TIME SERIES PROJECT

ASHWIN KUMAR AG
PGP DSBA

TABLE OF CONTENTS

PROBLEM:

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

```
rose = pd.read_csv('Rose.csv', header = 0, index_col = 0, parse_dates = True, squeeze = True)
```

```
rose.head()
```

```
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
```

```
rose.tail()
```

```
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
```

- *All values are loaded for the dataset with the index as pandas datetime format.*
- *The 'Rose' Time series has values in float64 datatype format.*

```
Sparkling = pd.read_csv('Sparkling.csv', header = 0, index_col = 0, parse_dates = True, squeeze = True)
```

```
Sparkling.head()
```

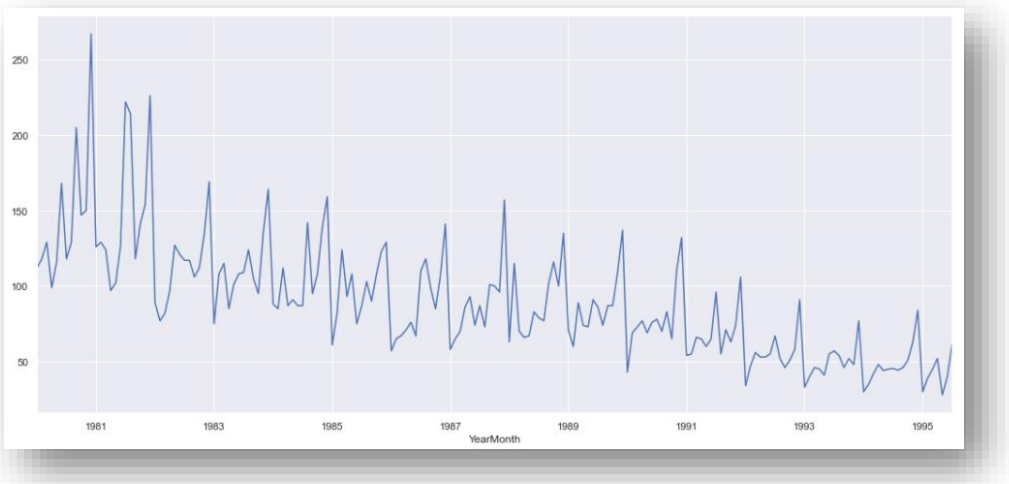
```
YearMonth
1980-01-01    1686
1980-02-01    1591
1980-03-01    2304
1980-04-01    1712
1980-05-01    1471
Name: Sparkling, dtype: int64
```

```
Sparkling.tail()
```

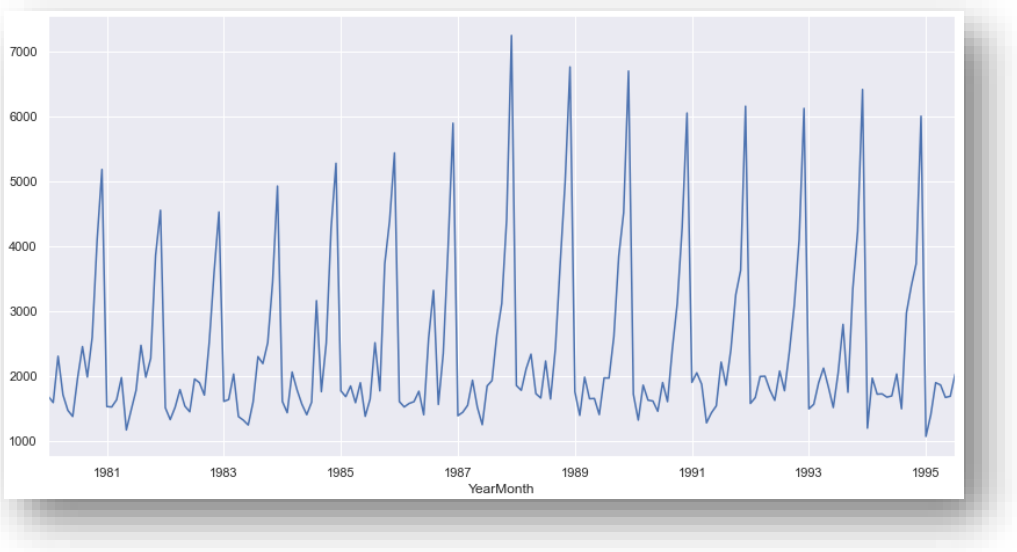
```
YearMonth
1995-03-01    1897
1995-04-01    1862
1995-05-01    1670
1995-06-01    1688
1995-07-01    2031
Name: Sparkling, dtype: int64
```

- *All values are loaded for the dataset with the index as pandas datetime format.*
- *The 'Rose' Time series has values in int64 datatype format.*

Time Series Plot of Rose Wines Data:



Time Series Plot of Sparkling Wines Data:



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

```
rose.describe()
```

```
count    187.000000
mean      89.907184
std       39.246679
min       28.000000
25%       62.500000
50%       85.000000
75%      111.000000
max      267.000000
Name: Rose, dtype: float64
```

```
Sparkling.describe()
```

```
count     187.000000
mean     2402.417112
std      1295.111540
min      1070.000000
25%      1605.000000
50%      1874.000000
75%      2549.000000
max      7242.000000
Name: Sparkling, dtype: float64
```

Null Values on Rose Data : 2

Null Values on Sparkling Data : 0

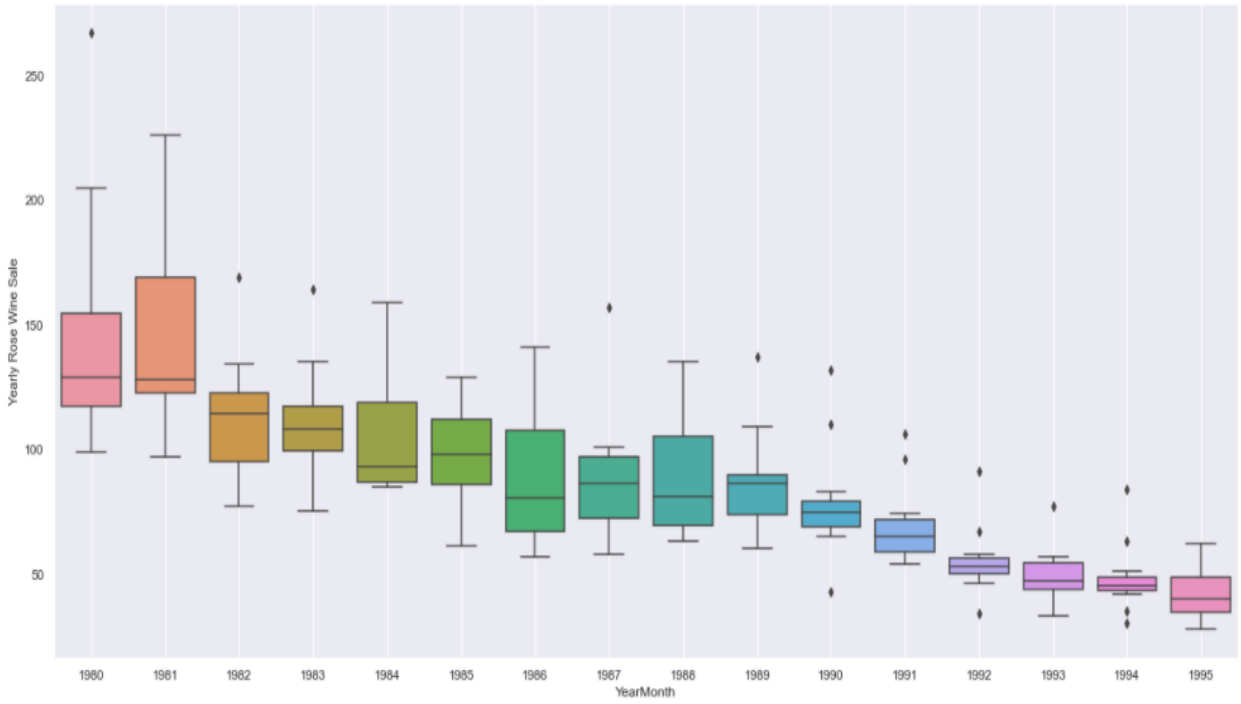
Null Values in Rose Data is replaced using "Polynomial Method" :

```
rose = rose.interpolate(method = 'polynomial', order = 2)
rose[rose.index.isin(nan_list) == True]
```

```
YearMonth
1994-07-01    45.364189
1994-08-01    44.279246
Name: Rose, dtype: float64
```

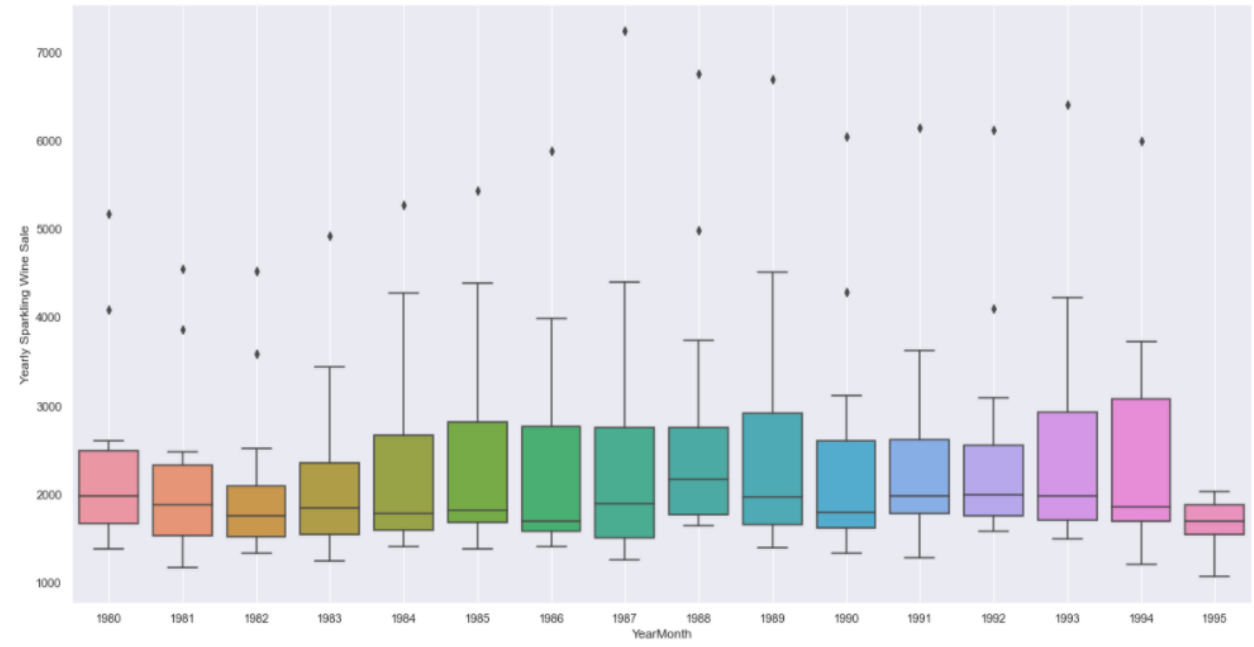
- *The missing values are imputed using polynomial interpolation of order 2. The new values for the respective index is obtained as shown above.*

Sales Across different year on Rose Wines:



The above plot shows there is a yearly decreasing trend present.

Sales Across different year on Sparkling Wines:

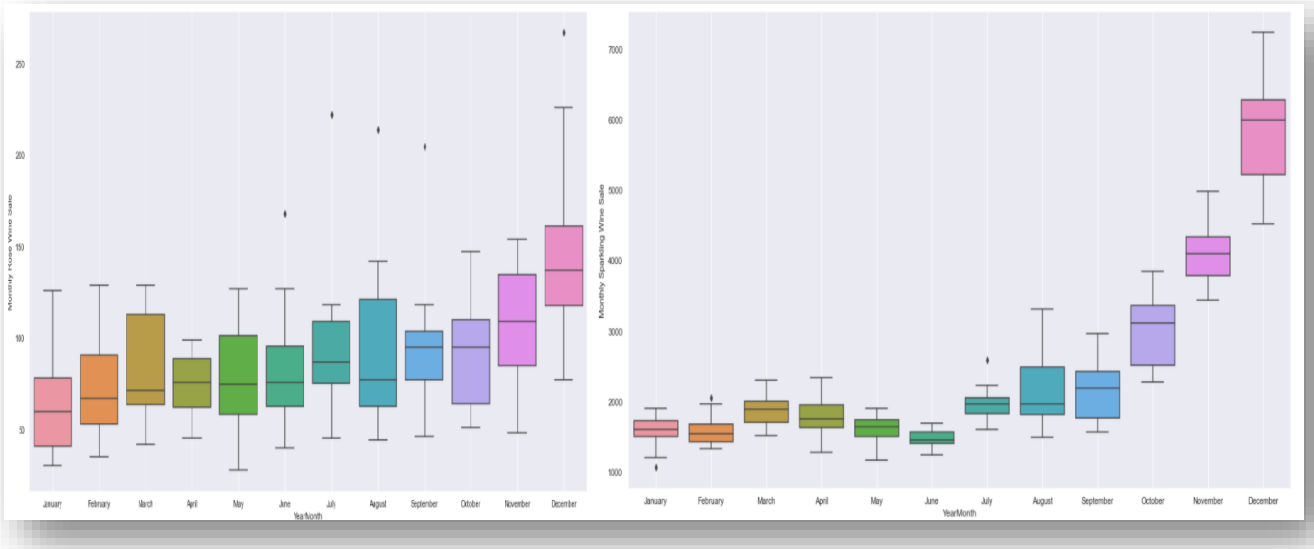


Pivot Table Data for Monthly Sales across different years :

Rose												
YearMonth	1	2	3	4	5	6	7	8	9	10	11	12
YearMonth												
1980	112.0	118.0	129.0	99.0	116.0	168.0	118.000000	129.000000	205.0	147.0	150.0	267.0
1981	126.0	129.0	124.0	97.0	102.0	127.0	222.000000	214.000000	118.0	141.0	154.0	226.0
1982	89.0	77.0	82.0	97.0	127.0	121.0	117.000000	117.000000	106.0	112.0	134.0	169.0
1983	75.0	108.0	115.0	85.0	101.0	108.0	109.000000	124.000000	105.0	95.0	135.0	164.0
1984	88.0	85.0	112.0	87.0	91.0	87.0	87.000000	142.000000	95.0	108.0	139.0	159.0
1985	61.0	82.0	124.0	93.0	108.0	75.0	87.000000	103.000000	90.0	108.0	123.0	129.0
1986	57.0	65.0	67.0	71.0	76.0	67.0	110.000000	118.000000	99.0	85.0	107.0	141.0
1987	58.0	65.0	70.0	86.0	93.0	74.0	87.000000	73.000000	101.0	100.0	96.0	157.0
1988	63.0	115.0	70.0	66.0	67.0	83.0	79.000000	77.000000	102.0	116.0	100.0	135.0
1989	71.0	60.0	89.0	74.0	73.0	91.0	86.000000	74.000000	87.0	87.0	109.0	137.0
1990	43.0	69.0	73.0	77.0	69.0	76.0	78.000000	70.000000	83.0	65.0	110.0	132.0
1991	54.0	55.0	66.0	65.0	60.0	65.0	96.000000	55.000000	71.0	63.0	74.0	106.0
1992	34.0	47.0	56.0	53.0	53.0	55.0	67.000000	52.000000	46.0	51.0	58.0	91.0
1993	33.0	40.0	46.0	45.0	41.0	55.0	57.000000	54.000000	46.0	52.0	48.0	77.0
1994	30.0	35.0	42.0	48.0	44.0	45.0	45.364189	44.279246	46.0	51.0	63.0	84.0
1995	30.0	39.0	45.0	52.0	28.0	40.0	62.000000	NaN	NaN	NaN	NaN	NaN

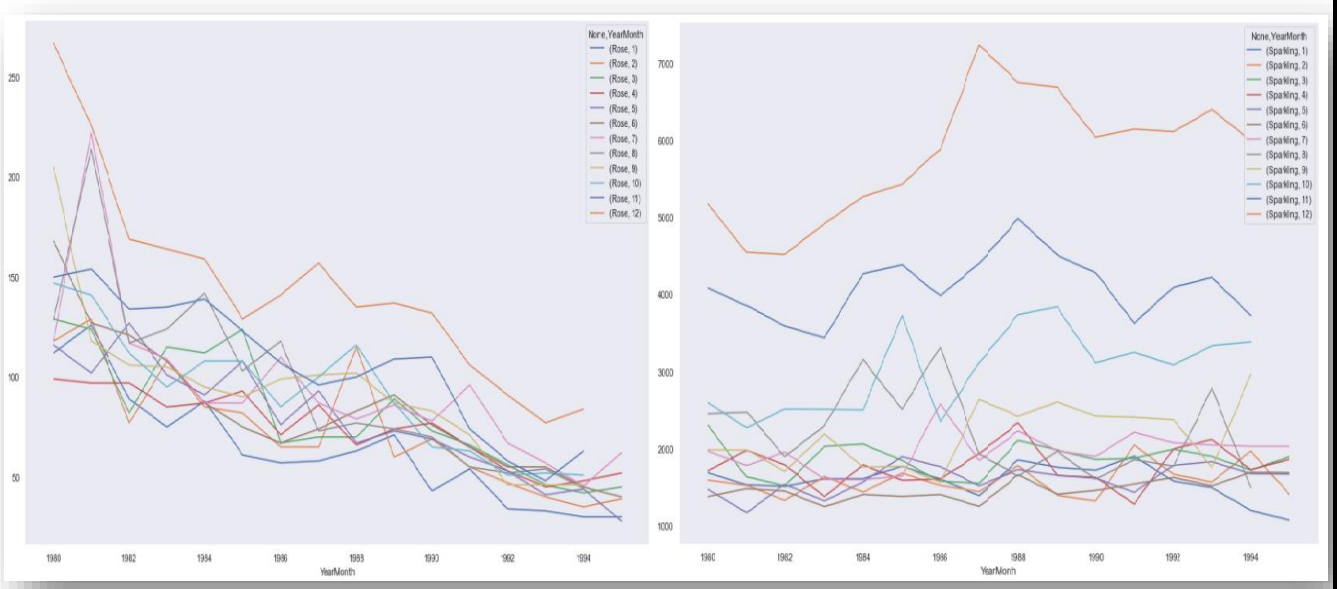
Sparkling												
YearMonth	1	2	3	4	5	6	7	8	9	10	11	12
YearMonth												
1980	1686.0	1591.0	2304.0	1712.0	1471.0	1377.0	1966.0	2453.0	1984.0	2596.0	4087.0	5179.0
1981	1530.0	1523.0	1633.0	1976.0	1170.0	1480.0	1781.0	2472.0	1981.0	2273.0	3857.0	4551.0
1982	1510.0	1329.0	1518.0	1790.0	1537.0	1449.0	1954.0	1897.0	1706.0	2514.0	3593.0	4524.0
1983	1609.0	1638.0	2030.0	1375.0	1320.0	1245.0	1600.0	2298.0	2191.0	2511.0	3440.0	4923.0
1984	1609.0	1435.0	2061.0	1789.0	1567.0	1404.0	1597.0	3159.0	1759.0	2504.0	4273.0	5274.0
1985	1771.0	1682.0	1846.0	1589.0	1896.0	1379.0	1645.0	2512.0	1771.0	3727.0	4388.0	5434.0
1986	1606.0	1523.0	1577.0	1605.0	1765.0	1403.0	2584.0	3318.0	1562.0	2349.0	3987.0	5891.0
1987	1389.0	1442.0	1548.0	1935.0	1518.0	1250.0	1847.0	1930.0	2638.0	3114.0	4405.0	7242.0
1988	1853.0	1779.0	2108.0	2336.0	1728.0	1661.0	2230.0	1645.0	2421.0	3740.0	4988.0	6757.0
1989	1757.0	1394.0	1982.0	1650.0	1654.0	1406.0	1971.0	1968.0	2608.0	3845.0	4514.0	6694.0
1990	1720.0	1321.0	1859.0	1628.0	1615.0	1457.0	1899.0	1605.0	2424.0	3116.0	4286.0	6047.0
1991	1902.0	2049.0	1874.0	1279.0	1432.0	1540.0	2214.0	1857.0	2408.0	3252.0	3627.0	6153.0
1992	1577.0	1667.0	1993.0	1997.0	1783.0	1625.0	2076.0	1773.0	2377.0	3088.0	4096.0	6119.0
1993	1494.0	1564.0	1898.0	2121.0	1831.0	1515.0	2048.0	2795.0	1749.0	3339.0	4227.0	6410.0
1994	1197.0	1968.0	1720.0	1725.0	1674.0	1693.0	2031.0	1495.0	2968.0	3385.0	3729.0	5999.0
1995	1070.0	1402.0	1897.0	1862.0	1670.0	1688.0	2031.0	NaN	NaN	NaN	NaN	NaN

Box Plot Visualization for Sales Across Different Months for Rose & Sparkling Wines :



- *The monthly plot shows that certain months have higher values than others indicating presence of seasonality.*

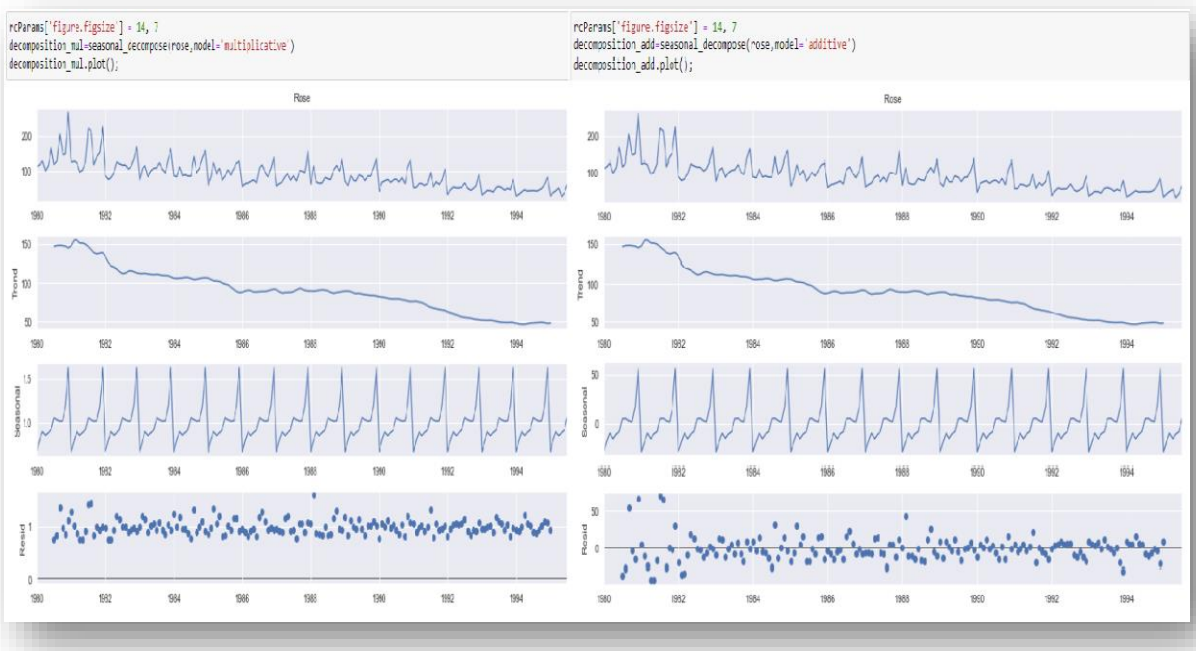
Yearly Sales Across Months: Graph Visualization



Rose Wines: A decreasing trend could be observed for the different months along the years.

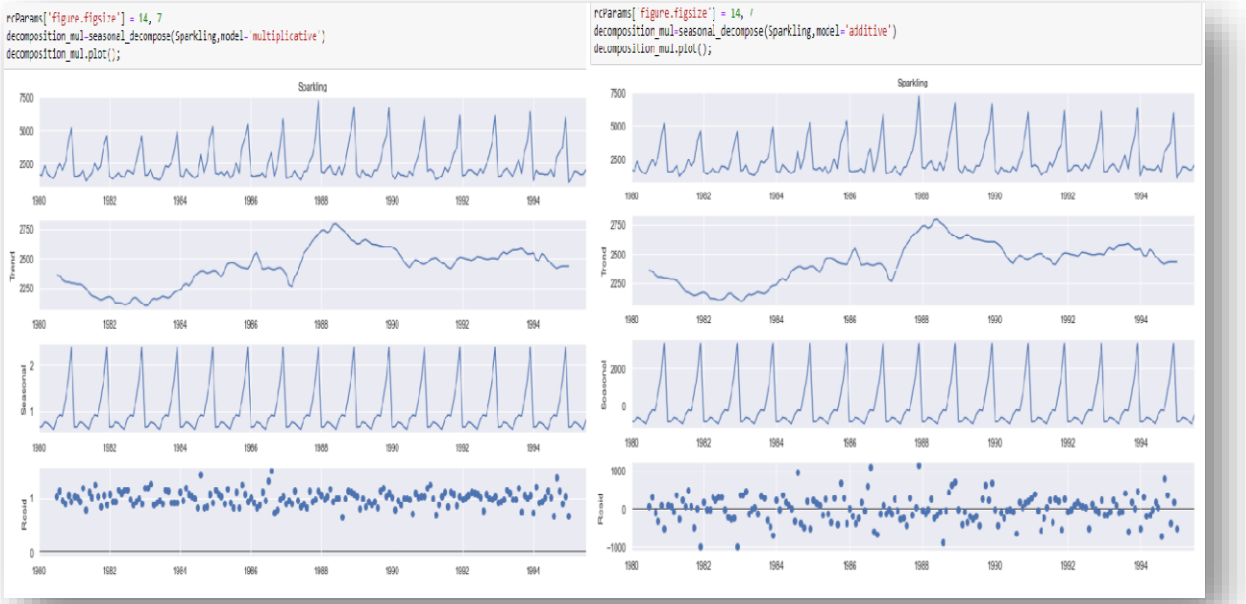
Sparkling Wines: Certain months have comparatively higher values throughout the years,

Performing Decomposition On Rose Wines Data:



Decomposition has been performed as observed in the above plot

Performing Decomposition On Sparkling Wines Data:



The time series is decomposed & the trend is observed for both Rose & Sparkling wines.

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

Train Test Split:

- *The test data is selected starting from 1991.*

1) Rose Wines

```
train_rose = rose[rose.index.year < 1991]
test_rose = rose[rose.index.year > 1990]

print('Length of the Train Data :',len(train_rose))
print('Length of the Train Data :',len(test_rose))
```

```
Length of the Train Data : 132
Length of the Train Data : 55
```

2) Sparkling Wines

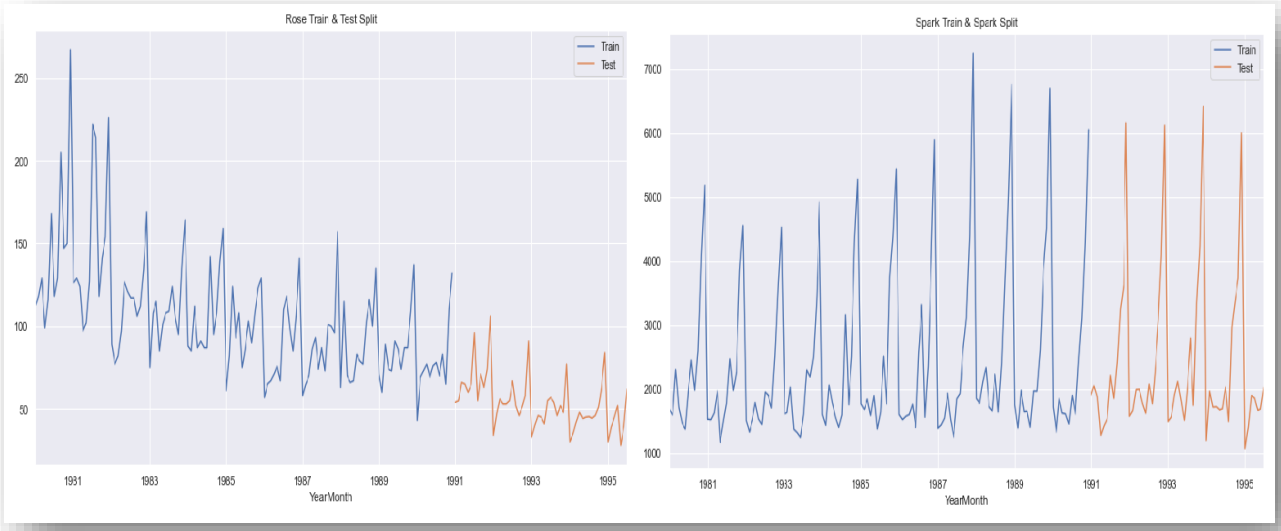
```
train_spark = Sparkling[Sparkling.index.year < 1991]
test_spark = Sparkling[Sparkling.index.year > 1990]

print('Length of the Train Data :',len(train_spark))
print('Length of the Train Data :',len(test_spark))
```

```
Length of the Train Data : 132
Length of the Train Data : 55
```

1) Time Series on Rose Wines Train & Test Data

2) Time Series on Sparkling Wines Train & Test Data



- *The train test split is done with the test data staring from the year 1991.*
- *There 132 values in the Train set and 55 values in the test set in Rose Wines Dataset & Sparkling Wines Dataset.*
- *A plot showing the train and test together is also observed.*

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 on Rose Wines Data & Sparkling Wines Data

1. *For this particular linear regression, we are going to regress the 'Rose' & 'Sparkling' variable against the order of the occurrence. To Modify the training data before fitting it into a linear regression & build a LR model. Finally, prediction of the output and measure RMSE on Test Data.*

TRAINING & TEST TIME INSTANCE: ROSE & SPARKLING WINES DATA

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

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

Linear Regression Model on Rose Wines ***Linear Regression Model on Sparkling Wines***

```
lr.fit(train_time,LR_train_rose)
```

```
LinearRegression()
```

```
lr.fit(train_time,LR_train_spark)
```

```
LinearRegression()
```

Train & Test Predictions : Rose Wines

```
train_predictions_reg.head()
```

```
YearMonth
1980-01-01    137.321144
1980-02-01    136.826766
1980-03-01    136.332388
1980-04-01    135.838010
1980-05-01    135.343632
dtype: float64
```

```
test_predictions_reg.head()
```

```
YearMonth
1991-01-01    116.557274
1991-02-01    116.062896
1991-03-01    115.568518
1991-04-01    115.074140
1991-05-01    114.579762
dtype: float64
```

Train & Test Predictions : Sparkling Wines

```
train_predictions_reg.head()
```

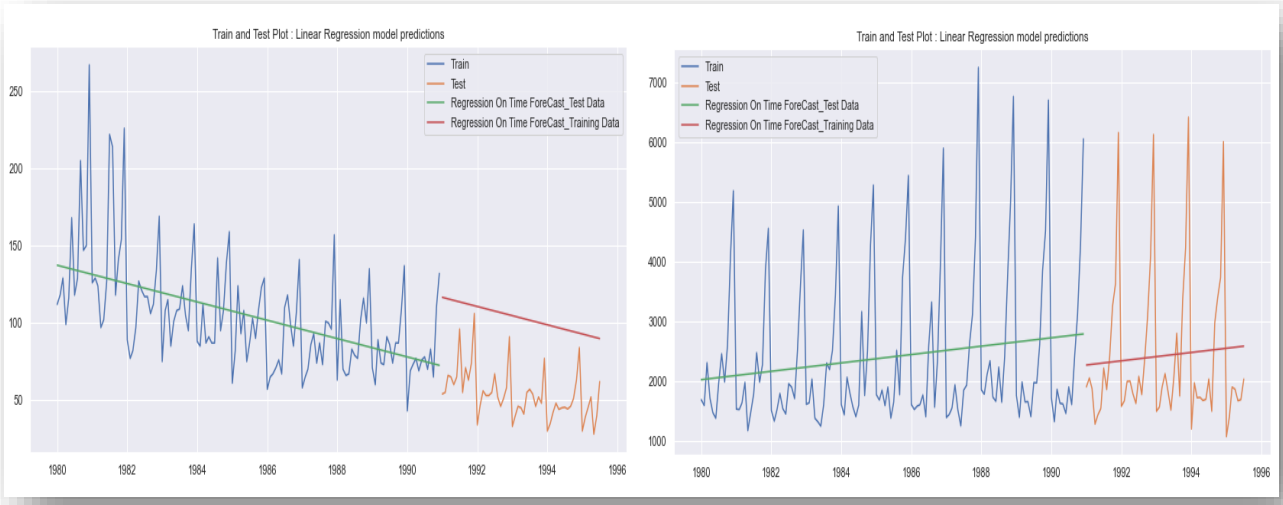
```
YearMonth
1980-01-01    2021.741171
1980-02-01    2027.573830
1980-03-01    2033.406488
1980-04-01    2039.239147
1980-05-01    2045.071805
dtype: float64
```

```
test_predictions_reg.head()
```

```
YearMonth
1991-01-01    2266.712828
1991-02-01    2272.545487
1991-03-01    2278.378145
1991-04-01    2284.210804
1991-05-01    2290.043462
dtype: float64
```

Time Series Plot with Regression (Train Vs Test)

Rose Wines Dataset ***Sparkling Wines Dataset***



Model Evaluation:

Regression On Time forecast on the Rose Test Data, RMSE is:

```
rmse_reg = np.sqrt(metrics.mean_squared_error(test_rose,test_predictions_reg))
print('Regression On Time Forecast on the Test Data,RMSE is %3.3F'%(rmse_reg))
```

```
Regression On Time Forecast on the Test Data,RMSE is 51.457
```

Regression On Time forecast on the Sparkling Test Data, RMSE is:

```
rmse_reg = np.sqrt(metrics.mean_squared_error(test_spark,test_predictions_reg))
print('Regression On Time Forecast on the Test Data,RMSE is %3.3F'%(rmse_reg))
```

```
Regression On Time Forecast on the Test Data,RMSE is 1275.867
```

Model 2: Naive Approach on Rose Wines Data & Sparkling Wines Data

The Naive model is a level prediction with the prediction as $\hat{y}_{t+1}=y_t$

To Build the Naive Model & Predict the output and measure RMSE on Test Data.

1) Rose Wines

```
NaiveModel_train = pd.Series(train_rose.iloc[-1], index = train_rose.index)
NaiveModel_train.head()
```

```
YearMonth
1980-01-01    132.0
1980-02-01    132.0
1980-03-01    132.0
1980-04-01    132.0
1980-05-01    132.0
dtype: float64
```

```
NaiveModel_test = pd.Series(train_rose.iloc[-1], index = test_rose.index)
NaiveModel_test.head()
```

```
YearMonth
1991-01-01    132.0
1991-02-01    132.0
1991-03-01    132.0
1991-04-01    132.0
1991-05-01    132.0
dtype: float64
```

2) Sparkling Wines

```
NaiveModel_train = pd.Series(train_spark.iloc[-1], index = train_spark.index)
NaiveModel_train.head()
```

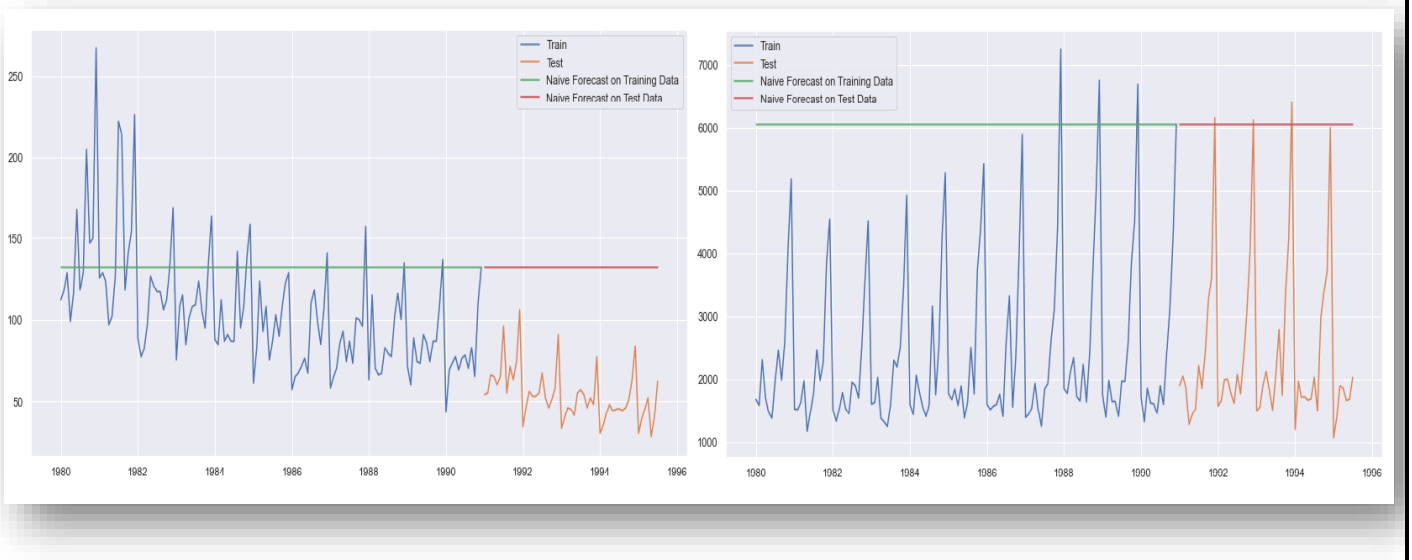
```
YearMonth
1980-01-01    6047
1980-02-01    6047
1980-03-01    6047
1980-04-01    6047
1980-05-01    6047
dtype: int64
```

```
NaiveModel_test = pd.Series(train_spark.iloc[-1], index = test_spark.index)
NaiveModel_test.head()
```

```
YearMonth
1991-01-01    6047
1991-02-01    6047
1991-03-01    6047
1991-04-01    6047
1991-05-01    6047
dtype: int64
```

Plot for Naive Forecast - Rose Wines Data:

Plot for Naive Forecast - Sparkling Wines Data:



Model Evaluation:

Naive Forecast on the Rose Test Data, RMSE is:

```
rmse_naive = np.sqrt(metrics.mean_squared_error(test_rose,NaiveModel_test))
print("For Naive forecast on the Test Data, RMSE is %3.3f " %(rmse_naive))
```

```
For Naive forecast on the Test Data, RMSE is 79.746
```

Naive Forecast on the Sparkling Test Data, RMSE is:

```
rmse_naive = np.sqrt(metrics.mean_squared_error(test_spark,NaiveModel_test))
print("For Naive forecast on the Test Data, RMSE is %3.3f " %(rmse_naive))
```

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

The Naive model is trained using the Train data and then the test values are predicted & the RMSE Values were obtained as shown above.

Method 3: Simple Average Method on Rose Wines Data & Sparkling Wines Data

The Simple Average model prediction is a level prediction with the prediction calculated by finding the average of the Training values. To Build the Simple Average model & predict the output and measure RMSE on Test Data. The forecasting technique which forecasts the expected value equal to the average of all previously observed points is Simple Average Technique.

1) Rose Wines

```
SimpleAverage_train = pd.Series(train_rose.mean(), index = train_rose.index)
SimpleAverage_train.head()
```

```
YearMonth
1980-01-01    104.939394
1980-02-01    104.939394
1980-03-01    104.939394
1980-04-01    104.939394
1980-05-01    104.939394
dtype: float64
```

```
SimpleAverage_test = pd.Series(train_rose.mean(), index = test_rose.index)
SimpleAverage_test.head()
```

```
YearMonth
1991-01-01    104.939394
1991-02-01    104.939394
1991-03-01    104.939394
1991-04-01    104.939394
1991-05-01    104.939394
dtype: float64
```

2) Sparkling Method

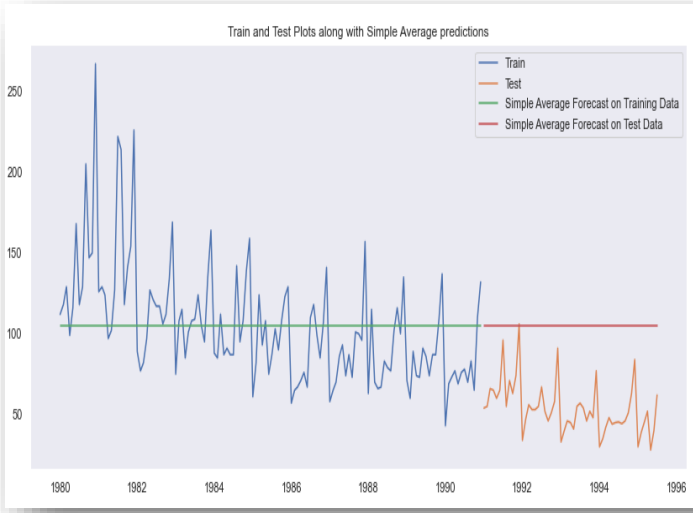
```
SimpleAverage_train = pd.Series(train_spark.mean(), index = train_spark.index)
SimpleAverage_train.head()
```

```
YearMonth
1980-01-01    2403.780303
1980-02-01    2403.780303
1980-03-01    2403.780303
1980-04-01    2403.780303
1980-05-01    2403.780303
dtype: float64
```

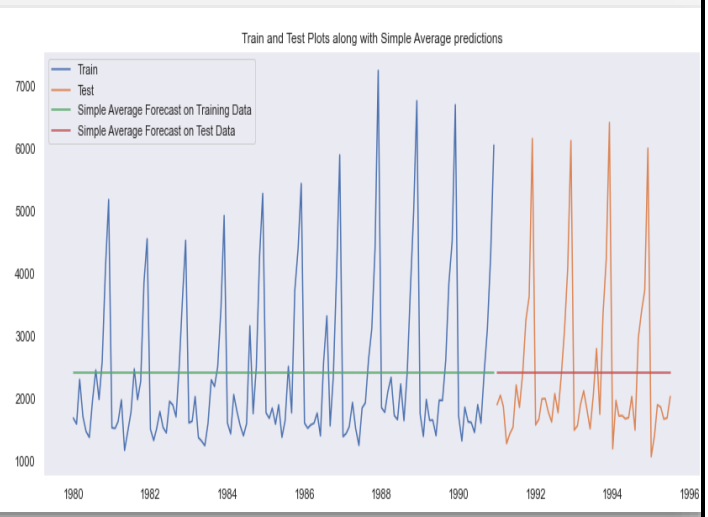
```
SimpleAverage_test = pd.Series(train_spark.mean(), index = test_spark.index)
SimpleAverage_test.head()
```

```
YearMonth
1991-01-01    2403.780303
1991-02-01    2403.780303
1991-03-01    2403.780303
1991-04-01    2403.780303
1991-05-01    2403.780303
dtype: float64
```

Plot for Simple Average Forecast - Rose Wines Data



Plot for Simple Average Forecast - Sparkling Wines Data



Model Evaluation:

For Simple Average forecast on the Rose Test Data, RMSE is:

```
rmse_simple_avg = np.sqrt(metrics.mean_squared_error(test_rose, SimpleAverage_test))
print("For Simple Average forecast on the Test Data, RMSE is %3.3f " %(rmse_simple_avg))
```

```
For Simple Average forecast on the Test Data, RMSE is 53.488
```

For Simple Average forecast on the Sparkling Test Data, RMSE is:

```
rmse_simple_avg = np.sqrt(metrics.mean_squared_error(test_spark, SimpleAverage_test))
print("For Simple Average forecast on the Test Data, RMSE is %3.3f " %(rmse_simple_avg))
```

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

Method 4: Simple Exponential Smoothing on Rose Wines Data & Sparkling Wines Data

The Simple Exponential Smoothing has the smoothing level parameter α which is optimized using inbuilt parameter optimized and also optimized iteratively based on Test RMSE values. To Build the Simple Exponential Smoothing model & predict the output and measure RMSE on Test Data.

Rose Wines Data: Autofit Params

```
model_SES_autofit = model_SES.fit(optimized=True)
model_SES_autofit.params
```

```
{'smoothing_level': 0.09874989825614361,
 'smoothing_trend': nan,
 'smoothing_seasonal': nan,
 'damping_trend': nan,
 'initial_level': 134.38702255613862,
 'initial_trend': nan,
 'initial_seasons': array([], dtype=float64),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

Sparkling Wines Data: Autofit Params

```
model_SES_autofit = model_SES.fit(optimized=True)
model_SES_autofit.params
```

```
{'smoothing_level': 0.04960736049406556,
 'smoothing_trend': nan,
 'smoothing_seasonal': nan,
 'damping_trend': nan,
 'initial_level': 2151.614314422547,
 'initial_trend': nan,
 'initial_seasons': array([], dtype=float64),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

Rose Wines data set with Predict values:

```
SES_train_predict = model_SES_autofit.fittedvalues
SES_train_predict.head()
```

```
YearMonth
1980-01-01    134.387023
1980-02-01    132.176306
1980-03-01    130.776398
1980-04-01    130.600978
1980-05-01    127.480385
dtype: float64
```

```
SES_test_predict = model_SES_autofit.forecast(steps=len(test_rose))
SES_test_predict.head()
```

```
1991-01-01    87.104999
1991-02-01    87.104999
1991-03-01    87.104999
1991-04-01    87.104999
1991-05-01    87.104999
Freq: MS, dtype: float64
```

Sparkling Wines data set with Predict values

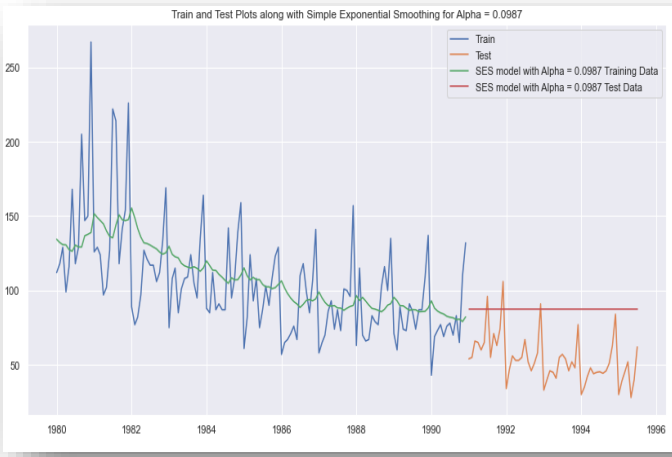
```
SES_train_predict = model_SES_autofit.fittedvalues
SES_train_predict.head()
```

```
YearMonth
1980-01-01    2151.614314
1980-02-01    2128.516417
1980-03-01    2101.851647
1980-04-01    2111.879693
1980-05-01    2092.042717
dtype: float64
```

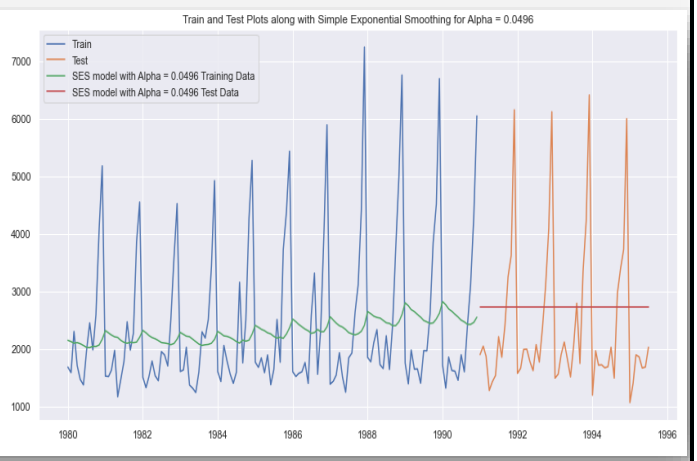
```
SES_test_predict = model_SES_autofit.forecast(steps=len(test_spark))
SES_test_predict.head()
```

```
1991-01-01    2725.336037
1991-02-01    2725.336037
1991-03-01    2725.336037
1991-04-01    2725.336037
1991-05-01    2725.336037
Freq: MS, dtype: float64
```

Plot for SES (α - 0.0987) - Rose Wines Data



Plot for SES (α 0.0496)- Sparkling Wines Data



Model Evaluation:

For (α - 0.0987) Simple Exponential Smoothing forecast on the Rose Test Data, RMSE is:

```
rmse_model5_test = np.sqrt(metrics.mean_squared_error(SES_test,SES_test_predict))
print("For Alpha = 0.0987 Simple Exponential Smoothing Model forecast on the Test Data,  RMSE is %3.3f " %(rmse_model5_test))
```

```
For Alpha = 0.0987 Simple Exponential Smoothing Model forecast on the Test Data,  RMSE is 36.824
```

For (α - 0.0496) Simple Exponential Smoothing forecast on the Sparkling Test Data, RMSE is:

```
rmse_model5_test = np.sqrt(metrics.mean_squared_error(SES_test,SES_test_predict))
print("For Alpha = 0.0496 Simple Exponential Smoothing Model forecast on the Test Data,  RMSE is %3.3f " %(rmse_model5_test))
```

```
For Alpha = 0.0496 Simple Exponential Smoothing Model forecast on the Test Data,  RMSE is 1316.135
```

Iterative Method for Simple Exponential Smoothing.

Setting different alpha Values to check if we could derive better RMSE Values.

Rose Wines Data

```
resultsDf_SES_iter.sort_values(by = 'Test RMSE')
```

	Alpha Values	Test RMSE
2	0.10	36.856268
1	0.05	37.039679
3	0.15	38.750307
4	0.20	41.389972
5	0.25	44.388786
6	0.30	47.532697
7	0.35	50.693433
8	0.40	53.795058
9	0.45	56.794685
10	0.50	59.669244
0	0.00	60.270818
11	0.55	62.406363
12	0.60	64.998585
13	0.65	67.440131
14	0.70	69.725329
15	0.75	71.847964
16	0.80	73.801055
17	0.85	75.576756
18	0.90	77.166259
19	0.95	78.559647
20	1.00	79.745697

Sparkling Wines Data

```
resultsDf_SES_iter.sort_values(by = 'Test RMSE')
```

	Alpha Values	Test RMSE
1	0.05	1316.411742
2	0.10	1375.393398
0	0.00	1460.954675
3	0.15	1466.203651
4	0.20	1595.206839
5	0.25	1755.488175
6	0.30	1935.507132
7	0.35	2123.914871
8	0.40	2311.919615
9	0.45	2493.786514
10	0.50	2666.351413
11	0.55	2828.246418
12	0.60	2979.204388
13	0.65	3119.560885
14	0.70	3249.944092
15	0.75	3371.100106
16	0.80	3483.801006
17	0.85	3588.797654
18	0.90	3686.794285
19	0.95	3778.432623
20	1.00	3864.279352

Inference:

- *Rose Wines Dataset: It is observed that RMSE on Test data is least for alpha = 0.10, a value of 36.85 which is very close to alpha = 0.0987 as obtained for optimization model. And hence we will keep the optimized value of Alpha as the best model parameter for Single Exponential Smoothing.*
- *Sparkling Wines Dataset: t is observed that RMSE on Test data is least for alpha = 0.05, a value of 1316.41 which is very close to alpha = 0.0496 as obtained for optimization model. And hence*

we will keep the optimized value of Alpha as the best model parameter for Single Exponential Smoothing.

Method 5: Double Exponential Smoothing (Holt's Model) on Rose Wines Data & Sparkling Wines Data

The Double Exponential Smoothing has two parameters smoothing level α and smoothing slope β Parameter which are optimized using inbuilt parameter optimized and also optimized iteratively based on Test RMSE values. To Build the Double Exponential Smoothing model & predict the output and measure RMSE on Test Data.

Rose Wines Data: Autofit Params

```
model_DES_autofit = model_DES.fit(optimized=True,use_brute=True)
model_DES_autofit.params
```

```
{'smoothing_level': 0.0001321339938406504,
 'smoothing_trend': 1.0513882039253742e-16,
 'smoothing_seasonal': nan,
 'damping_trend': nan,
 'initial_level': 136.22441775313408,
 'initial_trend': -0.4786757879461521,
 'initial_seasons': array([], dtype=float64),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

Sparkling Wines Data: Autofit Params

```
model_DES_autofit = model_DES.fit(optimized=True,use_brute=True)
model_DES_autofit.params
```

```
{'smoothing_level': 0.6885714285714285,
 'smoothing_trend': 9.999999999999999e-05,
 'smoothing_seasonal': nan,
 'damping_trend': nan,
 'initial_level': 1686.0,
 'initial_trend': -95.0,
 'initial_seasons': array([], dtype=float64),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

Rose Wines data set with Predict values:

```
DES_train_predict = model_DES_autofit.fittedvalues
DES_train_predict.head()
```

```
YearMonth
1980-01-01    135.745742
1980-02-01    135.263929
1980-03-01    134.782972
1980-04-01    134.303532
1980-05-01    133.820191
dtype: float64
```

```
DES_test_predict = model_DES_autofit.forecast(steps=len(test_rose))
DES_test_predict.head()
```

```
1991-01-01    72.569944
1991-02-01    72.091268
1991-03-01    71.612593
1991-04-01    71.133917
1991-05-01    70.655241
Freq: MS, dtype: float64
```

Sparkling Wines data set with Predict values

```
DES_train_predict = model_DES_autofit.fittedvalues
DES_train_predict.head()
```

```
YearMonth
1980-01-01    1591.000000
1980-02-01    1561.420827
1980-03-01    1486.796779
1980-04-01    1954.564417
1980-05-01    1692.589636
dtype: float64
```

```
DES_test_predict = model_DES_autofit.forecast(steps=len(test_spark))
DES_test_predict.head()
```

```
1991-01-01    5221.278699
1991-02-01    5127.886554
1991-03-01    5034.494409
1991-04-01    4941.102264
1991-05-01    4847.710119
Freq: MS, dtype: float64
```

MODEL EVALUATION:

For DES forecast on the Rose Test Data, RMSE (parameters as derived from Autofit Model) is:

```
rmse_model_test_des_autofit = np.sqrt(metrics.mean_squared_error(DES_test,DES_test_predict))
print(" Double Exponential Smoothing Model forecast on the Test Data,  RMSE is %3.3f "
      %(rmse_model_test_des_autofit))
```

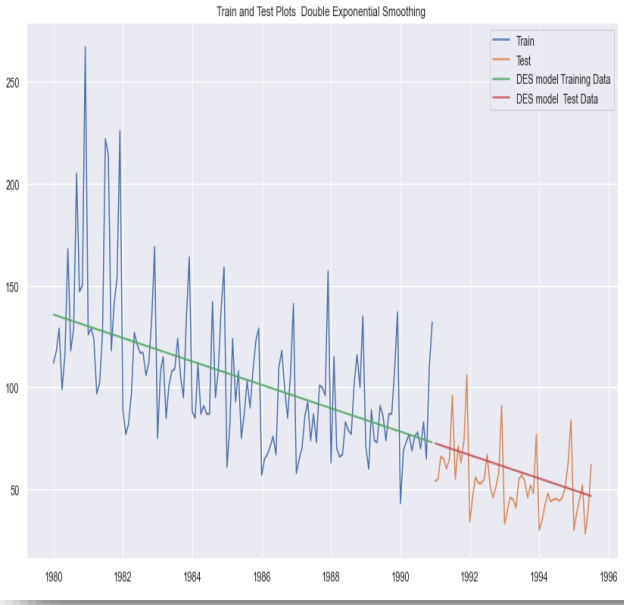
```
 Double Exponential Smoothing Model forecast on the Test Data,  RMSE is 15.580
```

For DES forecast on the Sparkling Test Data, RMSE (parameters as derived from Autofit Model) is:

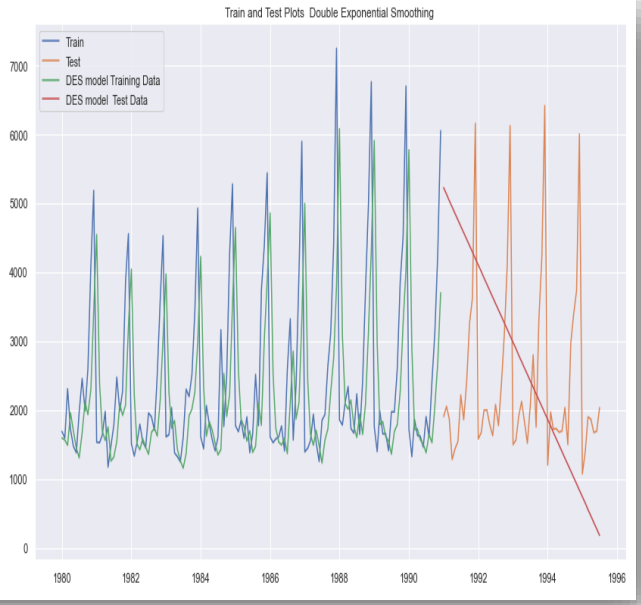
```
rmse_model_test_des_autofit = np.sqrt(metrics.mean_squared_error(DES_test,DES_test_predict))
print(" Double Exponential Smoothing Model forecast on the Test Data,  RMSE is %3.3f "
      %(rmse_model_test_des_autofit))
```

```
 Double Exponential Smoothing Model forecast on the Test Data,  RMSE is 2007.239
```

Plot for DES - Rose Wines Data



Plot for SES - Sparkling Wines Data



Iterative Method for Double Exponential Smoothing.

- *It is observed that the Trend is not properly captured by the optimized method. Setting different alpha & Beta values using iterative method to check for deriving better results.*

Rose Wine Dataset: α & β Values

```
resultsDf_DES_iter.sort_values(by=['Test RMSE']).head()
```

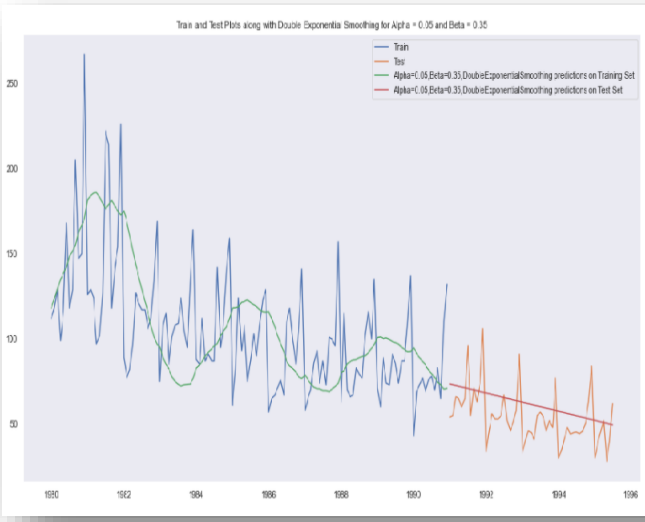
	Alpha Values	Beta Values	Test RMSE
28	0.05	0.35	16.343344
27	0.05	0.30	18.640750
24	0.05	0.15	23.741250
22	0.05	0.05	31.555972
29	0.05	0.40	31.609086

Sparkling Wine Dataset: α & β Values

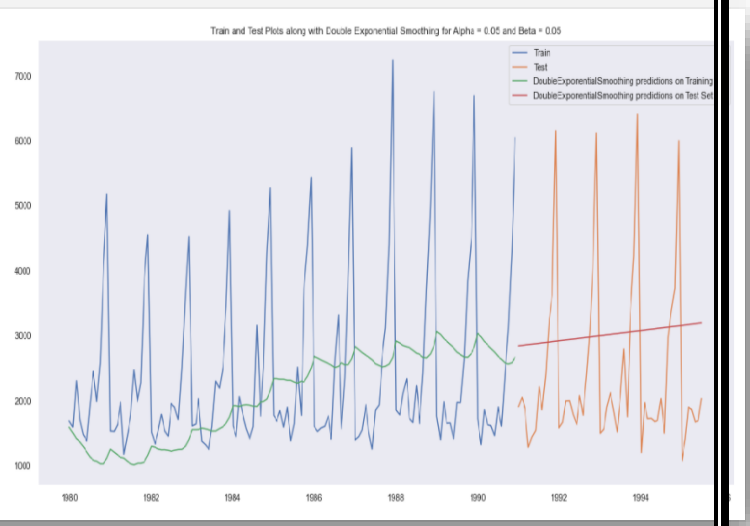
```
resultsDf_DES_iter.sort_values(by=['Test RMSE']).head()
```

	Alpha Values	Beta Values	Test RMSE
22	0.05	0.05	1418.407668
25	0.05	0.20	1443.099273
24	0.05	0.15	1457.041594
23	0.05	0.10	1466.899629
28	0.05	0.35	1547.022626

Plot for DES (α - 0.05 & β- 0.35) - Rose Wines Data



Plot for DES (α - 0.05 & β- 0.05)- Sparkling Wines Data



Model Evaluation:

For (α - 0.05 & β - 0.35) Double Exponential Smoothing forecast on the Rose Test Data, RMSE is

```
resultsDf_iter = pd.DataFrame({'Test RMSE': [resultsDf_DES_iter['Test RMSE'][0]]}
                          ,index=['Alpha=0.05, Beta=0.35, Double Exponential Smoothing Iterative'])

resultsDf = pd.concat([resultsDf, resultsDf_iter])
resultsDf.iloc[-1,:]
```

```
Test RMSE    16.343344
Name: Alpha=0.05, Beta=0.35, Double Exponential Smoothing Iterative, dtype: float64
```

For (α - 0.05 & β - 0.05) Double Exponential Smoothing forecast on the Sparkling Test Data, RMSE is

```
resultsDf_iter = pd.DataFrame({'Test RMSE': [resultsDf_DES_iter['Test RMSE'][0]]}
                          ,index=['Alpha=0.05, Beta=0.05, Double Exponential Smoothing Iterative'])

resultsDf = pd.concat([resultsDf, resultsDf_iter])
resultsDf.iloc[-1,:]
```

```
Test RMSE    1418.407668
Name: Alpha=0.05, Beta=0.05, Double Exponential Smoothing Iterative, dtype: float64
```

Inference:

- *Rose Wines Dataset: It is observed that for Double Exponential smoothing the minimum value of RMSE on Test is 16.34 as obtained by using the iterative method for smoothing level α = 0.05 and smoothing slope β = 0.35.*
- *Sparkling Wines Dataset: It is observed that for Double Exponential smoothing the minimum value of RMSE on Test is 1418.40 as obtained by using the iterative method for smoothing level α = 0.05 and smoothing slope = 0.05.*

Method 6: Triple Exponential Smoothing (Holt - Winter's Model) on Rose Wines Data & Sparkling Wines Data

The Triple Exponential Smoothing has three parameters smoothing level α, smoothing slope β and smoothing seasonal γ Parameter which are optimized using inbuilt parameter optimized and also optimized iteratively based on Test RMSE values. To Build the Double Exponential Smoothing model & predict the output and measure RMSE on Test Data.

Rose Wines Data: Autofit Params

```
model_TES_autofit.params
```

```
{'smoothing_level': 0.06467234615091698,
 'smoothing_trend': 0.05315920636255018,
 'smoothing_seasonal': 0.0,
 'damping_trend': nan,
 'initial_level': 50.880912909225756,
 'initial_trend': -0.31656840824205823,
 'initial_seasons': array([2.21583703, 2.51439498, 2.74693025, 2.40118428, 2.69936273,
        2.94338111, 3.2353888 , 3.44052906, 3.26420741, 3.19365239,
        3.72269442, 5.13435788]),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

Sparkling Wines Data: Autofit Params

```
model_TES_autofit.params
```

```
{'smoothing_level': 0.11108840858679117,
 'smoothing_trend': 0.061712060020663685,
 'smoothing_seasonal': 0.395081480215160 3,
 'damping_trend': nan,
 'initial_level': 1639.9088356475902,
 'initial_trend': -11.928143593549056,
 'initial_seasons': array([1.05065032, 1.02086214, 1.41078482, 1.20263518, 0.97315225,
        0.96689379, 1.31724304, 1.70471609, 1.37289733, 1.81035002,
        2.83962708, 3.60997333]),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

Rose Wines data set with Predict values

```
TES_train['auto_predict'] = model_TES_autofit.fittedvalues
TES_train.head()
```

YearMonth	Rose	auto_predict
1980-01-01	112.0	112.042347
1980-02-01	118.0	126.339483
1980-03-01	129.0	136.533254
1980-04-01	99.0	118.112130
1980-05-01	116.0	130.404942

```
TES_test['auto_predict'] = model_TES_autofit.forecast(steps=len(test_rose))
TES_test.head()
```

YearMonth	Rose	auto_predict
1991-01-01	54.0	56.755640
1991-02-01	55.0	64.211013
1991-03-01	66.0	69.939833
1991-04-01	65.0	60.953618
1991-05-01	60.0	68.316934

Sparkling Wines data set with Predict values

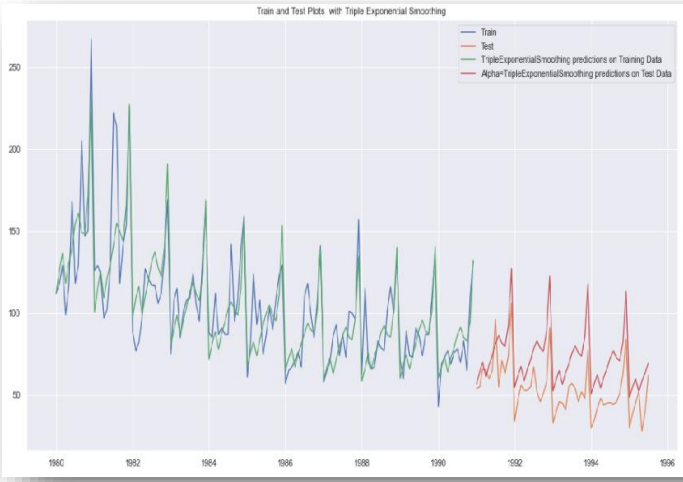
```
TES_train['auto_predict'] = model_TES_autofit.fittedvalues
TES_train.head()
```

YearMonth	Sparkling	auto_predict
1980-01-01	1686	1710.438443
1980-02-01	1591	1646.966216
1980-03-01	2304	2249.856923
1980-04-01	1712	1908.363831
1980-05-01	1471	1513.602655

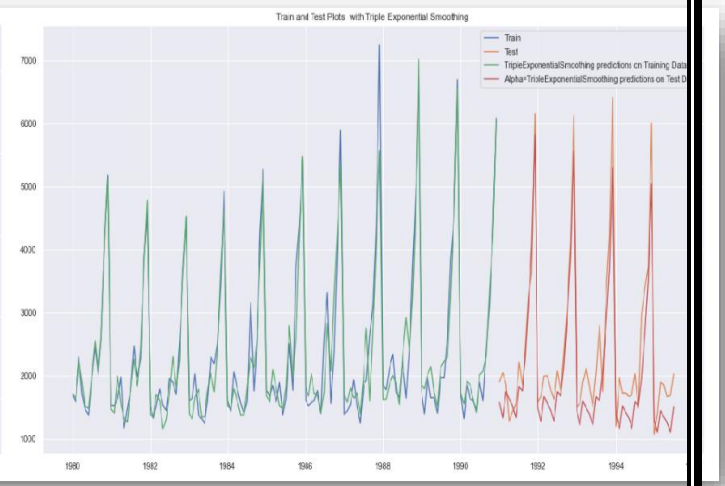
```
TES_test['auto_predict'] = model_TES_autofit.forecast(steps=len(test_spark))
TES_test.head()
```

YearMonth	Sparkling	auto_predict
1991-01-01	1902	1577.208163
1991-02-01	2049	1333.663154
1991-03-01	1874	1745.977341
1991-04-01	1279	1630.435405
1991-05-01	1432	1523.306429

Plot for TES - Rose Wines Data



Plot for TES - Sparkling Wines Data



MODEL EVALUATION:

For TES forecast on the Rose Test Data, RMSE (parameters as derived from Autofit Model) is:

```
rmse_model_test_TES_auto = np.sqrt(metrics.mean_squared_error(TES_test['Rose'],TES_test['auto_predict']))
print(" Triple Exponential Smoothing Model forecast on the Test Data, RMSE is %3.3f" %rmse_model_test_TES_auto)
```

```
 Triple Exponential Smoothing Model forecast on the Test Data, RMSE is 21.192
```

For TES forecast on the Sparkling Test Data, RMSE (parameters as derived from Autofit Model) is:

```
rmse_model_test_TES_auto = np.sqrt(metrics.mean_squared_error(TES_test['Sparkling'],TES_test['auto_predict']))
print(" Triple Exponential Smoothing Model forecast on the Test Data, RMSE is %3.3f" %rmse_model_test_TES_auto)
```

```
 Triple Exponential Smoothing Model forecast on the Test Data, RMSE is 469.659
```

Iterative Method for Double Exponential Smoothing.

Setting different alpha, Beta & Gamma values using iterative method to check for deriving better results.

Rose Wine Dataset: α, β & γ Values

```
resultsDf_TES_iter.sort_values(by=['Test RMSE']).head()
```

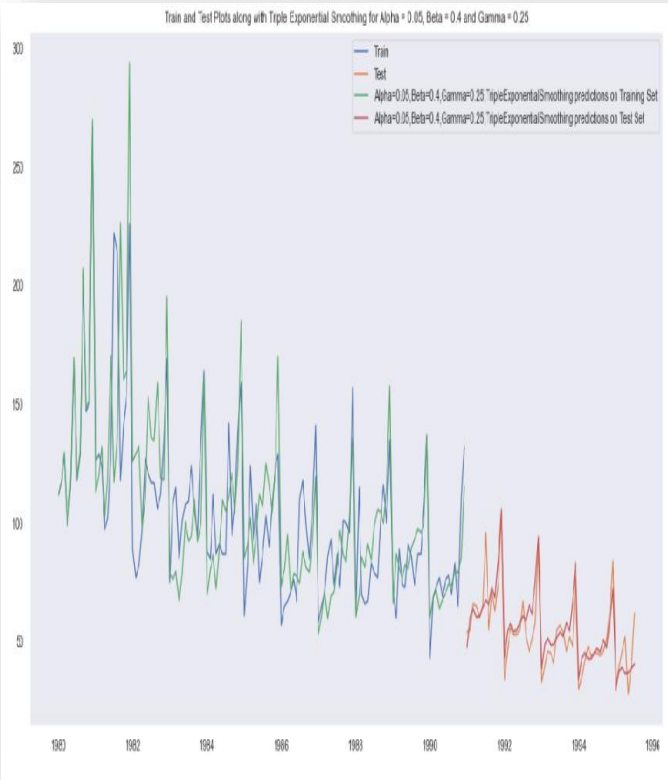
	Alpha Values	Beta Values	Gamma Values	Test RMSE
614	0.05	0.40	0.25	8.698606
613	0.05	0.40	0.20	8.717346
615	0.05	0.40	0.30	8.998488
597	0.05	0.35	0.45	9.038002
596	0.05	0.35	0.40	9.084235

Sparkling Wine Dataset: α, β & γ Values

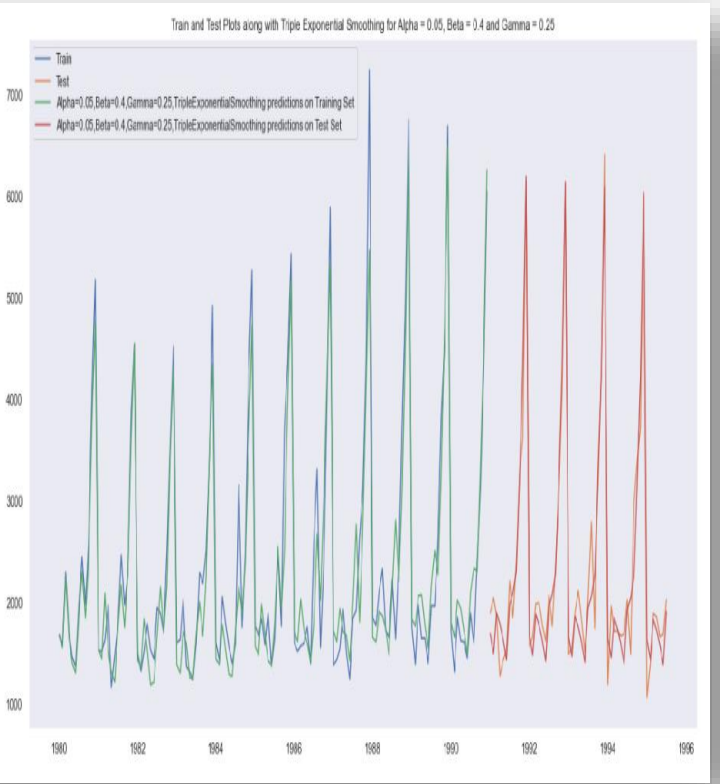
```
resultsDf_TES_iter.sort_values(by=['Test RMSE']).head()
```

	Alpha Values	Beta Values	Gamma Values	Test RMSE
467	0.05	0.05	0.25	302.454819
468	0.05	0.05	0.30	303.489118
466	0.05	0.05	0.20	306.817930
469	0.05	0.05	0.35	307.680766
803	0.05	0.85	0.25	309.959496

Plot for TES (α - 0.05, β- 0.35 & γ- 0.25) - Rose Wines Data



Plot for TES (α - 0.05 & β- 0.05)- Sparkling Wines Data



MODEL EVALUATION:

For (α - 0.05, β - 0.35 & γ - 0.25) Triple Exponential Smoothing forecast on the Rose Test Data, RMSE is:

```
resultsDf_iter = pd.DataFrame({'Test RMSE': [resultsDf_TES_iter['Test RMSE'][0]]}
                          ,index=['Alpha=0.05,Beta=0.4,Gamma=0.25, Triple Exponential Smoothing Iterative'])

resultsDf = pd.concat([resultsDf, resultsDf_iter])
resultsDf.iloc[-1,:]
```

```
Test RMSE    8.698606
Name: Alpha=0.05,Beta=0.4,Gamma=0.25, Triple Exponential Smoothing Iterative, dtype: float64
```

For (α - 0.05, β - 0.35 & γ - 0.25) Triple Exponential Smoothing forecast on the Sparkling Test Data, RMSE is:

```
resultsDf_iter = pd.DataFrame({'Test RMSE': [resultsDf_TES_iter['Test RMSE'][0]]}
                          ,index=['Alpha=0.05,Beta=0.4,Gamma=0.25, Triple Exponential Smoothing Iterative'])

resultsDf = pd.concat([resultsDf, resultsDf_iter])
resultsDf.iloc[-1,:]
```

```
Test RMSE    302.454819
Name: Alpha=0.05,Beta=0.4,Gamma=0.25, Triple Exponential Smoothing Iterative, dtype: float64
```

Inferences:

- *Rose Wines Dataset: It is observed that for Triple Exponential smoothing the minimum value of RMSE on Test is 8.6986 as obtained by using the iterative method for smoothing level α = 0.05, smoothing slope β = 0.4 and smoothing seasonal γ = 0.25.*

- *Sparkling Wines Dataset: It is observed that for Triple Exponential smoothing the minimum value of RMSE on Test is 302.45 as obtained by using the iterative method for smoothing level α = 0.05, smoothing slope β = 0.4 and smoothing seasonal γ = 0.25.*

Results:

Rose Wine Data:

	Test RMSE
Regression On Time	51.457439
Naive Model	79.745697
Simple Average Model	53.488233
Alpha=0.0987, Simple Exponential Smoothing Optimized	36.824478
Double Exponential Smoothing Optimized	15.580105
Alpha=0.05, Beta=0.35, Double Exponential Smoothing Iterative	16.343344
Alpha=0.05,Beta=0.4,Gamma=0.25, Triple Exponential Smoothing Iterative	8.698606

Sparkling Wine Data:

	Test RMSE
Regression On Time	1275.867052
Naive Model	3864.279352
Simple Average Model	1275.081804
Alpha=0.0496, Simple Exponential Smoothing Optimized	1316.135411
Double Exponential Smoothing Optimized	2007.238526
Alpha=0.05, Beta=0.35, Double Exponential Smoothing Iterative	1418.407668
Alpha=0.05,Beta=0.4,Gamma=0.25, Triple Exponential Smoothing Iterative	302.454819

- *From the above models, for Rose Wines Data till now the best Test RMSE score of 8.6986 is obtained for Triple Exponential Smoothing Model with smoothing level α = 0.05, smoothing slope β = 0.4 and smoothing seasonal γ = 0.25.*
- *From the above models, for Sparkling Wines Data till now the best Test RMSE score of 302.454 is obtained for Triple Exponential Smoothing Model with smoothing level α = 0.05, smoothing slope β = 0.4 and smoothing seasonal γ = 0.25.*

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.

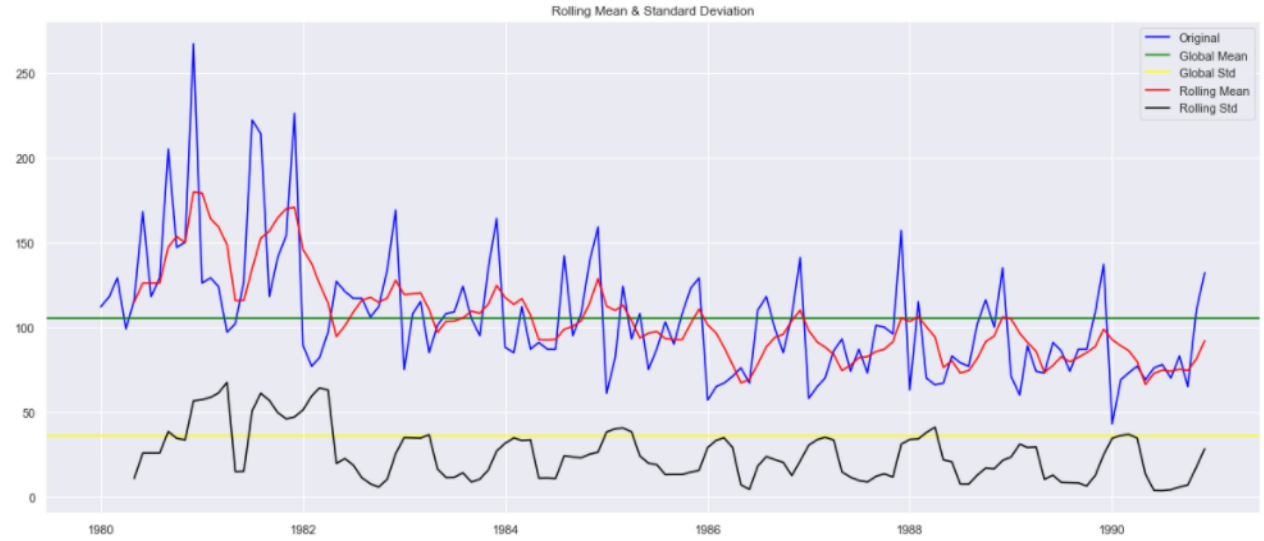
Stationarity should be checked at alpha = 0.05. Augmented Dickey Fuller (ADF) test is used. Here the Null and Alternate Hypothesis are as follows:

- *Null Hypothesis: The Time series is non-stationary.*
- *Alternate Hypothesis: The Time series is stationary.*

- *Here the Null hypothesis is rejected at a significance level of 95% (alpha = 0.05) i.e.: a p value of less than equal to 0.05 for the ADF test will be evident to reject the Null hypothesis and henceforth the time series will be considered as stationary.*

Test for stationarity on Rose Wines Data:

```
test_stationarity(train_rose)
```



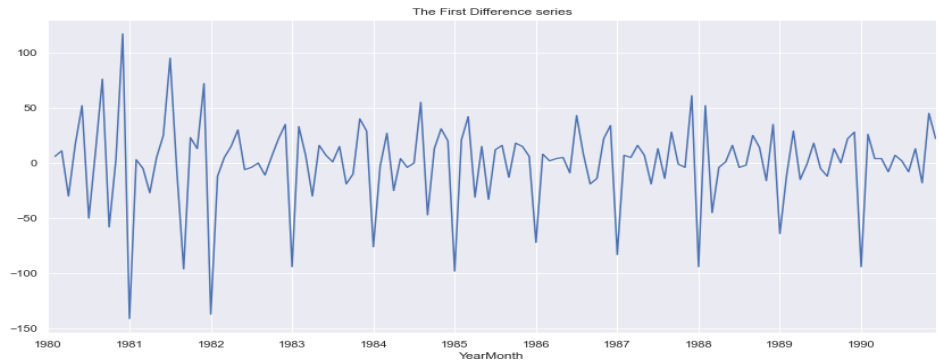

```
Results of Dickey-Fuller Test:
Test Statistic                  -2.164250
p-value                          0.219476
#Lags Used                      13.000000
Number of Observations Used    118.000000
Critical Value (1%)             -3.487022
Critical Value (5%)             -2.886363
Critical Value (10%)            -2.580009
dtype: float64
```

The ADF test gives a p value (0.22 approx.) > than significance level alpha = 0.05, Hence we fail to reject Null Hypothesis as Series is non-stationary.

Calculating the 1st difference:

This method is carried out to remove any additive trends present in the series & to make the data stationary.

```
train_rose_first_diff.plot()
plt.title('The First Difference series');
```

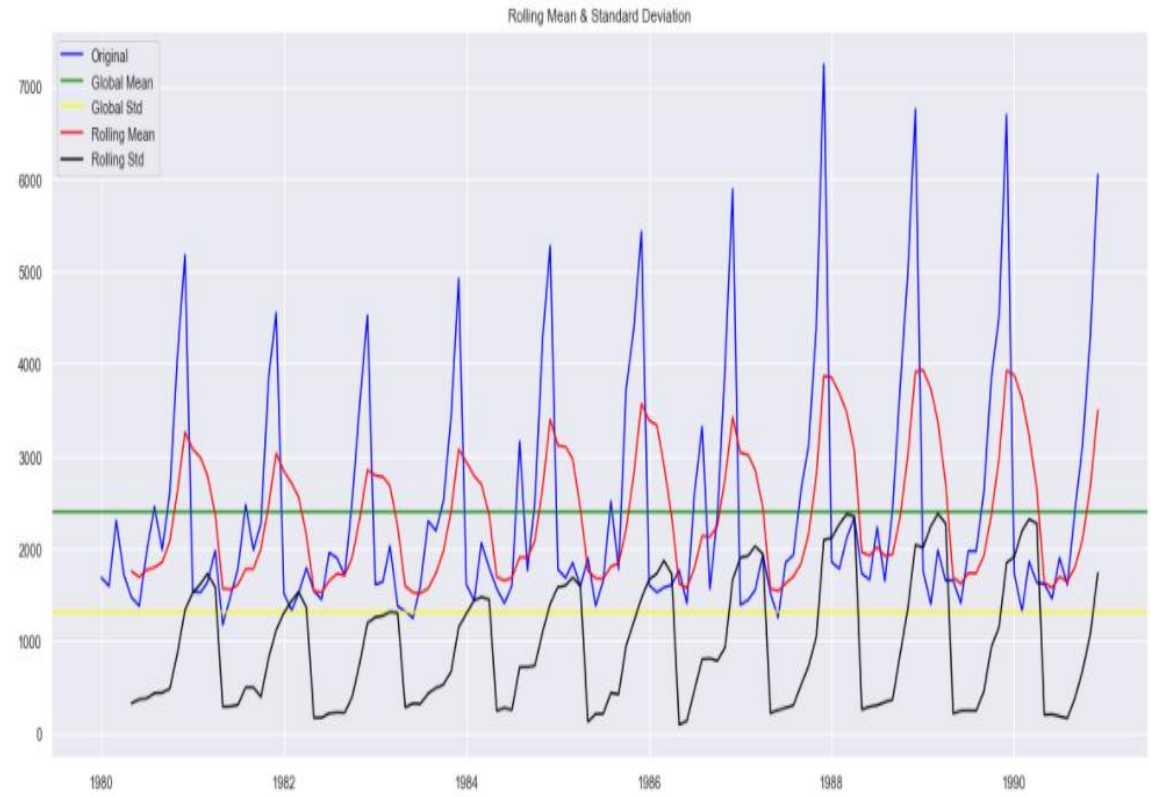


```
Results of Dickey-Fuller Test:
Test Statistic                 -6.592372e+00
p-value                         7.061944e-09
#Lags Used                      1.200000e+01
Number of Observations Used     1.180000e+02
Critical Value (1%)            -3.487022e+00
Critical Value (5%)            -2.886363e+00
Critical Value (10%)           -2.580009e+00
dtype: float64
```

After taking a difference of order 1 the series have become stationary at $\alpha = 0.05$ on Rose Wines Data.

Test for stationarity on Sparkling Wines Data:

```
test_stationarity(train_spark)
```



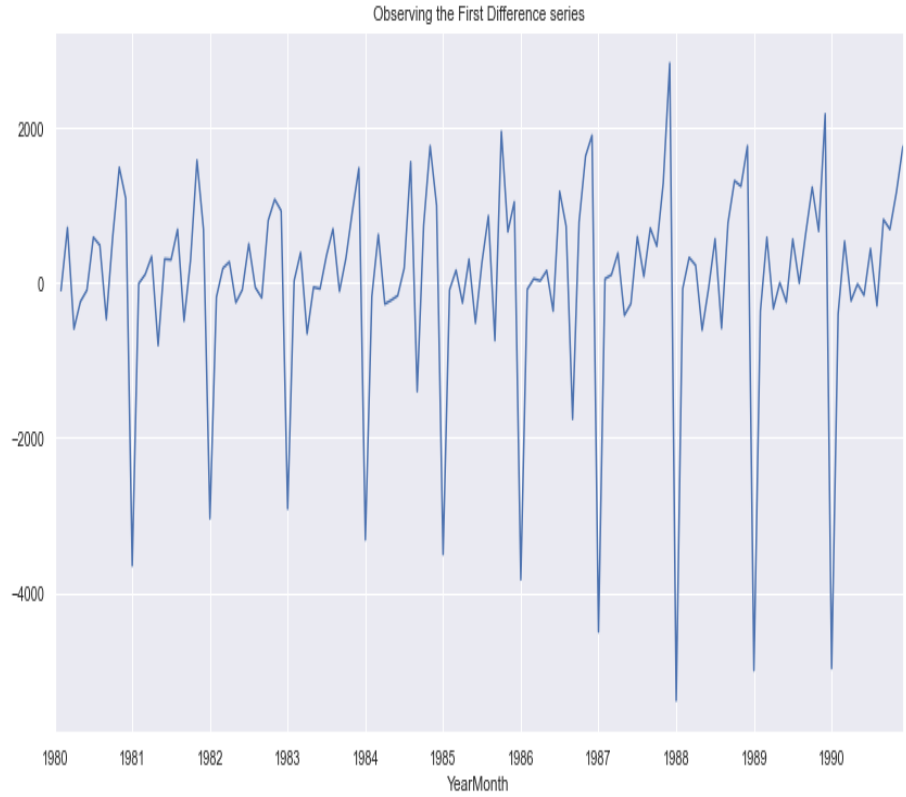

```
Results of Dickey-Fuller Test:
Test Statistic                 -1.208926
p-value                         0.669744
#Lags Used                     12.000000
Number of Observations Used    119.000000
Critical Value (1%)            -3.486535
Critical Value (5%)            -2.886151
Critical Value (10%)           -2.579896
dtype: float64
```

The ADF test gives a p value (0.66 approx.) > than significance level alpha = 0.05, Hence we fail to reject Null Hypothesis as Series is non-stationary.

Calculating the 1st difference:

This step is performed to make the data stationary.

```
train_spark_first_diff.plot()
plt.title('Observing the First Difference series');
```




```
Results of Dickey-Fuller Test:
Test Statistic                -8.005007e+00
p-value                        2.280104e-12
#Lags Used                     1.100000e+01
Number of Observations Used    1.190000e+02
Critical Value (1%)           -3.486535e+00
Critical Value (5%)           -2.886151e+00
Critical Value (10%)          -2.579896e+00
dtype: float64
```

After taking a difference of order 1 the series have become stationary at $\alpha = 0.05$ on Sparkling Wines Data.

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.

ARIMA MODEL:

Values of p and q parameter will be varied but the d parameter in ARIMA model will be fixed at 1 as we need the first difference series in order to make the data stationary.

Parameter Combinations for the ARIMA Model:

Rose Wine Data

	param	AIC
0	(3, 1, 3)	1273.194094
1	(0, 1, 2)	1276.835372
2	(1, 1, 2)	1277.359227
3	(1, 1, 1)	1277.775756
4	(0, 1, 3)	1278.074261
5	(2, 1, 1)	1279.045689
6	(2, 1, 2)	1279.298694
7	(1, 1, 3)	1279.312633
8	(3, 1, 1)	1279.605966
9	(0, 1, 1)	1280.726183

Sparkling Wine Data

	param	AIC
0	(2, 1, 2)	2210.619379
1	(3, 1, 3)	2225.661559
2	(4, 1, 3)	2226.954554
3	(2, 1, 3)	2227.558862
4	(3, 1, 2)	2228.927868
5	(4, 1, 2)	2230.896797
6	(2, 1, 1)	2232.360490
7	(0, 1, 2)	2232.783098
8	(0, 1, 3)	2233.016605
9	(1, 1, 2)	2233.597647

- *For Rose Wines Data, the ARIMA model the minimum AIC value is observed for ARIMA (3,1,3) model.*
- *For Sparkling Wines Data, the ARIMA model the minimum AIC value is observed for ARIMA (2,1,2) model.*

ARIMA MODEL RESULTS:

Rose Wine Data:

ARIMA Model Results

Dep. Variable:	D.Rose	No. Observations:	131
Model:	ARIMA(3, 1, 3)	Log Likelihood	-628.597
Method:	css-mle	S.D. of innovations	28.356
Date:	Sun, 25 Jul 2021	AIC	1273.194
Time:	12:22:47	BIC	1296.196
Sample:	02-01-1980 - 12-01-1990	HQIC	1282.541

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.4906	0.088	-5.547	0.000	-0.664	-0.317
ar.L1.D.Rose	-0.7243	0.086	-8.406	0.000	-0.893	-0.555
ar.L2.D.Rose	-0.7217	0.087	-8.337	0.000	-0.891	-0.552
ar.L3.D.Rose	0.2764	0.085	3.233	0.001	0.109	0.444
ma.L1.D.Rose	-0.0151	0.045	-0.340	0.734	-0.102	0.072
ma.L2.D.Rose	0.0151	0.044	0.341	0.733	-0.072	0.102
ma.L3.D.Rose	-1.0000	0.046	-21.887	0.000	-1.090	-0.910

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	-0.5011	-0.8661j	1.0006	-0.3335
AR.2	-0.5011	+0.8661j	1.0006	0.3335
AR.3	3.6136	-0.0000j	3.6136	-0.0000
MA.1	1.0000	-0.0000j	1.0000	-0.0000
MA.2	-0.4924	-0.8704j	1.0000	-0.3319
MA.3	-0.4924	+0.8704j	1.0000	0.3319

Sparkling Wine Data:

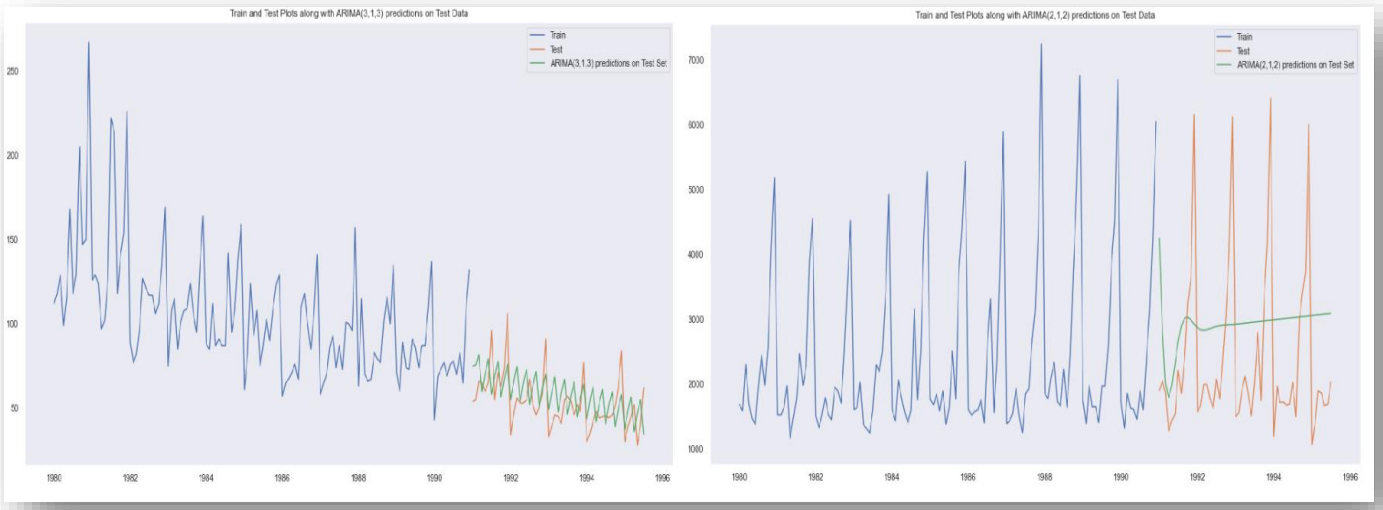
ARIMA Model Results

Dep. Variable:	D.Sparkling	No. Observations:	131
Model:	ARIMA(2, 1, 2)	Log Likelihood	-1099.310
Method:	css-mle	S.D. of innovations	1012.813
Date:	Sun, 25 Jul 2021	AIC	2210.619
Time:	15:27:47	BIC	2227.871
Sample:	02-01-1980 - 12-01-1990	HQIC	2217.629

	coef	std err	z	P>|z|	[0.025	0.975]
const	5.5852	0.517	10.799	0.000	4.571	6.599
ar.L1.D.Sparkling	1.2699	0.075	17.045	0.000	1.124	1.416
ar.L2.D.Sparkling	-0.5601	0.074	-7.617	0.000	-0.704	-0.416
ma.L1.D.Sparkling	-1.9976	0.042	-47.123	0.000	-2.081	-1.915
ma.L2.D.Sparkling	0.9976	0.042	23.510	0.000	0.914	1.081

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.1336	-0.7074j	1.3362	-0.0888
AR.2	1.1336	+0.7074j	1.3362	0.0888
MA.1	1.0001	+0.0000j	1.0001	0.0000
MA.2	1.0023	+0.0000j	1.0023	0.0000



ARIMA Predictions on Test Set - Rose Wines Data

ARIMA Predictions on Test Set Sparkling Wines Data

MODEL EVALUATION:

Forecast on the Test Data for ARIMA (3,1,3) Model on the Rose Test Data, RMSE is:

```
rmse = np.sqrt(mean_squared_error(test_rose,predict_test))
print("ARIMA(3,1,3) Model forecast on the Test Data,  RMSE is %3.3f" %rmse)
```

```
ARIMA(3,1,3) Model forecast on the Test Data,  RMSE is 15.998
```

Forecast on the Test Data for ARIMA (2,1,2) Model on the Sparkling Test Data, RMSE is:

```
rmse = np.sqrt(mean_squared_error(test_spark,predict_test))
print("ARIMA(2,1,2) Model forecast on the Test Data,  RMSE is %3.3f" %rmse)
```

```
ARIMA(2,1,2) Model forecast on the Test Data,  RMSE is 1374.549
```

SARIMA MODEL:

Parameter Seasonal Combinations for the SARIMA Model:

Rose Wine Data:

	param	seasonal	AIC
0	(0, 1, 2)	(2, 1, 2, 12)	774.969120
1	(1, 1, 2)	(2, 1, 2, 12)	776.940108
2	(2, 1, 2)	(2, 1, 2, 12)	776.996101
3	(0, 1, 1)	(2, 1, 2, 12)	782.153872
4	(2, 1, 2)	(2, 1, 1, 12)	783.703652
5	(1, 1, 1)	(2, 1, 2, 12)	783.899095
6	(0, 1, 2)	(0, 1, 2, 12)	784.014096
7	(2, 1, 2)	(0, 1, 2, 12)	784.140979
8	(2, 1, 1)	(2, 1, 1, 12)	784.892805
9	(0, 1, 2)	(1, 1, 2, 12)	785.823714

Sparkling Wine Data

	param	seasonal	AIC
0	(1, 1, 2)	(0, 1, 2, 12)	1382.347780
1	(0, 1, 2)	(0, 1, 2, 12)	1382.484254
2	(1, 1, 2)	(1, 1, 2, 12)	1384.137874
3	(2, 1, 2)	(0, 1, 2, 12)	1384.317618
4	(0, 1, 2)	(1, 1, 2, 12)	1384.398867
5	(1, 1, 2)	(2, 1, 2, 12)	1385.688721
6	(0, 1, 2)	(2, 1, 2, 12)	1386.023734
7	(2, 1, 2)	(1, 1, 2, 12)	1386.097242
8	(2, 1, 2)	(2, 1, 2, 12)	1387.627785
9	(1, 1, 1)	(0, 1, 2, 12)	1398.756167

SARIMAX RESULTS:

Rose Wine Data:

SARIMAX Results

Dep. Variable:	Rose	No. Observations:	132
Model:	SARIMAX(0, 1, 2)x(2, 1, 2, 12)	Log Likelihood	-380.485
Date:	Sun, 25 Jul 2021	AIC	774.969
Time:	12:25:27	BIC	792.622
Sample:	01-01-1980	HQIC	782.094
	- 12-01-1990		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ma.L1	-0.9524	0.184	-5.167	0.000	-1.314	-0.591
ma.L2	-0.0764	0.126	-0.605	0.545	-0.324	0.171
ar.S.L12	0.0481	0.177	0.272	0.786	-0.298	0.395
ar.S.L24	-0.0419	0.028	-1.513	0.130	-0.096	0.012
ma.S.L12	-0.7527	0.301	-2.504	0.012	-1.342	-0.163
ma.S.L24	-0.0720	0.204	-0.353	0.724	-0.471	0.327
sigma2	187.8484	45.269	4.150	0.000	99.122	276.575

Ljung-Box (L1) (Q):	0.06	Jarque-Bera (JB):	4.86
Prob(Q):	0.81	Prob(JB):	0.09
Heteroskedasticity (H):	0.91	Skew:	0.41
Prob(H) (two-sided):	0.79	Kurtosis:	3.77

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

Sparkling Wine Data:

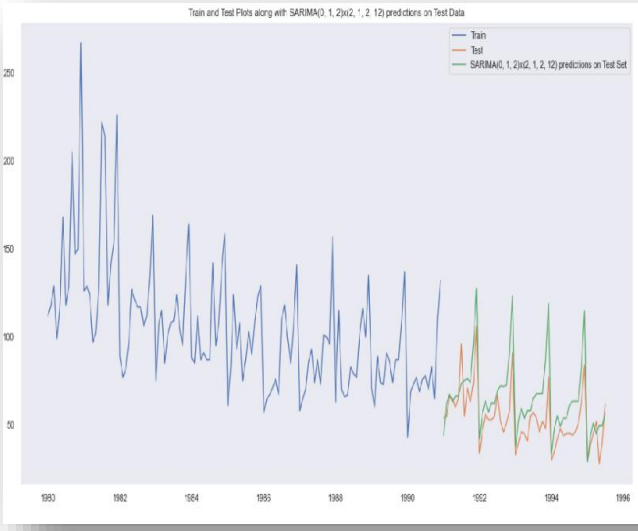
SARIMAX Results

Dep. Variable:	Sparkling	No. Observations:	132
Model:	SARIMAX(1, 1, 2)x(0, 1, 2, 12)	Log Likelihood	-685.174
Date:	Sun, 25 Jul 2021	AIC	1382.348
Time:	15:30:06	BIC	1397.479
Sample:	01-01-1980	HQIC	1388.455
	- 12-01-1990		
Covariance Type:	opg		

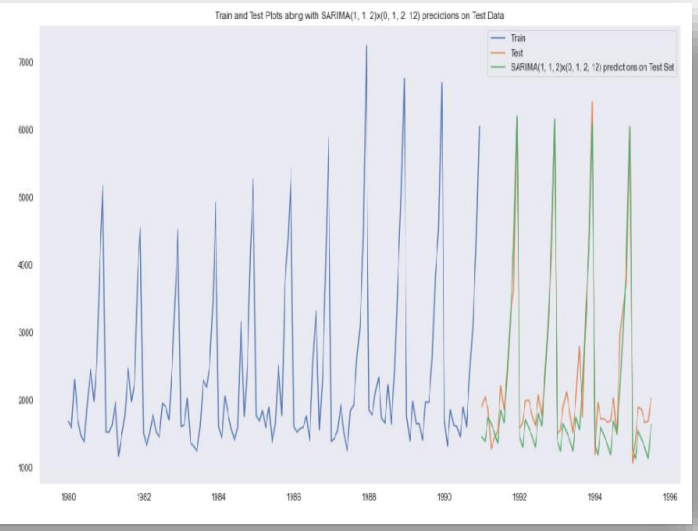
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.5507	0.287	-1.922	0.055	-1.112	0.011
ma.L1	-0.1612	0.235	-0.687	0.492	-0.621	0.299
ma.L2	-0.7218	0.175	-4.132	0.000	-1.064	-0.379
ma.S.L12	-0.4062	0.092	-4.401	0.000	-0.587	-0.225
ma.S.L24	-0.0274	0.138	-0.198	0.843	-0.298	0.243
sigma2	1.705e+05	2.45e+04	6.956	0.000	1.22e+05	2.19e+05

Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	13.48
Prob(Q):	0.95	Prob(JB):	0.00
Heteroskedasticity (H):	0.89	Skew:	0.60
Prob(H) (two-sided):	0.75	Kurtosis:	4.44

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



Plot: SARIMA Predictions on Test Set - Rose Wines Data



Plot : SARIMA Predictions on Test Set - Sparkling Wines Data

MODEL EVALUATION:

Forecast on the Test Data for SARIMA (0,1,2) * (2,1,2,12) Model on the Rose Test Data, RMSE is

```
rmse = np.sqrt(mean_squared_error(test_rose,predict_test))
print("SARIMA(0, 1, 2)x(2, 1, 2, 12) Model forecast on the Test Data,  RMSE is %3.3f" %rmse)
```

```
SARIMA(0, 1, 2)x(2, 1, 2, 12) Model forecast on the Test Data,  RMSE is 16.529
```

Forecast on the Test Data for SARIMA (1,1,2) * (0,1,2,12) Model on the Rose Test Data, RMSE is

```
rmse = np.sqrt(mean_squared_error(test_spark,predict_test))
print("SARIMA(0, 1, 2)x(2, 1, 2, 12) Model forecast on the Test Data,  RMSE is %3.3f" %rmse)
```

```
SARIMA(0, 1, 2)x(2, 1, 2, 12) Model forecast on the Test Data,  RMSE is 382.577
```

Inferences from ARIMA and SARIMA Automated models:

	Test RMSE
ARIMA(3,1,3) Automated AIC	15.997918
SARIMA(0, 1, 2)x(2, 1, 2, 12) Automated AIC	16.528855

	Test RMSE
ARIMA(2,1,2) Automated AIC	1374.549422
SARIMA(0, 1, 2)x(2, 1, 2, 12) Automated AIC	382.576709

- *Rose Wine Data: The ARIMA and SARIMA models are selected based on the AIC on Train Data. It is observed that ARIMA (3,1,3) and SARIMA (0,1,2) x (2,1,2,12) have the minimum AIC values. The Test RMSE for ARIMA (3,1,3) is found to be 15.99 and for SARIMA (0,1,2) x (2,1,2,12) 16.52 approximately.*
- *Sparkling Wine Data: The ARIMA and SARIMA models are selected based on the AIC on Train Data. It is observed that ARIMA (2,1,2) and SARIMA (1,1,2) x (0,1,2,12) have the minimum AIC values. The Test RMSE for ARIMA (2,1,2) is found to be 1374.54 and for SARIMA(1,1,2) x(0,1,2,12) 382.57 approximately.*

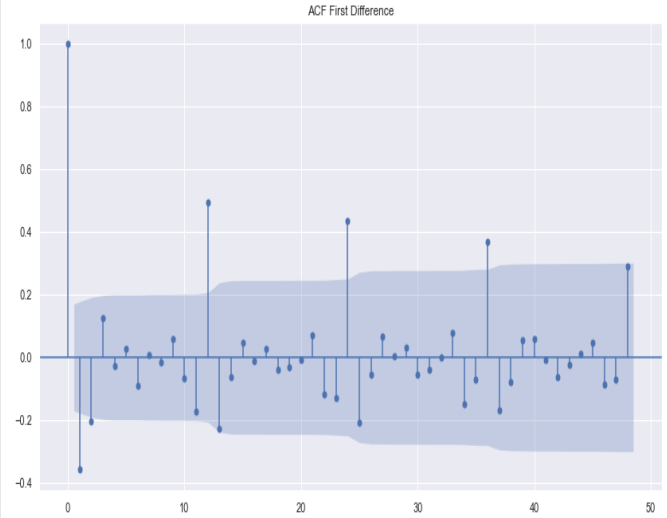
7. Build ARIMA/SARIMA models based on the cut-off points of ACF and PACF on the training data and evaluate this model on the test data using RMSE.

ACF Plot : First Difference

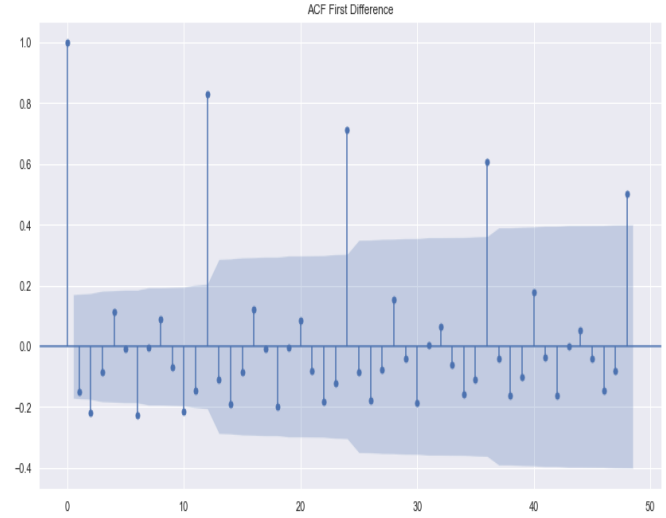
ACF : Rose Wine

ACF : Sparkling Wine

```
plot_acf(train_first_diff.dropna(),lags=48, alpha=0.05)
plt.title('ACF First Difference');
```

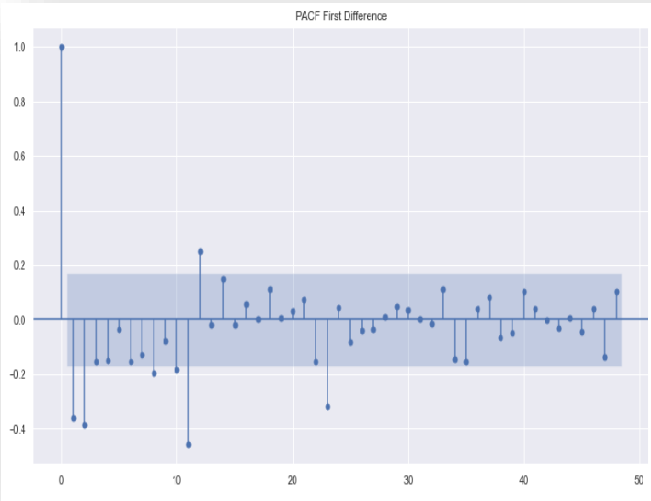



```
plot_acf(train_first_diff.dropna(),lags=48, alpha=0.05)
plt.title('ACF First Difference');
```

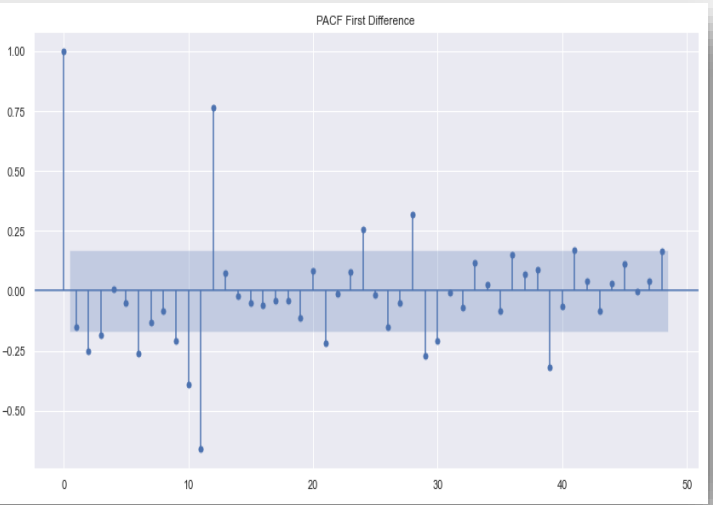


PACF Plot : First Difference

PACF : Rose Wine



PACF : Sparkling Wine



ARIMA MODEL MANUAL

```
mod = ARIMA(train_rose, order=(2,1,2), freq=train_rose.index.inferred_freq)

results_Arima = mod.fit()

print(results_Arima.summary())
```

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:                D.Rose   No. Observations:                  131
Model:                 ARIMA(2, 1, 2)  Log Likelihood                -633.649
Method:                       css-mle  S.D. of innovations             29.975
Date:                Sun, 25 Jul 2021  AIC                           1279.299
Time:                        12:25:29  BIC                           1296.550
Sample:                  02-01-1980    HQIC                          1286.309
                       - 12-01-1990
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.4911      0.081     -6.076      0.000      -0.649      -0.333
ar.L1.D.Rose  -0.4383      0.218     -2.015      0.044      -0.865      -0.012
ar.L2.D.Rose   0.0269      0.109      0.246      0.806      -0.188       0.241
ma.L1.D.Rose  -0.3316      0.203     -1.633      0.102      -0.729       0.066
ma.L2.D.Rose  -0.6684      0.201     -3.332      0.001      -1.062      -0.275
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           -2.0290           +0.0000j            2.0290            0.5000
AR.2           18.3390           +0.0000j           18.3390            0.0000
MA.1            1.0000           +0.0000j            1.0000            0.0000
MA.2           -1.4961           +0.0000j            1.4961            0.5000
-----------------------------------------------------------------------------
```

ARIMA Results : Rose Wine

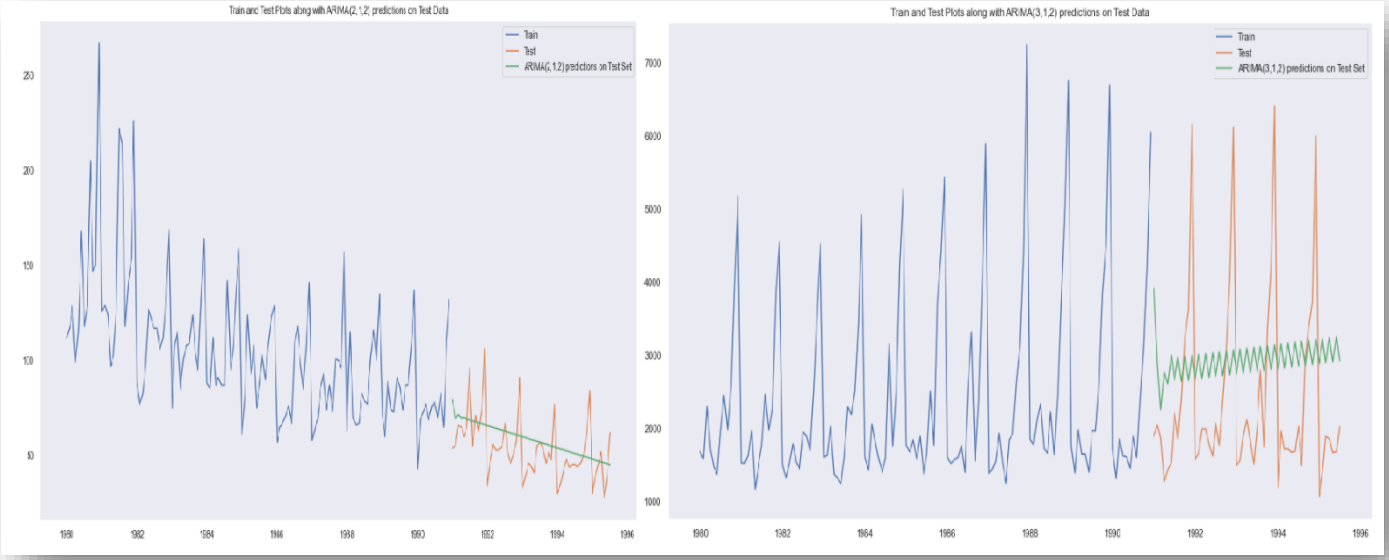
```
mod = ARIMA(train_spark, order=(3,1,2), freq=train_spark.index.inferred_freq)

results_Arima = mod.fit()

print(results_Arima.summary())
```

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:            D.Sparkling   No. Observations:                  131
Model:                 ARIMA(3, 1, 2)  Log Likelihood               -1107.464
Method:                       css-mle  S.D. of innovations           1106.230
Date:                Sun, 25 Jul 2021  AIC                           2228.928
Time:                        20:15:32  BIC                           2249.054
Sample:                  02-01-1980    HQIC                          2237.106
                       - 12-01-1990
==============================================================================
                    coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const              5.9846      3.643      1.643      0.100      -1.156      13.125
ar.L1.D.Sparkling -0.4419   1.21e-05  -3.64e+04      0.000      -0.442      -0.442
ar.L2.D.Sparkling  0.3079   4.36e-05   7059.190      0.000       0.308       0.308
ar.L3.D.Sparkling -0.2501    3.6e-05  -6940.595      0.000      -0.250      -0.250
ma.L1.D.Sparkling -0.0008      0.020     -0.039      0.969      -0.039       0.038
ma.L2.D.Sparkling -0.9992      0.020    -51.147      0.000      -1.038      -0.961
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           -1.0000           -0.0000j            1.0000           -0.5000
AR.2            1.1155           -1.6594j            1.9995           -0.1558
AR.3            1.1155           +1.6594j            1.9995            0.1558
MA.1            1.0000           +0.0000j            1.0000            0.0000
MA.2           -1.0008           +0.0000j            1.0008            0.5000
-----------------------------------------------------------------------------
```

ARIMA Results : Sparkling Wine



Plot :ARIMA Predictions on Test Set - Rose Wines Data

Plot : ARIMA Predictions on Test Set - Sparkling Wines Data

Since the series has seasonality, we observe that the ARIMA model build on it is not appropriate and we need to build a SARIMA model and derive the p,q parameters from the acf and pacf plots after considering the seasonal difference as well.

MODEL EVALUATION:

Forecast based on the Cut-Off Points for ARIMA (2,1,2) the Rose Test Data, RMSE is

```
rmse = np.sqrt(mean_squared_error(test_rose,predict_test))
print("ARIMA(2,1,2) Model forecast on the Test Data,  RMSE is %3.3f" %rmse)
```

```
ARIMA(2,1,2) Model forecast on the Test Data,  RMSE is 15.364
```

Forecast based on the Cut-Off Points for ARIMA (3,1,2) the Sparkling Test Data, RMSE is

```
rmse = np.sqrt(mean_squared_error(test_spark,predict_test))
print("ARIMA(3,1,2) Model forecast on the Test Data,  RMSE is %3.3f" %rmse)
```

```
ARIMA(3,1,2) Model forecast on the Test Data,  RMSE is 1379.052
```

Rose Wine

```
print('The p value of the Dickey-Fuller Test after First Differencing and First Seasonal Shift:',
      adfuller(train_first_seasonal_diff.dropna(), autolag='AIC')[1])
```

```
The p value of the Dickey-Fuller Test after First Differencing and First Seasonal Shift: 0.004222089763985825
```

- *The p value suggests that the time series after first difference and then first seasonal difference is stationary. Thus, we can consider the D value equal to 1.*

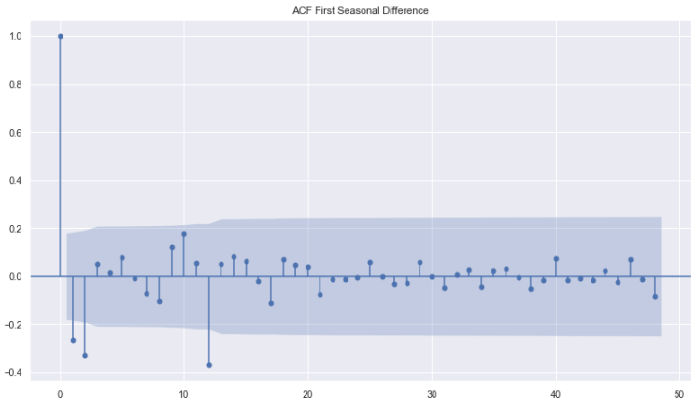
Sparkling Wine:

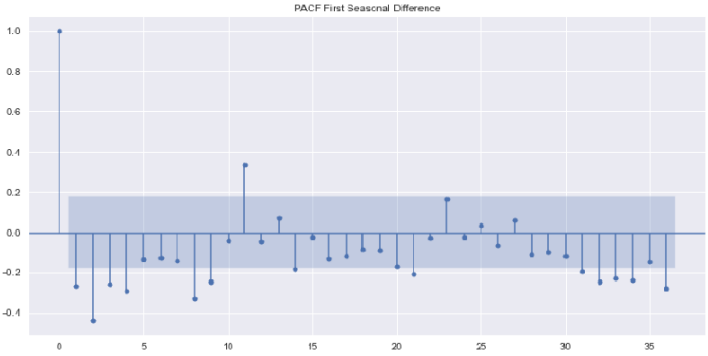
```
print('The p value of the Dickey-Fuller Test after First Differencing and First Seasonal Shift:',
      adfuller(train_first_seasonal_diff.dropna(), autolag='AIC')[1])
```

```
The p value of the Dickey-Fuller Test after First Differencing and First Seasonal Shift: 0.013066196701392049
```

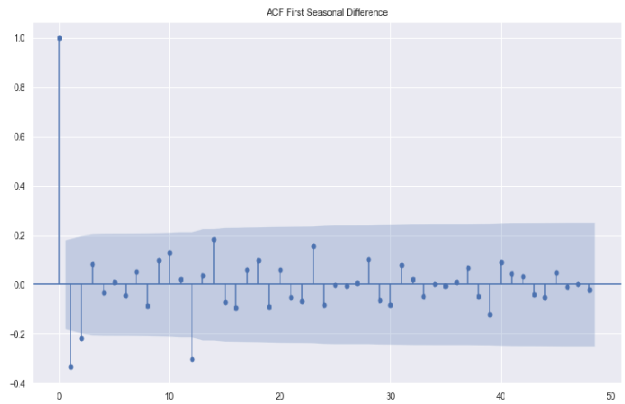
ACF & PACF Seasonal Difference:

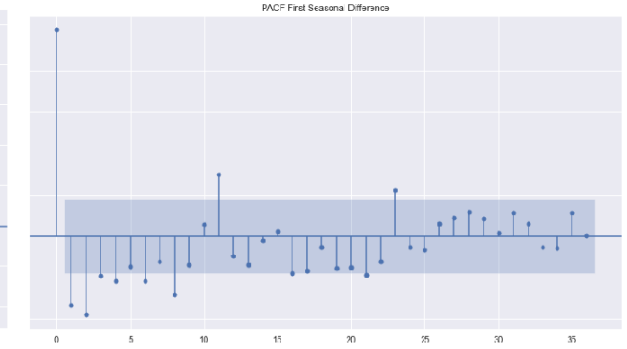
ACF & PACF: Rose Wine





ACF & PACF: Sparkling Wine





Inference:

- *When we used only first difference, we observed in the seasonal lags were strongly present suggesting seasonality. Now after first seasonal difference such tailing seasonal lags are not observed.*

SARIMA MODE MANUAL

```
mod = sm.tsa.statespace.SARIMAX(train_rose, freq=train_rose.index.inferred_freq,
                                order=(4,1,2),
                                seasonal_order=(1,1,1,12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)
results_SARIMA = mod.fit(maxiter=1000)
print(results_SARIMA.summary())
```

SARIMAX Results

Dep. Variable:	Rose	No. Observations:	132
Model:	SARIMAX(4, 1, 2)x(1, 1, [1], 12)	Log Likelihood	-443.798
Date:	Sun, 25 Jul 2021	AIC	905.596
Time:	12:25:32	BIC	929.308
Sample:	01-01-1980	HQIC	915.200
	- 12-01-1990		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.8065	0.131	-6.169	0.000	-1.063	-0.550
ar.L2	0.0590	0.152	0.389	0.697	-0.238	0.356
ar.L3	-0.2011	0.162	-1.240	0.215	-0.519	0.117
ar.L4	-0.1740	0.112	-1.547	0.122	-0.394	0.046
ma.L1	0.1292	0.151	0.855	0.392	-0.167	0.425
ma.L2	-0.8708	0.141	-6.181	0.000	-1.147	-0.595
ar.S.L12	-0.3660	0.071	-5.171	0.000	-0.505	-0.227
ma.S.L12	-0.0184	0.138	-0.133	0.894	-0.288	0.251
sigma2	313.3873	0.001	4.30e+05	0.000	313.386	313.389

Ljung-Box (L1) (Q):	0.01	Jarque-Bera (JB):	0.41
Prob(Q):	0.92	Prob(JB):	0.81
Heteroskedasticity (H):	0.51	Skew:	0.11
Prob(H) (two-sided):	0.05	Kurtosis:	3.21

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 6.45e+20. Standard errors may be unstable.

SARIMA Results: Rose Wine

```
mod = sm.tsa.statespace.SARIMAX(train_spark, freq=train_spark.index.inferred_freq,
                                order=(4,1,2),
                                seasonal_order=(1,1,1,12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)
results_SARIMA = mod.fit(maxiter=1000)
print(results_SARIMA.summary())
```

SARIMAX Results

Dep. Variable:	Sparkling	No. Observations:	132
Model:	SARIMAX(4, 1, 2)x(1, 1, [1], 12)	Log Likelihood	-765.235
Date:	Sun, 25 Jul 2021	AIC	1548.470
Time:	20:15:36	BIC	1572.183
Sample:	01-01-1980	HQIC	1558.074
	- 12-01-1990		
Covariance Type:	opg		

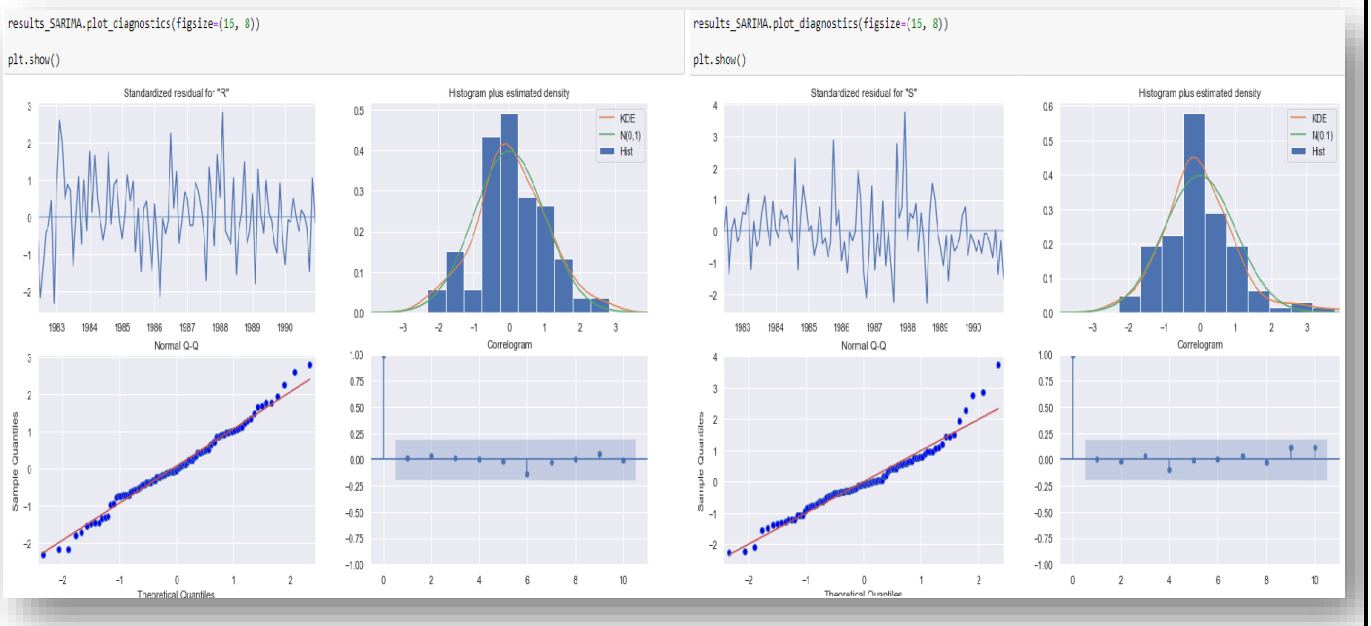
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.7935	0.132	-6.007	0.000	-1.052	-0.535
ar.L2	0.0722	0.156	0.464	0.643	-0.233	0.377
ar.L3	-0.0267	0.131	-0.203	0.839	-0.283	0.230
ar.L4	0.0229	0.112	0.205	0.838	-0.196	0.242
ma.L1	0.0794	0.159	0.499	0.618	-0.233	0.391
ma.L2	-0.9206	0.118	-7.788	0.000	-1.152	-0.689
ar.S.L12	-0.0541	0.227	-0.238	0.812	-0.499	0.391
ma.S.L12	-0.3991	0.237	-1.684	0.092	-0.863	0.065
sigma2	1.61e+05	1.47e-06	1.09e+11	0.000	1.61e+05	1.61e+05

Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	25.35
Prob(Q):	0.97	Prob(JB):	0.00
Heteroskedasticity (H):	1.10	Skew:	0.75
Prob(H) (two-sided):	0.79	Kurtosis:	4.91

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 6.34e+26. Standard errors may be unstable.

SARIMA Results: Sparkling Wine

SARIMAX PLOT: DIAGNOSTICS

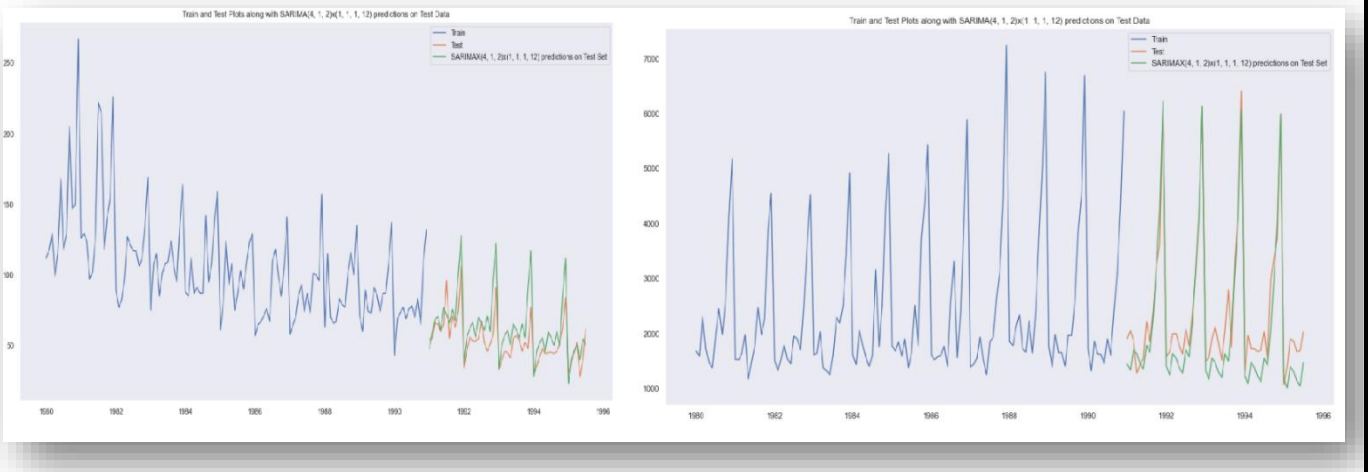


ROSE WINE DIAGNOSTICS: SARIMAX

SPARKLING WINE DIAGNOSTICS: SARIMAX

SARIMA Predictions on Test Set - Rose Wines Data

SARIMA Predictions on Test Set - Sparkling Wines Data



MODEL EVALUATION:

*Forecast on the Test Data for SARIMA (4,1,2) *(1,1,1,12) Model on the Rose Test Data, RMSE is*

```
rmse = np.sqrt(mean_squared_error(test_rose,predict_test))
print("SARIMA(4, 1, 2)x(1, 1, 1, 12) Model forecast on the Test Data,  RMSE is %3.3f" %rmse)
```

```
SARIMA(4, 1, 2)x(1, 1, 1, 12) Model forecast on the Test Data,  RMSE is 15.465
```

*Forecast on the Test Data for SARIMA (4,1,2) *(1,1,1,12) Model on the Sparkling Test Data, RMSE is*

```
rmse = np.sqrt(mean_squared_error(test_spark,predict_test))
print("SARIMA(4, 1, 2)x(1, 1, 1, 12) Model forecast on the Test Data,  RMSE is %3.3f" %rmse)
```

```
SARIMA(4, 1, 2)x(1, 1, 1, 12) Model forecast on the Test Data,  RMSE is 434.921
```

Inference:

- *It is observed from above SARIMA Diagnostics Plot that Residual is nearly Standard Normally distributed and Correlogram shows that there is no correlation in the residuals. These factors suggests that the SARIMA model captures the information and is a good enough model in both Rose Wine & Sparkling Wine.*

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.

RMSE Values: Rose Wine Test Set

```
resultsDf.sort_values(by = 'Test RMSE')
```

	Test RMSE
Alpha=0.05,Beta=0.4,Gamma=0.25, Triple Exponential Smoothing Iterative	8.698606
ARIMA(2,1,2) ACF/PACF	15.363931
SARIMA(4, 1, 2)x(1, 1, 1, 12) ACF/PACF	15.464686
Double Exponential Smoothing Optimized	15.580105
ARIMA(3,1,3) Automated AIC	15.997918
Alpha=0.05, Beta=0.35, Double Exponential Smoothing Iterative	16.343344
SARIMA(0, 1, 2)x(2, 1, 2, 12) Automated AIC	16.528855
Alpha=0.0987, Simple Exponential Smoothing Optimized	36.824478
Regression On Time	51.457439
Simple Average Model	53.488233
Naive Model	79.745697

RMSE Values: Sparkling Wine Test Set

```
resultsDf.sort_values(by = 'Test RMSE')
```

	Test RMSE
Alpha=0.05,Beta=0.4,Gamma=0.25, Triple Exponential Smoothing Iterative	302.454819
SARIMA(1, 1, 2)x(0, 1, 2, 12) Automated AIC	382.576709
SARIMA(4, 1, 2)x(1, 1, 1, 12) ACF/PACF	434.921059
Simple Average Model	1275.081804
Regression On Time	1275.867052
Alpha=0.0496, Simple Exponential Smoothing Optimized	1316.135411
ARIMA(2,1,2) Automated AIC	1374.549422
ARIMA(3,1,2) ACF/PACF	1379.051960
Alpha=0.05, Beta=0.05, Double Exponential Smoothing Iterative	1418.407668
Double Exponential Smoothing Optimized	2007.238526
Naive Model	3864.279352

Inference:

- *Rose Wine Data: The lowest RMSE score is 8.70 approximately for the model Triple Exponential Smoothing Iterative Model with parameters Alpha=0.05, Beta=0.4 and Gamma=0.25 respectively.*
- *Sparkling Wine Data: The lowest RMSE score is 302.45 approximately for the model Triple Exponential Smoothing Iterative Model with parameters Alpha=0.05, Beta=0.4 and Gamma=0.25 respectively.*

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.

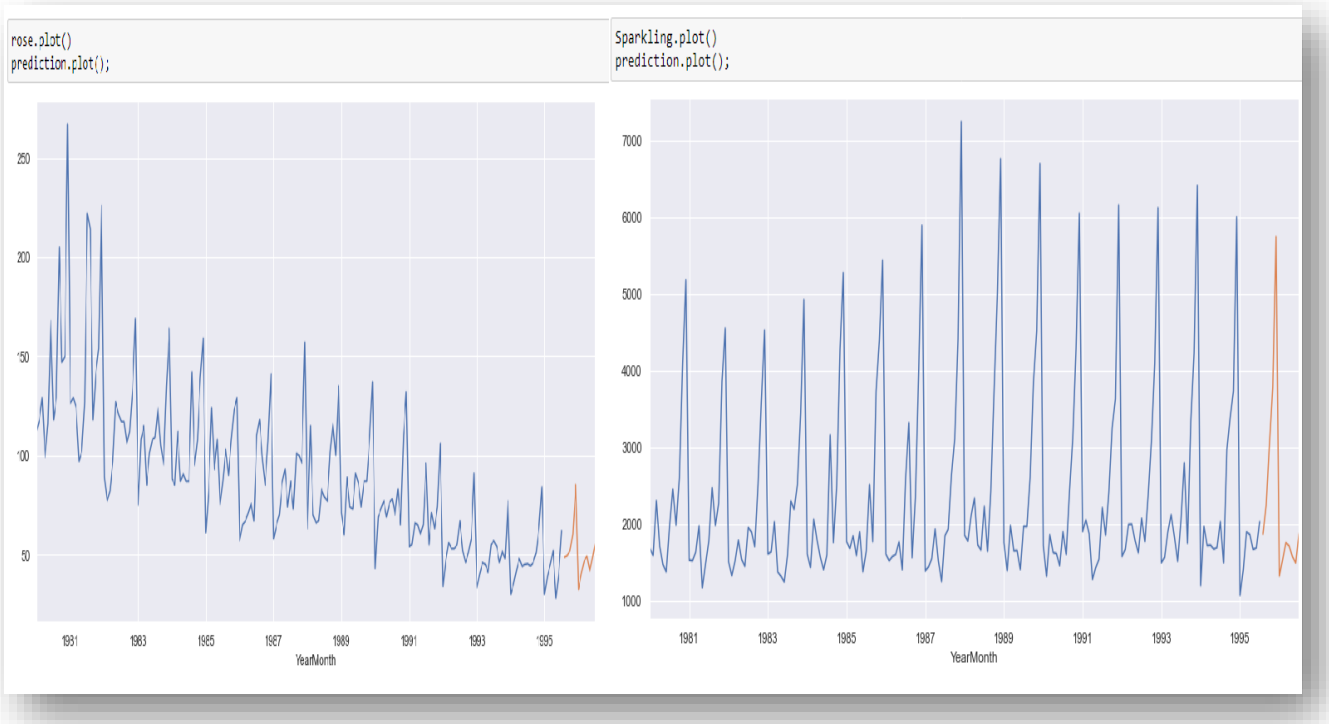
We will consider the entire data to build a model using the Triple Smoothing model with the mentioned parameters and then predict 12 months into the future (i.e. 12-time stamps ahead) with a confidence interval of 95%.

```
fullmodel = ExponentialSmoothing(rose,
                                 trend='additive',
                                 seasonal='multiplicative').fit(smoothing_level=0.05,
                                                                smoothing_slope=0.4,
                                                                smoothing_seasonal=0.25)
```

TIME SERIES PLOT:

Time Series Plot: Rose Wine

Time Series Plot: Sparkling Wine



PREDICTED SUMMARY:

Rose Wine

pred_rose

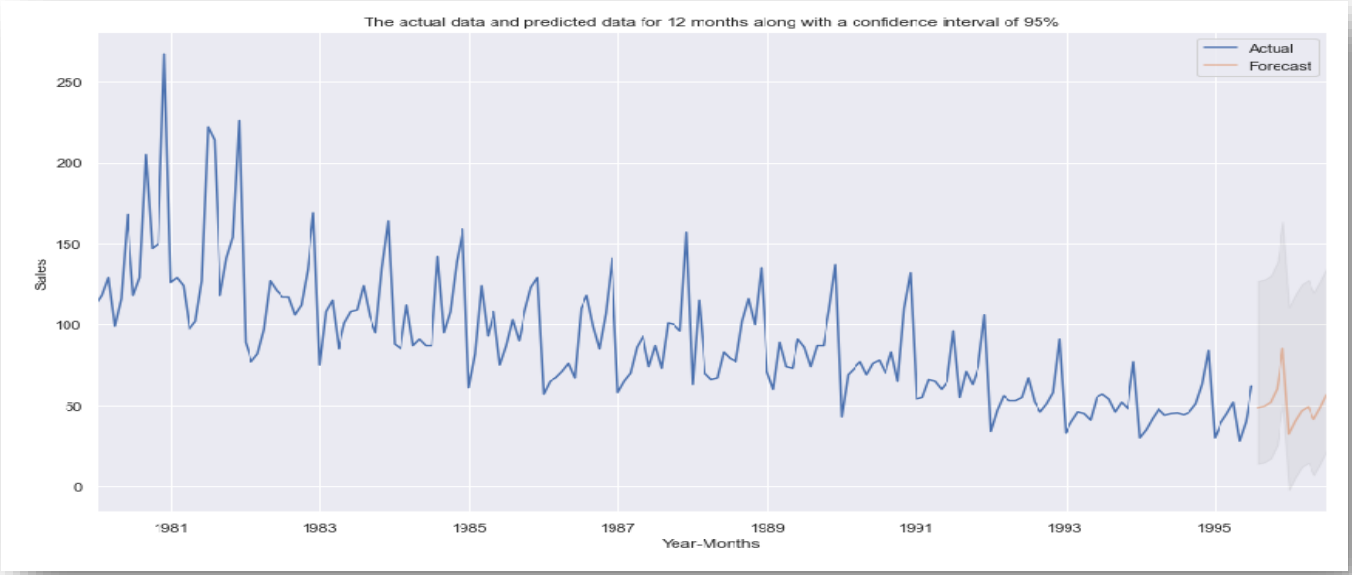
	lower_CI	prediction	upper_ci
1995-08-01	14.001066	48.580956	126.924756
1995-09-01	14.892407	49.472297	127.816097
1995-10-01	17.353016	51.932905	130.276705
1995-11-01	25.644487	60.224377	138.568177
1995-12-01	50.636795	85.216685	163.560485
1996-01-01	-2.192582	32.387307	110.731107
1996-02-01	5.780444	40.360334	118.704134
1996-03-01	12.092063	46.671952	125.015752
1996-04-01	14.580107	49.159997	127.503797
1996-05-01	6.879452	41.459342	119.803142
1996-06-01	13.968421	48.548310	126.892110
1996-07-01	22.069519	56.649409	134.993209

Sparkling Wine

pred_spark

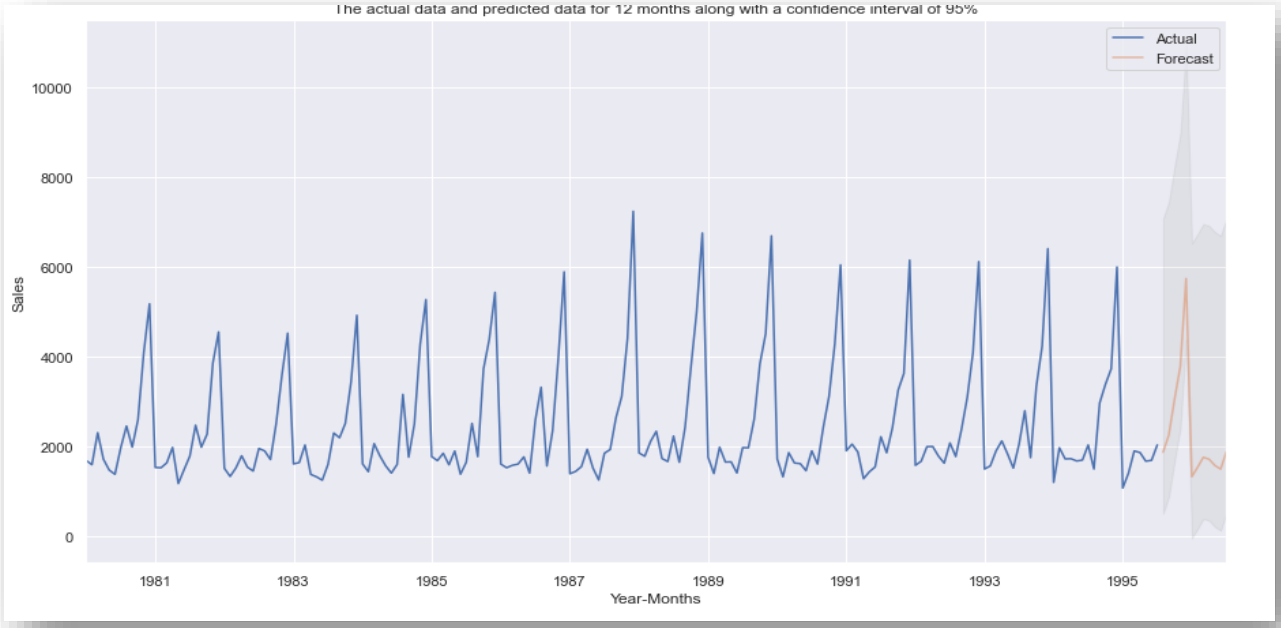
	lower_CI	prediction	upper_ci
1995-08-01	500.019166	1869.943161	7073.819599
1995-09-01	879.932145	2249.856140	7453.732578
1995-10-01	1681.309958	3051.233953	8255.110391
1995-11-01	2411.742094	3781.666089	8985.542527
1995-12-01	4373.411541	5743.335536	10947.211974
1996-01-01	-46.396255	1323.527740	6527.404178
1996-02-01	163.282019	1533.206014	6737.082452
1996-03-01	386.546550	1756.470545	6960.346983
1996-04-01	342.608027	1712.532022	6916.408460
1996-05-01	204.944085	1574.868080	6778.744518
1996-06-01	122.317400	1492.241395	6696.117833
1996-07-01	504.946192	1874.870187	7078.746625

PLOTTING THE FORECAST ALONG WITH CONFIDENCE INTERVAL OF 95%:



PREDICTED ROSE WINES DATA SALES

:



PREDICTED SPARKLING WINES DATA SALES

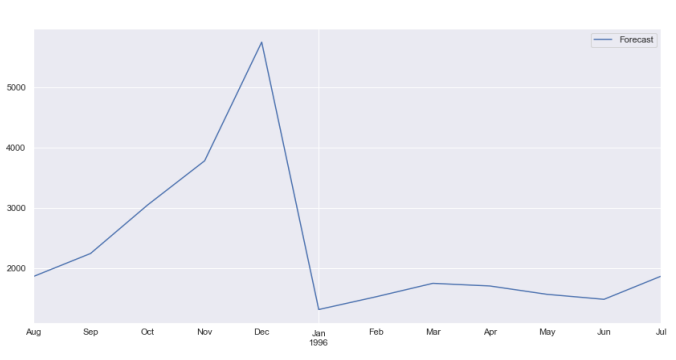
10. Comment on the model thus built and report your findings and suggest the measures that the company should be taking for future sales.

The entire data is used to build a model using the Triple Smoothing model with the mentioned parameters and then 12 months Sales data into the future (12 time stamps ahead) with a confidence interval of 95% is predicted.

TIME SERIES PREDICTED SALES USING TRIPLE EXPONENTIAL SMOOTHING MODEL:

Rose Wine Predicted Sales Plot

Sparkling Wine Predicted Sales Plot



For Rose Wine: *For Sparkling Wine*

prediction		prediction	
1995-08-01	48.580956	1995-08-01	1869.943161
1995-09-01	49.472297	1995-09-01	2249.856140
1995-10-01	51.932905	1995-10-01	3051.233953
1995-11-01	60.224377	1995-11-01	3781.666089
1995-12-01	85.216685	1995-12-01	5743.335536
1996-01-01	32.387307	1996-01-01	1323.527740
1996-02-01	40.360334	1996-02-01	1533.206014
1996-03-01	46.671952	1996-03-01	1756.470545
1996-04-01	49.159997	1996-04-01	1712.532022
1996-05-01	41.459342	1996-05-01	1574.868080
1996-06-01	48.548310	1996-06-01	1492.241395
1996-07-01	56.649409	1996-07-01	1874.870187
Freq: MS, dtype: float64		Freq: MS, dtype: float64	

- *Rose Wine Prediction: Forecast of the upcoming 12 months from the model suggest that the highest sales are expected for the month of December (approx. 85) and lowest are expected for the month of January (approx. 30).*
- *Sparkling Wine Prediction: Forecast of the upcoming 12 months from the model suggest that the highest sales are expected for the month of December (approx. 6000) and lowest are expected for the month of January (approx. 500).*

Recommendations:

- *For the last 15 years a downward trend in the sales of Rose wine is observed. Hence the reason behind such a drastic downward trend must be identified. This could because of external factors like increase in competitors, better markets, Prices/Promotions etc. Further the decreasing trend could be because of internal factors like Wine taste quality, Brand name, Supply chain mismanagement etc.*
- *Sales is in a continuous downward trend for Sparkling Wine from the month of Jan to July. During these months additional campaigns could be done to acquire sales in order to avoid deficits in sales over Last Year.*
- *There is a peak in sales especially for the month of December for both the Rose & Sparkling Wines. This opportunity of higher sales in this month needs to be exploited by providing additional marketing supports, Promotional drives to boost the branding to recreate the Sales Opportunity observed in the Past.*