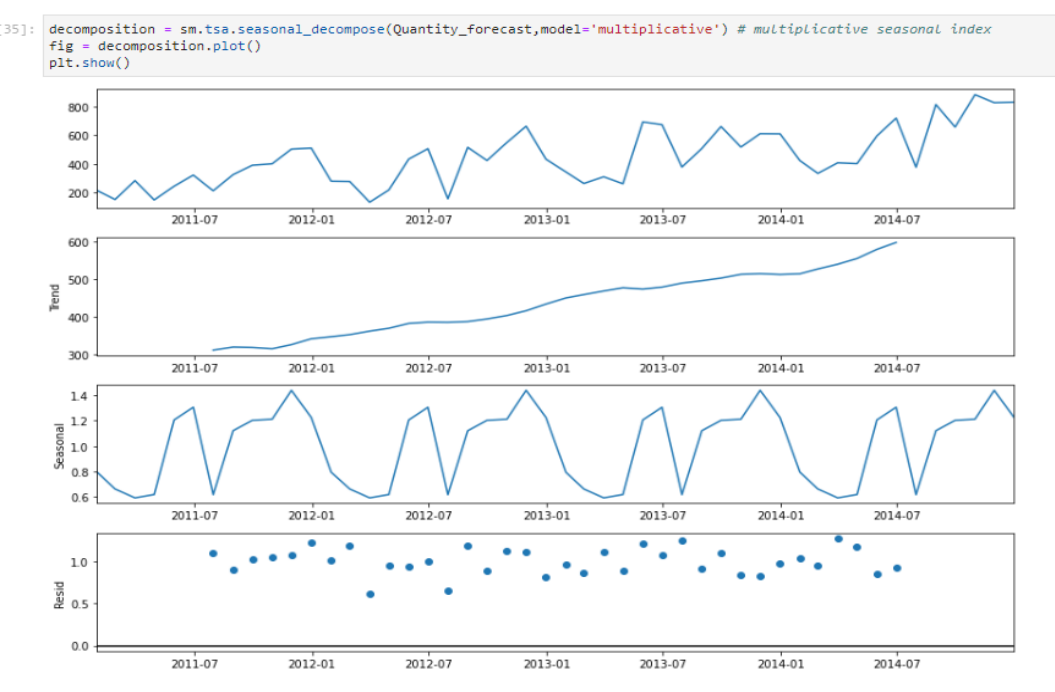


In this additive seasonal decomposition upward Trend and Seasonality component is clearly shown

Time series Decomposition

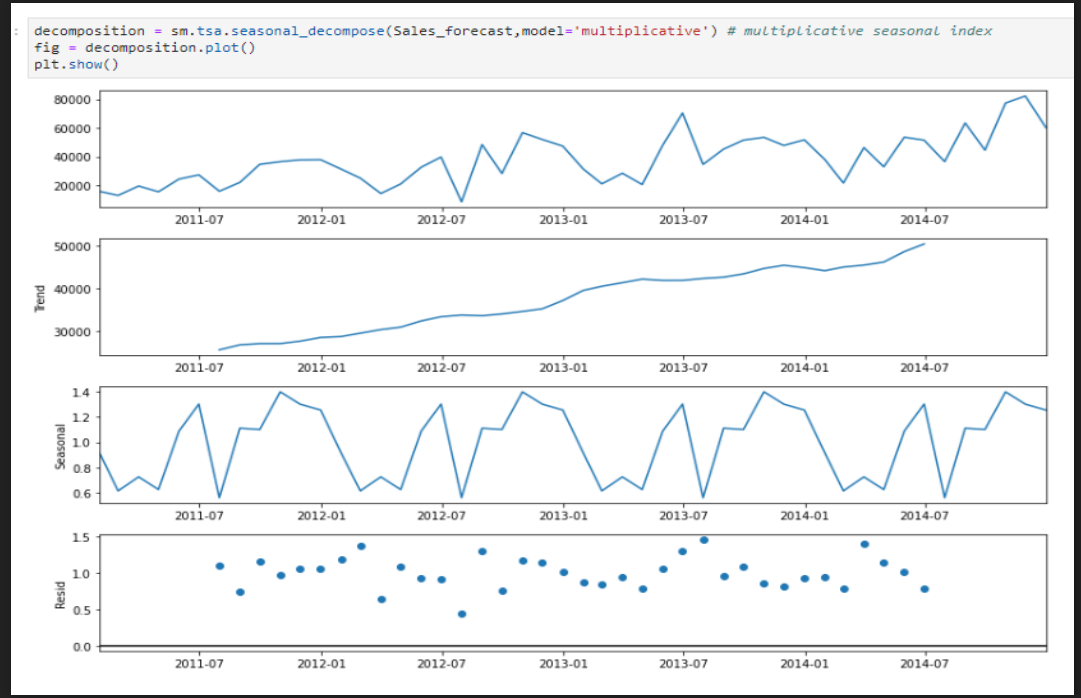
Multiplicative seasonal decomposition of Quantity_Forecast

```
decomposition = sm.tsa.seasonal_decompose(Quantity_forecast,model='multiplicative') # multiplicative seasonal index
fig = decomposition.plot()
plt.show()
```



Multiplicative seasonal decomposition of Sales_Forecast

```
decomposition = sm.tsa.seasonal_decompose(Sales_forecast,model='multiplicative') # multiplicative seasonal index
fig = decomposition.plot()
plt.show()
```



Multiplicative Seasonal Decomposition - the individual components can be multiplied to get the time-series data

Algorithm for Forecasting based on 2 different techniques of models

Data Preparation (collect, analyse)

- Created 21 Market-Segment buckets.
- Aggregated buckets by Sales, Profit, Quantity.
- Split the data into train(42) and test(6).
- Calculated Coefficient of Variation(CoV) for train data(42).
- Selected the most profitable market_segment (APAC Consumer) based on least CoV value.
- Filter the data with most profitable market_segments
- created the monthly aggregated data on 'Order Date' & 'Sales'.and 'Order Date' & 'Quantity'

Modeling using smoothing techniques

- From the monthly aggregated data for 'Order Date' and 'Sales' & 'Order Date' and 'Quantity of APAC Consumer.
- Split the data into train & test sets of size 42 & 6 months.
- Build the model by using 3 smoothing techniques(Simple exponential smoothing,Holt's exponential smoothing,Holt-Winters' exponential smoothing (both additive and multiplicative techniques))

Modeling using ARIMA set of techniques

- Converted data into stationary by using box-cox transformation and differencing method.
- Builded the model by using ARIMA set of techniques:-
- 1) AR model
- 2) MA model
- 3) ARMA model
- 4) ARIMA model
- 5) SARIMA model
- **ARIMAX and SARIMAX model (can not be applied, as there is no exogenous variable in the data).**

Forecasting

- Models able to predict the forecast future 6 months sales & quantity from January 2011 to December 2014 closer to the actual values are :
- Holt-Winters' exponential smoothing and SARIMA model

Modeling based on two techniques

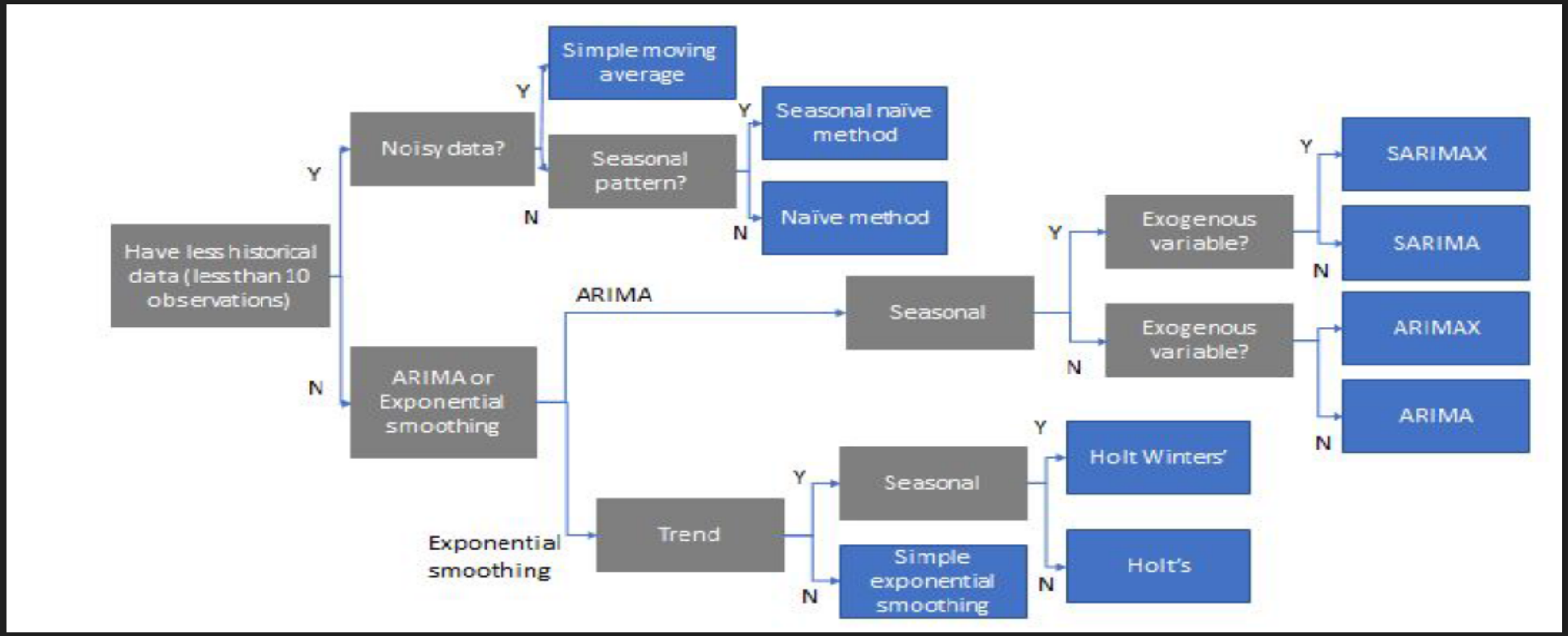
1. Smoothing Techniques
2. Autoregressive models

Smoothing Techniques

Smoothing , data removes random variation and shows trends and cyclic components.

Inherent in the collection of data taken over time is some form of random variation. There exist methods for reducing of canceling the effect due to random variation. An often-used **technique** in industry is "**smoothing**".

Flow Chart


Have less historical data (less than 10 observations)
Y
Noisy data?
Y
Simple moving average
N
Seasonal pattern?
Y
Seasonal naïve method
N
Naïve method
N
ARIMA or Exponential smoothing
ARIMA
Seasonal
Y
Exogenous variable?
Y
SARIMAX
N
SARIMA
N
Exogenous variable?
Y
ARIMAX
N
ARIMA
Exponential smoothing
Trend
Y
Seasonal
Y
Holt Winters'
N
Holt's
N
Simple exponential smoothing

Smoothing Techniques

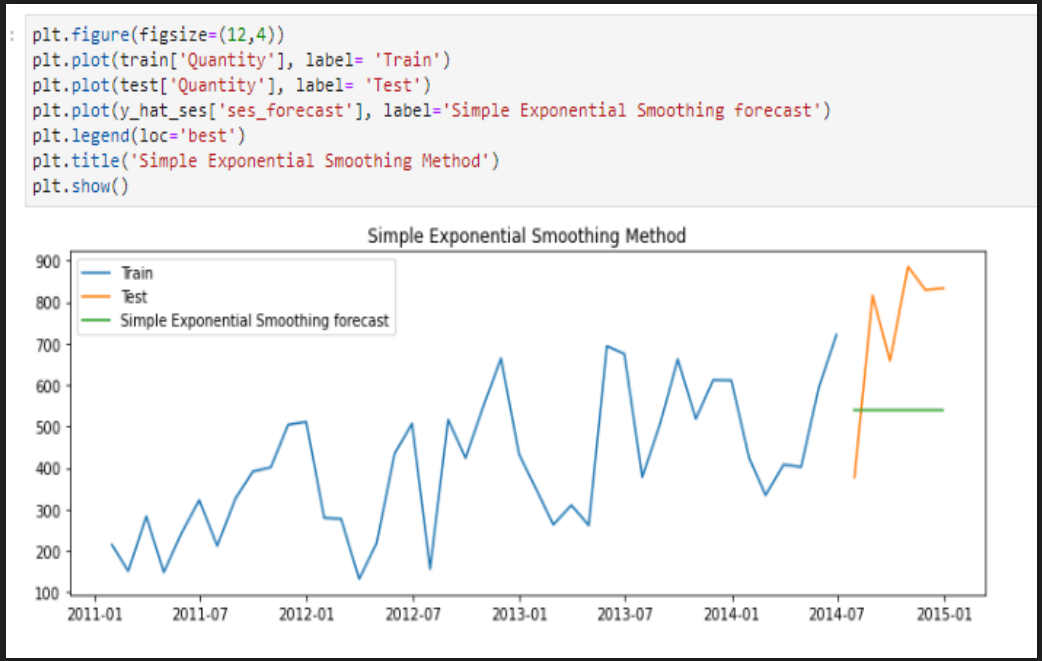
Work on Train data size= 42 and Test data size= 6 and compare the Quantity_forecast and Sales_forecast results

1. Simple exponential smoothing
2. Holt's Exponential Smoothing Method with Trend
3. Holt Winters's Additive Exponential Smoothing Method with Trend and Seasonality
4. Holt Winters's Multiplicative Exponential Smoothing Method with Trend and Seasonality

Simple Exponential Smoothing

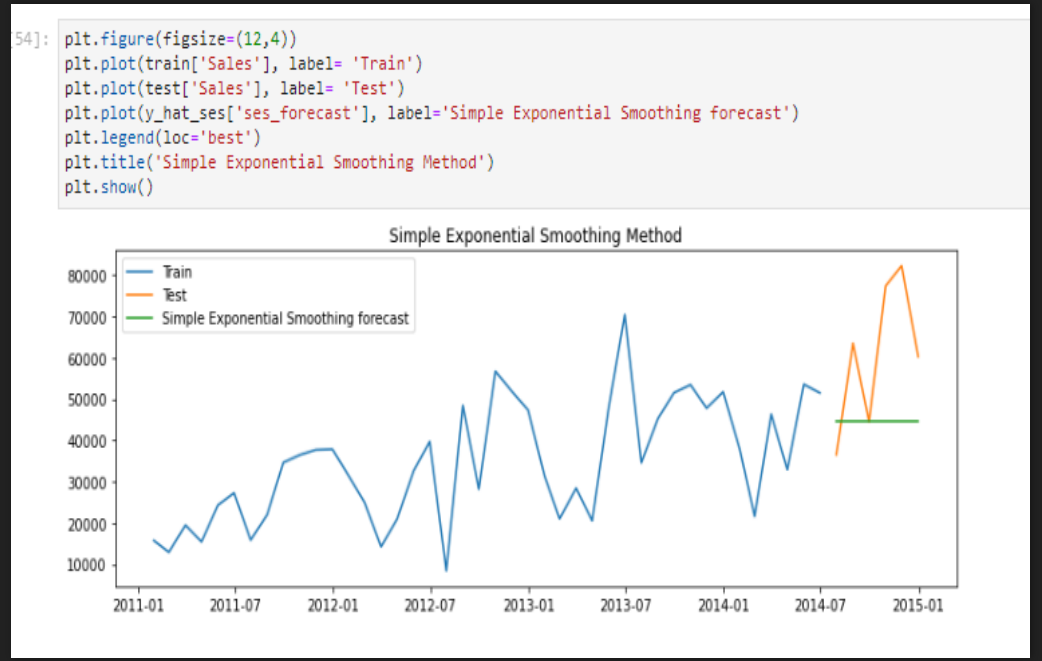
On Quantity_forecast

```
plt.figure(figsize=(12,4))
plt.plot(train['Quantity'], label= 'Train')
plt.plot(test['Quantity'], label= 'Test')
plt.plot(y_hat_ses['ses_forecast'], label='Simple Exponential Smoothing forecast')
plt.legend(loc='best')
plt.title('Simple Exponential Smoothing Method')
plt.show()
```



On Sales_forecast

```
[54]: plt.figure(figsize=(12,4))
plt.plot(train['Sales'], label= 'Train')
plt.plot(test['Sales'], label= 'Test')
plt.plot(y_hat_ses['ses_forecast'], label='Simple Exponential Smoothing forecast')
plt.legend(loc='best')
plt.title('Simple Exponential Smoothing Method')
plt.show()
```



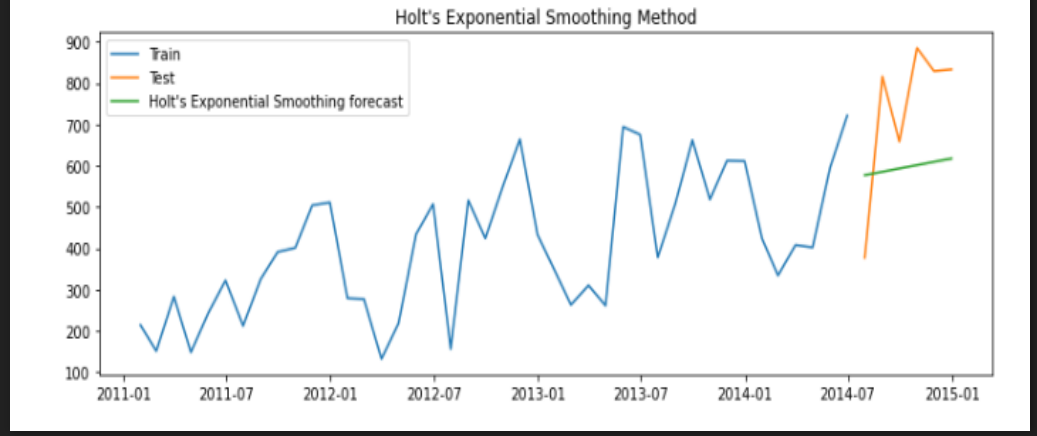
In the weighted average technique, the underlying idea of this technique is that each observation influencing y(t+1) is assigned a specific weight. More recent observations get more weight, whereas the previous observations get less weight. Suppose you consider a time series data of 12 months and are forecasting y t+1 .

- Simple exponential smoothing technique that helps us in capturing the level of time series data.

Holt's Exponential Smoothing Method with Trend

On Quantity_forecast

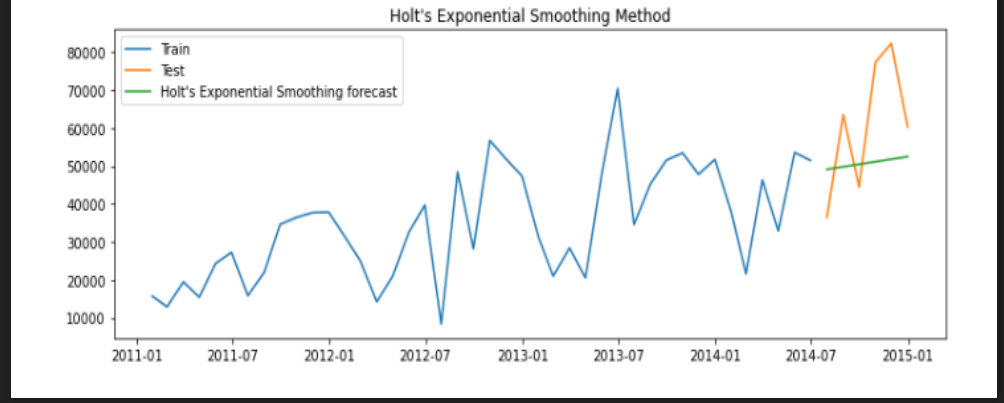
```
59]: plt.figure(figsize=(12,4))
plt.plot(train['Quantity'], label= 'Train')
plt.plot(test['Quantity'], label= 'Test')
plt.plot(y_hat_hes['holt_forecast'], label= "Holt's Exponential Smoothing forecast")
plt.legend(loc='best')
plt.title("Holt's Exponential Smoothing Method")
plt.show()
```



On Sales_forecast

Plot train, test and forecast

```
58]: plt.figure(figsize=(12,4))
plt.plot(train['Sales'], label= 'Train')
plt.plot(test['Sales'], label= 'Test')
plt.plot(y_hat_hes['holt_forecast'], label= "Holt's Exponential Smoothing forecast")
plt.legend(loc='best')
plt.title("Holt's Exponential Smoothing Method")
plt.show()
```

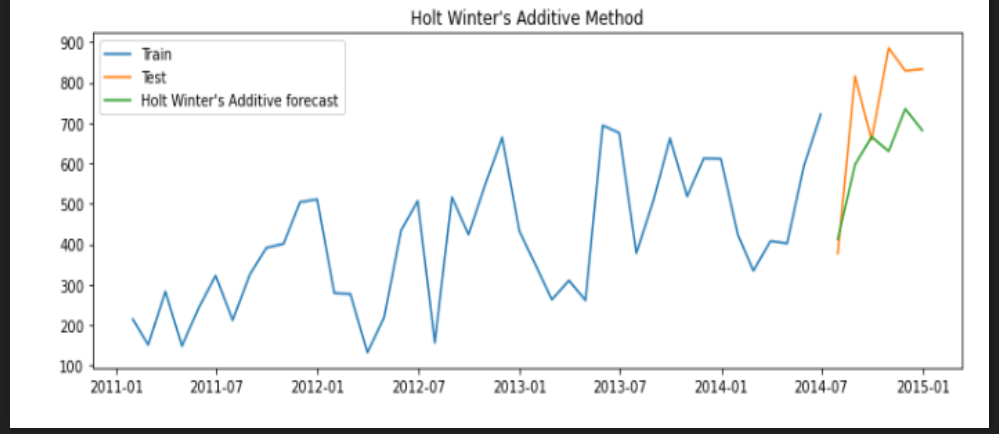


- Technique which forecasts based on level, trend of a time series

Holt Winters's Additive Exponential Smoothing Method(Trend and Seasonality)

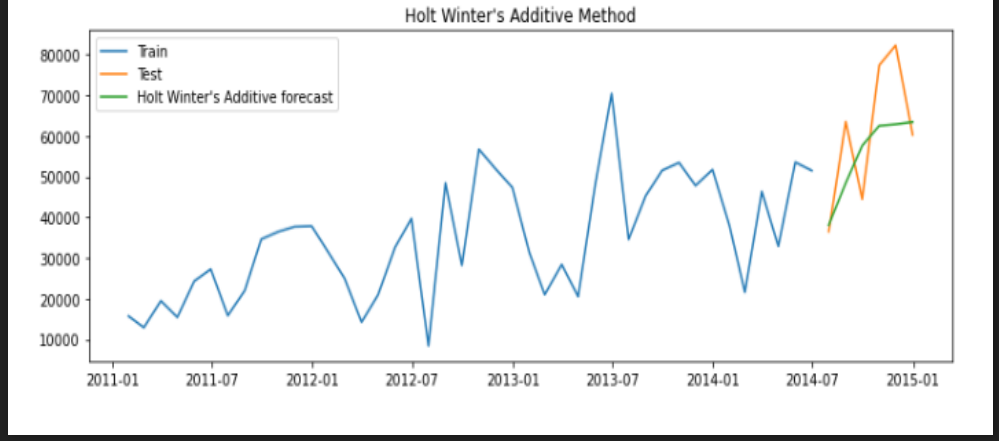
On Quantity_forecast

```
plt.figure(figsize=(12,4))
plt.plot(train['Quantity'], label= 'Train')
plt.plot(test['Quantity'], label= 'Test')
plt.plot(y_hat_hwa['hwa_forecast'], label= "Holt Winter's Additive forecast")
plt.legend(loc='best')
plt.title("Holt Winter's Additive Method")
plt.show()
```



On Sales_forecast

```
plt.figure(figsize=(12,4))
plt.plot(train['Sales'], label= 'Train')
plt.plot(test['Sales'], label= 'Test')
plt.plot(y_hat_hwa['hwa_forecast'], label= "Holt Winter's Additive forecast")
plt.legend(loc='best')
plt.title("Holt Winter's Additive Method")
plt.show()
```

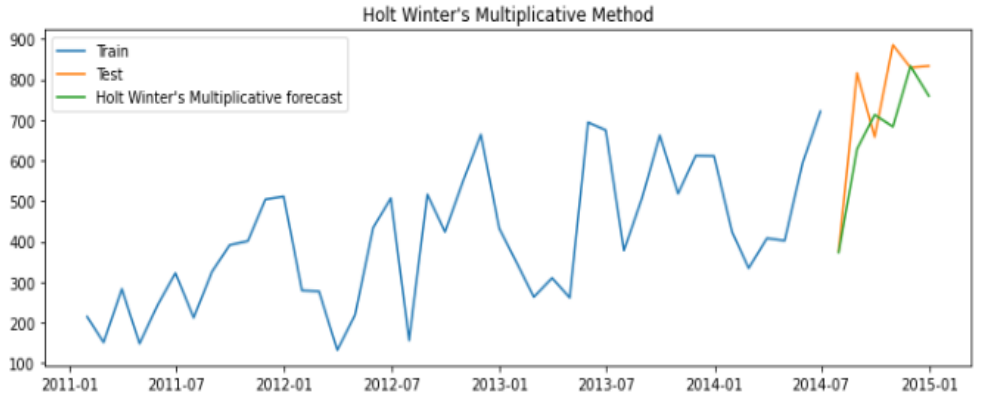


Technique which forecasts based on level, trend and seasonality of a time series. In a time-series data, if the seasonality is not a function of the level component or the difference between subsequent troughs of the time series data does not increase as you progress in the graph, then the Holt-Winters' additive method works best.

Holt Winters's Multiplicative Exponential Smoothing Method (Trend and Seasonality)

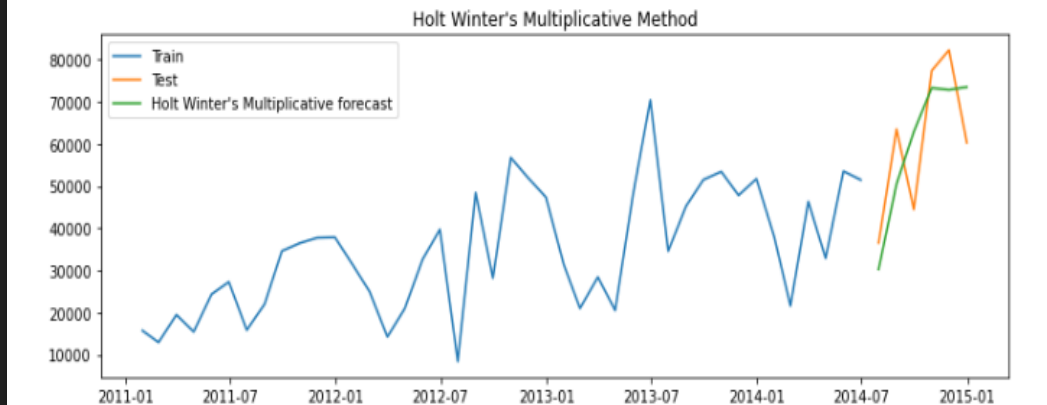
On Quantity_forecast

```
plt.figure(figsize=(12,4))
plt.plot(train['Quantity'], label= 'Train')
plt.plot(test['Quantity'], label= 'Test')
plt.plot(y_hat_hwm['hwm_forecast'], label= "Holt Winter's Multiplicative forecast")
plt.legend(loc='best')
plt.title("Holt Winter's Multiplicative Method")
plt.show()
```



On Sales_forecast

```
plt.figure(figsize=(12,4))
plt.plot(train['Sales'], label= 'Train')
plt.plot(test['Sales'], label= 'Test')
plt.plot(y_hat_hwm['hwm_forecast'], label= "Holt Winter's Multiplicative forecast")
plt.legend(loc='best')
plt.title("Holt Winter's Multiplicative Method")
plt.show()
```



seasonality is a function of the level and the difference between the troughs of the time series data increases as you progress in the graph, then you use the multiplicative method.

Best Smoothing Technique based on least MAPE value

On Quantity_forecast

```
rmse= np.sqrt(mean_squared_error(test['Quantity'], y_hat_hwm['hwm_forecast'])).round(2)
mape= np.round(np.mean(np.abs(test['Quantity']-  y_hat_hwm['hwm_forecast'])/test['Quantity'])*100,2)

tempresults= pd.DataFrame({"Method":["Holt Winter's Multiplicative Method"], 'MAPE':[mape], 'RMSE':[rmse]})
results= pd.concat([results, tempresults])
results
```

66]:

	Method	MAPE	RMSE
0	Naive Method	26.24	174.37
0	Simple Average Method	42.16	371.15
0	Simple Moving Average Method	35.55	279.36
0	Simple exponential smoothing forecast	34.15	261.71
0	Holt's Exponential Smoothing Method	29.29	213.30
0	Holt Winter's Additive Method	15.98	156.24
0	Holt Winter's Multiplicative Method	10.73	118.69

On Sales_forecast

```
[65]: from sklearn.metrics import mean_squared_error
rmse= np.sqrt(mean_squared_error(test['Sales'], y_hat_hwm['hwm_forecast'])).round(2)
mape= np.round(np.mean(np.abs(test['Sales']-  y_hat_hwm['hwm_forecast'])/test['Sales'])*100,2)

tempresults= pd.DataFrame({"Method":["Holt Winter's Multiplicative Method"], 'MAPE':[mape], 'RMSE':[rmse]})
results= pd.concat([results, tempresults])
results
```

[65]:

	Method	MAPE	RMSE
0	Naive Method	26.86	18774.05
0	Simple Average Method	38.18	30846.00
0	Simple Moving Average Method	28.15	23383.65
0	Simple exponential smoothing forecast	27.73	22992.40
0	Holt's Exponential Smoothing Method	25.60	18542.90
0	Holt Winter's Additive Method	17.61	12971.01
0	Holt Winter's Multiplicative Method	19.62	11753.42

INFERENCE : In both the forecasting (Sales and Quantity) , Holt's Winter Multiplicative method is giving best results as its MAPE is least as well as according to the chart with both trend and seasonality present in data as well as its forecast is able to predict the sales and Quantity closer to the actual values Holt's winter multiplicative method is the best option to choose.

Autoregressive models

In an autoregressive model, The regression technique is used to formulate a time series problem. In order to implement autoregressive models, we forecast the future observations using a linear combination of past observations of the same variable

Flow Chart for performing Autoregressive Models

I) Test for Stationarity vs non-Stationarity time series	• Augmented Dickey-Fuller (ADF) Test • KPSS test
The two tools to convert a non-stationary series into stationary series	• Transformation • Differencing
II) Autocorrelation	• ACF • PACF

Then perform different Autoregressive models

1. AR model
2. MA model
3. ARMA model
4. ARIMA model
5. SARIMA model

Satisfy below steps for Auto-Regressive Models

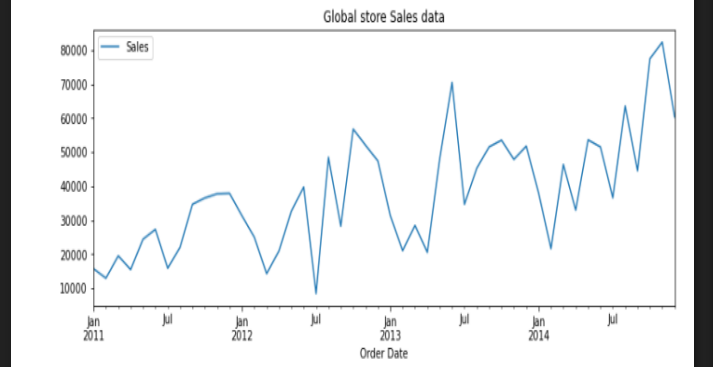
I) Stationarity vs non-Stationarity time series

Test for finding Stationarity and Non-Stationarity

1.Augmented Dickey-Fuller (ADF) Test

2.KPSS test

```
[67]: Sales_forecast.plot(figsize=(12, 4))
      plt.legend(loc='best')
      plt.title('Global store Sales data')
      plt.show(block=False)
```




```
[5]: Quantity_forecast.plot(figsize=(12, 4))
     plt.legend(loc='best')
     plt.title('Global store Quantity data')
     plt.show(block=False)
```

Both Sales and Quantity forecast is shown that data is not Stationarity

Can Check Staionarity or not with tests too(ADF/KPSS) on both data.

ADF test results

Null Hypothesis (H 0): The series is not stationary
○ p−value>0.05
● Alternate Hypothesis (H 1): The series is stationary
○ p−value≤0.05

```
: from statsmodels.tsa.stattools import adfuller
  adf_test= adfuller(Sales_forecast['Sales'])

  print('ADF statistic: %f' % adf_test[0])
  print('Critical value @ 0.05: % .2f' % adf_test[4]['5%'])
  print('p-value: %f' % adf_test[1])
  # p-value is more than 0.05. We fail to reject the null hypot

ADF statistic: -2.220857
Critical value @ 0.05: -2.93
p-value: 0.198763
```

Null Hypothesis (H 0): The series is not stationary
○ p−value>0.05
● Alternate Hypothesis (H 1): The series is stationary
○ p−value≤0.05

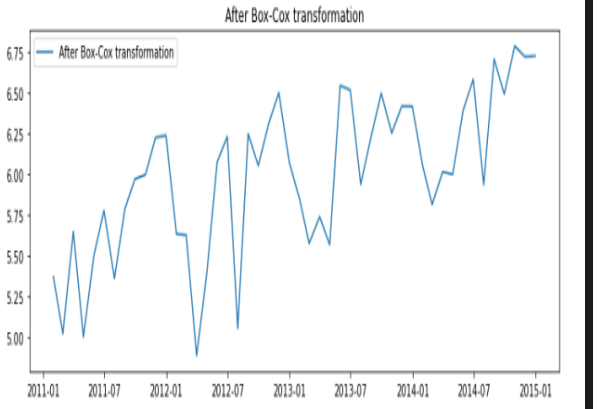
```
]: from statsmodels.tsa.stattools import adfuller
   adf_test= adfuller(Quantity_forecast['Quantity'])

   print('ADF statistic: %f' % adf_test[0])
   print('Critical value @ 0.05: % .2f' % adf_test[4]['5%'])
   print('p-value: %f' % adf_test[1])
   # p-value is more than 0.05. We fail to reject the null hyp

ADF statistic: 0.293145
Critical value @ 0.05: -2.94
p-value: 0.977028
```

P-value is more than 0.5 , hence both sales and Quantity data is not Stationarity

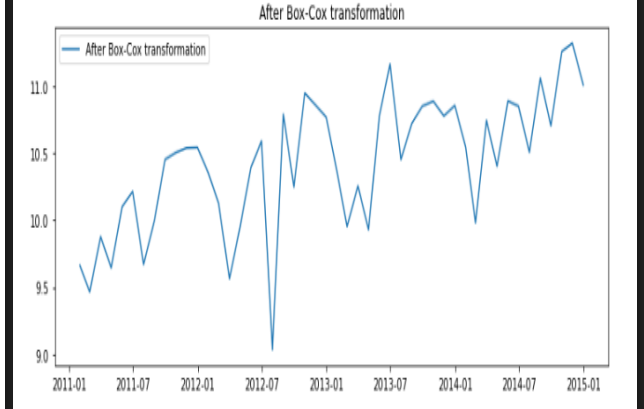
The two tools to convert a non-stationary series into stationary series

1. Box-Cox transformation on Quantity and sales

```
from scipy.stats import boxcox
data_boxcox= pd.Series(boxcox(Quantity_forecast['Quantity'], lmbda=0), index= Quantity_forecast.index)

plt.figure(figsize=(12,4))
plt.plot(data_boxcox, label='After Box-Cox transformation')
plt.legend(loc='best')
plt.title('After Box-Cox transformation')
plt.show()
```



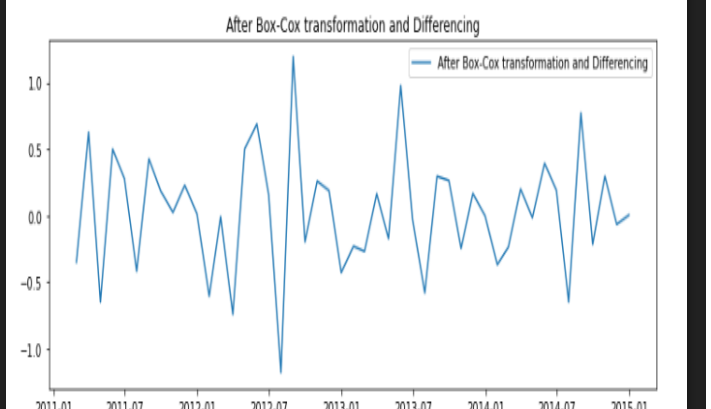

```
from scipy.stats import boxcox
data_boxcox= pd.Series(boxcox(Sales_forecast['Sales'], lmbda=0), index= Sales_forecast.index)

plt.figure(figsize=(12,4))
plt.plot(data_boxcox, label='After Box-Cox transformation')
plt.legend(loc='best')
plt.title('After Box-Cox transformation')
plt.show()
```



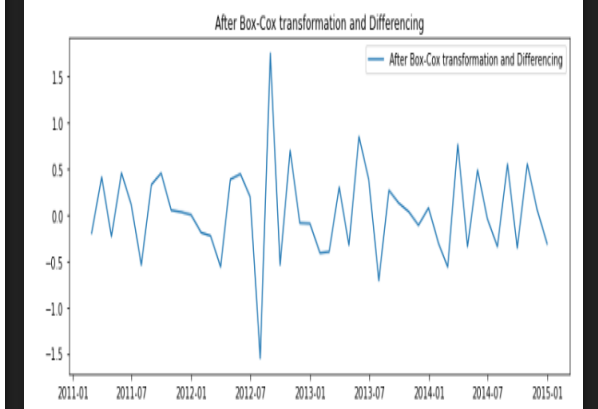
This method is to introduce the stationarity by making the variance constant.

2. Differencing on Quantity and sales

```
data_boxcox_diff= pd.Series(data_boxcox - data_boxcox.shift(), index= Quantity_forecast.index)
data_boxcox_diff.dropna(inplace=True)

plt.figure(figsize=(12,4))
plt.plot(data_boxcox_diff, label='After Box-Cox transformation and Differencing')
plt.legend(loc='best')
plt.title('After Box-Cox transformation and Differencing')
plt.show()
```




```
data_boxcox_diff= pd.Series(data_boxcox - data_boxcox.shift(), index= Sales_forecast.index)
data_boxcox_diff.dropna(inplace=True)

plt.figure(figsize=(12,4))
plt.plot(data_boxcox_diff, label='After Box-Cox transformation and Differencing')
plt.legend(loc='best')
plt.title('After Box-Cox transformation and Differencing')
plt.show()
```



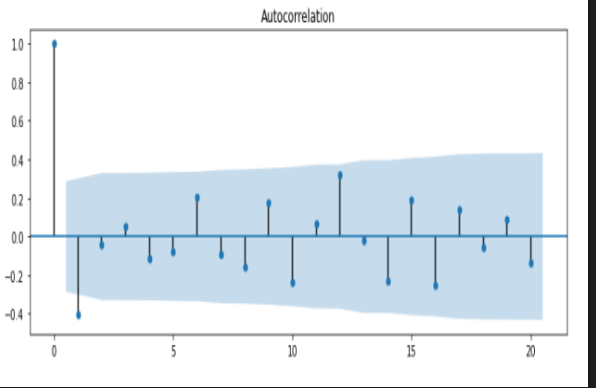
This method used because trend and sesonality will be removed using differentiation

II) Autocorrelation :

The autocorrelation function tells about the correlation between an observation with its lagged values.

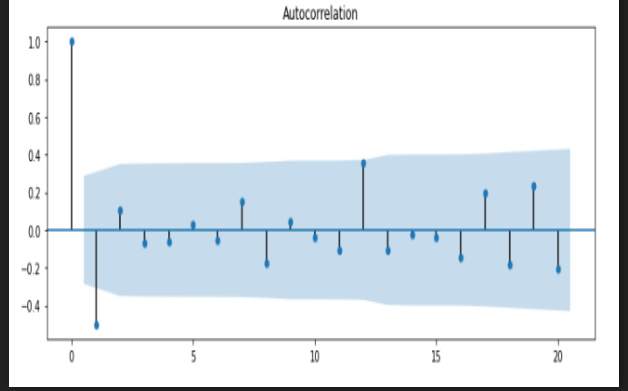
1.ACF of Quantity

```
from statsmodels.graphics.tsaplots import plot_acf

plt.figure(figsize=(12,4))
acf= plot_acf(data_boxcox_diff, ax= plt.gca(), lags=20)
plt.show()
```



ACF of Sales

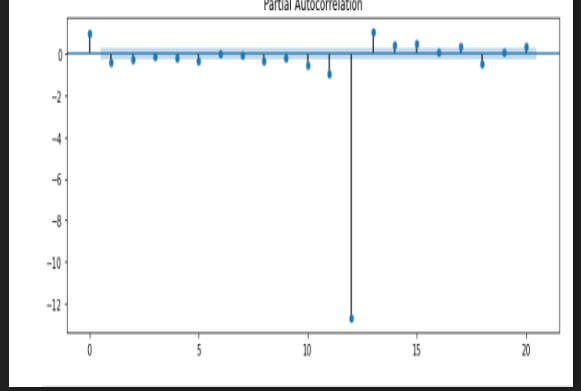
```
from statsmodels.graphics.tsaplots import plot_acf

plt.figure(figsize=(12,4))
acf= plot_acf(data_boxcox_diff, ax= plt.gca(), lags=20)
plt.show()
```



capture indirect relationship

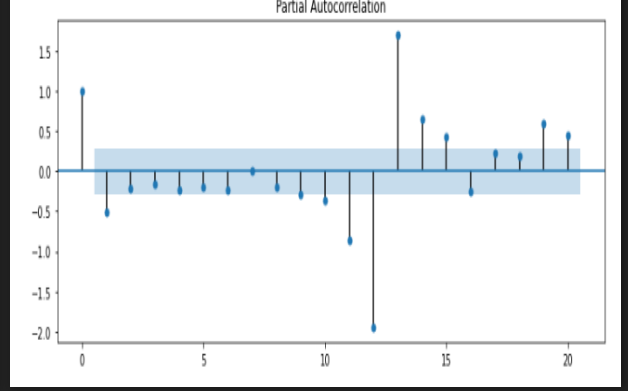
2.PACF of Quality

```
from statsmodels.graphics.tsaplots import plot_pacf
plt.figure(figsize=(12,4))
pacf= plot_pacf(data_boxcox_diff, ax= plt.gca(), lags=20)
plt.show()
```



PACF for sales

```
from statsmodels.graphics.tsaplots import plot_pacf
plt.figure(figsize=(12,4))
pacf= plot_pacf(data_boxcox_diff, ax= plt.gca(), lags=20)
plt.show()
```



capture direct relationship

Then perform different Autoregressive models

1.AR model

2. MA model

3. ARMA model

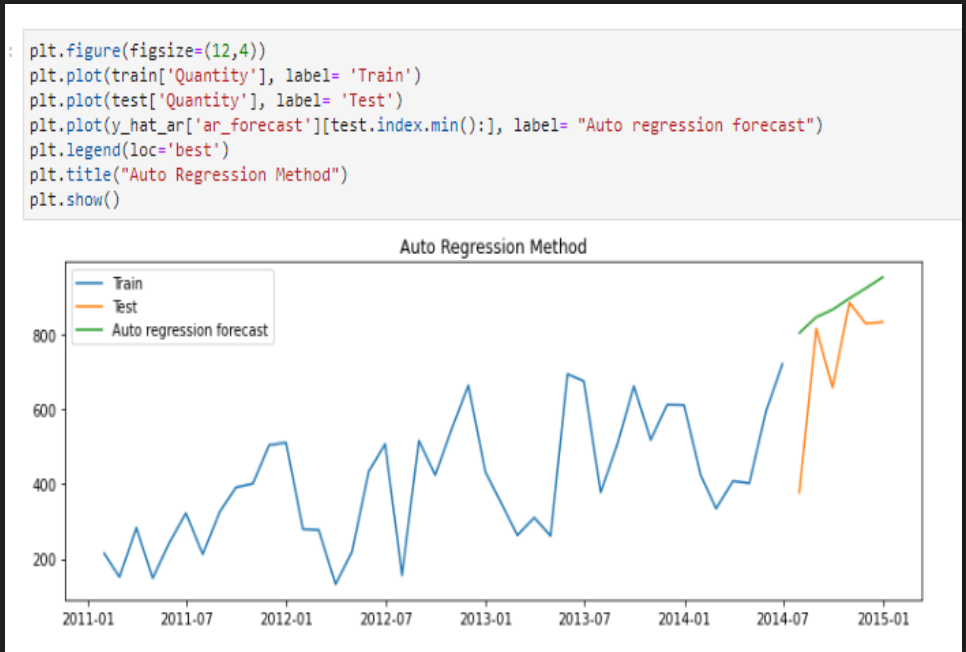
4. ARIMA model

5. SARIMA model

A Simple Auto Regressive model

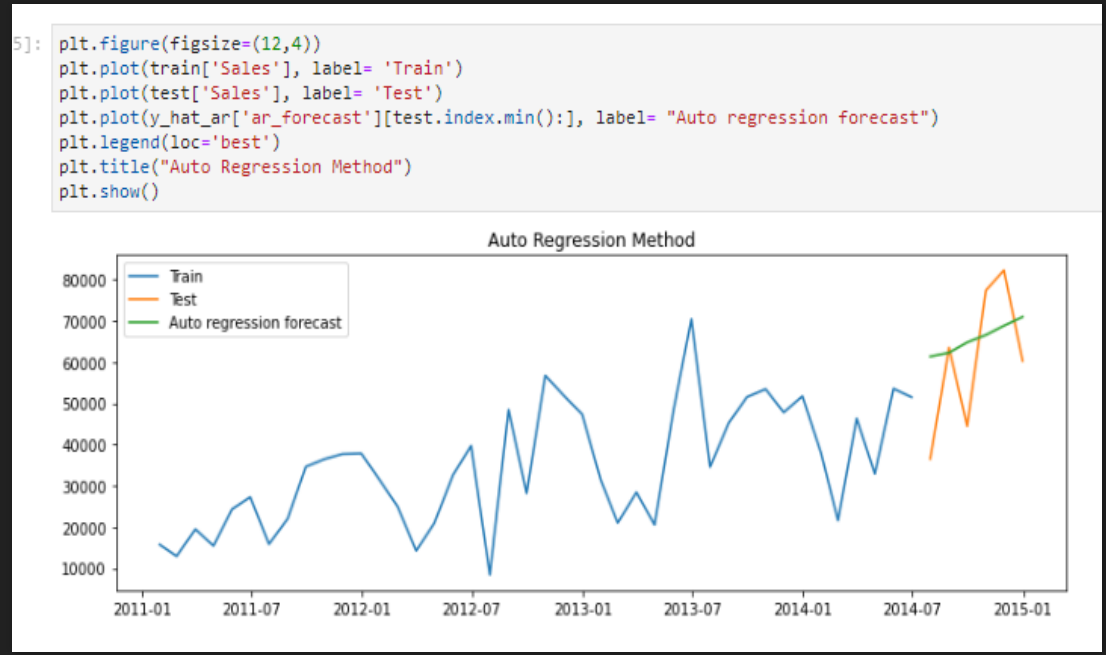
On Quantity_forecast

```
plt.figure(figsize=(12,4))
plt.plot(train['Quantity'], label= 'Train')
plt.plot(test['Quantity'], label= 'Test')
plt.plot(y_hat_ar['ar_forecast'][test.index.min():], label= "Auto regression forecast")
plt.legend(loc='best')
plt.title("Auto Regression Method")
plt.show()
```



On Sales_forecast

```
plt.figure(figsize=(12,4))
plt.plot(train['Sales'], label= 'Train')
plt.plot(test['Sales'], label= 'Test')
plt.plot(y_hat_ar['ar_forecast'][test.index.min():], label= "Auto regression forecast")
plt.legend(loc='best')
plt.title("Auto Regression Method")
plt.show()
```



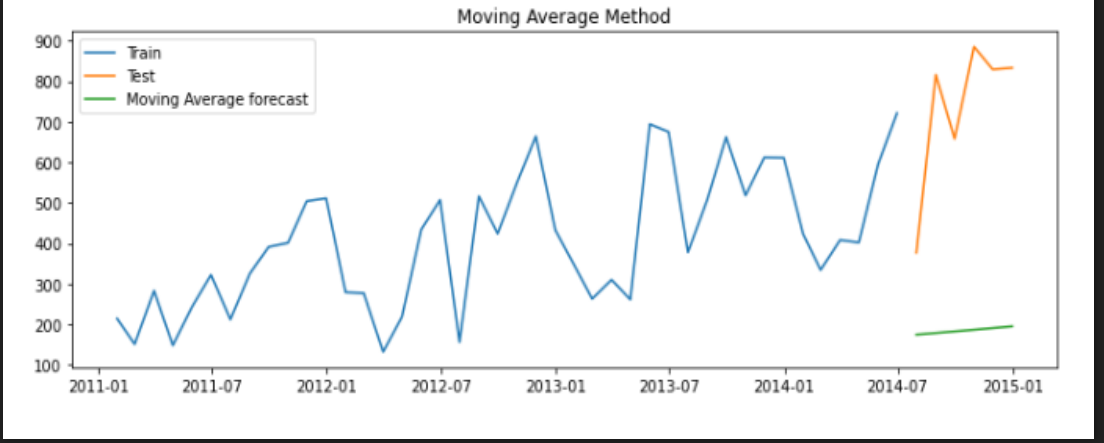
The Simple Auto Regressive model predicts the future observation as linear regression of one or more past observations. In simpler terms, the simple autoregressive model forecasts the dependent variable (future observation) when one or more independent variables are known (past observations). This model has a parameter ‘p’ called lag order . Lag order is the maximum number of lags used to build ‘p’ number of past data points to predict future data points.

here p=1 given

Moving Average Model (MA) model

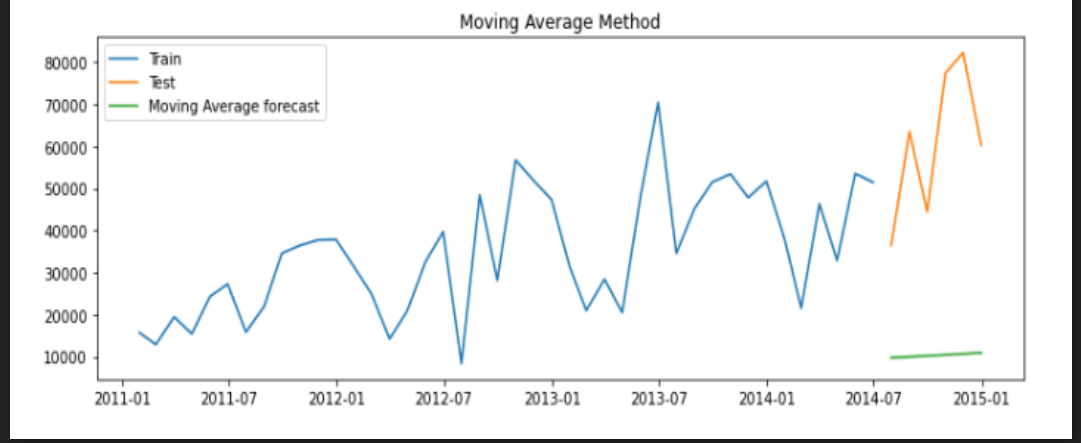
On Quantity_forecast

```
plt.figure(figsize=(12,4))
plt.plot(train['Quantity'], label= 'Train')
plt.plot(test['Quantity'], label= 'Test')
plt.plot(y_hat_ma['ma_forecast'][test.index.min():], label= "Moving Average forecast")
plt.legend(loc='best')
plt.title("Moving Average Method")
plt.show()
```



On Sales_forecast

```
plt.figure(figsize=(12,4))
plt.plot(train['Sales'], label= 'Train')
plt.plot(test['Sales'], label= 'Test')
plt.plot(y_hat_ma['ma_forecast'][test.index.min():], label= "Moving Average forecast")
plt.legend(loc='best')
plt.title("Moving Average Method")
plt.show()
```



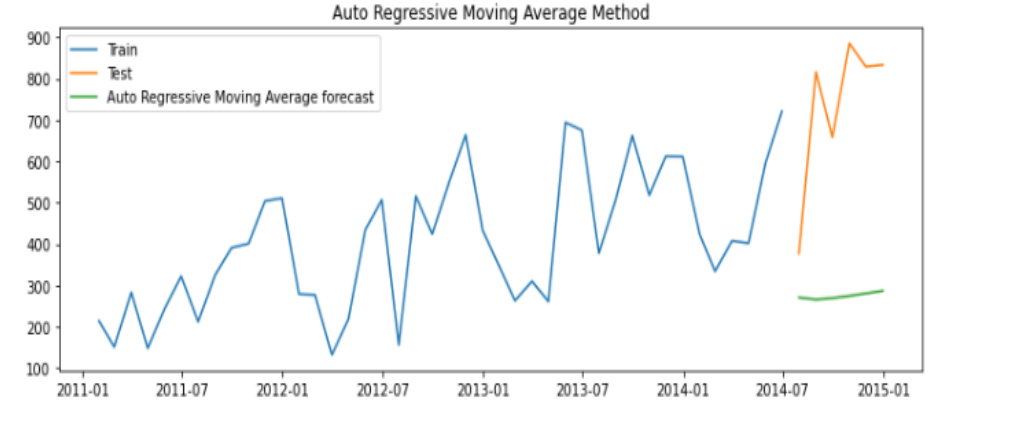
The Moving Average Model models the future forecasts using past forecast errors in a regression-like model. This model has a parameter ‘q’ called window size over which linear combination of errors are calculated.

here q=1 given

Auto Regressive Moving Average (ARMA) Method

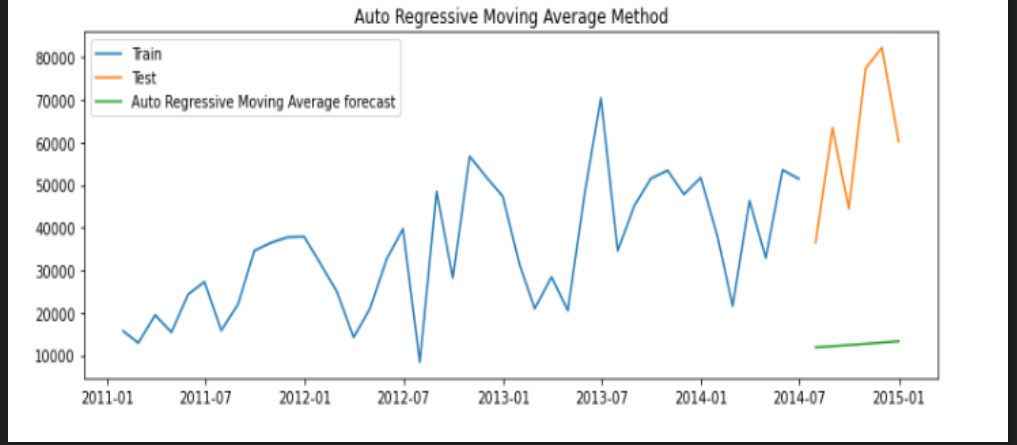
On Quantity_forecast

```
plt.figure(figsize=(12,4))
plt.plot(train['Quantity'], label= 'Train')
plt.plot(test['Quantity'], label= 'Test')
plt.plot(y_hat_arma['arma_forecast'][test.index.min():], label= "Auto Regressive Moving Average forecast")
plt.legend(loc='best')
plt.title("Auto Regressive Moving Average Method")
plt.show()
```



On Sales_forecast

```
plt.figure(figsize=(12,4))
plt.plot(train['Sales'], label= 'Train')
plt.plot(test['Sales'], label= 'Test')
plt.plot(y_hat_arma['arma_forecast'][test.index.min():], label= "Auto Regressive Moving Average forecast")
plt.legend(loc='best')
plt.title("Auto Regressive Moving Average Method")
plt.show()
```



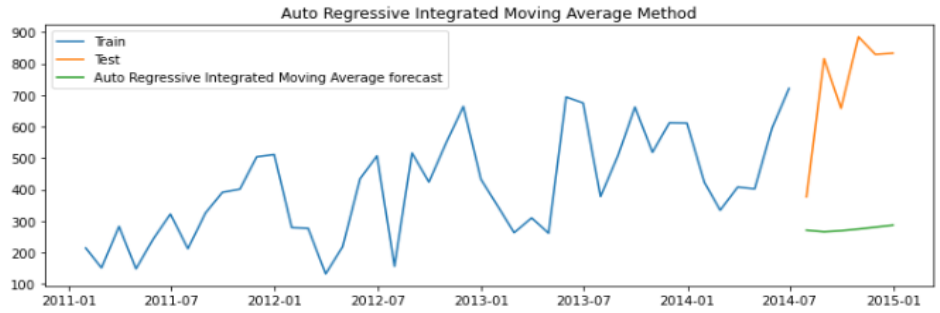
A time series that exhibits the characteristics of an AR(p) and/or an MA(q) process can be modelled using an ARMA(p,q) model. To determine the parameters ‘p’ and ‘q’ —Plot autocorrelation function (ACF) and partial autocorrelation function (PACF)If you check the plot for PACF, you will see that you need to select p as the highest lag where partial autocorrelation is significantly high.Similarly from the ACF plot, select q as the highest lag beyond which autocorrelation dies down.

here p=1 and q=1 are given

Auto Regressive Integrated Moving Average (ARIMA)

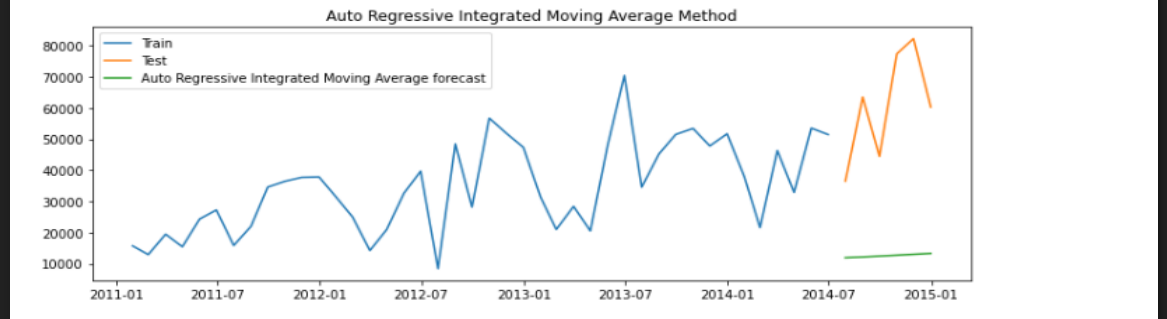
On Quantity_forecast

```
plt.figure(figsize=(12,4))
plt.plot(train['Quantity'], label= 'Train')
plt.plot(test['Quantity'], label= 'Test')
plt.plot(y_hat_arima['arima_forecast'][test.index.min():], label= "Auto Regressive Integrated Moving Average forecast")
plt.legend(loc='best')
plt.title("Auto Regressive Integrated Moving Average Method")
plt.show()
```



On Sales_forecast

```
plt.figure(figsize=(12,4))
plt.plot(train['Sales'], label= 'Train')
plt.plot(test['Sales'], label= 'Test')
plt.plot(y_hat_arima['arima_forecast'][test.index.min():], label= "Auto Regressive Integrated Moving Average forecast")
plt.legend(loc='best')
plt.title("Auto Regressive Integrated Moving Average Method")
plt.show()
```



Steps of ARIMA model

Original time series is differenced to make it stationary

Differenced series is modeled as a linear regression of

1.One or more past observations

2.Past forecast errors

ARIMA model has three parameters

here p=1, q=1 and d=1 given

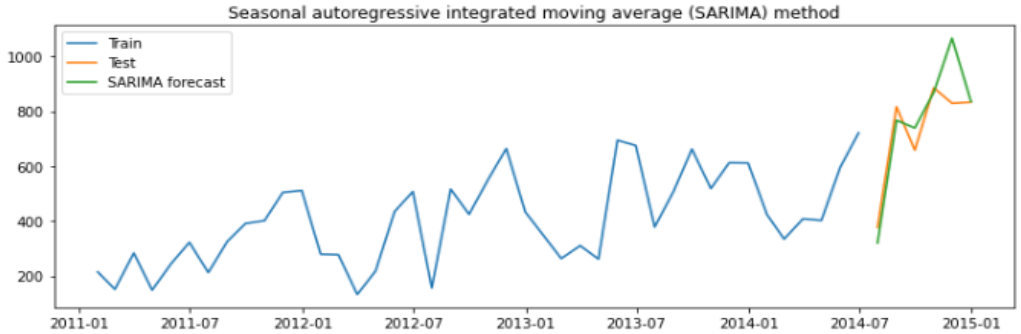
○ Here the new parameter introduced is the ‘I’ part called integrated. It removes the trend (non-stationarity) and later integrates the trend to the original series.Initially you applied both the boxcox transformation and differencing in order to covert the data into a stationary time-series data.

In ARIMA, just applying boxcox before building the model and letting the model take care of the differencing i.e. the trend component itself.

Seasonal Auto Regressive Integrated Moving Average(SARIMA)

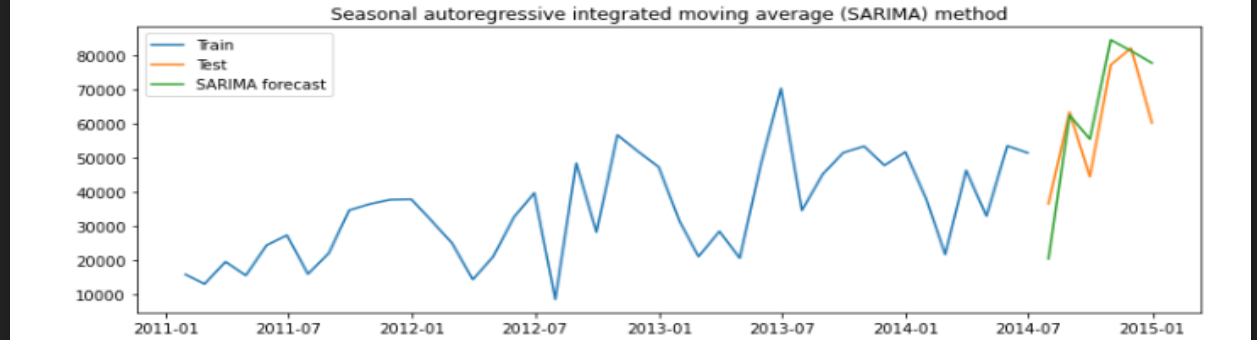
On Quantity_forecast

```
plt.figure(figsize=(12,4))
plt.plot(train['Quantity'], label='Train')
plt.plot(test['Quantity'], label='Test')
plt.plot(y_hat_sarima['sarima_forecast'][test.index.min():], label='SARIMA forecast')
plt.legend(loc='best')
plt.title('Seasonal autoregressive integrated moving average (SARIMA) method')
plt.show()
```



On Sales_forecast

```
plt.figure(figsize=(12,4))
plt.plot(train['Sales'], label='Train')
plt.plot(test['Sales'], label='Test')
plt.plot(y_hat_sarima['sarima_forecast'][test.index.min():], label='SARIMA forecast')
plt.legend(loc='best')
plt.title('Seasonal autoregressive integrated moving average (SARIMA) method')
plt.show()
```



SARIMA brings all the features of an ARIMA model with an extra feature - seasonality.
The non-seasonal elements of SARIMA
1.Time series is differenced to make it stationary.
2. Models future observation as linear regression of past observations and past forecast errors.

The seasonal elements of SARIMA
Perform seasonal differencing on time series.
Model future seasonality as linear regression of past observations of seasonality and past forecast errors of seasonality.
The parameters ‘p’, ‘d’, ‘q’ and ‘P’, ‘D’, ‘Q’
here p=q=d=1 and P=D=Q=1 and m=12

Best Auto-Regressive model based on least MAPE value

On Quantity forecast

	Method	RMSE	MAPE
0	Auto Regressive(AR) Method	204.28	29.30
0	Moving Average (MA) Method	573.80	72.76
0	Auto Regressive Moving Average (ARMA) Method	489.45	59.25
0	Auto Regressive Integrated Moving Average (ARI...	489.45	59.25
0	Seasonal autoregressive integrated moving aver...	106.93	10.70

On Sales forecast

	Method	RMSE	MAPE
0	Auto Regressive(AR) Method	15505.03	27.27
0	Moving Average (MA) Method	52903.35	81.64
0	Auto Regressive Moving Average (ARMA) Method	50762.48	77.67
0	Auto Regressive Integrated Moving Average (ARI...	50762.48	77.67
0	Seasonal autoregressive integrated moving aver...	11179.02	18.38

INFERENCE: In both the forecasting (Sales and Quantity) , Seasonal Auto Regressive Integrated Moving Average(SARIMA) method is giving best results as its MAPE is least as well as its forecast is able to predict the sales and Quantity closer to the actual values and according to the chart also with both trend and seasonality present in data , Seasonal Auto Regressive Integrated Moving Average(SARIMA) is the best option to choose.

Forecast the sales and quantity of the products for the next 6 months

Quantity prediction for next 6 month

Order Date	Quantity	SARIMA	Holt Winter Additive	Holt Winter Multlipicative
2014-07-31	377	319.802291	412.603446	373.227992
2014-08-31	816	766.920546	596.452647	627.627281
2014-09-30	658	738.828170	665.429487	712.543317
2014-10-31	885	868.352196	630.085274	683.503273
2014-11-30	829	1065.870980	734.348021	832.421480
2014-12-31	833	835.176232	681.357068	759.278085

Sales prediction for next 6 month

Order Date	Sales	SARIMA	Holt Winter Additive	Holt Winter Multlipicative
2014-07-31	36524.3028	20330.503144	38120.571471	30278.415968
2014-08-31	63521.7729	62601.942526	48451.426795	50494.663578
2014-09-30	44477.2662	55507.834089	57615.965549	62901.684796
2014-10-31	77379.8286	84720.456714	62518.929420	73280.128482
2014-11-30	82286.3583	81487.233021	62885.180755	72836.867938
2014-12-31	60292.1310	77914.288386	63445.198577	73505.060185

Conclusion

- Segment APAC’s Consumer is the most profitable and consistent.
- seasonal pattern is shown by both the markets Sales and Quantity .
- Slightly increasing upward trend is present in both sales and quantity
- We created total 12 forecasting models for the most profitable market_segment (APAC consumer) out of which the methods whose forecast is able to predict the sales and Quantity closer to the actual values are **Holt Winter's (additive and multiplicative) from Smoothing Techniques** and **SARIMA** method from Autoregressive models

Thank you