Great Learning
POWER AHEAD

Rose Wine Data Project Report

DEEPTI AGRAWAL

Submission Date:
25th July 2021

CONTENTS

Rose Wine Dataset:

LIST OF FIGURES

Rose 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 Rose Wine data.

INTRODUCTION

The purpose of this whole exercise is to forecast the Sales of Rose 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.

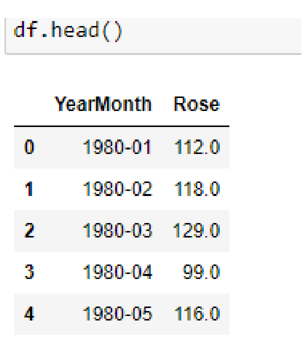
Q 1 Read the data as an appropriate Time Series data and plot the data.

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**.**

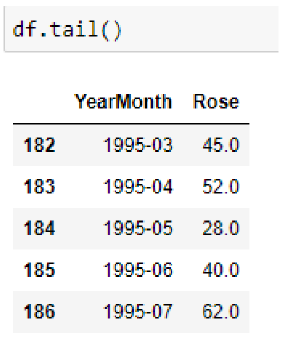
Time series forecasting is applied to extract information from historical series and is used to predict future behavior of the same series based on past pattern.

<u>Sample of the Dataset:</u>



```
df.head()
```

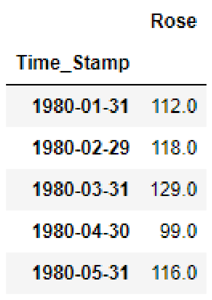
	YearMonth	Rose
0	1980-01	112.0
1	1980-02	118.0
2	1980-03	129.0
3	1980-04	99.0
4	1980-05	116.0



```
df.tail()
```

	YearMonth	Rose
182	1995-03	45.0
183	1995-04	52.0
184	1995-05	28.0
185	1995-06	40.0
186	1995-07	62.0

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



Time_Stamp	Rose
1980-01-31	112.0
1980-02-29	118.0
1980-03-31	129.0
1980-04-30	99.0
1980-05-31	116.0

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

Plotting the time series

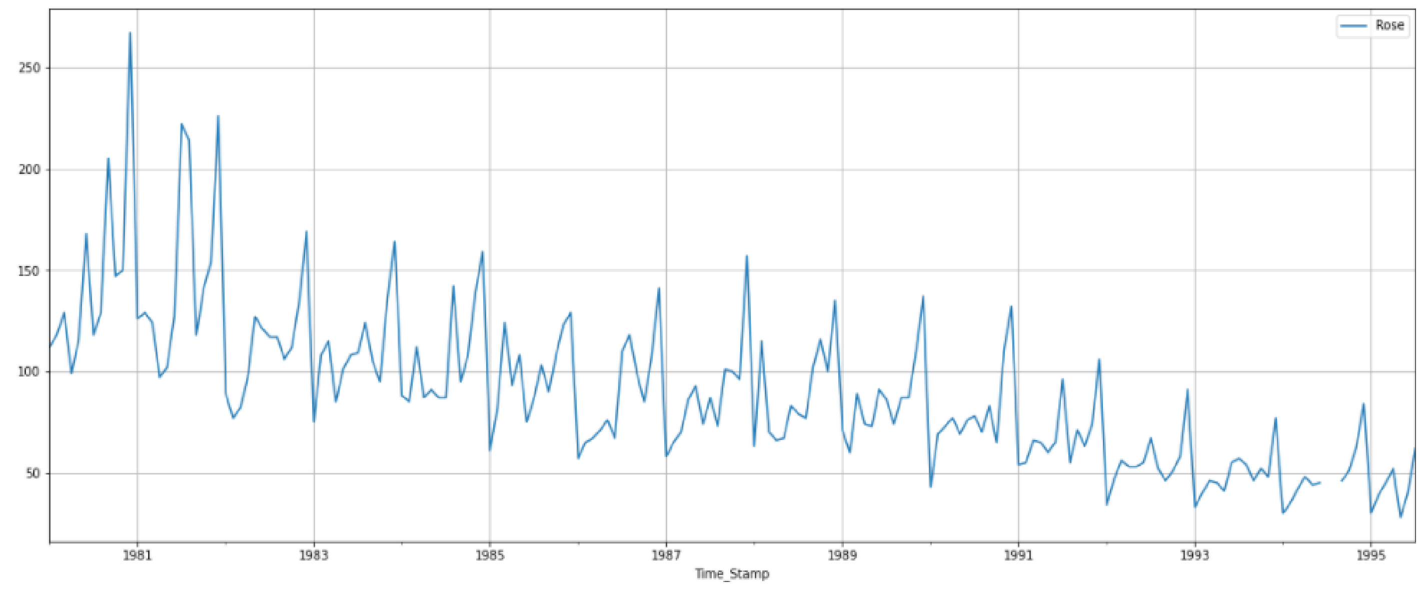


Fig 1: Rose Dataset Plot

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.

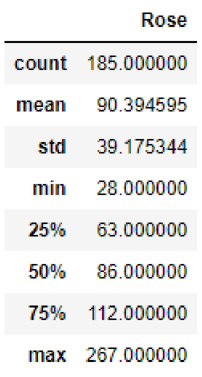
Q 2 Perform appropriate Exploratory Data Analysis to understand the data and also perform decomposition.

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.



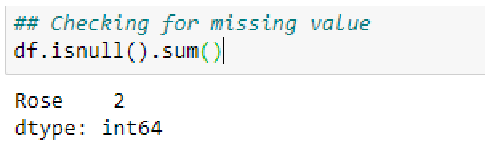
	Rose
count	185.000000
mean	90.394595
std	39.175344
min	28.000000
25%	63.000000
50%	86.000000
75%	112.000000
max	267.000000

The basic measures of descriptive statistics tell us how the Rose 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 90.39
- The maximum Sales recorded during this time period is 267

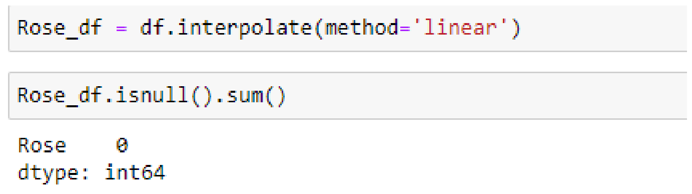
Check for missing values



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

```
Rose     2
dtype: int64
```

From the above result, we can see that there are 2 missing values present in our dataset. Therefore, we will interpolate the data.



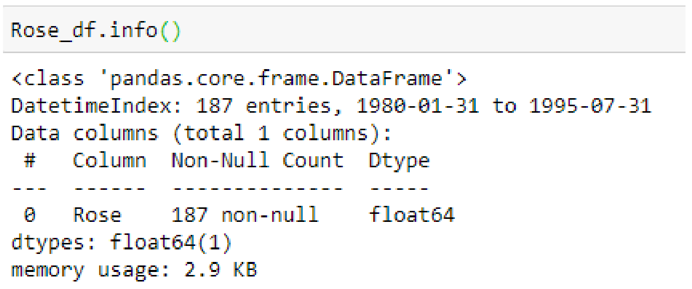
```
Rose_df = df.interpolate(method='linear')
```

```
Rose_df.isnull().sum()
```

```
Rose     0
dtype: int64
```

Basic Info

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



```
Rose_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   Rose    187 non-null    float64
dtypes: float64(1)
memory usage: 2.9 KB
```

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

Yearly Boxplot

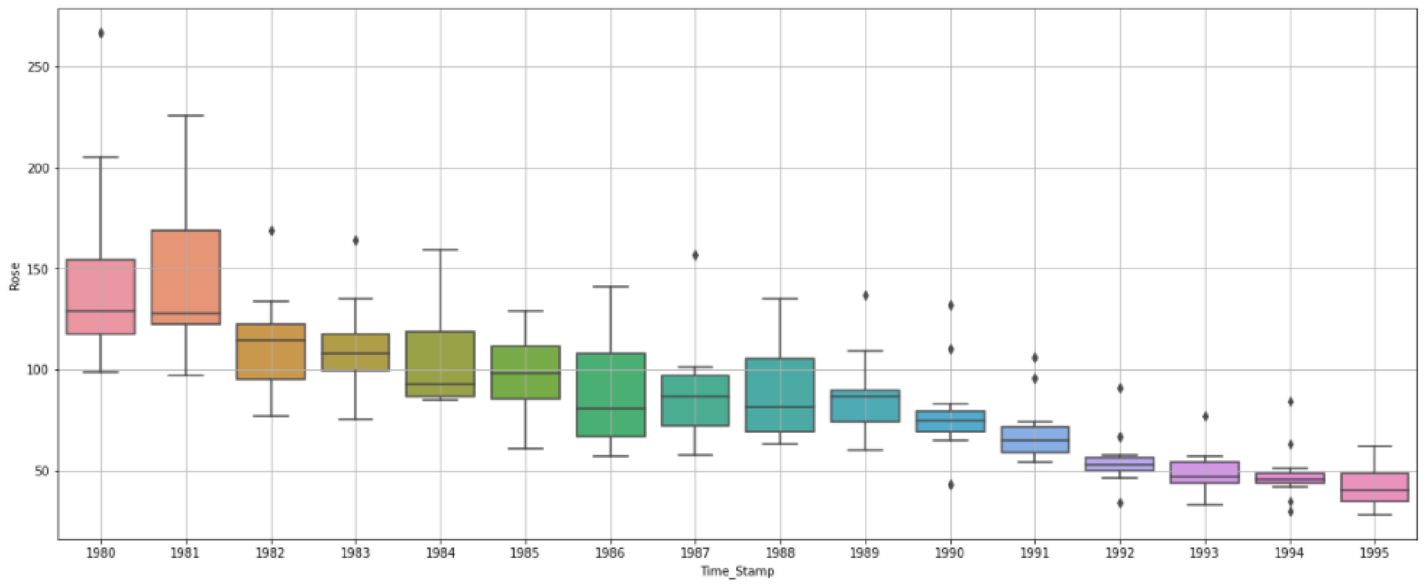


Fig 2: Yearly Boxplot Graph

This yearly graph shows that the Sales have drastically dropped in all the years.

Monthly Boxplot

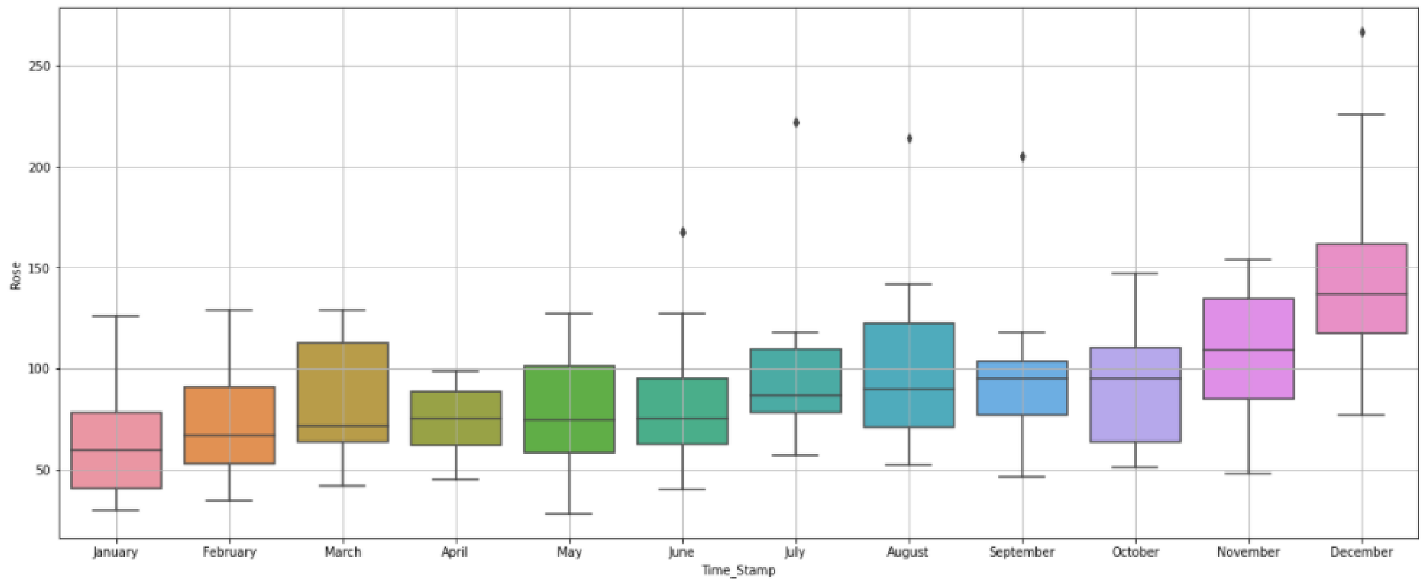


Fig 3: Monthly Boxplot Graph

The monthly graph shows that every year, last few month have recorded the highest Sales specially the month of December.
Also, the Sales in all the other months have also shown a good number altogether.

Time Series Month plot

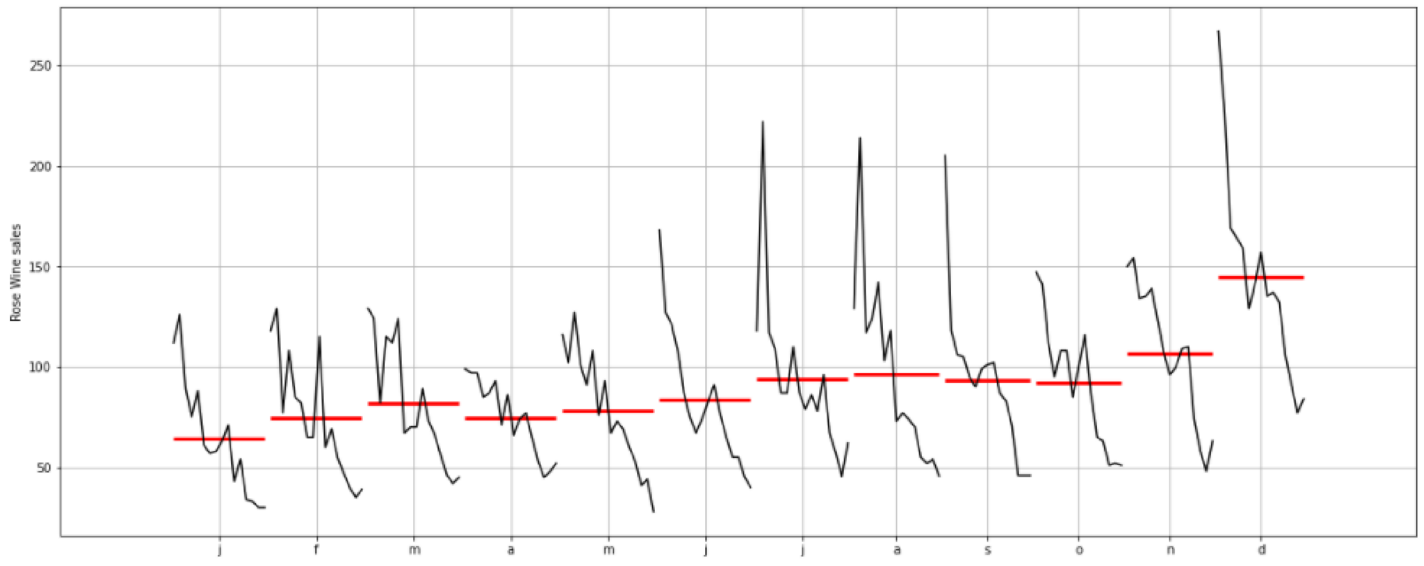


Fig 4: Time Series Monthly Plot

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

Monthly Sales across Years

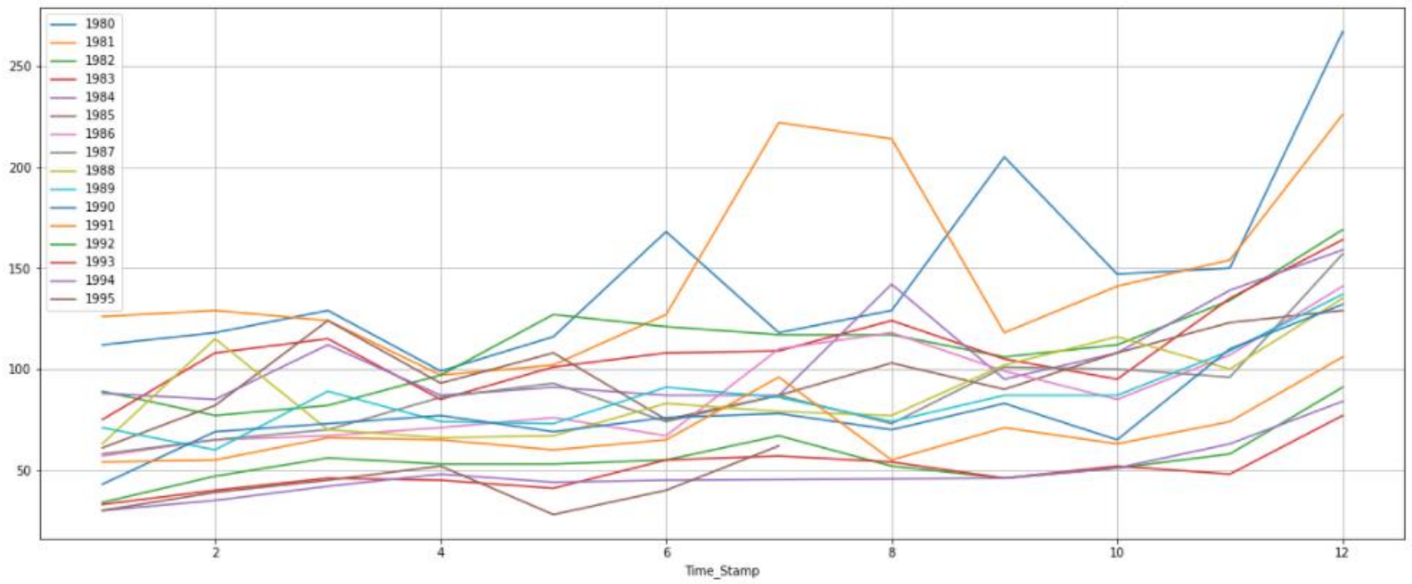


Fig 5: Yearly Sales across Months

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

Empirical Distribution

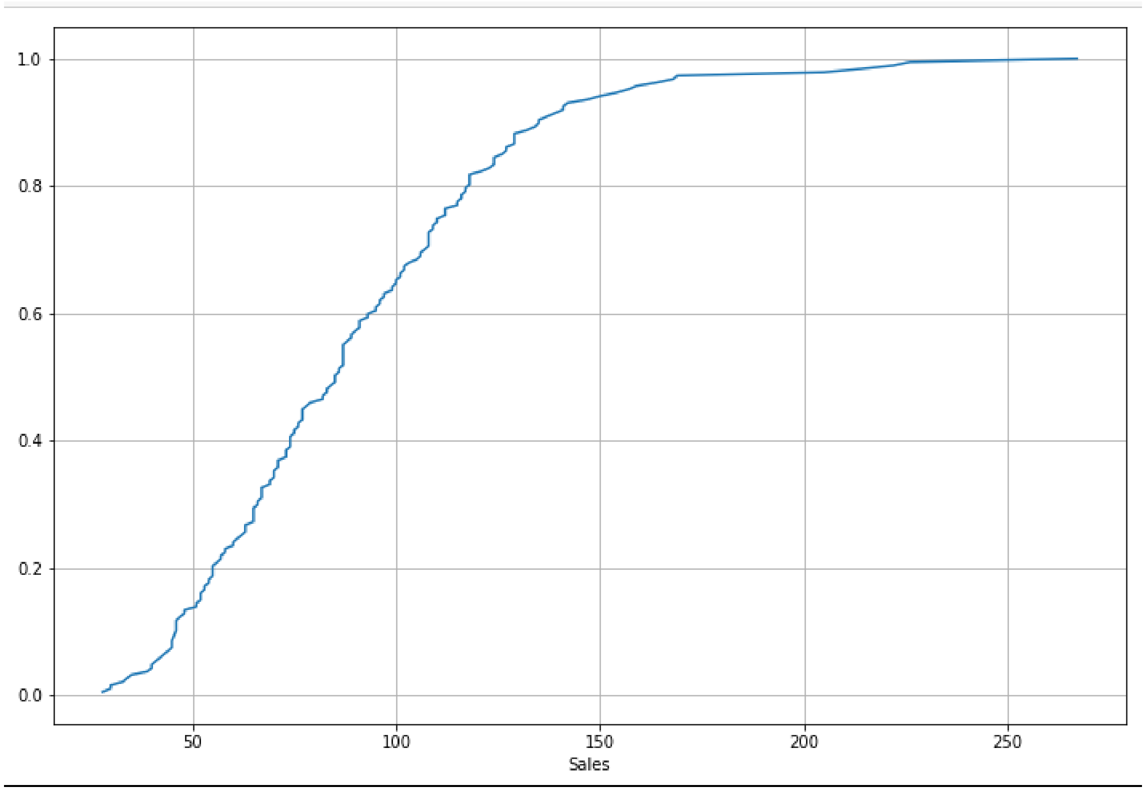


Fig 6: Empirical Distribution

This particular graph tells us what percentage of data points refer to what number of Sales. 80% of the sales are below 120. Maximum sales is close 250

Average Monthly Sales & Month on Month Percent Change in Sales

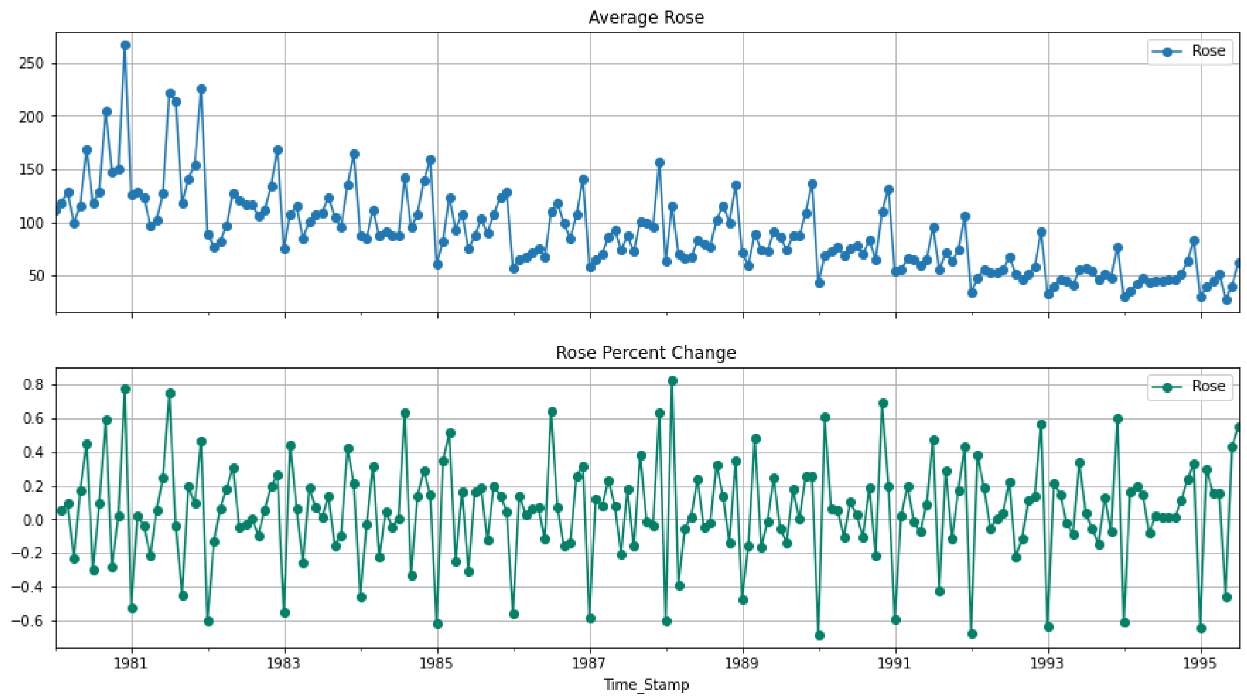


Fig 7: Average Sales and Monthly Percent Change in Sales

<u>Decomposing the Time Series</u>

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. <u>Additive Decomposition</u>

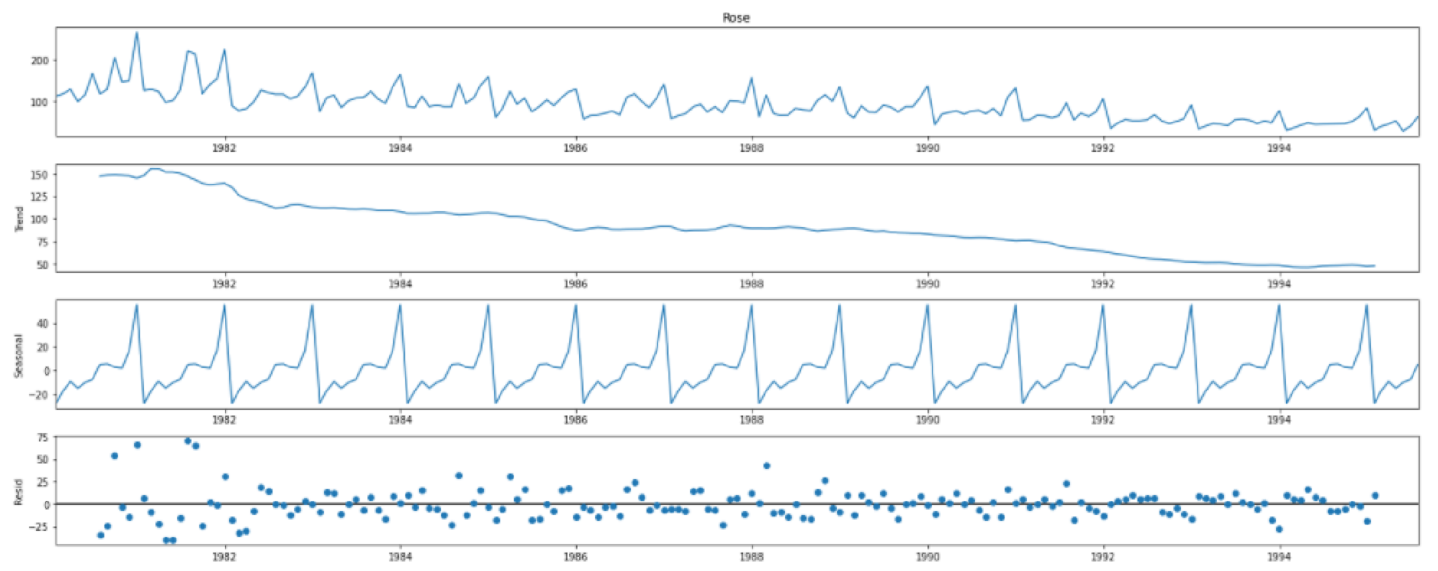


Fig 8: Additive Decomposition

2. <u>Multiplicative Decomposition</u>

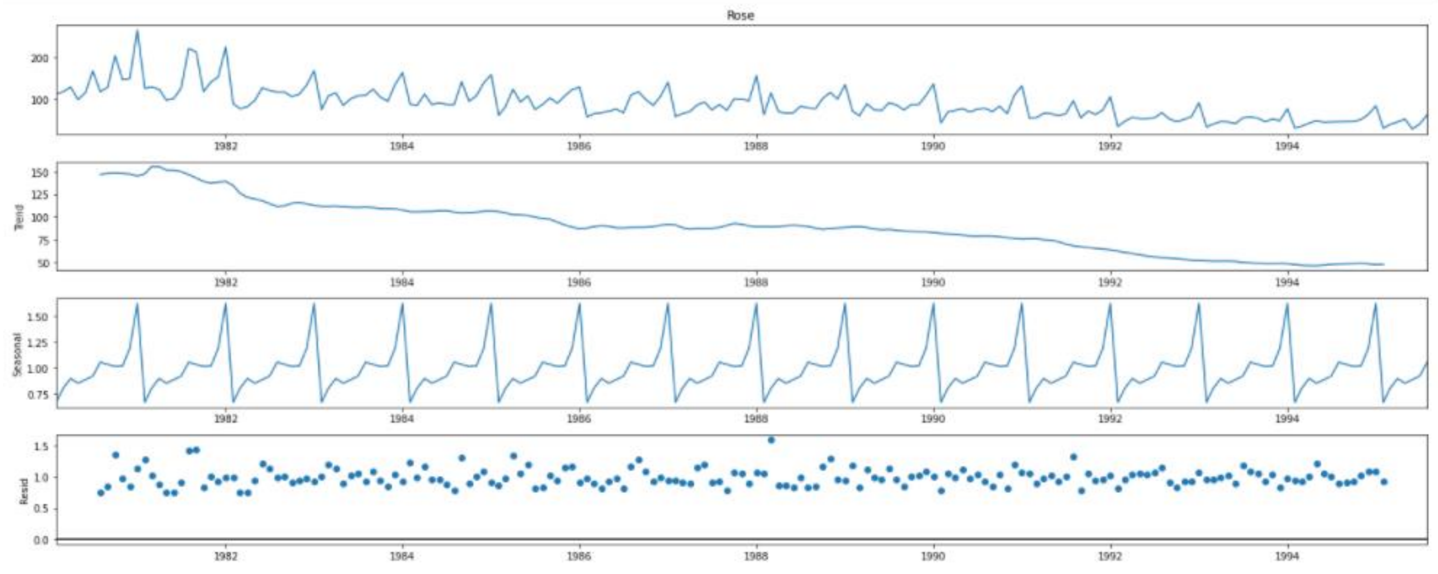


Fig 9: Multiplicative Decomposition

Auto Correlation Plots

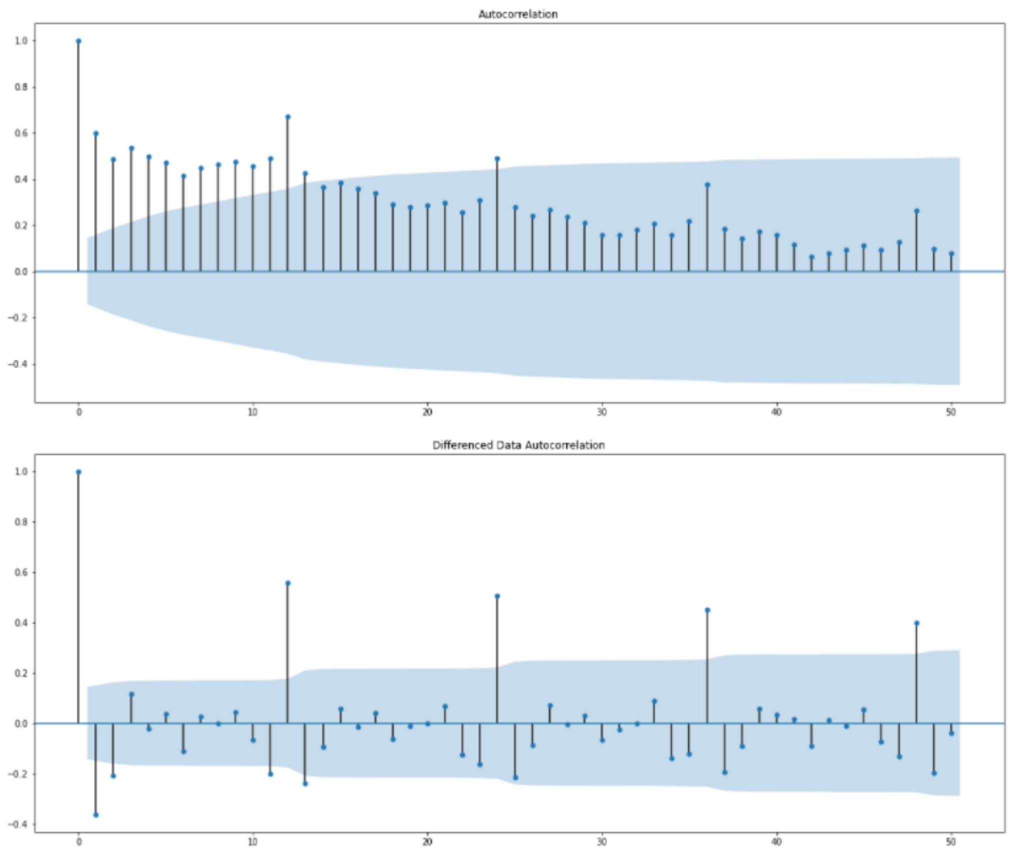


Fig 10: Auto Correlation Plots

Partial Auto Correlation Plots

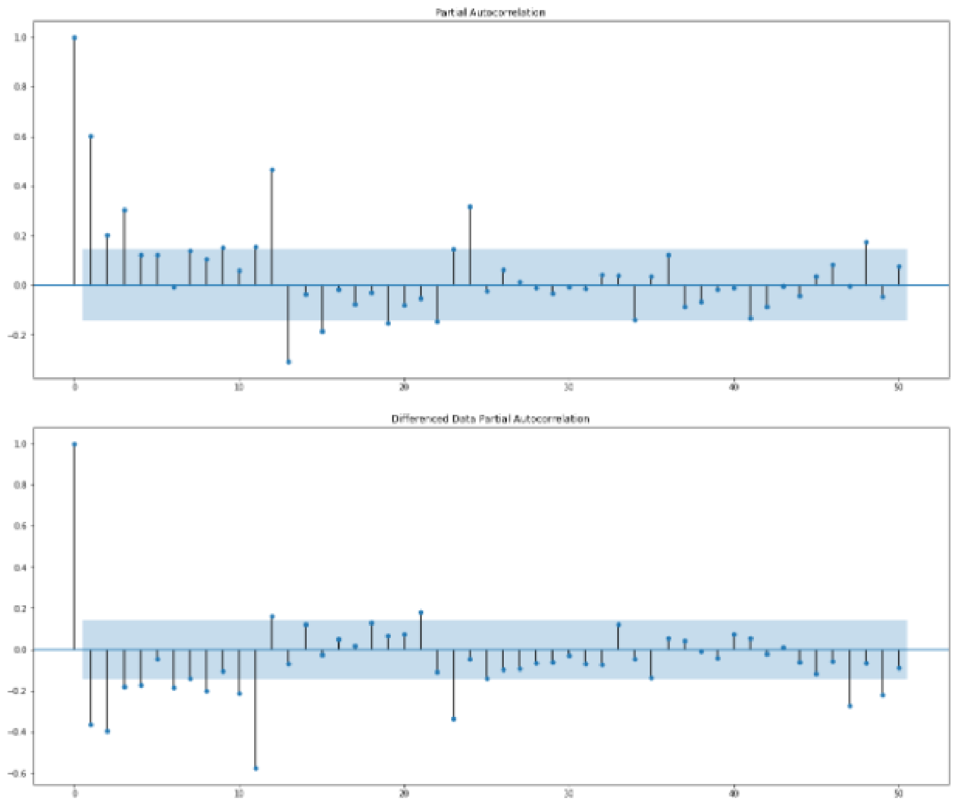


Fig 11: Partial Auto Correlation Plots

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=Rose_df[Rose_df.index.year < 1991]
test=Rose_df[Rose_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

             Rose
Time_Stamp
1980-01-31  112.0
1980-02-29  118.0
1980-03-31  129.0
1980-04-30   99.0
1980-05-31  116.0
```

```
Last few rows of Training Data

             Rose
Time_Stamp
1990-08-31   70.0
1990-09-30   83.0
1990-10-31   65.0
1990-11-30  110.0
1990-12-31  132.0
```

Testing data (contains 55 observations)

```
First few rows of Test Data

             Rose
Time_Stamp
1991-01-31   54.0
1991-02-28   55.0
1991-03-31   66.0
1991-04-30   65.0
1991-05-31   60.0
```

```
Last few rows of Test Data

             Rose
Time_Stamp
1995-03-31   45.0
1995-04-30   52.0
1995-05-31   28.0
1995-06-30   40.0
1995-07-31   62.0
```

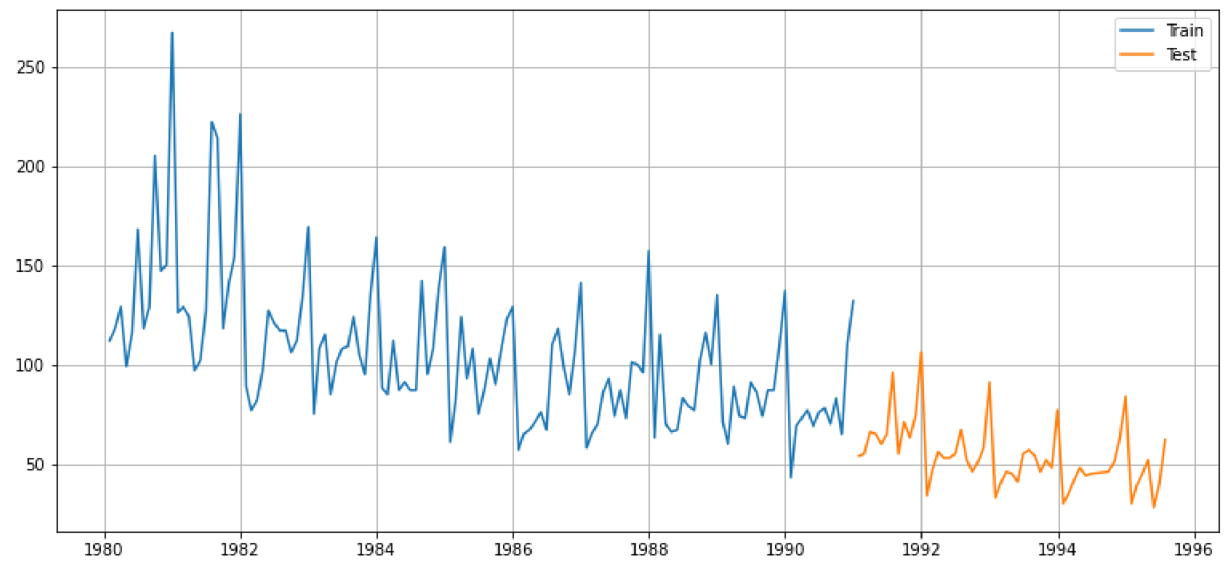


Fig 12: Train/Test Split

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 Rose 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              First few rows of Test Data
              Rose  time                                   Rose  time
Time_Stamp                                   Time_Stamp
1980-01-31  112.0     1                      1991-01-31  54.0    43
1980-02-29  118.0     2                      1991-02-28  55.0    44
1980-03-31  129.0     3                      1991-03-31  66.0    45
1980-04-30   99.0     4                      1991-04-30  65.0    46
1980-05-31  116.0     5                      1991-05-31  60.0    47

Last few rows of Training Data               Last few rows of Test Data
              Rose  time                                   Rose  time
Time_Stamp                                   Time_Stamp
1990-08-31   70.0   128                      1995-03-31  45.0    93
1990-09-30   83.0   129                      1995-04-30  52.0    94
1990-10-31   65.0   130                      1995-05-31  28.0    95
1990-11-30  110.0   131                      1995-06-30  40.0    96
1990-12-31  132.0   132                      1995-07-31  62.0    97
```

This is how the train and test data looks like.

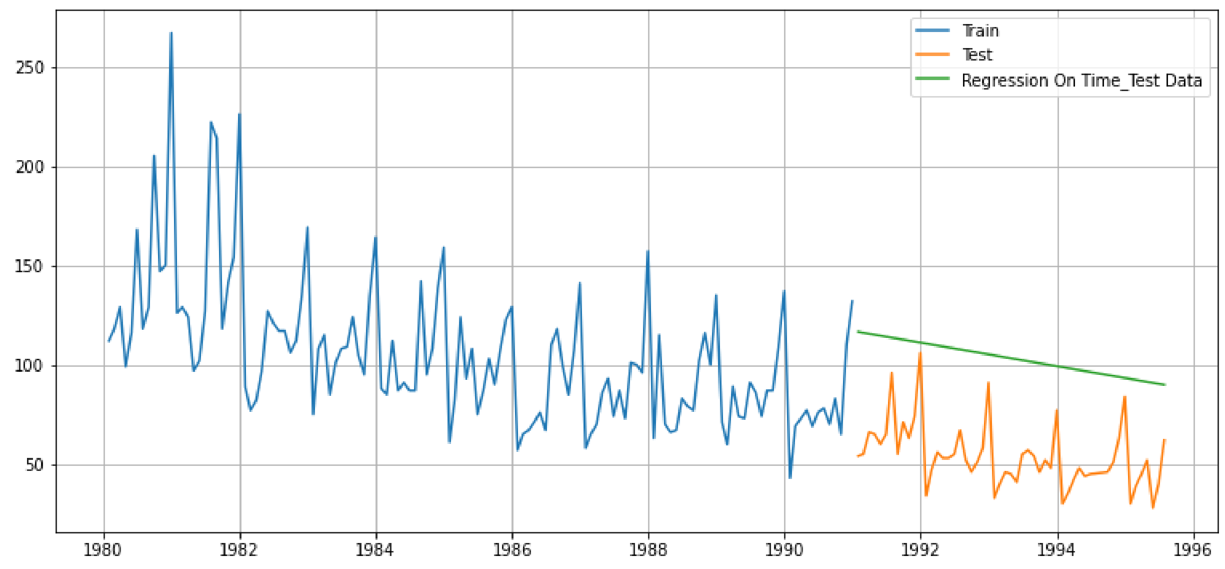


Fig 13: Linear Regression Model Forecast

Model Evaluation:

```
For RegressionOnTime forecast on the Test Data,  RMSE is 51.433
```

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

	Test RMSE
RegressionOnTime	51.433312

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).

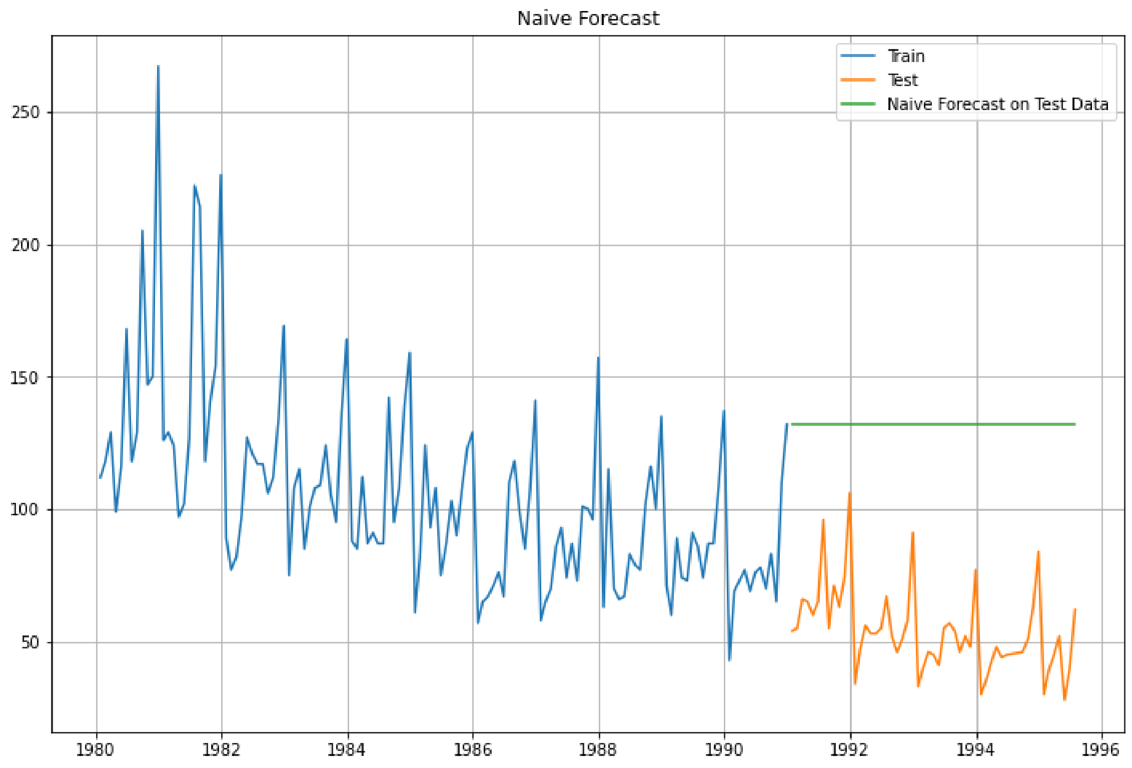


Fig 14: Naive Model Forecast

Model Evaluation:

```
For RegressionOnTime forecast on the Test Data,  RMSE is 79.719
```

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

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

	Test RMSE
RegressionOnTime	51.433312
NaiveModel	79.718773

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.

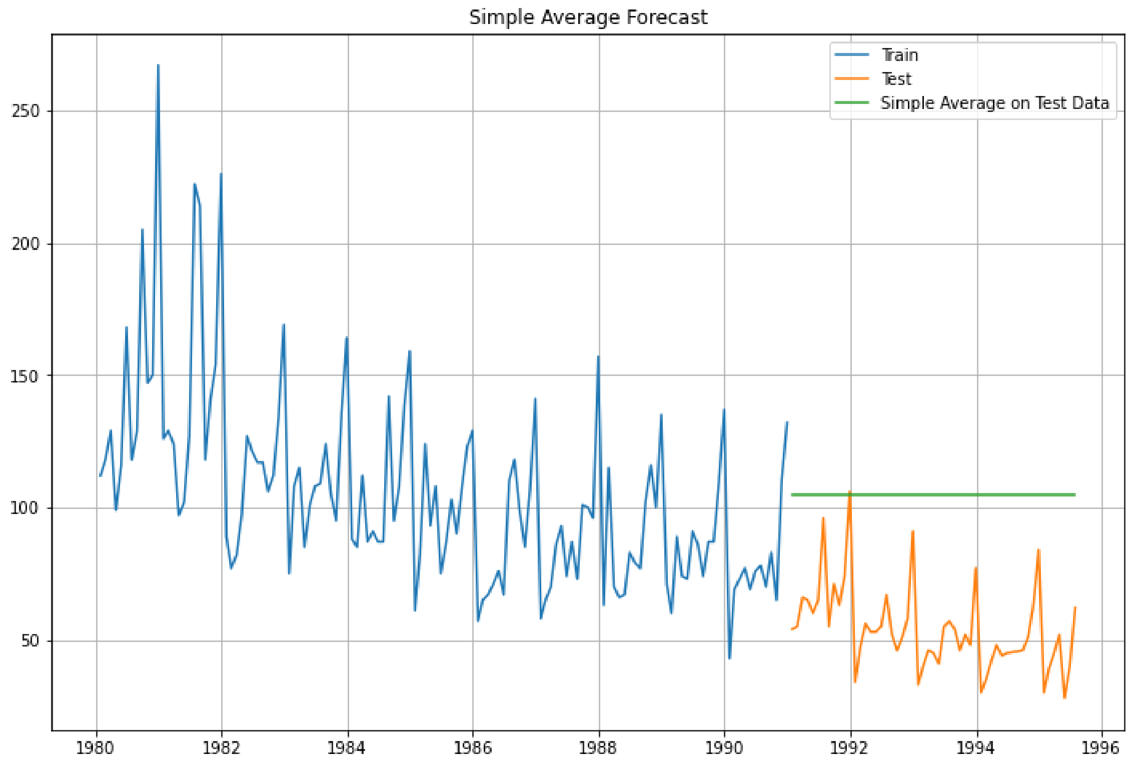


Fig 15: Simple Average Model Forecast

```
For Simple Average forecast on the Test Data,  RMSE is 53.461
```

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

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

	Test RMSE
RegressionOnTime	51.433312
NaiveModel	79.718773
SimpleAverageModel	53.460570

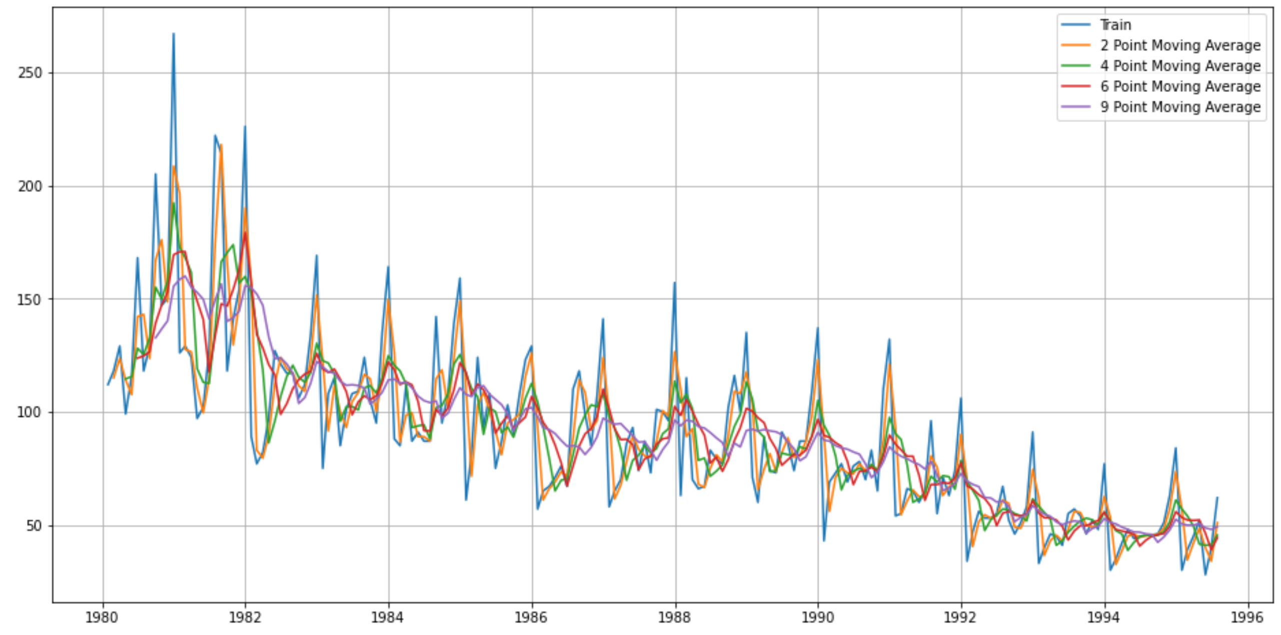
4. MOVING AVERAGE MODEL

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.

	Rose	Trailing_2	Trailing_4	Trailing_6	Trailing_9
Time_Stamp					
1980-01-31	112.0	NaN	NaN	NaN	NaN
1980-02-29	118.0	115.0	NaN	NaN	NaN
1980-03-31	129.0	123.5	NaN	NaN	NaN
1980-04-30	99.0	114.0	114.5	NaN	NaN
1980-05-31	116.0	107.5	115.5	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.

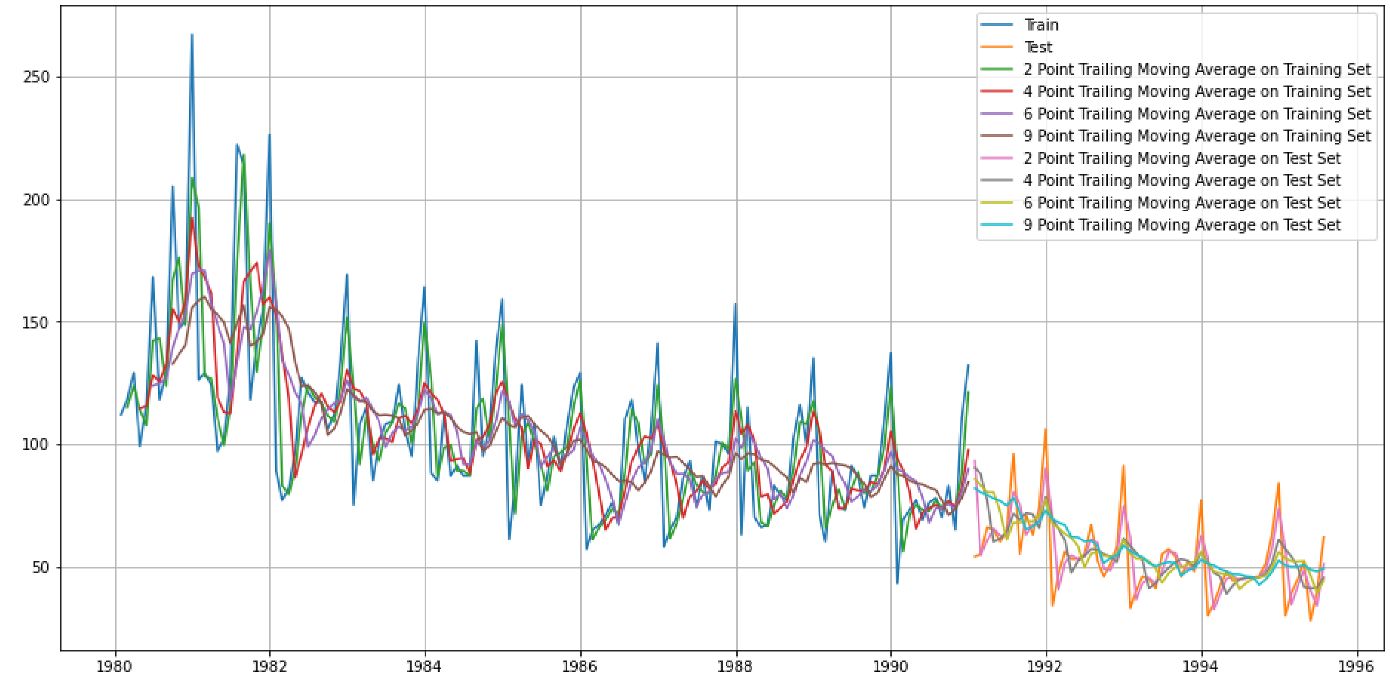


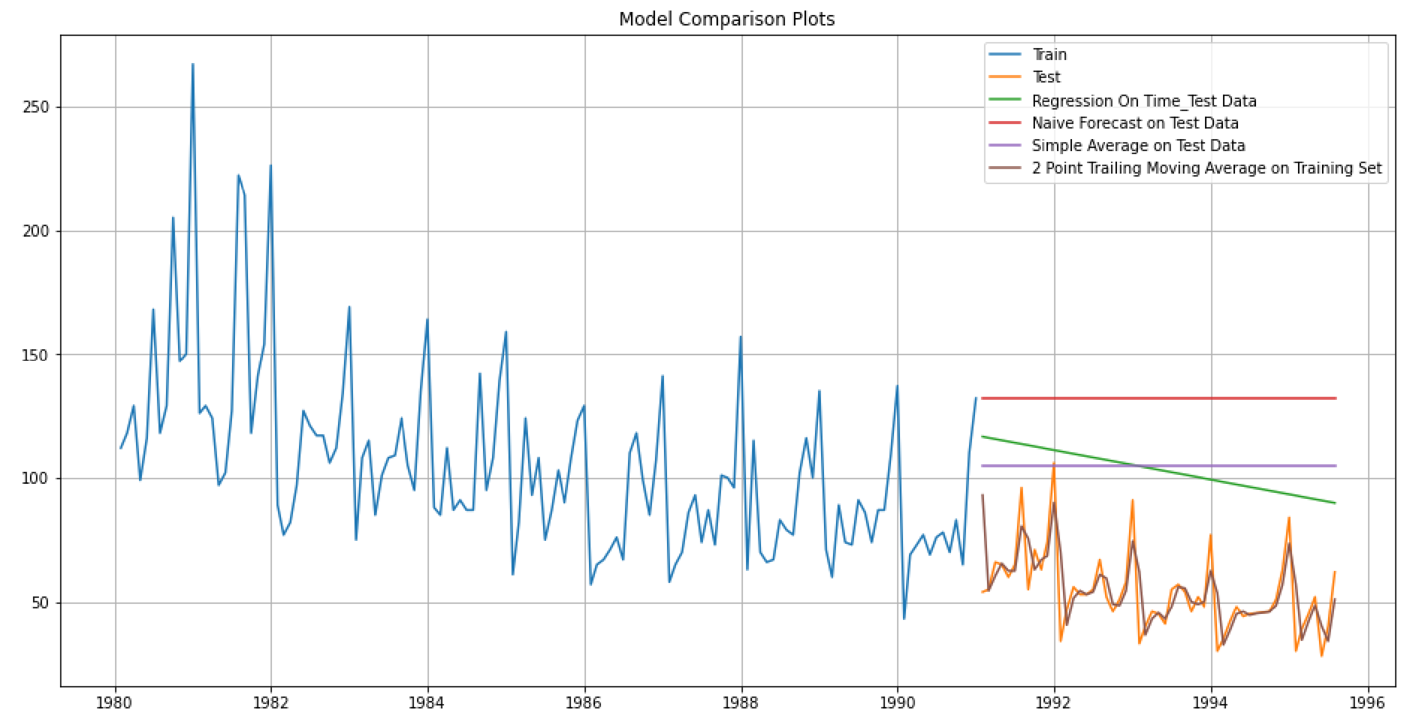
Fig 16: Moving Average Model Forecast

```
For 2 point Moving Average Model forecast on the Training Data,  RMSE is 11.529
For 4 point Moving Average Model forecast on the Training Data,  RMSE is 14.451
For 6 point Moving Average Model forecast on the Training Data,  RMSE is 14.566
For 9 point Moving Average Model forecast on the Training Data,  RMSE is 14.728
```

	Test RMSE
RegressionOnTime	51.433312
NaiveModel	79.718773
SimpleAverageModel	53.460570
2pointTrailingMovingAverage	11.529278
4pointTrailingMovingAverage	14.451403
6pointTrailingMovingAverage	14.566327
9pointTrailingMovingAverage	14.727630

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': 0.0987493111726833,
 'smoothing_trend': nan,
 'smoothing_seasonal': nan,
 'damping_trend': nan,
 'initial_level': 134.38720226208358,
 'initial_trend': nan,
 'initial_seasons': array([], dtype=float64),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

The auto model suggests value of alpha is 0.09874 for which we will predict on our test set.

	Rose	predict
Time_Stamp		
1991-01-31	54.00	87.10
1991-02-28	55.00	87.10
1991-03-31	66.00	87.10
1991-04-30	65.00	87.10
1991-05-31	60.00	87.10

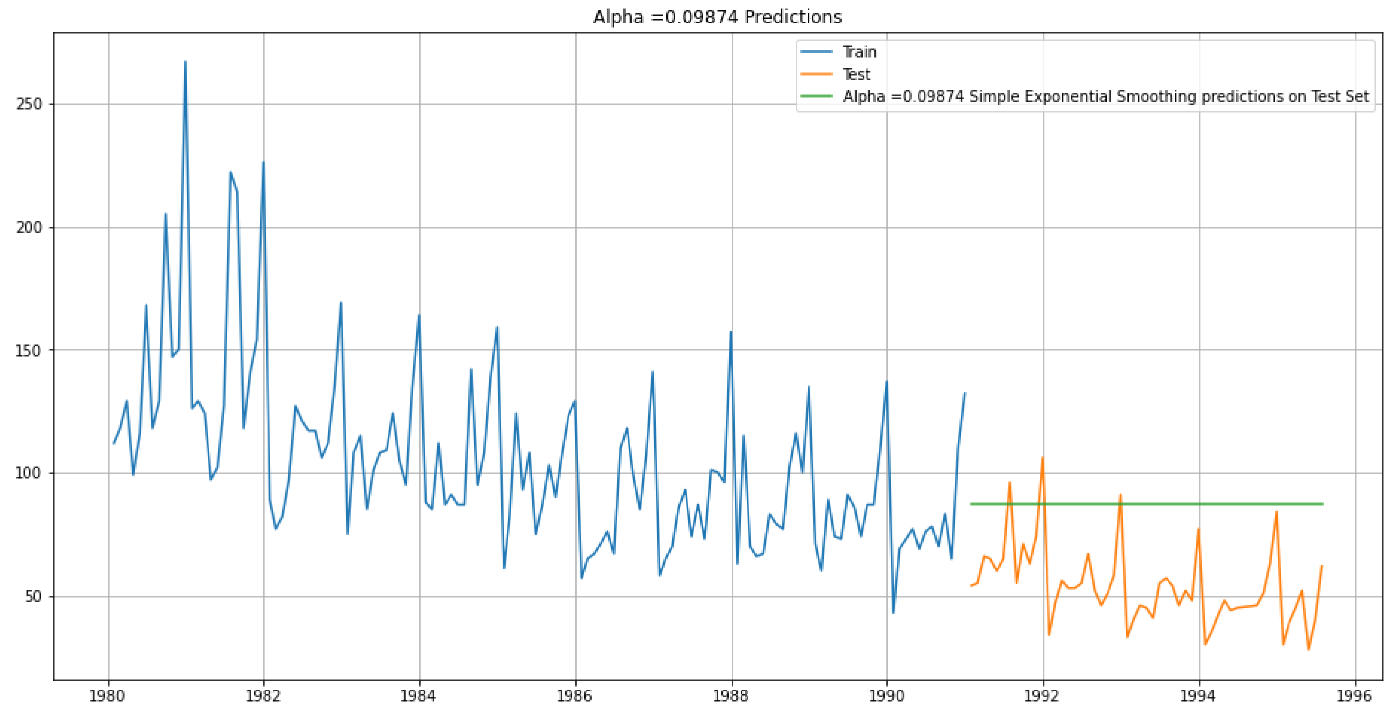


Fig 17: alpha=0.0987 Simple Exponential Smoothing Model Forecast

Model Evaluation:

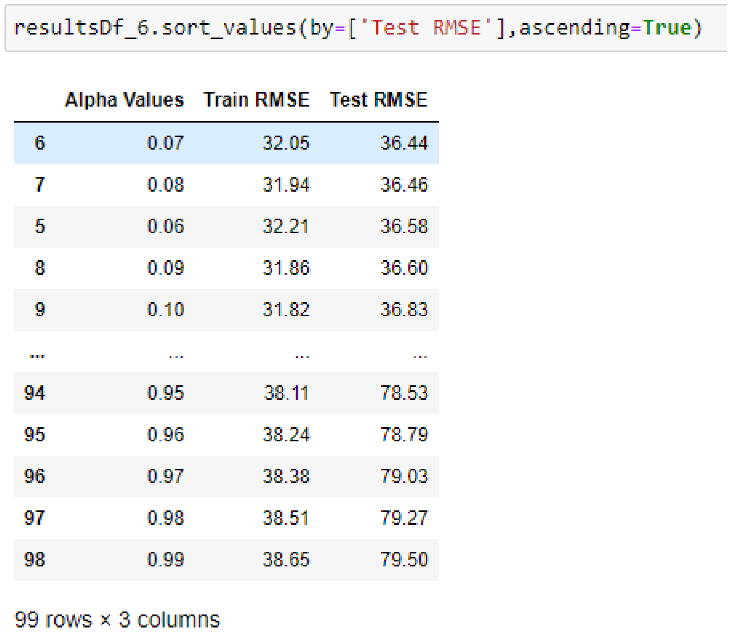
```
For Alpha =0.09874 Simple Exponential Smoothing Model forecast on the Test Data, RMSE is 36.796
```

```
resultsDf_5 = pd.DataFrame({'Test RMSE': [rmse_model5_test_1]},index=['Alpha=0.09874,SimpleExponentialSmoothing'])

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

	Test RMSE
RegressionOnTime	51.43
NaiveModel	79.72
SimpleAverageModel	53.46
2pointTrailingMovingAverage	11.53
4pointTrailingMovingAverage	14.45
6pointTrailingMovingAverage	14.57
9pointTrailingMovingAverage	14.73
Alpha=0.09874,SimpleExponentialSmoothing	36.80

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.



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

	Alpha Values	Train RMSE	Test RMSE
6	0.07	32.05	36.44
7	0.08	31.94	36.46
5	0.06	32.21	36.58
8	0.09	31.86	36.60
9	0.10	31.82	36.83
...	...	...	...
94	0.95	38.11	78.53
95	0.96	38.24	78.79
96	0.97	38.38	79.03
97	0.98	38.51	79.27
98	0.99	38.65	79.50

99 rows × 3 columns

Here, we can observe that alpha =0.07 works best for predicting our series.

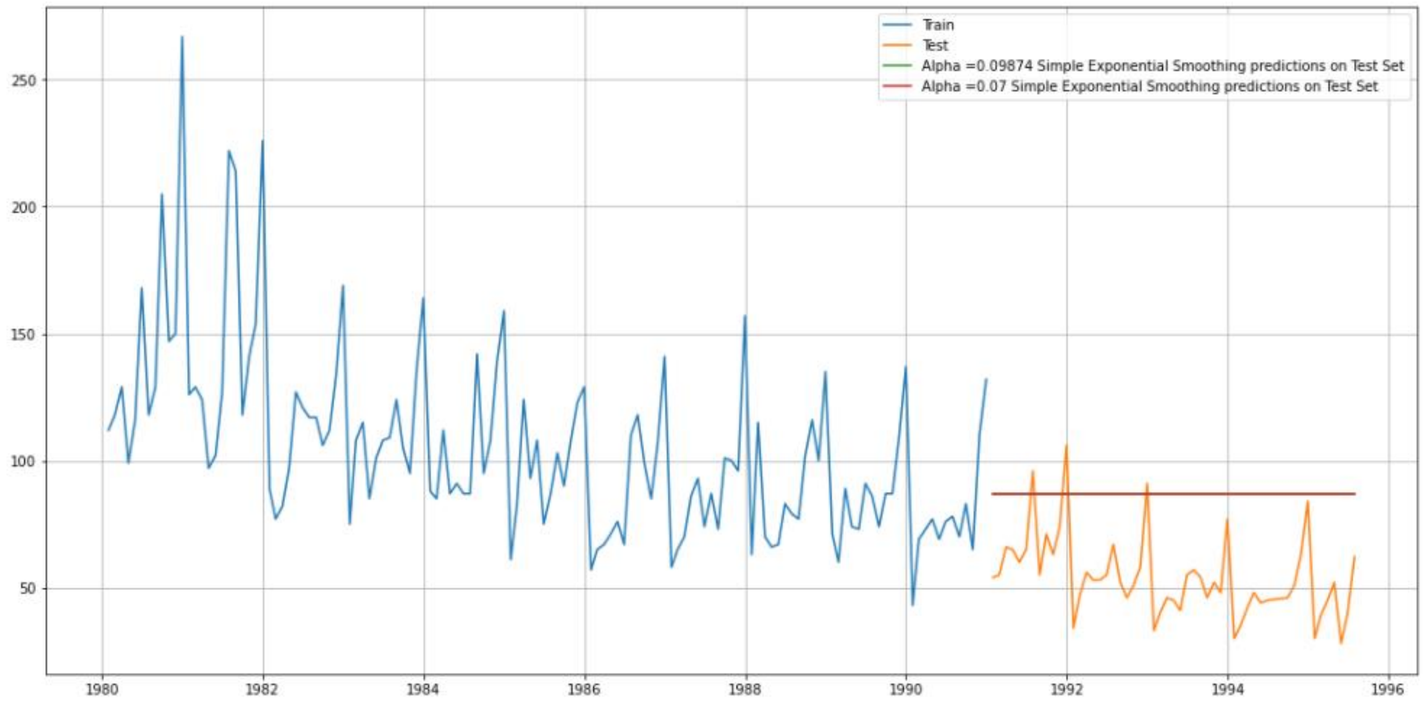


Fig 18: alpha=0.07 Simple Exponential Smoothing Model Forecast

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:

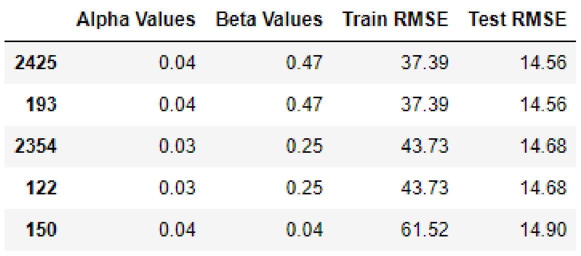
Forecast equation: $\hat{Y}t+1=lt+bt$

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
2425	0.04	0.47	37.39	14.56
193	0.04	0.47	37.39	14.56
2354	0.03	0.25	43.73	14.68
122	0.03	0.25	43.73	14.68
150	0.04	0.04	61.52	14.90

Here, we can observe that alpha =0.04 and beta= 0.47 works best for predicting our series.

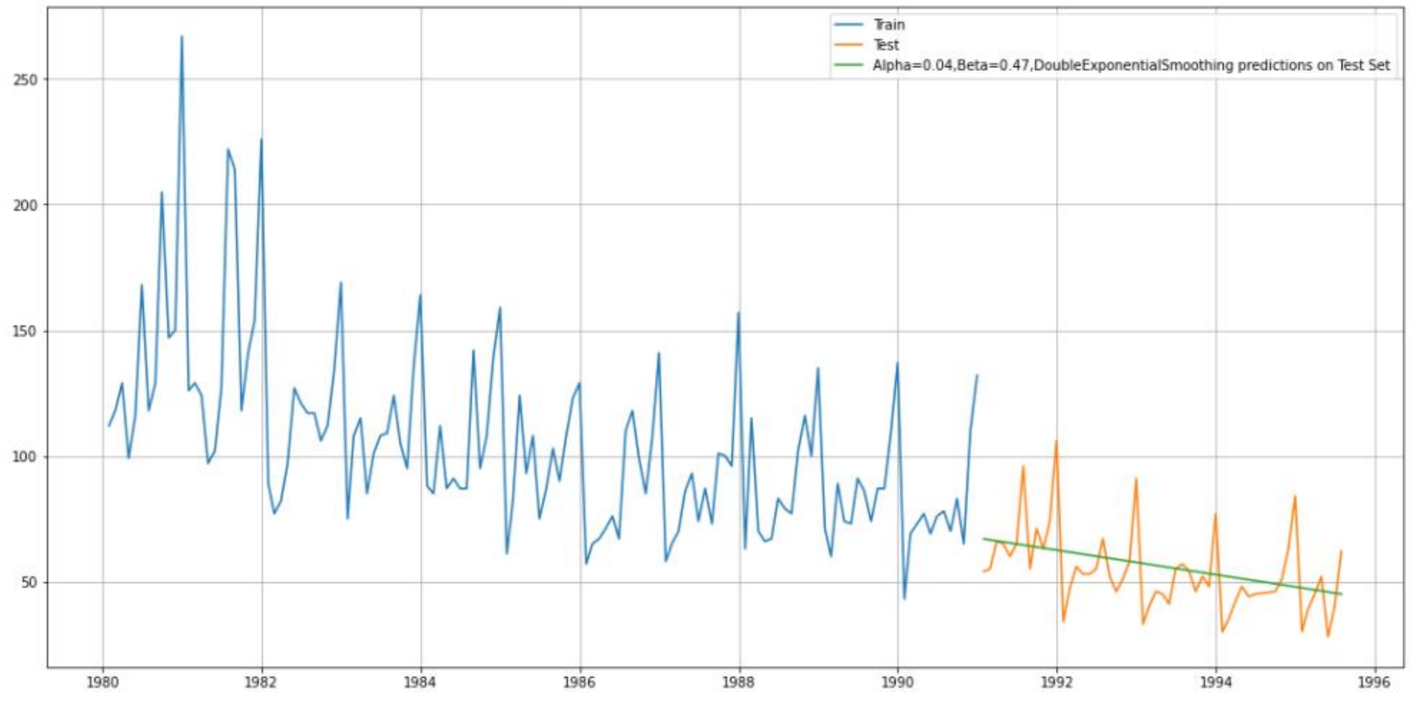


Fig 19: alpha=0.04 and beta=0.47 Double Exponential Smoothing Model Forecast

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.

```
{'smoothing_level': 0.08863830320528249,
 'smoothing_trend': 4.108217096893454e-06,
 'smoothing_seasonal': 5.24162348371366e-08,
 'damping_trend': nan,
 'initial_level': 77.07931882365665,
 'initial_trend': -0.549008945400508,
 'initial_seasons': array([ 38.66747679,  51.00822176,  58.9981034 ,  48.34675   ,
         57.12520488,  62.56658922,  72.49775197,  78.54589082,
         74.55040899,  72.70253166,  90.75516685, 133.03428153]),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

The auto model suggests value of alpha=0.088, beta=4.108 and gamma=5.241 for which we will predict on our test set.

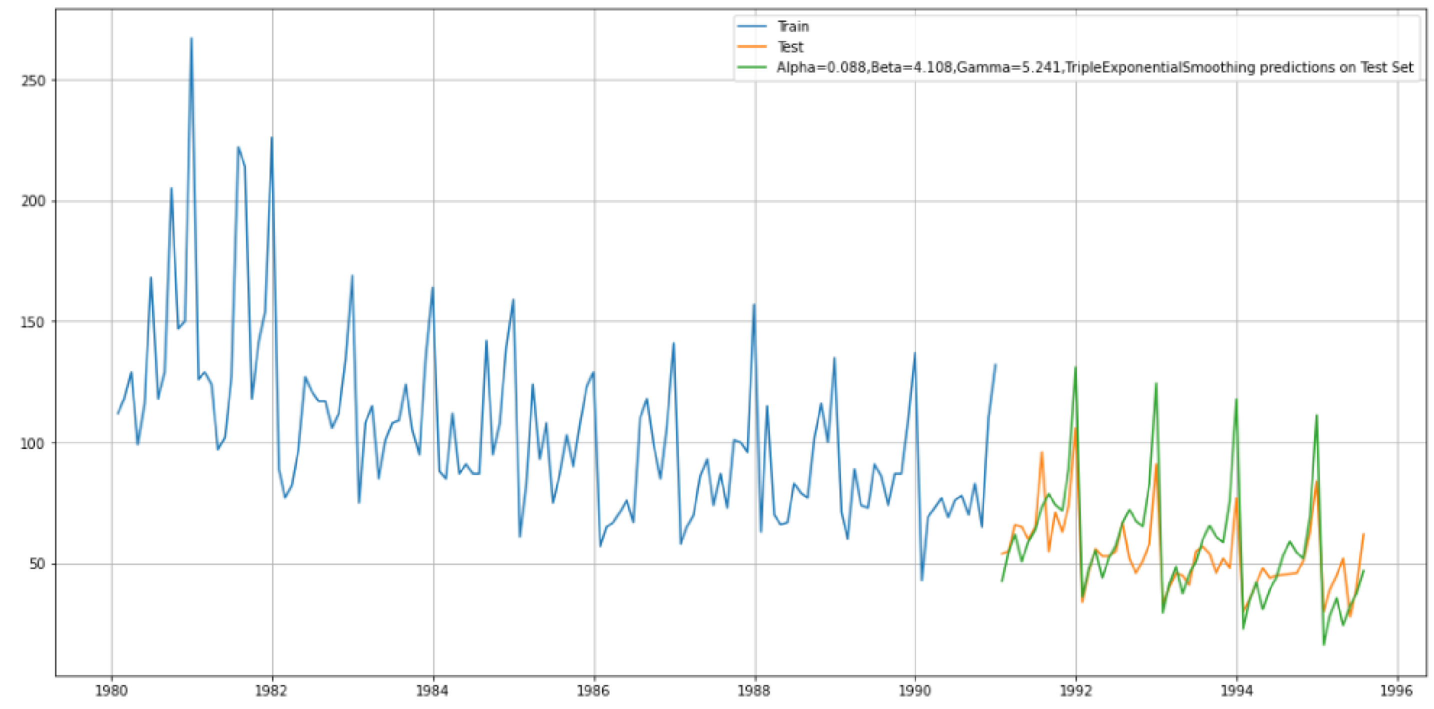


Fig 20: alpha=0.088, beta=4.108 and gamma=5.241 Triple Exponential Smoothing Model Forecast

```
## Test Data

rmse_model6_test_1 = metrics.mean_squared_error(TES_test['Rose'],TES_test['auto_predict'],squared=False)
print("For Alpha=0.088,Beta=4.108,Gamma=5.241, Triple Exponential Smoothing Model forecast on the Test Data,  RMSE is %3.3f" %(rr
```

```
For Alpha=0.088,Beta=4.108,Gamma=5.241, Triple Exponential Smoothing Model forecast on the Test Data,  RMSE is 14.234
```

	Alpha Values	Beta Values	Gamma Values	Train RMSE	Test RMSE
32881	0.00	3.80	3.00	11990.09	68654.22
32801	0.00	3.40	3.00	11990.09	68654.22
32821	0.00	3.50	3.00	11990.09	68654.22
32701	0.00	2.90	3.00	11990.09	68654.22
32641	0.00	2.60	3.00	11990.09	68654.22

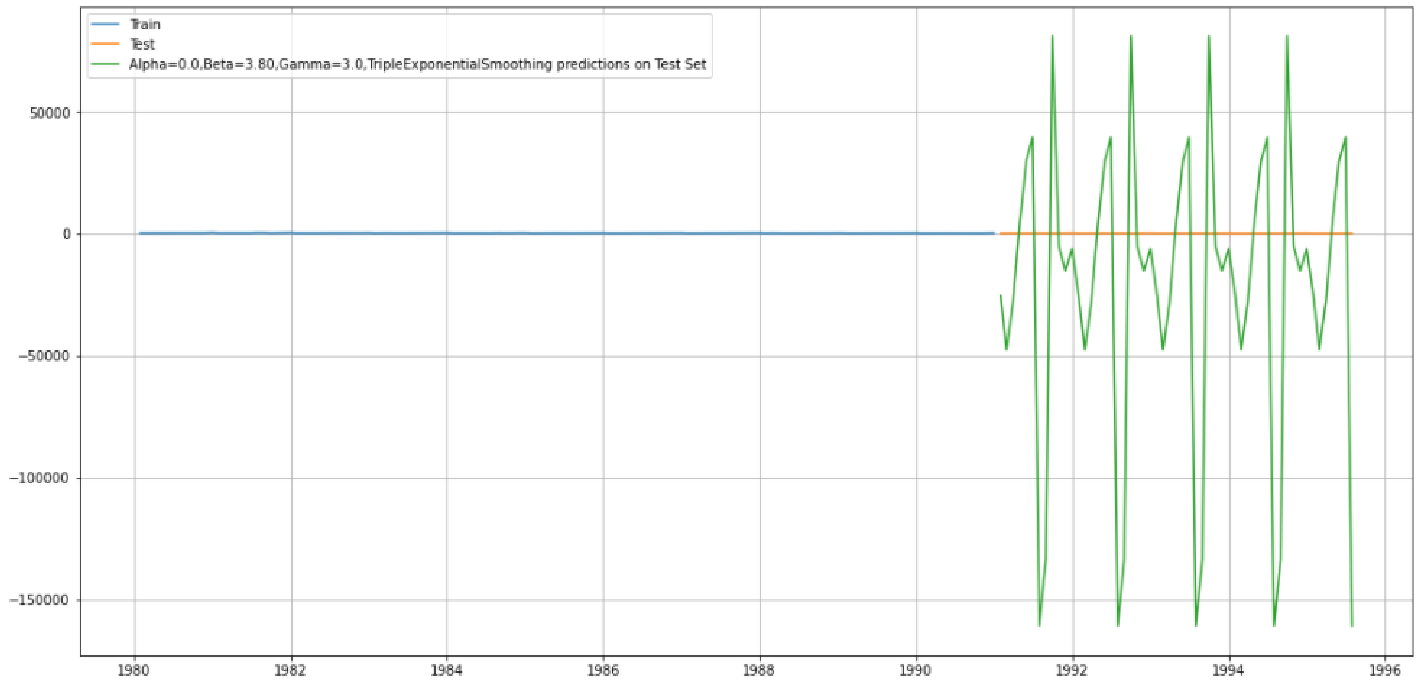


Fig 21: alpha=0.0, beta=.3.80 and gamma=3.0 Triple Exponential Smoothing Model Forecast

The RMSE score for Triple exponential smoothing with parameters Alpha=0.0, Beta=3.80, Gamma=3.0 is the worst out of all the models.

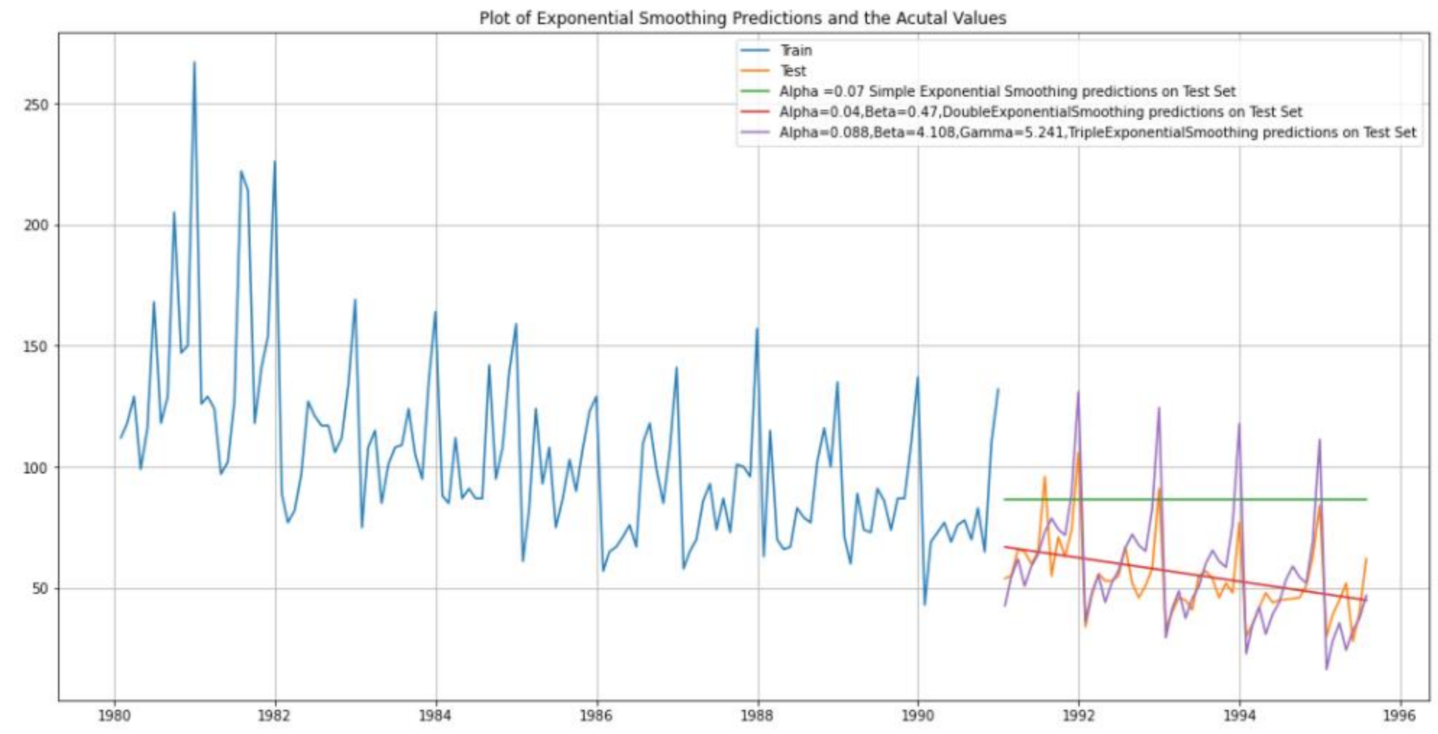


Fig 22: Plot of Exponential Smoothing Predictions and the Actual Values

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.

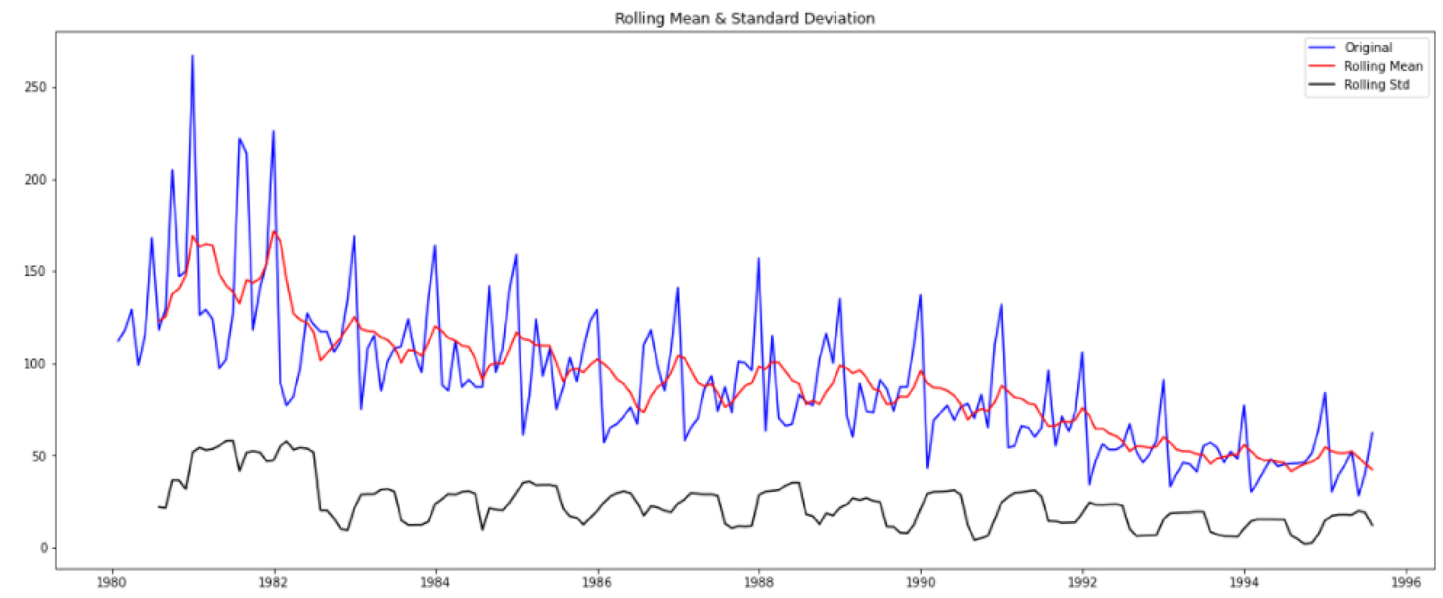
The Augmented Dickey-Fuller (ADF) test is a unit root test which determines whether there is a unit root and subsequently whether the series is non-stationary.

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(Rose_df['Rose'])
```



```
Results of Dickey-Fuller Test:
Test Statistic                 -1.88
p-value                         0.34
#Lags Used                     13.00
Number of Observations Used   173.00
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

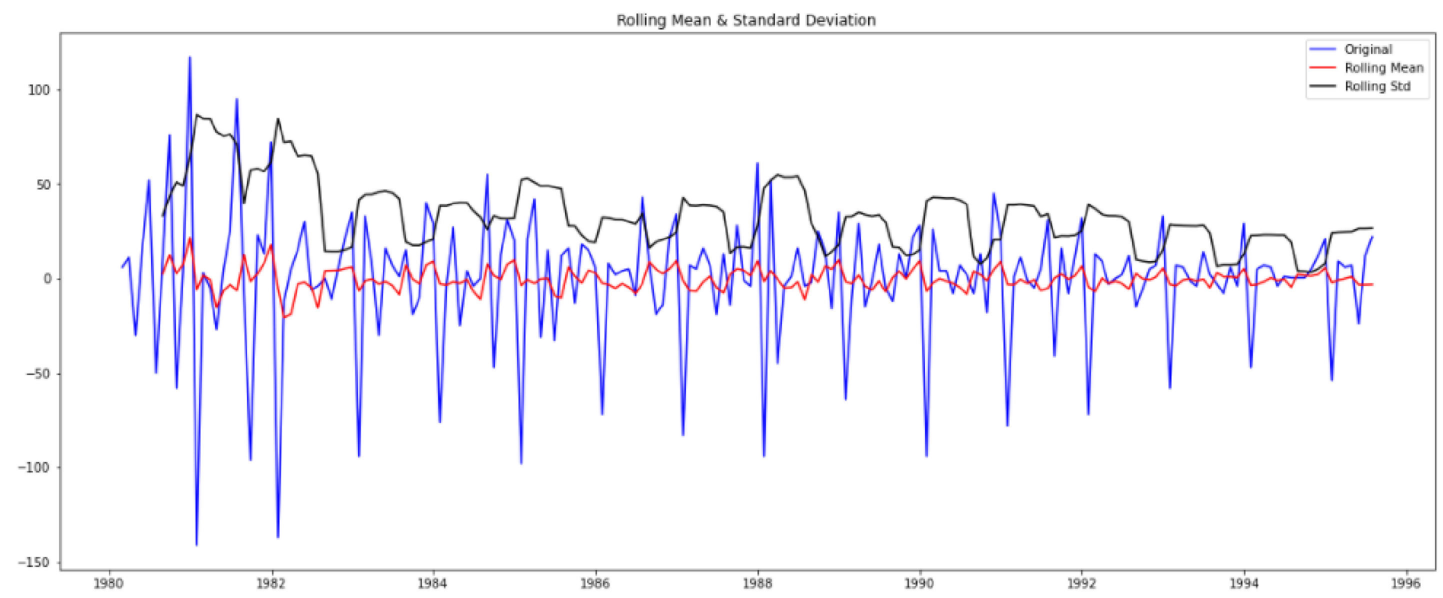
We see that at 5% significant level the Time Series is non-stationary.

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(Rose_df['Rose'].diff().dropna())
```



```
Results of Dickey-Fuller Test:
Test Statistic                  -8.04
p-value                          0.00
#Lags Used                      12.00
Number of Observations Used    173.00
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 $\alpha = 0.05$ the Time Series is indeed stationary.
We see that after taking a difference of order 1 the series have become stationary at $\alpha = 0.05$.

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.

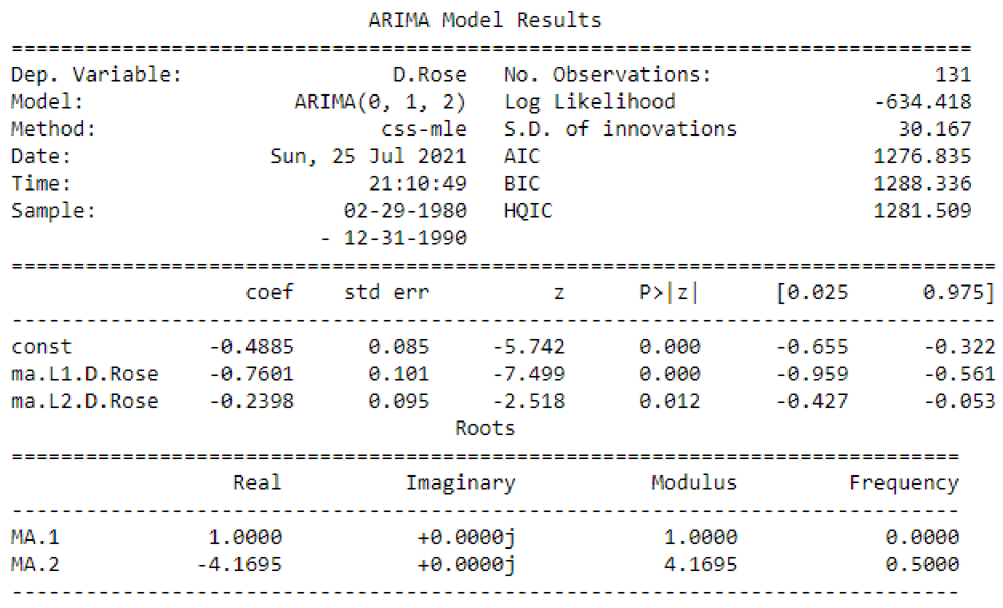
ARIMA has three components – AR (autoregressive term), I (differencing term) and MA (moving average term)–

- 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.

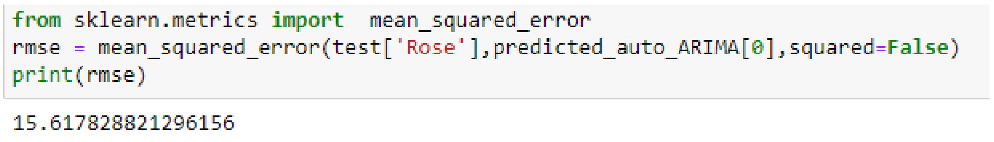
Before implementing ARIMA, we need to make the series stationary, and determine the values of p and q using the plots. Auto ARIMA makes this task really simple for us as it eliminates few steps. Below are the steps you should follow for implementing auto ARIMA:

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 (0,1,2) upon fitting auto ARIMA based on AIC value.



```
                             ARIMA Model Results
==============================================================================
Dep. Variable:                 D.Rose   No. Observations:                  131
Model:                 ARIMA(0, 1, 2)   Log Likelihood                -634.418
Method:                       css-mle   S.D. of innovations             30.167
Date:                Sun, 25 Jul 2021   AIC                           1276.835
Time:                        21:10:49   BIC                           1288.336
Sample:                    02-29-1980   HQIC                          1281.509
                         - 12-31-1990
================================================================================
                   coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
const           -0.4885      0.085     -5.742      0.000      -0.655      -0.322
ma.L1.D.Rose    -0.7601      0.101     -7.499      0.000      -0.959      -0.561
ma.L2.D.Rose    -0.2398      0.095     -2.518      0.012      -0.427      -0.053
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
MA.1            1.0000           +0.0000j            1.0000            0.0000
MA.2           -4.1695           +0.0000j            4.1695            0.5000
-----------------------------------------------------------------------------
```



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

```
15.617828821296156
```

The RMSE value for auto ARIMA (0,1,2) is 15.617

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.

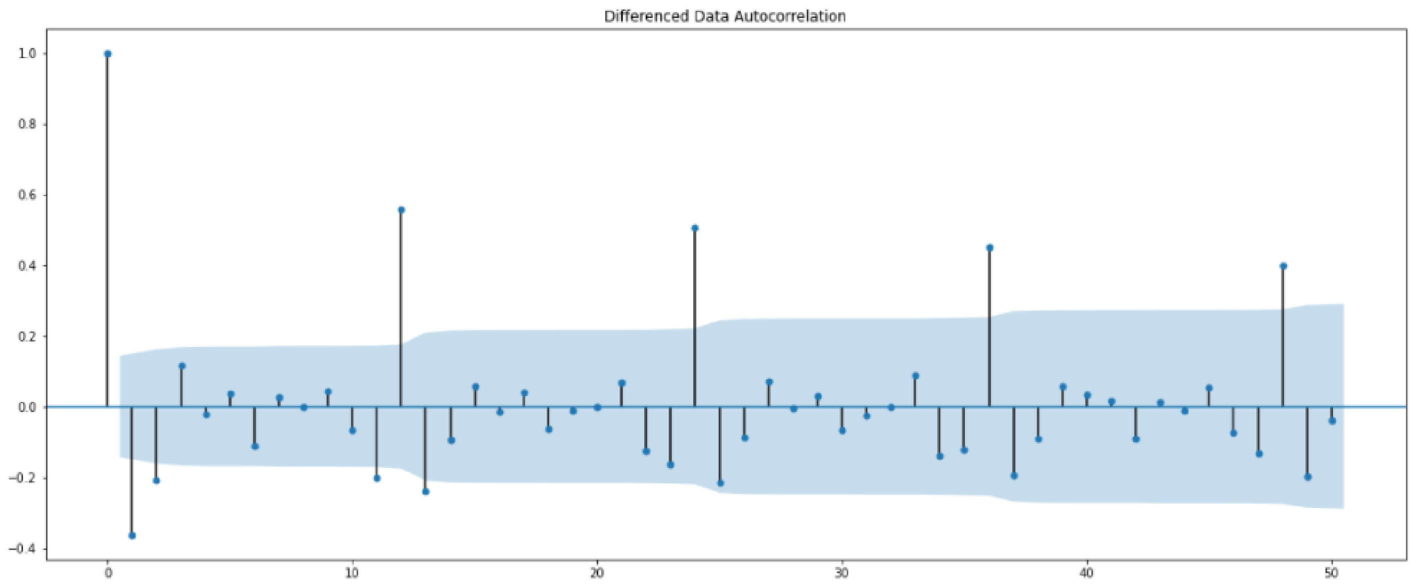
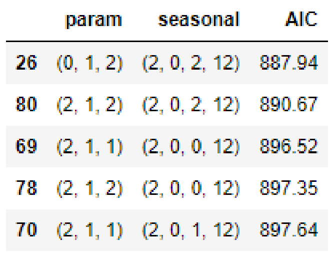


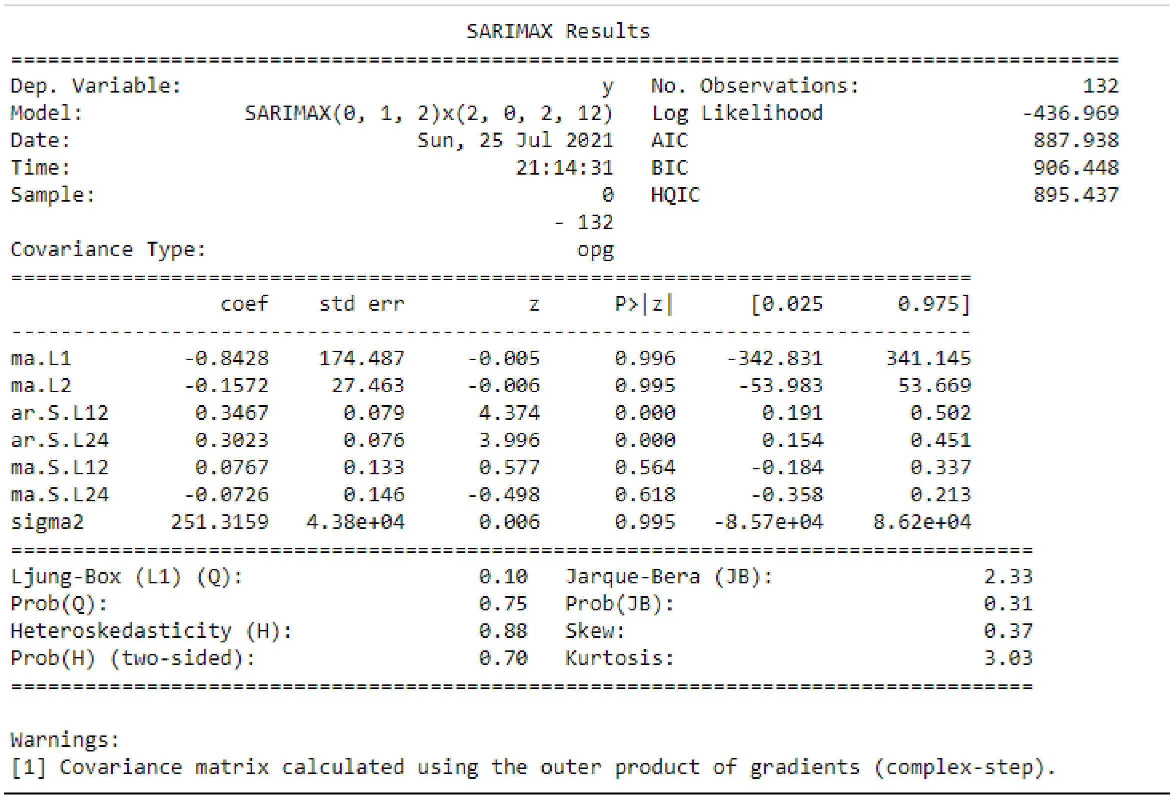
Fig 25: Differenced Auto Correlation Plot

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.



	param	seasonal	AIC
26	(0, 1, 2)	(2, 0, 2, 12)	887.94
80	(2, 1, 2)	(2, 0, 2, 12)	890.67
69	(2, 1, 1)	(2, 0, 0, 12)	896.52
78	(2, 1, 2)	(2, 0, 0, 12)	897.35
70	(2, 1, 1)	(2, 0, 1, 12)	897.64



```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                                 y   No. Observations:                  132
Model:             SARIMAX(0, 1, 2)x(2, 0, 2, 12)   Log Likelihood                -436.969
Date:                           Sun, 25 Jul 2021   AIC                            887.938
Time:                                   21:14:31   BIC                            906.448
Sample:                                        0   HQIC                           895.437
                                           - 132
Covariance Type:                             opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.8428    174.487     -0.005      0.996    -342.831     341.145
ma.L2         -0.1572     27.463     -0.006      0.995     -53.983      53.669
ar.S.L12       0.3467      0.079      4.374      0.000       0.191       0.502
ar.S.L24       0.3023      0.076      3.996      0.000       0.154       0.451
ma.S.L12       0.0767      0.133      0.577      0.564      -0.184       0.337
ma.S.L24      -0.0726      0.146     -0.498      0.618      -0.358       0.213
sigma2       251.3159   4.38e+04      0.006      0.995   -8.57e+04    8.62e+04
===================================================================================
Ljung-Box (L1) (Q):                   0.10   Jarque-Bera (JB):                 2.33
Prob(Q):                              0.75   Prob(JB):                         0.31
Heteroskedasticity (H):               0.88   Skew:                             0.37
Prob(H) (two-sided):                  0.70   Kurtosis:                         3.03
===================================================================================

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

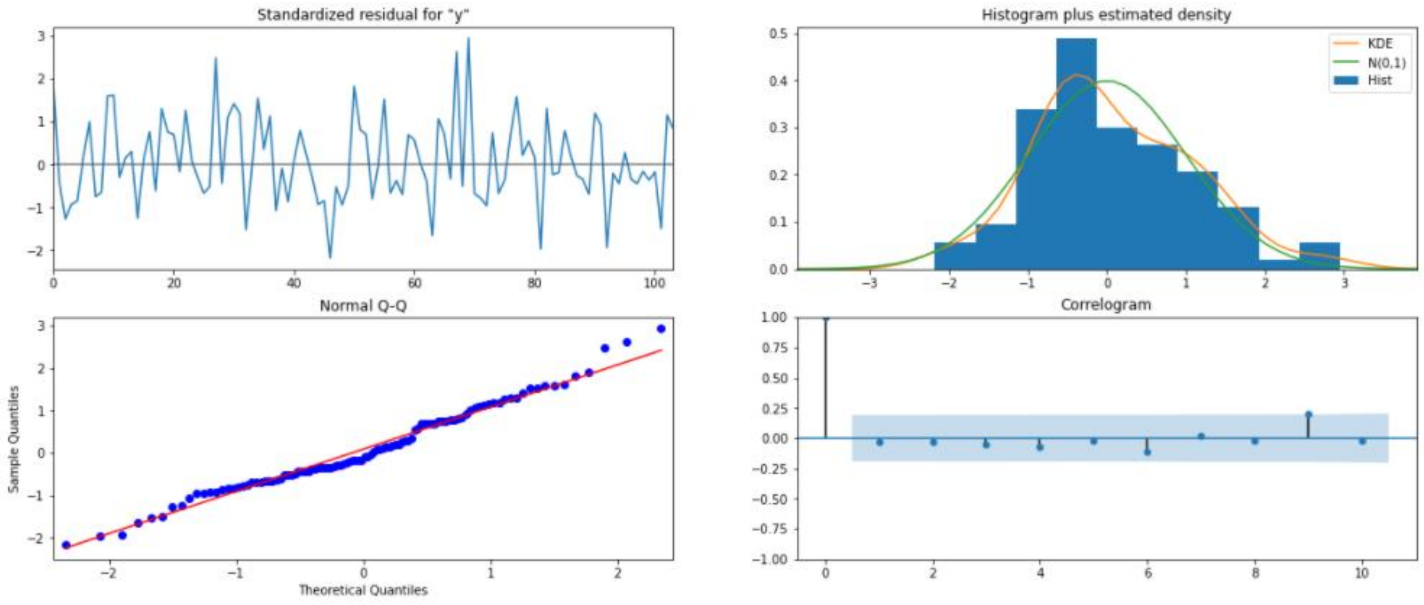


Fig 26: Model Diagnostics SARIMA (0,1,2)X(2,0,2,12)

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 (0,1,2) x(2,0,2,12)

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

y	mean	mean_se	mean_ci_lower	mean_ci_upper
0	62.87	15.93	31.65	94.09
1	70.54	16.15	38.89	102.19
2	77.36	16.15	45.71	109.01
3	76.21	16.15	44.56	107.86
4	72.75	16.15	41.10	104.40

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

```
26.92936799005941
```

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

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

	param	seasonal	AIC
23	(0, 1, 2)	(1, 0, 2, 24)	687.41
50	(1, 1, 2)	(1, 0, 2, 24)	689.08
26	(0, 1, 2)	(2, 0, 2, 24)	689.16
77	(2, 1, 2)	(1, 0, 2, 24)	690.14
53	(1, 1, 2)	(2, 0, 2, 24)	690.80

```
                                      SARIMAX Results
==========================================================================================
Dep. Variable:                                  y   No. Observations:                  132
Model:             SARIMAX(0, 1, 2)x(1, 0, 2, 24)   Log Likelihood                -337.703
Date:                            Sun, 25 Jul 2021   AIC                            687.405
Time:                                    21:18:49   BIC                            701.697
Sample:                                         0   HQIC                           693.135
                                            - 132
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.9642      0.237     -4.074      0.000      -1.428      -0.500
ma.L2         -0.0566      0.139     -0.408      0.684      -0.329       0.215
ar.S.L24       0.8743      0.051     17.145      0.000       0.774       0.974
ma.S.L24      -0.9389      0.277     -3.391      0.001      -1.482      -0.396
ma.S.L48      -0.0611      0.194     -0.315      0.753      -0.441       0.319
sigma2       167.0885      0.002   1.01e+05      0.000     167.085     167.092
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                 3.78
Prob(Q):                              0.96   Prob(JB):                         0.15
Heteroskedasticity (H):               0.70   Skew:                             0.53
Prob(H) (two-sided):                  0.35   Kurtosis:                         3.18
===================================================================================
```

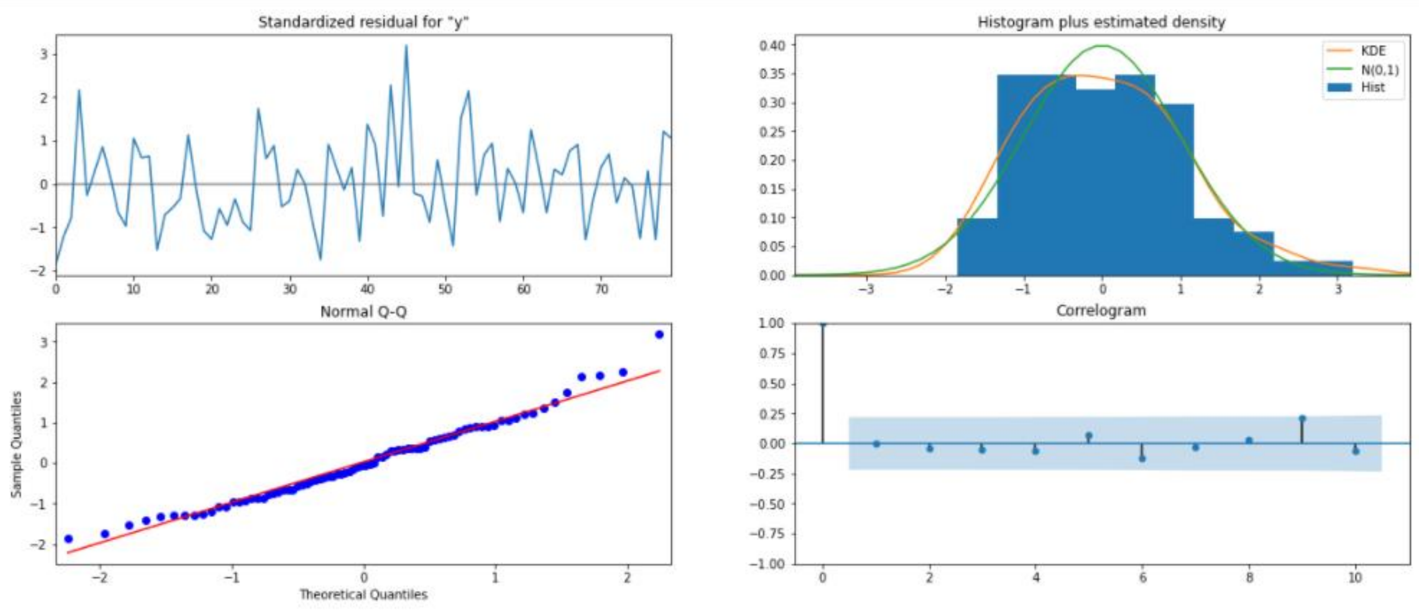


Fig 27: Model Diagnostics SARIMA (0,1,2)X(1,0,2,24)

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.

<u>Model Evaluation for SARIMA (0,1,2) x(1,0,2,24)</u>

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

y	mean	mean_se	mean_ci_lower	mean_ci_upper
0	57.52	14.80	28.52	86.52
1	66.25	14.85	37.15	95.35
2	80.70	14.85	51.59	109.81
3	68.60	14.85	39.49	97.71
4	75.32	14.86	46.20	104.44

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

```
19.93114544255051
```

We see that the RMSE value have 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.

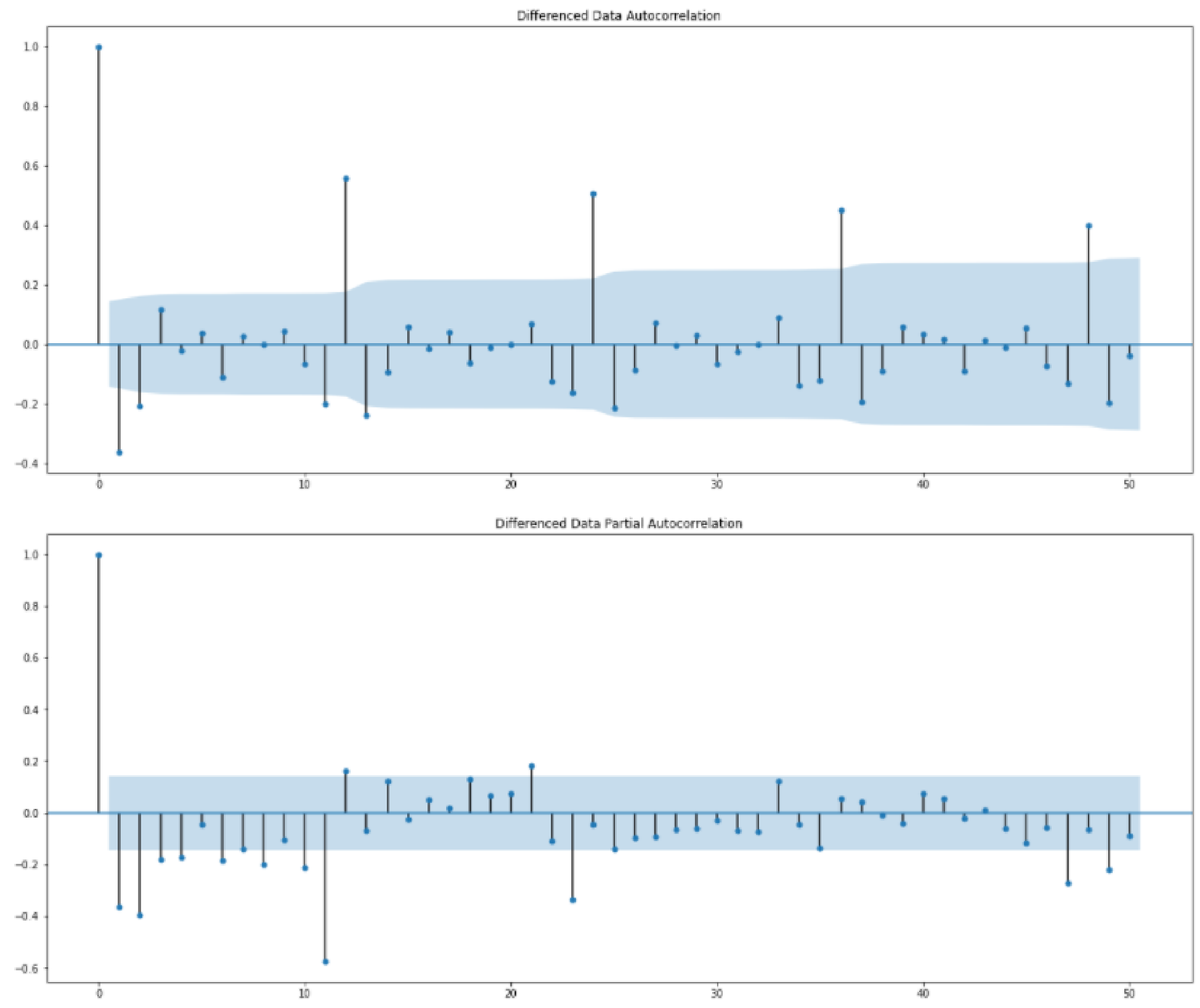


Fig 28: ACF/PACF Plots

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:                   Rose   No. Observations:                  132
Model:                     ARMA(0, 0)   Log Likelihood                -660.450
Method:                           css   S.D. of innovations             36.034
Date:                Sun, 25 Jul 2021   AIC                           1324.900
Time:                        21:23:01   BIC                           1330.665
Sample:                    01-31-1980   HQIC                          1327.243
                         - 12-31-1990
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const        104.9394      3.136     33.459      0.000      98.792     111.087
==============================================================================
```

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

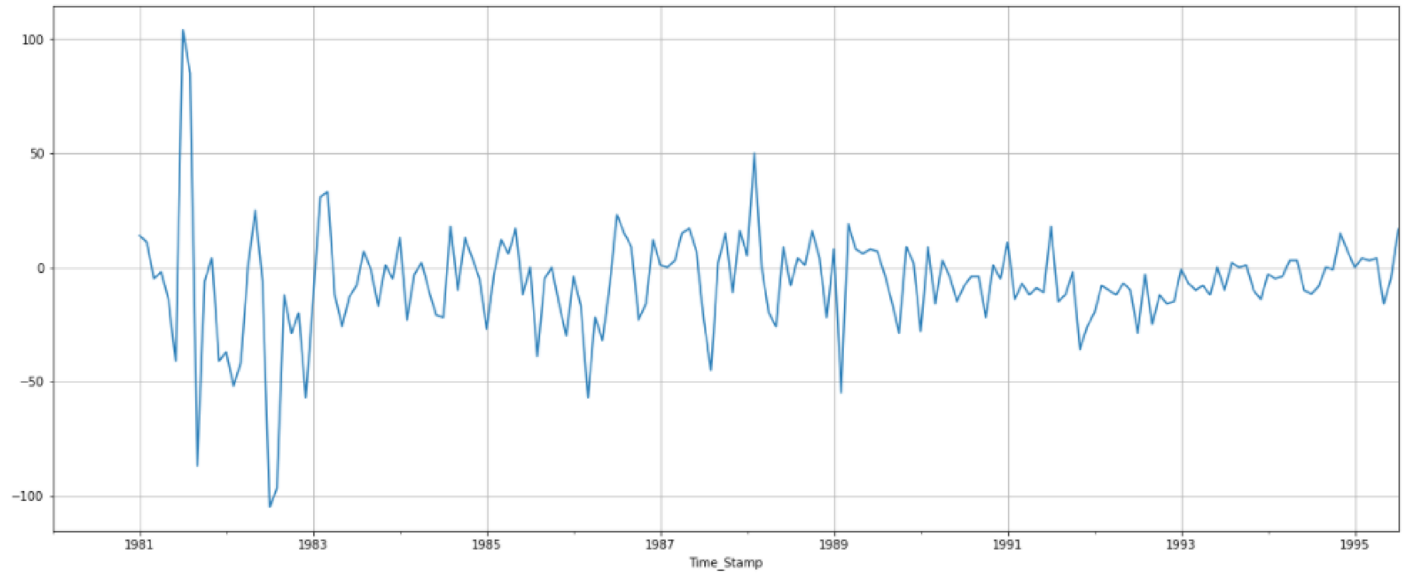
```
53.46056964611445
```

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)

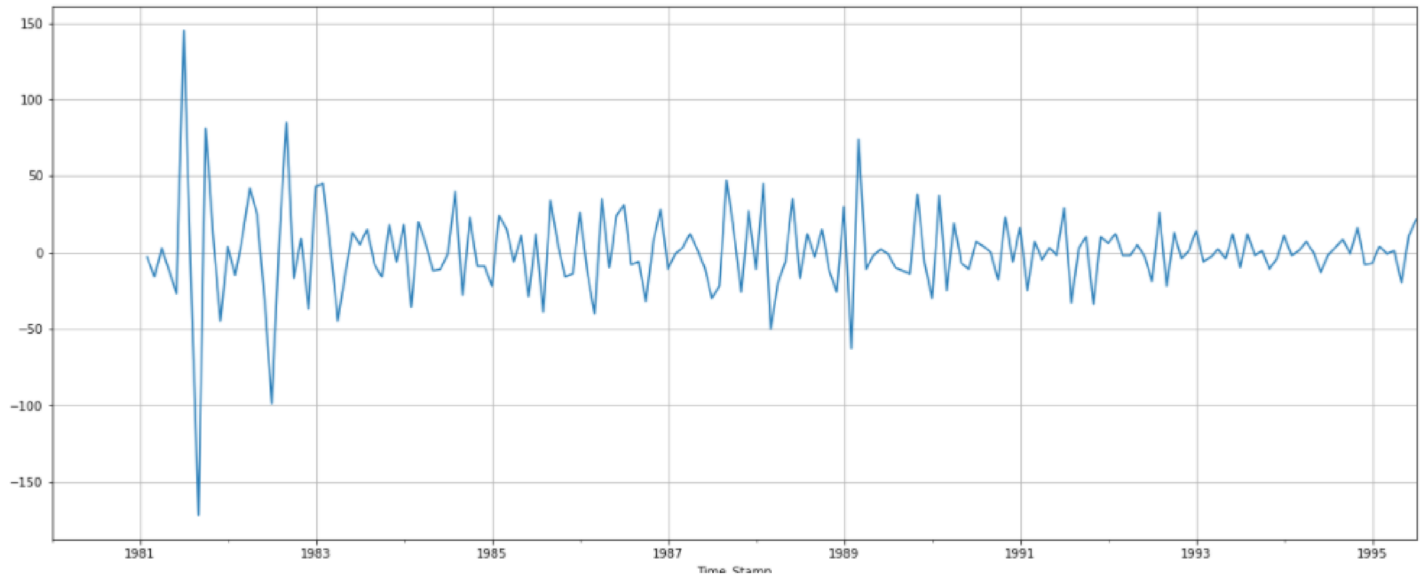
We see that our ACF plot at the seasonal interval (12) does not taper off. So, we go ahead and take a seasonal differencing of the original series.

```
(Rose_df['Rose'].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.

```
(Rose_df['Rose'].diff(12)).diff().plot()
plt.grid();
```



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.

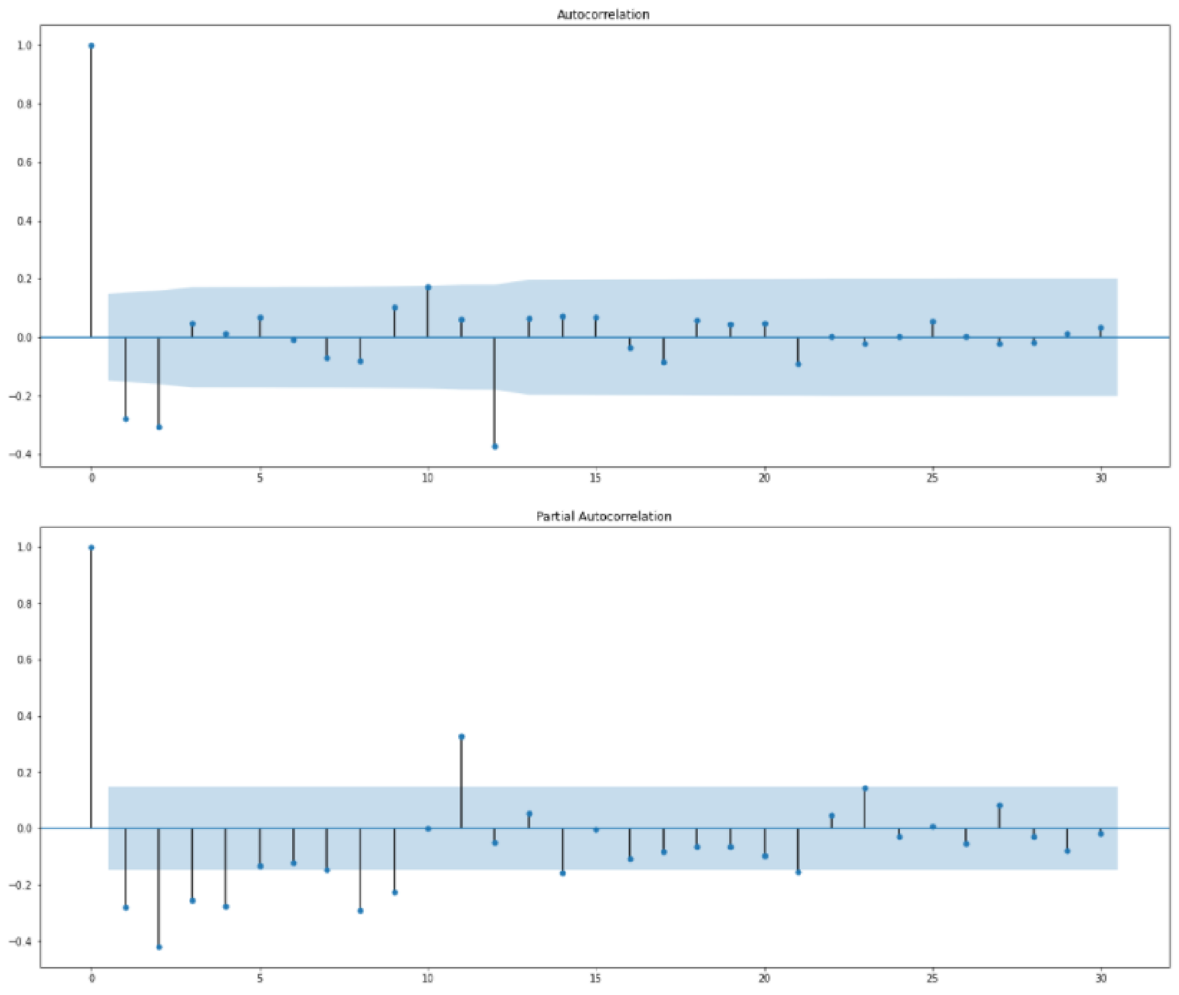


Fig 29: Modified Time Series ACF/PACF Plots

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                -557.449
Date:                    Sun, 25 Jul 2021   AIC                           1116.899
Time:                            21:27:53   BIC                           1119.572
Sample:                                 0   HQIC                          1117.982
                                    - 132
Covariance Type:                      opg
============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------
sigma2      1961.6675    135.590     14.468      0.000    1695.916    2227.419
===================================================================================
Ljung-Box (L1) (Q):                   6.38   Jarque-Bera (JB):               181.93
Prob(Q):                              0.01   Prob(JB):                         0.00
Heteroskedasticity (H):               0.24   Skew:                            -1.39
Prob(H) (two-sided):                  0.00   Kurtosis:                         8.75
===================================================================================
```

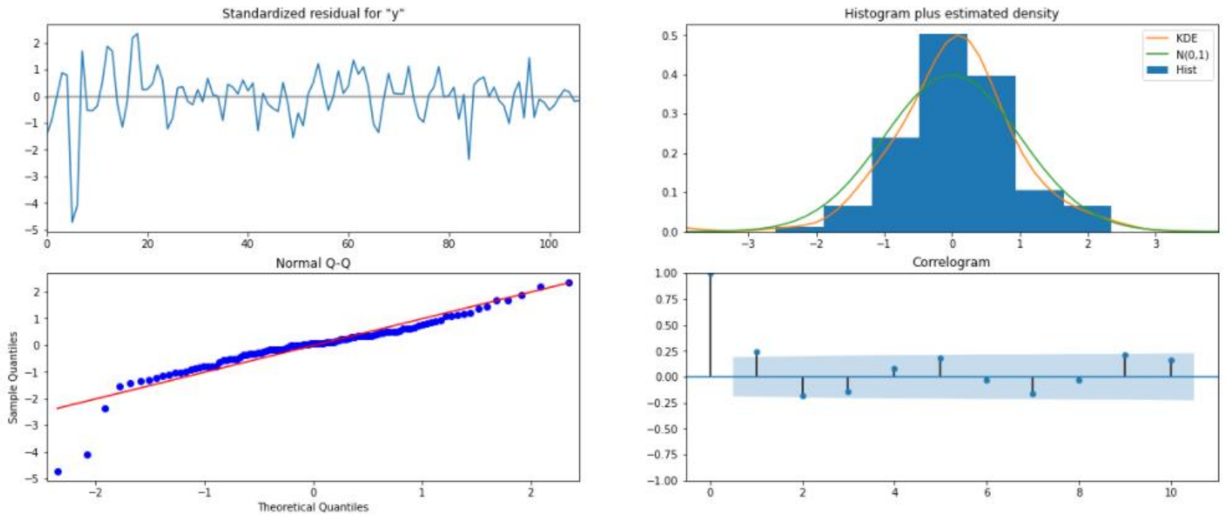


Fig 30: Model Diagnostics SARIMA (0,0,0)X(0,2,0,12)

```
predicted_manual_SARIMA_12.summary_frame(alpha=0.05).head()
```

y	mean	mean_se	mean_ci_lower	mean_ci_upper
0	15.00	44.29	-71.81	101.81
1	78.00	44.29	-8.81	164.81
2	57.00	44.29	-29.81	143.81
3	80.00	44.29	-6.81	166.81
4	65.00	44.29	-21.81	151.81

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

```
41.231252244091706
```

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	51.43
NaiveModel	79.72
SimpleAverageModel	53.46
2pointTrailingMovingAverage	11.53
4pointTrailingMovingAverage	14.45
6pointTrailingMovingAverage	14.57
9pointTrailingMovingAverage	14.73
Alpha=0.09874,SimpleExponentialSmoothing	36.80
Alpha=0.07,SimpleExponentialSmoothing	36.44
Alpha=0.04,Beta=0.47,DoubleExponentialSmoothing	14.56
Alpha=0.088,Beta=4.108,Gamma=5.241,TripleExponentialSmoothing	14.23
Alpha=0.0,Beta=3.80,Gamma=3.0,TripleExponentialSmoothing	68654.22
ARIMA(0,1,2)	15.62
ARIMA(1,1,1)	15.73
SARIMA(0,1,2)(2,0,2,12)	26.93
SARIMA(0,1,2)(1,0,2,24)	19.93
SARIMA(0,0,0)(0,2,0,12)/ACF-PACF	41.23

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

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

	Test RMSE
2pointTrailingMovingAverage	11.53
Alpha=0.088,Beta=4.108,Gamma=5.241,TripleExponentialSmoothing	14.23
4pointTrailingMovingAverage	14.45
Alpha=0.04,Beta=0.47,DoubleExponentialSmoothing	14.56
6pointTrailingMovingAverage	14.57
9pointTrailingMovingAverage	14.73
ARIMA(0,1,2)	15.62
ARIMA(1,1,1)	15.73
SARIMA(0,1,2)(1,0,2,24)	19.93
SARIMA(0,1,2)(2,0,2,12)	26.93
Alpha=0.07,SimpleExponentialSmoothing	36.44
Alpha=0.09874,SimpleExponentialSmoothing	36.80
SARIMA(0,0,0)(0,2,0,12)/ACF-PACF	41.23
RegressionOnTime	51.43
SimpleAverageModel	53.46
NaiveModel	79.72
Alpha=0.0,Beta=3.80,Gamma=3.0,TripleExponentialSmoothing	68654.22

Here, we can conclude that 2 point trailing Moving Average Model is the best model built so far with RMSE value as 11.53

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 2 point trailing Moving Average Model. Therefore, we will build the full model on this.

```
MovingAverage.head()
```

Time_Stamp	Rose	Trailing_2	Trailing_4	Trailing_6	Trailing_9
1980-01-31	112.00	nan	nan	nan	nan
1980-02-29	118.00	115.00	nan	nan	nan
1980-03-31	129.00	123.50	nan	nan	nan
1980-04-30	99.00	114.00	114.50	nan	nan
1980-05-31	116.00	107.50	115.50	nan	nan

The best model gives an RMSE score of 11.53

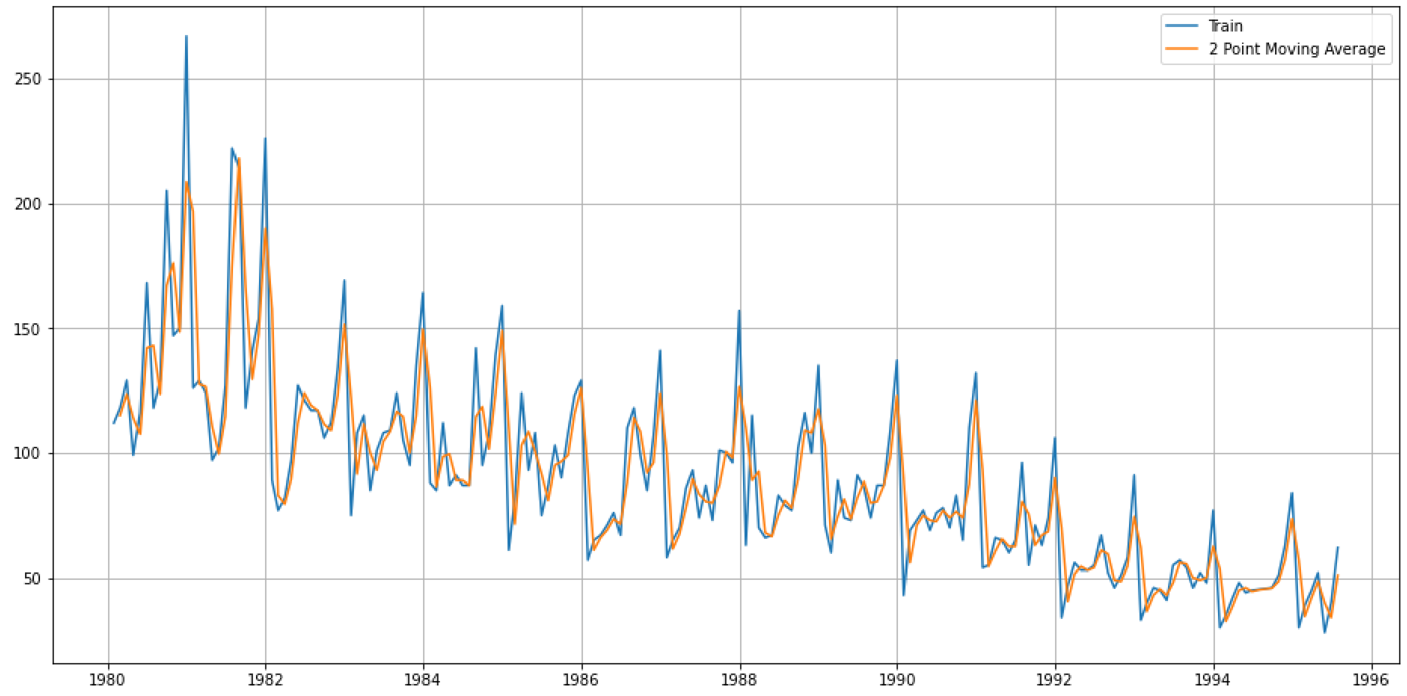


Fig 31: Full Model prediction on best model

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 2 point trailing moving average model which gives an RMSE score of 11.53
- We could predict the future Sales very accurately with this model.

Insights and recommendations:

- The Sales of Rose Wine is maximum in the last few months of the year i.e. October, November, December
- The Sales have dropped drastically with the increasing years
- People in 1980 preferred rose wine but now generally they have many wine options, that is why the Sales might have dropped
- The minimum Sales have been recorded generally in the 4th month i.e, April 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 Rose Wine at the end of year, the price can be slightly increased to maximize the profit.