Great Learning
POWER AHEAD

Sparkling Wine Data Project Report

DEEPTI AGRAWAL

Submission Date:
25th July 2021

CONTENTS

Sparkling Wine Dataset:

LIST OF FIGURES

Sparkling Wine Dataset:

EXECUTIVE SUMMARY

For this particular assignment, the data of different types of wine sales in the 20th century is to be analyzed. As an analyst in the ABC Estate Wines, we are tasked to analyze and forecast Wine Sales in the 20th century. Here, we are analyzing Sparkling Wine data.

INTRODUCTION

The purpose of this whole exercise is to forecast the Sales of Sparkling Wine in 20th century. We will do the Exploratory Data Analysis and will be making and analyzing multiple machine learning models and forecast the data based on the analysis.

Q 1 Read the data as an appropriate Time Series data and plot the data.

The data has been loaded correctly to a data frame as a time series.

A collection of observations that has been observed at regular time intervals for a certain variable over a given duration is called a time series**.**

Time series forecasting is applied to extract information from historical series and is used to predict future behavior of the same series based on past pattern.

Sample of the Dataset:

```
df.head()
```

	YearMonth	Sparkling
0	1980-01	1686
1	1980-02	1591
2	1980-03	2304
3	1980-04	1712
4	1980-05	1471

```
df.tail()
```

	YearMonth	Sparkling
182	1995-03	1897
183	1995-04	1862
184	1995-05	1670
185	1995-06	1688
186	1995-07	2031

Dataset shows the Sales of Sparkling Wine from a period of 1980 to year 1995.

Time_Stamp	Sparkling
1980-01-31	1686
1980-02-29	1591
1980-03-31	2304
1980-04-30	1712
1980-05-31	1471

We have created certain monthly time stamps in this period and added it as an index to our working dataset.

Plotting the time series

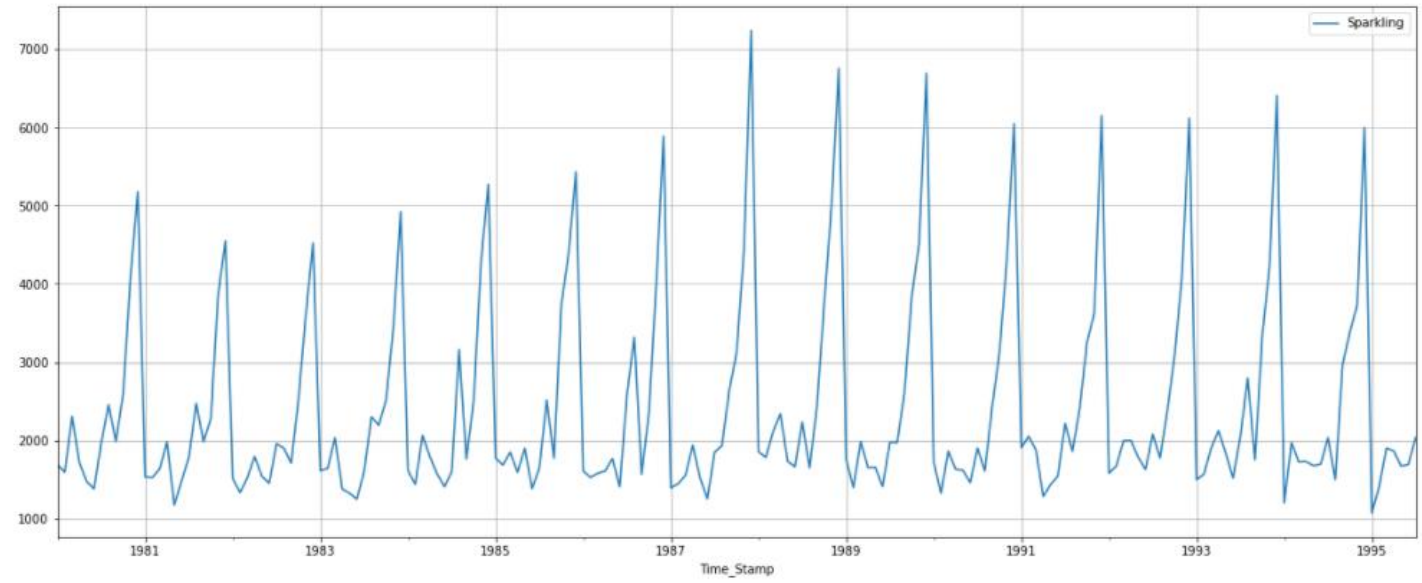


Fig 1: Sparkling Dataset Plot

The three important components of a time series are:

1. **Trend** (Long term movement): When the series increases (or decreases) over the entire length of time.
2. **Seasonal component**: Intra-year stable fluctuations repeatable over the entire length of the series. When a series is observed with more frequently than a year (quarterly or monthly for example), the series is subject to rhythmic fluctuations which are stable and repeatable each year.
3. **Irregular component** (Random movements). These fluctuations are purely random, erratic, unforeseen, and unpredictable. This is the random component of time series.

We can say that there might be a trend pattern but seasonality is definitely seen in the data.

Q 2 Perform appropriate Exploratory Data Analysis to understand the data and also perform decomposition.

Exploratory Data Analysis

We will be performing initial investigations on data so as to discover patterns, to spot anomalies and to check assumptions with the help of summary statistics and graphical representations.

Descriptive statistics

We will proceed with checking the descriptive statistics of the data which helps understand its features by giving short summaries about the sample and measures of data.

```
df.describe()
```

	Sparkling
count	187.000000
mean	2402.417112
std	1295.111540
min	1070.000000
25%	1605.000000
50%	1874.000000
75%	2549.000000
max	7242.000000

The basic measures of descriptive statistics tell us how the Sparkling Sales have varied across years. But for this measure of descriptive statistics we have averaged over the whole data without taking the time component into account.

From the above statistics, we can see that

- The mean Sales is 2402.4
- The maximum Sales recorded during this time period is 7242

Basic Info

Let us check the types of variables in the data frame.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 187 entries, 1980-01-31 to 1995-07-31
Data columns (total 1 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Sparkling  187 non-null    int64
dtypes: int64(1)
memory usage: 2.9 KB
```

The data has 187 entries over a time period of 1980-01-31 to 1995-07-31

Check for missing values

```
## Checking for missing value
df.isnull().sum()
```

```
Sparkling    0
dtype: int64
```

From the above result, we can see that there are no missing values present in our dataset.

The following steps are to be followed:

Part. I. Checking the pattern of the Wine sales series.
Part. II. Identification of the components of the series
Part. III. Proposing best model for the series

Yearly Boxplot

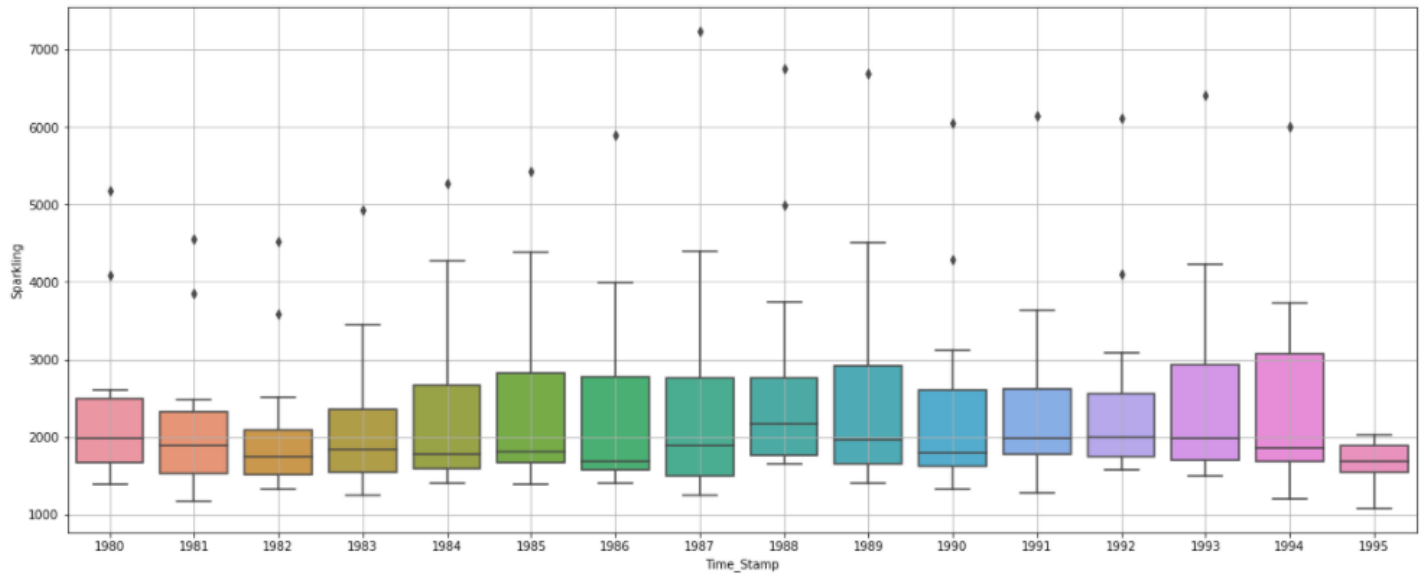


Fig 2: Yearly Boxplot Graph

This yearly graph shows that there was no drastic change observed in Sales in all the years from 1980 to 1994. But, 1995 recorded a steep drop in Sales.

Monthly Boxplot

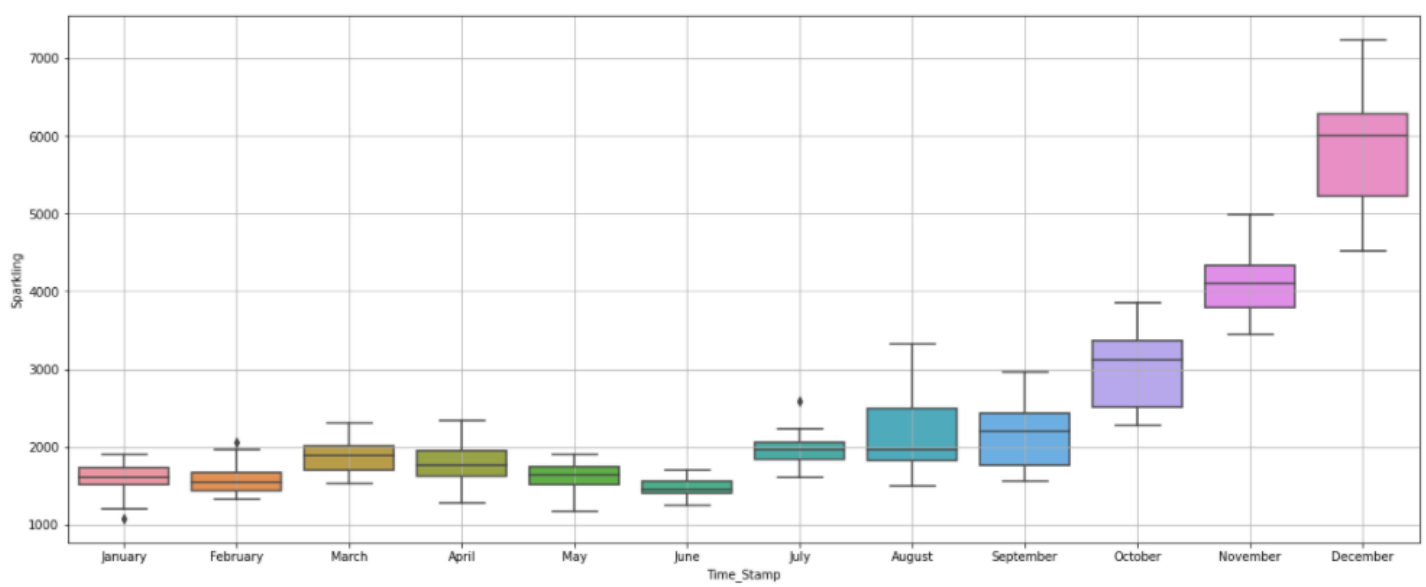


Fig 3: Monthly Boxplot Graph

The monthly graph shows that every year, last few month have recorded the highest Sales specially the month of December.

Time Series Month plot

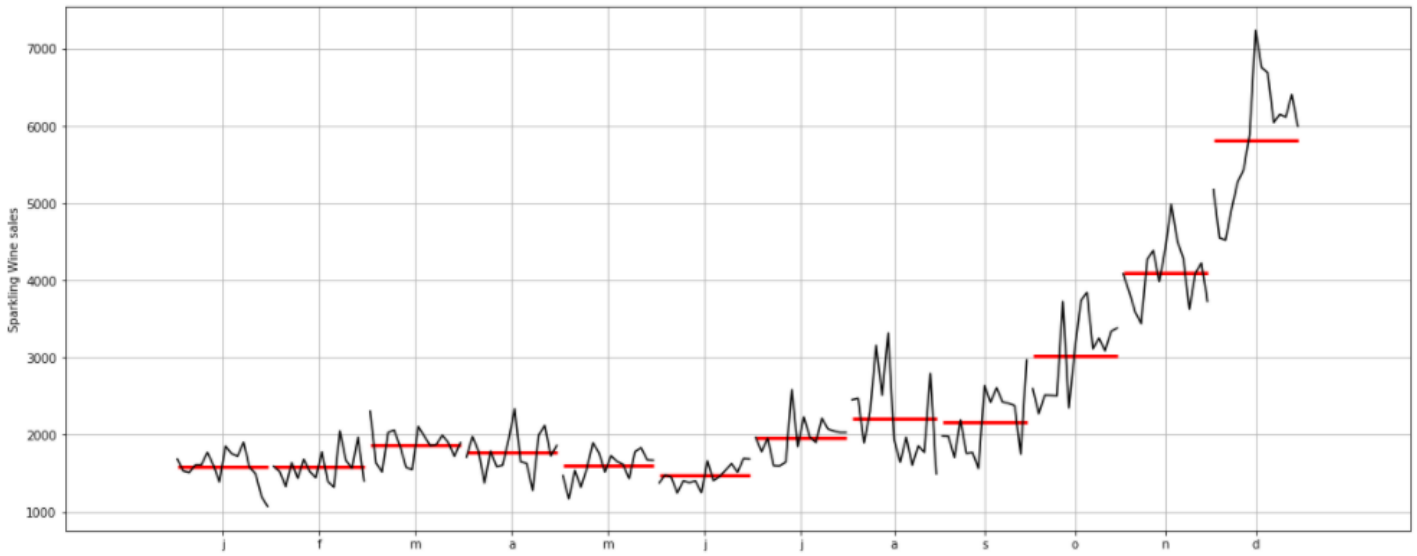


Fig 4: Time Series Monthly Plot

This plot shows us the behavior of the Time Series ('Sparkling' in this case) across various months. The red line is the median value.

Monthly Sales across Years

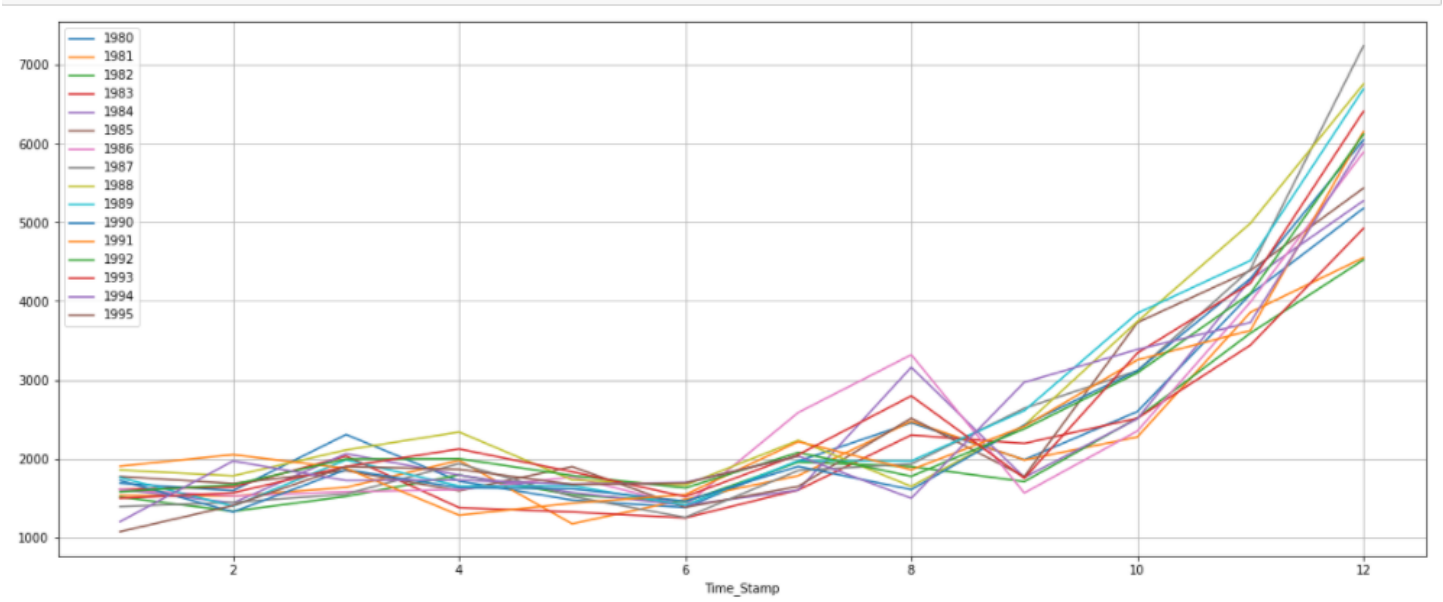


Fig 5: Yearly Sales across Months

This graph suggests that every year the Sales of Sparkling Wine increases towards the end of the year. All the years have almost similar kind of performance in terms of Sales of Sparkling Wine.

Empirical Distribution

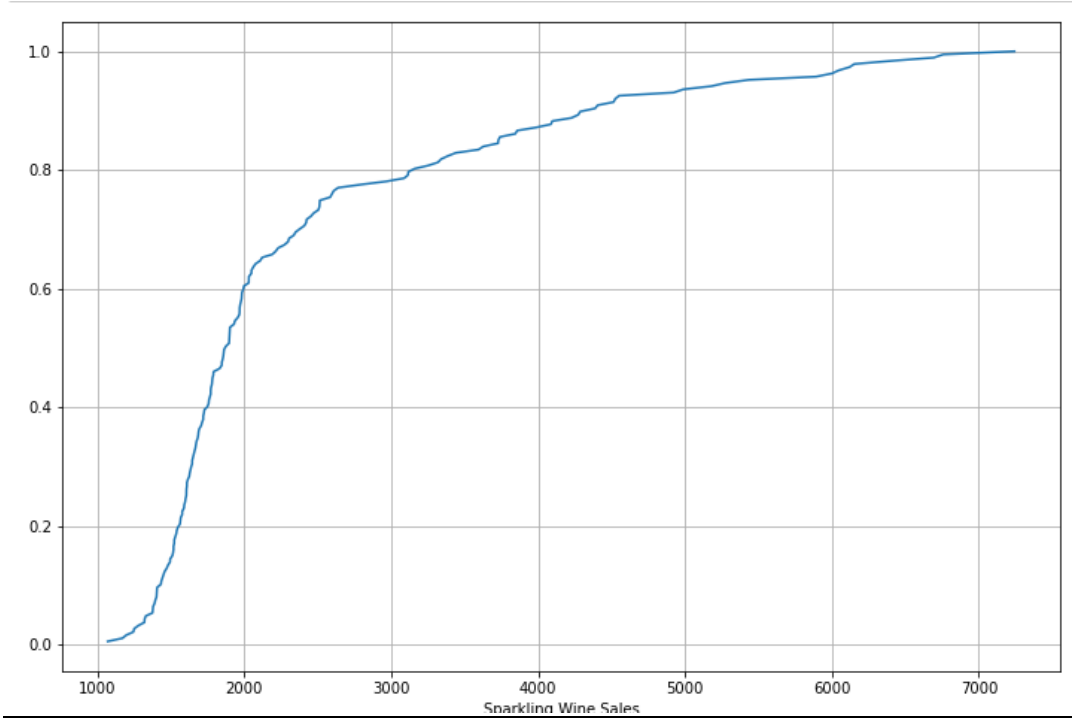


Fig 6: Empirical Distribution

This particular graph tells us what percentage of data points refer to what number of Sales. 80% of the sales are below 3000. Maximum sales is close to 7100

Average Monthly Sales & Month on Month Percent Change in Sales

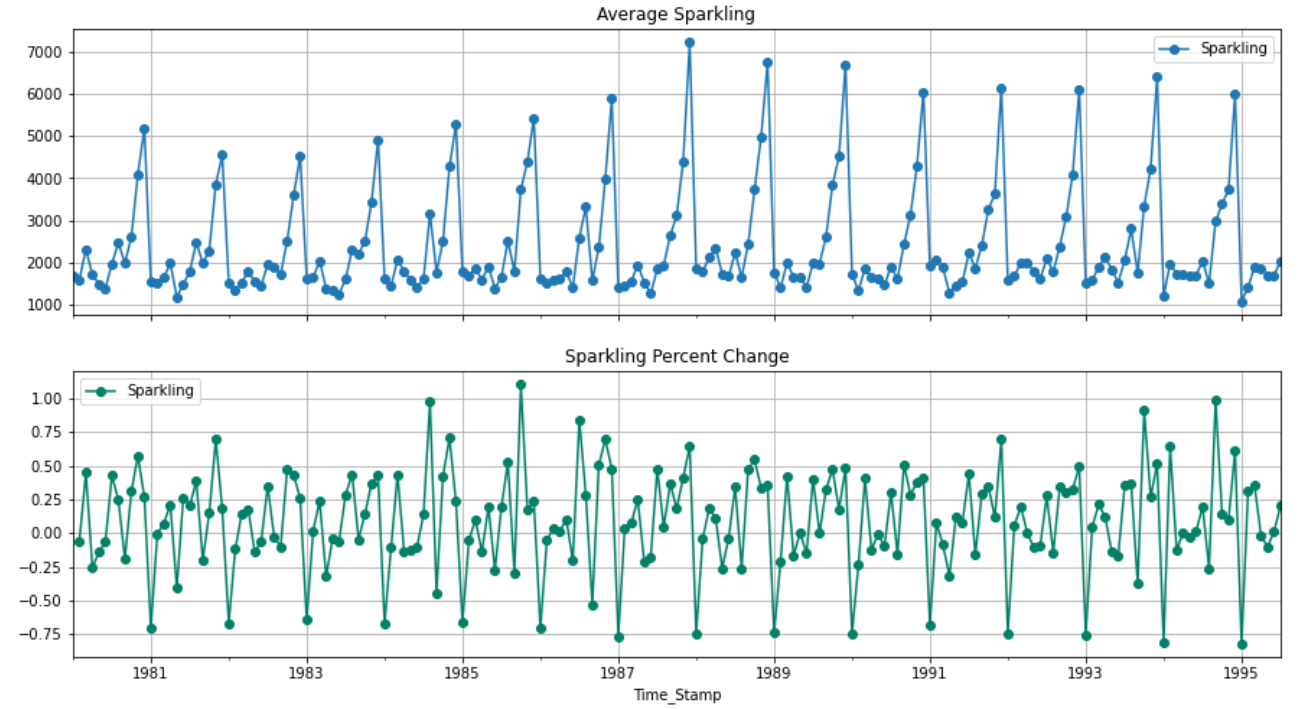


Fig 7: Average Sales and Monthly Percent Change in Sales

Decomposing the Time Series

Primary objective of decomposition is to study the components of the time series, NOT forecasting.

- Additive Model: Yt = Tt + St + It is considered when the resultant series is the sum of the components.
- Multiplicative Model: Yt = Tt * St * It is considered when the resultant time series is the product of the components.

A series may be considered multiplicative series when the seasonal fluctuations increase as trend increases.

1. Additive Decomposition

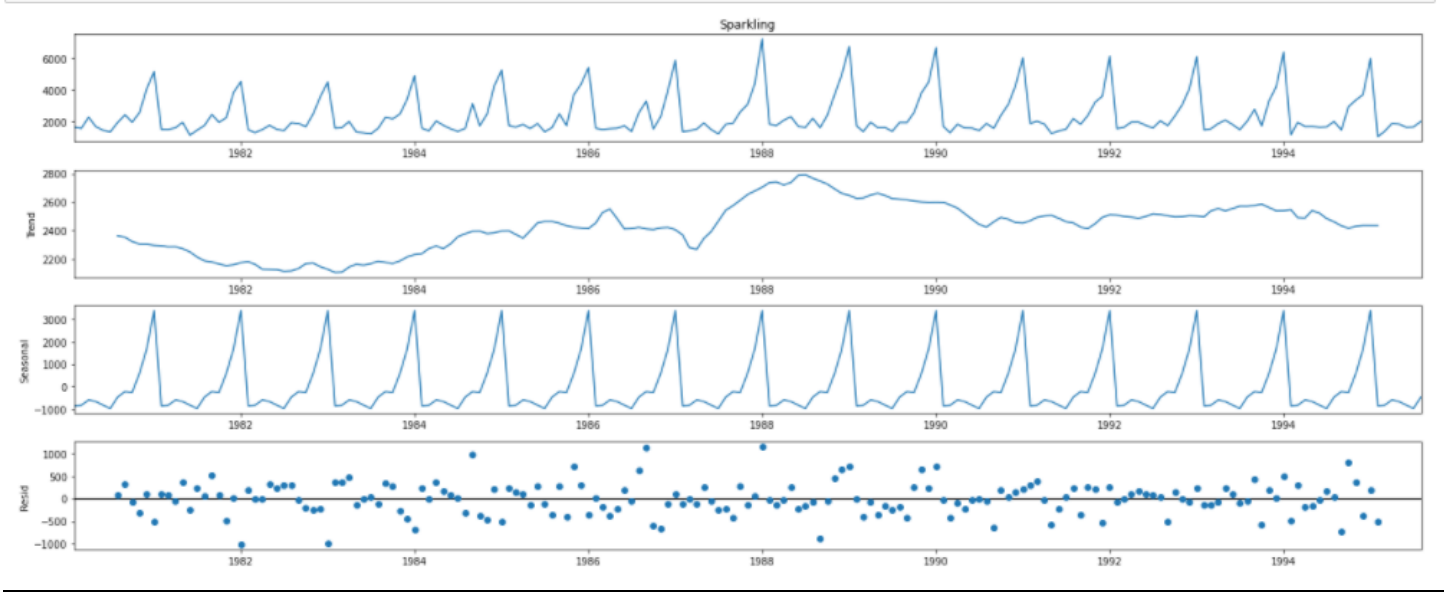


Fig 8: Additive Decomposition

2. Multiplicative Decomposition

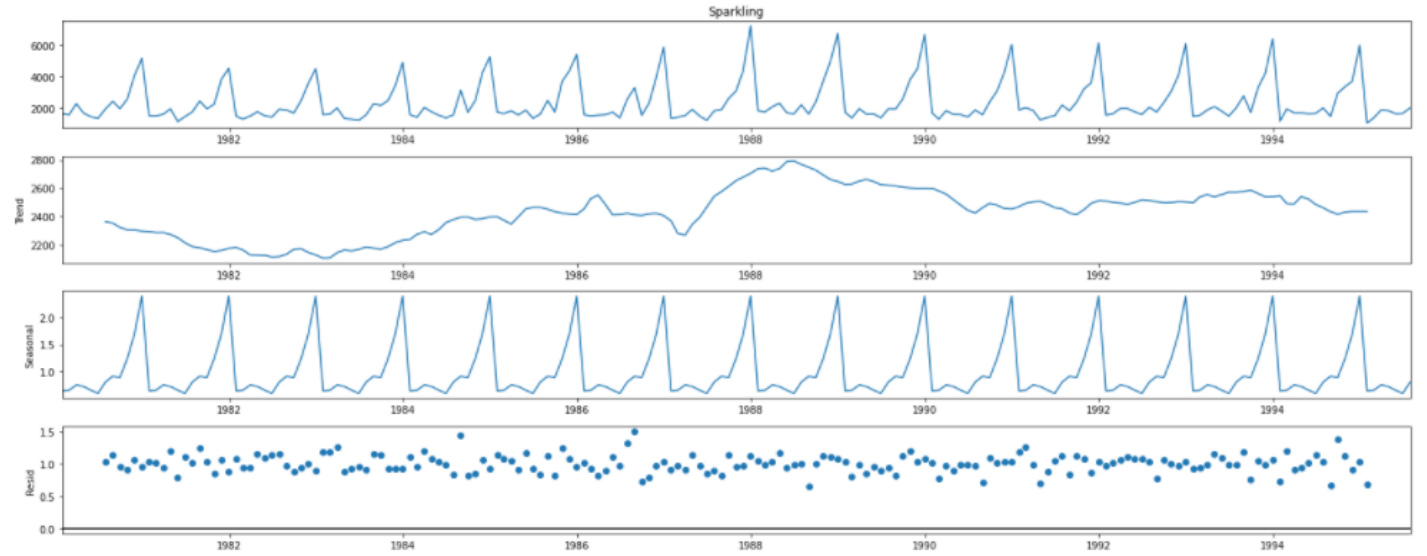


Fig 9: Multiplicative Decomposition

Auto Correlation Plots

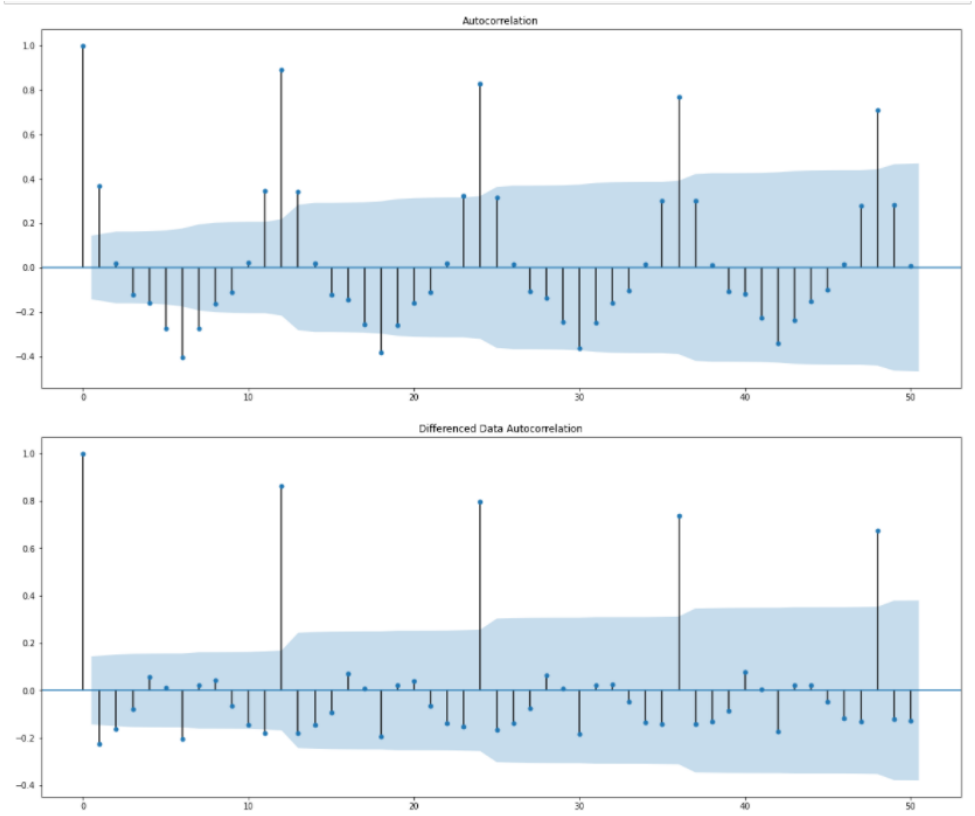


Fig 10: Auto Correlation Plots

Partial Auto Correlation Plots

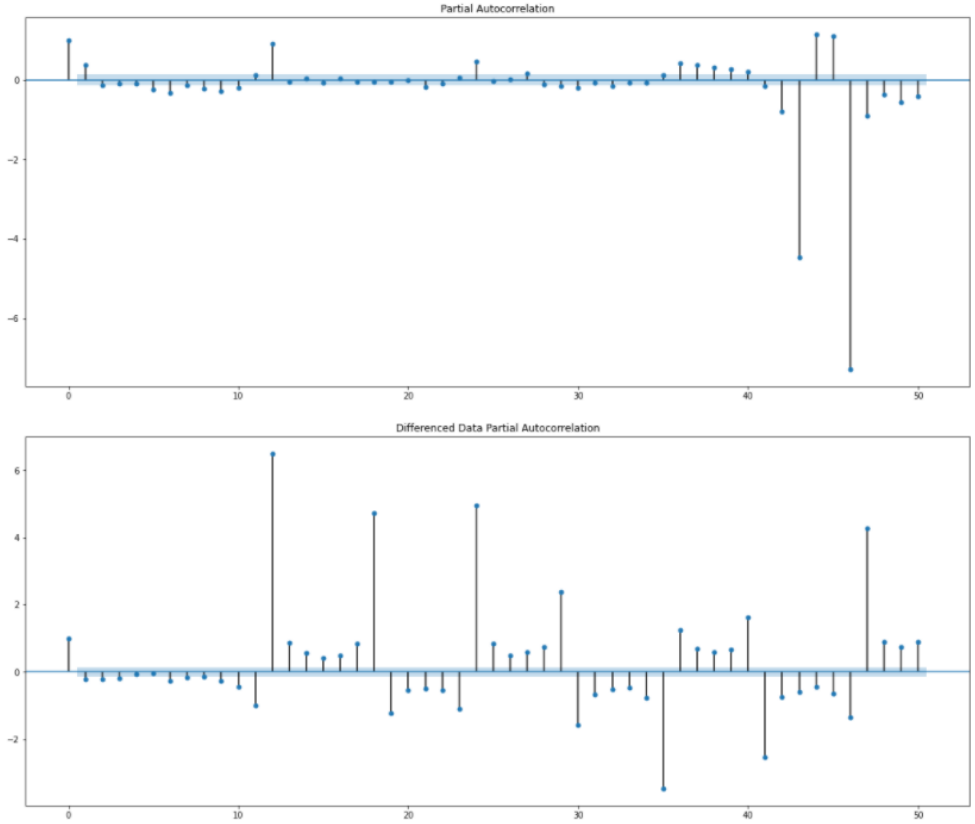


Fig 11: Partial Auto Correlation Plots

Inference from basic time series analysis

- Data values are stored in correct time order and no data is missing.
- The sales shows the same pattern in all years.
- Intra-year stable fluctuations are indicative of seasonal component.
- The highest Sales in each year is observed in the last few months.

Q 3. Split the data into training and test. The test data should start in 1991.

The train-test split is a technique for evaluating the performance of a machine learning algorithm. It can be used for classification or regression problems and can be used for any supervised learning algorithm.

Before a forecast method is proposed, the method needs to be validated. For that purpose, data has to be split into two sets i.e. training and testing. Training data helps in identifying and fitting right model(s) and test data is used to validate the same.

In case of time series data, the test data is the most recent part of the series so that the ordering in the data is preserved.

```
train=df[df.index.year < 1991]
test=df[df.index.year >= 1991]
```

We have split the data in such a way that train data consists of data from 1980 to 1990 and test data consists of data from 1991 to 1995

Training data (contains 132 observations)

First few rows of Training Data

Time_Stamp	Sparkling
1980-01-31	1686
1980-02-29	1591
1980-03-31	2304
1980-04-30	1712
1980-05-31	1471

Last few rows of Training Data

Time_Stamp	Sparkling
1990-08-31	1605
1990-09-30	2424
1990-10-31	3116
1990-11-30	4286
1990-12-31	6047

Testing data (contains 55 observations)

First few rows of Test Data

Time_Stamp	Sparkling
1991-01-31	1902
1991-02-28	2049
1991-03-31	1874
1991-04-30	1279
1991-05-31	1432

Last few rows of Test Data

Time_Stamp	Sparkling
1995-03-31	1897
1995-04-30	1862
1995-05-31	1670
1995-06-30	1688
1995-07-31	2031

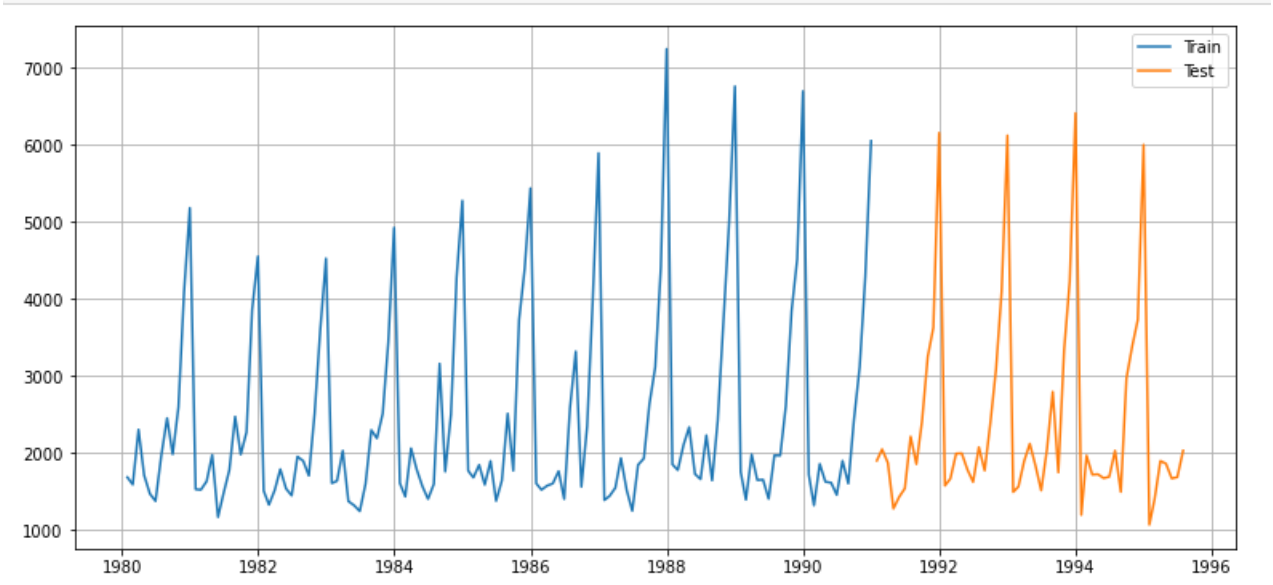


Fig 12: Train/Test Split

Q 4. Build various exponential smoothing models on the training data and evaluate the model using RMSE on the test data.

Other models such as regression, naïve forecast models, simple average models etc. should also be built on the training data and check the performance on the test data using RMSE.

Now, on the previously splitted training and testing data we will be used in building different models.

Forecasting accuracy measures compare the predicted values against the observed values to quantify the predictive power of the proposed model. Mathematically, it can be defined as

Forecast error et for period t is given by: et = Ŷt - Yt

Where

Ŷt = forecast value for time period t

Yt = actual value in time period t

n = No. of observations

Various measures of forecasting errors can be defined as: -

Mean Absolute Deviation (MAD):

$$\frac{1}{n}\sum_{t=1}^{n}|e_t|$$

Mean Absolute Percentage Error (MAPE): This method is used extensively in time series because it acts as a unit free criteria; therefore, performance of forecasted values can be easily compared.

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|e_t|}{Y_t} * 100$$

MAPE is usually expressed as a percentage.

Mean Square Error (MSE):

$$MSE = \frac{1}{n}\sum_{t=1}^{n} e_t^2$$

Root Mean Square Error (RMSE):

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$$

Here, we will be evaluating our models by evaluation RMSE value on the test data.

The mean squared error, or MSE, is calculated as the average of the squared forecast error values. Squaring the forecast error values forces them to be positive; it also has the effect of putting more weight on large errors.

Very large or outlier forecast errors are squared, which in turn has the effect of dragging the mean of the squared forecast errors out resulting in a larger mean squared error score. In effect, the score gives worse performance to those models that make large wrong forecasts.

The mean squared error described above is in the squared units of the predictions.

It can be transformed back into the original units of the predictions by taking the square root of the mean squared error score. This is called the root mean squared error, or **RMSE.**

1. LINEAR REGRESSION MODEL

The regression model allows for a linear relationship between the forecast variable y and a single predictor variable x:

$$y_t = \beta_0 + \beta_1 x_t + \varepsilon_t.$$

The coefficients β0 and β1 denote the intercept and the slope of the line respectively. The intercept β0 represents the predicted value of y when x=0. The slope β1 represents the average predicted change in y resulting from a one unit increase in x.

For this particular linear regression, we are going to regress the 'Sparkling Sales' variable against the order of the occurrence. For this we need to modify our training data before fitting it into a linear regression.

```
Training Time instance
 [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 3
4, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65,
66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97,
98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123,
124, 125, 126, 127, 128, 129, 130, 131, 132]
Test Time instance
 [43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 7
4, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97]
```

We see that we have successfully the generated the numerical time instance order for both the training and test set. Now we will add these values in the training and test set.

First few rows of Training Data

Time_Stamp	Sparkling	time
1980-01-31	1686	1
1980-02-29	1591	2
1980-03-31	2304	3
1980-04-30	1712	4
1980-05-31	1471	5

Last few rows of Training Data

Time_Stamp	Sparkling	time
1990-08-31	1605	128
1990-09-30	2424	129
1990-10-31	3116	130
1990-11-30	4286	131
1990-12-31	6047	132

First few rows of Test Data

Time_Stamp	Sparkling	time
1991-01-31	1902	43
1991-02-28	2049	44
1991-03-31	1874	45
1991-04-30	1279	46
1991-05-31	1432	47

Last few rows of Test Data

Time_Stamp	Sparkling	time
1995-03-31	1897	93
1995-04-30	1862	94
1995-05-31	1670	95
1995-06-30	1688	96
1995-07-31	2031	97

This is how the train and test data looks like.

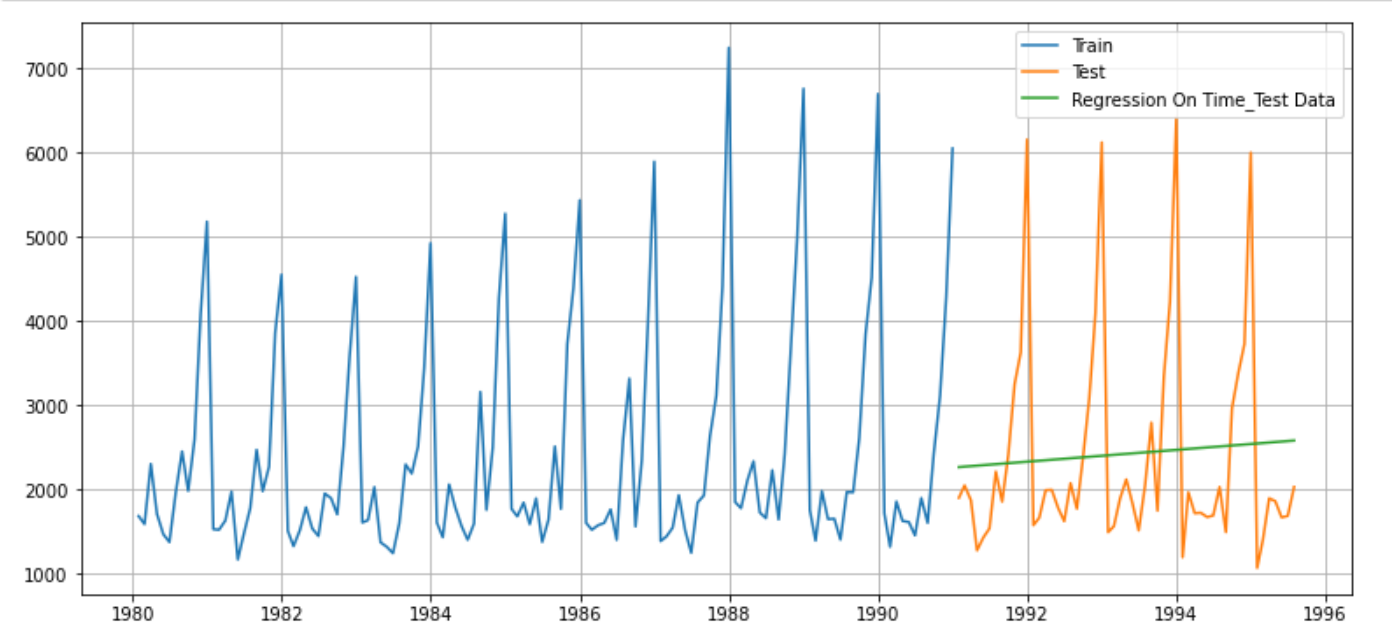


Fig 13: Linear Regression Model Forecast

Model Evaluation:

```
For RegressionOnTime forecast on the Test Data,  RMSE is 1275.867
```

```
resultsDf = pd.DataFrame({'Test RMSE': [rmse_model1_test]},index=['RegressionOnTime'])
resultsDf
```

	Test RMSE
RegressionOnTime	1275.867052

2. NAÏVE MODEL

For this particular naive model, we say that the prediction for tomorrow is the same as today and the prediction for day after tomorrow is tomorrow and since the prediction of tomorrow is same as today therefore the prediction for day after tomorrow is also today.

For naïve forecasts, we simply set all forecasts to be the value of the last observation. When our series is seasonal, we set each forecast to be equal to the last observed value from the same season of the year (e.g., the same month of the previous year). Formally, the forecast for time T + h is written as

$$\hat{y}_{T+h|T} = y_{T+h-m(k+1)},$$

Where m= the seasonal period, and k is the integer part of (h−1)/m (h−1)/m (i.e., the number of complete years in the forecast period prior to time T+ h).

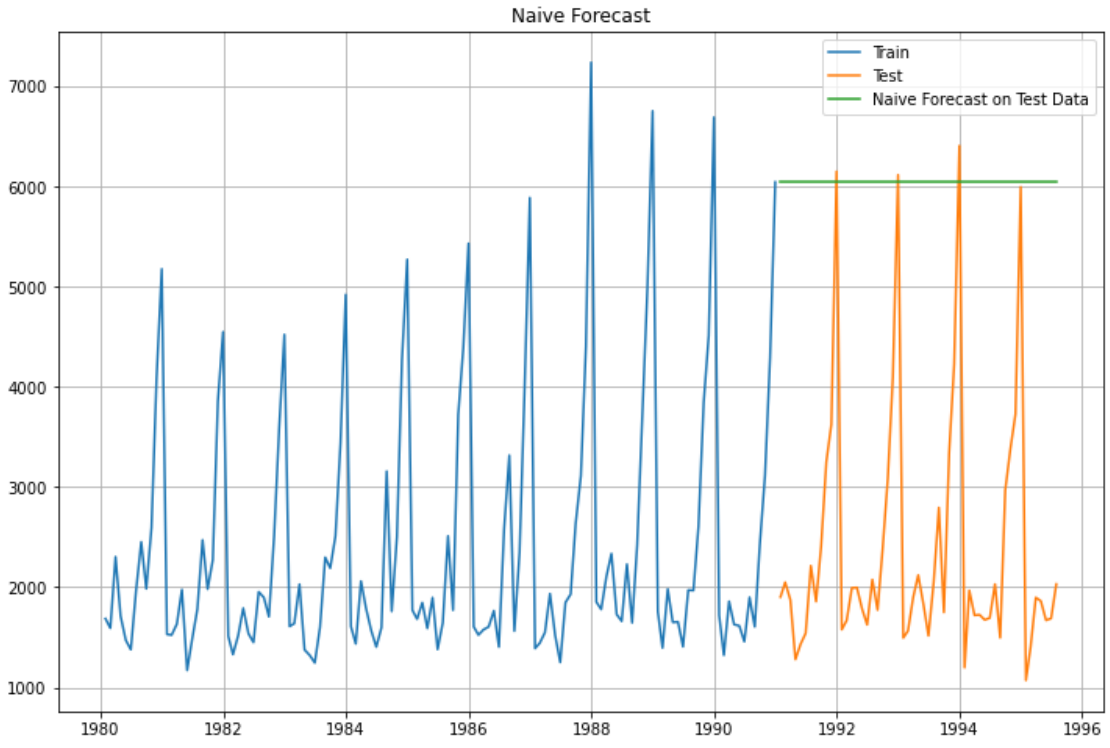


Fig 14: Naive Model Forecast

Model Evaluation:

```
For RegressionOnTime forecast on the Test Data,  RMSE is 3864.279
```

```
resultsDf_2 = pd.DataFrame({'Test RMSE': [rmse_model2_test]},index=['NaiveModel'])

resultsDf = pd.concat([resultsDf, resultsDf_2])
resultsDf
```

	Test RMSE
RegressionOnTime	1275.867052
NaiveModel	3864.279352

Naïve Model shows higher RMSE value than the Linear Regression Model.

3. SIMPLE AVERAGE MODEL

This method is used when the time series variable consists of only the seasonal and random components. The effect of taking average of data corresponding to the same period (say first quarter of each year) is to eliminate the effect of random component and thus, the resulting averages consist of only seasonal component. These averages are then converted into seasonal indices.

This is a simplest method of measuring seasonal variations. However this method is based on the unrealistic assumption that the trend and cyclical variations are absent from the data.

For this particular simple average method, we will forecast by using the average of the training values.

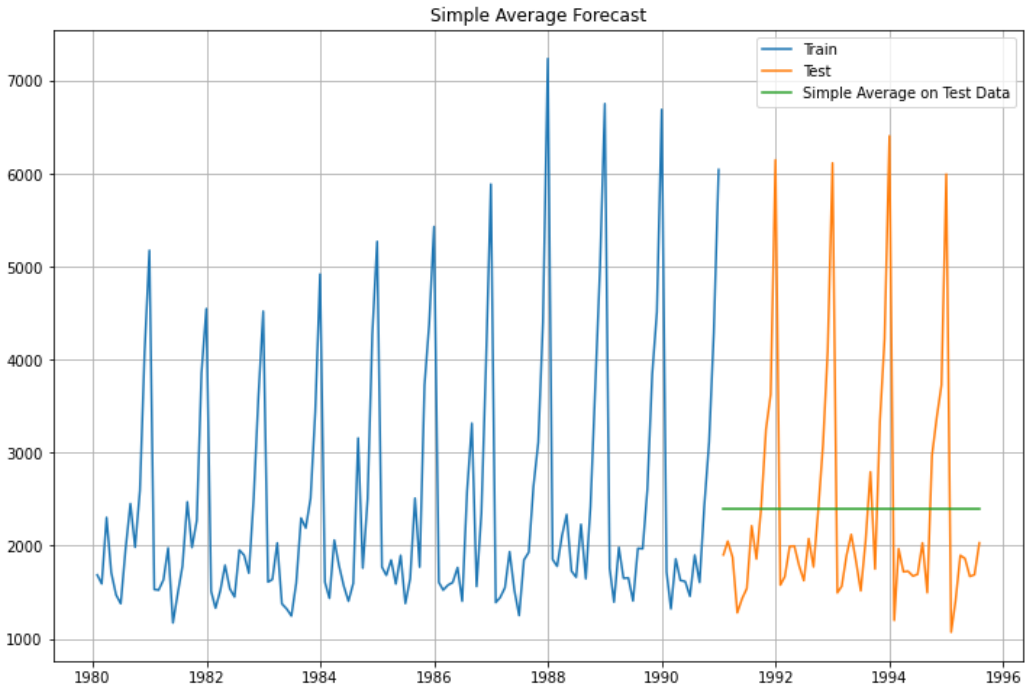


Fig 15: Simple Average Model Forecast

```
For Simple Average forecast on the Test Data,  RMSE is 1275.082
```

```
resultsDf_3 = pd.DataFrame({'Test RMSE': [rmse_model3_test]},index=['SimpleAverageModel'])

resultsDf = pd.concat([resultsDf, resultsDf_3])
resultsDf
```

	Test RMSE
RegressionOnTime	1275.867052
NaiveModel	3864.279352
SimpleAverageModel	1275.081804

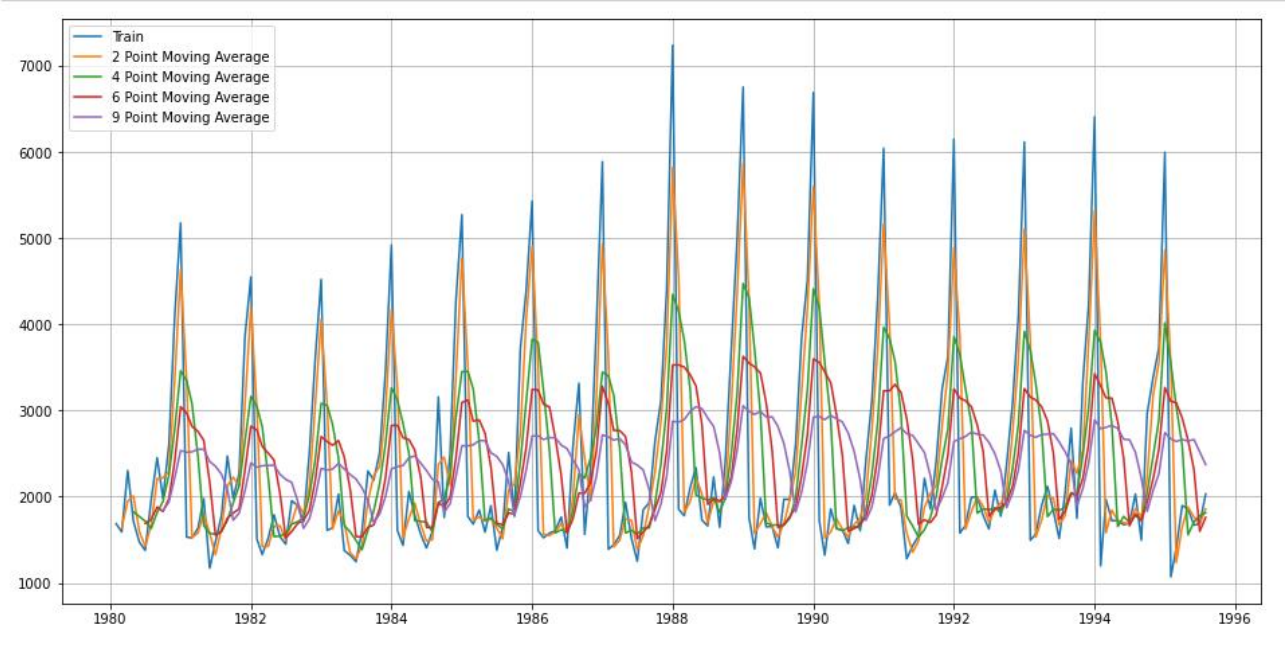
4. MOVING AVERAGE MODEL

For the moving average model, we are going to calculate rolling means (or moving averages) for different intervals. The best interval can be determined by the maximum accuracy (or the minimum error) over here. We are going to average over the entire data.

This method is based on the principle that the total effect of periodic variations at different points of time in its cycle gets completely neutralized, i.e. ∑St = 0 in one year and ∑Ct = 0 in the periods of cyclical variations.

Time_Stamp	Sparkling	Trailing_2	Trailing_4	Trailing_6	Trailing_9
1980-01-31	1686	NaN	NaN	NaN	NaN
1980-02-29	1591	1638.5	NaN	NaN	NaN
1980-03-31	2304	1947.5	NaN	NaN	NaN
1980-04-30	1712	2008.0	1823.25	NaN	NaN
1980-05-31	1471	1591.5	1769.50	NaN	NaN

This data is calculated over the original dataset.



We will now split the data into train and test and plot this Time Series. The window of the moving average is need to be carefully selected as too big a window will result in not having any test set as the whole series might get averaged over.

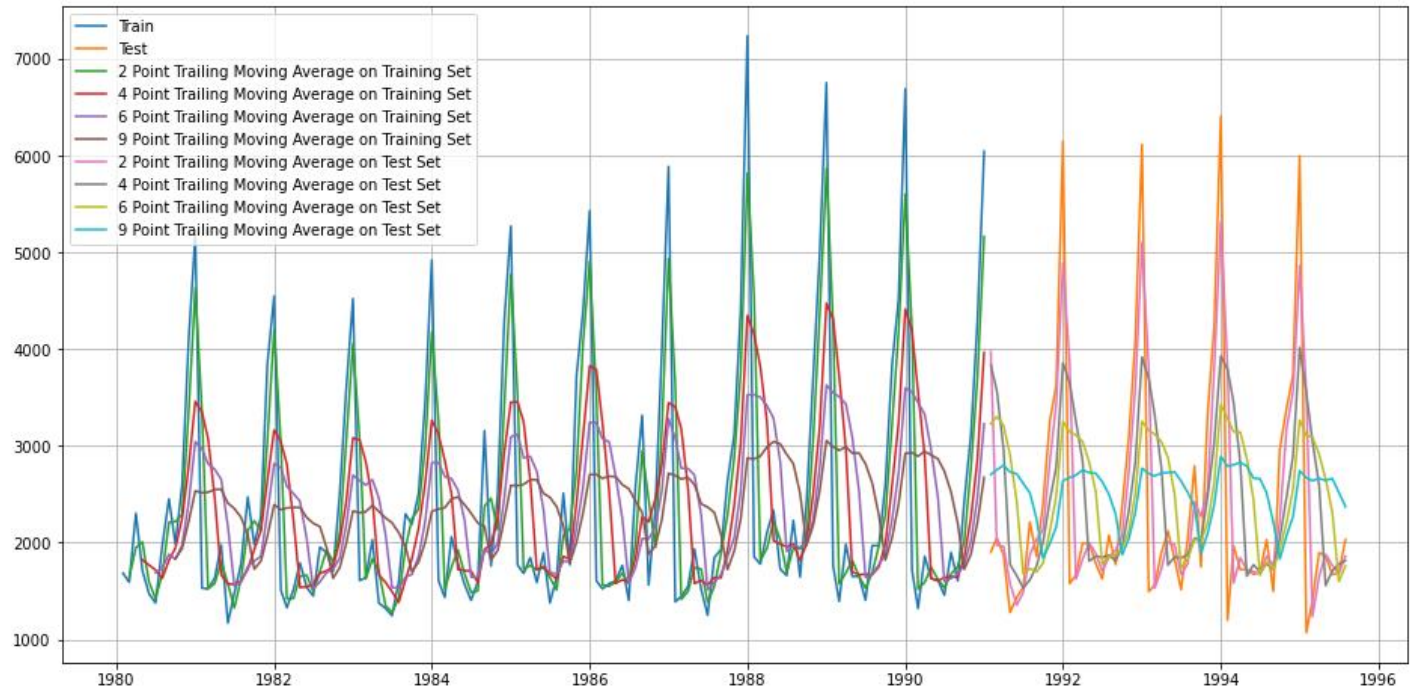


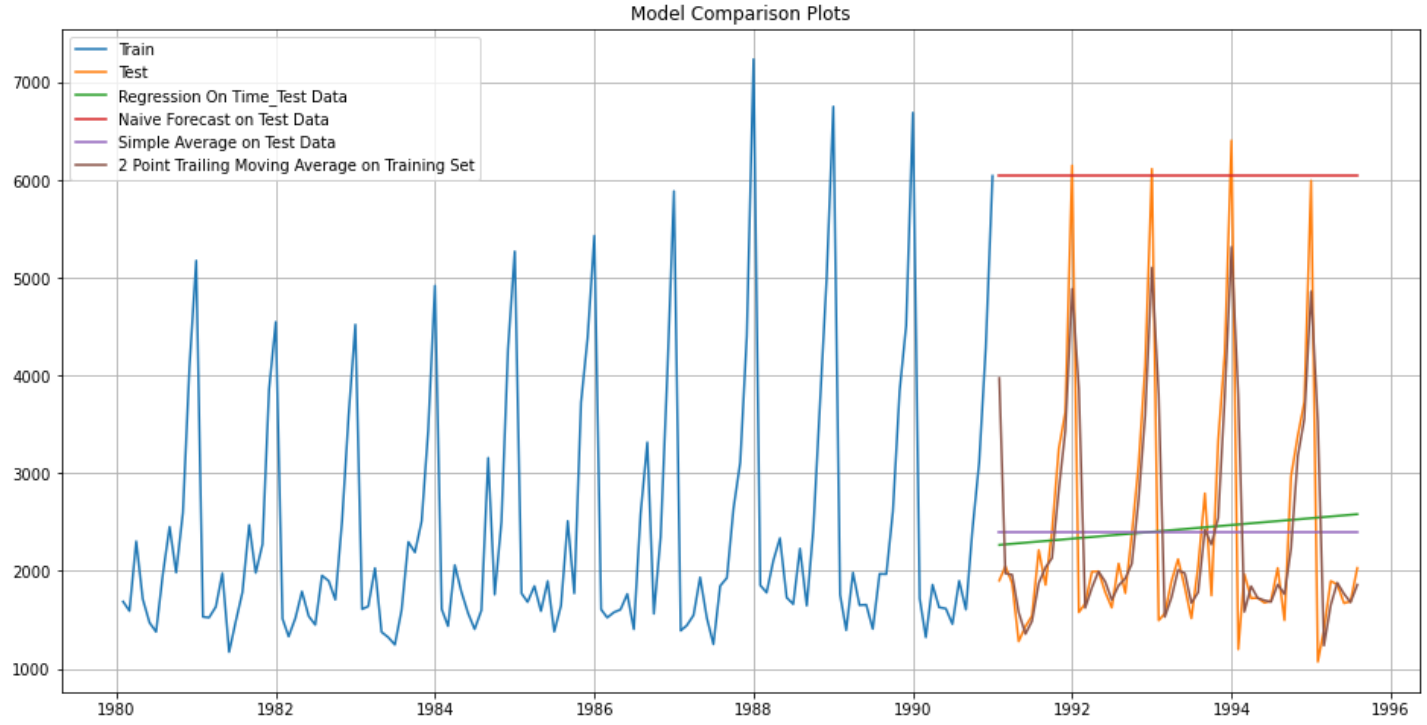
Fig 16: Moving Average Model Forecast

```
For 2 point Moving Average Model forecast on the Training Data,  RMSE is 813.401
For 4 point Moving Average Model forecast on the Training Data,  RMSE is 1156.590
For 6 point Moving Average Model forecast on the Training Data,  RMSE is 1283.927
For 9 point Moving Average Model forecast on the Training Data,  RMSE is 1346.278
```

	Test RMSE
RegressionOnTime	1275.867052
NaiveModel	3864.279352
SimpleAverageModel	1275.081804
2pointTrailingMovingAverage	813.400684
4pointTrailingMovingAverage	1156.589694
6pointTrailingMovingAverage	1283.927428
9pointTrailingMovingAverage	1346.278315

Till now, the best RMSE score has been observed in 2 point trailing moving average model.

Model Comparison of all the model built until now.



Here, 2 point trailing moving average model is predicting closest on the test set with the best RMSE score.

Now, we will go forward with building smoothing models.

5. SIMPLE EXPONENTIAL SMOOTHING MODEL

This is an extension of moving (rolling) average method where more recent observations get higher weight.

SES or one-parameter exponential smoothing is applicable to time series which do not contain either of trend or seasonality. Forecast by SES is given by:

$$\hat{Y}t+1 = \alpha Yt+\alpha(1-\alpha)t-1+\alpha(1-\alpha)2Yt-1+\cdots,\ 0<\alpha<1$$

Where, α is the smoothing parameter for the level. In reality such a series is hard to find. This is a one step-ahead forecast where all the forecast values are identical.

```
model_SES_autofit.params
```

```
{'smoothing_level': 0.049607360581862936,
 'smoothing_trend': nan,
 'smoothing_seasonal': nan,
 'damping_trend': nan,
 'initial_level': 1818.535750008871,
 'initial_trend': nan,
 'initial_seasons': array([], dtype=float64),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

The auto model suggests value of alpha is 0.0496 for which we will predict on our test set.

Time_Stamp	Sparkling	predict
1991-01-31	1902	2724.93
1991-02-28	2049	2724.93
1991-03-31	1874	2724.93
1991-04-30	1279	2724.93
1991-05-31	1432	2724.93

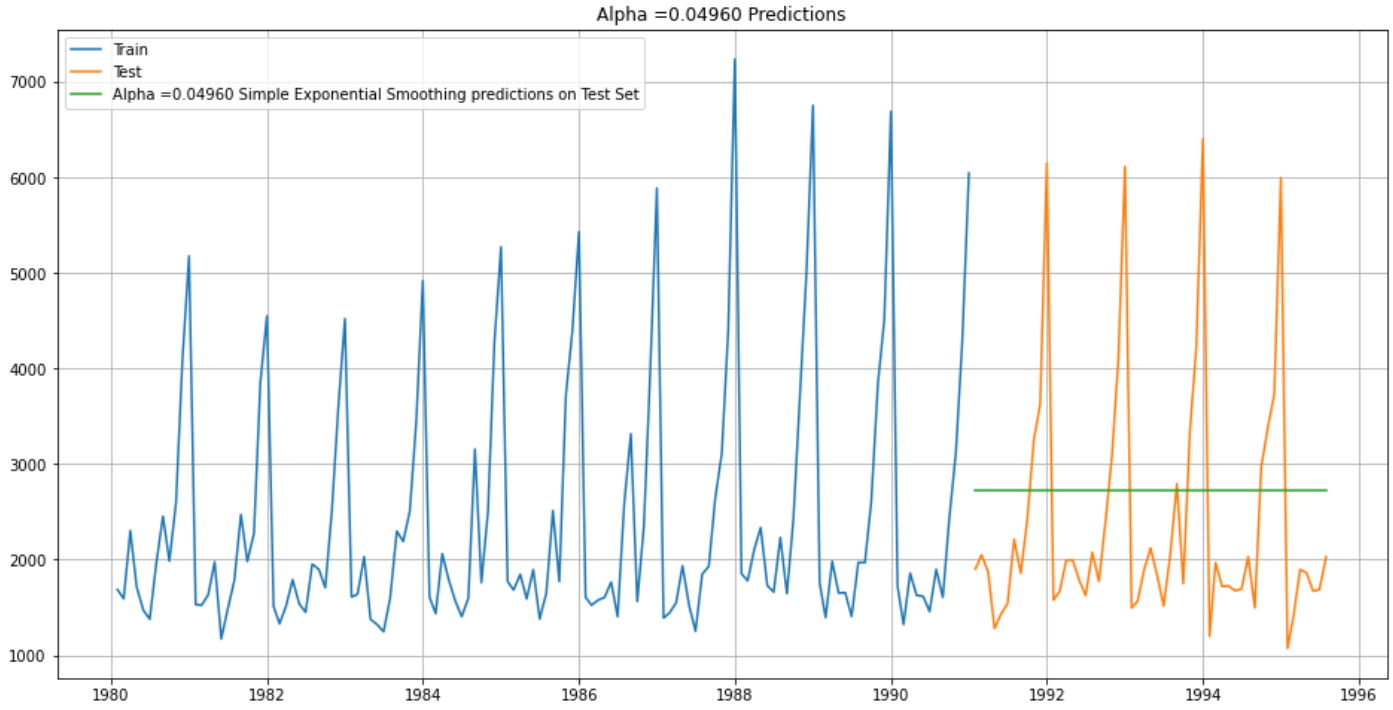


Fig 17: alpha=0.04960 Simple Exponential Smoothing Model Forecast

Model Evaluation:

```
For Alpha =0.04960 Simple Exponential Smoothing Model forecast on the Test Data, RMSE is 1316.035
```

```
resultsDf_5 = pd.DataFrame({'Test RMSE': [rmse_model5_test_1]},index=['Alpha=0.04960,SimpleExponentialSmoothing'])

resultsDf = pd.concat([resultsDf, resultsDf_5])
resultsDf
```

	Test RMSE
RegressionOnTime	1275.87
NaiveModel	3864.28
SimpleAverageModel	1275.08
2pointTrailingMovingAverage	813.40
4pointTrailingMovingAverage	1156.59
6pointTrailingMovingAverage	1283.93
9pointTrailingMovingAverage	1346.28
Alpha=0.04960,SimpleExponentialSmoothing	1316.04

The higher the alpha value more weightage is given to the more recent observation. That means, what happened recently will happen again. We will run a loop with different alpha values to understand which particular value works best for alpha on the test set.

Model Evaluation

```
resultsDf_6.sort_values(by=['Test RMSE'],ascending=True)
```

	Alpha Values	Train RMSE	Test RMSE
1	0.02	1328.41	1279.50
0	0.01	1362.00	1281.03
2	0.03	1318.85	1293.11
3	0.04	1317.14	1305.46
4	0.05	1318.43	1316.41
...	...	...	...
94	0.95	1363.59	3778.43
95	0.96	1365.35	3796.05
96	0.97	1367.18	3813.44
97	0.98	1369.08	3830.60
98	0.99	1371.04	3847.55

99 rows × 3 columns

Here, we can observe that alpha =0.02 works best for predicting our series.

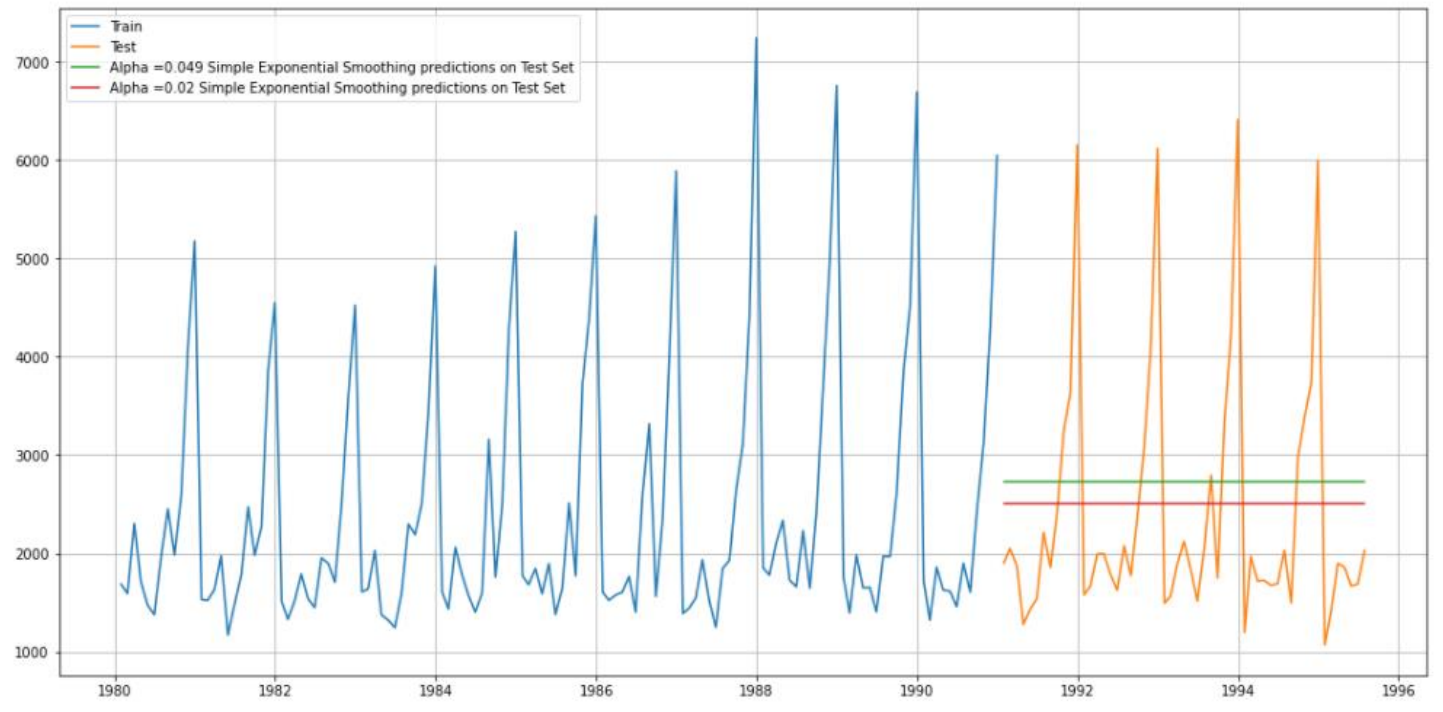


Fig 18: alpha=0.02 Simple Exponential Smoothing Model Forecast

6. DOUBLE EXPONENTIAL SMOOTHING MODEL

This method is an extension of SES method, proposed by Holt in 1957. This method is applicable where trend is present in the data but no seasonality.

The forecast values are given as:

Forecast equation: $\hat{Y}t+1=lt+bt$

Level Equation: $lt=\alpha Yt+(1-\alpha)Yt-1$, $0 < \alpha < 1$

Trend Equation: $bt=(lt-lt-1)+(1-\beta)bt-1$, $0 < \beta < 1$

Where, lt is the estimate of level and bt is the trend estimate.

$\alpha$is the smoothing parameter for the level and β is the smoothing parameter for trend which are estimated in this model.

	Alpha Values	Beta Values	Train RMSE	Test RMSE
148	0.02	0.50	1414.59	1274.63
115	0.02	0.17	1488.67	1275.11
254	0.03	0.57	1438.85	1276.03
255	0.03	0.58	1441.36	1278.43
253	0.03	0.56	1436.27	1278.59

Here, we can observe that alpha =0.02 and beta= 0.50 works best for predicting our series.

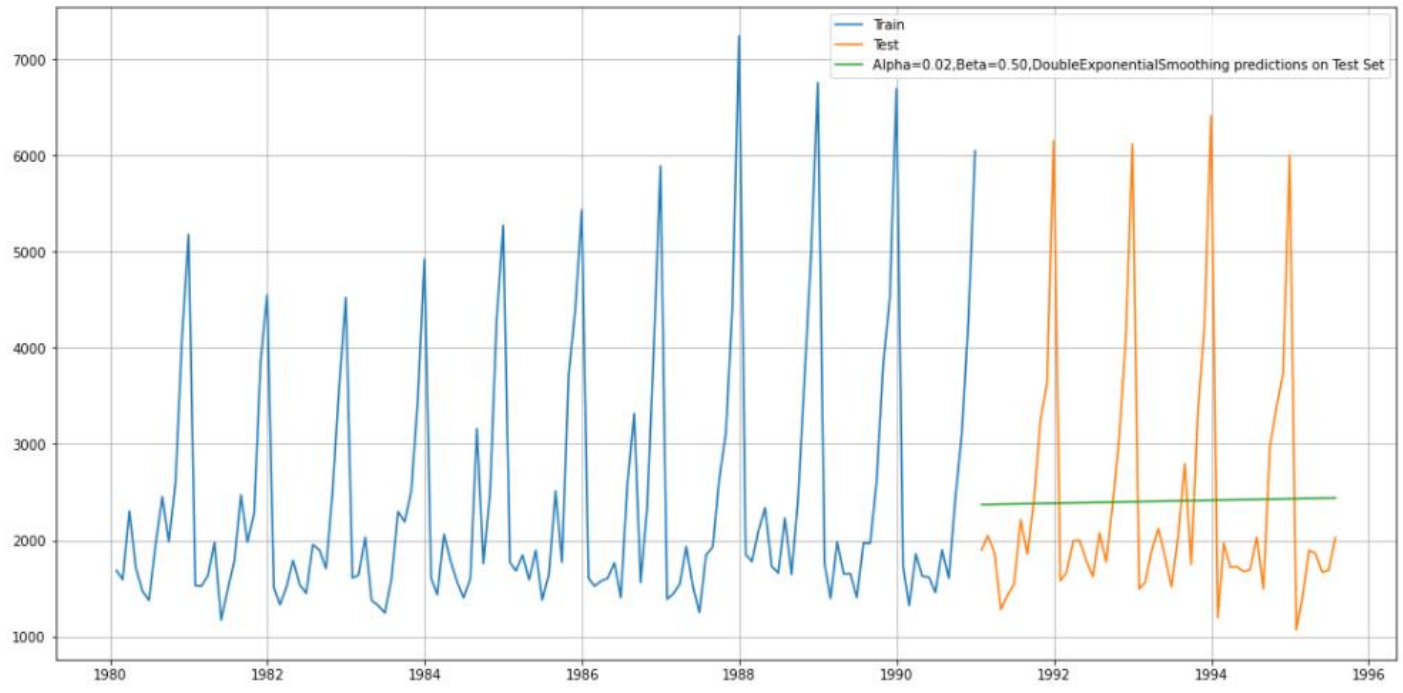


Fig 19: alpha=0.02 and beta=0.50 Double Exponential Smoothing Model Forecast

7. TRIPLE EXPONENTIAL SMOOTHING MODEL

This is an extension of Holt's method when seasonality is found in the data.

Forecast equation: $Yt+1=lt+bt+st-(k+1)$

Level Equation: $lt=(Yt-st-m)+\alpha(1-\alpha)Yt-1$, $0 < \alpha < 1$

Trend Equation: $bt=(lt-lt-1)+(1-\beta)bt-1$, $0 < \beta < 1$

Seasonal Equation: $(Yt-lt-1-bt-1)+(1-\gamma)st-m$, $0 < \gamma < 1$

This is also known as three parameters exponential or triple exponential because of the three smoothing parameters α, β and γ. This is a general method and a true multi-step ahead forecast.

```
model_TES_autofit.params
```

```
{'smoothing_level': 0.11107180566178311,
 'smoothing_trend': 0.06170656134860671,
 'smoothing_seasonal': 0.3950794485810031,
 'damping_trend': nan,
 'initial_level': 1640.0003926139427,
 'initial_trend': -15.114515968081134,
 'initial_seasons': array([1.03196652, 0.98906977, 1.40589009, 1.20112018, 0.93876755,
        0.95192914, 1.29600207, 1.68005327, 1.35640448, 1.79372962,
        2.82508328, 3.59807574]),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

The auto model suggests value of alpha=0.0111, beta=0.0617 and gamma=0.3950 for which we will predict on our test set.

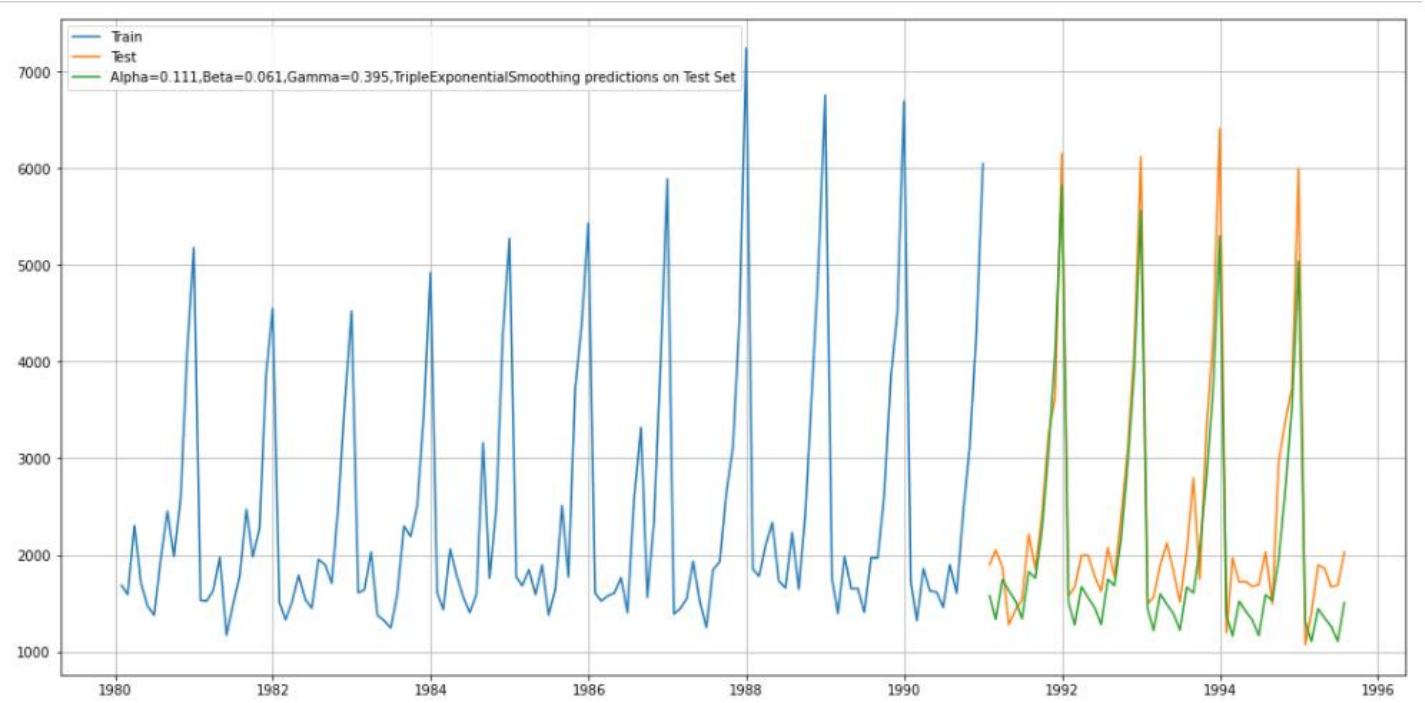


Fig 20: alpha=0.0111, beta=0.0617 and gamma=0.395 Triple Exponential Smoothing Model Forecast

```
## Test Data

rmse_model6_test_1 = metrics.mean_squared_error(TES_test['Sparkling'],TES_test['auto_predict'],squared=False)
print("For Alpha=0.111,Beta=0.061,Gamma=0.395, Triple Exponential Smoothing Model forecast on the Test Data,  RMSE is %3.3f
```

```
For Alpha=0.111,Beta=0.061,Gamma=0.395, Triple Exponential Smoothing Model forecast on the Test Data,  RMSE is 469.593
```

This is so far the best RMSE value. Also from the previous prediction graph we can see that the prediction is very close.

	Alpha Values	Beta Values	Gamma Values	Train RMSE	Test RMSE
83	0.03	0.02	0.06	576.58	370.58
119	0.04	0.02	0.06	506.23	372.31
191	0.06	0.02	0.06	445.00	372.95
155	0.05	0.02	0.06	467.71	373.10
203	0.06	0.04	0.06	406.59	377.04

Here, we can observe that alpha =0.03, beta=0.02 and gamma= 0.06 works best for predicting our series.

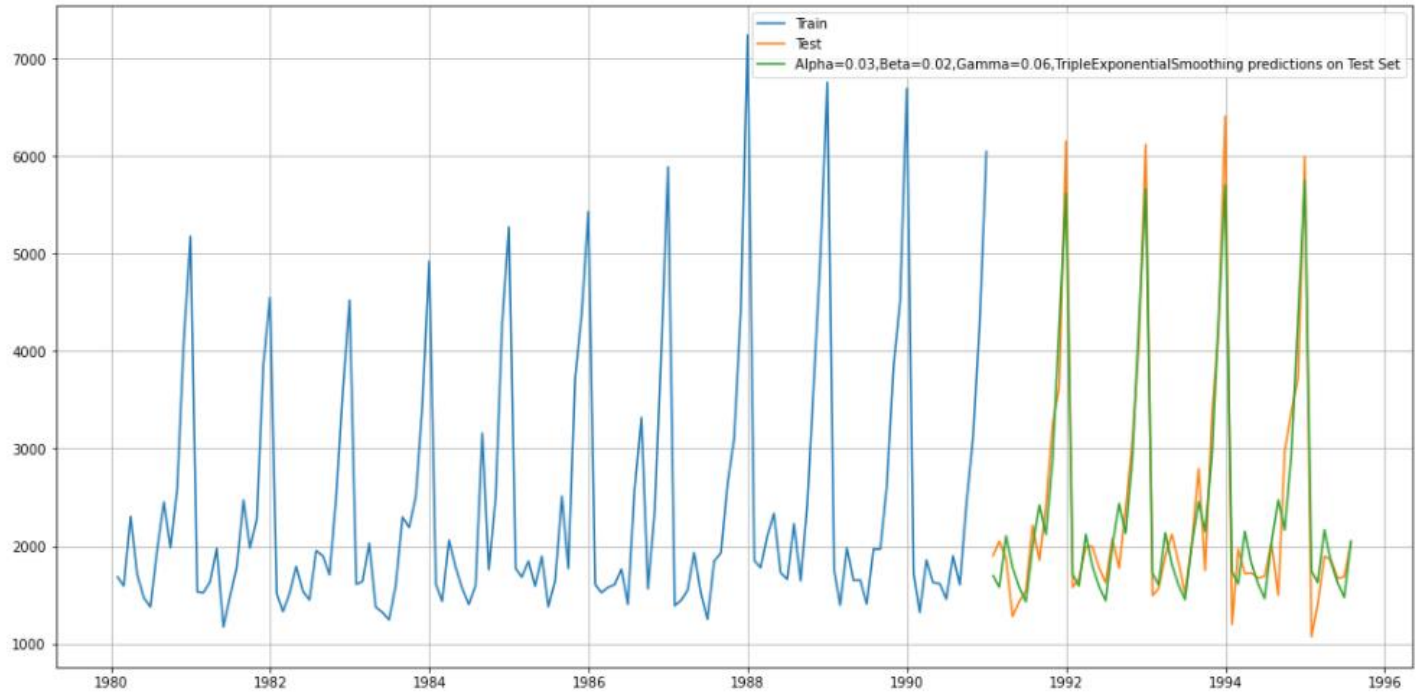


Fig 21: alpha=0.03, beta=0.02 and gamma=0.06 Triple Exponential Smoothing Model Forecast

	Test RMSE
Alpha=0.03,Beta=0.02,Gamma=0.06,TripleExponentialSmoothing	370.58

For this data, we had both trend and seasonality so by definition Triple Exponential Smoothing is supposed to work better than the Simple Exponential Smoothing as well as the Double Exponential Smoothing. However,

since this was a model building exercise we had gone on to build different models on the data and have compared these model with the best RMSE value on the test data.

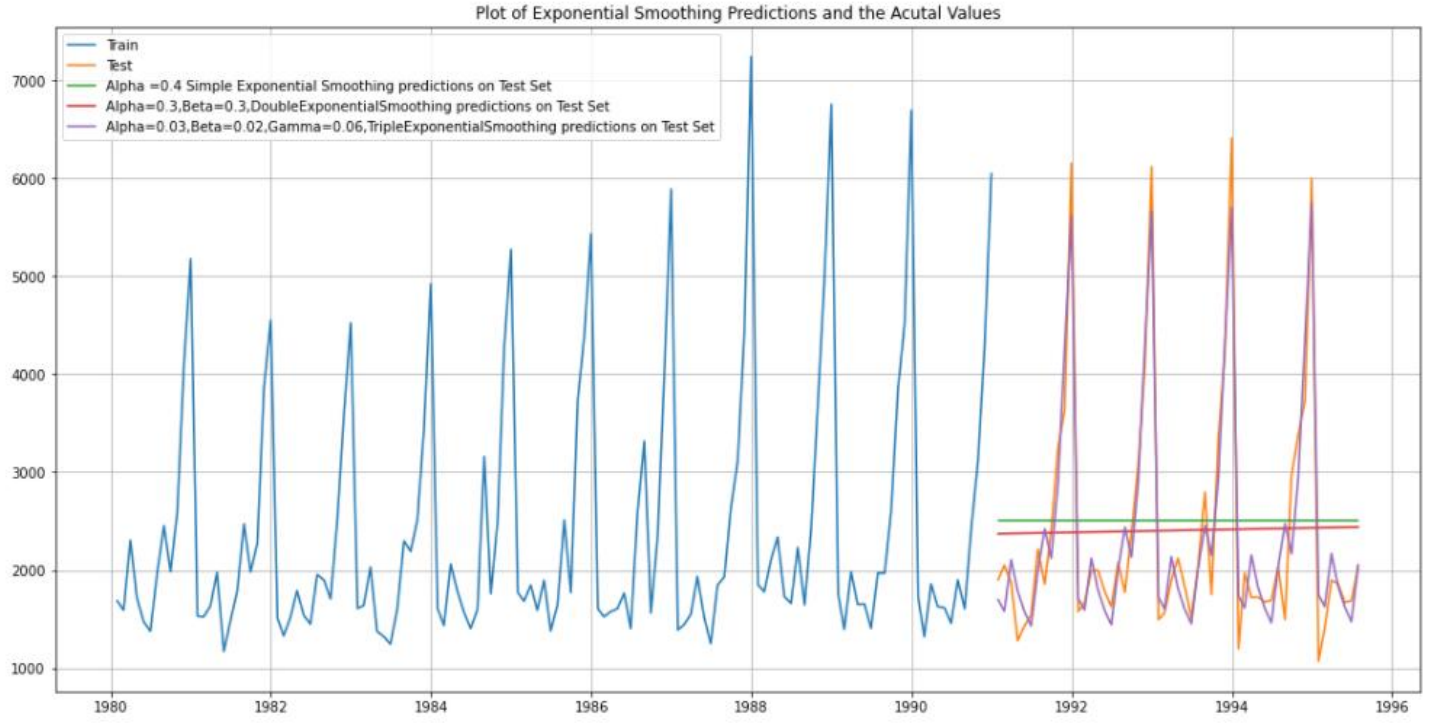


Fig 22: Plot of Exponential Smoothing Predictions and the Actual Values

The best model built so far is the Triple Exponential Model which parameters alpha=0.03, beta=0.02 and gamma= 0.06

Q 5. Check for the stationarity of the data on which the model is being built on using appropriate statistical tests and also mention the hypothesis for the statistical test. If the data is found to be non-stationary, take appropriate steps to make it stationary. Check the new data for stationarity and comment.

Note: Stationarity should be checked at alpha = 0.05.

A common assumption in many time series techniques is that the data are stationary.

A stationary process has the property that the mean, variance and autocorrelation structure do not change over time. Stationarity can be defined in precise mathematical terms, but for our purpose we mean a flat looking series, without trend, constant variance over time, a constant autocorrelation structure over time and no periodic fluctuations (seasonality).

If the time series is not stationary, we can often transform it to stationarity with one of the following techniques.

1. We can difference the data. That is, given the series Zt, we create the new series Yi=Zi−Zi−1. The differenced data will contain one less point than the original data.

2. If the data contain a trend, we can fit some type of curve to the data and then model the residuals from that fit.
3. For non-constant variance, taking the logarithm or square root of the series may stabilize the variance.

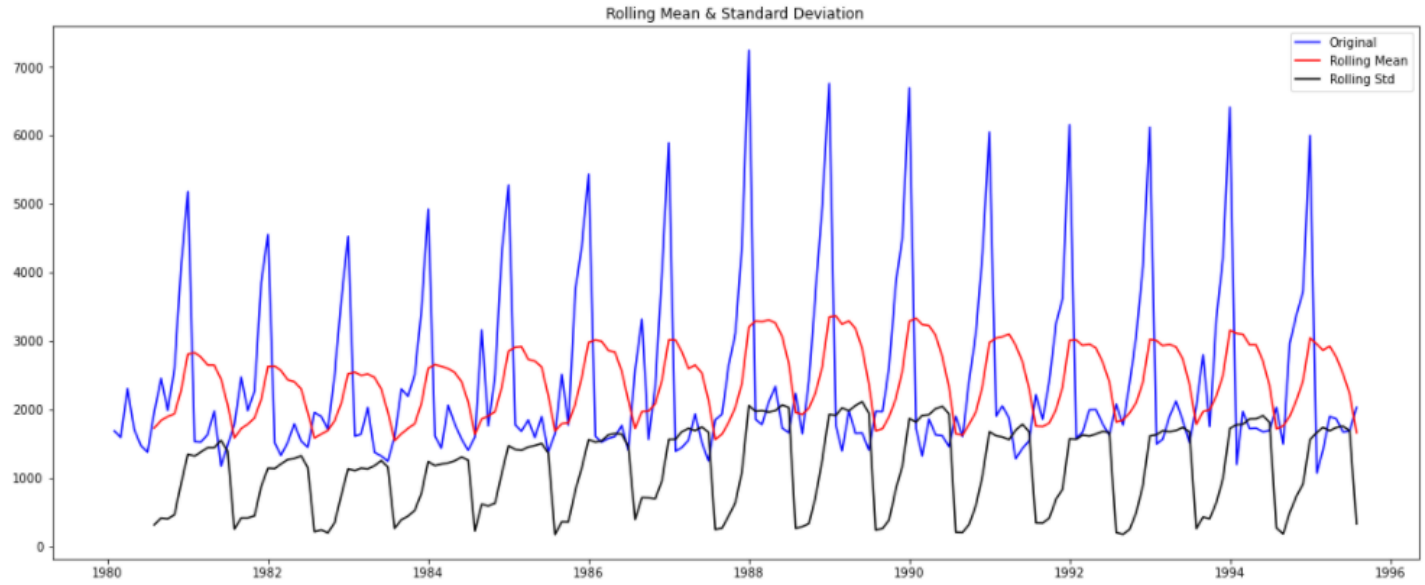
The Augmented Dickey-Fuller (ADF) test is a unit root test which determines whether there is a unit root and subsequently whether the series is non-stationary.

The hypothesis in a simple form for the ADF test is:

Null Hypothesis (Ho): The Time Series has a unit root and is thus non-stationary.

Alternate Hypothesis (Ha): The Time Series does not have a unit root and is thus stationary.

```
test_stationarity(df['Sparkling'])
```




```
Results of Dickey-Fuller Test:
Test Statistic                  -1.36
p-value                          0.60
#Lags Used                      11.00
Number of Observations Used    175.00
Critical Value (1%)             -3.47
Critical Value (5%)             -2.88
Critical Value (10%)            -2.58
dtype: float64
```

Fig 23: Augmented Dickey-Fuller Test and Statistic

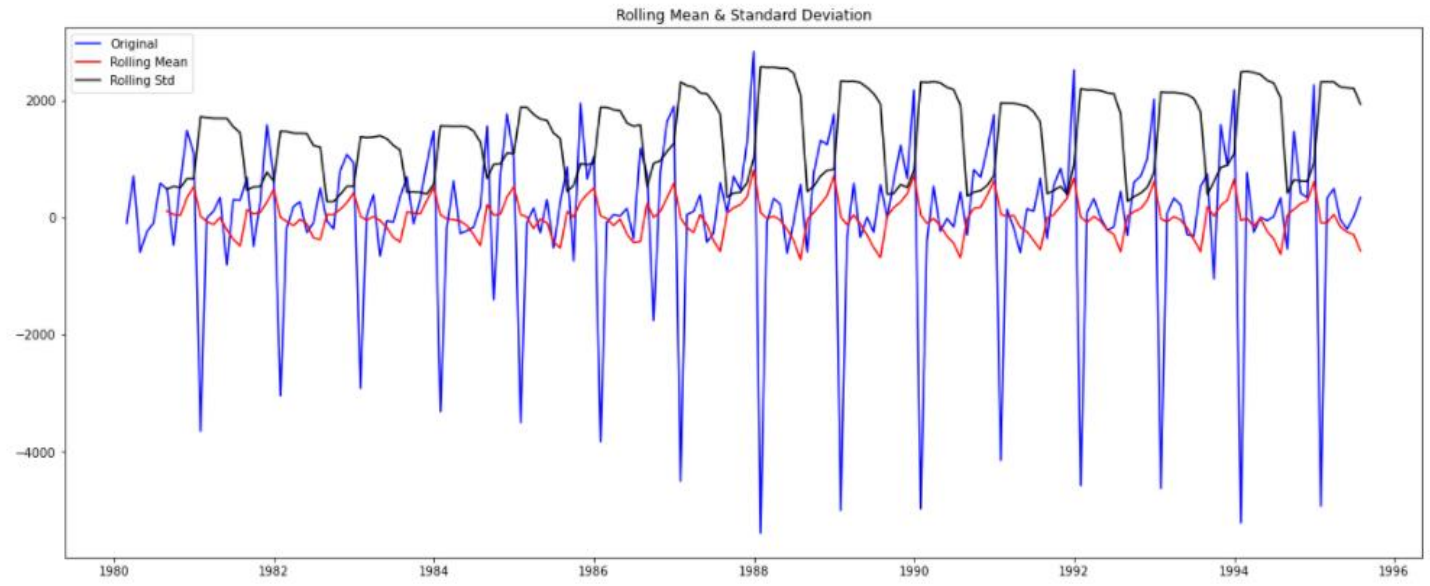
We see that at 5% significant level the Time Series is non-stationary.

Let us take a difference of order 1 and check whether the Time Series is stationary or not.

$$Yi=Zi-Zi-1$$

The differenced data will contain one less point than the original data.

```
test_stationarity(df['Sparkling'].diff().dropna())
```




```
Results of Dickey-Fuller Test:
Test Statistic                -45.05
p-value                         0.00
#Lags Used                     10.00
Number of Observations Used   175.00
Critical Value (1%)            -3.47
Critical Value (5%)            -2.88
Critical Value (10%)           -2.58
dtype: float64
```

Fig 24: Differenced Data Augmented Dickey-Fuller Test and Statistic

We see that at α = 0.05 the Time Series is indeed stationary.
We see that after taking a difference of order 1 the series have become stationary at α = 0.05.

Q 6. Build an automated version of the ARIMA/SARIMA model in which the parameters are selected using the lowest Akaike Information Criteria (AIC) on the training data and evaluate this model on the test data using RMSE.

1. AUTOMATED ARIMA MODEL

ARIMA, short for 'Auto Regressive Integrated Moving Average' is actually a class of models that 'explains' a given time series based on its own past values, that is, its own lags and the lagged forecast errors, so that equation can be used to forecast future values.

In an auto regression model, we forecast the variable of interest using a linear combination of past values of the variable. The term auto regression indicates that it is a regression of the variable against itself.

ARIMA models work on the following assumptions –

- The data series is stationary, which means that the mean and variance should not vary with time. A series can be made stationary by using log transformation or differencing the series.

- The data provided as input must be a univariate series, since arima uses the past values to predict the future values.

ARIMA has three components – AR (autoregressive term), I (differencing term) and MA (moving average term)–

- AR term refers to the past values used for forecasting the next value. The AR term is defined by the parameter 'p' in arima. The value of 'p' is determined using the PACF plot.
- MA term is used to define number of past forecast errors used to predict the future values. The parameter 'q' in arima represents the MA term. ACF plot is used to identify the correct 'q' value.
- Order of differencing specifies the number of times the differencing operation is performed on series to make it stationary. Test like ADF and KPSS can be used to determine whether the series is stationary and help in identifying the d value.

Before implementing ARIMA, we need to make the series stationary, and determine the values of p and q using the plots. Auto ARIMA makes this task really simple for us as it eliminates few steps. Below are the steps you should follow for implementing auto ARIMA:

1. Load the data: Load the data into your notebook
2. Pre-processing data: The input should be univariate, hence drop the other columns
3. Fit Auto ARIMA: Fit the model on the univariate series
4. Predict values on validation set: Make predictions on the validation set
5. Calculate RMSE: Check the performance of the model using the predicted values against the actual values

We got the best values as (2,1,2) upon fitting auto ARIMA based on AIC value.

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:            D.Sparkling   No. Observations:                  131
Model:                 ARIMA(2, 1, 2)   Log Likelihood               -1099.312
Method:                       css-mle   S.D. of innovations            1013.589
Date:                Sat, 24 Jul 2021   AIC                            2210.624
Time:                        22:31:47   BIC                            2227.875
Sample:                    02-29-1980   HQIC                           2217.634
                         - 12-31-1990
==========================================================================================
                             coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                      5.5845      0.519     10.767      0.000       4.568       6.601
ar.L1.D.Sparkling          1.2699      0.075     17.041      0.000       1.124       1.416
ar.L2.D.Sparkling         -0.5602      0.074     -7.618      0.000      -0.704      -0.416
ma.L1.D.Sparkling         -1.9960      0.043    -46.887      0.000      -2.079      -1.913
ma.L2.D.Sparkling          0.9960      0.043     23.336      0.000       0.912       1.080
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            1.1334           -0.7074j            1.3361           -0.0888
AR.2            1.1334           +0.7074j            1.3361            0.0888
MA.1            1.0003           +0.0000j            1.0003            0.0000
MA.2            1.0037           +0.0000j            1.0037            0.0000
-----------------------------------------------------------------------------
```

```
from sklearn.metrics import  mean_squared_error
rmse = mean_squared_error(test['Sparkling'],predicted_auto_ARIMA[0],squared=False)
print(rmse)
```

```
1374.1084496024253
```

The RMSE value for auto ARIMA(2,1,2) is 1374.108

6. AUTOMATED SARIMA MODEL

When the data is seasonal, like it happen after a certain period of time then SARIMA is used.

Here we will have to add one more term that is seasonal_order (p,d,q,period)

P, q, d values will remains the same.

period value will be the value after what period of time seasonality occurs.

We need to first find out the lags by looking at the ACF plot.

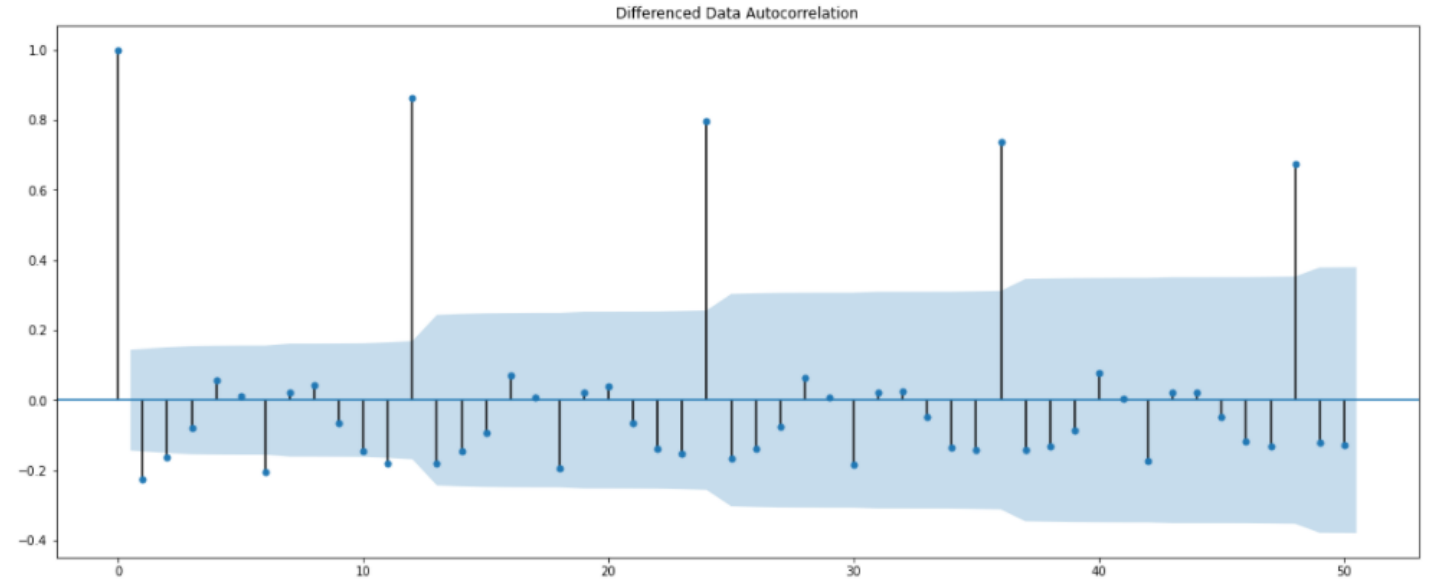


Fig 25: Differenced Auto Correlation Plot

Seasonal behaviour: We look at what's going on around lags 0, 12, 24, 36 and so on. The ACF tapers in multiples of S. We see that there can be a seasonality of 0, 12 as well as 24. We will run our auto SARIMA models by setting seasonality at both 12 and 24

Setting the seasonality as 12 for the first iteration of the auto SARIMA model.

```
SARIMA_AIC.sort_values(by=['AIC']).head()
```

	param	seasonal	AIC
50	(1, 1, 2)	(1, 0, 2, 12)	1555.58
53	(1, 1, 2)	(2, 0, 2, 12)	1556.08
26	(0, 1, 2)	(2, 0, 2, 12)	1557.12
23	(0, 1, 2)	(1, 0, 2, 12)	1557.16
80	(2, 1, 2)	(2, 0, 2, 12)	1557.69

SARIMAX Results

Dep. Variable:	y	No. Observations:	132
Model:	SARIMAX(1, 1, 2)x(1, 0, 2, 12)	Log Likelihood	-770.792
Date:	Sat, 24 Jul 2021	AIC	1555.584
Time:	22:59:45	BIC	1574.095
Sample:	0	HQIC	1563.083
	- 132		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.6281	0.255	-2.463	0.014	-1.128	-0.128
ma.L1	-0.1041	0.225	-0.463	0.643	-0.545	0.337
ma.L2	-0.7276	0.154	-4.734	0.000	-1.029	-0.426
ar.S.L12	1.0439	0.014	72.843	0.000	1.016	1.072
ma.S.L12	-0.5551	0.098	-5.663	0.000	-0.747	-0.363
ma.S.L24	-0.1355	0.120	-1.133	0.257	-0.370	0.099
sigma2	1.506e+05	2.03e+04	7.400	0.000	1.11e+05	1.9e+05

Ljung-Box (L1) (Q):	0.04	Jarque-Bera (JB):	11.72
Prob(Q):	0.84	Prob(JB):	0.00
Heteroskedasticity (H):	1.47	Skew:	0.36
Prob(H) (two-sided):	0.26	Kurtosis:	4.48

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

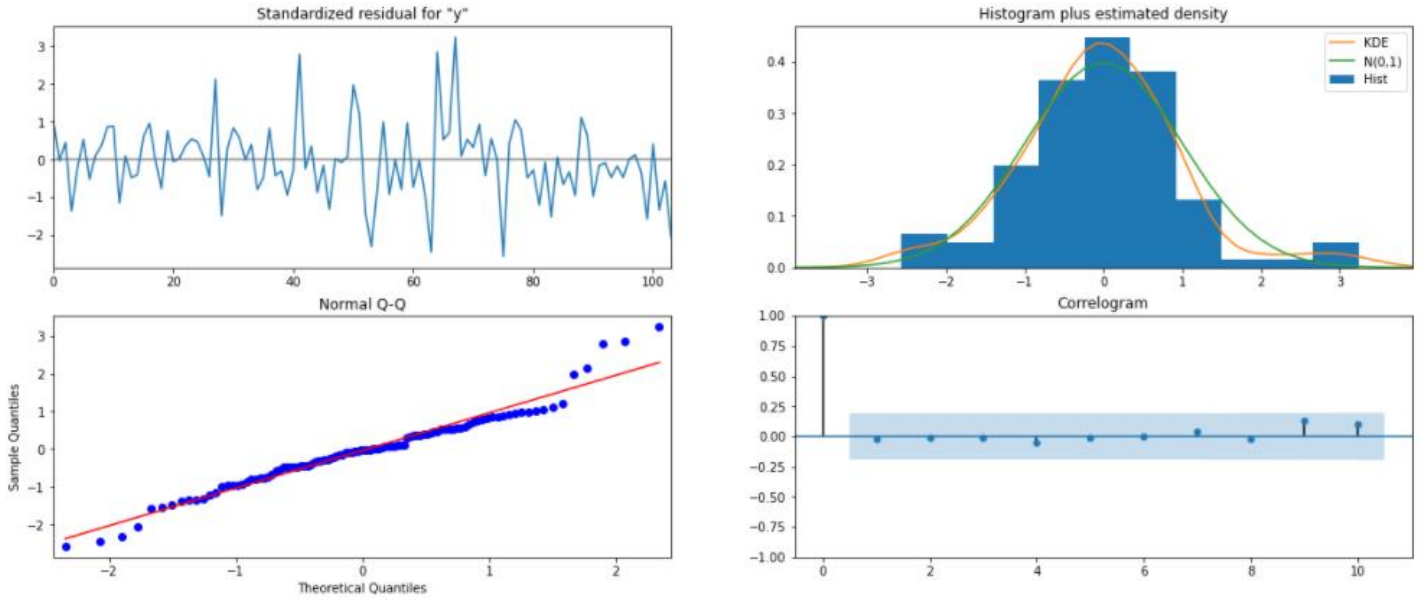


Fig 26: Model Diagnostics SARIMA (1,1,2)X(1,0,2,12)

From the model diagnostics plot, we can see that all the individual diagnostics plots almost follow the theoretical numbers and thus we cannot develop any pattern from these plots.

Model Evaluation for SARIMA (1,1,2) x(1,0,2,12)

```
predicted_auto_SARIMA_12.summary_frame(alpha=0.05).head()
```

y	mean	mean_se	mean_ci_lower	mean_ci_upper
0	1327.37	388.35	566.22	2088.52
1	1315.11	402.02	527.18	2103.05
2	1621.60	402.01	833.67	2409.52
3	1598.86	407.25	800.67	2397.05
4	1392.69	407.98	593.07	2192.31

```
rmse = mean_squared_error(test['Sparkling'],predicted_auto_SARIMA_12.predicted_mean,squared=False)
print(rmse)
```

```
528.6024251728568
```

Setting the seasonality as 24 for the second iteration of the auto SARIMA model.

```
SARIMA_AIC.sort_values(by=['AIC']).head()
```

	param	seasonal	AIC
26	(0, 1, 2)	(2, 0, 2, 24)	1223.38
53	(1, 1, 2)	(2, 0, 2, 24)	1224.28
80	(2, 1, 2)	(2, 0, 2, 24)	1225.18
50	(1, 1, 2)	(1, 0, 2, 24)	1229.14
23	(0, 1, 2)	(1, 0, 2, 24)	1229.28

```
                                 SARIMAX Results
==========================================================================================
Dep. Variable:                                 y   No. Observations:                  132
Model:             SARIMAX(0, 1, 2)x(2, 0, 2, 24)   Log Likelihood                -604.691
Date:                            Sat, 24 Jul 2021   AIC                           1223.383
Time:                                    23:05:55   BIC                           1240.057
Sample:                                         0   HQIC                          1230.068
                                              - 132
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -1.2397      0.299     -4.150      0.000      -1.825      -0.654
ma.L2          0.1253      0.215      0.582      0.560      -0.297       0.547
ar.S.L24       0.4486      0.136      3.297      0.001       0.182       0.715
ar.S.L48       0.7541      0.155      4.862      0.000       0.450       1.058
ma.S.L24       0.2283      1.100      0.207      0.836      -1.929       2.385
ma.S.L48      -0.9695      0.694     -1.396      0.163      -2.330       0.391
sigma2      9.876e+04   1.02e+05      0.966      0.334   -1.02e+05    2.99e+05
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                28.04
Prob(Q):                              0.95   Prob(JB):                         0.00
Heteroskedasticity (H):               1.25   Skew:                             0.15
Prob(H) (two-sided):                  0.56   Kurtosis:                         5.88
===================================================================================
```

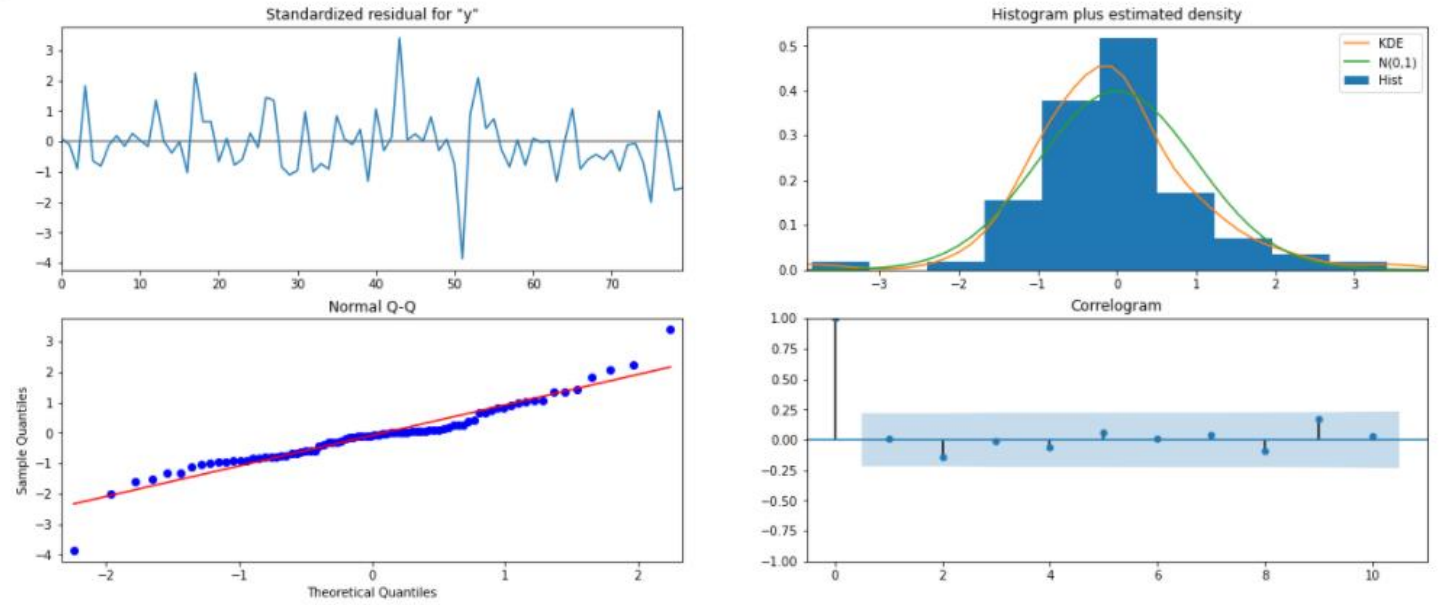


Fig 27: Model Diagnostics SARIMA (0,1,2)X(2,0,2,24)

Similar to the last iteration of the model where the seasonality parameter was taken as 12, here also we see that the model diagnostics plot does not indicate any remaining information that we can get.

Model Evaluation for SARIMA (0,1,2) x(2,0,2,24)

```
predicted_auto_SARIMA_24.summary_frame(alpha=0.05).head()
```

	y	mean	mean_se	mean_ci_lower	mean_ci_upper
0		1525.77	430.57	681.86	2369.68
1		1250.17	430.65	406.11	2094.23
2		1688.26	432.84	839.91	2536.61
3		1388.49	435.10	535.71	2241.27
4		1297.47	437.35	440.28	2154.66

```
rmse = mean_squared_error(test['Sparkling'],predicted_auto_SARIMA_24.predicted_mean,squared=False)
print(rmse)
```

```
639.8833123545458
```

We see that the RMSE value have not reduced further when the seasonality parameter was changed to 24.

Q 7. Build ARIMA/SARIMA models based on the cut-off points of ACF and PACF on the training data and evaluate this model on the test data using RMSE.

1. ARIMA MODEL (PACF/ACF)

The ACF stands for Autocorrelation function, and the PACF for Partial Autocorrelation function. Looking at these two plots together can help us form an idea of what models to fit. Autocorrelation computes and plots the autocorrelations of a time series. Autocorrelation is the correlation between observations of a time series separated by k time units.

Similarly, partial autocorrelations measure the strength of relationship with other terms being accounted for, in this case other terms being the intervening lags present in the model.

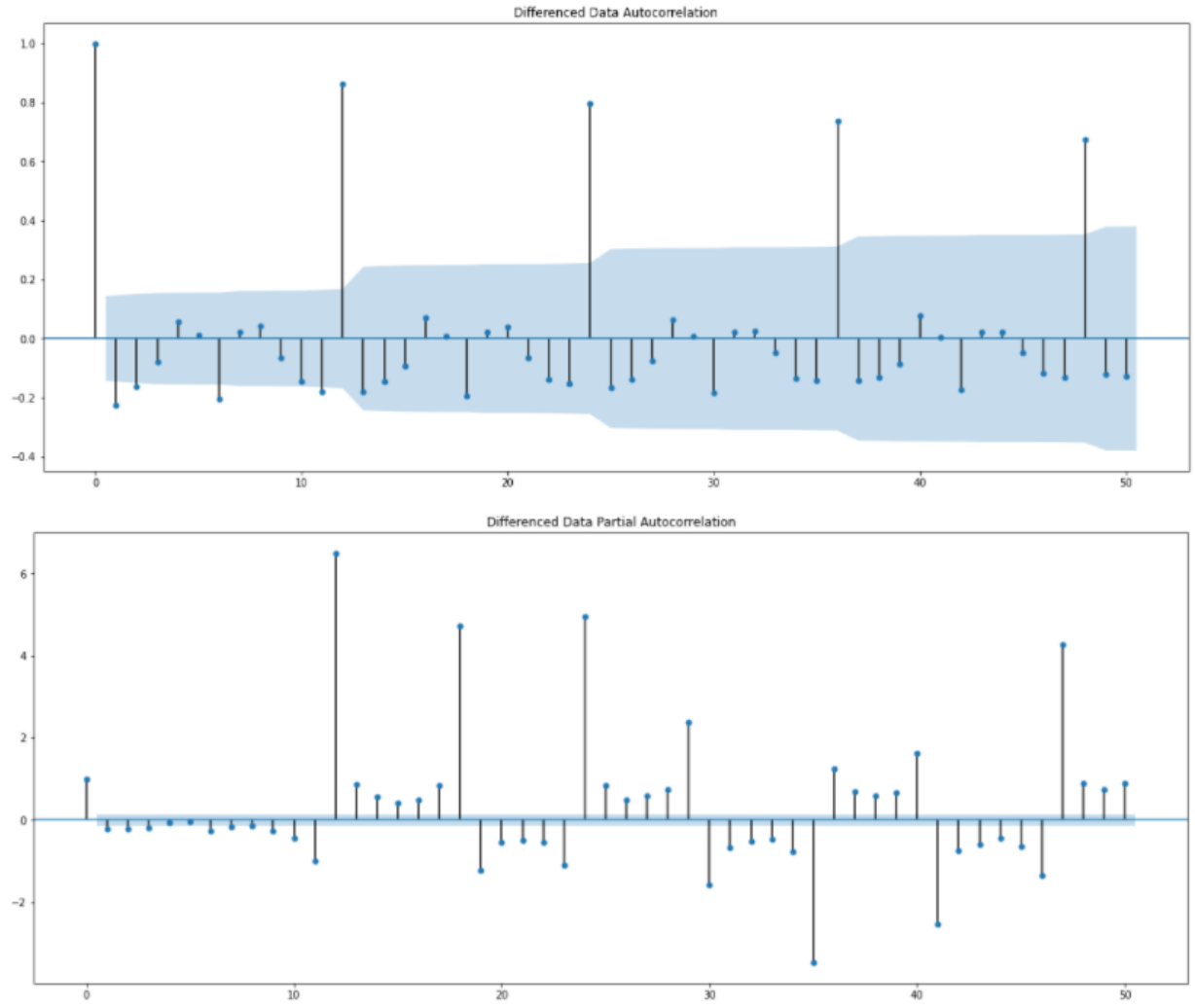


Fig 28: ACF/PACF Plots

Here, we have taken alpha=0.05.

- The Auto-Regressive parameter in an ARIMA model is 'p' which comes from the significant lag before which the PACF plot cuts-off to 0
- The Moving-Average parameter in an ARIMA model is 'q' which comes from the significant lag before the ACF plot cuts-off to 0
- Difference here is also 0

Therefore we will build ARIMA on (0, 0, 0)

```
                              ARMA Model Results
==============================================================================
Dep. Variable:              Sparkling   No. Observations:                  132
Model:                     ARMA(0, 0)   Log Likelihood               -1133.602
Method:                           css   S.D. of innovations           1298.484
Date:                Sun, 25 Jul 2021   AIC                           2271.203
Time:                        13:54:32   BIC                           2276.969
Sample:                  01-31-1980   HQIC                          2273.546
                       - 12-31-1990
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const       2403.7803    113.018     21.269      0.000    2182.268    2625.292
==============================================================================
```

```
rmse = mean_squared_error(test['Sparkling'],predicted_manual_ARIMA[0],squared=False)
print(rmse)
```

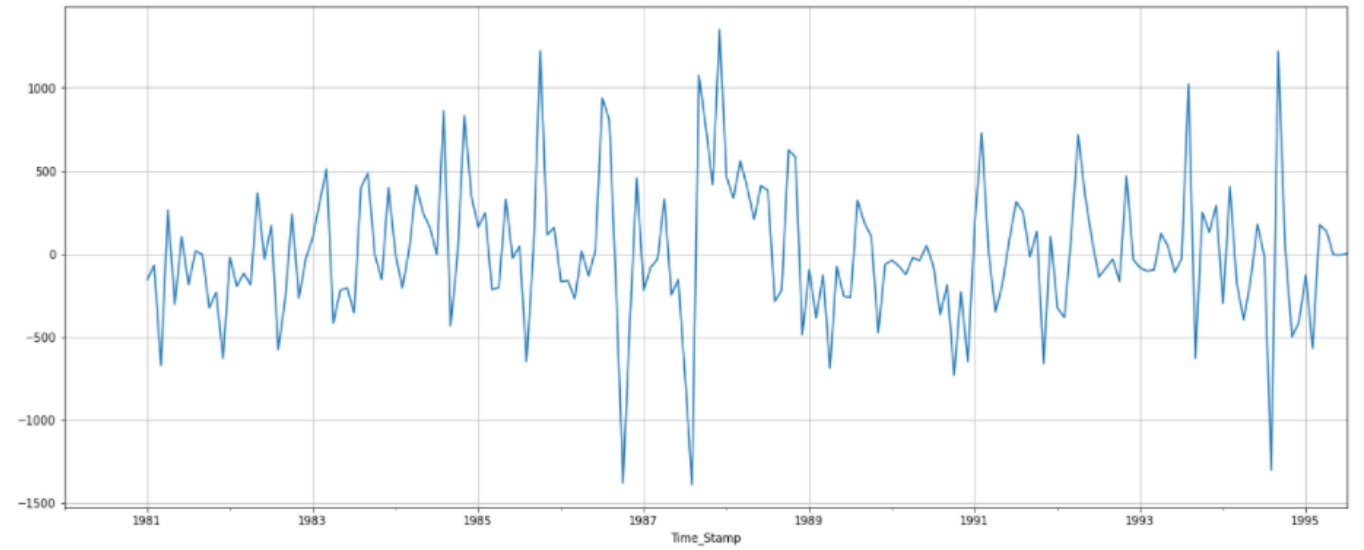
```
1275.0818036965309
```

We see that there is difference in the RMSE values for both the models, but remember that the second model is a much simpler model.

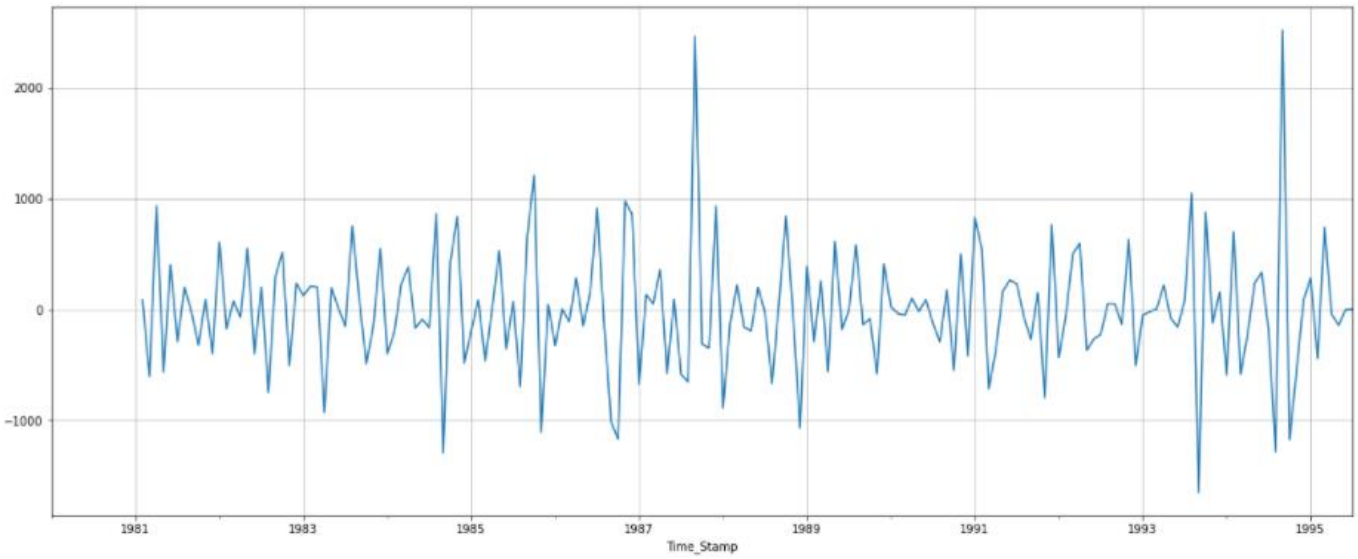
2. SARIMA MODEL (PACF/ACF)

We see that our ACF plot at the seasonal interval (12) does not taper off. So, we go ahead and take a seasonal differencing of the original series.

```
(df['Sparkling'].diff(12)).plot()
plt.grid();
```



We took a seasonality difference of 12 on our original series. We see that there might be a slight trend which can be noticed in the data. So we take a differencing of first order on the seasonally differenced series.



Now we see that there is almost no trend present in the data. Seasonality is only present in the data.

Checking the ACF and the PACF plots for the new modified Time Series.

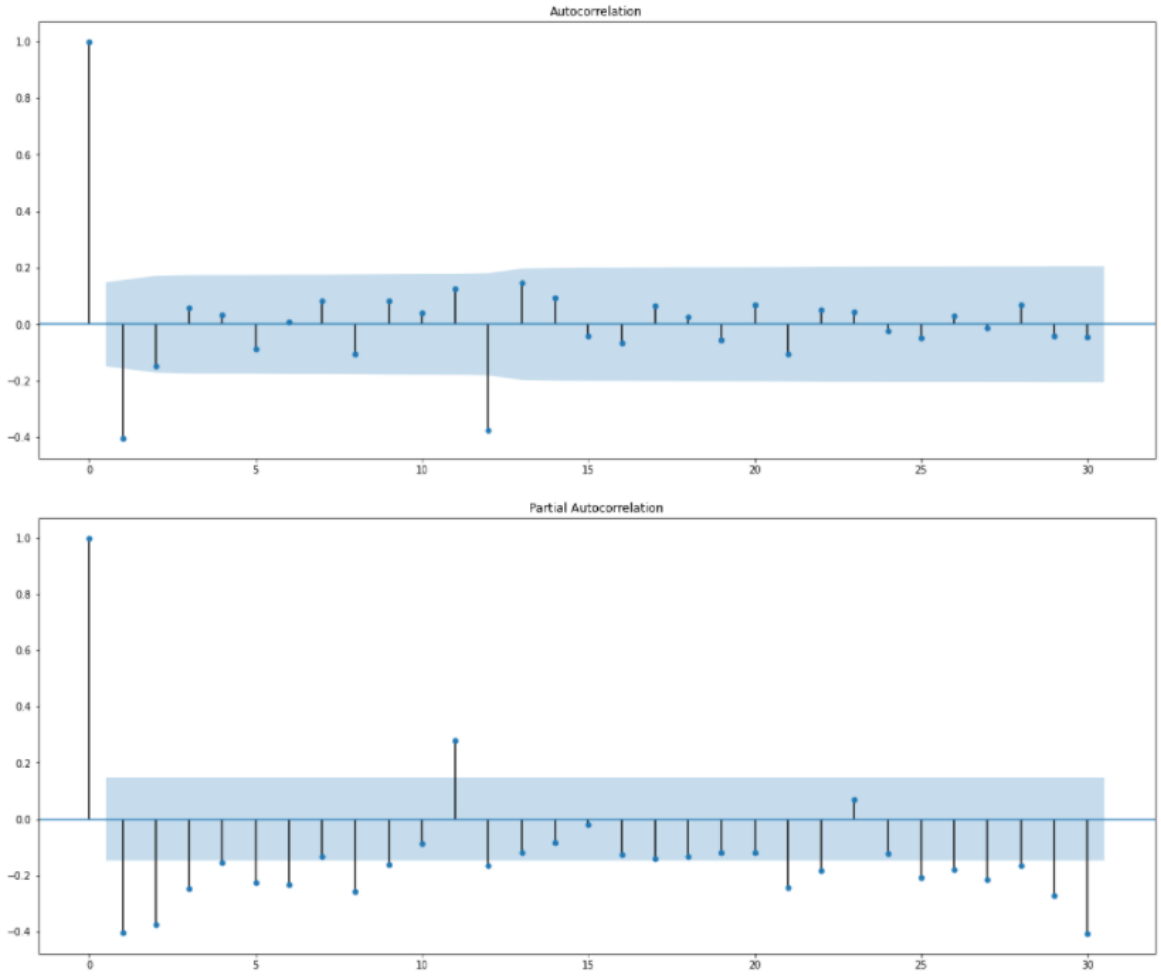


Fig 29: Modified Time Series ACF/PACF Plots

Here, we have taken alpha=0.05.

We are going to take the seasonal period as 12. We will keep the p(1) and q(1) parameters same as the ARIMA model.

- The Auto-Regressive parameter in an SARIMA model is 'P' which comes from the significant lag after which the PACF plot cuts-off to 0.
- The Moving-Average parameter in an SARIMA model is 'q' which comes from the significant lag after which the ACF plot cuts-off to 0.

```
                                 SARIMAX Results
==========================================================================================
Dep. Variable:                           y   No. Observations:                  132
Model:             SARIMAX(0, 2, 0, 12)   Log Likelihood                -857.674
Date:                  Sun, 25 Jul 2021   AIC                           1717.347
Time:                          14:15:05   BIC                           1720.020
Sample:                               0   HQIC                          1718.431
                                  - 132
Covariance Type:                    opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
sigma2      5.34e+05    5.4e+04      9.890      0.000    4.28e+05     6.4e+05
===================================================================================
Ljung-Box (L1) (Q):                   6.22   Jarque-Bera (JB):                17.23
Prob(Q):                              0.01   Prob(JB):                         0.00
Heteroskedasticity (H):               1.87   Skew:                            -0.57
Prob(H) (two-sided):                  0.06   Kurtosis:                         4.61
===================================================================================
```

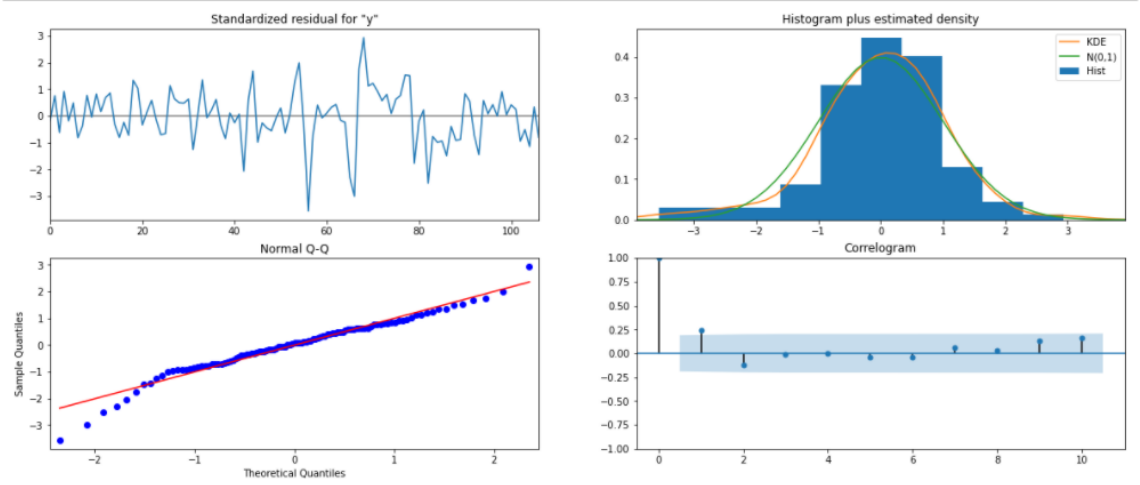


Fig 30: Model Diagnostics SARIMA (0,0,0)X(0,2,0,12)

	y	mean	mean_se	mean_ci_lower	mean_ci_upper
0	1683.00	730.75	250.76	3115.24
1	1248.00	730.75	-184.24	2680.24
2	1736.00	730.75	303.76	3168.24
3	1606.00	730.75	173.76	3038.24
4	1576.00	730.75	143.76	3008.24

```
rmse = mean_squared_error(test['Sparkling'],predicted_manual_SARIMA_12.predicted_mean,squared=False)
print(rmse)
```

```
944.0149941027996
```

Q 8. Build a table with all the models built along with their corresponding parameters and the respective RMSE values on the test data.

	Test RMSE
RegressionOnTime	1275.87
NaiveModel	3864.28
SimpleAverageModel	1275.08
2pointTrailingMovingAverage	813.40
4pointTrailingMovingAverage	1156.59
6pointTrailingMovingAverage	1283.93
9pointTrailingMovingAverage	1346.28
Alpha=0.04960,SimpleExponentialSmoothing	1316.04
Alpha=0.02,SimpleExponentialSmoothing	1279.50
Alpha=0.02,Beta=0.50,DoubleExponentialSmoothing	1274.63
Alpha=0.111,Beta=0.061,Gamma=0.395,TripleExponentialSmoothing	469.59
Alpha=0.03,Beta=0.02,Gamma=0.06,TripleExponentialSmoothing	370.58
ARIMA(2,1,2)	1374.11
ARIMA(1,1,1)	1461.67
SARIMA(1,1,2)(1,0,2,12)	528.60
SARIMA(0,1,2)(2,0,2,24)	639.88
ARIMA(0,0,0)/ACF-PACF	1275.08
SARIMA(0,0,0)(0,2,0,12)/ACF-PACF	944.01

These were all the models which we have built so far. Now, we will sort them in ascending order as per their RMSE value.

	Test RMSE
Alpha=0.03,Beta=0.02,Gamma=0.06,TripleExponentialSmoothing	370.58
Alpha=0.111,Beta=0.061,Gamma=0.395,TripleExponentialSmoothing	469.59
SARIMA(1,1,2)(1,0,2,12)	528.60
SARIMA(0,1,2)(2,0,2,24)	639.88
2pointTrailingMovingAverage	813.40
SARIMA(0,0,0)(0,2,0,12)/ACF-PACF	944.01
4pointTrailingMovingAverage	1156.59
Alpha=0.02,Beta=0.50,DoubleExponentialSmoothing	1274.63
SimpleAverageModel	1275.08
ARIMA(0,0,0)/ACF-PACF	1275.08
RegressionOnTime	1275.87
Alpha=0.02,SimpleExponentialSmoothing	1279.50
6pointTrailingMovingAverage	1283.93
Alpha=0.04960,SimpleExponentialSmoothing	1316.04
9pointTrailingMovingAverage	1346.28
ARIMA(2,1,2)	1374.11
ARIMA(1,1,1)	1461.67
NaiveModel	3864.28

Here, we can conclude that Triple Exponential Model with parameters alpha=0.03, beta=0.02 and gamma=0.06 have been the best model built out of all with the best RMSE score as 370.58

Q9. Based on the model-building exercise, build the most optimum model(s) on the complete data and predict 12 months into the future with appropriate confidence intervals/bands.

The most optimum model is the Triple Exponential Model. Therefore, we will build the full model on this.

Here, we have a scenario where our training data was stationary but our full data was not stationary. So, we will use the same parameters as our training data but with adding a level of differencing which is needed for the data to be stationary.

The model to be built on the whole data are the following:

- Alpha=0.03,Beta=0.02,Gamma=0.06,TripleExponentialSmoothing

```
fullmodel = ExponentialSmoothing(df,
                                 trend='additive',
                                 seasonal='additive').fit(smoothing_level=0.03,
                                                          smoothing_trend=0.02,
                                                          smoothing_seasonal=0.06)
```

```
RMSE_fullmodel = metrics.mean_squared_error(df['Sparkling'],fullmodel.fittedvalues,squared=False)

print('RMSE:',RMSE_fullmodel1)
```

```
RMSE: 379.8042317541343
```

```
mape = mape(df['Sparkling'],fullmodel.fittedvalues)
print('MAPE:', mape)
```

```
MAPE: 12.614088597664475
```

We will calculate the upper and lower confidence bands at 95% confidence level. Here we are taking the multiplier to be 1.96 as we want to plot with respect to a 95% confidence intervals.

	lower_CI	prediction	upper_ci
1995-08-31	1535.86	2281.32	3026.79
1995-09-30	1548.12	2293.59	3039.06
1995-10-31	2405.70	3151.16	3896.63
1995-11-30	3446.44	4191.91	4937.37
1995-12-31	5207.72	5953.19	6698.66

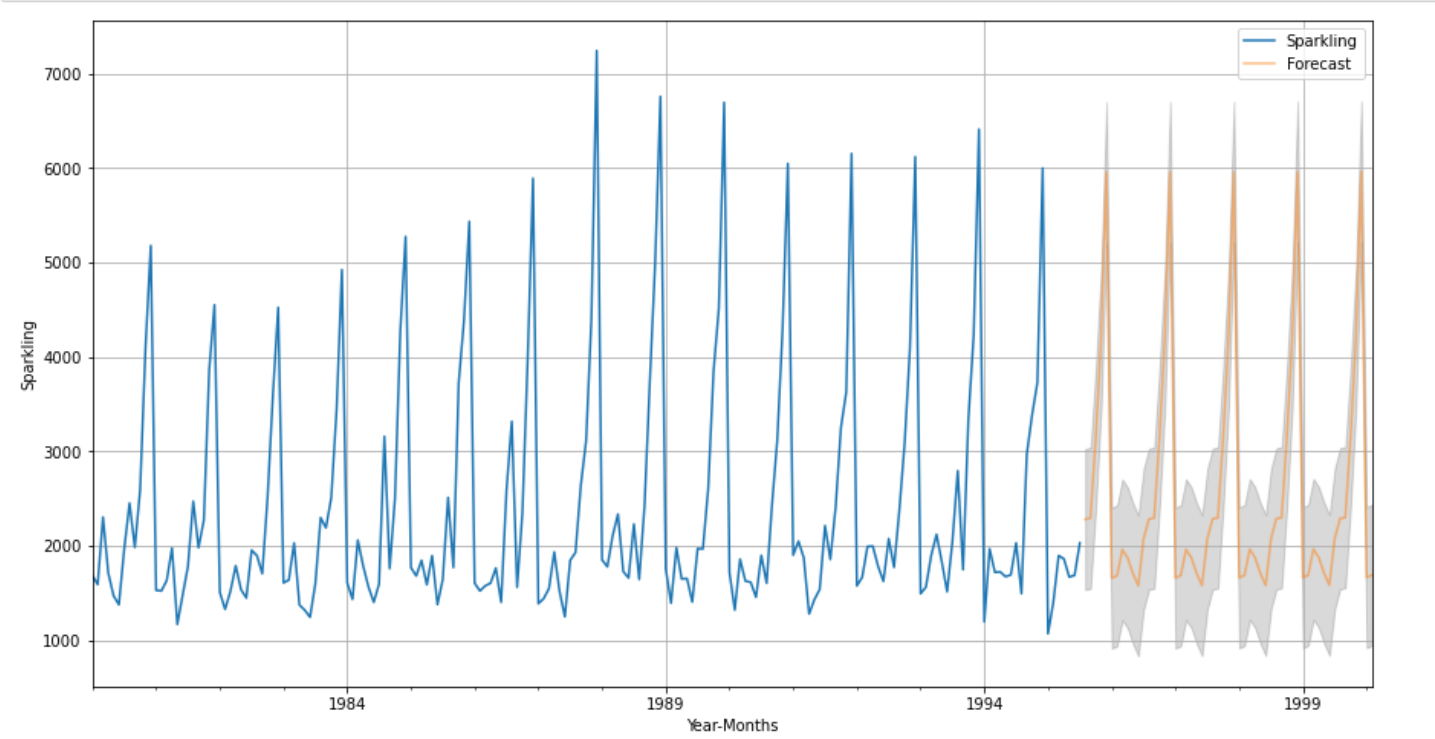


Fig 31: Full Model prediction on best model

Getting the predictions for the same number of times stamps that are present in the test data.

Here we have got the RMSE value as 379.80 on the full data and MAPE value as12.61

Now, we will predict 12 months into future using our model on the full data.

Predicting 12 months into the future.

```
# Getting the predictions for 12 months into future
prediction_12 = fullmodel.forecast(steps=12)
```

```
df.plot()
prediction_12.plot();
```

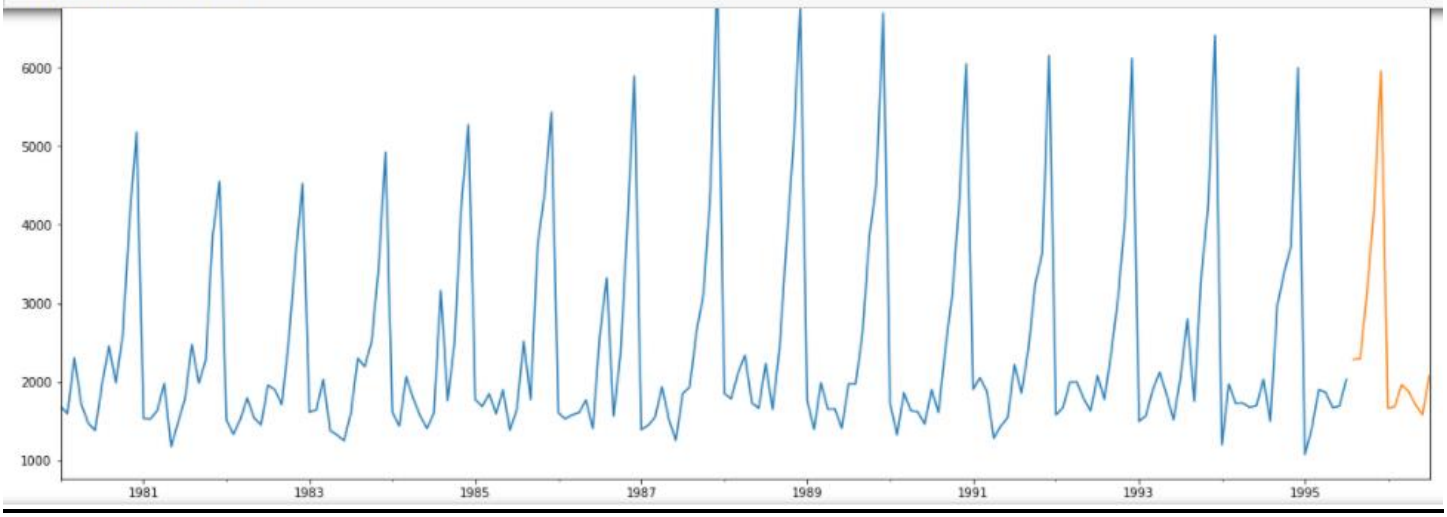
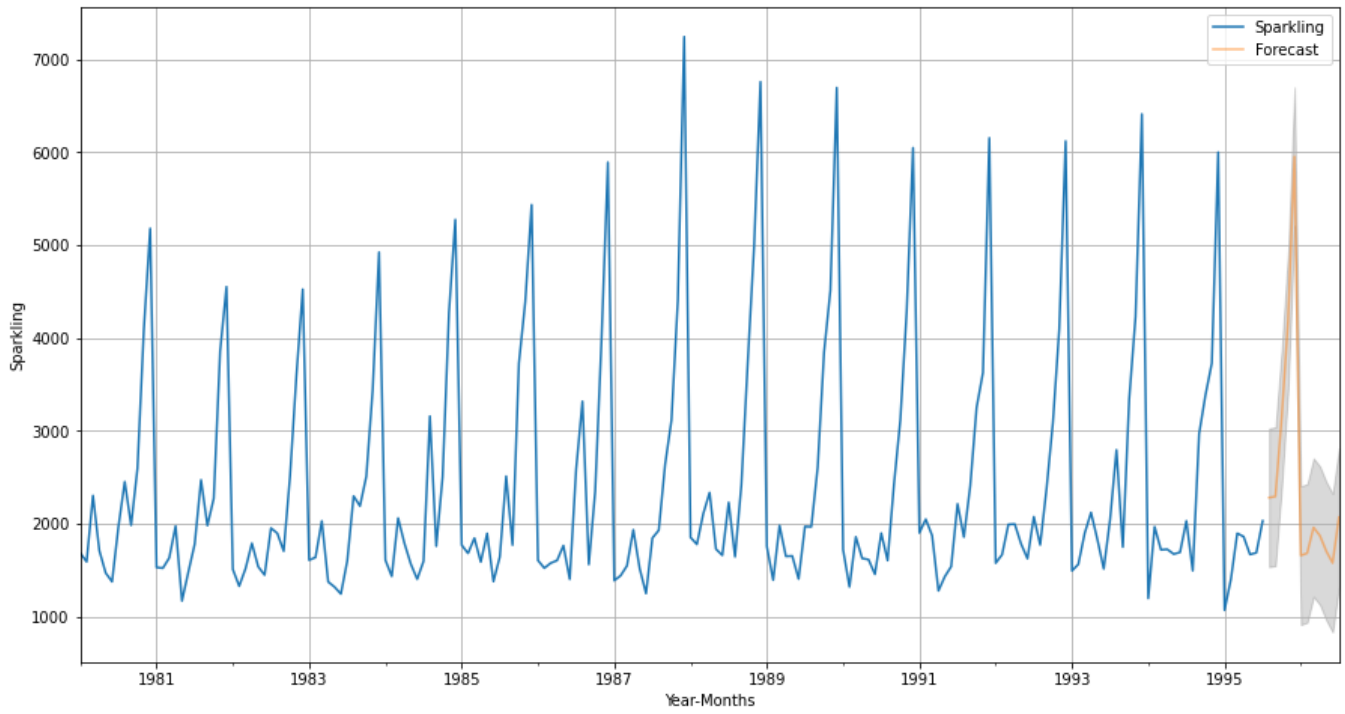


Fig 32: Prediction of 12 months into future on best model

We will calculate the upper and lower confidence bands at 95% confidence level. Here we are taking the multiplier to be 1.96 as we want to plot with respect to a 95% confidence intervals.

	lower_CI	prediction	upper_ci
1995-08-31	1535.86	2281.32	3026.79
1995-09-30	1548.12	2293.59	3039.06
1995-10-31	2405.70	3151.16	3896.63
1995-11-30	3446.44	4191.91	4937.37
1995-12-31	5207.72	5953.19	6698.66



Q 10. Comment on the model thus built and report your findings and suggest the measures that the company should be taking for future sales.

Inference:

- The final model built is Triple Exponential Model with parameters alpha=0.03, beta=0.02 and gamma=0.06 and with the best RMSE score as 370.58
- We could predict the future Sales of next 12 months very accurately with this model.

Insights and recommendations:

- The Sales of Sparkling Wine is maximum in the last few months of the year i.e. October, November, December
- The Sales have remained almost the same in all these years.
- The Sales of Sparkling Wine has drastically dropped in 1995. This might be because we are considering the data up to only the 7th month of 1995.
- The minimum Sales have been recorded generally in the 2nd month i.e, February of every year. The reason for this should be investigated.
- The company can give offers on wine in the start of every year i.e. the first 4 months to boost Sales at that period.
- As people prefer Sparkling Wine at the end of year, the price can be slightly increased to maximize the profit.
