NAME : SAYAK DAS

FORECASTING OF THE NEXT DEMAND

ts is a time period component. And the monthly sales of mixer grinder from 2016 to 2019 is shown. I have 4 years of data. "ts" represents the time series index or time period component. It is a sequential numbering or labeling of the data points in the time series dataset. Each value of "**ts**" corresponds to a specific time point or period, in this case, represented as a combination of month and year.

Year	ts	Month	Sales
2016	1	1	460
2016	2	2	446
2016	3	3	413
2016	4	4	399
2016	5	5	401
2016	6	6	373
2016	7	7	380
2016	8	8	354
2016	9	9	592
2016	10	10	482
2016	11	11	574
2016	12	12	699
2017	13	1	488
2017	14	2	505
2017	15	3	539
2017	16	4	412
2017	17	5	533
2017	18	6	368
2017	19	7	484
2017	20	8	512
2017	21	9	651
2017	22	10	568
2017	23	11	647
2017	24	12	863
2018	25	1	612
2018	26	2	675
2018	27	3	610
2018	28	4	679
2018	29	5	402

2018	29	5	402		
2018	30	6	451		
2018	31	7	501		
2018	32	8	695		
2018	33	9	610		
2018	34	10	636		
2018	35	11	821		
2018	36	12	669		
2019	37	1	484		
2019	38	2	662		
2019	39	3	472		
2019	40	4	420		
2019	41	5	339		
2019	42	6	364		
2019	43	7	373		
2019	44	8	317		
2019	45	9	333		
2019	46	10	333		
2019	47	11	300		

Decomposition of a time series into its trend, seasonality, and error components.
The trend component represents the overall direction of the time series, whether it is increasing, decreasing, or staying constant. Seasonality refers to patterns that repeat over a specific period, such as daily, weekly, or yearly patterns. The error component represents the unexplained or random variation in the data.

To estimate the components of a time series, you can use various methods.
One simple approach is the moving average method, where you calculate the average of a certain number of consecutive observations to smooth out the fluctuations and estimate the trend component.
For example, using a 3-period moving average, you would calculate the average of the current observation, the one before it, and the one after it.

Once you have estimated the trend and seasonality components, you can analyze the remaining error component to understand how other variables, such as price or competing product prices, affect the series. This can be done through regression analysis or by examining the relationship between the error component and the relevant variables.

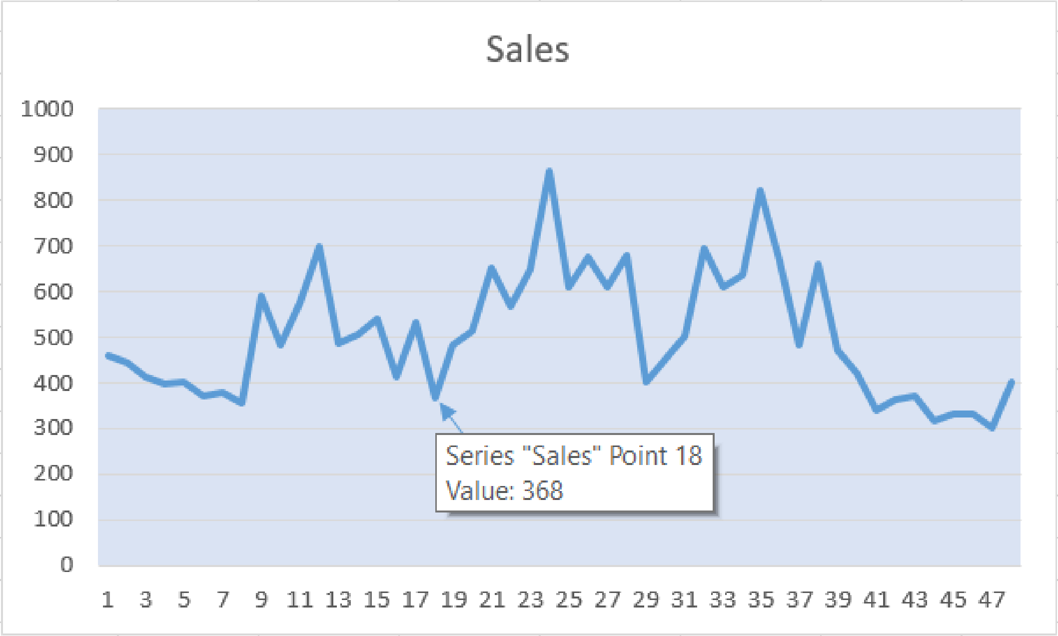
Overall, decomposing a time series into its components allows you to understand and model the different factors that contribute to its behavior.
By analyzing and modeling each component separately, you can gain insights into the underlying patterns and relationships within the data.

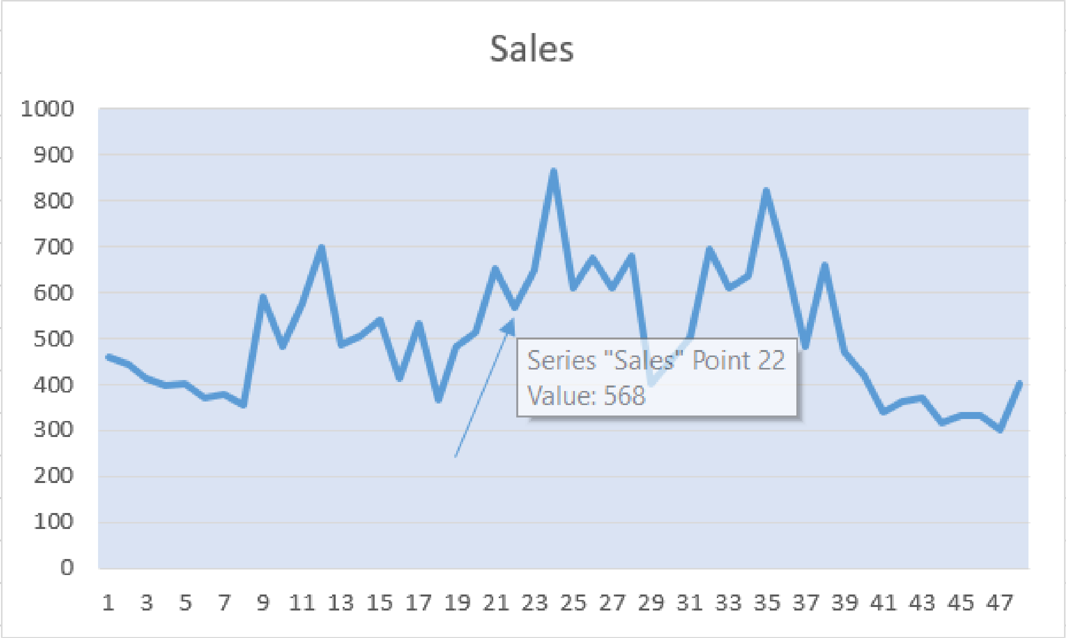
To identify the seasonality component, you can analyze the patterns that repeat over a specific period.
For instance, if you have monthly data and observe a pattern that repeats every 12 months, you can infer a yearly seasonality. You can estimate the seasonality component by subtracting the trend component from the original data.

The remaining component after removing the trend and seasonality is the error component.
It represents the part of the time series that cannot be explained by the trend and seasonality factors alone. This component may include random fluctuations, measurement errors, or other factors that influence the series.

We can find the error part and do a regression analysis after removing the trend and seasonality components from time series component or x variable

So let's use the moving average for smoothening the seasonality change

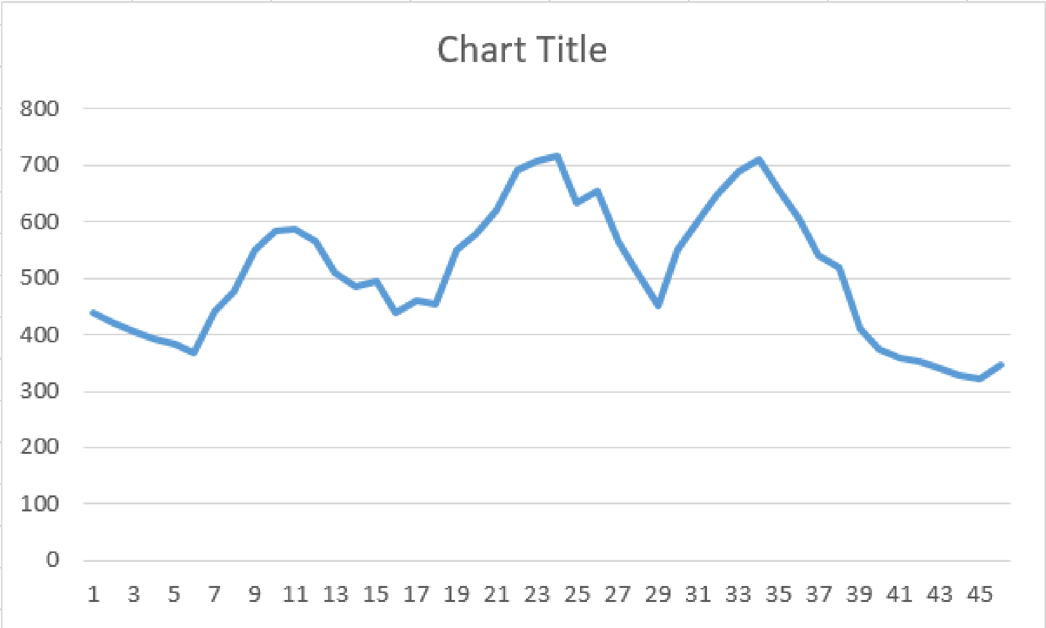




Hence we can do odd smoothing with seasonality 3 or quarterly with 3 period moving average

Year	ts	Month	Sales	SMA(SIMPLE MOVING AVAERAG)
2016	1	1	460	
2016	2	2	446	439.6666667
2016	3	3	413	419.3333333
2016	4	4	399	404.3333333
2016	5	5	401	391
2016	6	6	373	384.6666667
2016	7	7	380	369
2016	8	8	354	442
2016	9	9	592	476
2016	10	10	482	549.3333333
2016	11	11	574	585
2016	12	12	699	587
2017	13	1	488	564
2017	14	2	505	510.6666667
2017	15	3	539	485.3333333
2017	16	4	412	494.6666667
2017	17	5	533	437.6666667
2017	18	6	368	461.6666667
2017	19	7	484	454.6666667
2017	20	8	512	549
2017	21	9	651	577
2017	22	10	568	622
2017	23	11	647	692.6666667
2017	24	12	863	707.3333333
2018	25	1	612	716.6666667
2018	26	2	675	632.3333333
2018	27	3	610	654.6666667
2018	28	4	679	563.6666667
2018	29	5	402	510.6666667
2018	30	6	451	451.3333333

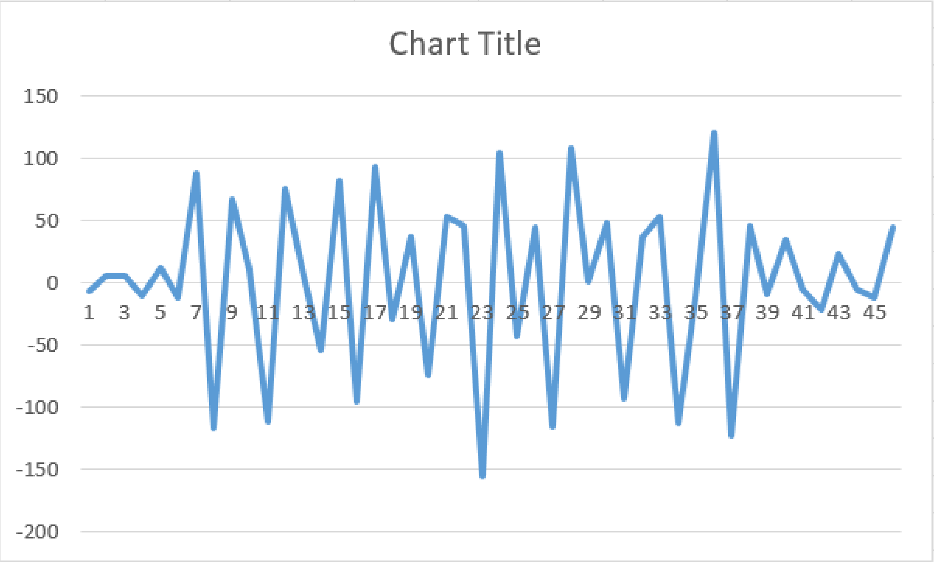
This cell is left blank because there is no previous data



Therefore, on plotting the seasonality is now smoothened

F3 =E3-D3

	A	B	C	D	E	F
1	Year	ts	Month	Sales	SMA(SIMPLE MOVING AVAERAG)	error
2	2016	1	1	460		
3	2016	2	2	446	439.6666667	-6.33333
4	2016	3	3	413	419.3333333	6.333333
5	2016	4	4	399	404.3333333	5.333333
6	2016	5	5	401	391	-10
7	2016	6	6	373	384.6666667	11.66667
8	2016	7	7	380	369	-11
9	2016	8	8	354	442	88
10	2016	9	9	592	476	-116
11	2016	10	10	482	549.3333333	67.33333
12	2016	11	11	574	585	11
13	2016	12	12	699	587	-112
14	2017	13	1	488	564	76
15	2017	14	2	505	510.6666667	5.666667
16	2017	15	3	539	485.3333333	-53.6667
17	2017	16	4	412	494.6666667	82.66667
18	2017	17	5	533	437.6666667	-95.3333
19	2017	18	6	368	461.6666667	93.66667
20	2017	19	7	484	454.6666667	-29.3333
21	2017	20	8	512	549	37
22	2017	21	9	651	577	-74
23	2017	22	10	568	622	54
24	2017	23	11	647	692.6666667	45.66667
25	2017	24	12	863	707.3333333	-155.667
26	2018	25	1	612	716.6666667	104.6667
27	2018	26	2	675	632.3333333	-42.6667
28	2018	27	3	610	654.6666667	44.66667
29	2018	28	4	679	563.6666667	-115.333
30	2018	29	5	402	510.6666667	108.6667
31	2018	30	6	451	451.3333333	0.333333
32	2018	31	7	501	549	48



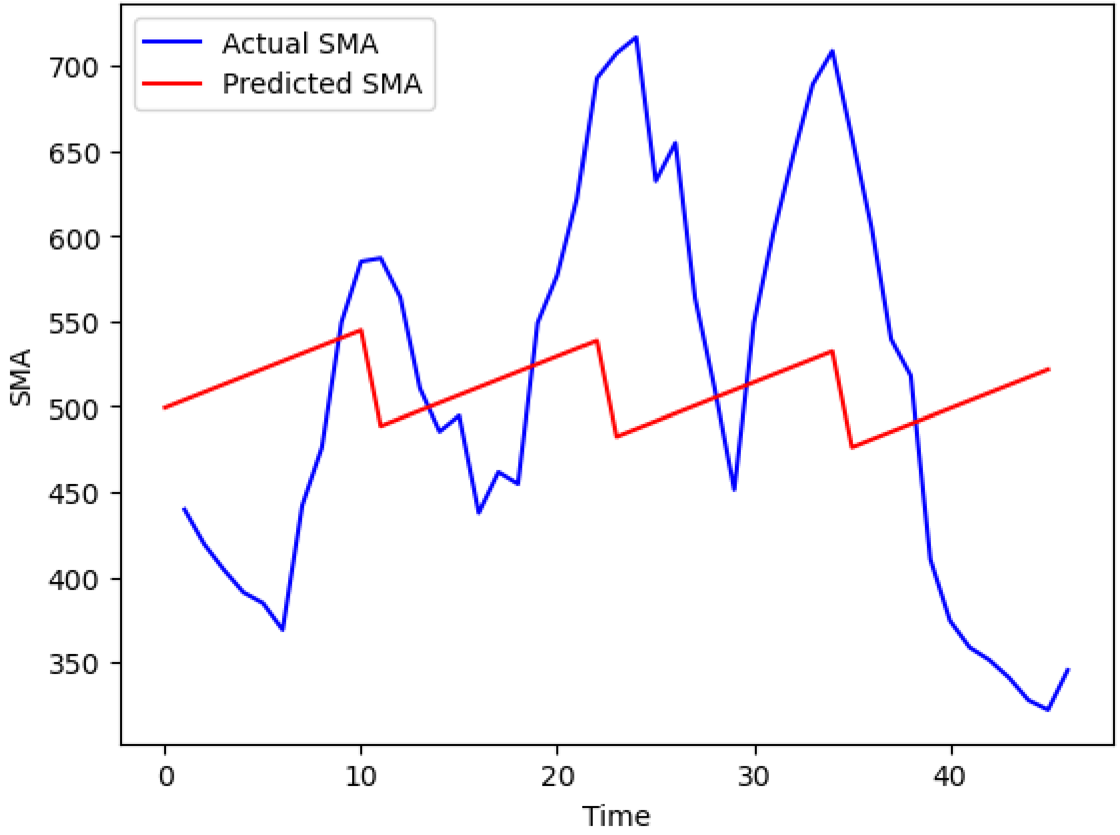
On plotting the error, we see that there is a seasonality of 2 & 3 in it, this is the jump is after each 2 or 3 value

We can also find the error by fitting a linear regression model and do the prediction

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Read the dataset
df=pd.read_csv("D:\\data ANALYTICS AND SCIENCE\\NPTEL- MARKETING Analytics\\sales  error.csv")

# Remove rows with missing values
df_1 = df.dropna()

# Convert month to factor
df_1['Month'] = df_1['Month'].astype('category')


# Fit the linear regression model
model = LinearRegression()
model.fit(df_1[['ts', 'Month']], df_1['SMA'])

# Predict values based on the model
predicted = model.predict(df_1[['ts', 'Month']])

# Calculate the errors
errors = df_1['SMA']- predicted

# Calculate RMSE
rmse = np.sqrt(mean_squared_error(df_1['SMA'], predicted))
rmse

# Plot line graph
plt.plot(df_1['SMA'], color='blue', label='Actual SMA')
plt.plot(predicted, color='red', label='Predicted SMA')
plt.xlabel('Time')
plt.ylabel('SMA')
plt.legend()
plt.show()
```

```
# Calculate RMSE
rmse = np.sqrt(mean_squared_error(df_1['SMA'], predicted))
rmse
```

```
114.42655729979907
```



An RMSE of 114.4266 suggests that, on average, the predicted SMA values differ from the actual SMA values by approximately 114.4266 units.

THIS MODEL IS NOT PREDICTING WELL SO WE CAN USE OTHER MODEL

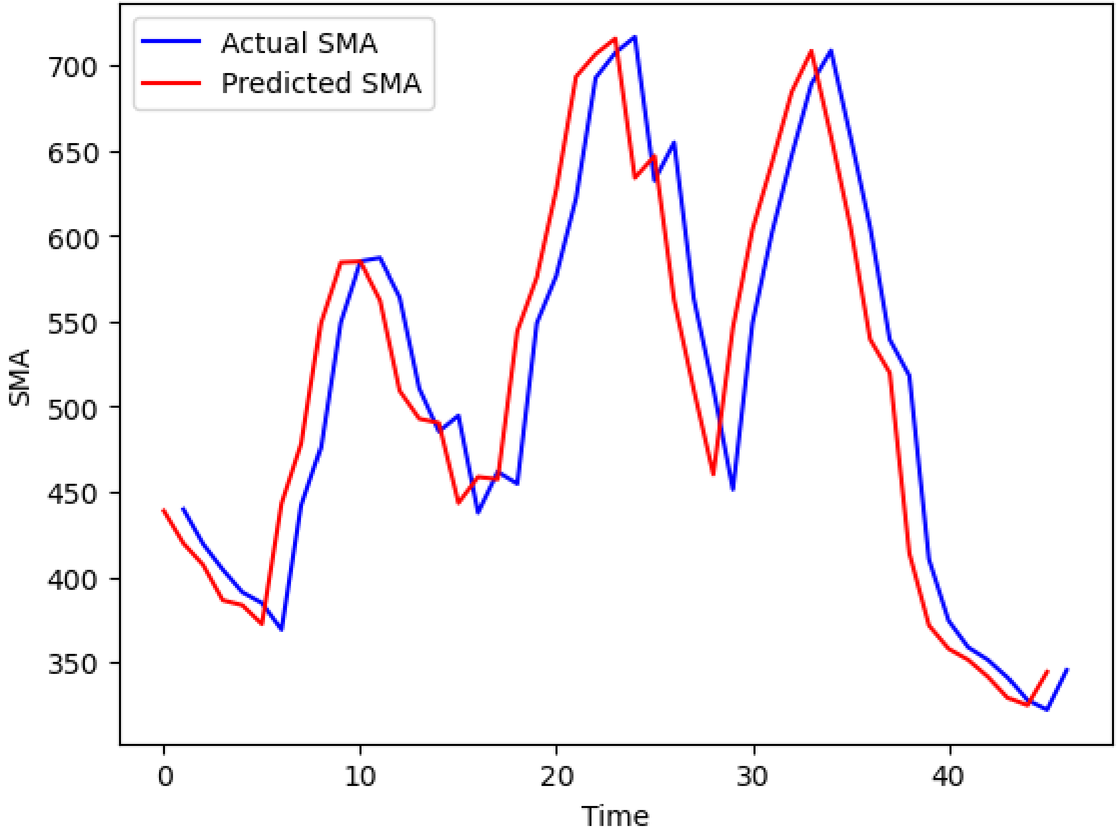
GRADEINT BOOSTING

```
from sklearn.ensemble import GradientBoostingRegressor

# Read the dataset
df = pd.read_csv("D:\\data ANALYTICS AND SCIENCE\\NPTEL- MARKETING Analytics\\sales  error.csv")

# Remove rows with missing values
df_1 = df.dropna()

# Convert month to factor in a new DataFrame
df_2 = df_1.copy()
df_2['Month'] = df_2['Month'].astype('category')

# Fit the Gradient Boosting model
model = GradientBoostingRegressor()
model.fit(df_2[['ts', 'Month']], df_2['SMA'])

# Predict values based on the model
predicted = model.predict(df_2[['ts', 'Month']])

# Calculate the errors
errors = df_2['SMA'] - predicted

# Calculate RMSE
rmse = np.sqrt(mean_squared_error(df_2['SMA'], predicted))
rmse
```

```
3.1807005323769713
```

gradient boost has hugely reduced the error

```
# Plot line graph
plt.plot(df_2['SMA'], color='blue', label='Actual SMA')
plt.plot(predicted, color='red', label='Predicted SMA')
plt.xlabel('Time')
plt.ylabel('SMA')
plt.legend()
plt.show()
```



So the GRADIENT BOOST has decreased the moving average error

Forecasting next year's 1st quarter sales

```
pred=pd.read_csv("D:\\data ANALYTICS AND SCIENCE\\NPTEL- MARKETING Analytics\\next 4 month prediction.csv")
pred
```

	Year	ts	Month
0	2020	49	1
1	2020	50	2
2	2020	51	3
3	2020	52	4

Utilizing the trained Gradient Boosting model to predict future values of the target variable (SMA) based on the input features (ts and Month). This can help to estimate the expected values for the upcoming time periods.

```
# Convert month to factor in the "pred" dataframe
pred['Month'] = pred['Month'].astype('category')

# Make predictions using the trained model
predicted_pred = model.predict(pred[['ts', 'Month']])

# Add the predicted values to the "pred" dataframe
pred['Predicted_SMA'] = predicted_pred

# Print the "pred" dataframe with predicted values
print(pred)
```

```
   Year  ts Month  Predicted_SMA
0  2020  49     1     417.548106
1  2020  50     2     383.685415
2  2020  51     3     385.415404
3  2020  52     4     381.031363
```