Business Report

DSBA

Project – Time Series Forecasting

Name: Sandeep Immadi

Date: 07/11/2021

CONTENTS:

Problem:

greatlearning
Learning for Life

List of Figures:

ROSE WINE SALES DATA

41. Residual diagnostics for the new series
42. Optimum model with best parameters
43. Plot of forecast with confidence band
44. Predicted model for the rose wine sales for final model
45. Prediction Graph with upper and lower confidence interval

SPARKLING WINE SALES DATA

46. Visual form of data
47. Plot of yearly sales across years
48. Plot of monthly sales across years
49. Yearly Boxplots
50. Monthly Boxplots
51. Seasonal Plot-monthwise
52. Empirical Cumulative Distribution
53. Average and percent change in Wine sales
54. Decomposed sparkling wine series into components using additive decomposition
55. Decomposed sparkling wine series into components using moving average decomposition
56. Data(split into training and testing purpose)
57. Forecasted sales using Linear Regression model
58. Forecasted sales using Naïve Bayes Approach
59. Forecasted sales using Simple Average Approach
60. Forecasted sales using Trailing Moving Average
61. Comparison of all time series plots
62. Performance of all the models on test data using RMSE
63. Forecasted sales using Simple Exponential Smoothing with alpha=0.995
64. Forecasted sales using Simple Exponential Smoothing with alpha=1 and 0.3
65. Forecasted sales using Double Exponential Smoothing with alpha=0.3 and beta=0.3
66. Forecasted sales using Triple Exponential Smoothing with alpha=0.676, beta=0.088 and gamma=0.323
67. Exponential Smoothing predictions on the actual values
68. Exponential Smoothing on the whole data
69. Plot of Actual vs Forecasted sales using HW's method for 1991-1995 years
70. Visual form of non-stationary series
71. Visual form of stationary series
72. Summary statistics of ARIMA model
73. Residual Diagnostics of the (2,1,2) ARIMA model
74. Summary statistics of SARIMA model
75. Residual Diagnostics of the (2,1,2) *(2,0,2,1,2) SARIMA model
76. Differenced Data Autocorrelation for ARIMA model
77. Differenced Data Partial Autocorrelation for ARIMA model
78. Summary statistics for manual ARIMA model
79. Differenced Data Autocorrelation for SARIMA model
80. Differenced Data Partial Autocorrelation for SARIMA model

List of Tables

ROSE WINE SALES DATA

SPARKLING WINE SALES DATA

greatlearning
Learning for Life

Problem:

For this particular assignment, the data of different types of wine sales in the 20th century is to be analysed. Both of these data are from the same company but of different wines. As an analyst in the ABC Estate Wines, you are tasked to analyse and forecast Wine Sales in the 20th century.

Here we are analysing and forecasting the sale of Sparkling wine in the 20th century. Sparkling wine is a product of ABC Estate Wines.

1.1 Read the data as an appropriate Time Series data and plot the data.

Data loading and overview:

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 187 entries, 0 to 186
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   YearMonth  187 non-null    object
 1   Sparkling  187 non-null    int64 
dtypes: int64(1), object(1)
memory usage: 3.0+ KB
```

Fig.1

```
YearMonth
1980-01-01    1686
1980-02-01    1591
1980-03-01    2304
1980-04-01    1712
1980-05-01    1471
Name: Sparkling, dtype: int64
```

```
YearMonth
1995-03-01    1897
1995-04-01    1862
1995-05-01    1670
1995-06-01    1688
1995-07-01    2031
Name: Sparkling, dtype: int64
```

Fig.2

```
DatetimeIndex(['1980-01-31', '1980-02-29', '1980-03-31', '1980-04-30',
               '1980-05-31', '1980-06-30', '1980-07-31', '1980-08-31',
               '1980-09-30', '1980-10-31',
               ...
               '1994-10-31', '1994-11-30', '1994-12-31', '1995-01-31',
               '1995-02-28', '1995-03-31', '1995-04-30', '1995-05-31',
               '1995-06-30', '1995-07-31'],
              dtype='datetime64[ns]', length=187, freq='M')
```

Fig.3

	YearMonth	Sparkling	Time_Stamp
0	1980-01	1686	1980-01-31
1	1980-02	1591	1980-02-29
2	1980-03	2304	1980-03-31
3	1980-04	1712	1980-04-30
4	1980-05	1471	1980-05-31

Fig.4

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

Fig.5

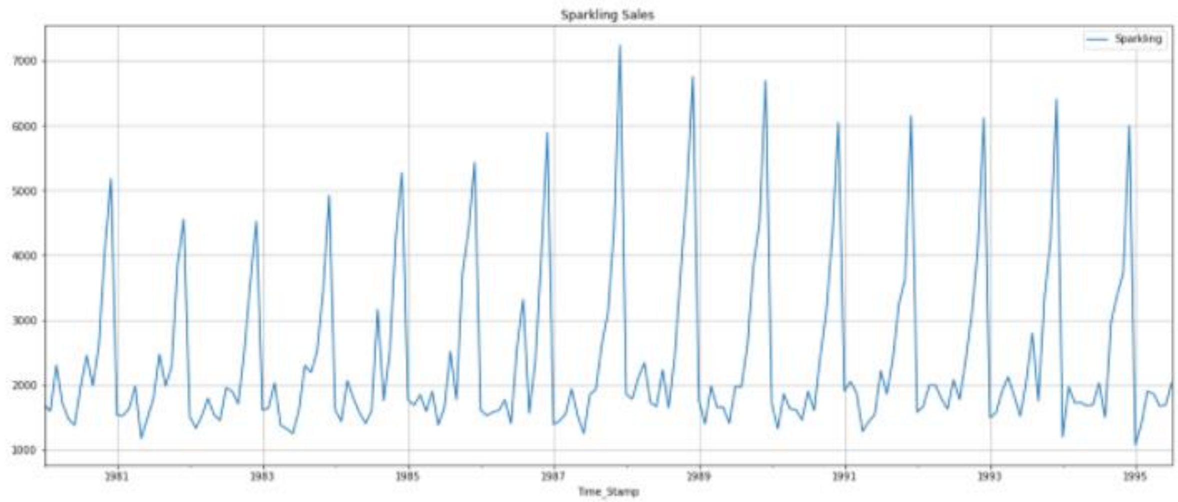


Fig.6

Inferences:

- Here we have collected monthly sales data of Sparkling wine, starting from January,1980 to July,1995 which is of 15 years. We have total 187 records.
- In this data set, we have two columns (YearMonth and Sparkling) and 187 rows.
- Here 'YearMonth' column is of object data type which is indicating the time of sale and 'Sparkling' column is of float data type which gives us the value of Sparkling wine sale.
- As this is a Time series data, so 'YearMonth' column should be in Timestamp format not in object type.

- Therefore, we have added a timestamp column(‘Time_Stamp’) according to our ’YearMonth’ column value in our data set.
- As it is recommended that for time series analysis, we should put Timeseries reference column as Index because it makes easy for slicing and dicing the data for future analysis.
- Therefore, we make our new Time stamp column ‘Time_Stamp’ as index and drop ‘YearMonth’ column because its value is same with new column ‘Time_Stamp ’In this data set there are no null value
- We can notice that there is not much of trend in the plot of 'Sparkling' wine sales down by the year. The Seasonality seems to have pattern on yearly basis.

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

Checking null values for 'Sparkling' wine sales:

```
Sparkling    0
dtype: int64
```

Fig.7

Descriptive Analysis of 'Sparkling' Wine Sales:

	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

Fig.8

- However, for this measure of descriptive statistics we have averaged over the whole data without taking the time component into account hence should look at the box plots year wise and month wise

Plotting Boxplot for year wise:

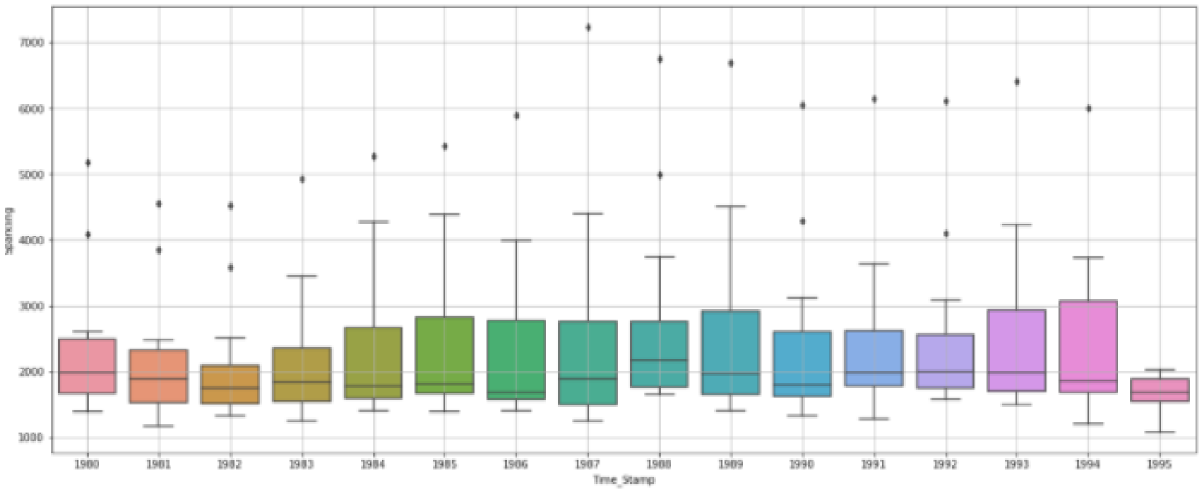


Fig.9

Plotting Boxplot for Month wise:

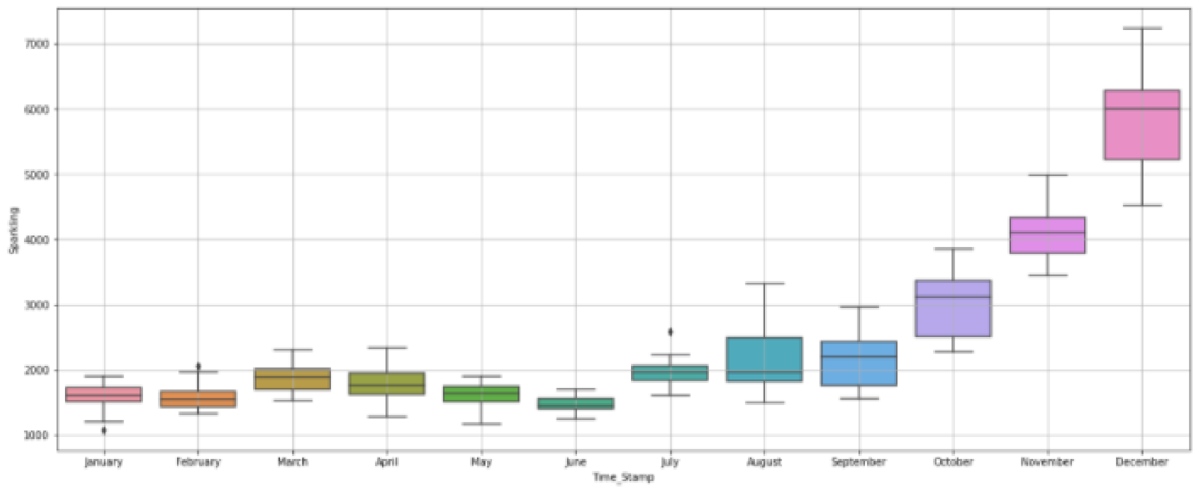


Fig.10

Plotting the Sparkling wine year month wise sales - Line plot:

Time_Stamp	April	August	December	February	January	July	June	March	May	November	October	September
Time_Stamp												
1980	1712.0	2453.0	5179.0	1591.0	1686.0	1966.0	1377.0	2304.0	1471.0	4087.0	2596.0	1984.0
1981	1976.0	2472.0	4551.0	1523.0	1530.0	1781.0	1480.0	1633.0	1170.0	3857.0	2273.0	1981.0
1982	1790.0	1897.0	4524.0	1329.0	1510.0	1954.0	1449.0	1518.0	1537.0	3593.0	2514.0	1706.0
1983	1375.0	2298.0	4923.0	1638.0	1609.0	1600.0	1245.0	2030.0	1320.0	3440.0	2511.0	2191.0
1984	1789.0	3159.0	5274.0	1435.0	1609.0	1597.0	1404.0	2061.0	1567.0	4273.0	2504.0	1759.0
1985	1589.0	2512.0	5434.0	1682.0	1771.0	1645.0	1379.0	1846.0	1896.0	4388.0	3727.0	1771.0
1986	1605.0	3318.0	5891.0	1523.0	1606.0	2584.0	1403.0	1577.0	1765.0	3987.0	2349.0	1562.0
1987	1935.0	1930.0	7242.0	1442.0	1389.0	1847.0	1250.0	1548.0	1518.0	4405.0	3114.0	2638.0
1988	2336.0	1645.0	6757.0	1779.0	1853.0	2230.0	1661.0	2108.0	1728.0	4988.0	3740.0	2421.0
1989	1650.0	1968.0	6694.0	1394.0	1757.0	1971.0	1406.0	1982.0	1654.0	4514.0	3845.0	2608.0
1990	1628.0	1605.0	6047.0	1321.0	1720.0	1899.0	1457.0	1859.0	1615.0	4286.0	3116.0	2424.0
1991	1279.0	1857.0	6153.0	2049.0	1902.0	2214.0	1540.0	1874.0	1432.0	3627.0	3252.0	2408.0
1992	1997.0	1773.0	6119.0	1667.0	1577.0	2076.0	1625.0	1993.0	1783.0	4096.0	3088.0	2377.0
1993	2121.0	2795.0	6410.0	1564.0	1494.0	2048.0	1515.0	1898.0	1831.0	4227.0	3339.0	1749.0
1994	1725.0	1495.0	5999.0	1968.0	1197.0	2031.0	1693.0	1720.0	1674.0	3729.0	3385.0	2968.0
1995	1862.0	NaN	NaN	1402.0	1070.0	2031.0	1688.0	1897.0	1670.0	NaN	NaN	NaN

Fig.11

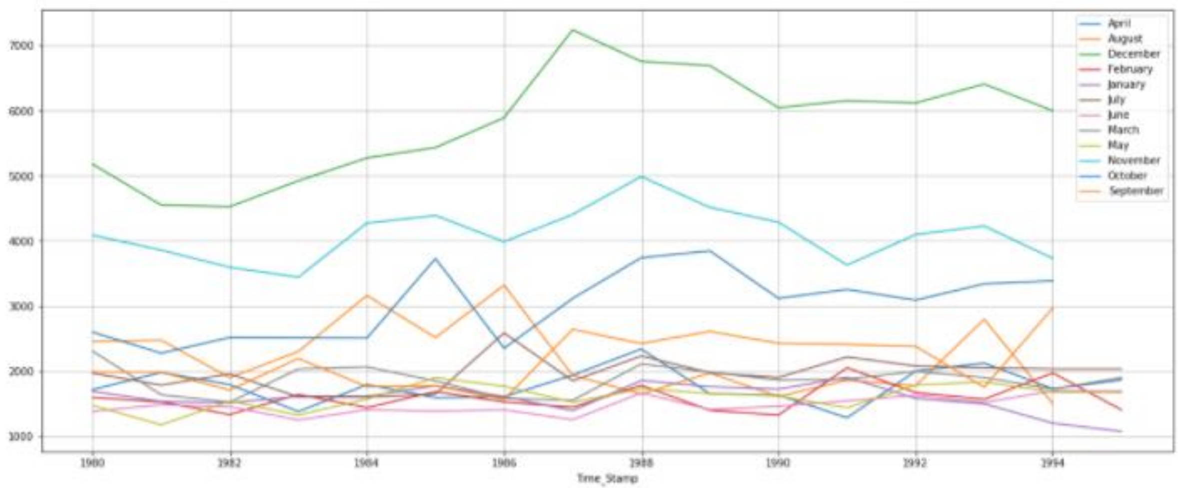


Fig.12

Plotting a month plot to check the sales in different years and within different month across:

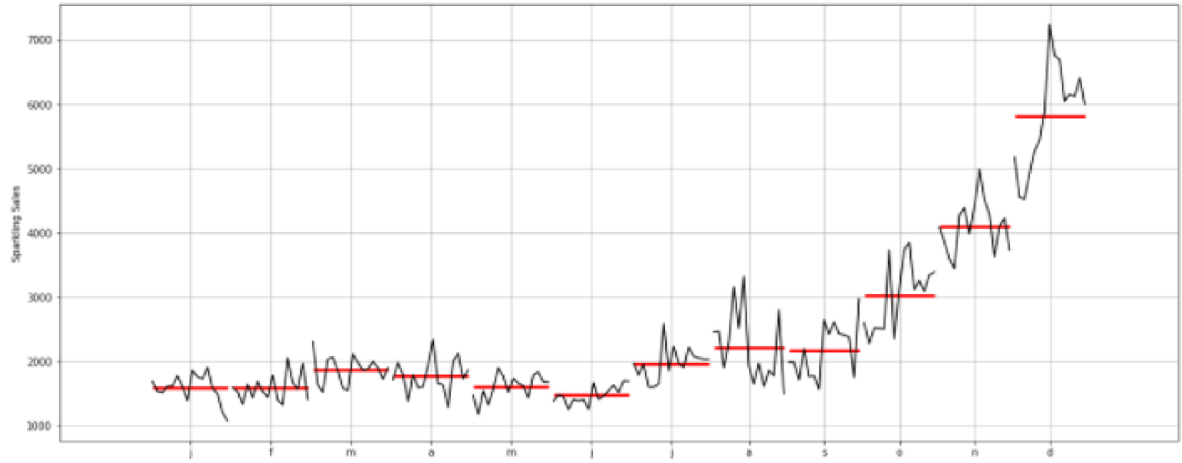


Fig.13

Plotting the Empirical Cumulative Distribution:

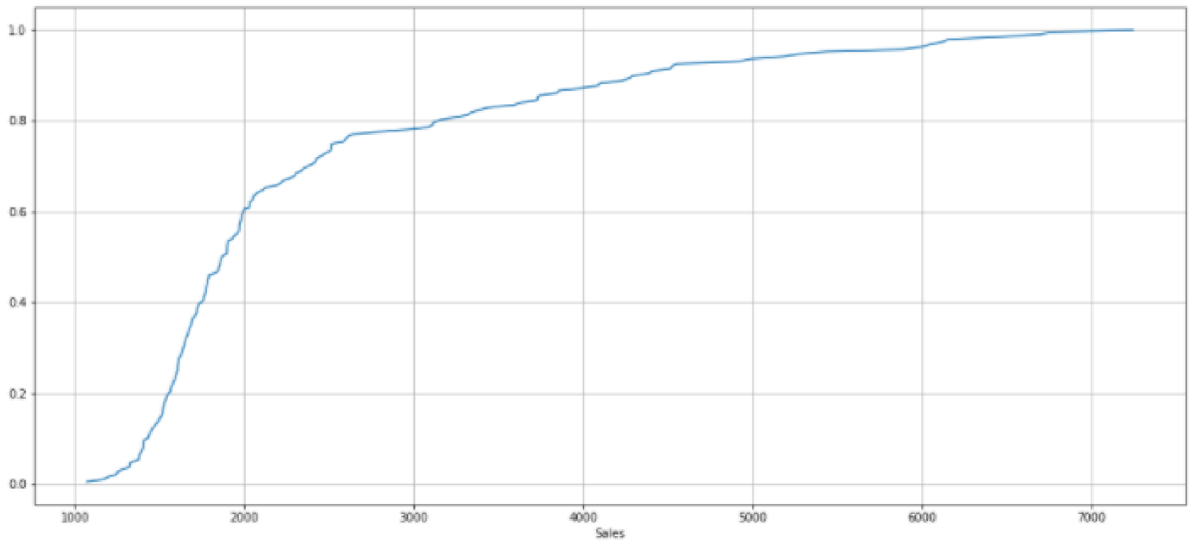


Fig.14

Plot the average Sparkling Wine Sales per month and the month-on-month percentage change of Sparkling Wine Sales:

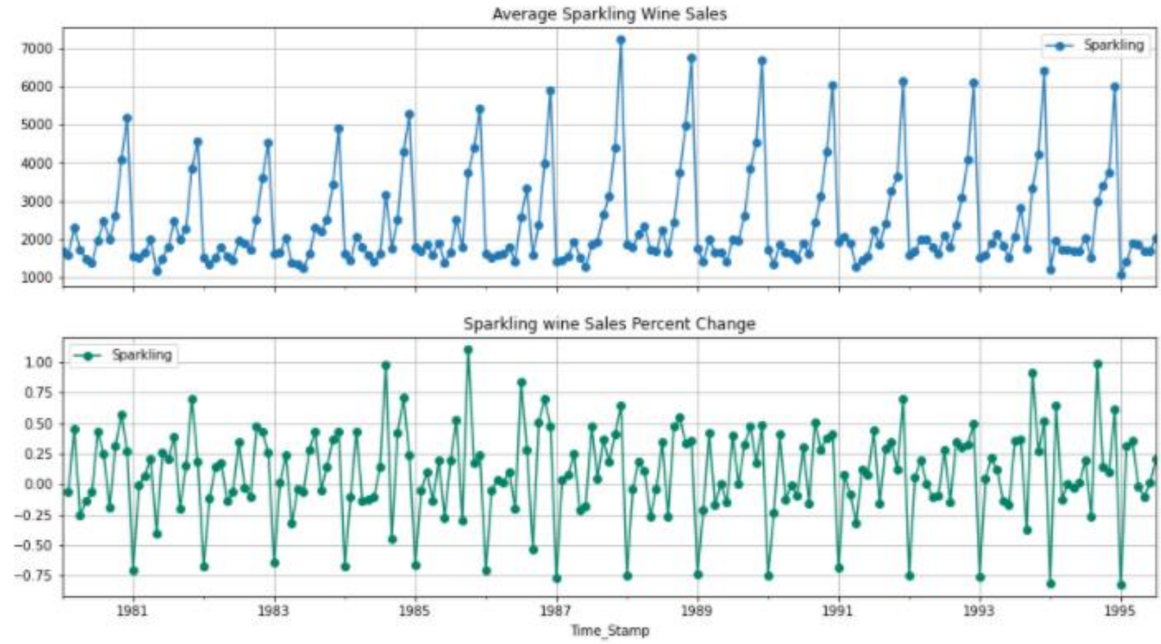


Fig.15

Decomposing the time series into Additive decomposition and plot:

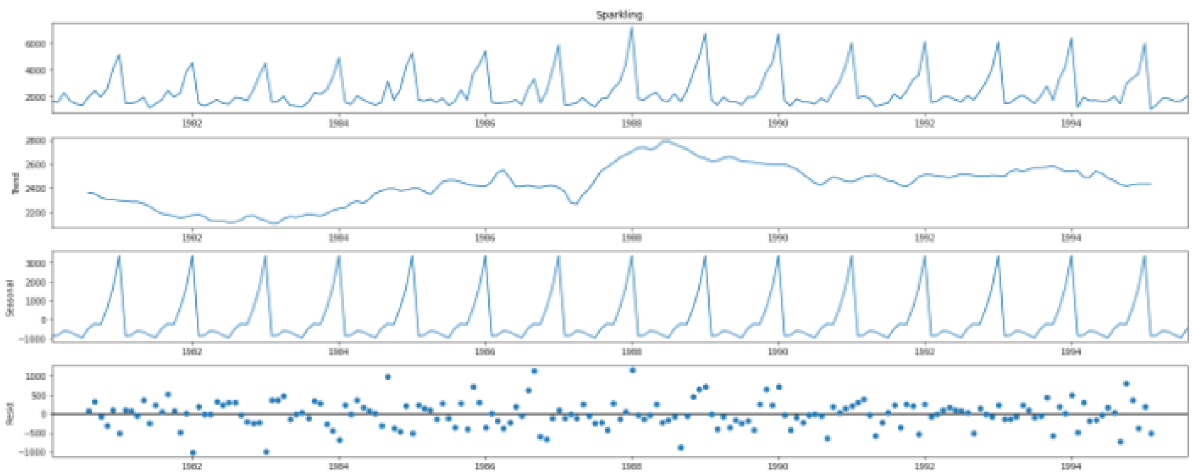


Fig.16

```
Trend
 Time_Stamp
1980-01-31            NaN
1980-02-29            NaN
1980-03-31            NaN
1980-04-30            NaN
1980-05-31            NaN
1980-06-30            NaN
1980-07-31    2360.666667
1980-08-31    2351.333333
1980-09-30    2320.541667
1980-10-31    2303.583333
1980-11-30    2302.041667
1980-12-31    2293.791667
Name: trend, dtype: float64

Seasonality
 Time_Stamp
1980-01-31    -854.260599
1980-02-29    -830.350678
1980-03-31    -592.356630
1980-04-30    -658.490559
1980-05-31    -824.416154
1980-06-30    -967.434011
1980-07-31    -465.502265
1980-08-31    -214.332821
1980-09-30    -254.677265
1980-10-31     599.769957
1980-11-30    1675.067179
1980-12-31    3386.983846
Name: seasonal, dtype: float64

Residual
 Time_Stamp
1980-01-31           NaN
1980-02-29           NaN
1980-03-31           NaN
1980-04-30           NaN
1980-05-31           NaN
1980-06-30           NaN
1980-07-31     70.835599
1980-08-31    315.999487
1980-09-30    -81.864401
1980-10-31   -307.353290
1980-11-30    109.891154
1980-12-31   -501.775513
Name: resid, dtype: float64
```

Fig.17

Decomposing the time series into multiplicative decomposition and plot:

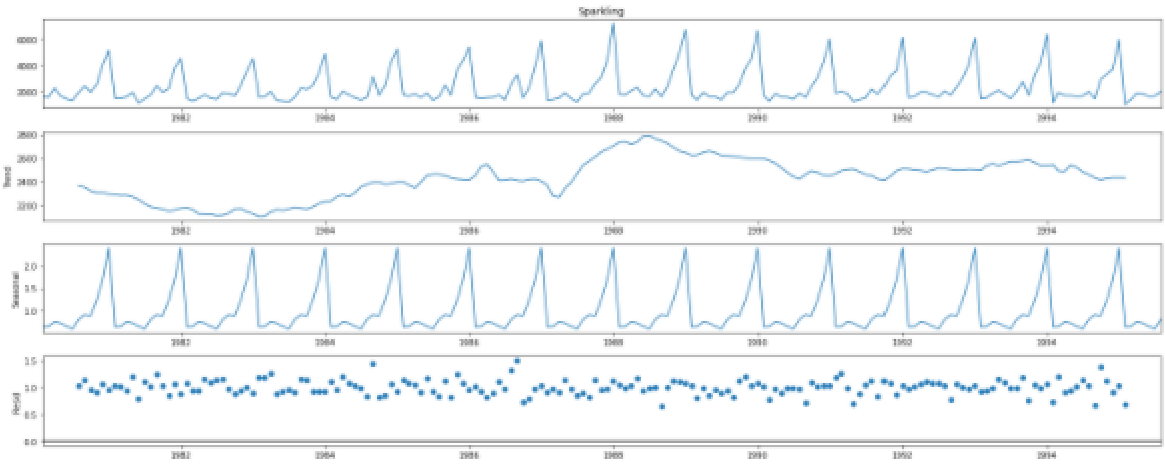


Fig.18

```
Trend
 Time_Stamp
1980-01-31            NaN
1980-02-29            NaN
1980-03-31            NaN
1980-04-30            NaN
1980-05-31            NaN
1980-06-30            NaN
1980-07-31    2360.666667
1980-08-31    2351.333333
1980-09-30    2320.541667
1980-10-31    2303.583333
1980-11-30    2302.041667
1980-12-31    2293.791667
Name: trend, dtype: float64

Seasonality
 Time_Stamp
1980-01-31    0.649843
1980-02-29    0.659214
1980-03-31    0.757440
1980-04-30    0.730351
1980-05-31    0.660609
1980-06-30    0.603468
1980-07-31    0.809164
1980-08-31    0.918822
1980-09-30    0.894367
1980-10-31    1.241789
1980-11-30    1.690158
1980-12-31    2.384776
Name: seasonal, dtype: float64

Residual
 Time_Stamp
1980-01-31         NaN
1980-02-29         NaN
1980-03-31         NaN
1980-04-30         NaN
1980-05-31         NaN
1980-06-30         NaN
1980-07-31    1.029230
1980-08-31    1.135407
1980-09-30    0.955954
1980-10-31    0.907513
1980-11-30    1.050423
1980-12-31    0.946770
Name: resid, dtype: float64
```

Fig.19

Inferences:

- There are no null values in the dataset.
- We have 187 data points in 'Sparkling' wine data.
- As we can observe Time series plot, the boxplots over also does not indicate any trend
- Also, we can observe that there is some outlier's presence in almost all the year except for 1995
- We also observe that 'December' month has the highest sales of 'Sparkling' wine.
- We can observe from line plot of Year/Month wise sales data of 'Sparkling' wine December has highest sales followed by November.
- The median values are stable from January to June and has an increasing trend from July to December. January to December months. The Average Sales value does not show a trend.
- Additive decomposition we see the residuals 0 and 1.
- Multiplicative decomposition we can see that residuals are around 1.
- The time series for the Sparking is Additive.

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

- Splitted the data into train and test data

Displaying multiple data frames from one cell:

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

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

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

Fig.20

Shape of the train and test data:

```
(132, 1)
(55, 1)
```

Fig.21

plotting the graph for train and test set:

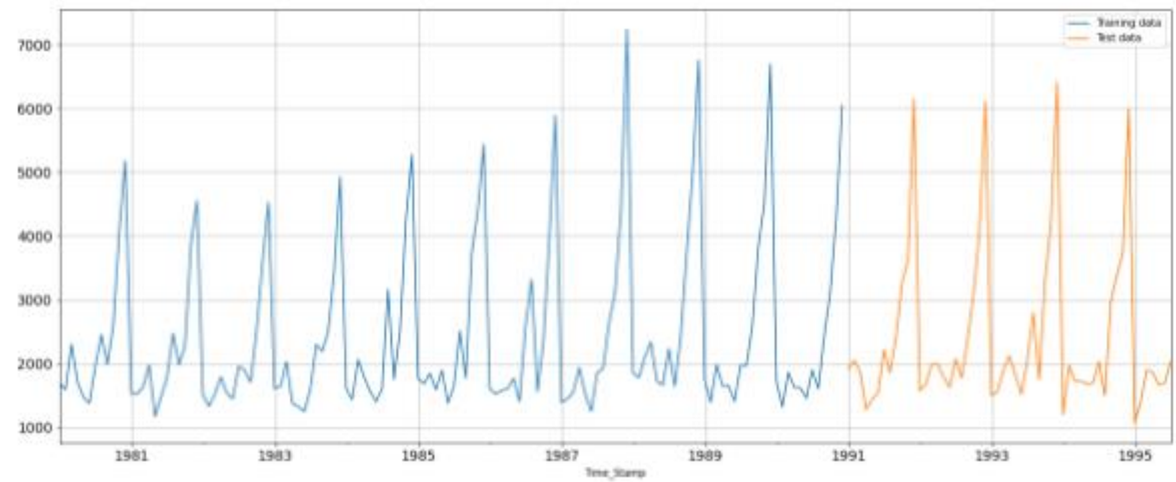


Fig.22

Inferences:

- The Train data of Sparkling wine sales has been split for data up to 1990 and has 132 data points.
- The Test data of Sparkling wine sales has been split for data from 1991 and has 55 data points.
- From our train-test split we are predicting the future sales as compared to the past years.

1.4 **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 and simple average models. should also be built on the training data and check the performance on the test data using RMSE.**

Model 1: Linear Regression

For this particular linear regression, we are going to regress the 'Sparkling' 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, 2
1, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 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, 11
2, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 12
7, 128, 129, 130, 131, 132]


Test Time instance
 [133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 14
7, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 16
2, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 17
7, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187]
```

Fig.23

```
First few rows of Training Data
            Sparkling  time
Time_Stamp
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
            Sparkling  time
Time_Stamp
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
            Sparkling  time
Time_Stamp
1991-01-31       1902   133
1991-02-28       2049   134
1991-03-31       1874   135
1991-04-30       1279   136
1991-05-31       1432   137

Last few rows of Test Data
            Sparkling  time
Time_Stamp
1995-03-31       1897   183
1995-04-30       1862   184
1995-05-31       1670   185
1995-06-30       1688   186
1995-07-31       2031   187
```

Fig.24

Now that our training and test data has been modified, let us go ahead use "Linear Regression" to build the model on the training data and test the model on the test data

Plotting Linear Regression Forecast:

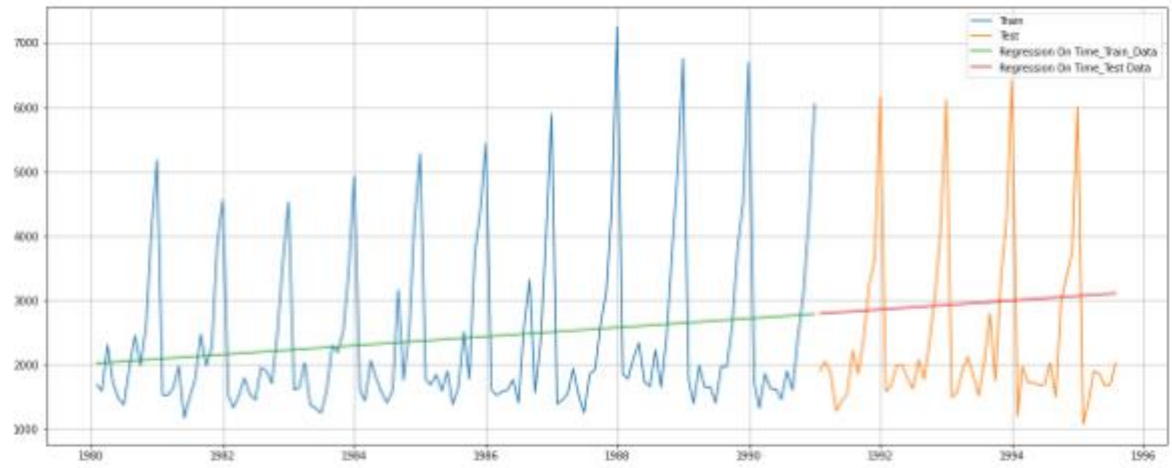


Fig.25

<u>Model 1 Evaluation:</u>

Train - RMSE score: For RegressionOnTime forecast on the Train Data, RMSE is 1279.322

Test - RMSE score: For RegressionOnTime forecast on the Test Data, RMSE is 1389.135

Creating Data frame:

	Train_RMSE	Test_RMSE
RegressionOnTime	1279.322346	1389.135175

Fig.26

Model 2: Naïve

```
Time_Stamp
1980-01-31    6047
1980-02-29    6047
1980-03-31    6047
1980-04-30    6047
1980-05-31    6047
Name: naive, dtype: int64
```

```
Time_Stamp
1991-01-31    6047
1991-02-28    6047
1991-03-31    6047
1991-04-30    6047
1991-05-31    6047
Name: naive, dtype: int64
```

Fig.27

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.

Plotting Naive Forecast:

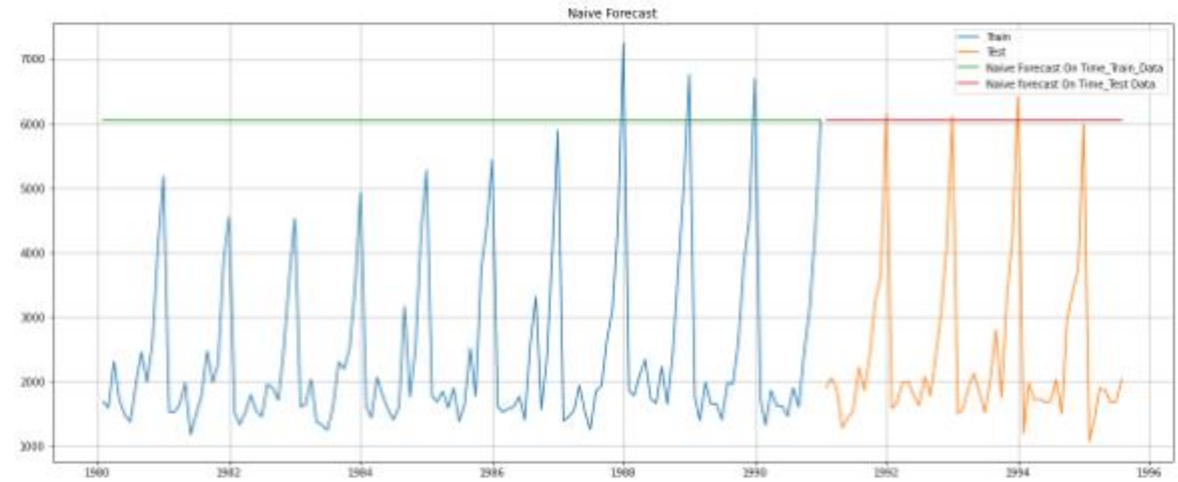


Fig.28

<u>Model 2 Evaluation:</u>

Train - RMSE score: For Naive Model forecast on the Train Data, RMSE is 3867.701

Test - RMSE score: For Naive Model forecast on the Test Data, RMSE is 3864.279

Creating Data frame:

	Train_RMSE	Test_RMSE
RegressionOnTime	1279.322346	1389.135175
Naive Model	3867.700802	3864.279352

Fig.29

Model 3: Simple Average

Time_Stamp	Sparkling	mean_forecast
1980-01-31	1686	2403.780303
1980-02-29	1591	2403.780303
1980-03-31	2304	2403.780303
1980-04-30	1712	2403.780303
1980-05-31	1471	2403.780303

Time_Stamp	Sparkling	mean_forecast
1991-01-31	1902	2403.780303
1991-02-28	2049	2403.780303
1991-03-31	1874	2403.780303
1991-04-30	1279	2403.780303
1991-05-31	1432	2403.780303

Fig.30

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

Plotting Simple Average Forecast:

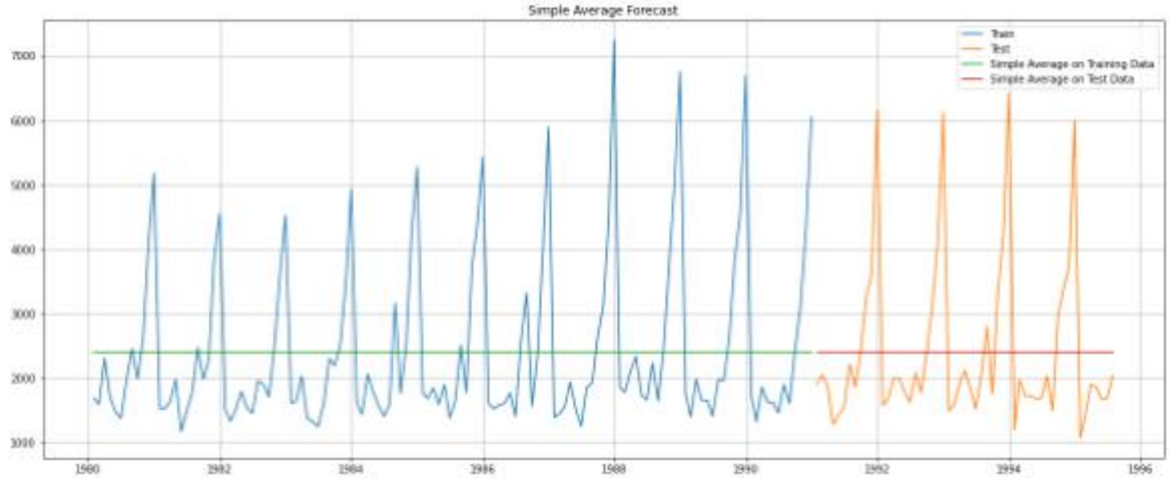


Fig.31

Model 3 Evaluation:

Train - RMSE score: For Simple Average Model forecast on the Train Data, RMSE is 1298.484

Test - RMSE score: For Simple Average Model forecast on the Test Data, RMSE is 1275.082

Creating Data frame:

	Train_RMSE	Test_RMSE
RegressionOnTime	1279.322346	1389.135175
Naive Model	3867.700802	3864.279352
Simple Average	1298.483628	1275.081804

Fig.32

Model 4: Moving Average

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

Fig.33

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

Fig.34

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.

Plotting Moving Average Forecast:

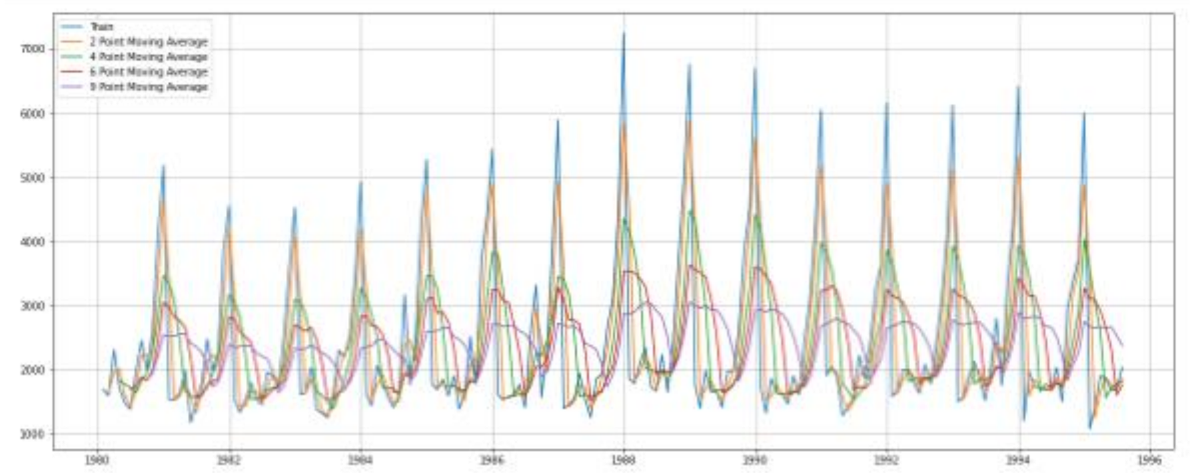


Fig.35

Let us 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.

Plotting Graph for Trailing MA:

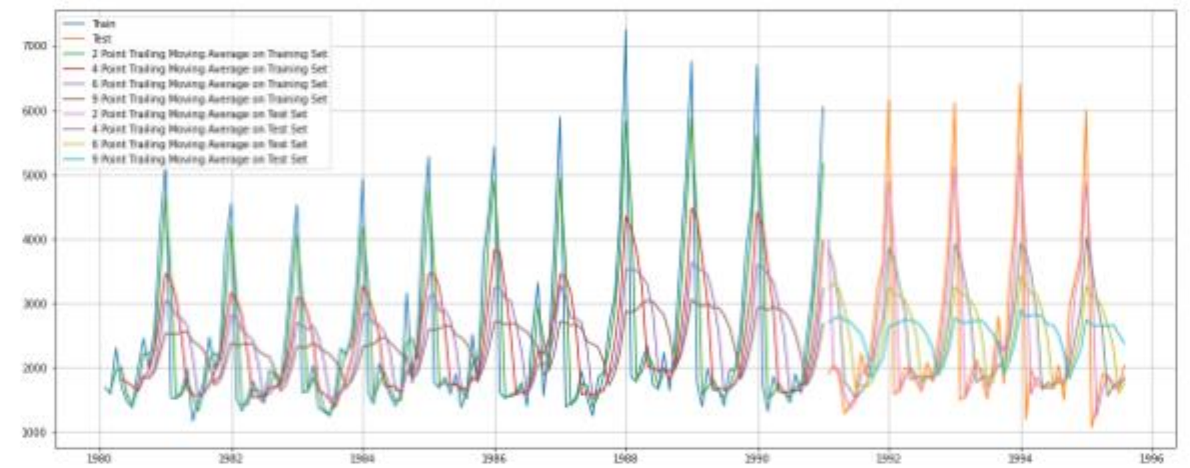


Fig.36

Model 4 Evaluation:

For MA forecast on the Train Data, RMSE is 3867.701

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

Creating Data frame:

	Train_RMSE	Test_RMSE
RegressionOnTime	1279.322346	1389.135175
Naive Model	3867.700802	3864.279352
Simple Average	1298.483628	1275.081804
2pointTrailingMovingAverage	NaN	813.400684
4pointTrailingMovingAverage	NaN	1156.589694
6pointTrailingMovingAverage	NaN	1283.927428
9pointTrailingMovingAverage	NaN	1346.278315

Fig.37

Model 5: Simple Exponential Smoothing

```
{'smoothing_level': 0.995,
 'smoothing_trend': nan,
 'smoothing_seasonal': nan,
 'damping_trend': nan,
 'initial_level': 1686.0,
 'initial_trend': nan,
 'initial_seasons': array([], dtype=float64),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

Fig.38

Predicting for training data:

Time_Stamp	Sparkling	predict
1980-01-31	1686	1686.000000
1980-02-29	1591	1686.000000
1980-03-31	2304	1591.475000
1980-04-30	1712	2300.437375
1980-05-31	1471	1714.942187

Fig.39

Predicting for Test data:

Time_Stamp	Sparkling	predict
1991-01-31	1902	6038.165663
1991-02-28	2049	6038.165663
1991-03-31	1874	6038.165663
1991-04-30	1279	6038.165663
1991-05-31	1432	6038.165663

Fig.40

Plotting Simple Exponential Smoothing forecast:

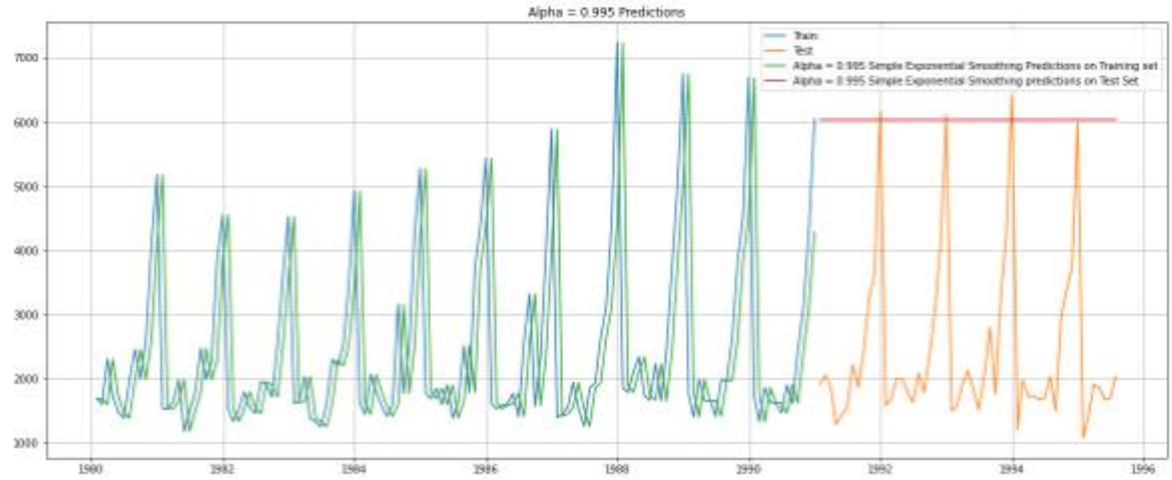


Fig.41

Train - RMSE score: For Alpha =0.995 Simple Exponential Smoothing Model forecast on the Train Data, RMSE is 1372.055

Test - RMSE score: For Alpha =0.995 Simple Exponential Smoothing Model forecast on the Test Data, RMSE is 3855.941

Creating DataFrame:

	Train_RMSE	Test_RMSE
RegressionOnTime	1279.322346	1389.135175
Naive Model	3867.700802	3864.279352
Simple Average	1298.483628	1275.081804
2pointTrailingMovingAverage	NaN	813.400684
4pointTrailingMovingAverage	NaN	1156.589694
6pointTrailingMovingAverage	NaN	1283.927428
9pointTrailingMovingAverage	NaN	1346.278315
Alpha=0.995,SimpleExponentialSmoothing	1372.054747	3855.940897

Fig.42

Setting Different Alpha Values: We will run a loop with different alpha values to understand which particular value works best for alpha on the test set and Train Set.

Alpha_Values	Train_RMSE	Test_RMSE

Model Evaluation:

	Alpha_Values	Train_RMSE	Test_RMSE
0	0.3	1359.511747	1935.507132
1	0.4	1352.588879	2311.919615
2	0.5	1344.004369	2666.351413
3	0.6	1338.805381	2979.204388
4	0.7	1338.844308	3249.944092
5	0.8	1344.462091	3483.801006
6	0.9	1355.723518	3686.794285

Fig.43

Plotting graph for different Alpha values of Training and Test data:

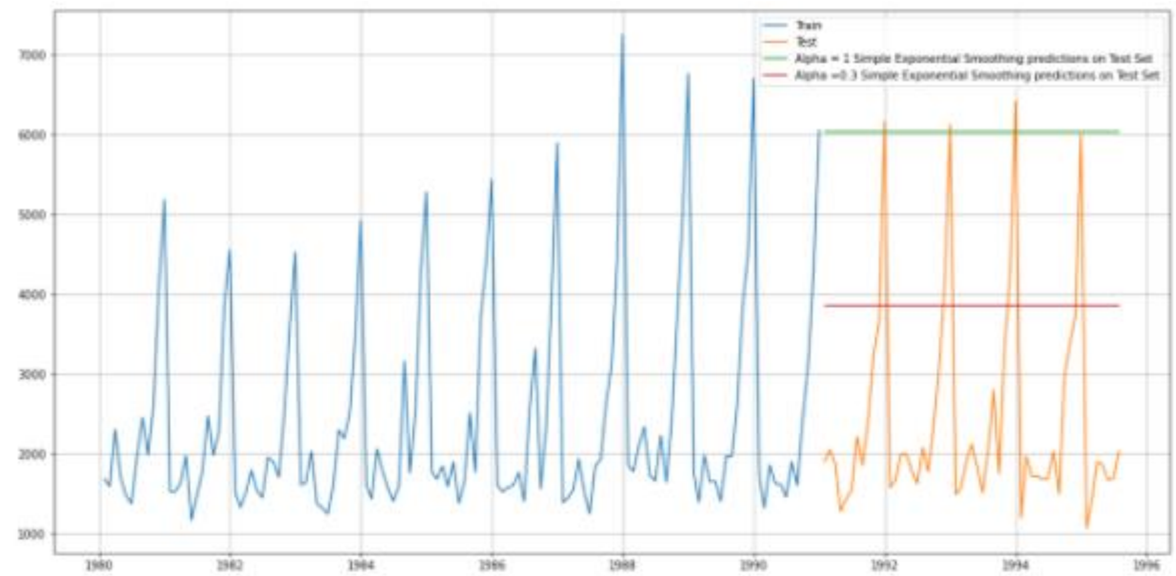


Fig.44

Creating Data frame:

	Train_RMSE	Test_RMSE
RegressionOnTime	1279.322346	1389.135175
Naive Model	3867.700802	3864.279352
Simple Average	1298.483628	1275.081804
2pointTrailingMovingAverage	NaN	813.400684
4pointTrailingMovingAverage	NaN	1156.589694
6pointTrailingMovingAverage	NaN	1283.927428
9pointTrailingMovingAverage	NaN	1346.278315
Alpha=0.995,SimpleExponentialSmoothing	1372.054747	3855.940897
Alpha=0.3,SimpleExponentialSmoothing	1338.805381	1935.507132

Fig.45

Model 6: Double Exponential Smoothing

Two parameters α and β are estimated in this model. Level and Trend are accounted for in this model

First, we will define an empty data frame to store our values from the loop:

Alpha_Values	Beta_Values	Train_RMSE	Test_RMSE

	Alpha_Values	Beta_Values	Train_RMSE	Test_RMSE
0	0.3	0.3	1592.292788	18259.110704
8	0.4	0.3	1569.338606	23878.496940
1	0.3	0.4	1682.573828	26069.841401
16	0.5	0.3	1530.575845	27095.532414
24	0.6	0.3	1506.449870	29070.722592

Fig.46

Plotting Graph on Both Training and Test data:

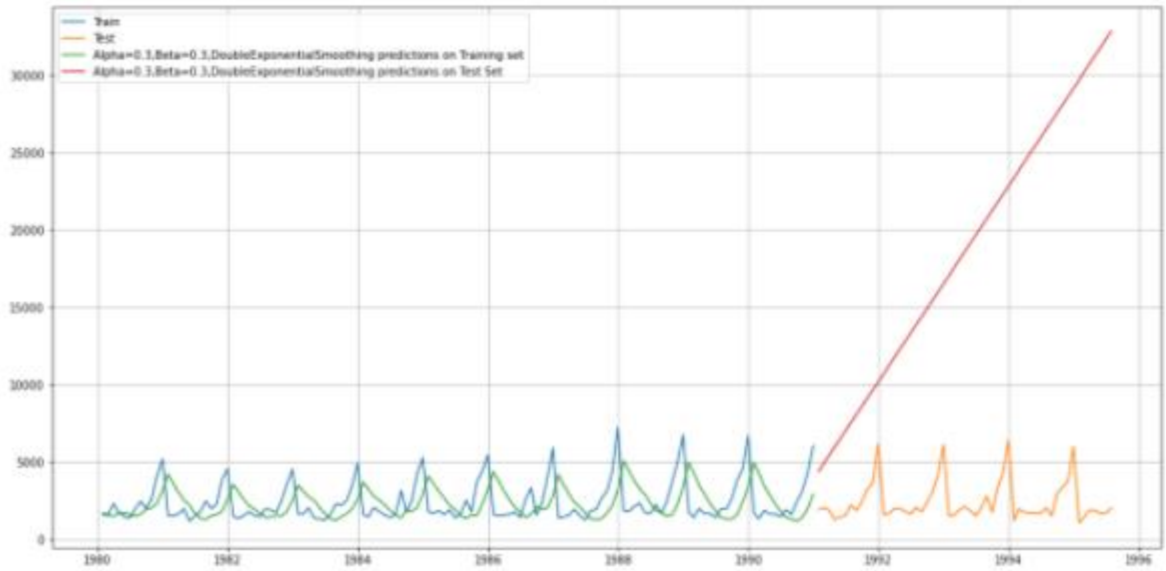


Fig.47

Creating DataFrame:

	Train_RMSE	Test_RMSE
RegressionOnTime	1279.322346	1389.135175
Naive Model	3867.700802	3864.279352
Simple Average	1298.483628	1275.081804
2pointTrailingMovingAverage	NaN	813.400684
4pointTrailingMovingAverage	NaN	1156.589694
6pointTrailingMovingAverage	NaN	1283.927428
9pointTrailingMovingAverage	NaN	1346.278315
Alpha=0.995,SimpleExponentialSmoothing	1372.054747	3855.940897
Alpha=0.3,SimpleExponentialSmoothing	1338.805381	1935.507132
Alpha=0.3,Beta=0.3,DoubleExponentialSmoothing	1500.689062	18259.110704

Fig.48

Model 7: Triple Exponential Smoothing

Two parameters α and β are estimated in this model. Level and Trend are accounted for in this model

```
{'smoothing_level': 0.11057044018305404,
 'smoothing_trend': 0.06076609768412894,
 'smoothing_seasonal': 0.39187601902826213,
 'damping_trend': nan,
 'initial_level': 1621.806699459997,
 'initial_trend': -8.175193580026457,
 'initial_seasons': array([1.07304448, 1.02730389, 1.39469706, 1.20333193,
0.98008967,
        0.97664181, 1.39261648, 1.70888984, 1.37519684, 1.81953245,
        2.82880203, 3.59142504]),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

Fig.49

Prediction on Train Set:

Time_Stamp	Sparkling	auto_predict
1980-01-31	1686	1731.498384
1980-02-29	1591	1644.182518
1980-03-31	2304	2211.921279
1980-04-30	1712	1907.144721
1980-05-31	1471	1526.491107

Fig.50

Prediction on Test set:

Time_Stamp	Sparkling	auto_predict
1991-01-31	1902	1578.528263
1991-02-28	2049	1336.087202
1991-03-31	1874	1747.686817
1991-04-30	1279	1632.972086
1991-05-31	1432	1525.031468

Fig.51

Plotting graph using Training set and Test set using autofit:

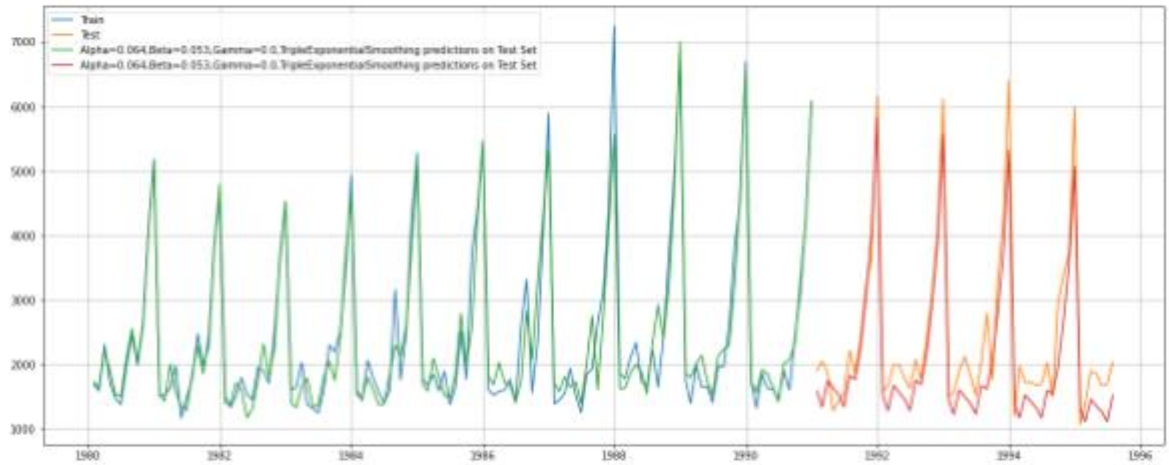


Fig.52

Train - RMSE score: For Alpha=0.064, Beta=0.053, Gamma=0.0, Triple Exponential Smoothing Model forecast on the Train Data, RMSE is 356.782

Test - RMSE score: For Alpha=0.064, Beta=0.053, Gamma=0.0, Triple Exponential Smoothing Model forecast on the Test Data, RMSE is 463.502

Creating DataFrame:

	Train_RMSE	Test_RMSE
RegressionOnTime	1279.322346	1389.135175
Naive Model	3867.700802	3864.279352
Simple Average	1298.483628	1275.081804
2pointTrailingMovingAverage	NaN	813.400684
4pointTrailingMovingAverage	NaN	1156.589694
6pointTrailingMovingAverage	NaN	1283.927428
9pointTrailingMovingAverage	NaN	1346.278315
Alpha=0.995,SimpleExponentialSmoothing	1372.054747	3855.940897
Alpha=0.3,SimpleExponentialSmoothing	1338.805381	1935.507132
Alpha=0.3,Beta=0.3,DoubleExponentialSmoothing	1500.689062	18259.110704
Alpha=0.064,Beta=0.053,Gamma=0.0,TripleExponentialSmoothing	356.782453	463.501976

Fig.53

Defining an empty dataframe to store our values from the loop:

	Alpha_Values	Beta_Values	Gamma_Values	Train_RMSE	Test_RMSE
0	0.3	0.3	0.3	404.513320	392.786198
8	0.3	0.4	0.3	424.828055	410.854547
65	0.4	0.3	0.4	435.553595	421.409170
296	0.7	0.8	0.3	700.317756	518.188752
130	0.5	0.3	0.5	498.239915	542.175497

Fig.54

Plotting on both the Training and Test data using brute force alpha, beta:

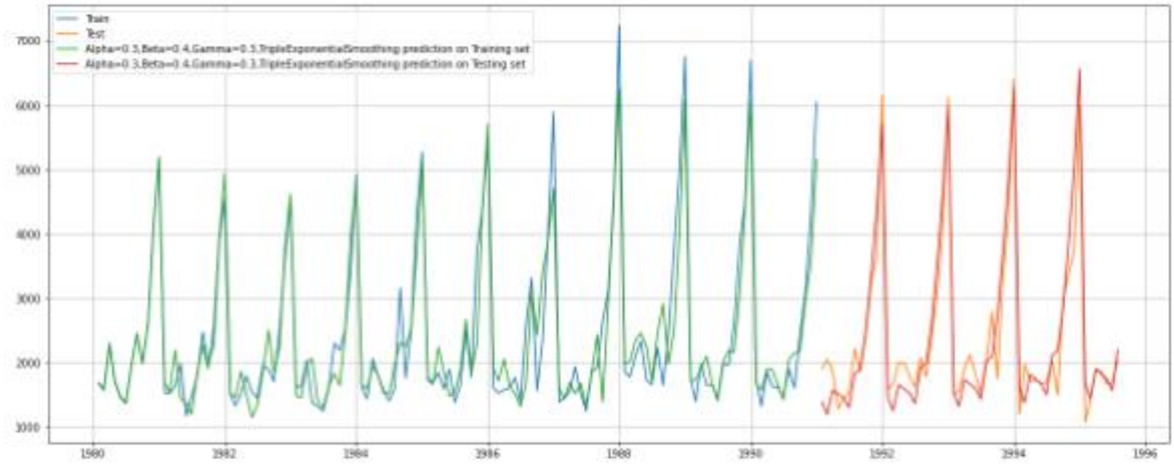


Fig.55

Creating Dataframe:

	Train_RMSE	Test_RMSE
RegressionOnTime	1279.322346	1389.135175
Naive Model	3867.700802	3864.279352
Simple Average	1298.483628	1275.081804
2pointTrailingMovingAverage	NaN	813.400684
4pointTrailingMovingAverage	NaN	1156.589694
6pointTrailingMovingAverage	NaN	1283.927428
9pointTrailingMovingAverage	NaN	1346.278315
Alpha=0.995,SimpleExponentialSmoothing	1372.054747	3855.940897
Alpha=0.3,SimpleExponentialSmoothing	1338.805381	1935.507132
Alpha=0.3,Beta=0.3,DoubleExponentialSmoothing	1500.689062	18259.110704
Alpha=0.064,Beta=0.053,Gamma=0.0,TripleExponentialSmoothing	356.782453	463.501976
Alpha=0.3,Beta=0.4,Gamma=0.3,TripleExponentialSmoothing	404.513320	392.786198

Fig.56

Sorted by RMSE values on Test Data

	Train_RMSE	Test_RMSE
Alpha=0.3,Beta=0.4,Gamma=0.3,TripleExponentialS...	404.513320	392.786198
Alpha=0.064,Beta=0.053,Gamma=0.0,TripleExponent...	356.782453	463.501976
2pointTrailingMovingAverage	NaN	813.400684
4pointTrailingMovingAverage	NaN	1156.589694
Simple Average	1298.483628	1275.081804
6pointTrailingMovingAverage	NaN	1283.927428
9pointTrailingMovingAverage	NaN	1346.278315
RegressionOnTime	1279.322346	1389.135175
Alpha=0.3,SimpleExponentialSmoothing	1338.805381	1935.507132
Alpha=0.995,SimpleExponentialSmoothing	1372.054747	3855.940897
Naive Model	3867.700802	3864.279352
Alpha=0.3,Beta=0.3,DoubleExponentialSmoothing	1500.689062	18259.110704

Sorted by RMSE values on Train Data

	Train_RMSE	Test_RMSE
Alpha=0.064,Beta=0.053,Gamma=0.0,TripleExponent...	356.782453	463.501976
Alpha=0.3,Beta=0.4,Gamma=0.3,TripleExponentialS...	404.513320	392.786198
RegressionOnTime	1279.322346	1389.135175
Simple Average	1298.483628	1275.081804
Alpha=0.3,SimpleExponentialSmoothing	1338.805381	1935.507132
Alpha=0.995,SimpleExponentialSmoothing	1372.054747	3855.940897
Alpha=0.3,Beta=0.3,DoubleExponentialSmoothing	1500.689062	18259.110704
Naive Model	3867.700802	3864.279352
2pointTrailingMovingAverage	NaN	813.400684
4pointTrailingMovingAverage	NaN	1156.589694
6pointTrailingMovingAverage	NaN	1283.927428
9pointTrailingMovingAverage	NaN	1346.278315

Fig.57

Logistic Regression:

Steps:

- For this particular linear regression, we are going to regress the 'Sparkling' variable against the order of the occurrence. For this we need to modify our training data before fitting it into a linear regression.
- We have generated the numerical time instance order for both the training and test set. After that we add these values in the training and test set.
- Initiate LogisticRegression () classifier.
- Fit into train data set
- Predict the target variable for train and test data
- Logistic Regression RMSE score for test data: 1389.135

- From above graph we can see Regression on time graph is a straight line which is not overlapping with the Test data at all. This is not the desired output. This is not the correct model.

Naive 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.
- After performing Naïve model, we get the RMSE value for test data = 3864.279.
- From above graph we can see Naive on time graph is a straight line which is not overlapping with the Test data at all.
- This is not the desired output. This is not the correct model.

Simple Average:

- For this particular simple average method, we will forecast by using the average of the training values.
- After performing Simple average, we get the RMSE value for test data = 1275.082
- From above graph we can see Simple average on time graph is a straight line which is not overlapping with the Test data at all. This is not the desired output. This is not the correct model.

Moving Average:

- This method uses averaging to forecast the values based on window sizes ie. The window keeps on moving with the size constant for newer points to be forecasted.
- For 9 point Moving Average Model forecast on the Training Data, RMSE is 1346.27

Exponential Smoothing:

- Exponential smoothing of time series data assigns exponentially decreasing weights for newest to oldest observations. That implies, that the older data get less priority (“weight”) and the newer data is more relevant and is assigned more weight.
- Exponential smoothing is usually used to make short term forecasts, as longer-term forecasts using this technique cannot be more reliable.
- Simple exponential smoothing uses a weighted moving average with exponentially decreasing weights. This method is suitable for forecasting data with no clear trend or seasonal pattern.
- Holt's linear method with additive errors or double exponential smoothing is applicable when data has Trend but no seasonality

- Triple exponential smoothing (also called the Multiplicative Holt-Winters) is applicable for the data that shows trends and seasonality.
- As our data shows both trend and seasonality, we will go for Holt-Winters exponential method.
- With the help of 'ExponentialSmoothing' function we performed Triple exponential smoothing on our data set.

Steps:

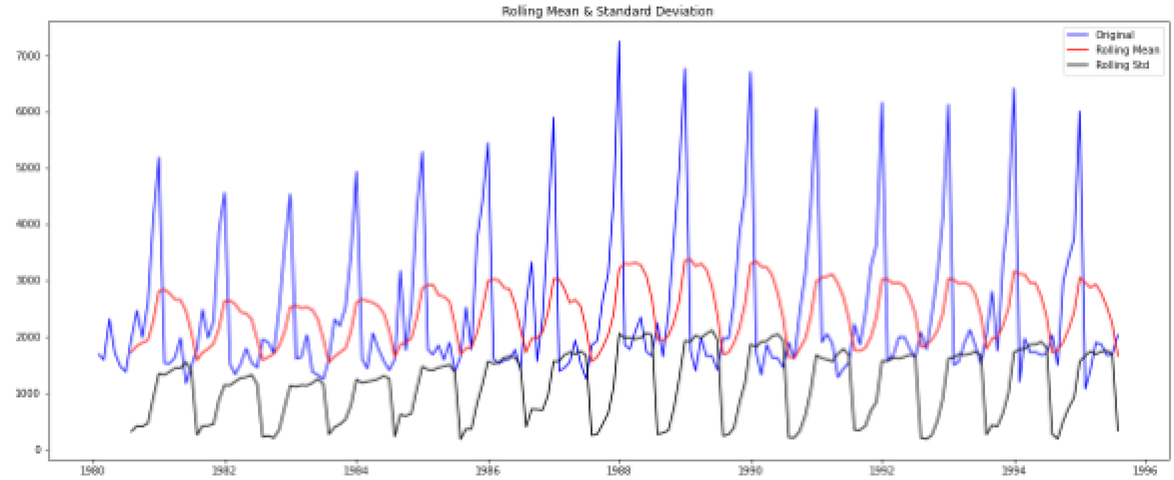
- Initializing the Double Exponential Smoothing Model
- Fitting the model
- Forecasting using this model for the duration of the test set

We got the smoothing parameter value:

- Looking at the RMSE values for all the models, we can conclude that the Triple Exponential Model with alpha, beta, gamma as 0.3,0.4 and 0.3 respectively performs the best (392.786).
- So, we build a complete model on sparkling dataset on these parameters.

1.5 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 series is said to be stationary if its mean and variance are constant over a period of time
- Check for stationarity of the whole Time Series data.
- We have performed Augmented Dickey-Fuller test to check stationarity.
- The Augmented Dickey-Fuller test is 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:
- * H0: The Time Series has a unit root and it is non-stationary.
- * H1: The Time Series does not have a unit root and it is stationary.
- To build ARIMA models we would want the Time series to be stationary and thus we would
- want the p-value of this test to be less than the alpha = 0.05.
- After performing Augmented Dickey-Fuller test, we get p-value = 0.601
- Hence, p-value is higher than alpha, we cannot reject the null hypothesis and can say that at
- 5% significant level the Time Series is non-stationary.




```
Results of Dickey-Fuller Test:
Test Statistic                  -1.360497
p-value                          0.601061
#Lags Used                      11.000000
Number of Observations Used    175.000000
Critical Value (1%)             -3.468280
dtype: float64

Test Statistic                  -1.360497
p-value                          0.601061
#Lags Used                      11.000000
Number of Observations Used    175.000000
Critical Value (1%)             -3.468280
Critical Value (5%)             -2.878202
dtype: float64

Test Statistic                  -1.360497
p-value                          0.601061
#Lags Used                      11.000000
Number of Observations Used    175.000000
Critical Value (1%)             -3.468280
Critical Value (5%)             -2.878202
Critical Value (10%)            -2.575653
dtype: float64
```

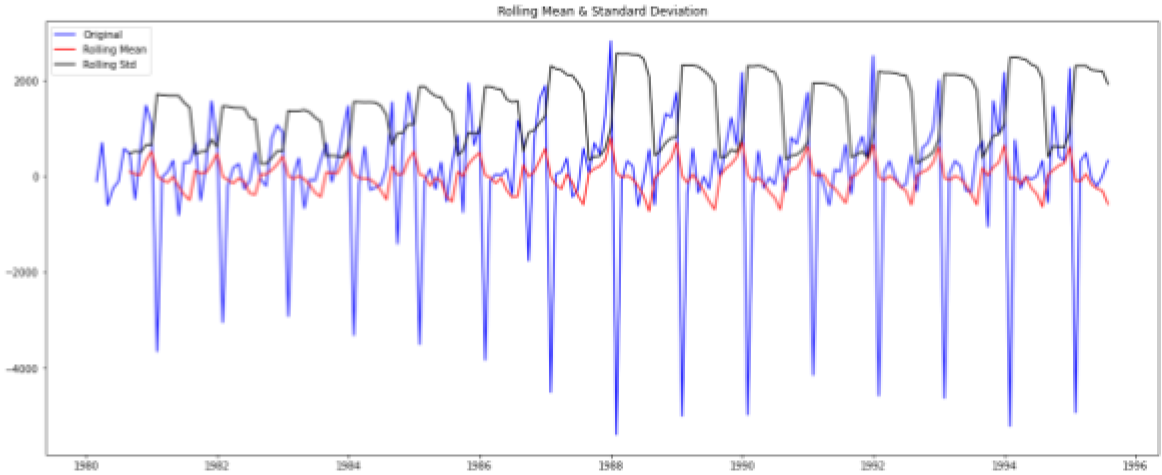
Fig.58

From above graph, it is clear that rolling mean and standard deviation is not constant at every data point. So that it is not stationary.

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

Make it Stationary:

- We have taken a difference of order 1 and check whether the Time Series is stationary or not. After performing ADF test on differenced of order 1 data we get p-value = 0.00
- Hence, p-value is lower than alpha, we can reject the null hypothesis and can say that at 5% significant level the Time Series is stationary




```
Results of Dickey-Fuller Test:
Test Statistic                 -45.050301
p-value                          0.000000
#Lags Used                      10.000000
Number of Observations Used    175.000000
Critical Value (1%)             -3.468280
dtype: float64

Test Statistic                 -45.050301
p-value                          0.000000
#Lags Used                      10.000000
Number of Observations Used    175.000000
Critical Value (1%)             -3.468280
Critical Value (5%)             -2.878202
dtype: float64

Test Statistic                 -45.050301
p-value                          0.000000
#Lags Used                      10.000000
Number of Observations Used    175.000000
Critical Value (1%)             -3.468280
Critical Value (5%)             -2.878202
Critical Value (10%)            -2.575653
dtype: float64
```

Fig.59

We have performed ADF test on Train data also to check stationarity. We need to take difference of order 1 and to make the Time Series is stationary.

1.6 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.

- Here we performed both ARIMA and SARIMA.
- ARIMA(p,d,q) Model: ARIMA is defined by 3 parameters
- p: No of autoregressive terms
- d: No of differencing to stationaries the series
- q: No of moving average terms

- Here we have taken
- p value = 0 to 3,
- q value = 0 to 3 and
- d = 1, as we have taken difference of order 1 to make the series stationary.
- After that, to calculate AIC value we used combination of p,d,q and sort the result to get the lowest AIC

```
Some parameter combinations for the Model...
Model: (0, 1, 1)
Model: (0, 1, 2)
Model: (1, 1, 0)
Model: (1, 1, 1)
Model: (1, 1, 2)
Model: (2, 1, 0)
Model: (2, 1, 1)
Model: (2, 1, 2)
```

Fig.60

	param	AIC
8	(2, 1, 2)	2210.618508
7	(2, 1, 1)	2232.360490
2	(0, 1, 2)	2232.783098
5	(1, 1, 2)	2233.597647
4	(1, 1, 1)	2235.013945
6	(2, 1, 0)	2262.035600
1	(0, 1, 1)	2264.906437
3	(1, 1, 0)	2268.528061
0	(0, 1, 0)	2269.582796

Fig.61

```
                              ARIMA Model Results
==============================================================================
===
Dep. Variable:            D.Sparkling   No. Observations:
131
Model:                 ARIMA(0, 1, 2)   Log Likelihood             -1112.
392
Method:                       css-mle   S.D. of innovations         1159.
696
Date:                Thu, 28 Oct 2021   AIC                         2232.
783
Time:                        19:14:16   BIC                         2244.
284
Sample:                    02-29-1980   HQIC                        2237.
456
                         - 12-31-1990
==============================================================================
==========
                        coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
----------
const                 6.2472      3.800      1.644      0.100      -1.201
13.696
ma.L1.D.Sparkling    -0.5555      0.073     -7.583      0.000      -0.699
-0.412
ma.L2.D.Sparkling    -0.4445      0.071     -6.247      0.000      -0.584
-0.305
                                    Roots
==============================================================================
==
                  Real          Imaginary           Modulus         Frequen
cy
------------------------------------------------------------------------------
--
MA.1            1.0000           +0.0000j            1.0000            0.00
00
MA.2           -2.2495           +0.0000j            2.2495            0.50
00
------------------------------------------------------------------------------
```

Fig.62

- We get the lowest Akaike Information Criteria (AIC) for (p,d,q) = (2,1,2)
- AIC for this order = 2210.618

Steps:

- Create combination of parameters for the model using 'itertools'
- Calculate AIC value for each combination of(p,d,q) .

- Sort the AIC value and got the lowest AIC value and parameter combination.
- Initiate ARIMA () classifier.
- Fit into train data set
- Predict the sale for test data

Predict on the Test Set using this model and evaluate the model: 1417.502239430773

Prediction graph is not following the trend with test data.

	RMSE
ARIMA(0,1,2)	1417.502239

Fig.63

Build an Automated version of a SARIMA model for which the best parameters are selected in accordance with the lowest Akaike Information Criteria (AIC):

	param	seasonal	AIC
50	(1, 1, 2)	(1, 0, 2, 12)	1555.584247
53	(1, 1, 2)	(2, 0, 2, 12)	1556.076770
26	(0, 1, 2)	(2, 0, 2, 12)	1557.121563
23	(0, 1, 2)	(1, 0, 2, 12)	1557.160507
77	(2, 1, 2)	(1, 0, 2, 12)	1557.340409

Fig.64

```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                                    y   No. Observations:                  132
Model:             SARIMAX(0, 1, 2)x(2, 0, 2, 12)   Log Likelihood                -771.561
Date:                              Thu, 28 Oct 2021   AIC                           1557.122
Time:                                      19:16:05   BIC                           1575.632
Sample:                                           0   HQIC                          1564.621
                                              - 132
Covariance Type:                                opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.7771      0.101     -7.680      0.000      -0.975      -0.579
ma.L2         -0.1239      0.121     -1.022      0.307      -0.361       0.114
ar.S.L12       0.6982      0.751      0.930      0.353      -0.774       2.170
ar.S.L24       0.3598      0.783      0.460      0.646      -1.174       1.894
ma.S.L12       1.6415      3.715      0.442      0.659      -5.640       8.923
ma.S.L24      -1.5933      2.663     -0.598      0.550      -6.813       3.627
sigma2      2.915e+04   9.51e+04      0.307      0.759   -1.57e+05    2.15e+05
===================================================================================
Ljung-Box (L1) (Q):                   0.03   Jarque-Bera (JB):                22.07
Prob(Q):                              0.85   Prob(JB):                         0.00
Heteroskedasticity (H):               1.45   Skew:                             0.49
Prob(H) (two-sided):                  0.28   Kurtosis:                         5.04
===================================================================================

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

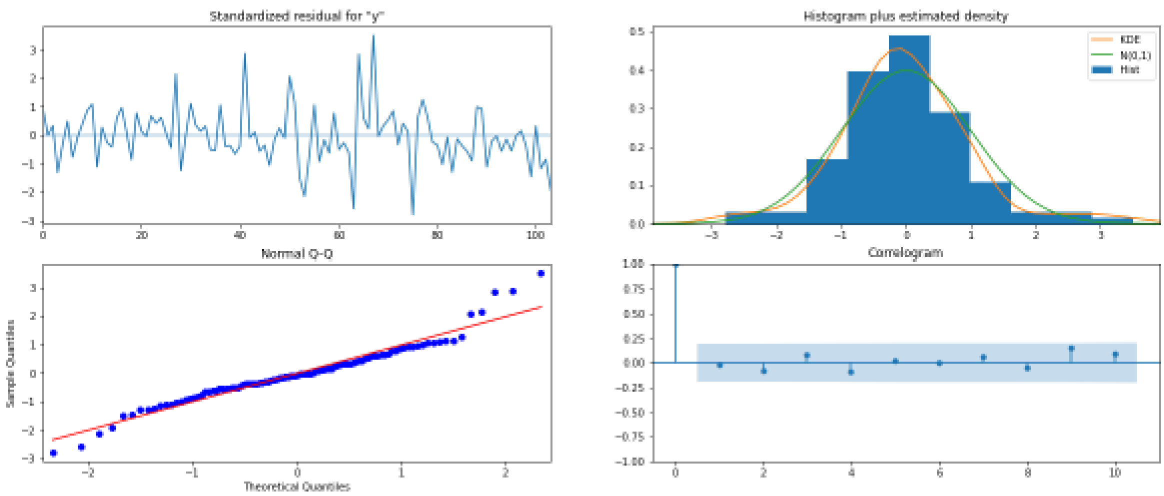


Fig.65

Predicting on the Test Set using this model and evaluate the model:

	RMSE
ARIMA(0,1,2)	1417.502239
SARIMA(0,1,2)(2,0,2,12)	526.462879

Fig.66

- Seasonal ARIMA model with seasonal frequency.
- These models are used when time series data has significant seasonality.
- The most general form of seasonal ARIMA is $ARIMA$(p,d,q)$*ARIMA$(p,d,q)[m], where p,d,q are defined as seasonal AR component, seasonal difference and seasonal MA component respectively. And, 'm' represents the frequency (time interval) at which the data is observed.
- To calculate AIC value, we have taken combination of (p,d,q) and [P,D,Q]. to make this combination we used 'itertools'.
- Here our m = 12

Steps:

- Create combination of parameters for the model using 'itertools'
- Calculate AIC value for each combination of(p,d,q) and [P, D,Q,m].
- Sort the AIC value and got the lowest AIC value and parameter combination.
- Initiate SARIMAX classifier.
- Fit into train data set
- Predict the sale for test data

- RMSE for test data: 526.462879
- We can see that predicted graph is following the trend with the test data.

1.7 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.

We have plotted ACF and PACF for differenced Train data.

ACF:

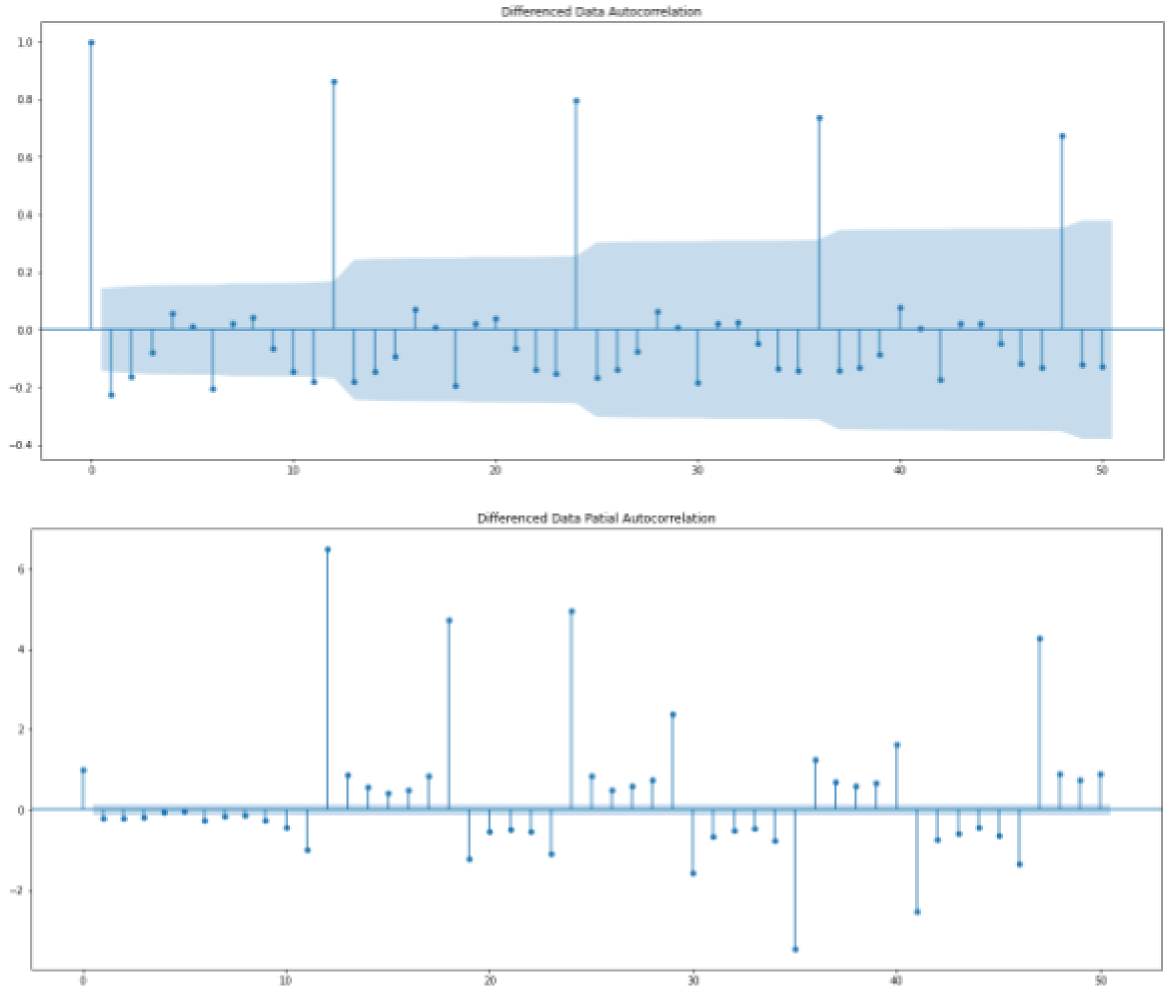


Fig.67

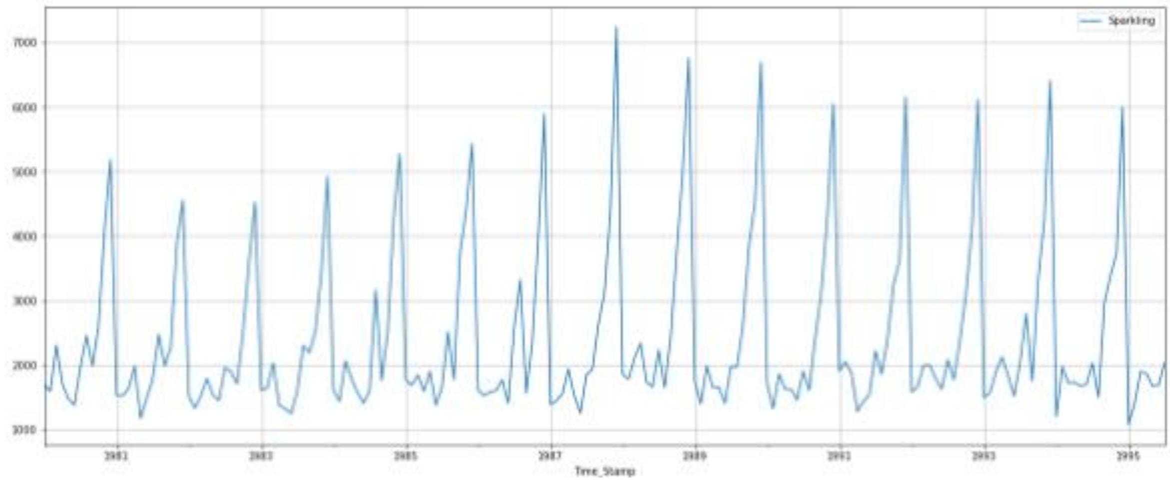


Fig.68

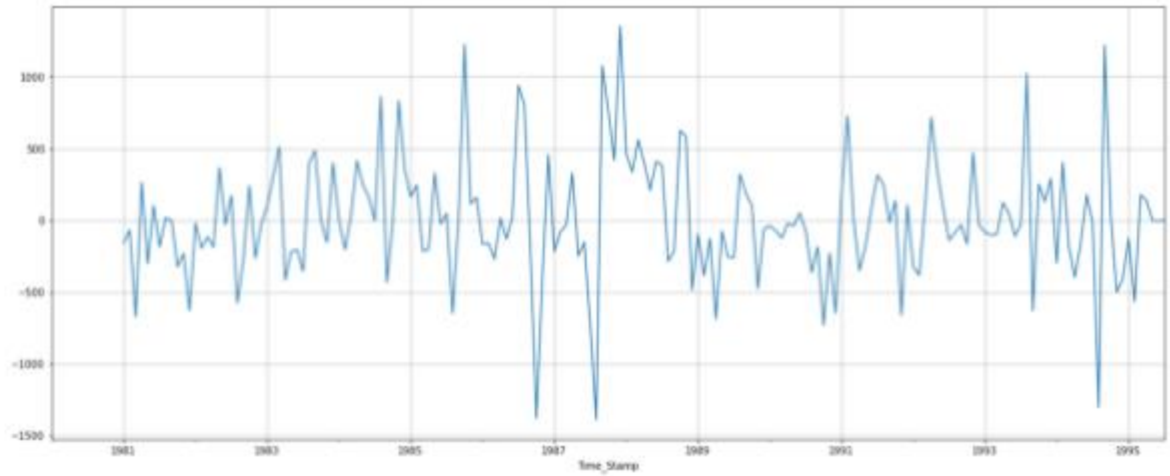


Fig.69

checking the stationarity of the above series before fitting the SARIMA model:


```
Results of Dickey-Fuller Test:
Test Statistic                  -3.342905
p-value                          0.013066
#Lags Used                      10.000000
Number of Observations Used    108.000000
Critical Value (1%)             -3.492401
dtype: float64

Test Statistic                  -3.342905
p-value                          0.013066
#Lags Used                      10.000000
Number of Observations Used    108.000000
Critical Value (1%)             -3.492401
Critical Value (5%)             -2.888697
dtype: float64

Test Statistic                  -3.342905
p-value                          0.013066
#Lags Used                      10.000000
Number of Observations Used    108.000000
Critical Value (1%)             -3.492401
Critical Value (5%)             -2.888697
Critical Value (10%)            -2.581255
dtype: float64
```

Fig.70

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

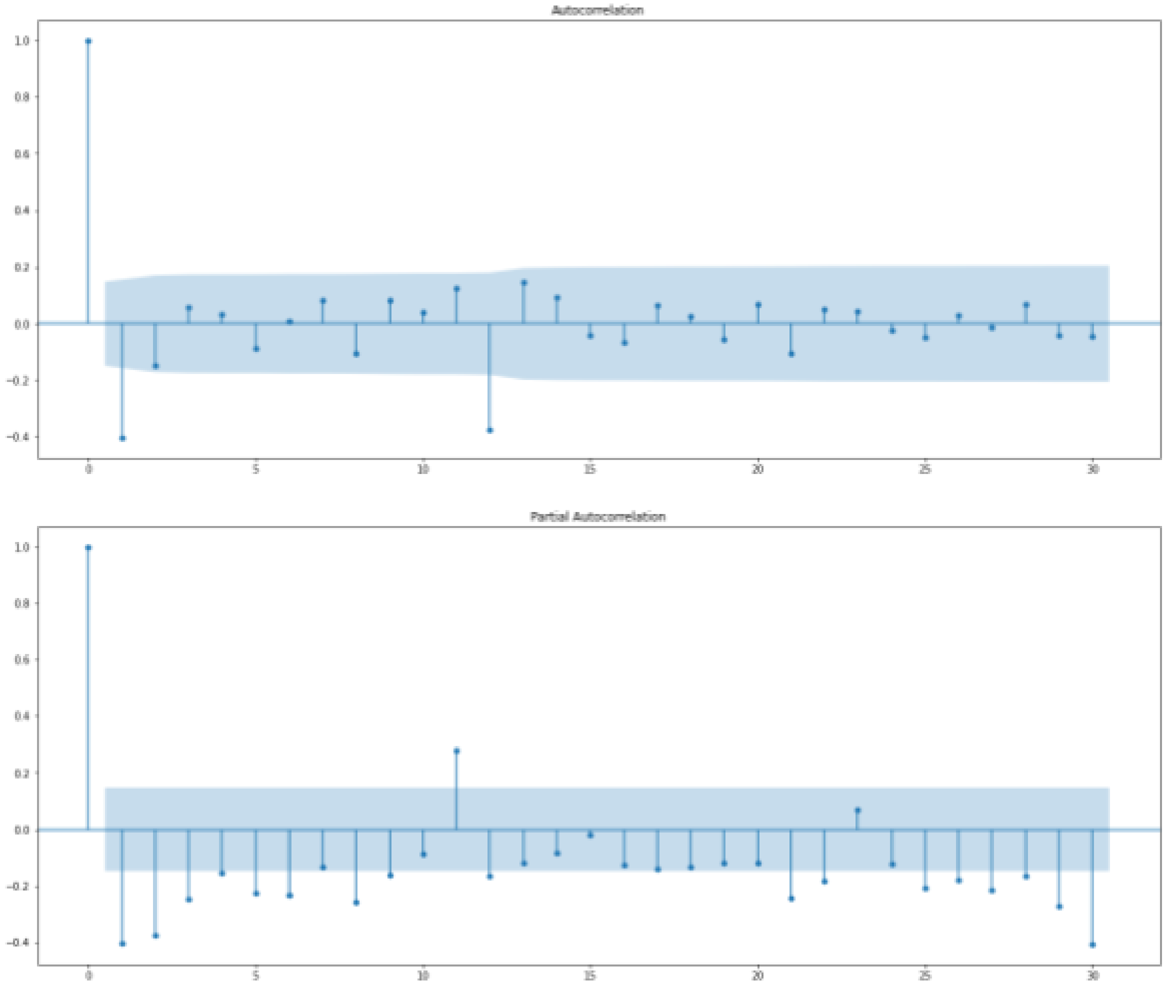


Fig.71

We are going to take the seasonal period as 12:

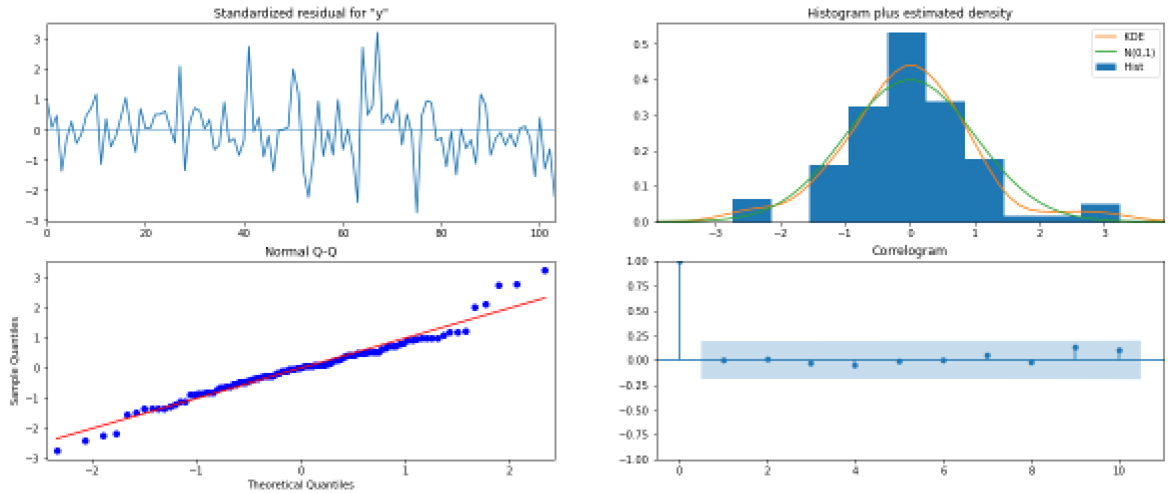


Fig.72

Predicting on the Test Set using this model and evaluate the model:

y	mean	mean_se	mean_ci_lower	mean_ci_upper
0	1311.826675	389.899488	547.637721	2076.015628
1	1300.651859	400.572863	515.543475	2085.760243
2	1562.780641	400.600108	777.618856	2347.942426
3	1567.089134	408.005644	767.412766	2366.765502
4	1344.581341	409.042152	542.873455	2146.289228

Fig.73

RMSE Score: 580.6029005672854

	RMSE
ARIMA(0,1,2)	1417.502239
SARIMA(0,1,2)(2,0,2,12)	526.462879
SARIMA(2,1,2)(2,0,2,12)	580.602901

Fig.74

Manual ARIMA model:

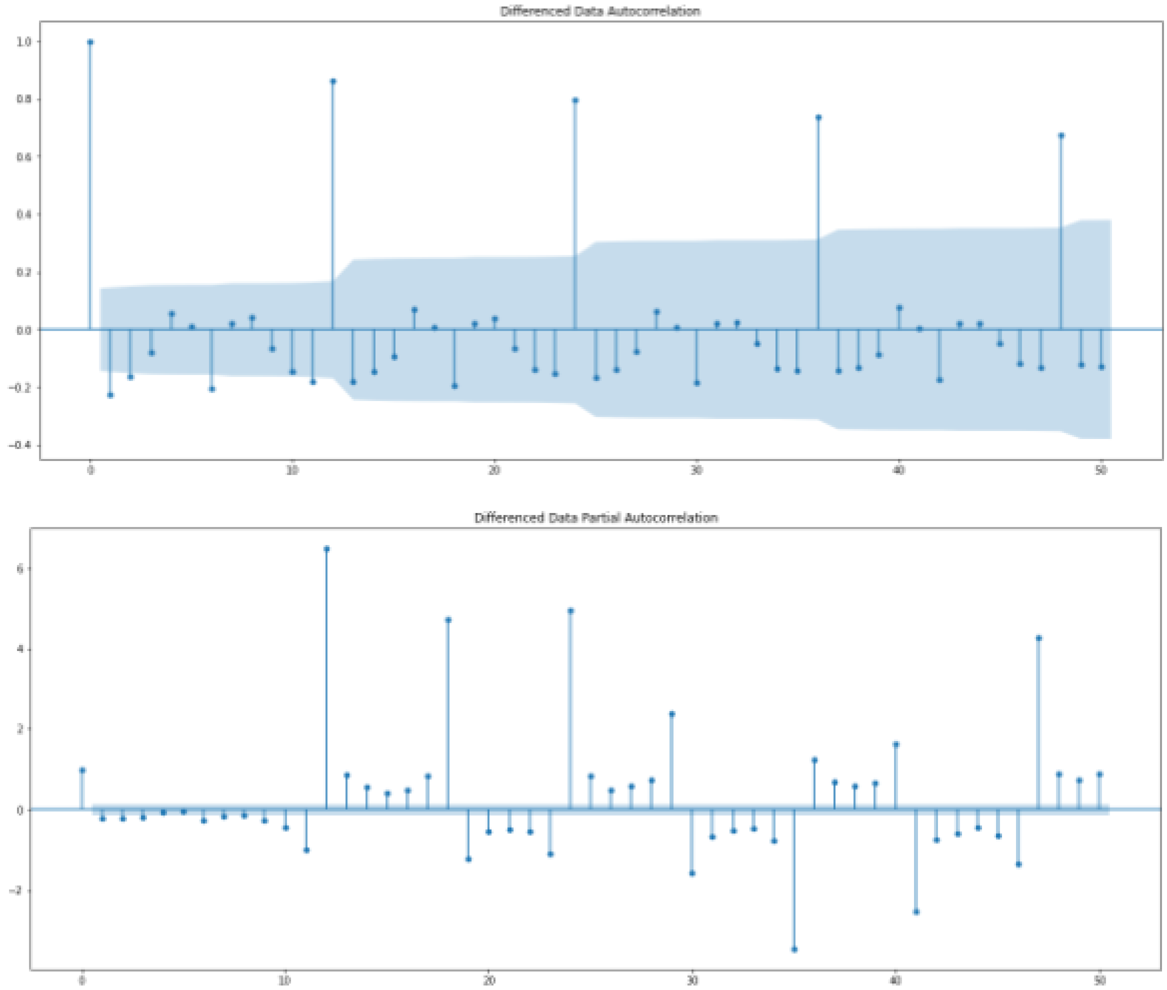


Fig.75

Predicting on the Test Set using this model and evaluate the model:

RMSE Score: 1374.6638702744171

	RMSE
ARIMA(0,1,2)	1417.502239
SARIMA(0,1,2)(2,0,2,12)	526.462879
SARIMA(2,1,2)(2,0,2,12)	580.602901
ARIMA(2,1,2)	1374.663870

Fig.76

- From above ACF plot we can see after lag zero, there are significant lag, as all lags are inside the shaded area or cut off is = 0
- Hence, q (moving average terms) = 2
- From above PACF plot we can see after lag zero, there are significant lag, as all lags are inside the shaded area or cut off is = 0
- Hence, p (moving average terms) = 2
- The value of d = 1, as we have taken difference of order 1
- From above plot we can see that seasonality is repeating at lag 12. Thus we will take P = 0 and Q = 0 and m =12
- As we don’t need to perform any difference for seasonality, thus D = 0
- We are going to plot of SARIMA model because of data has seasonality
- Building a version of the SARIMA model for which the best parameters are selected by looking at the ACF and the PACF plots. - Seasonality at 12.
- 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.
- From the original series. We see that there is a seasonality. So, now we take a seasonal differencing and check the 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.
- Let us go ahead and check the stationarity of the above series before fitting the SARIMA model.
- We see that at α = 0.05 the Time Series is indeed stationary.
- Checking the ACF and the PACF plots for the new modified Time Series.
- Here, we have taken alpha=0.05.
- We are going to take the seasonal period as 12.
- The Auto-Regressive parameter in an SARIMA model is 'P' which comes from the significant lag after which the PACF plot cuts-off to 2.
- The Moving-Average parameter in an SARIMA model is 'q' which comes from the significant lag after which the ACF plot cuts-off to 1.
- Remember to check the ACF and the PACF plots only at multiples of 12 (since 12 is the seasonal period)
- 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 2.
- The Moving-Average parameter in an ARIMA model is 'q' which comes from the significant lag before the ACF plot cuts-off to 1.

1.8 Build a table (create a data frame) with all the models built along with their corresponding parameters and the respective RMSE values on the test data.

We have created a data frame with all the models built along with their corresponding parameters and the respective RMSE values on the test data.

	Train_RMSE	Test_RMSE
Alpha=0.3,Beta=0.4,Gamma=0.3,TripleExponentialSmoothing	404.513320	392.786198
Alpha=0.064,Beta=0.053,Gamma=0.0,TripleExponentialSmoothing	356.782453	463.501976
SARIMA(0,1,2)(2,0,2,12)	NaN	526.462879
SARIMA(2,1,2)(2,0,2,12)	NaN	580.602901
2pointTrailingMovingAverage	NaN	813.400684
4pointTrailingMovingAverage	NaN	1156.589694
Simple Average	1298.483628	1275.081804
6pointTrailingMovingAverage	NaN	1283.927428
9pointTrailingMovingAverage	NaN	1346.278315
ARIMA(2,1,2)	NaN	1374.663870
RegressionOnTime	1279.322346	1389.135175
ARIMA(0,1,2)	NaN	1417.502239
Alpha=0.3,SimpleExponentialSmoothing	1338.805381	1935.507132
Alpha=0.995,SimpleExponentialSmoothing	1372.054747	3855.940897
Naive Model	3867.700802	3864.279352
Alpha=0.3,Beta=0.3,DoubleExponentialSmoothing	1500.689062	18259.110704

Fig.77

1.9 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.

```
{'smoothing_level': 0.07543819037108564,
 'smoothing_trend': 0.07543816249429056,
 'smoothing_seasonal': 0.27294445154459185,
 'damping_trend': nan,
 'initial_level': 1577.446620339965,
 'initial_trend': -13.373823757745805,
 'initial_seasons': array([1.06875929, 1.02018982, 1.47799192, 1.19853279,
0.98227378,
        0.9560861 , 1.32099751, 1.70927388, 1.39014226, 1.89705741,
        2.90535421, 3.77843847]),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

Fig.78

```
1995-08-31    1929.565002
1995-09-30    2347.047048
1995-10-31    3172.507937
1995-11-30    3909.340833
1995-12-31    5969.868543
1996-01-31    1353.953302
1996-02-29    1593.630283
1996-03-31    1824.817614
1996-04-30    1784.274585
1996-05-31    1635.069228
1996-06-30    1549.085128
1996-07-31    1955.711909
Freq: M, dtype: float64
```

Fig.79

Plotting The Forecast with Confidence Bond:

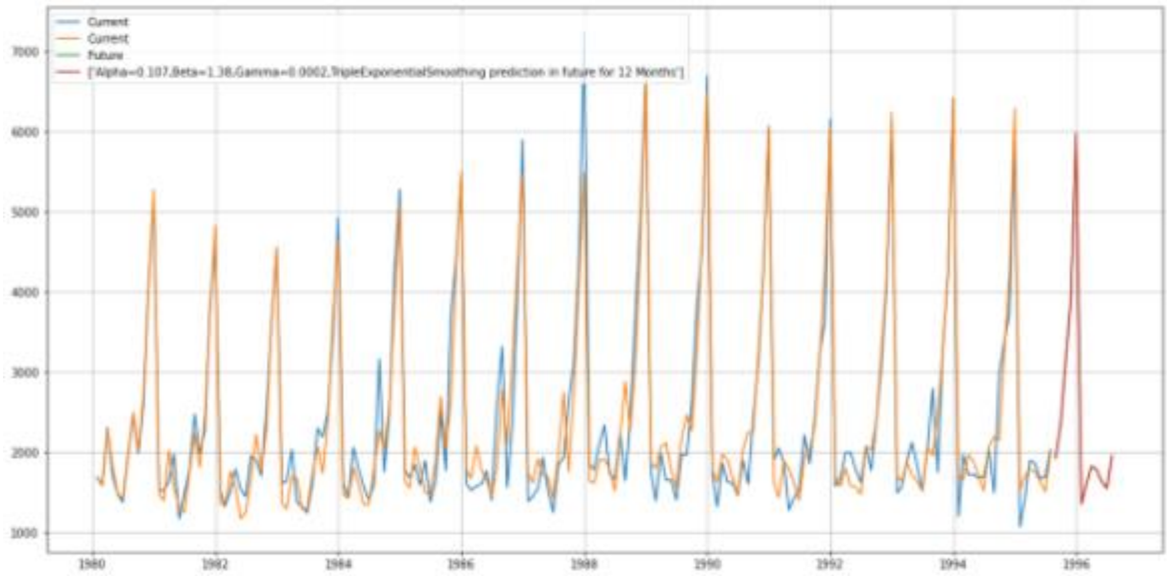


Fig.80

Final Result: For Alpha=0.107, Beta=1.38, Gamma=0.0002, Triple exponential Smoothing model Forecasting 346.1249703143661

Predicted Model for the Sparkling Sales for final Model:

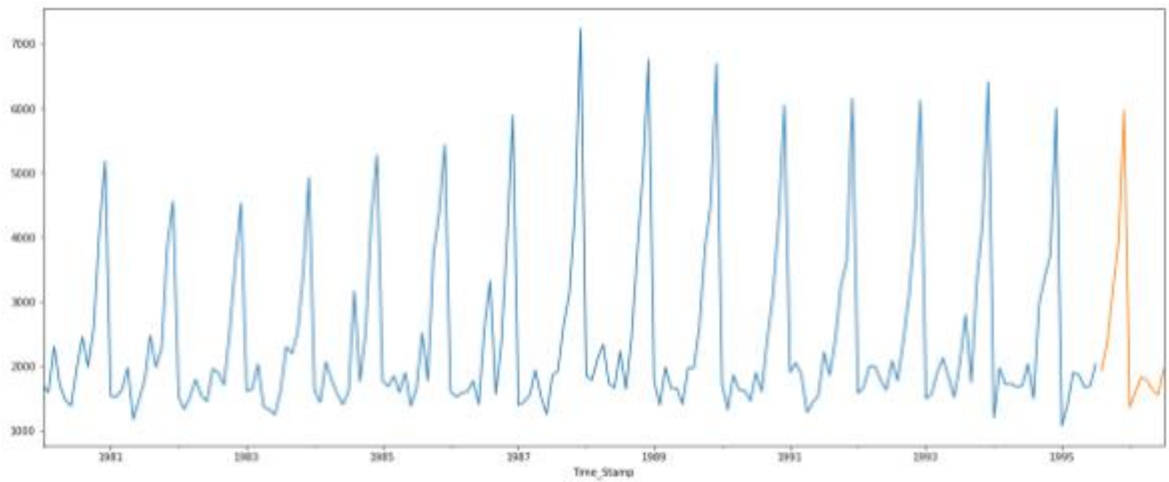


Fig.81

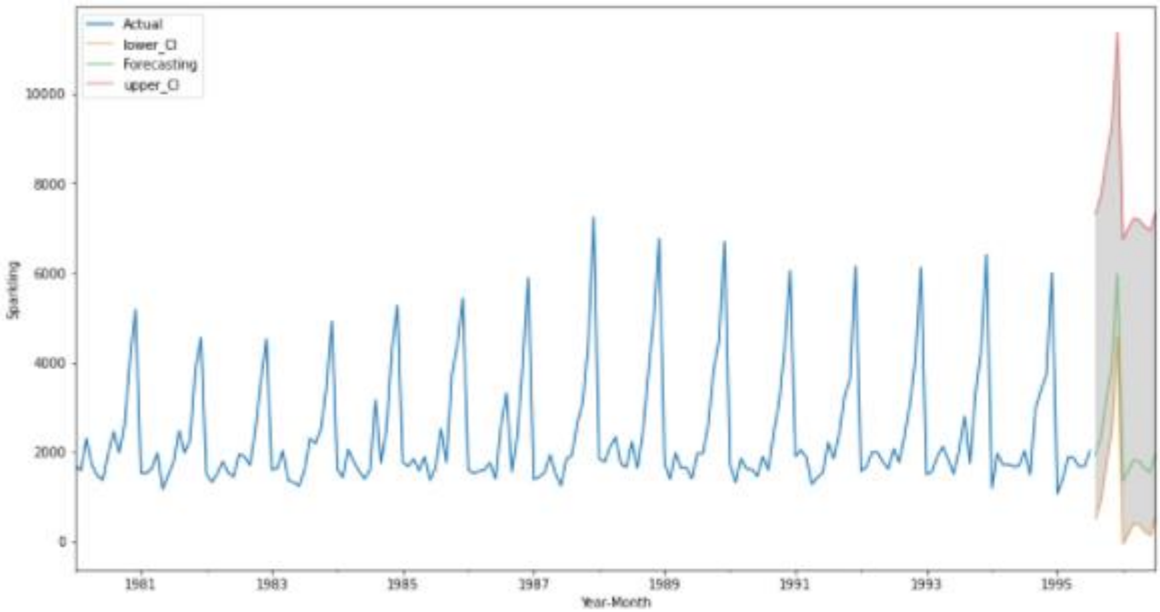


Fig.82

Prediction graph is following the previous trend for next 12 month.

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

- Our data set has mixed trend and seasonality. First it was decreasing then it is increased a bit then there is again a drop in sale and after that it is quite same.
- We have added Time stamp column to convert time series instance to Time stamp datatype.
- We have split our data set into train and test. Train set contain before 1991-year data and Test set contains all the data from 1991
- Based on analysis we clearly say that the
- Sparkling sales shows stabilized values and not much trend compared to previous years.
- December month shows the highest Sales across the years from 1980-1994.
- The models are built considering the Trend and Seasonality in to account and we see from the output plot that the future prediction is in line with the trend and seasonality in the previous years.
- The Sales of Sparling wine is seasonal; hence the company cannot have the same stock through the year. The predictions would help here to plan the Stock need basis the forecasted sales.

- The company should use the prediction results and capitalize on the high demand seasons and ensure to source and supply the high demand.
- The company should use the prediction results to plan the low demand seasons to stock as per the demand.

Recommendation:

- As we can see there is a chance of highest sale in November – December we can use this time.
- This is a festive season. Christmas, Thanksgiving Day will come in this time period.
- So, if the company introduce some new offer (such as: buy one get 1), discount, offer for bulk buyer, many people can be interested to buy this wine and they can increase their sale.
- Although the company can check the product quality, if it is value for money or not.

Here we are analysing and forecasting the sale of Rose wine in the 20th century. Rose wine is a product of ABC Estate Wines.

1.1 Read the data as an appropriate Time Series data and plot the data.

<u>Data loading and overview:</u>

	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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 187 entries, 0 to 186
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   YearMonth  187 non-null    object 
 1   Rose       185 non-null    float64
dtypes: float64(1), object(1)
memory usage: 3.0+ KB
```

Fig.83

```
YearMonth
1980-01-01    112.0
1980-02-01    118.0
1980-03-01    129.0
1980-04-01     99.0
1980-05-01    116.0
Name: Rose, dtype: float64


YearMonth
1995-03-01    45.0
1995-04-01    52.0
1995-05-01    28.0
1995-06-01    40.0
1995-07-01    62.0
Name: Rose, dtype: float64
```

Fig.84

```
DatetimeIndex(['1980-01-31', '1980-02-29', '1980-03-31', '1980-04-30',
               '1980-05-31', '1980-06-30', '1980-07-31', '1980-08-31',
               '1980-09-30', '1980-10-31',
               ...
               '1994-10-31', '1994-11-30', '1994-12-31', '1995-01-31',
               '1995-02-28', '1995-03-31', '1995-04-30', '1995-05-31',
               '1995-06-30', '1995-07-31'],
              dtype='datetime64[ns]', length=187, freq='M')
```

Fig.85

	YearMonth	Rose	Time_Stamp
0	1980-01	112.0	1980-01-31
1	1980-02	118.0	1980-02-29
2	1980-03	129.0	1980-03-31
3	1980-04	99.0	1980-04-30
4	1980-05	116.0	1980-05-31

Fig.86

	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

Fig.87

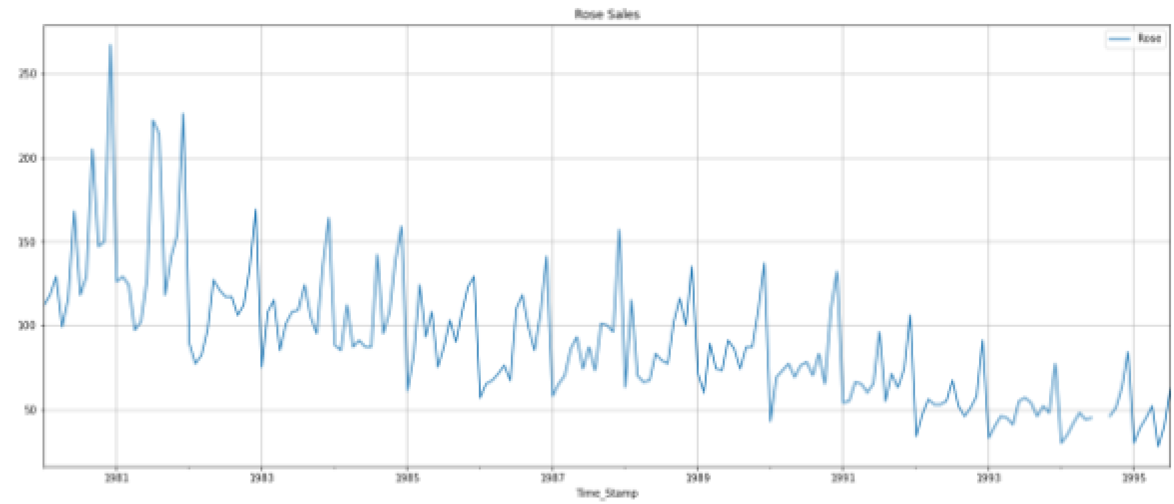


Fig.88

Inferences:

- Here we have collected monthly sales data of Rose wine, starting from January,1980 to July,1995 which is of 15 years. We have total 187 records.
- In this data set, we have two columns (YearMonth and Rose) and 187 rows.
- Here 'Year Month' column is of object data type which is indicating the time of sale and 'Rose' column is of float data type which gives us the value of Rose wine sale.
- As this is a Time series data, so 'YearMonth' column should be in Timestamp format not in object type.
- Therefore, we have added a timestamp column('Time_Stamp') according to our 'YearMonth' column value in our data set.
- As it is recommended that for time series analysis, we should put Timeseries reference column as Index because it makes easy for slicing and dicing the data for future analysis.
- Therefore, we make our new Time stamp column 'Time_Stamp' as index and drop 'YearMonth' column because its value is same with new column 'Time_Stamp Two null values in this data.
- The plot depicts a decreasing trend of sales over the period of 1980 to 1995. The Seasonality seems to have pattern on yearly basis.

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

Checking null values for Rose wine sales:

```
Rose     2
dtype: int64
```

Fig.89

Interpolating null values:

```
Rose     0
dtype: int64
```

Descriptive Analysis of Rose Wine Sales:

	Rose
count	187.000000
mean	89.914439
std	39.238325
min	28.000000
25%	62.500000
50%	85.000000
75%	111.000000
max	267.000000

Fig.90

- However, for this measure of descriptive statistics we have averaged over the whole data without taking the time component into account hence should look at the box plots year wise and month wise

Plotting Boxplot for year wise:

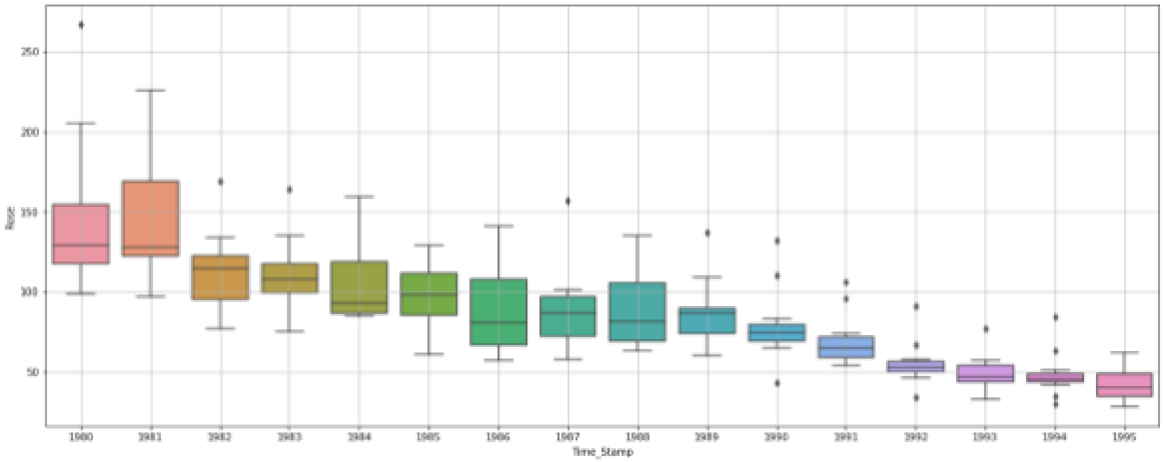


Fig.91

Plotting Boxplot for Month wise:

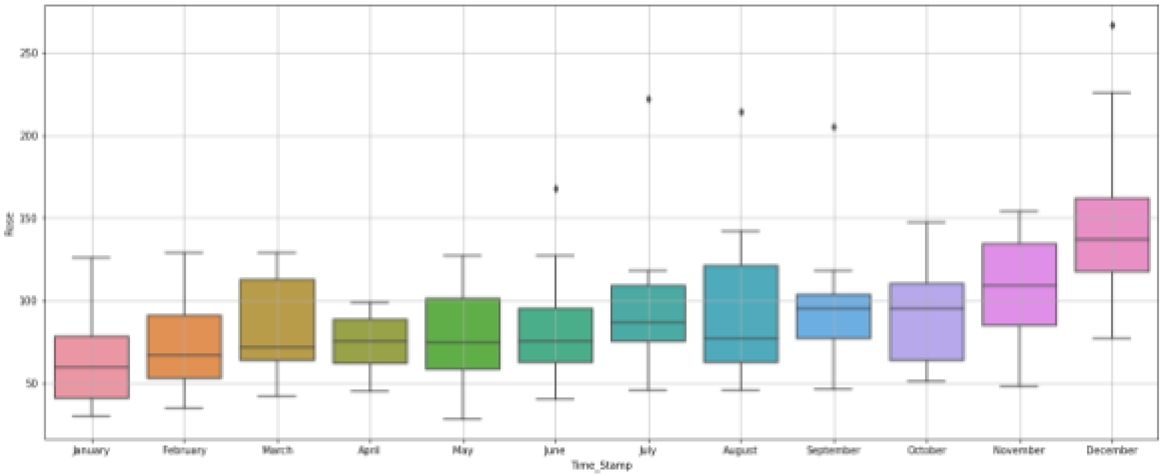


Fig.92

Plotting the Rose wine year month wise sales - Line plot:

Time_Stamp	April	August	December	February	January	July	June	March	May	November	October	September
Time_Stamp												
1980	99.0	129.000000	267.0	118.0	112.0	118.000000	168.0	129.0	116.0	150.0	147.0	205.0
1981	97.0	214.000000	226.0	129.0	126.0	222.000000	127.0	124.0	102.0	154.0	141.0	118.0
1982	97.0	117.000000	169.0	77.0	89.0	117.000000	121.0	82.0	127.0	134.0	112.0	106.0
1983	85.0	124.000000	164.0	108.0	75.0	109.000000	108.0	115.0	101.0	135.0	95.0	105.0
1984	87.0	142.000000	159.0	85.0	88.0	87.000000	87.0	112.0	91.0	139.0	108.0	95.0
1985	93.0	103.000000	129.0	82.0	61.0	87.000000	75.0	124.0	108.0	123.0	108.0	90.0
1986	71.0	118.000000	141.0	65.0	57.0	110.000000	67.0	67.0	76.0	107.0	85.0	99.0
1987	86.0	73.000000	157.0	65.0	58.0	87.000000	74.0	70.0	93.0	96.0	100.0	101.0
1988	66.0	77.000000	135.0	115.0	63.0	79.000000	83.0	70.0	67.0	100.0	116.0	102.0
1989	74.0	74.000000	137.0	60.0	71.0	86.000000	91.0	89.0	73.0	109.0	87.0	87.0
1990	77.0	70.000000	132.0	69.0	43.0	78.000000	76.0	73.0	69.0	110.0	65.0	83.0
1991	65.0	55.000000	106.0	55.0	54.0	96.000000	65.0	66.0	60.0	74.0	63.0	71.0
1992	53.0	52.000000	91.0	47.0	34.0	67.000000	55.0	56.0	53.0	58.0	51.0	46.0
1993	45.0	54.000000	77.0	40.0	33.0	57.000000	55.0	46.0	41.0	48.0	52.0	46.0
1994	48.0	45.666667	84.0	35.0	30.0	45.333333	45.0	42.0	44.0	63.0	51.0	46.0
1995	52.0	NaN	NaN	39.0	30.0	62.000000	40.0	45.0	28.0	NaN	NaN	NaN

Fig.11

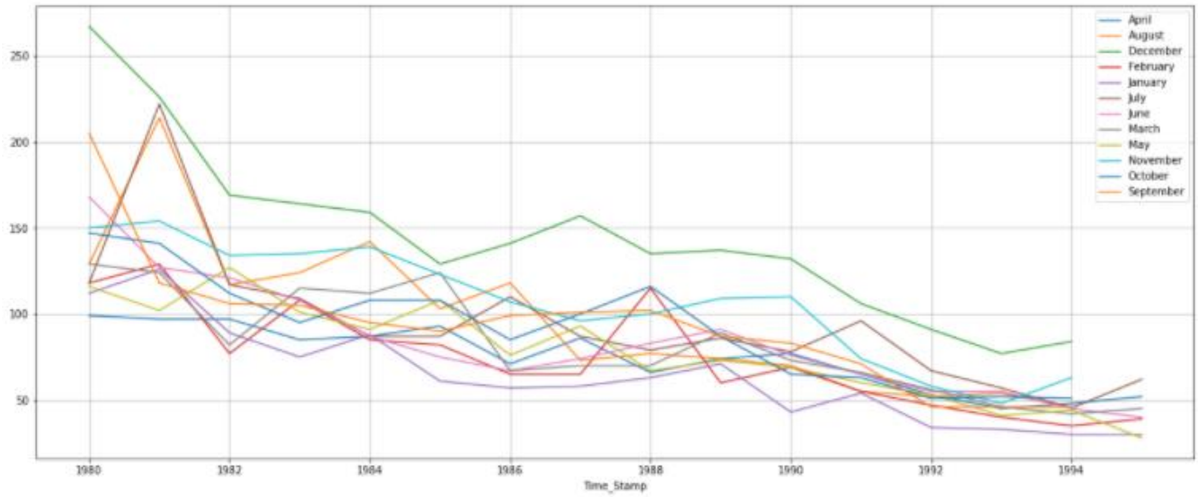


Fig.92

Plotting a month plot to check the sales in different years and within different month across:

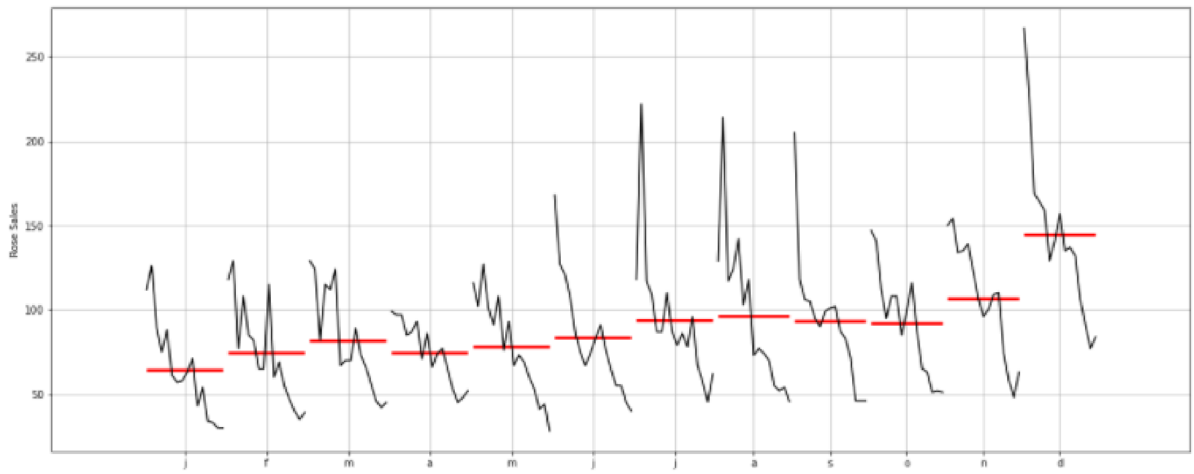


Fig.93

Plotting the Empirical Cumulative Distribution:

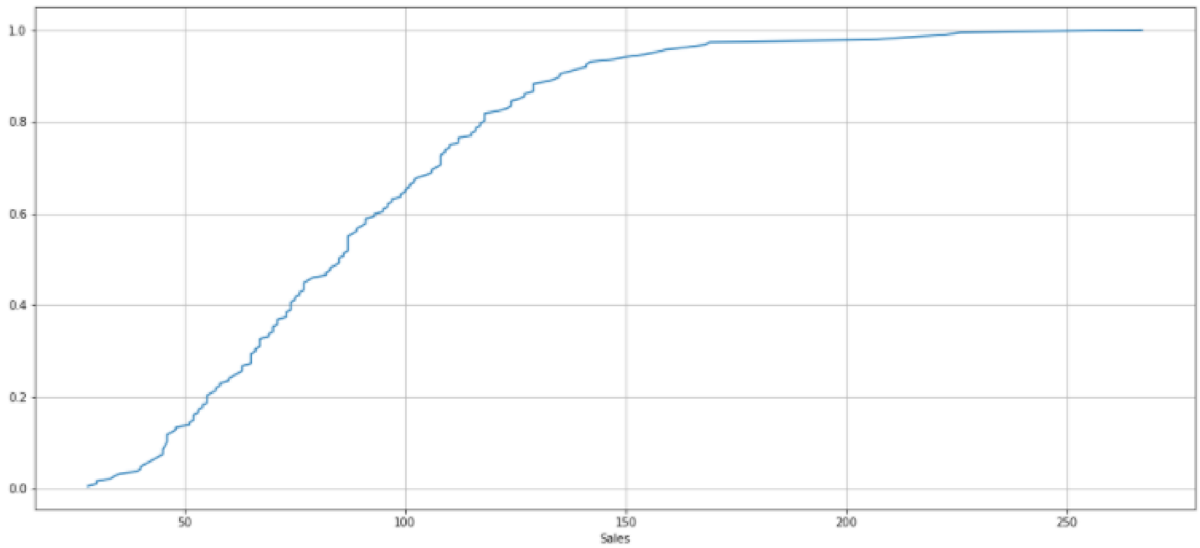


Fig.94

Plot the average Rose Wine Sales per month and the month-on-month percentage change of Rose Wine Sales:

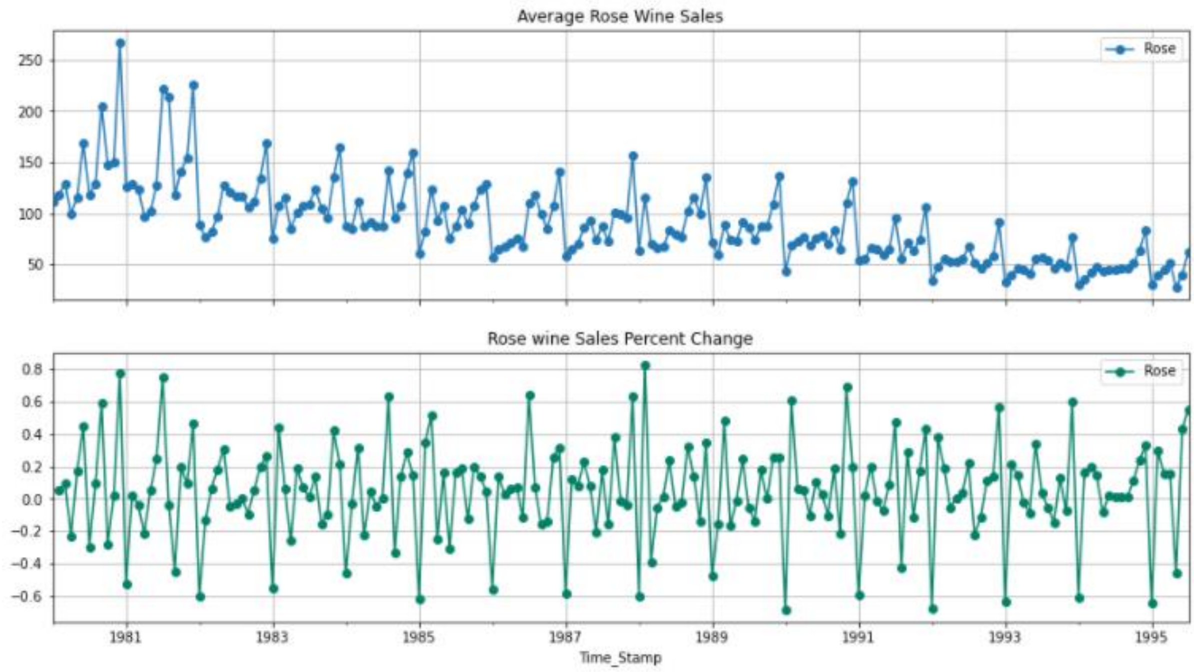


Fig.95

Decomposing the time series into Additive decomposition and plot:

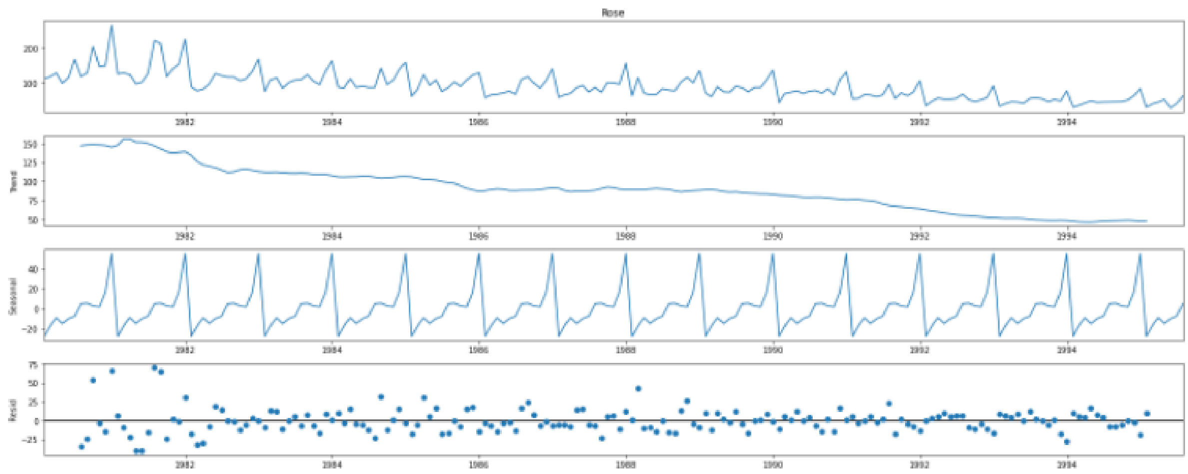


Fig.96

```
Trend
 Time_Stamp
1980-01-31           NaN
1980-02-29           NaN
1980-03-31           NaN
1980-04-30           NaN
1980-05-31           NaN
1980-06-30           NaN
1980-07-31    147.083333
1980-08-31    148.125000
1980-09-30    148.375000
1980-10-31    148.083333
1980-11-30    147.416667
1980-12-31    145.125000
Name: trend, dtype: float64

Seasonality
 Time_Stamp
1980-01-31   -27.908647
1980-02-29   -17.435632
1980-03-31    -9.285830
1980-04-30   -15.098330
1980-05-31   -10.196544
1980-06-30    -7.678687
1980-07-31     4.896908
1980-08-31     5.499686
1980-09-30     2.774686
1980-10-31     1.871908
1980-11-30    16.846908
1980-12-31    55.713575
Name: seasonal, dtype: float64

Residual
 Time_Stamp
1980-01-31          NaN
1980-02-29          NaN
1980-03-31          NaN
1980-04-30          NaN
1980-05-31          NaN
1980-06-30          NaN
1980-07-31   -33.980241
1980-08-31   -24.624686
1980-09-30    53.850314
1980-10-31    -2.955241
1980-11-30   -14.263575
1980-12-31    66.161425
Name: resid, dtype: float64
```

Fig.97

Decomposing the time series into multiplicative decomposition and plot:

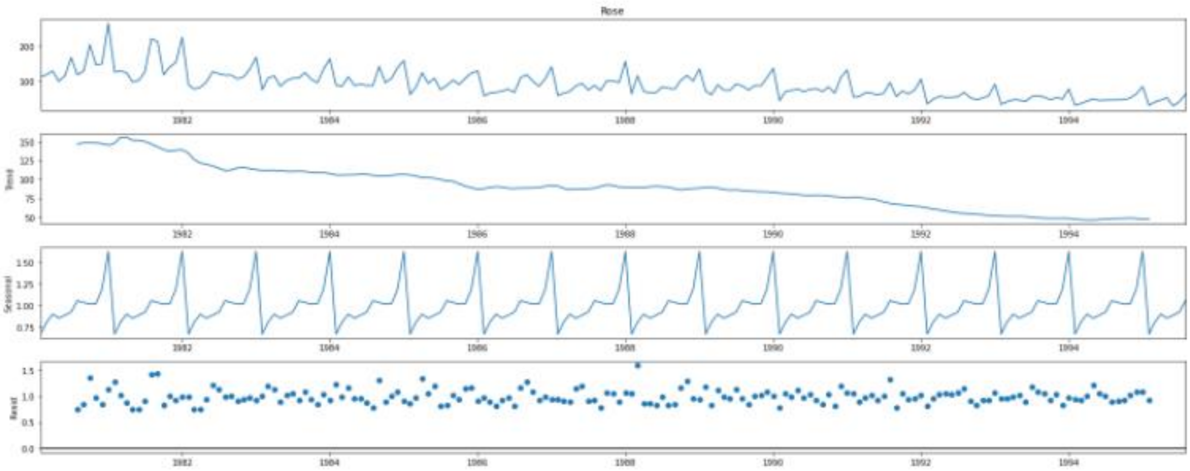


Fig.98

```
Trend
 Time_Stamp
1980-01-31           NaN
1980-02-29           NaN
1980-03-31           NaN
1980-04-30           NaN
1980-05-31           NaN
1980-06-30           NaN
1980-07-31    147.083333
1980-08-31    148.125000
1980-09-30    148.375000
1980-10-31    148.083333
1980-11-30    147.416667
1980-12-31    145.125000
Name: trend, dtype: float64

Seasonality
 Time_Stamp
1980-01-31    0.670111
1980-02-29    0.806163
1980-03-31    0.901164
1980-04-30    0.854024
1980-05-31    0.889415
1980-06-30    0.923985
1980-07-31    1.058038
1980-08-31    1.035881
1980-09-30    1.017648
1980-10-31    1.022573
1980-11-30    1.192349
1980-12-31    1.628646
Name: seasonal, dtype: float64

Residual
 Time_Stamp
1980-01-31         NaN
1980-02-29         NaN
1980-03-31         NaN
1980-04-30         NaN
1980-05-31         NaN
1980-06-30         NaN
1980-07-31    0.758258
1980-08-31    0.840720
1980-09-30    1.357674
1980-10-31    0.970771
1980-11-30    0.853378
1980-12-31    1.129646
Name: resid, dtype: float64
```

Fig.99

Inferences:

- Total 187 records and min sales of 28.0 and max sales of 267. 50 percentile is 85.
- We have datetime format and Float datatype (modified).
- There 2 missing values.
- Now we have to replace the null values, so we use the interpolate function and use the method as spline to generate values and replace them with missing values, as we know, we had 2 missing values in 1994, we have to replace them.
- In order to treat the missing values, we will make use of Interpolate method.
- There are 187 columns and 1 row in the dataset.
- The above picture shows the trend over the period between 1990 to 1995 with outliers in most of the years.
- And also shows the trend of sales in months, December seems to have the highest sales and January seems to have lowest sales.
- 1980 had the highest sales of all years in December.
- 1995 May had the least sales.
- February month has the decent sales of all months.
- Average sales drop over the years. Percentage drips as the year passes by.
- We now have all the values, so we can decompose our series to check the seasonality, trend and residual components.
- There are 2 methods- additive and multiplicative.
- As per the 'additive' decomposition, we see that there is a decreasing trend in sales of Rose wine
- starting from 1980 to 1994 sales has fallen only.
- Seasonality is also clearly visible from the seasonal graph where trend lines are forming the peaks with different height every year.
- Residuals seems to be scattered from the 0 level. Indicating that the series is not additive. As per the 'additive' decomposition, we see that there is a decreasing trend in sales of Rose wine starting from 1980 to 1994 sales has fallen only.
- Seasonality is also clearly visible from the seasonal graph where trend lines are forming the peaks with different height every year.
- Residuals seems to be scattered from the 0 level. Indicating that the series is not additive.
- Multiplicative Decomposition: The trend and seasonality are present same as in case of additive model. But residuals plot is clearly showing the concentration of data towards 1 point.
- Hence it can be concluded that series is multiplicative.

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

- Splitted the data into train and test data

Displaying multiple data frames from one cell:

```
First few rows of Training Data
```

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

```
Last few rows of Training Data
```

Time_Stamp	Rose
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

```
First few rows of Test Data
```

Time_Stamp	Rose
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
```

Time_Stamp	Rose
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.100

Shape of the train and test data:

```
(132, 1)
(55, 1)
```

plotting the graph for train and test set:

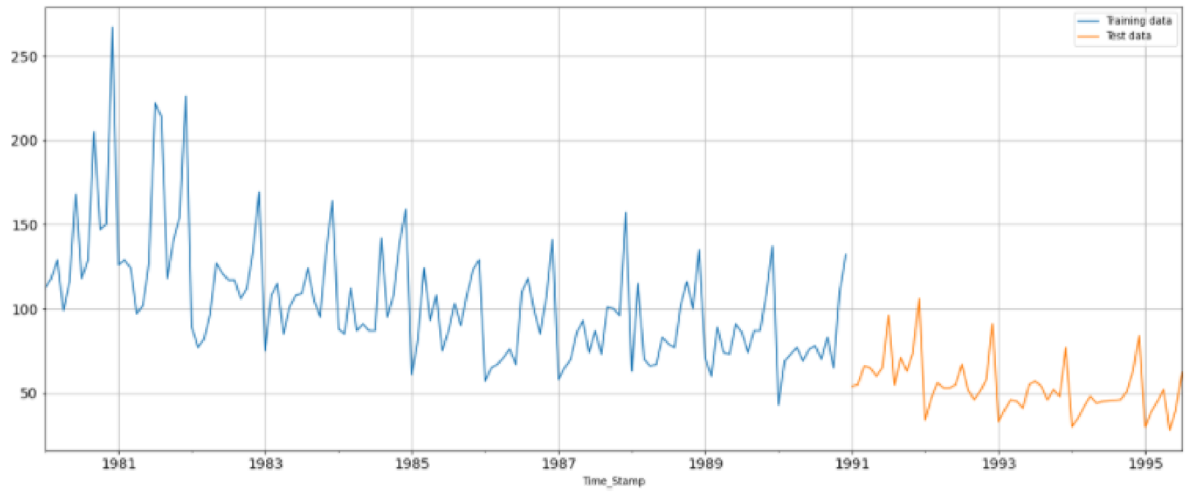


Fig.101

Inferences:

- The Train data of Rose wine sales has been split for data up to 1990 and has 132 data points.
- The Test data of Rose wine sales has been split for data from 1991 and has 55 data points.
- From our train-test split we are predicting the future sales as compared to the past years.

1.4 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 and simple average models. should also be built on the training data and check the performance on the test data using RMSE.

Model 1: Linear Regression

For this particular linear regression, we are going to regress the Rose 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, 2
1, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 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, 11
2, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 12
7, 128, 129, 130, 131, 132]


Test Time instance
 [133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 14
7, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 16
2, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 17
7, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187]
```

```
First few rows of Training Data
             Rose  time
Time_Stamp
1980-01-31  112.0     1
1980-02-29  118.0     2
1980-03-31  129.0     3
1980-04-30   99.0     4
1980-05-31  116.0     5

Last few rows of Training Data
             Rose  time
Time_Stamp
1990-08-31   70.0   128
1990-09-30   83.0   129
1990-10-31   65.0   130
1990-11-30  110.0   131
1990-12-31  132.0   132

First few rows of Test Data
            Rose  time
Time_Stamp
1991-01-31  54.0   133
1991-02-28  55.0   134
1991-03-31  66.0   135
1991-04-30  65.0   136
1991-05-31  60.0   137

Last few rows of Test Data
            Rose  time
Time_Stamp
1995-03-31  45.0   183
1995-04-30  52.0   184
1995-05-31  28.0   185
1995-06-30  40.0   186
1995-07-31  62.0   187
```

Fig.102

Now that our training and test data has been modified, let us go ahead use "Linear Regression" to build the model on the training data and test the model on the test data

Plotting Linear Regression Forecast:

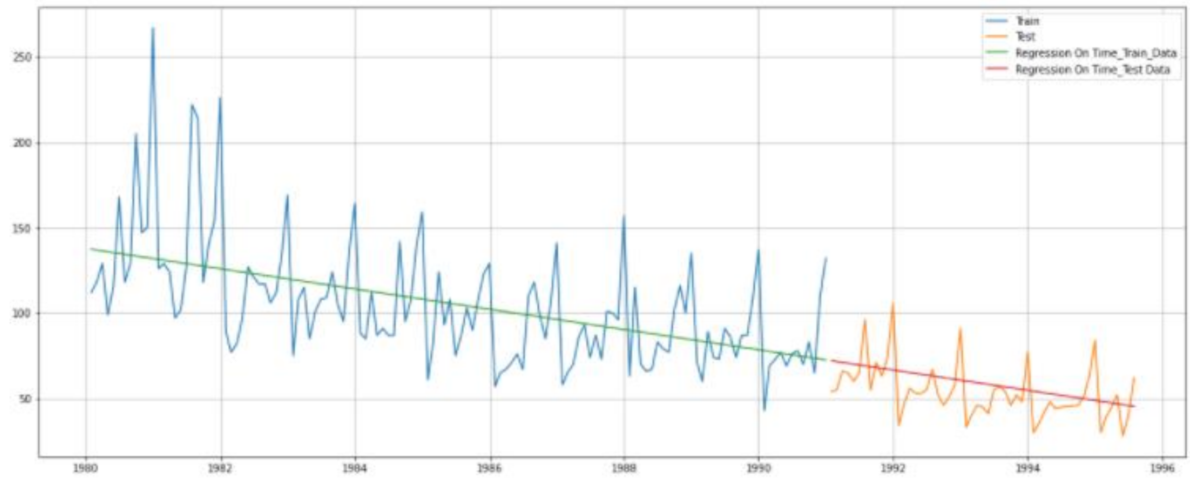


Fig.103

Model 1 Evaluation:

Train - RMSE score: For RegressionOnTime forecast on the Train Data, RMSE is 30.718

Test - RMSE score: For RegressionOnTime forecast on the Test Data, RMSE is 15.269

Creating Data frame:

	Train_RMSE	Test_RMSE
RegressionOnTime	30.718135	15.268955

Model 2: Naïve

```
Time_Stamp
1980-01-31    132.0
1980-02-29    132.0
1980-03-31    132.0
1980-04-30    132.0
1980-05-31    132.0
Name: naive, dtype: float64

Time_Stamp
1991-01-31    132.0
1991-02-28    132.0
1991-03-31    132.0
1991-04-30    132.0
1991-05-31    132.0
Name: naive, dtype: float64
```

Fig.104

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.

Plotting Naive Forecast:

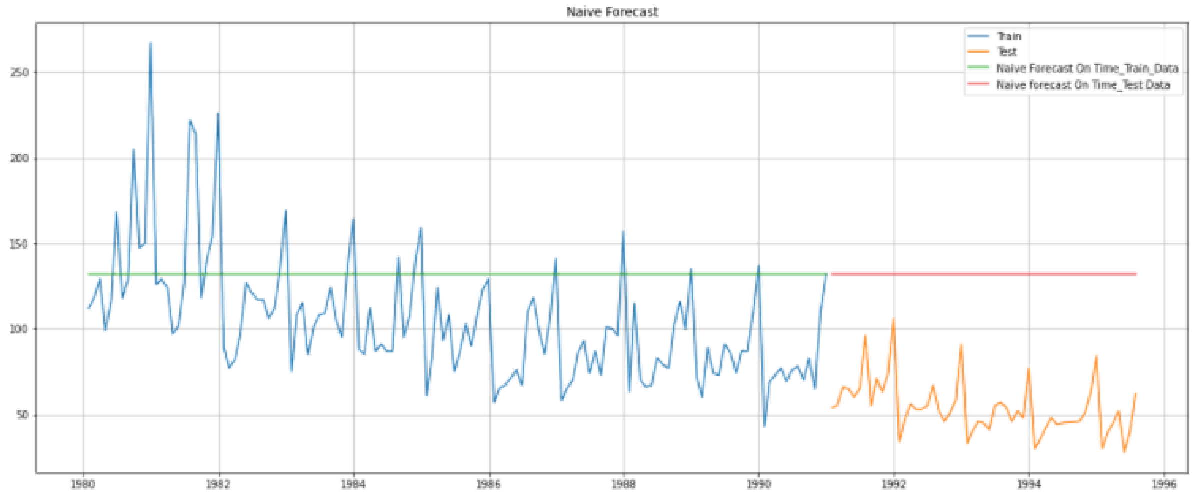


Fig.105

Model 2 Evaluation:

Train - RMSE score: For Naive Model forecast on the Train Data, RMSE is 45.064

Test - RMSE score: For Naive Model forecast on the Test Data, RMSE is 79.719

Creating Data frame:

	Train_RMSE	Test_RMSE
RegressionOnTime	30.718135	15.268955
Naive Model	45.063760	79.718773

Fig.105

Model 3: Simple Average

Time_Stamp	Rose	mean_forecast
1980-01-31	112.0	104.939394
1980-02-29	118.0	104.939394
1980-03-31	129.0	104.939394
1980-04-30	99.0	104.939394
1980-05-31	116.0	104.939394

Time_Stamp	Rose	mean_forecast
1991-01-31	54.0	104.939394
1991-02-28	55.0	104.939394
1991-03-31	66.0	104.939394
1991-04-30	65.0	104.939394
1991-05-31	60.0	104.939394

Fig.106

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

Plotting Simple Average Forecast:

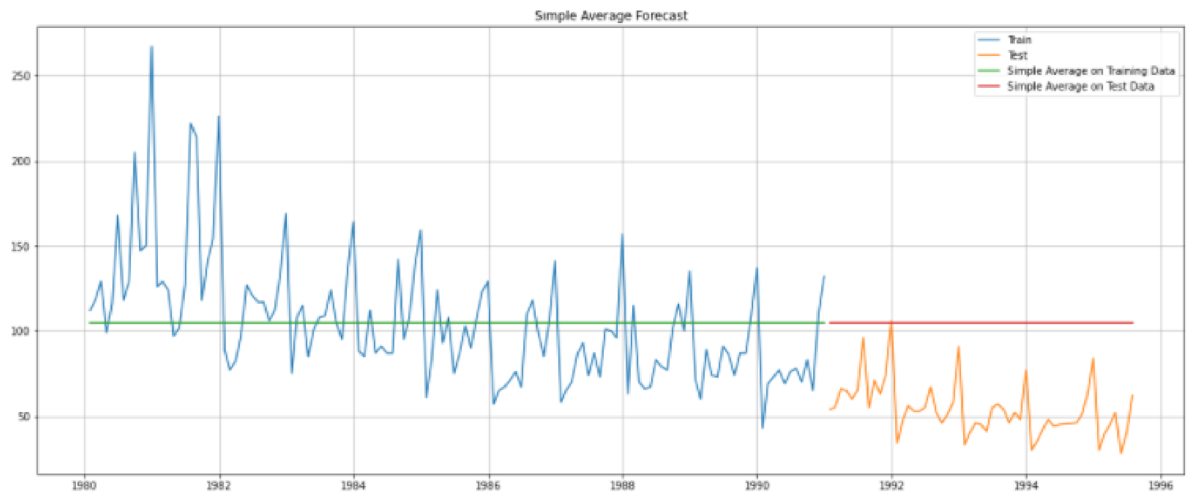


Fig.107

<u>Model 3 Evaluation:</u>

Train - RMSE score: For Simple Average Model forecast on the Train Data, RMSE is 36.034

Test - RMSE score: For Simple Average Model forecast on the Test Data, RMSE is 53.461

Creating Data frame:

	Train_RMSE	Test_RMSE
RegressionOnTime	30.718135	15.268955
Naive Model	45.063760	79.718773
Simple Average	36.034234	53.460570

Fig.108

Model 4: Moving Average

	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

	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

Fig.109

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.

Plotting Moving Average Forecast:

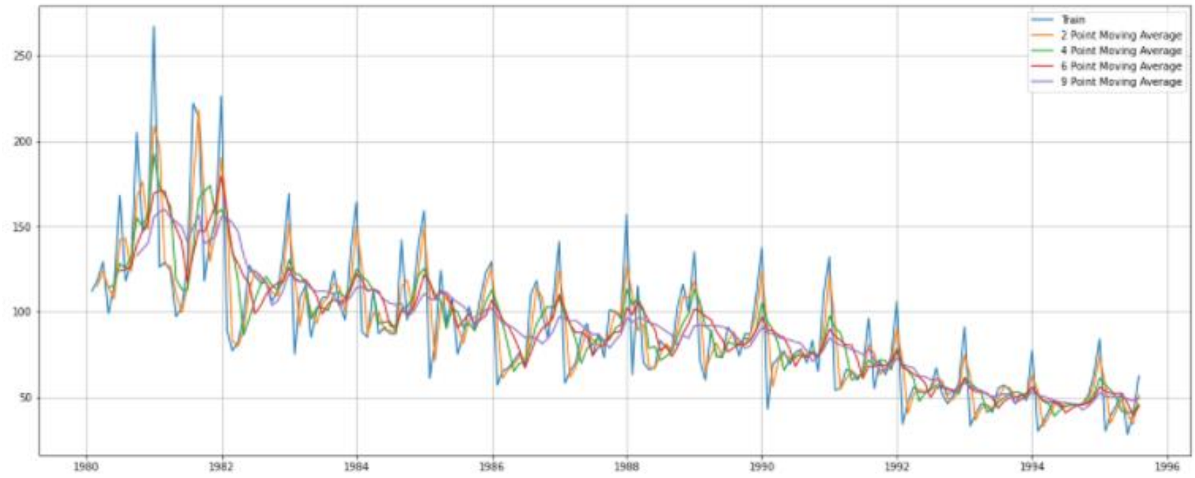


Fig.110

Let us 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.

Plotting Graph for Trailing MA:

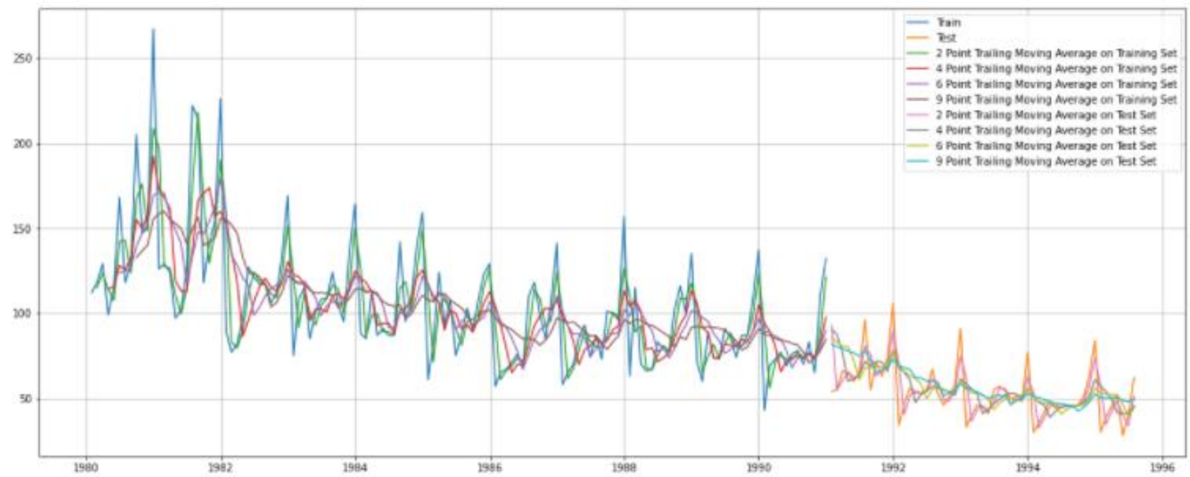


Fig.111

Model 4 Evaluation:

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

Creating Data frame:

	Train_RMSE	Test_RMSE
RegressionOnTime	30.718135	15.268955
Naive Model	45.063760	79.718773
Simple Average	36.034234	53.460570
2pointTrailingMovingAverage	NaN	11.529278
4pointTrailingMovingAverage	NaN	14.451403
6pointTrailingMovingAverage	NaN	14.566327
9pointTrailingMovingAverage	NaN	14.727630

Model 5: Simple Exponential Smoothing

```
{'smoothing_level': 0.995,
 'smoothing_trend': nan,
 'smoothing_seasonal': nan,
 'damping_trend': nan,
 'initial_level': 112.03027126088527,
 'initial_trend': nan,
 'initial_seasons': array([], dtype=float64),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

Fig.112

Predicting for training data:

Time_Stamp	Rose	predict
1980-01-31	112.0	112.030271
1980-02-29	118.0	112.000151
1980-03-31	129.0	117.970001
1980-04-30	99.0	128.944850
1980-05-31	116.0	99.149724

Fig.113

Predicting for Test data:

Time_Stamp	Rose	predict
1991-01-31	54.0	131.888877
1991-02-28	55.0	131.888877
1991-03-31	66.0	131.888877
1991-04-30	65.0	131.888877
1991-05-31	60.0	131.888877

Fig.114

Plotting Simple Exponential Smoothing forecast:

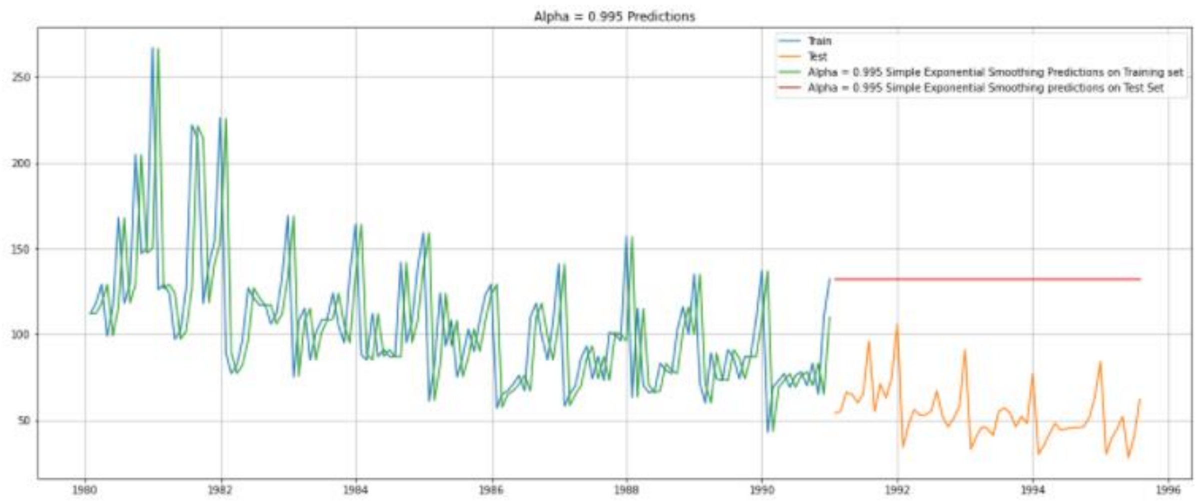


Fig.115

Train - RMSE score: For Alpha =0.995 Simple Exponential Smoothing Model forecast on the Train Data, RMSE is 38.715

Test - RMSE score: For Alpha =0.995 Simple Exponential Smoothing Model forecast on the Test Data, RMSE is 79.610

Creating DataFrame:

	Train_RMSE	Test_RMSE
RegressionOnTime	30.718135	15.268955
Naive Model	45.063760	79.718773
Simple Average	36.034234	53.460570
2pointTrailingMovingAverage	NaN	11.529278
4pointTrailingMovingAverage	NaN	14.451403
6pointTrailingMovingAverage	NaN	14.566327
9pointTrailingMovingAverage	NaN	14.727630
Alpha=0.995,SimpleExponentialSmoothing	38.714722	79.609847

Fig.116

Setting Different Alpha Values: We will run a loop with different alpha values to understand which particular value works best for alpha on the test set and Train Set.

Alpha_Values	Train_RMSE	Test_RMSE

Model Evaluation:

	Alpha_Values	Train_RMSE	Test_RMSE
0	0.3	32.470164	47.504821
1	0.4	33.035130	53.767406
2	0.5	33.682839	59.641786
3	0.6	34.441171	64.971288
4	0.7	35.323261	69.698162
5	0.8	36.334596	73.773992
6	0.9	37.482782	77.139276

Fig.117

Plotting graph for different Alpha values of Training and Test data:

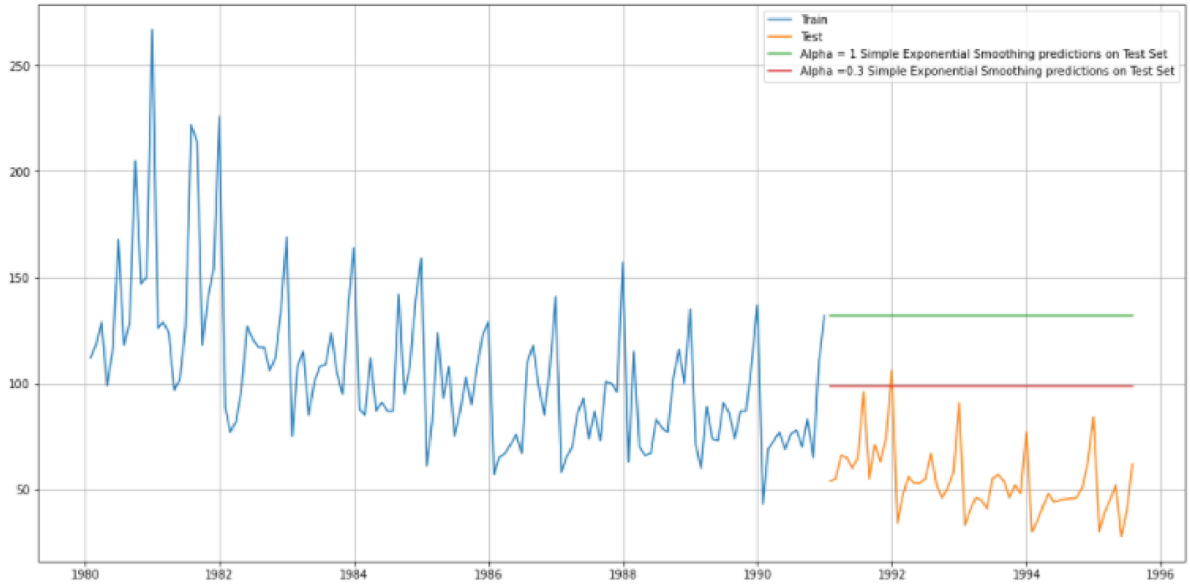


Fig.118

Creating Data frame:

	Train_RMSE	Test_RMSE
RegressionOnTime	30.718135	15.268955
Naive Model	45.063760	79.718773
Simple Average	36.034234	53.460570
2pointTrailingMovingAverage	NaN	11.529278
4pointTrailingMovingAverage	NaN	14.451403
6pointTrailingMovingAverage	NaN	14.566327
9pointTrailingMovingAverage	NaN	14.727630
Alpha=0.995,SimpleExponentialSmoothing	38.714722	79.609847
Alpha=0.3,SimpleExponentialSmoothing	32.470164	47.504821

Fig.119

Model 6: Double Exponential Smoothing

Two parameters α and β are estimated in this model. Level and Trend are accounted for in this model

First, we will define an empty data frame to store our values from the loop:

Alpha_Values	Beta_Values	Train_RMSE	Test_RMSE

	Alpha_Values	Beta_Values	Train_RMSE	Test_RMSE
0	0.3	0.3	35.944983	265.567594
8	0.4	0.3	36.749123	339.306534
1	0.3	0.4	37.393239	358.750942
16	0.5	0.3	37.433314	394.272629
24	0.6	0.3	38.348984	439.296033

Fig.120

Plotting Graph on Both Training and Test data:

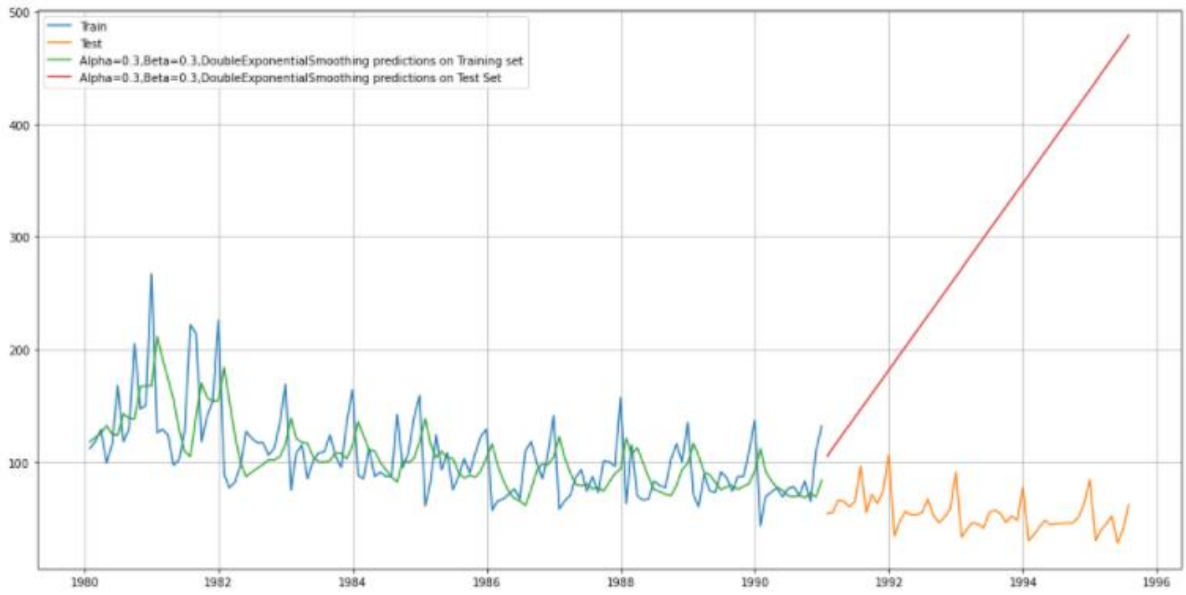


Fig.121

Creating DataFrame:

	Train_RMSE	Test_RMSE
RegressionOnTime	30.718135	15.268955
Naive Model	45.063760	79.718773
Simple Average	36.034234	53.460570
2pointTrailingMovingAverage	NaN	11.529278
4pointTrailingMovingAverage	NaN	14.451403
6pointTrailingMovingAverage	NaN	14.566327
9pointTrailingMovingAverage	NaN	14.727630
Alpha=0.995,SimpleExponentialSmoothing	38.714722	79.609847
Alpha=0.3,SimpleExponentialSmoothing	32.470164	47.504821
Alpha=0.3,Beta=0.3,DoubleExponentialSmoothing	35.944983	265.567594

Fig.122

Model 7: Triple Exponential Smoothing

Two parameters α and β are estimated in this model. Level and Trend are accounted for in this model

```
{'smoothing_level': 0.06571007449183297,
 'smoothing_trend': 0.051867105713176015,
 'smoothing_seasonal': 0.0015637515713898,
 'damping_trend': nan,
 'initial_level': 47.81887301367471,
 'initial_trend': -0.2961562797665537,
 'initial_seasons': array([2.35763018, 2.67367218, 2.92146068, 2.55308191, 2.87099548,
        3.13124987, 3.44178442, 3.66118656, 3.47154364, 3.39670325,
        3.95879831, 5.46173463]),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

Fig.123

Prediction on Train Set:

Time_Stamp	Rose	auto_predict
1980-01-31	112.0	112.040991
1980-02-29	118.0	126.265128
1980-03-31	129.0	136.477432
1980-04-30	99.0	118.033622
1980-05-31	116.0	130.346231

Fig.124

Prediction on Test set:

Time_Stamp	Rose	auto_predict
1991-01-31	54.0	56.677627
1991-02-28	55.0	64.136371
1991-03-31	66.0	69.860745
1991-04-30	65.0	60.897998
1991-05-31	60.0	68.228324

Fig.125

Plotting graph using Training set and Test set using autofit:

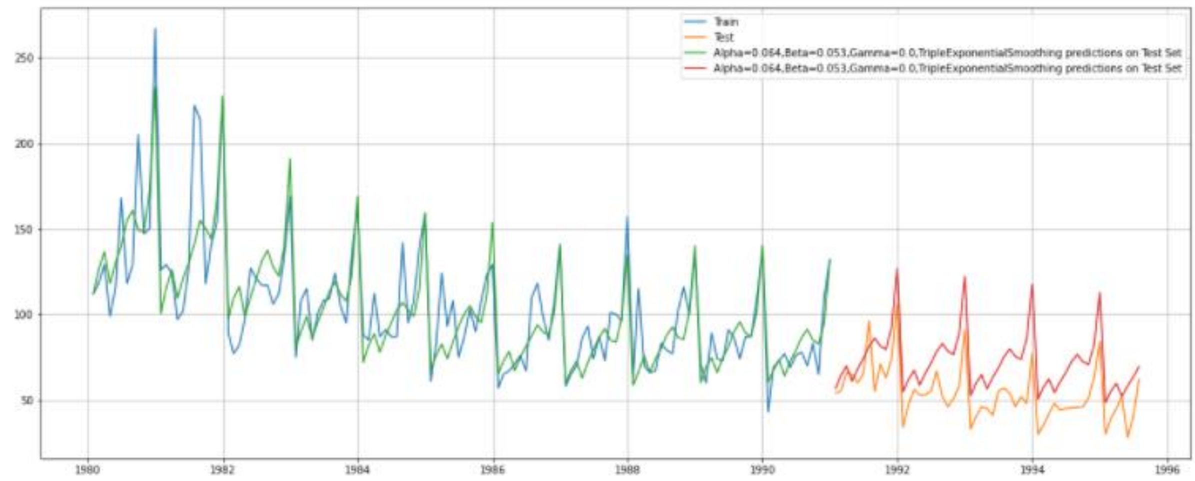


Fig.126

Train - RMSE score: For Alpha=0.064, Beta=0.053, Gamma=0.0, Triple Exponential Smoothing Model forecast on the Train Data, RMSE is 18.415

Test - RMSE score: For Alpha=0.064, Beta=0.053, Gamma=0.0, Triple Exponential Smoothing Model forecast on the Test Data, RMSE is 20.990

Creating DataFrame:

	Train_RMSE	Test_RMSE
RegressionOnTime	30.718135	15.268955
Naive Model	45.063760	79.718773
Simple Average	36.034234	53.460570
2pointTrailingMovingAverage	NaN	11.529278
4pointTrailingMovingAverage	NaN	14.451403
6pointTrailingMovingAverage	NaN	14.566327
9pointTrailingMovingAverage	NaN	14.727630
Alpha=0.995,SimpleExponentialSmoothing	38.714722	79.609847
Alpha=0.3,SimpleExponentialSmoothing	32.470164	47.504821
Alpha=0.3,Beta=0.3,DoubleExponentialSmoothing	35.944983	265.567594
Alpha=0.064,Beta=0.053,Gamma=0.0,TripleExponentialSmoothing	18.414568	20.990268

Fig.126

Defining an empty dataframe to store our values from the loop:

	Alpha_Values	Beta_Values	Gamma_Values	Train_RMSE	Test_RMSE
8	0.3	0.4	0.3	28.111886	10.945435
1	0.3	0.3	0.4	27.399095	11.201633
69	0.4	0.3	0.8	32.601491	12.615607
16	0.3	0.5	0.3	29.087520	14.414604
131	0.5	0.3	0.6	32.144773	16.720720

Plotting on both the Training and Test data using brute force alpha, beta:

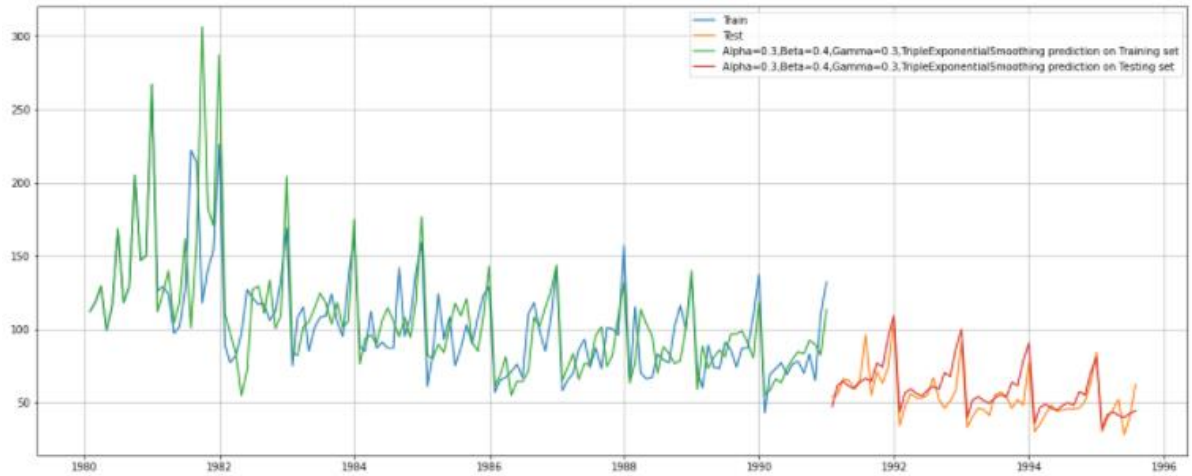


Fig.127

Creating Dataframe:

	Train_RMSE	Test_RMSE
RegressionOnTime	30.718135	15.268955
Naive Model	45.063760	79.718773
Simple Average	36.034234	53.460570
2pointTrailingMovingAverage	NaN	11.529278
4pointTrailingMovingAverage	NaN	14.451403
6pointTrailingMovingAverage	NaN	14.566327
9pointTrailingMovingAverage	NaN	14.727630
Alpha=0.995,SimpleExponentialSmoothing	38.714722	79.609847
Alpha=0.3,SimpleExponentialSmoothing	32.470164	47.504821
Alpha=0.3,Beta=0.3,DoubleExponentialSmoothing	35.944983	265.567594
Alpha=0.064,Beta=0.053,Gamma=0.0,TripleExponentialSmoothing	18.414568	20.990268
Alpha=0.3,Beta=0.4,Gamma=0.3,TripleExponentialSmoothing	28.111886	10.945435

Fig.128

```
Sorted by RMSE values on Test Data
                                                 Train_RMSE   Test_RMSE
Alpha=0.3,Beta=0.4,Gamma=0.3,TripleExponentialS...  28.111886   10.945435
2pointTrailingMovingAverage                               NaN   11.529278
4pointTrailingMovingAverage                               NaN   14.451403
6pointTrailingMovingAverage                               NaN   14.566327
9pointTrailingMovingAverage                               NaN   14.727630
RegressionOnTime                                  30.718135   15.268955
Alpha=0.064,Beta=0.053,Gamma=0.0,TripleExponent...  18.414568   20.990268
Alpha=0.3,SimpleExponentialSmoothing              32.470164   47.504821
Simple Average                                    36.034234   53.460570
Alpha=0.995,SimpleExponentialSmoothing            38.714722   79.609847
Naive Model                                       45.063760   79.718773
Alpha=0.3,Beta=0.3,DoubleExponentialSmoothing     35.944983  265.567594


Sorted by RMSE values on Train Data
                                                 Train_RMSE   Test_RMSE
Alpha=0.064,Beta=0.053,Gamma=0.0,TripleExponent...  18.414568   20.990268
Alpha=0.3,Beta=0.4,Gamma=0.3,TripleExponentialS...  28.111886   10.945435
RegressionOnTime                                  30.718135   15.268955
Alpha=0.3,SimpleExponentialSmoothing              32.470164   47.504821
Alpha=0.3,Beta=0.3,DoubleExponentialSmoothing     35.944983  265.567594
Simple Average                                    36.034234   53.460570
Alpha=0.995,SimpleExponentialSmoothing            38.714722   79.609847
Naive Model                                       45.063760   79.718773
2pointTrailingMovingAverage                               NaN   11.529278
4pointTrailingMovingAverage                               NaN   14.451403
6pointTrailingMovingAverage                               NaN   14.566327
9pointTrailingMovingAverage                               NaN   14.727630
```

Fig.129

Logistic Regression:

Steps:

- For this particular linear regression, we are going to regress the 'Rose' variable against the order of the occurrence. For this we need to modify our training data before fitting it into a linear regression.
- We have generated the numerical time instance order for both the training and test set. After that we add these values in the training and test set.
- Initiate LogisticRegression () classifier.
- Fit into train data set
- Predict the target variable for train and test data
- The predicted trend is downwards indicating further decrease in sales of Rose wine.

- Units sold in 1991 can be 60 which can fall down to 40 Units by year 1995.
- The RSME on test data value is 15.275, value is not very high but since seasonality is also not taken care by model this model is not suitable predictions on Rose time series data.

Naive 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.
- RSME is 79.738 which is higher than the linear regression model. Thus, this model being too simple, not taking care of seasonality with very high RSME.

Simple Average:

- For this particular simple average method, we will forecast by using the average of the training values.
- RMSE is 53.41 which is less than Naïve model but higher than regression model and without seasonality component.

Moving Average:

- 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.
- For Moving Average, we are going to average over the entire data.
- Considering criteria that testing data should start from 1991 onwards, training and testing data is prepared.
- On the testing data set having the orange color trend line the predicted values in green trend line fits the most which is for the 2-point trailing moving average.
- Comparison plot shows the best fit model in brown color line for 2nd moving average aptly fitting on the actual test values.

Exponential Smoothing:

- Exponential smoothing of time series data assigns exponentially decreasing weights for newest to oldest observations. That implies, that the older data get less priority (“weight”) and the newer data is more relevant and is assigned more weight.
- Exponential smoothing is usually used to make short term forecasts, as longer-term forecasts using this technique cannot be more reliable.
- Simple exponential smoothing uses a weighted moving average with exponentially decreasing weights. This method is suitable for forecasting data with no clear trend or seasonal pattern.

- Holt's linear method with additive errors or double exponential smoothing is applicable when data has Trend but no seasonality
- Triple exponential smoothing (also called the Multiplicative Holt-Winters) is applicable for the data that shows trends and seasonality.
- As our data shows both trend and seasonality, we will go for Holt-Winters exponential method.
- With the help of 'ExponentialSmoothing' function we performed Triple exponential smoothing on our data set.

Steps:

- Initializing the Double Exponential Smoothing Model
- Fitting the model
- Forecasting using this model for the duration of the test set

We got the smoothing parameter value:

- Looking at the RMSE values for all the models, we can conclude that the Triple Exponential Model with alpha, beta, gamma as 0.1,0.2 and 0.2 respectively performs the best. So we build a complete model on Rose dataset on these parameters.

1.5 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 series is said to be stationary if its mean and variance are constant over a period of time
- Check for stationarity of the whole Time Series data.
- We have performed Augmented Dickey-Fuller test to check stationarity.
- The Augmented Dickey-Fuller test is 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:
- * H0: The Time Series has a unit root and it is non-stationary.
- * H1: The Time Series does not have a unit root and it is stationary.
- To build ARIMA models we would want the Time series to be stationary and thus we would
- want the p-value of this test to be less than the alpha = 0.05.
- After performing Augmented Dickey-Fuller test, we get p-value = 0.343
- Hence, p-value is higher than alpha, we cannot reject the null hypothesis and can say that at
- 5% significant level the Time Series is non-stationary.


```
Results of Dickey-Fuller Test:
Test Statistic                  -1.876699
p-value                          0.343101
#Lags Used                      13.000000
Number of Observations Used    173.000000
Critical Value (1%)             -3.468726
dtype: float64

Test Statistic                  -1.876699
p-value                          0.343101
#Lags Used                      13.000000
Number of Observations Used    173.000000
Critical Value (1%)             -3.468726
Critical Value (5%)             -2.878396
dtype: float64

Test Statistic                  -1.876699
p-value                          0.343101
#Lags Used                      13.000000
Number of Observations Used    173.000000
Critical Value (1%)             -3.468726
Critical Value (5%)             -2.878396
Critical Value (10%)            -2.575756
dtype: float64
```

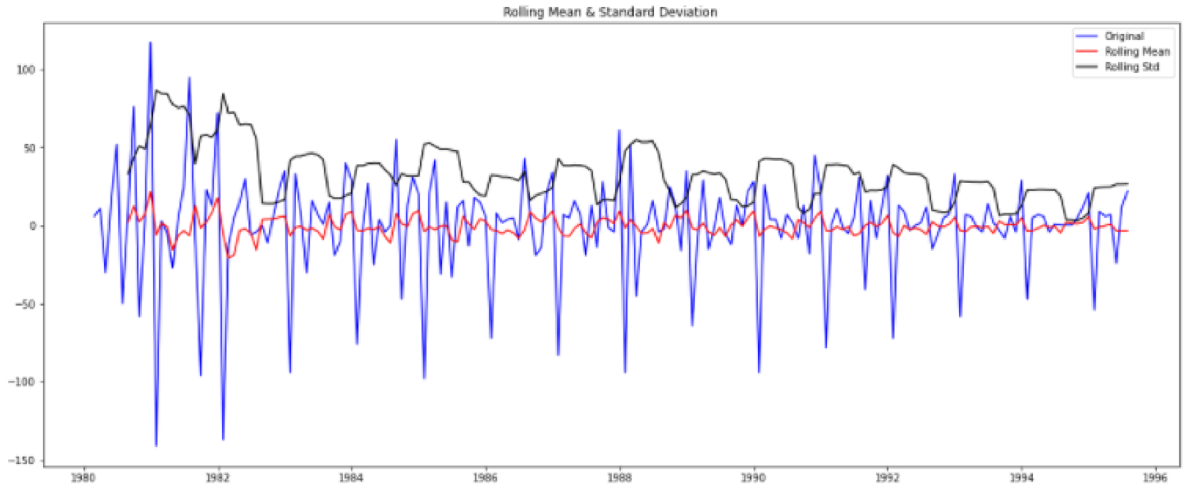
Fig.130

From above graph, it is clear that rolling mean and standard deviation is not constant at every data point. So that it is not stationary.

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

Make it Stationary:

- We have taken a difference of order 1 and check whether the Time Series is stationary or not. After performing ADF test on differenced of order 1 data we get p-value = 1.810895e-12
- Hence, p-value is lower than alpha, we can reject the null hypothesis and can say that at 5% significant level the Time Series is stationary




```
Results of Dickey-Fuller Test:
Test Statistic                -8.044392e+00
p-value                        1.810895e-12
#Lags Used                     1.200000e+01
Number of Observations Used    1.730000e+02
Critical Value (1%)           -3.468726e+00
dtype: float64

Test Statistic                -8.044392e+00
p-value                        1.810895e-12
#Lags Used                     1.200000e+01
Number of Observations Used    1.730000e+02
Critical Value (1%)           -3.468726e+00
Critical Value (5%)           -2.878396e+00
dtype: float64

Test Statistic                -8.044392e+00
p-value                        1.810895e-12
#Lags Used                     1.200000e+01
Number of Observations Used    1.730000e+02
Critical Value (1%)           -3.468726e+00
Critical Value (5%)           -2.878396e+00
Critical Value (10%)          -2.575756e+00
dtype: float64
```

Fig.131

We have performed ADF test on Train data also to check stationarity. We need to take difference of order 1 and to make the Time Series is stationary.

1.6 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.

- Here we performed both ARIMA and SARIMA.
- ARIMA(p,d,q) Model: ARIMA is defined by 3 parameters
- p: No of autoregressive terms
- d: No of differencing to stationaries the series
- q: No of moving average terms

- Here we have taken
- p value = 0 to 3,
- q value = 0 to 3 and
- d = 1, as we have taken difference of order 1 to make the series stationary.
- After that, to calculate AIC value we used combination of p,d,q and sort the result to get the lowest AIC

```
Some parameter combinations for the Model...
Model: (0, 1, 1)
Model: (0, 1, 2)
Model: (1, 1, 0)
Model: (1, 1, 1)
Model: (1, 1, 2)
Model: (2, 1, 0)
Model: (2, 1, 1)
Model: (2, 1, 2)
```

	param	AIC
2	(0, 1, 2)	1276.835373
5	(1, 1, 2)	1277.359225
4	(1, 1, 1)	1277.775757
7	(2, 1, 1)	1279.045689
8	(2, 1, 2)	1279.298694
1	(0, 1, 1)	1280.726183
6	(2, 1, 0)	1300.609261
3	(1, 1, 0)	1319.348311
0	(0, 1, 0)	1335.152658

Fig.132

```
                             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:                Thu, 28 Oct 2021   AIC                           1276.835
Time:                        18:25:20   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
-----------------------------------------------------------------------------
```

Fig.135

- We get the lowest Akaike Information Criteria (AIC) for (p,d,q) = (2,1,2)
- AIC for this order = 15.61800957004907

Steps:

- Create combination of parameters for the model using 'itertools'
- Calculate AIC value for each combination of(p,d,q) .
- Sort the AIC value and got the lowest AIC value and parameter combination.
- Initiate ARIMA() classifier.
- Fit into train data set
- Predict the sale for test data

Predict on the Test Set using this model and evaluate the model: 1417.502239430773

Prediction graph is not following the trend with test data.

	RMSE
ARIMA(0,1,2)	15.61801

Build an Automated version of a SARIMA model for which the best parameters are selected in accordance with the lowest Akaike Information Criteria (AIC):

	param	seasonal	AIC
26	(0, 1, 2)	(2, 0, 2, 12)	887.937509
53	(1, 1, 2)	(2, 0, 2, 12)	889.871767
80	(2, 1, 2)	(2, 0, 2, 12)	890.668799
69	(2, 1, 1)	(2, 0, 0, 12)	896.518161
78	(2, 1, 2)	(2, 0, 0, 12)	897.346444

Fig.133

```
                                      SARIMAX Results
==========================================================================================
Dep. Variable:                                  y   No. Observations:                  132
Model:             SARIMAX(0, 1, 2)x(2, 0, 2, 12)   Log Likelihood                -436.969
Date:                            Thu, 28 Oct 2021   AIC                            887.938
Time:                                    18:31:16   BIC                            906.448
Sample:                                         0   HQIC                           895.437
                                            - 132
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.8427    189.863     -0.004      0.996    -372.968     371.282
ma.L2         -0.1573     29.829     -0.005      0.996     -58.620      58.306
ar.S.L12       0.3467      0.079      4.375      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       251.3137   4.77e+04      0.005      0.996   -9.33e+04    9.38e+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.134

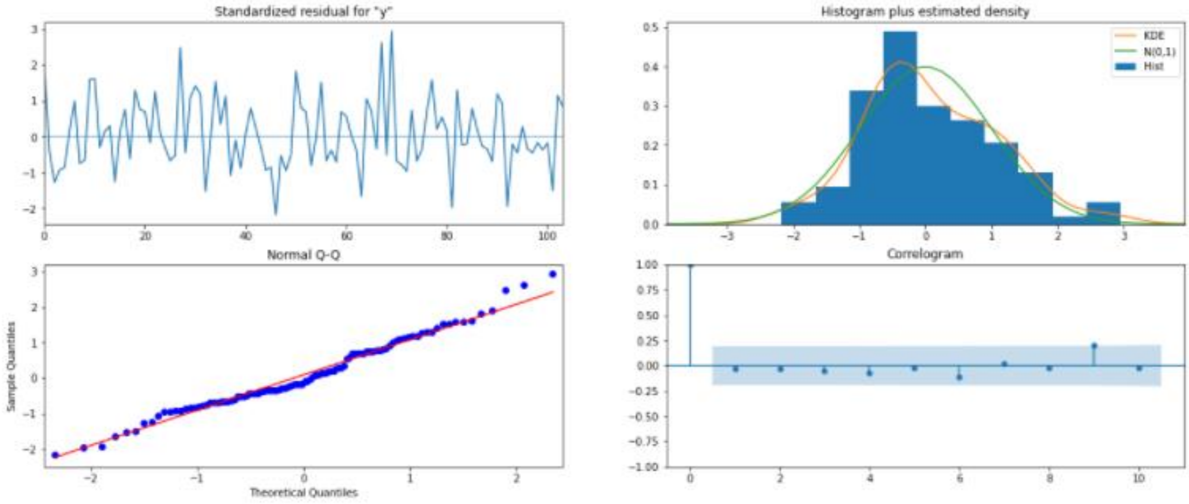


Fig.136

Predicting on the Test Set using this model and evaluate the model:

	RMSE
ARIMA(0,1,2)	15.618010
SARIMA(0,1,2)(2,0,2,12)	26.928362

- Seasonal ARIMA model with seasonal frequency.
- These models are used when time series data has significant seasonality.
- The most general form of seasonal ARIMA is $ARIMA$(p,d,q)$*$$ARIMA$(p,d,q)[m], where p,d,q are defined as seasonal AR component, seasonal difference and seasonal MA component respectively. And, ‘m’ represents the frequency (time interval) at which the data is observed.
- To calculate AIC value, we have taken combination of (p,d,q) and [P,D,Q]. to make this combination we used ‘itertools’.
- Here our m = 12

Steps:

- Create combination of parameters for the model using ‘itertools’
- Calculate AIC value for each combination of(p,d,q) and [P, D,Q,m].
- Sort the AIC value and got the lowest AIC value and parameter combination.
- Initiate SARIMAX classifier.
- Fit into train data set
- Predict the sale for test data
- Since the best performance is at (0,1,2) we build a model on that.

- Now, we predict accuracy on the test set by RMSE- RMSE= 15.618.
- Looking at this we can see that the seasonality is repeated 6 and 12 months. So we build
- SARIMA models for 6 and 12 months.
- Based on the AIC values, we build model on parameter- (1,1,2) seasonal- (2,0,2,12).
- The accuracy is given by RMSE, RMSE= 26.92

1.7 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.

We have plotted ACF and PACF for differenced Train data.

ACF:

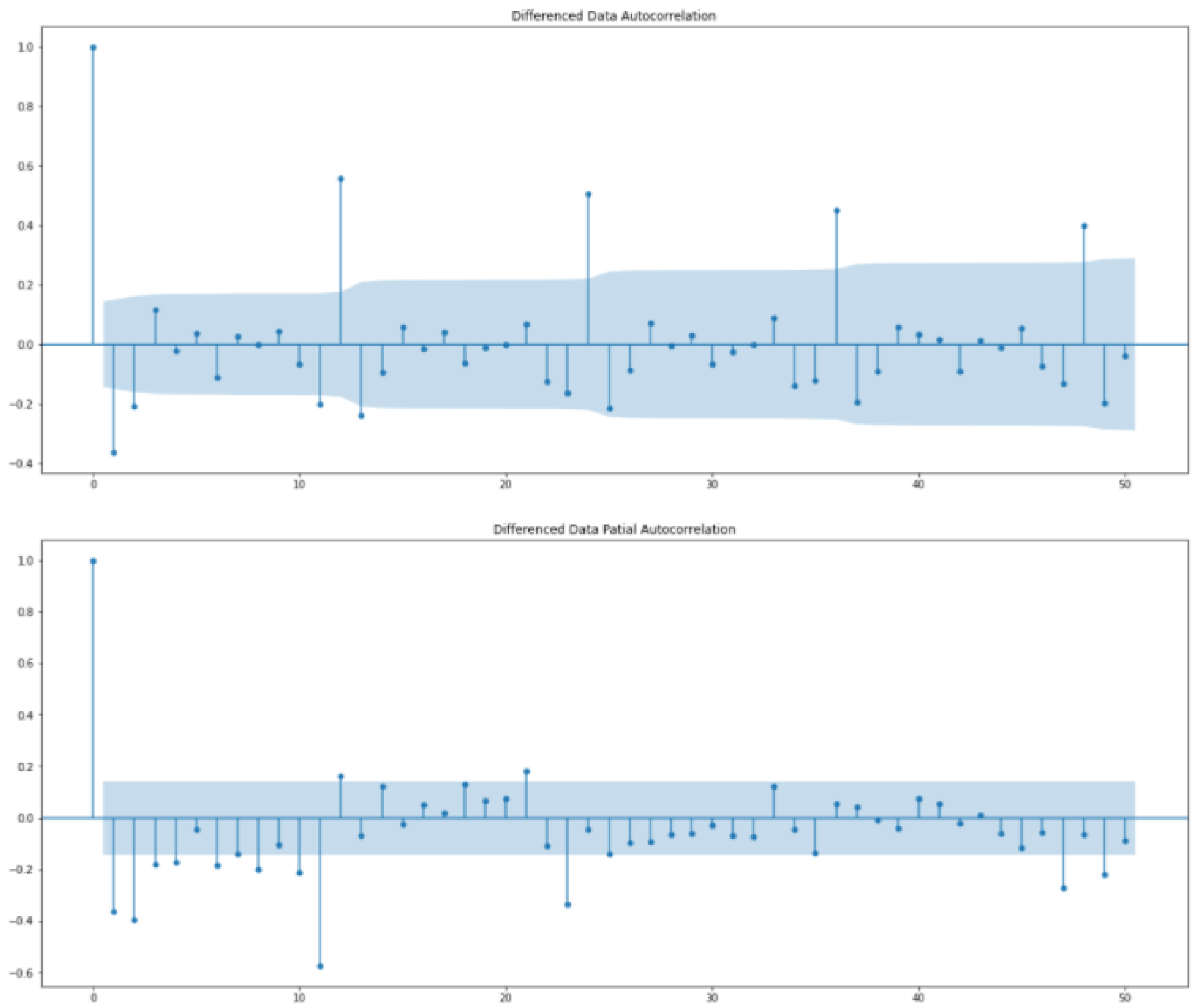


Fig.137

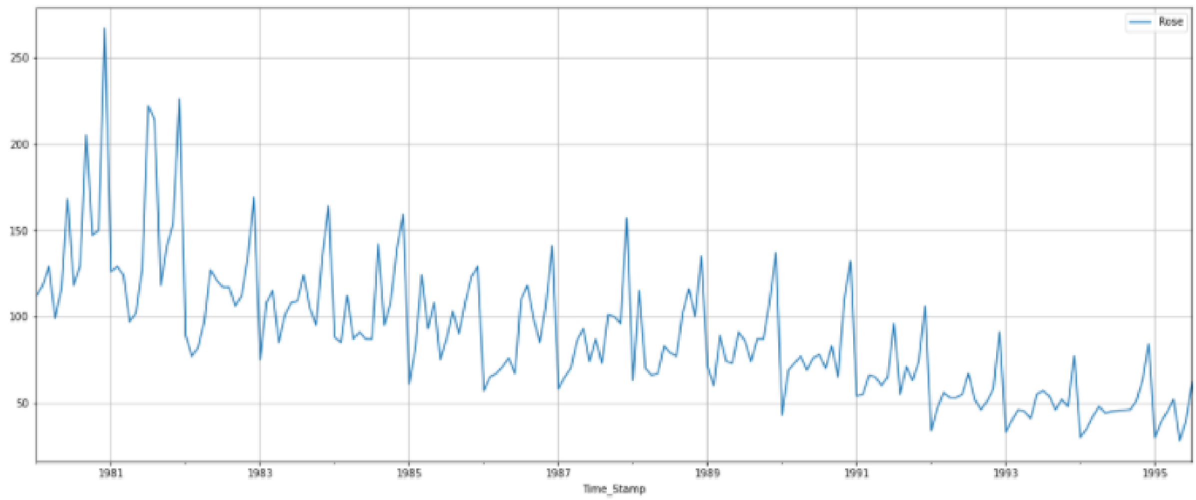


Fig.138

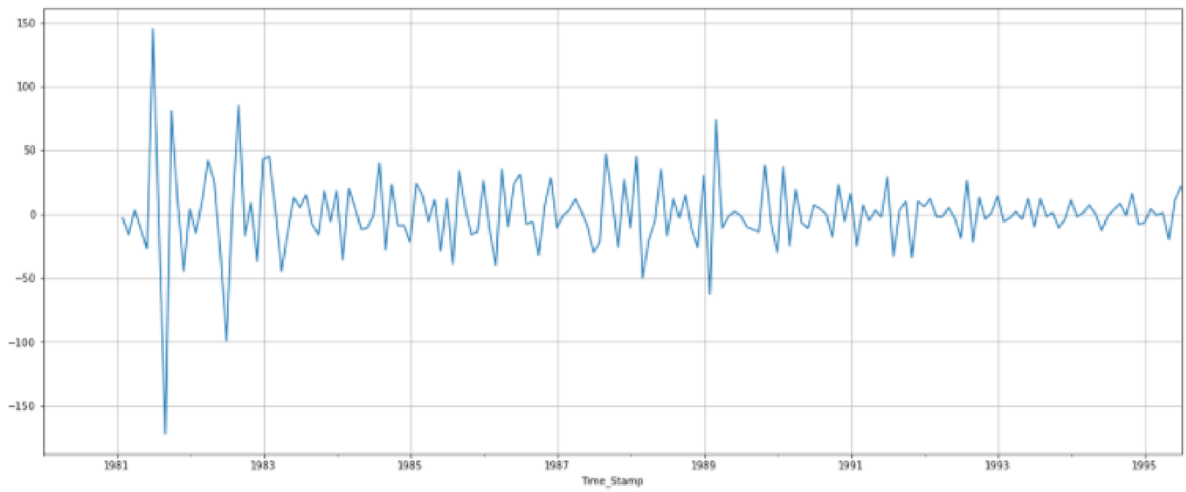
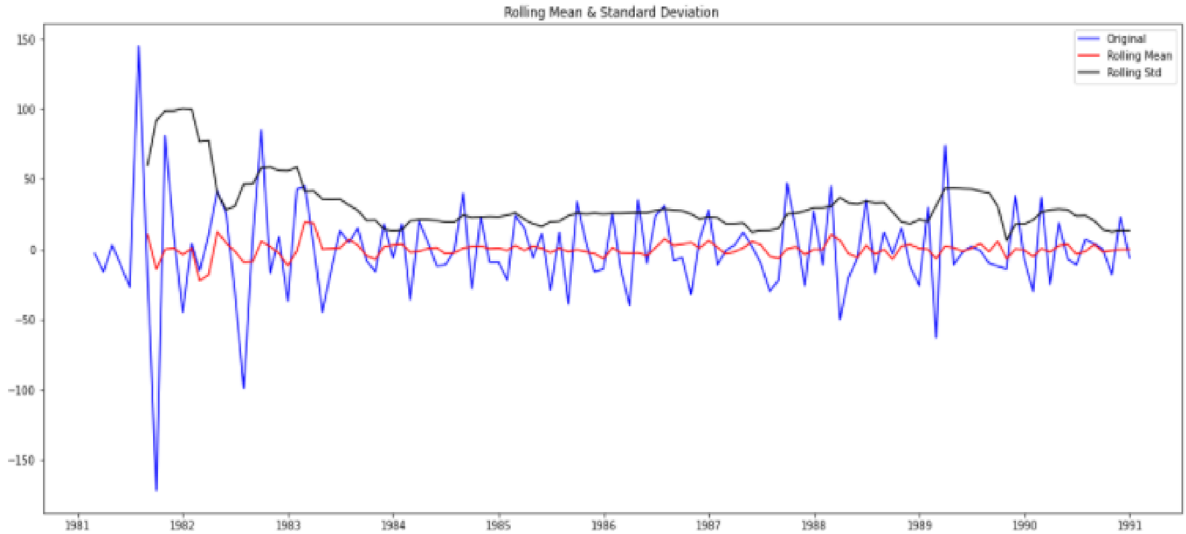


Fig.139

checking the stationarity of the above series before fitting the SARIMA model:




```
Results of Dickey-Fuller Test:
Test Statistic                  -3.692348
p-value                          0.004222
#Lags Used                      11.000000
Number of Observations Used    107.000000
Critical Value (1%)             -3.492996
dtype: float64

Test Statistic                  -3.692348
p-value                          0.004222
#Lags Used                      11.000000
Number of Observations Used    107.000000
Critical Value (1%)             -3.492996
Critical Value (5%)             -2.888955
dtype: float64

Test Statistic                  -3.692348
p-value                          0.004222
#Lags Used                      11.000000
Number of Observations Used    107.000000
Critical Value (1%)             -3.492996
Critical Value (5%)             -2.888955
Critical Value (10%)            -2.581393
dtype: float64
```

Fig.140

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

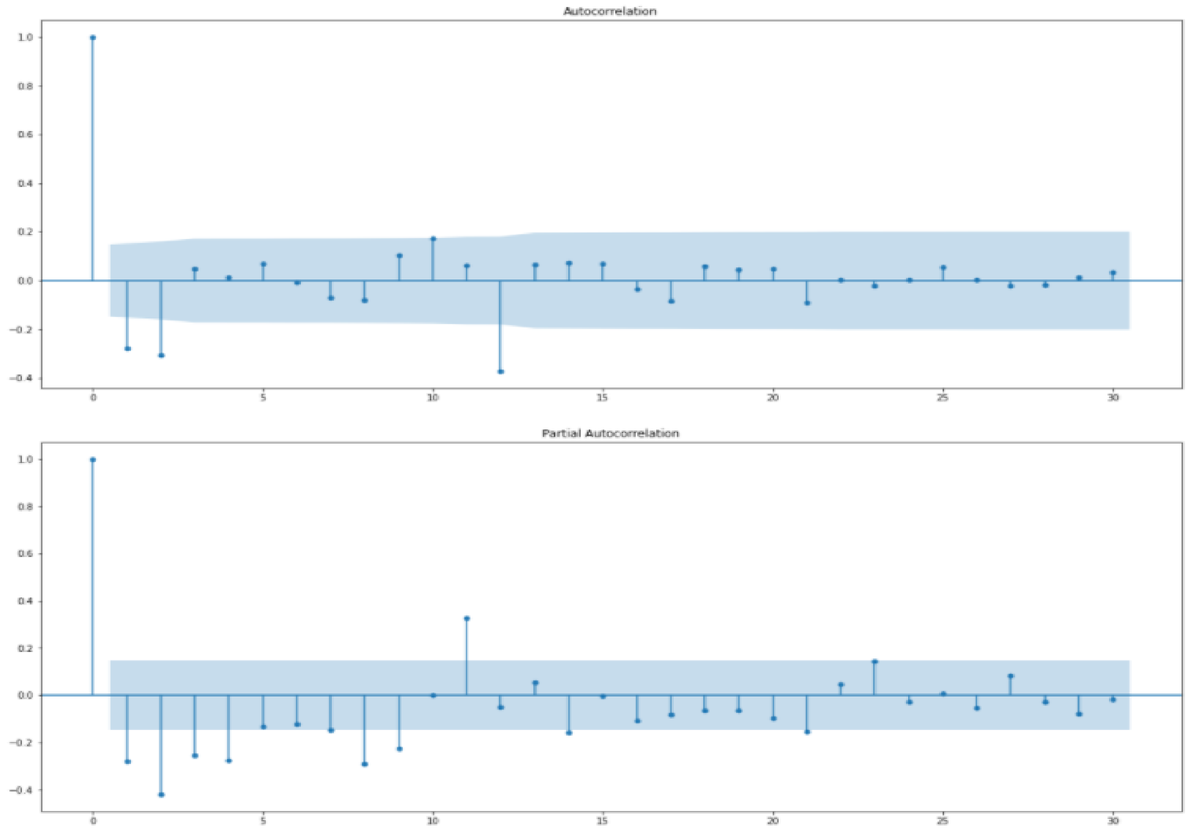


Fig.141

We are going to take the seasonal period as 12:

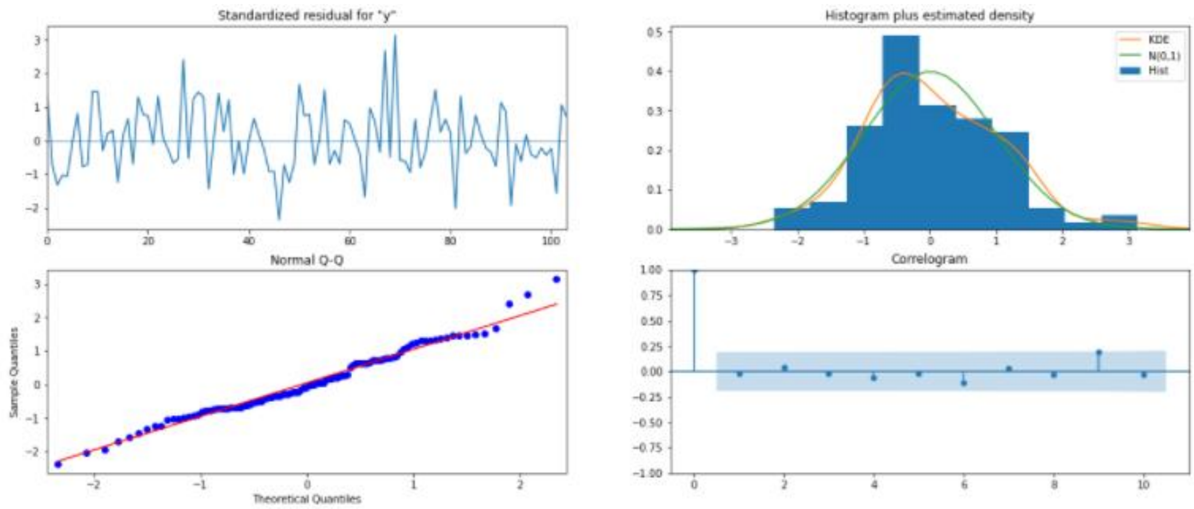


Fig.142

Predicting on the Test Set using this model and evaluate the model:

y	mean	mean_se	mean_ci_lower	mean_ci_upper
0	62.232440	15.796450	31.271968	93.192913
1	69.150517	15.979715	37.830851	100.470182
2	76.828021	16.016818	45.435636	108.220407
3	76.627751	16.029605	45.210302	108.045201
4	73.150494	16.028896	41.734436	104.566552

Fig.143

RMSE Score: 27.46343

	RMSE
ARIMA(0,1,2)	15.618010
SARIMA(0,1,2)(2,0,2,12)	26.928362
SARIMA(2,1,2)(2,0,2,12)	27.463437

Fig.144

Manual ARIMA model:

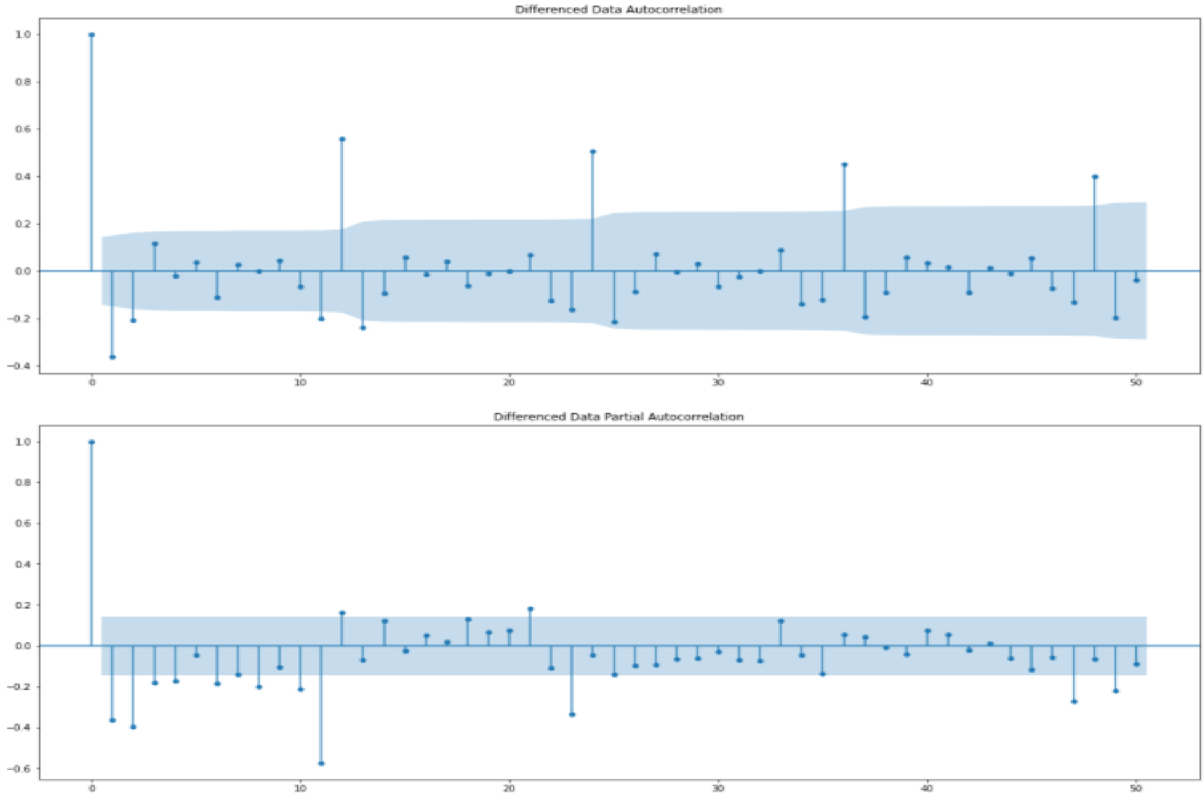


Fig.145

Predicting on the Test Set using this model and evaluate the model:

RMSE Score: 15.354883102386067

	RMSE
ARIMA(0,1,2)	15.618010
SARIMA(0,1,2)(2,0,2,12)	26.928362
SARIMA(2,1,2)(2,0,2,12)	27.463437
ARIMA(2,1,2)	15.354883

Fig.146

- Looking at the ACF and PACF plots of the differenced series we see our first significant value at lag 4 for ACF and at the same lag 4 for the PACF which suggest to use p = 4 and q = 4.
- We also have a big value at lag 12 in the ACF plot which suggests our season is S = 12 and since this lag is positive it suggests P = 1 and Q = 0.
- Since this is a differenced series for SARIMA we set d = 1, and since the seasonal pattern is not stable over time, we set D = 0.
- All together this gives us a SARIMA (4,1,4) (1,0,0) [12] model. Next, we run SARIMA with these values to fit a model on our training data.
- By looking at the above pictures we can say the two tests has been done JB and Ljung BOX test

JB test distribution normal for null hypothesis:

- P value is below 0.05 hence it is not normal distribution
- Ljung box test errors and Residual are independent of each other since we fail to reject the null hypothesis
- Heteroskedasticity means the residuals do have relation with independent variables. In this case less than 0.05 hence it is heteroskedastic
- By looking at the model diagrams we can say that there is no seasonality
- KDE plot of the residuals are not normally distributed.
- Residuals are distributed following a linear trend of the samples taken from standard normal
- Distribution in Normal Q-Q plot.
- Time series residuals have low correlation with lagged version by itself.

1.11 Build a table (create a data frame) with all the models built along with their corresponding parameters and the respective RMSE values on the test data.

We have created a data frame with all the models built along with their corresponding parameters and the respective RMSE values on the test data.

	Test_RMSE
Alpha=0.3,Beta=0.4,Gamma=0.3,TripleExponentialSmoothing	10.945435
2pointTrailingMovingAverage	11.529278
4pointTrailingMovingAverage	14.451403
6pointTrailingMovingAverage	14.566327
9pointTrailingMovingAverage	14.727630
RegressionOnTime	15.268955
ARIMA(2,1,2)	15.354883
ARIMA(0,1,2)	15.618010
Alpha=0.064,Beta=0.053,Gamma=0.0,TripleExponentialSmoothing	20.990268
SARIMA(0,1,2)(2,0,2,12)	26.928362
SARIMA(2,1,2)(2,0,2,12)	27.463437
Alpha=0.3,SimpleExponentialSmoothing	47.504821
Simple Average	53.460570
Alpha=0.995,SimpleExponentialSmoothing	79.609847
Naive Model	79.718773
Alpha=0.3,Beta=0.3,DoubleExponentialSmoothing	265.567594

Fig.147

1.12 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.

```
{'smoothing_level': 0.07543819037108564,
 'smoothing_trend': 0.07543816249429056,
 'smoothing_seasonal': 0.27294445154459185,
 'damping_trend': nan,
 'initial_level': 1577.446620339965,
 'initial_trend': -13.373823757745805,
 'initial_seasons': array([1.06875929, 1.02018982, 1.47799192, 1.19853279,
0.98227378,
        0.9560861 , 1.32099751, 1.70927388, 1.39014226, 1.89705741,
        2.90535421, 3.77843847]),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

Fig.148

```
1995-08-31    1929.565002
1995-09-30    2347.047048
1995-10-31    3172.507937
1995-11-30    3909.340833
1995-12-31    5969.868543
1996-01-31    1353.953302
1996-02-29    1593.630283
1996-03-31    1824.817614
1996-04-30    1784.274585
1996-05-31    1635.069228
1996-06-30    1549.085128
1996-07-31    1955.711909
Freq: M, dtype: float64
```

Fig.149

Plotting The Forecast with Confidence Bond:

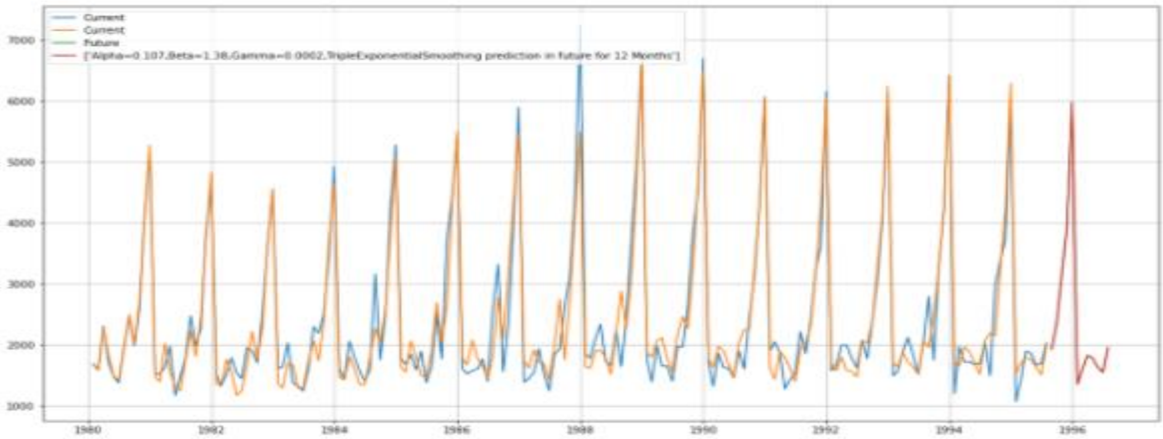


Fig.150

Final Result: For Alpha=0.107, Beta=1.38, Gamma=0.0002, Triple exponential Smoothing model Forecasting 16.09617002398434

Predicted Model for the Sparkling Sales for final Model:

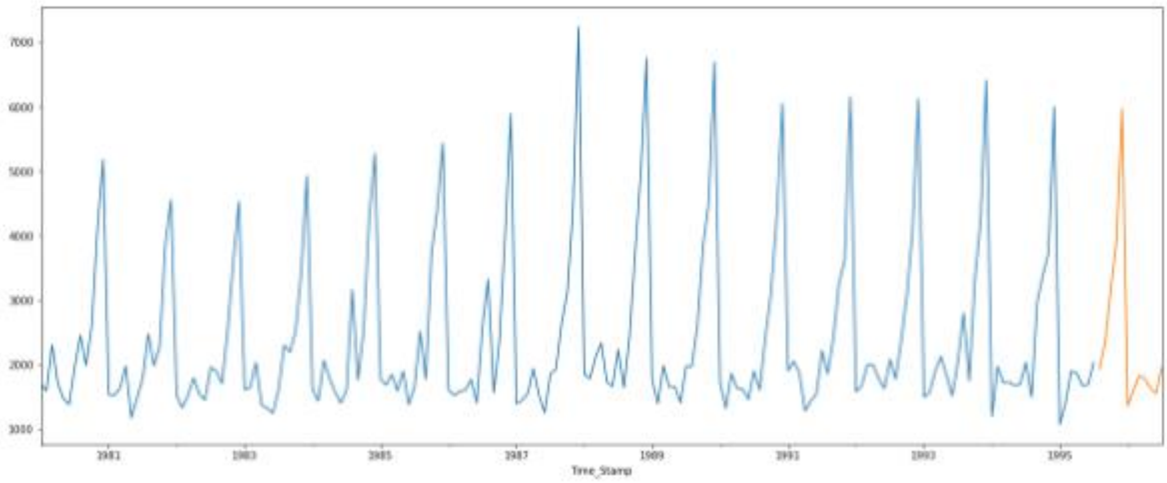


Fig.151

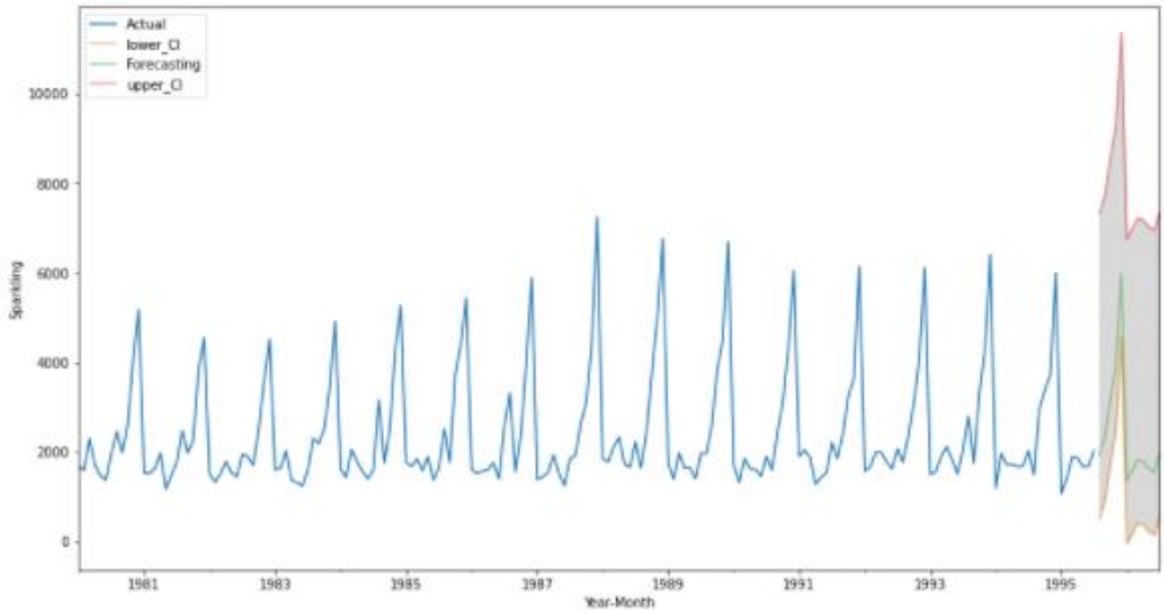


Fig.152

Prediction graph is following the previous trend for next 12 month.

- TES model alpha: 0.1, beta: 0.1 and gamma: 0.2 with trend as additive and seasonal as multiplicative is found to be the best model in terms of accuracy scored against the full data.
- Based on the overall performance the TES and SARIMA are selected for the evaluation on the full data.
- The SARIMA model is built with parameters (3, 1, 1) x (1, 0, 1, 12), is the most optimal SARIMA model.
- SARIMA model has reflected the trend and seasonality of the series continuing into the future year as well.
- SARIMA model shows better fitment and shows high variations in the farthest periods of observations.
- The model predicts continuation of the trend in sales and seasonality in year-end sales. The prediction shows a stabilization of downward trend.

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

From the above models and conclusions, we can suggest the following business insights and recommendations:

- Based upon the Test RMSE scores, the Alpha=0.3, Beta=0.3, Gamma=0.4, TripleExponentialSmoothing model is most suitable to consider as it is having least RMSE score.
- Time series analysis involves understanding various aspects about the inherent nature of the series so that you are better informed to create meaningful and accurate forecasts.

Inference:

- Rose wine sales shown a decrease in trend on year-on-year basis.
- December month has the highest sales in a year for Rose wine as well.
- Model plot was build based on the trend and seasonality. We see the future prediction is in line with the previous year predictions.

Recommendations:

- The company should encourage sales in the summer and other seasons by putting some discounts and other offers which encourages more customers to buy mainly during the first 3 months and July-August-September.
- Rose wine sales are seasonal.
- We are able to see the Rose wines are sold highly during March/August/October till December.

- Company should plan a head and keep enough stock for March/August/October till December to capitalize on the demand.
- In order to increase the sales company should plan some promotional offers during the low sale period.