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

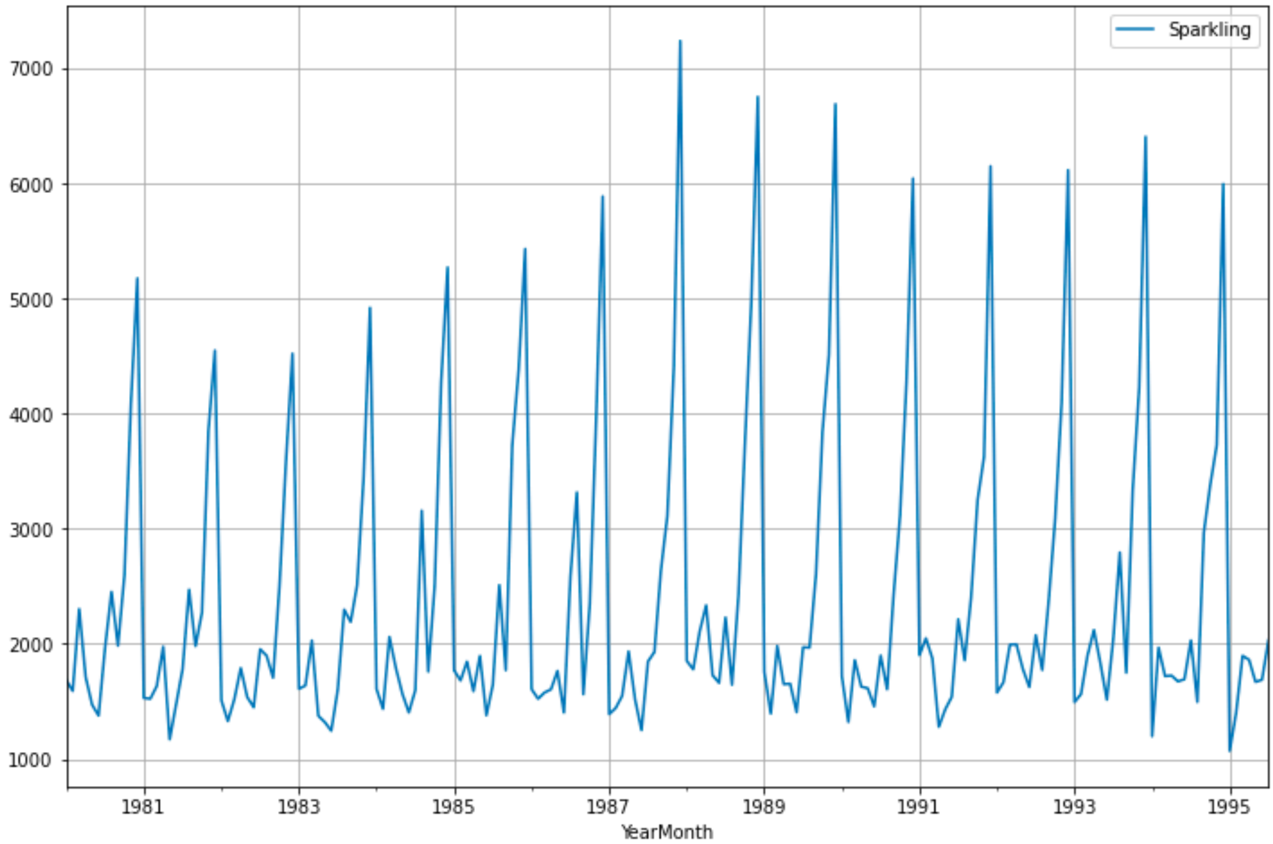
1 Read the data as an appropriate Time Series data and plot the data.

— Reading Rose Dataset. Checking the data frame -

YearMonth	Sparkling
1980-01-01	1686
1980-02-01	1591
1980-03-01	2304
1980-04-01	1712
1980-05-01	1471
...	...
1995-03-01	1897
1995-04-01	1862
1995-05-01	1670
1995-06-01	1688
1995-07-01	2031

187 rows × 1 columns

As we can see the Sparkling wine sales data is from the year 1980 to 1995



<u>– Plotting time series for sparkling data set-</u>

From the above figure we can state:
1. There is a some trend in the data.
2. There seems to be seasonality.

<u>2) Perform appropriate Exploratory Data Analysis to understand the data and also perform decomposition.</u>

– df.shape

(187 , 1)

There are 187 rows in our dataset.

– df.isnull().sum()
Sparkling   0
dtype: int64

There are no missing value in sparkling data set.

– df.info()

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

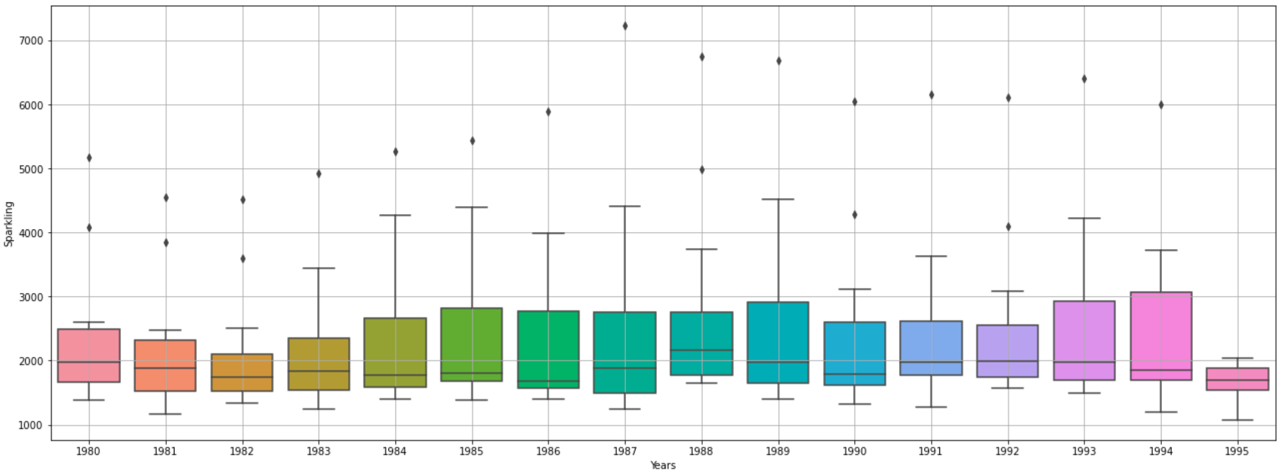
– df.describe()

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

1. The max number of sales in Rose is 7242.
2. The average sales here for rose is 2402.

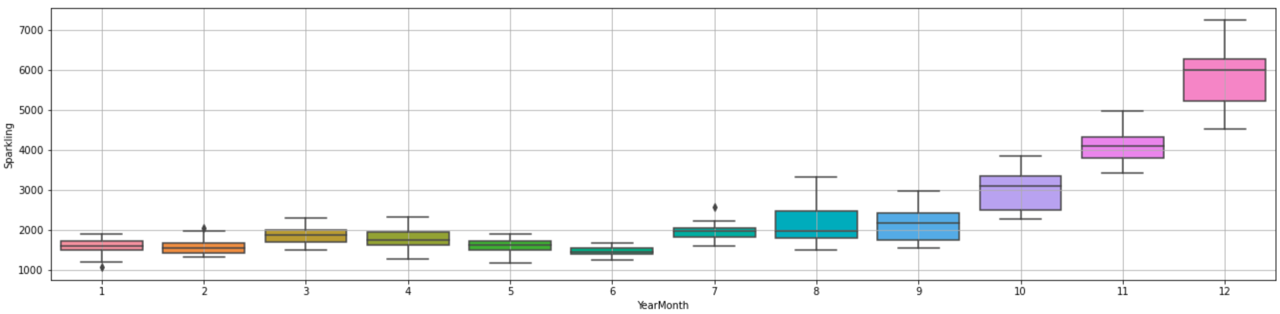
– Plotting a year on year boxplot for the Rose wine production.

Now, let us plot a box and whisker (1.5* IQR) plot to understand the spread of the data and check for outliers in each year, if any.



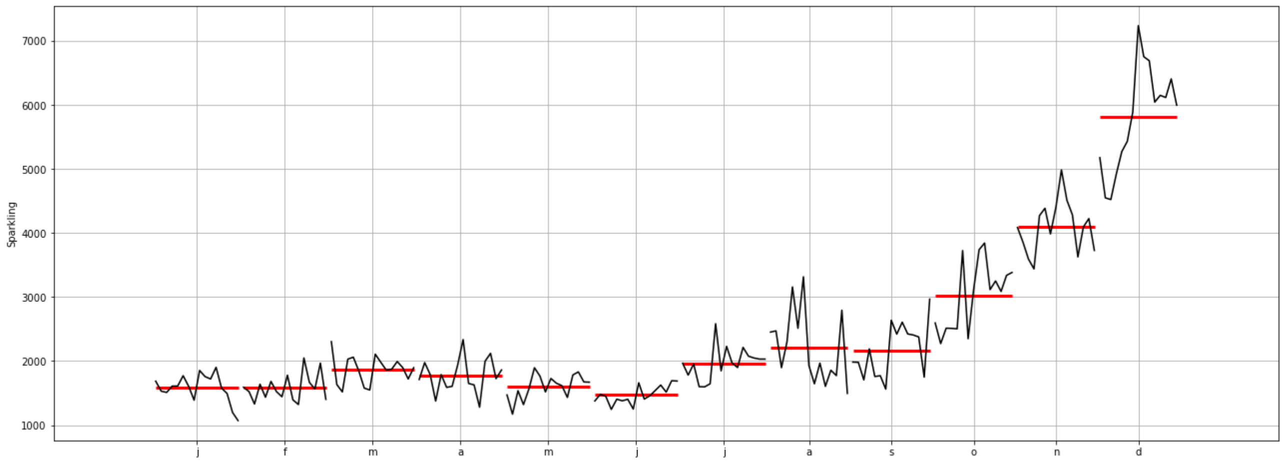
As we got to know from the Time Series plot, the boxplots over here also indicates a measure of trend being present. Also, we see that the sales of wine has some outliers for certain years.

– Plot a monthly boxplot for the sales taking all the years into account.



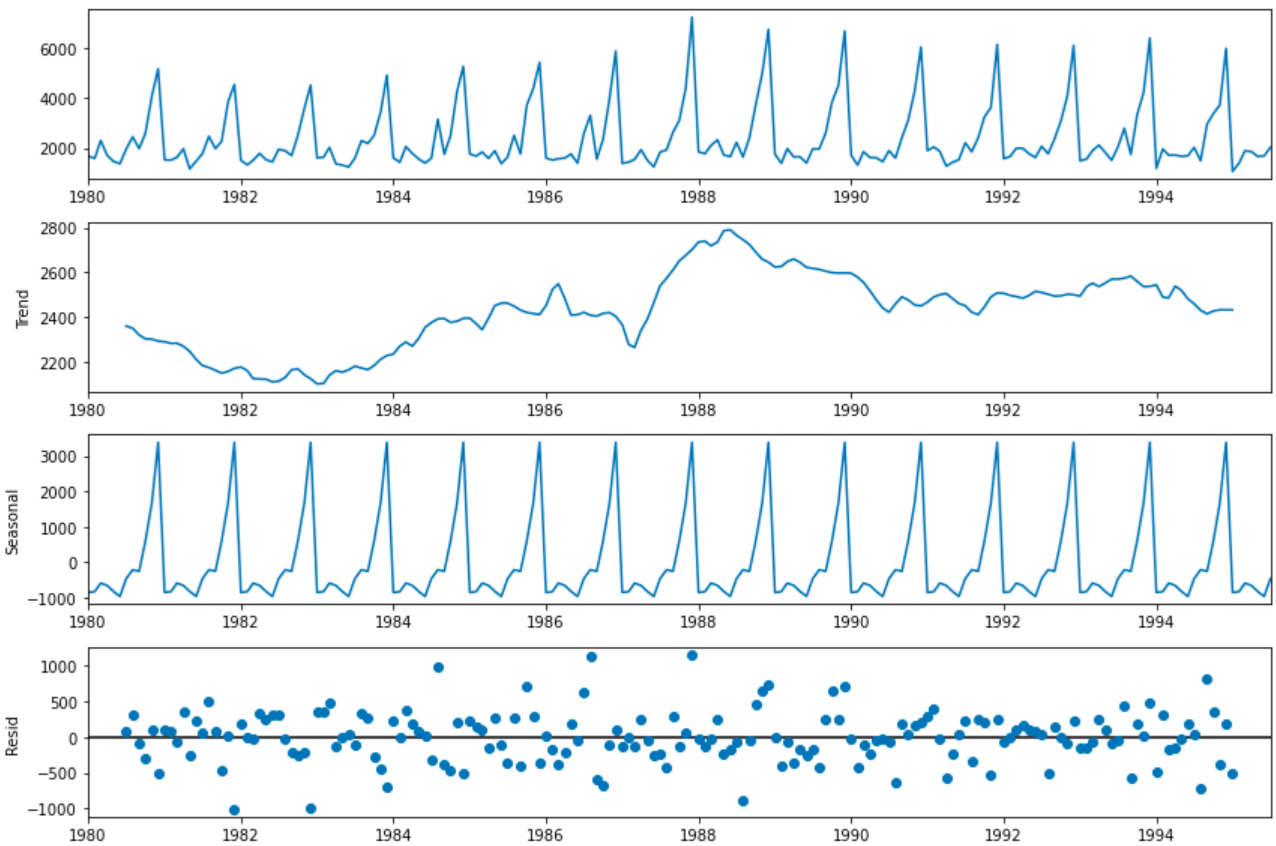
The boxplots for the monthly production for different years show very few outliers.

– Plotting a month-plot of the give Time Series.

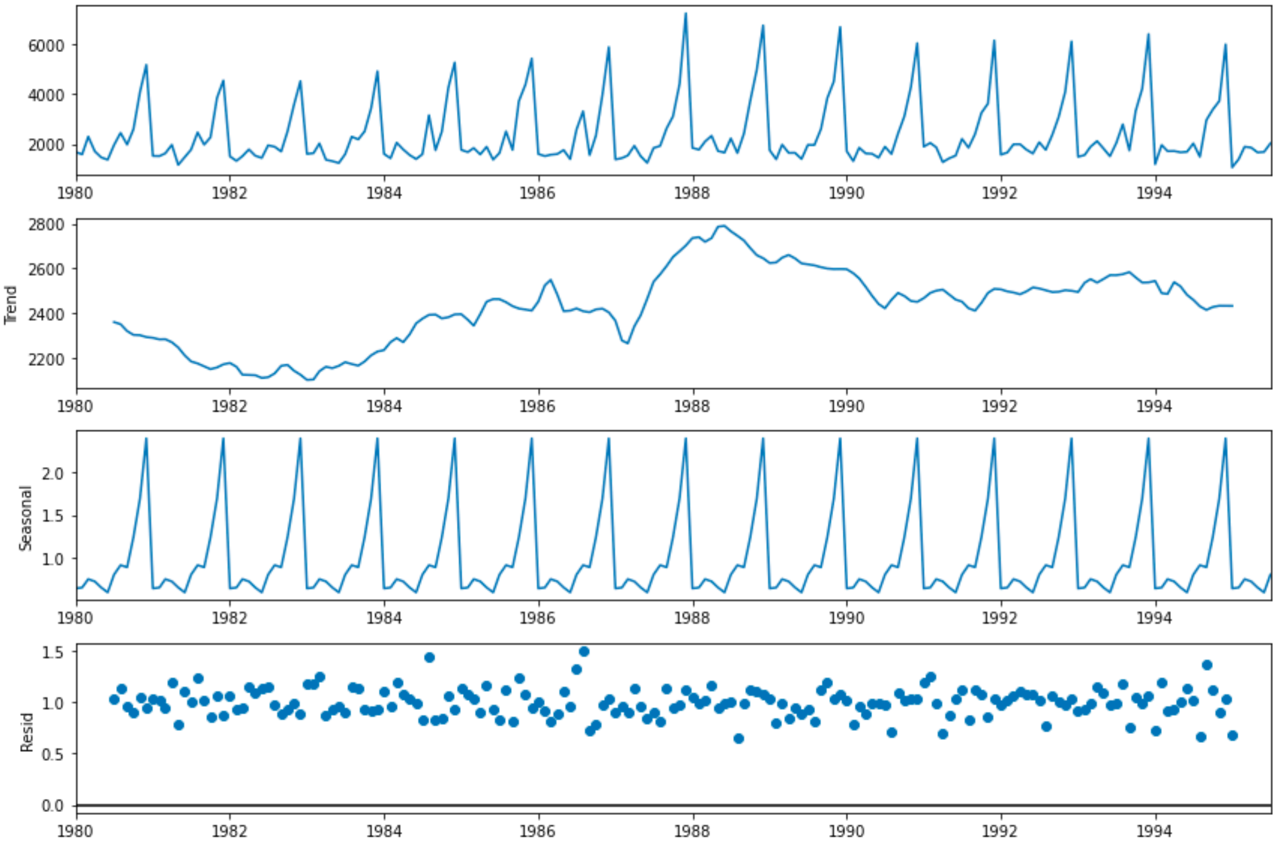


– Decompose the time series using Additive and Multiplicative model.

Additive Model –



Multiplicative Model -



As per the decomposition, we see that there is a pronounced trend in the earlier years of the data. There is a seasonality as well.

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

After splitting the data in train and test. Let's check the data of both train and test using head function.

train.head()

Checking test data after taking year from 1991.

YearMonth	Sparkling
1980-01-01	1686
1980-02-01	1591
1980-03-01	2304
1980-04-01	1712
1980-05-01	1471

```
Test.head()
```

YearMonth	Sparkling
1991-01-01	1902
1991-02-01	2049
1991-03-01	1874
1991-04-01	1279
1991-05-01	1432

Test.shape

(55,1)

There are total 55 records in the test dataset.

4) Build various exponential smoothing models on the training data and evaluate the model using RMSE on the test data. Other models such as regression,naïve forecast models, simple average models etc. should also be built on the training data and check the performance on the test data using RMSE.
Please do try to build as many models as possible and as many iterations of models as possible with different parameters.

Model 1 - Simple Exponential Smoothing-

Single Exponential Smoothing, SES for short, also called Simple Exponential Smoothing, is a time series forecasting method for univariate data without a trend or seasonality.

It requires a single parameter, called alpha (a), also called the smoothing factor or smoothing coefficient.

This parameter controls the rate at which the influence of the observations at prior time steps decay exponentially. Alpha is often set to a value between 0 and 1. Large values mean that the model pays attention mainly to the most recent past observations, whereas smaller values mean more of the history is taken into account when making a prediction.

Fitting and calling model -

```
model_SES = SimpleExpSmoothing(train)

model_SES_autofit = model_SES.fit(optimized=True)

model_SES_autofit.params
```

Checking params for this model -

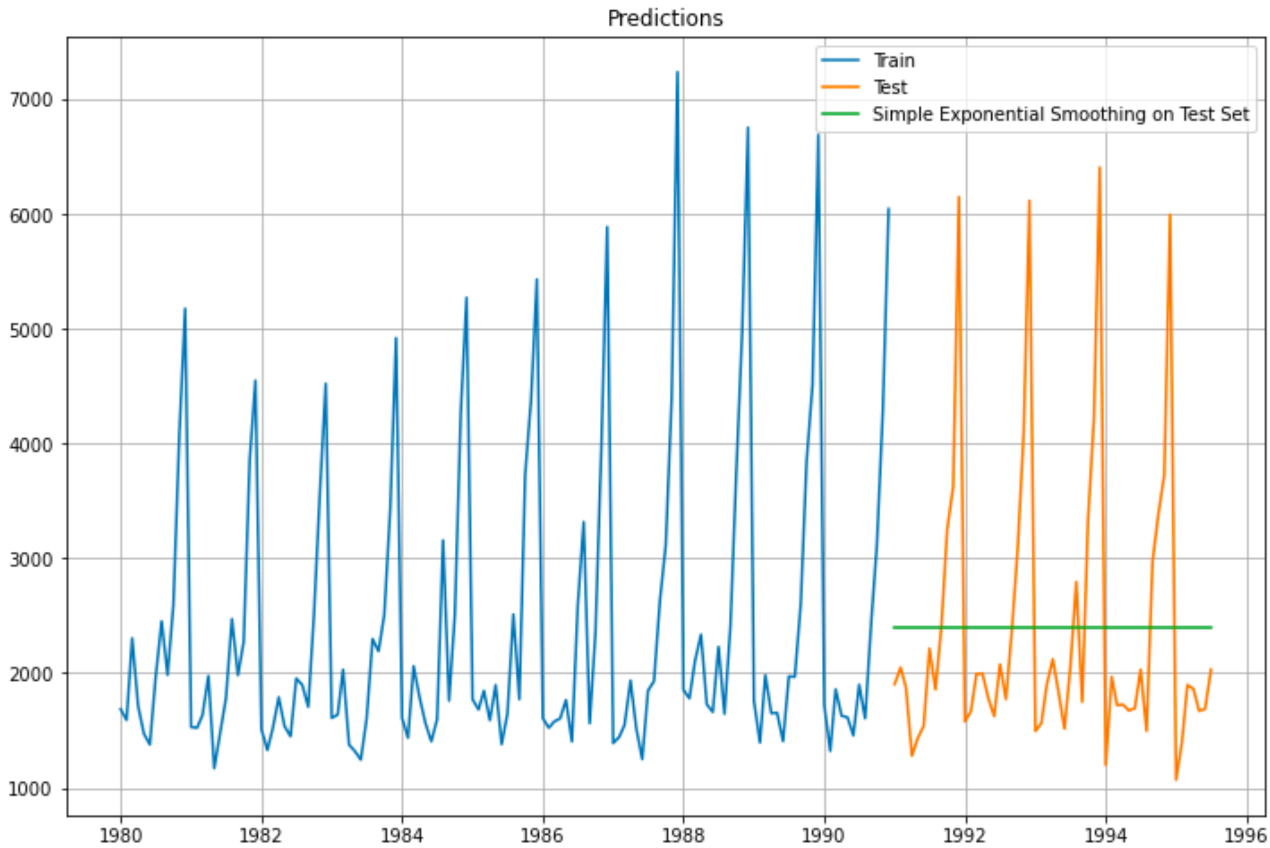
```
{'smoothing_level': 0.0,
 'smoothing_slope': nan,
 'smoothing_seasonal': nan,
 'damping_slope': nan,
 'initial_level': 2403.7785129693348,
 'initial_slope': nan,
 'initial_seasons': array([], dtype=float64),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

Predicting on test dataset-

```
SES_predict = model_SES_autofit.forecast(steps=len(test))
SES_predict

1991-01-01    2403.778513
1991-02-01    2403.778513
1991-03-01    2403.778513
1991-04-01    2403.778513
1991-05-01    2403.778513
1991-06-01    2403.778513
1991-07-01    2403.778513
1991-08-01    2403.778513
1991-09-01    2403.778513
1991-10-01    2403.778513
1991-11-01    2403.778513
1991-12-01    2403.778513
1992-01-01    2403.778513
1992-02-01    2403.778513
1992-03-01    2403.778513
```

```
1992-04-01    2403.778513
1992-05-01    2403.778513
1992-06-01    2403.778513
1992-07-01    2403.778513
1992-08-01    2403.778513
1992-09-01    2403.778513
1992-10-01    2403.778513
1992-11-01    2403.778513
1992-12-01    2403.778513
1993-01-01    2403.778513
1993-02-01    2403.778513
1993-03-01    2403.778513
1993-04-01    2403.778513
1993-05-01    2403.778513
1993-06-01    2403.778513
1993-07-01    2403.778513
1993-08-01    2403.778513
1993-09-01    2403.778513
1993-10-01    2403.778513
1993-11-01    2403.778513
1993-12-01    2403.778513
1994-01-01    2403.778513
1994-02-01    2403.778513
1994-03-01    2403.778513
1994-04-01    2403.778513
1994-05-01    2403.778513
1994-06-01    2403.778513
1994-07-01    2403.778513
1994-08-01    2403.778513
1994-09-01    2403.778513
1994-10-01    2403.778513
1994-11-01    2403.778513
1994-12-01    2403.778513
1995-01-01    2403.778513
1995-02-01    2403.778513
1995-03-01    2403.778513
1995-04-01    2403.778513
1995-05-01    2403.778513
1995-06-01    2403.778513
1995-07-01    2403.778513
Freq: MS, dtype: float64
```

Plotting time series on train and test data-



–RMSE score on Test data –

```
resultsDf = pd.DataFrame({'Test RMSE':
[em.rmse(test.values,SES_predict.values)
[0]]},index=['SES'])
resultsDf
```

	Test RMSE
SES	1275.081797

Model 2 - Double Exponential Smoothing

Double Exponential Smoothing is an extension to Exponential Smoothing that explicitly adds support for trends in the univariate time series.

In addition to the alpha parameter for controlling smoothing factor for the level, an additional smoothing factor is added to control the decay of the influence of the change in trend called beta (b)

Creating and fitting the model -

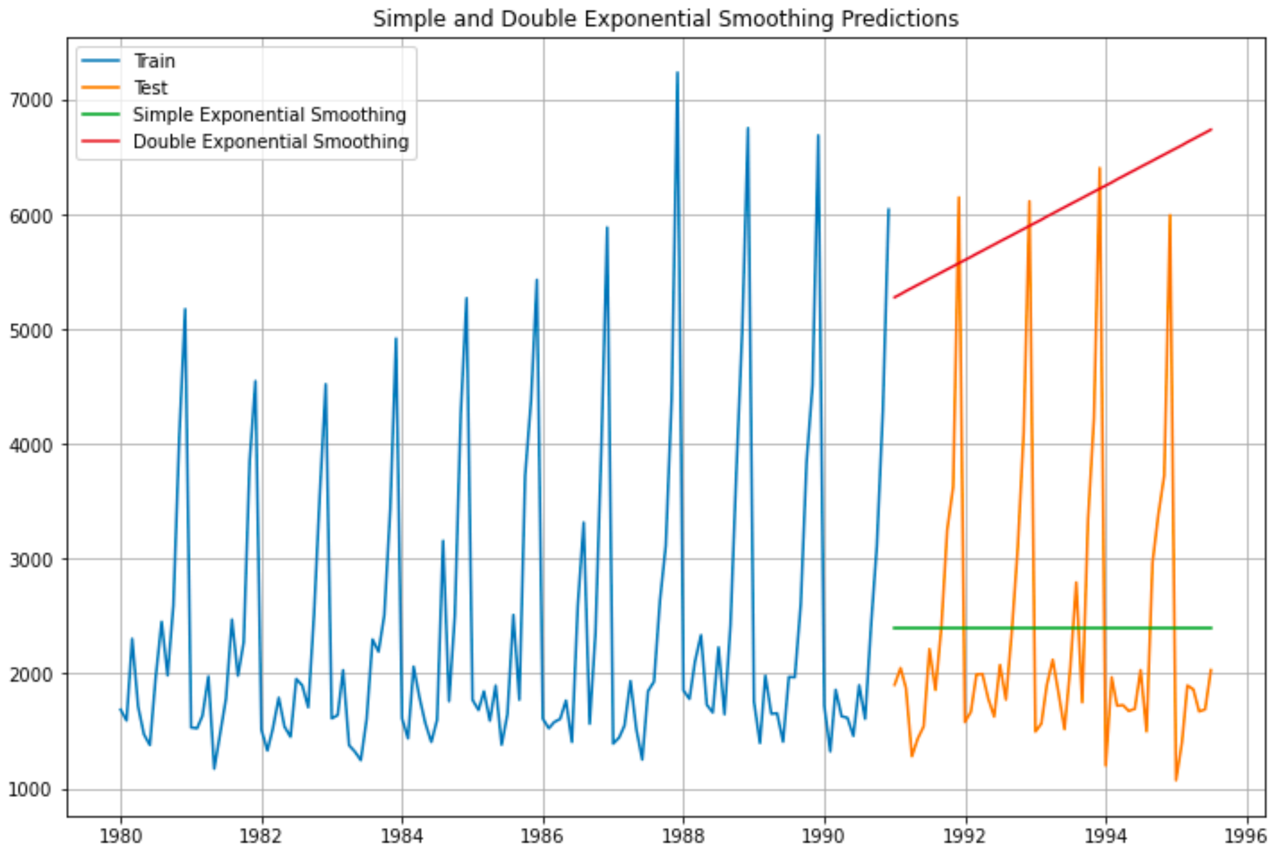
```
model_DES = Holt(train)
# Fitting the model
model_DES = model_DES.fit()

print(model_DES.params)
```

— Predicting on test data —

```
1991-01-01    5281.503812
1991-02-01    5308.564924
1991-03-01    5335.626036
1991-04-01    5362.687148
1991-05-01    5389.748260
1991-06-01    5416.809372
1991-07-01    5443.870484
1991-08-01    5470.931596
1991-09-01    5497.992708
1991-10-01    5525.053820
1991-11-01    5552.114932
1991-12-01    5579.176044
1992-01-01    5606.237156
1992-02-01    5633.298268
1992-03-01    5660.359380
1992-04-01    5687.420492
1992-05-01    5714.481604
```

1992-06-01	5741.542716
1992-07-01	5768.603828
1992-08-01	5795.664940
1992-09-01	5822.726052
1992-10-01	5849.787164
1992-11-01	5876.848276
1992-12-01	5903.909388
1993-01-01	5930.970500
1993-02-01	5958.031612
1993-03-01	5985.092724
1993-04-01	6012.153836
1993-05-01	6039.214948
1993-06-01	6066.276060
1993-07-01	6093.337172
1993-08-01	6120.398284
1993-09-01	6147.459396
1993-10-01	6174.520509
1993-11-01	6201.581621
1993-12-01	6228.642733
1994-01-01	6255.703845
1994-02-01	6282.764957
1994-03-01	6309.826069
1994-04-01	6336.887181
1994-05-01	6363.948293
1994-06-01	6391.009405
1994-07-01	6418.070517
1994-08-01	6445.131629
1994-09-01	6472.192741
1994-10-01	6499.253853
1994-11-01	6526.314965
1994-12-01	6553.376077
1995-01-01	6580.437189
1995-02-01	6607.498301
1995-03-01	6634.559413
1995-04-01	6661.620525
1995-05-01	6688.681637
1995-06-01	6715.742749
1995-07-01	6742.803861

– Plotting time series on train and test data–



Double Exponential Model allows the forecasting of data with a trend.

Here we are forecasting sales of sparkling wine between year 1980 and 1995. We can see there is some trend.

RMSE score on test data- 3851.089353

Model 3 Triple Exponential Smoothing (Additive)

Triple Exponential Smoothing is an extension of Exponential Smoothing that explicitly adds support for seasonality to the univariate time series.

This method is sometimes called Holt-Winters Exponential Smoothing, named for two contributors to the method: Charles Holt and Peter Winters.

In addition to the alpha and beta smoothing factors, a new parameter is added called gamma (g) that controls the influence on the seasonal component.

—Creating and fitting the model

```
model_TES =
ExponentialSmoothing(train,trend='additive',seasonal='add
itive')
# Fitting the model
model_TES = model_TES.fit()
print(model_TES.params)
```

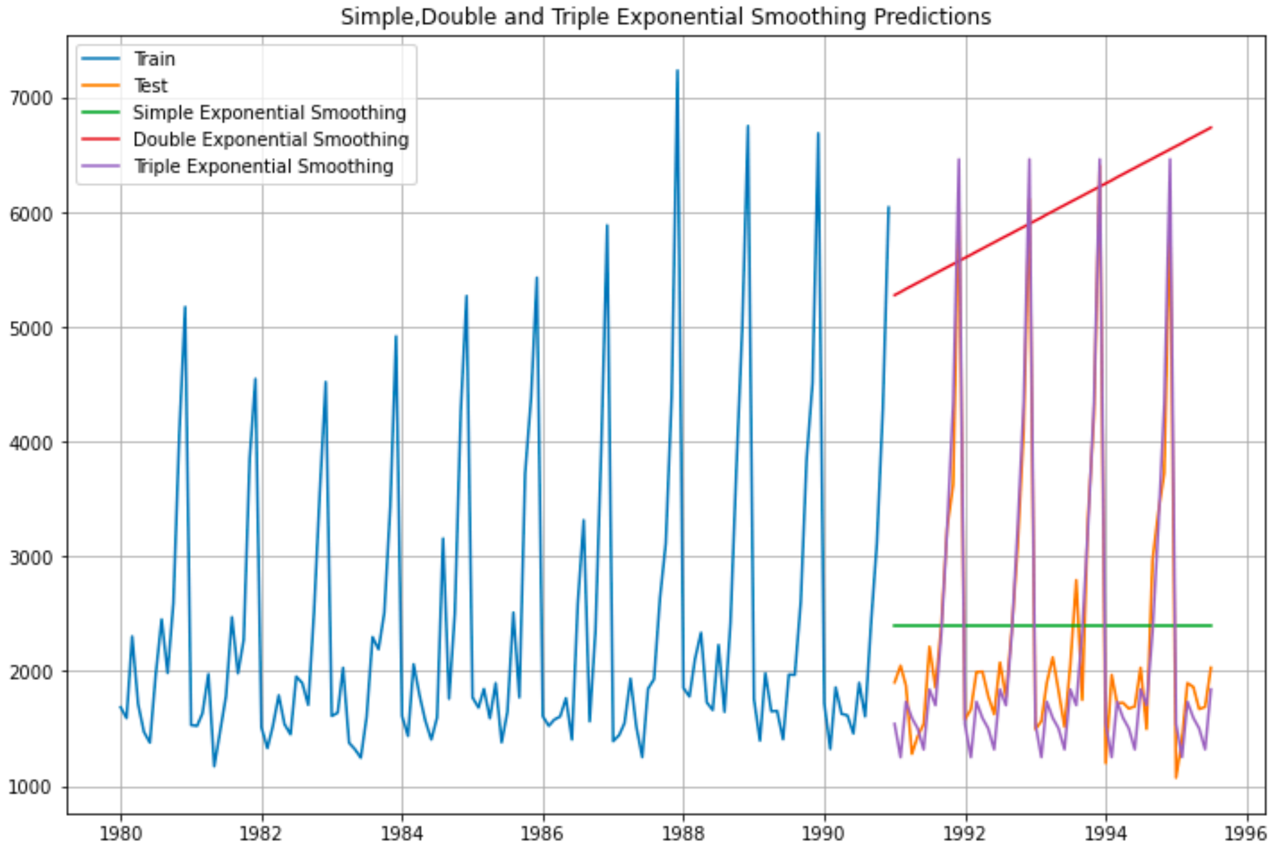
— Predicting on test data-

```
TES_predict =  model_TES.forecast(len(test))
TES_predict
```

```
1991-01-01    1539.837031
1991-02-01    1249.397229
1991-03-01    1733.492420
1991-04-01    1591.489504
1991-05-01    1500.107435
1991-06-01    1317.238310
1991-07-01    1840.821499
1991-08-01    1702.887969
1991-09-01    2342.292584
1991-10-01    3253.431313
1991-11-01    4328.731241
1991-12-01    6466.470708
1992-01-01    1540.119649
```

```
1992-02-01    1249.679847
1992-03-01    1733.775038
1992-04-01    1591.772122
1992-05-01    1500.390053
1992-06-01    1317.520928
1992-07-01    1841.104117
1992-08-01    1703.170587
1992-09-01    2342.575202
1992-10-01    3253.713930
1992-11-01    4329.013859
1992-12-01    6466.753326
1993-01-01    1540.402267
1993-02-01    1249.962465
1993-03-01    1734.057656
1993-04-01    1592.054740
1993-05-01    1500.672671
1993-06-01    1317.803546
1993-07-01    1841.386735
1993-08-01    1703.453205
1993-09-01    2342.857820
1993-10-01    3253.996548
1993-11-01    4329.296477
1993-12-01    6467.035944
1994-01-01    1540.684885
1994-02-01    1250.245083
1994-03-01    1734.340274
1994-04-01    1592.337358
1994-05-01    1500.955289
1994-06-01    1318.086164
1994-07-01    1841.669353
1994-08-01    1703.735823
1994-09-01    2343.140438
1994-10-01    3254.279166
1994-11-01    4329.579095
1994-12-01    6467.318562
1995-01-01    1540.967503
1995-02-01    1250.527701
1995-03-01    1734.622892
1995-04-01    1592.619976
1995-05-01    1501.237907
1995-06-01    1318.368782
1995-07-01    1841.951971
Freq: MS, dtype: float64
```

— Plotting time series on train and test data-



Clearly the model has identified the weekly seasonal pattern and the increasing trend at the end of the data, and the forecasts are a close match to the test data.

RMSE score on test data-

RMSE score is 360.34 for Triple Exponential Method.

Triple Exponential Smoothing (Multiplicative)

Creating and fitting the model

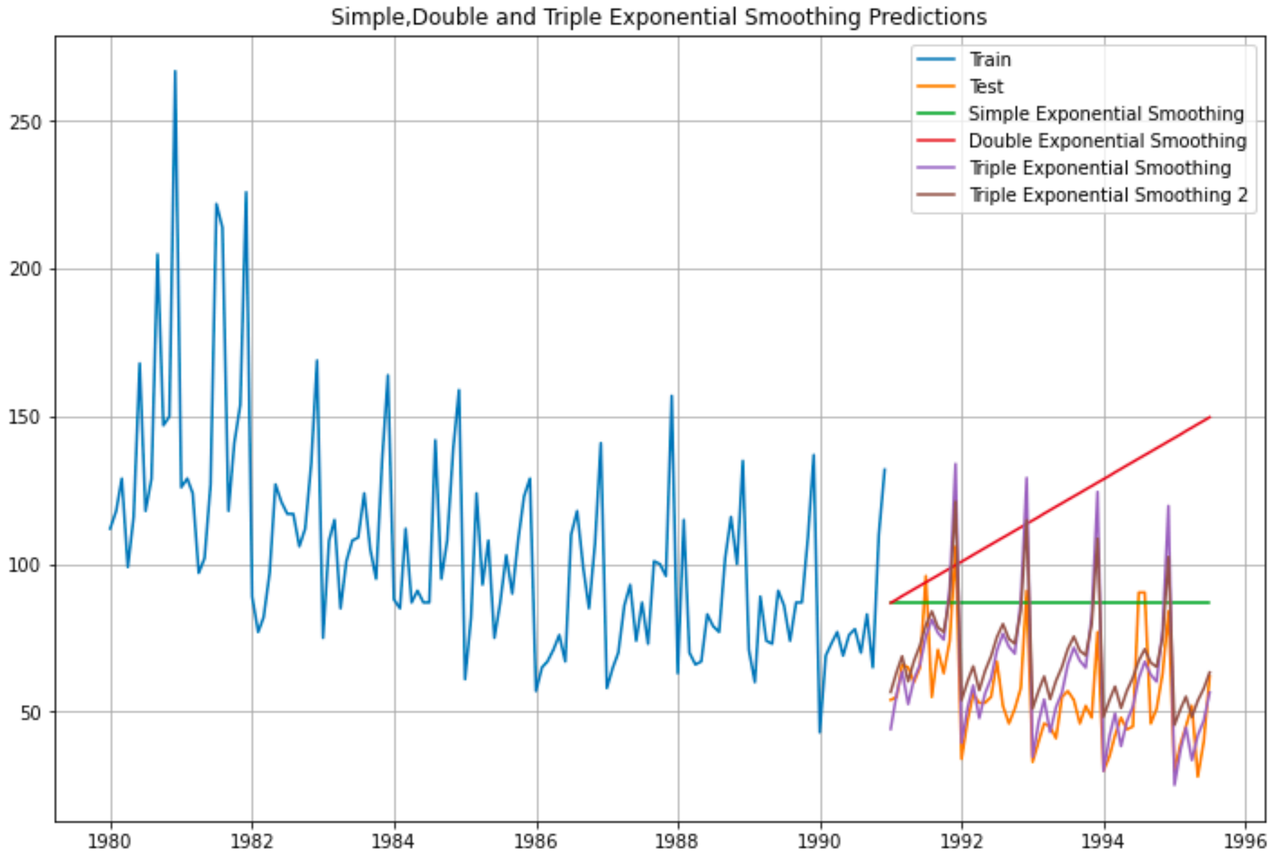
```
model_TES =
ExponentialSmoothing(train,trend='additive',seasonal='add
itive')
# Fitting the model
model_TES = model_TES.fit()
print(model_TES.params)
```

Predicting on test data -

```
1991-01-01    1602.185727
1991-02-01    1373.879845
1991-03-01    1807.433549
1991-04-01    1704.567367
1991-05-01    1602.372641
1991-06-01    1415.478327
1991-07-01    1944.850620
1991-08-01    1910.053047
1991-09-01    2435.202599
1991-10-01    3333.458551
1991-11-01    4407.781796
1991-12-01    6328.533064
1992-01-01    1656.055043
1992-02-01    1419.943906
1992-03-01    1867.865145
1992-04-01    1761.401275
1992-05-01    1655.651124
1992-06-01    1462.412576
1992-07-01    2009.160030
1992-08-01    1973.038265
1992-09-01    2515.284895
1992-10-01    3442.780646
1992-11-01    4551.942764
1992-12-01    6534.951553
1993-01-01    1709.924358
1993-02-01    1466.007966
1993-03-01    1928.296740
1993-04-01    1818.235184
```

1993-05-01	1708.929607
1993-06-01	1509.346825
1993-07-01	2073.469441
1993-08-01	2036.023483
1993-09-01	2595.367191
1993-10-01	3552.102741
1993-11-01	4696.103731
1993-12-01	6741.370042
1994-01-01	1763.793674
1994-02-01	1512.072027
1994-03-01	1988.728336
1994-04-01	1875.069092
1994-05-01	1762.208091
1994-06-01	1556.281074
1994-07-01	2137.778851
1994-08-01	2099.008701
1994-09-01	2675.449487
1994-10-01	3661.424836
1994-11-01	4840.264698
1994-12-01	6947.788531
1995-01-01	1817.662989
1995-02-01	1558.136087
1995-03-01	2049.159932
1995-04-01	1931.903000
1995-05-01	1815.486574
1995-06-01	1603.215323
1995-07-01	2202.088262

Plotting on train and test data -



RMSE score -

RMSE score is 383.16.

Model 4 : Linear Regression Model

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

—Length of Train data

132

—Length of test data

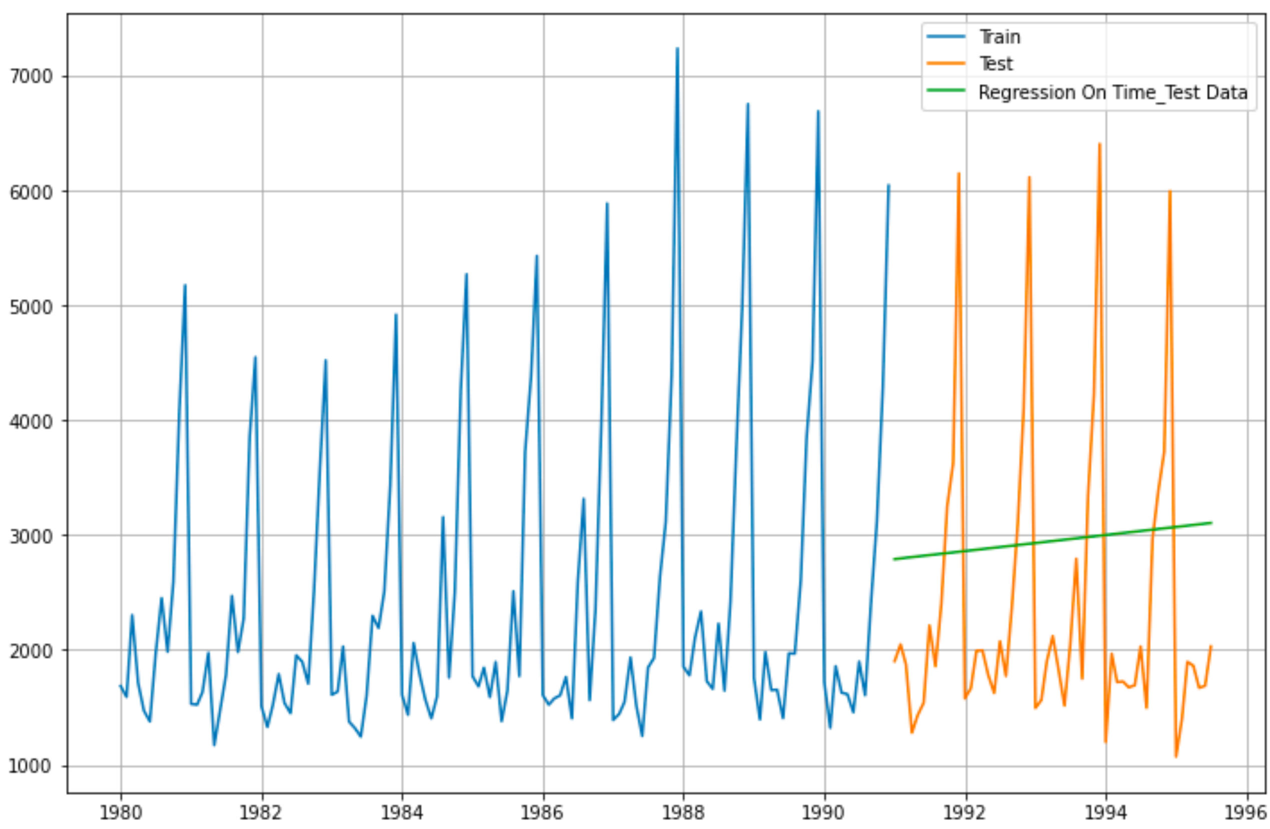
55

Fitting the model-

lr = LinearRegression()

lr.fit(LinearRegression_train[['time']],LinearRegression_train['Sparkling'])

Now plotting on train and test data-



RMSE score on test data -

```
rmse_lr_test =
metrics.mean_squared_error(test['Sparkling'],test_predict
ions_model1,squared=False)
```

RMSE score is 1389.13

Model 5 : Naive Approach

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

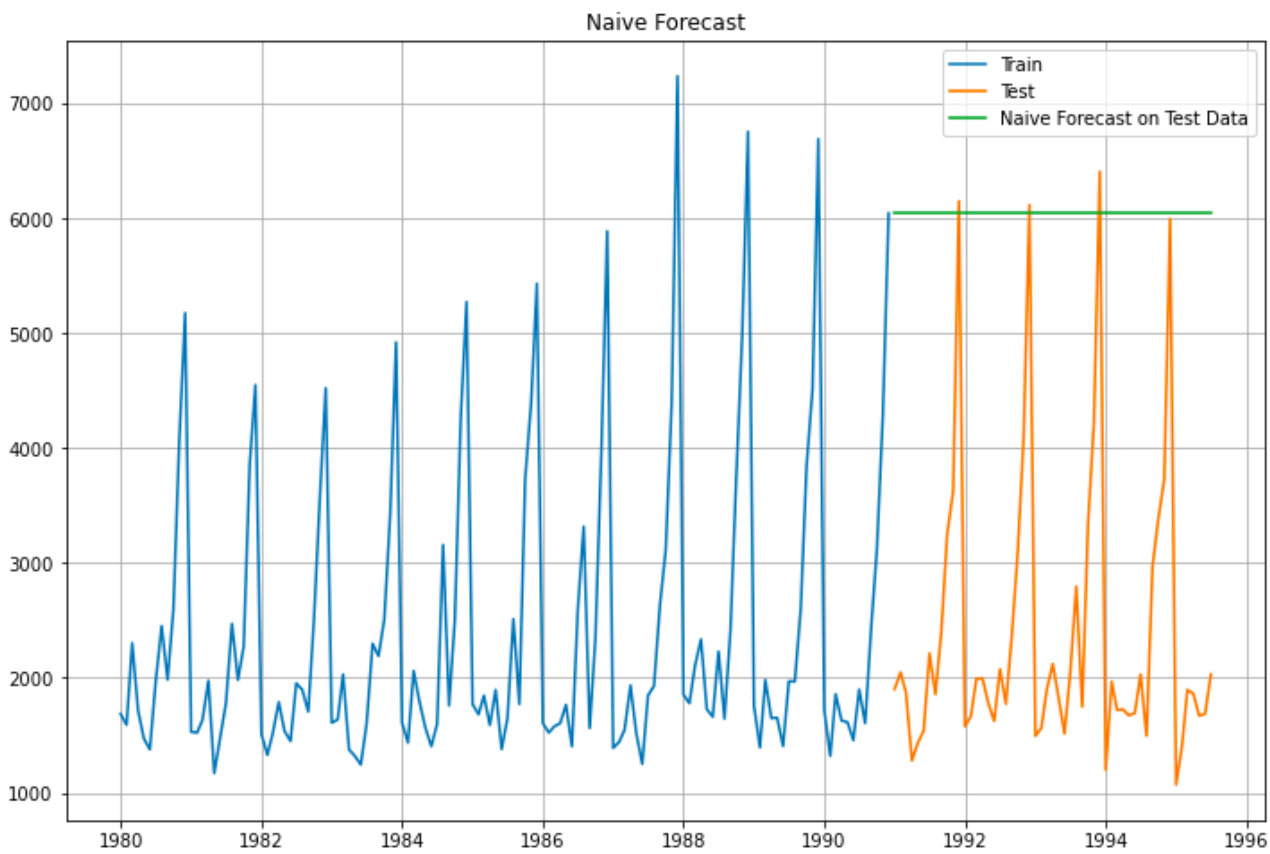
```
NaiveModel_train = train.copy()
NaiveModel_test = test.copy()

NaiveModel_train.head()
```

	Sparkling
YearMonth	
1980-01-01	1686
1980-02-01	1591
1980-03-01	2304
1980-04-01	1712
1980-05-01	1471

```
NaiveModel_test.head()
```

	Sparkling
YearMonth	
1991-01-01	1902
1991-02-01	2049
1991-03-01	1874
1991-04-01	1279
1991-05-01	1432

Plotting on train and test data-



RMSE score on test data -

```
rmse_nb_test =
metrics.mean_squared_error(test[‘Sparkling'],NaiveModel_t
est['naive'],squared=False)
```

RMSE score is 3864.279

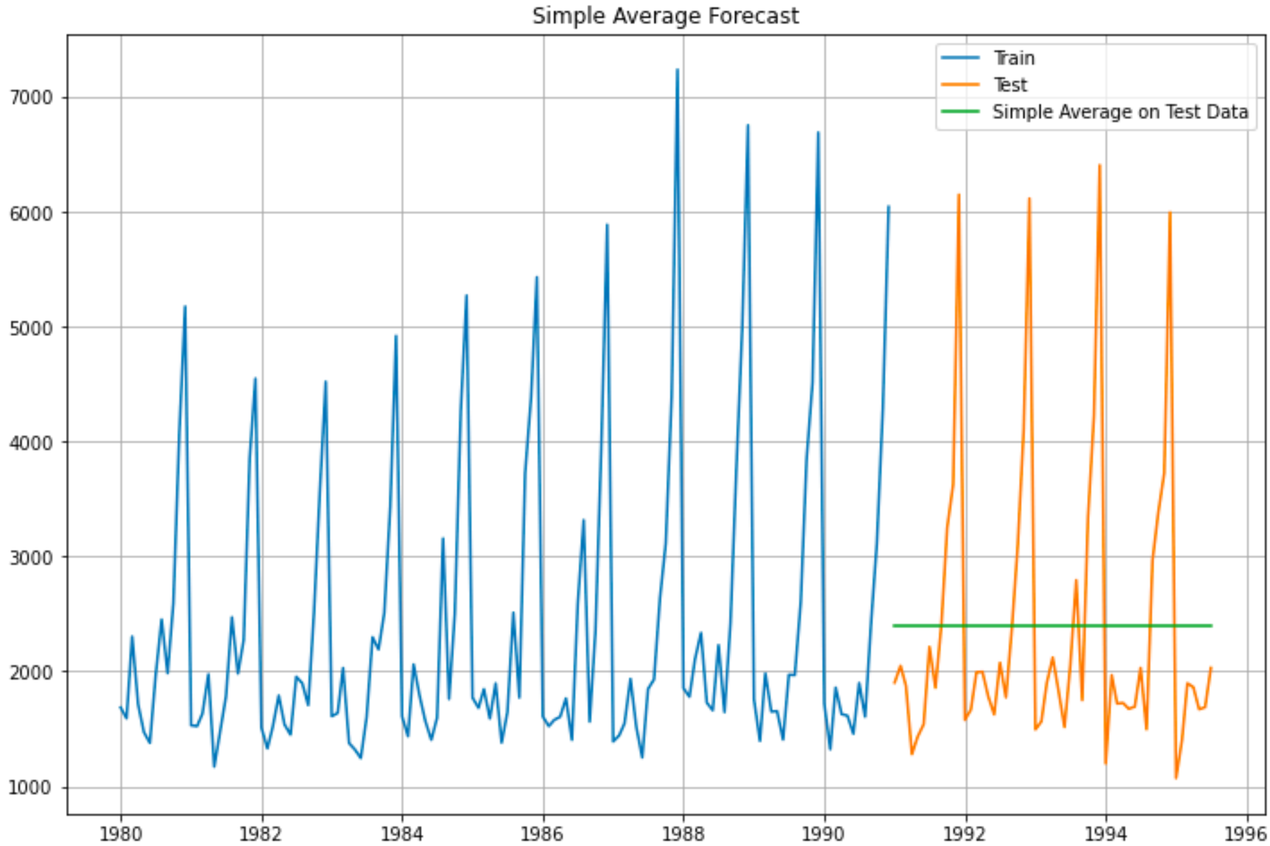
Model 6 : Simple Average

Taking out the average of the data and plotting it.

```
SimpleAverage_train = train.copy()
SimpleAverage_test = test.copy()


SimpleAverage_test['mean_forecast'] =
train['Sparkling'].mean()
SimpleAverage_test.head()
```

	Sparkling	mean_forecast
YearMonth		
1991-01-01	1902	2403.780303
1991-02-01	2049	2403.780303
1991-03-01	1874	2403.780303
1991-04-01	1279	2403.780303
1991-05-01	1432	2403.780303

Plotting train and test Times series -




```
rmse_sm_test =
metrics.mean_squared_error(test[‘Sparkling'],SimpleAverag
e_test['mean_forecast'],squared=False)
```

<u>RMSE score on test data is -</u>

1275.08

Model 7 : Moving Average -

Calculating a moving average involves creating a new series where the values are comprised of the average of raw observations in the original time series.

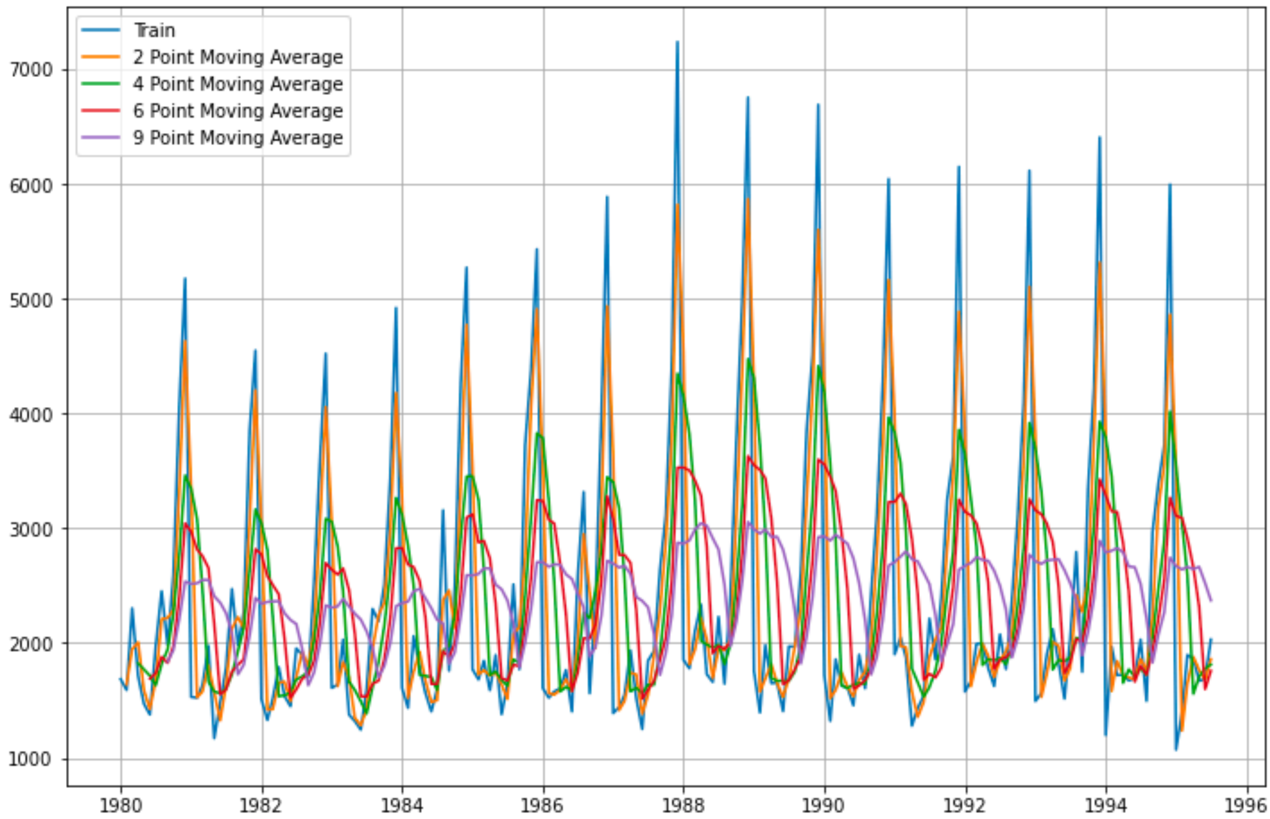
A moving average requires that you specify a window size called the window width. This defines the number of raw observations used to calculate the moving average value.

The “moving” part in the moving average refers to the fact that the window defined by the window width is slide along the time series to calculate the average values in the new series.

```
MovingAverage = df.copy()
MovingAverage.head()
```

	Sparkling
YearMonth	
1980-01-01	1686
1980-02-01	1591
1980-03-01	2304
1980-04-01	1712
1980-05-01	1471

Plotting moving average



Creating and train and test set -

```
trailing_MovingAverage_train=MovingAverage[MovingAverage.
index.year < 1991]
trailing_MovingAverage_test=MovingAverage[MovingAverage.i
ndex.year >= 1991]
```

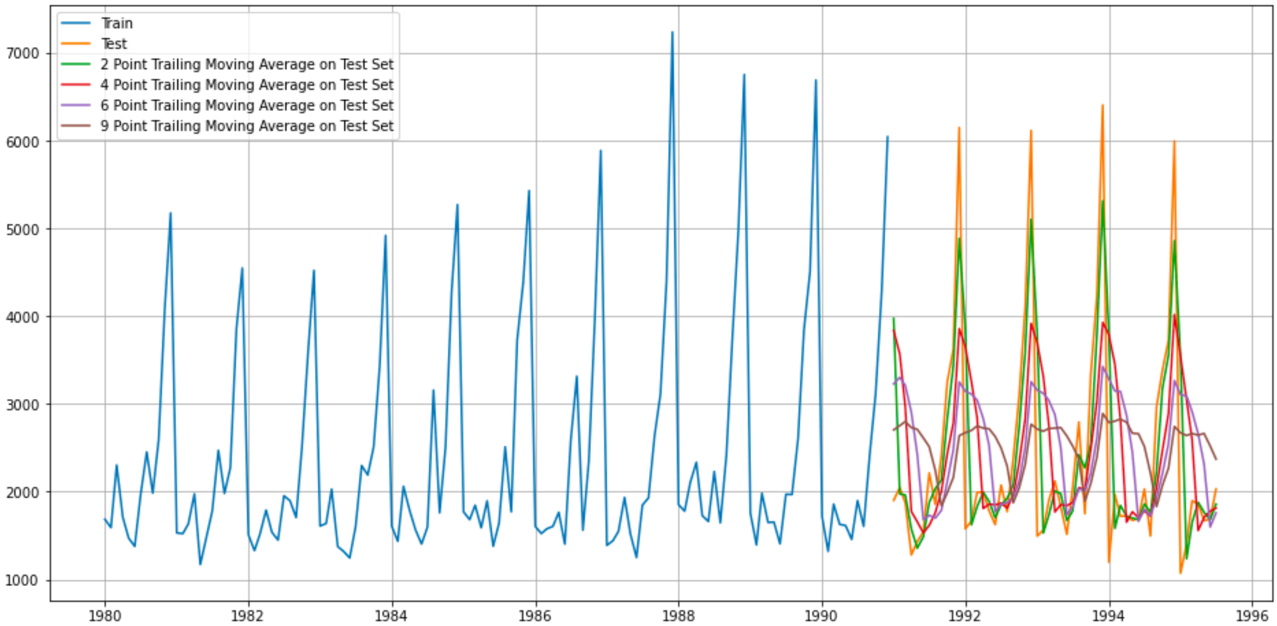
```
trailing_MovingAverage_train.head()
```

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

```
trailing_MovingAverage_test.head()
```

YearMonth	Sparkling	Trailing_2	Trailing_4	Trailing_6	Trailing_9
1991-01-01	1902	3974.5	3837.75	3230.000000	2705.666667
1991-02-01	2049	1975.5	3571.00	3304.000000	2753.888889
1991-03-01	1874	1961.5	2968.00	3212.333333	2800.222222
1991-04-01	1279	1576.5	1776.00	2906.166667	2731.333333
1991-05-01	1432	1355.5	1658.50	2430.500000	2712.111111

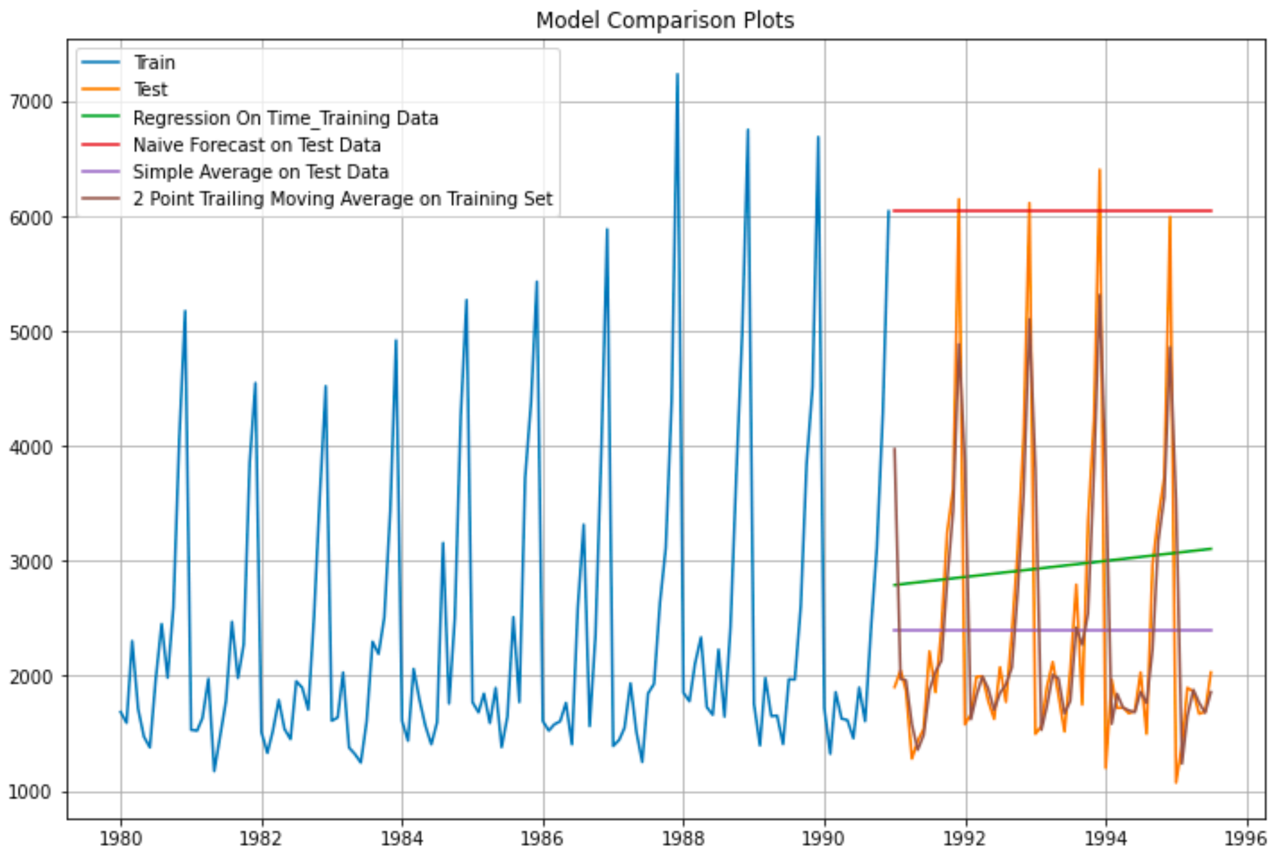
Plotting Moving average time series -



Let's see RMSE score on Training Data -

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.

Let's do all the models comparison



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.

The null and alternative hypothesis states that -

\# H0: Null Hypothesis: The time series is not stationary
\# Ha: Alternate Hypothesis: The time series is stationary

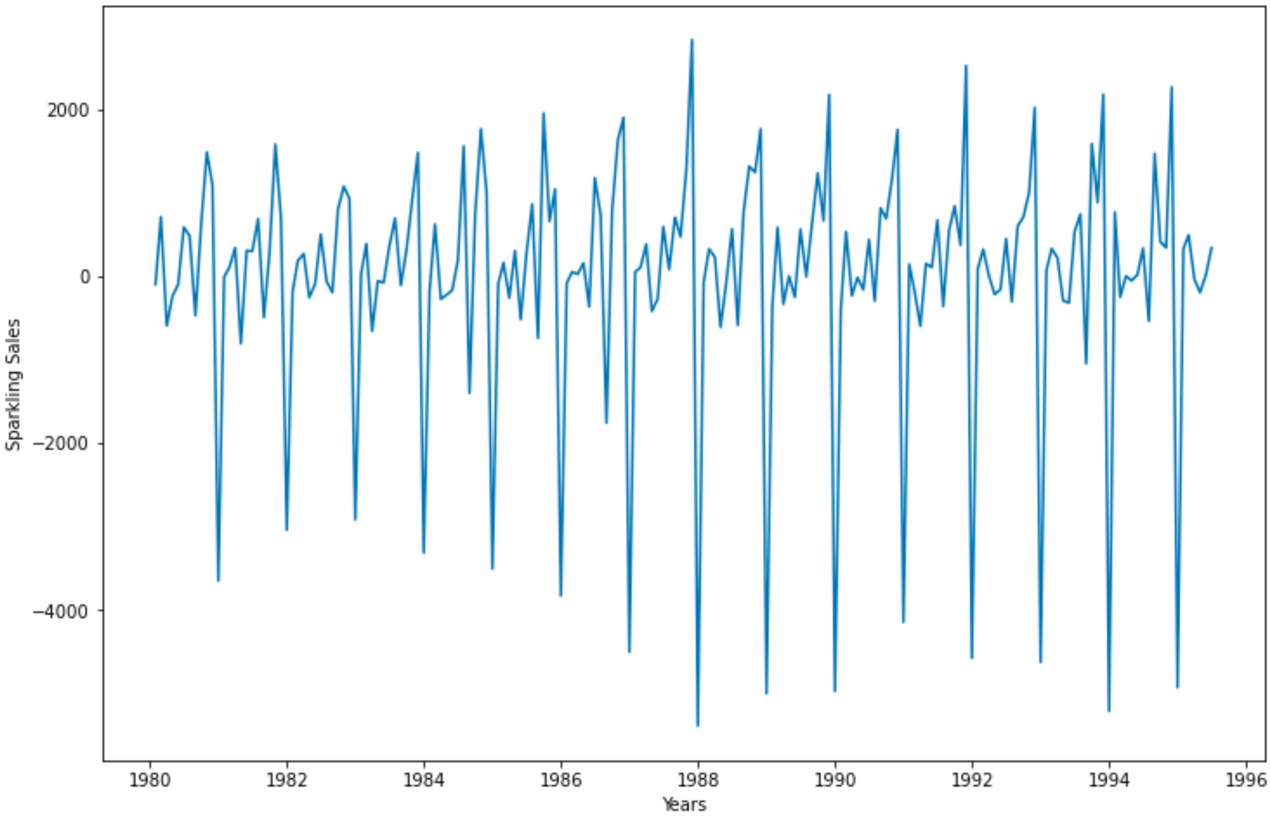
```
dftest = adfuller(df)
dftest
print('DF test statistic is %3.3f' %dftest[0])
print('DF test p-value is %1.4f' %dftest[1])
```

DF test statistic is -1.360
DF test p-value is 0.6011

As p-value is > 0.05 we fail to reject H0 and we can say that the time series is not stationary.

Let's take data difference with period =1

```
data_diff = df.diff(periods=1)
data_diff.dropna(inplace=True)
```



```
dftest_diff = adfuller(data_diff)
dftest_diff
print('DF test statistic is %3.3f' %dftest_diff[0])
print('DF test p-value is %1.4f' %dftest_diff[1])

DF test statistic is -45.050
DF test p-value is 0.0000
```

As we can see after taking difference p-value drops to 0 which is < 0.05 so we reject H0 and conclude that
the time series is now stationary and we can proceed with our models.

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

Any 'non-seasonal' time series that exhibits patterns and is not a random white noise can be modeled with ARIMA models.

An ARIMA model is characterized by 3 terms: p, d, q where,
p is the order of the AR term
q is the order of the MA term

d is the number of differencing required to make the time series stationary

If a time series, has seasonal patterns, then we need to add seasonal terms and it becomes SARIMA, short for 'Seasonal ARIMA'

```
# Define the p, d and q parameters to take any value
between 0 and 2
p = d = q = range(0, 2)

# Generate all different combinations of p, d and q
triplets
pdq = list(itertools.product(p, d, q))

# Generate all different combinations of seasonal p, q
and q triplets
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in
list(itertools.product(p, d, q))]
```

pdq

```
[(0, 0, 0),
 (0, 0, 1),
 (0, 1, 0),
 (0, 1, 1),
 (1, 0, 0),
 (1, 0, 1),
 (1, 1, 0),
 (1, 1, 1)]
```

seasonal_pdq

```
[(0, 0, 0, 12),
 (0, 0, 1, 12),
 (0, 1, 0, 12),
 (0, 1, 1, 12),
 (1, 0, 0, 12),
 (1, 0, 1, 12),
 (1, 1, 0, 12),
 (1, 1, 1, 12)]
```

1.ARIMA Model

Let's check for the best AIC. After looping through (p,d,q) values for ARIMA.

```
Parmas(0, 0, 0) - AICs-2271.203212328525
Parmas(0, 0, 1) - AICs-2245.268852081529
Parmas(0, 1, 0) - AICs-2269.582796371201
Parmas(0, 1, 1) - AICs-2264.906439225404
Parmas(1, 0, 0) - AICs-2247.3482714177458
Parmas(1, 0, 1) - AICs-2245.9490900488477
Parmas(1, 1, 0) - AICs-2268.5280606863257
Parmas(1, 1, 1) - AICs-2235.013945350335
```

So the best AIC is AIC:2235.775752684674 with p = 1, d = 1 and q = 1

```
arima_model = ARIMA(train,order=(1,1,1)).fit()
```

Stats Summary of ARIMA model -

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:                 D.Rose   No. Observations:                  131
Model:                 ARIMA(1, 1, 1)   Log Likelihood                -634.888
Method:                       css-mle   S.D. of innovations             30.279
Date:                Fri, 06 Nov 2020   AIC                           1277.776
Time:                        00:02:25   BIC                           1289.277
Sample:                    02-01-1980   HQIC                          1282.449
                         - 12-01-1990
=================================================================================
                    coef    std err          z      P>|z|      [0.025      0.975]
---------------------------------------------------------------------------------
const            -0.4871      0.086     -5.656      0.000      -0.656      -0.318
ar.L1.D.Rose      0.2006      0.087      2.293      0.022       0.029       0.372
ma.L1.D.Rose     -0.9999      0.035    -28.646      0.000      -1.068      -0.932
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            4.9856           +0.0000j            4.9856            0.0000
MA.1            1.0001           +0.0000j            1.0001            0.0000
-----------------------------------------------------------------------------
```

As we can see both MA part of order 1 AR part of order 1 are more significant as they have a p-value of 0.000.

Let us predict the model on the test set as the test data size is 55 we will take the steps = 55 as the number of observations we need to predict.

Test RMSE Score -

1461.6676526007445

2. SARIMA Model -

```
best_aic = np.inf
best_pdq = None
best_seasonal_pdq = None
temp_model = None
```

Let's check for the AIC -

```
AICs 2465.5831209790667 (0, 0, 0) (0, 0, 0, 12)
AICs 2336.0736172365087 (0, 0, 0) (0, 0, 1, 12)
AICs 1800.605209089621 (0, 0, 0) (0, 1, 0, 12)
AICs 1781.4739548921407 (0, 0, 0) (0, 1, 1, 12)
AICs 2025.459339690266 (0, 0, 0) (1, 0, 0, 12)
AICs 2010.5722826744268 (0, 0, 0) (1, 0, 1, 12)
AICs 1786.622560559047 (0, 0, 0) (1, 1, 0, 12)
AICs 1783.4420050136005 (0, 0, 0) (1, 1, 1, 12)
AICs 2372.4532492708477 (0, 0, 1) (0, 0, 0, 12)
AICs 2250.686154292587 (0, 0, 1) (0, 0, 1, 12)
AICs 1795.9583536204057 (0, 0, 1) (0, 1, 0, 12)
AICs 1778.457988521774 (0, 0, 1) (0, 1, 1, 12)
AICs 2014.5361051874834 (0, 0, 1) (1, 0, 0, 12)
AICs 2002.0683232941767 (0, 0, 1) (1, 0, 1, 12)
AICs 1783.5777819001144 (0, 0, 1) (1, 1, 0, 12)
AICs 1780.263720158094 (0, 0, 1) (1, 1, 1, 12)
AICs 2267.6630357855465 (0, 1, 0) (0, 0, 0, 12)
```

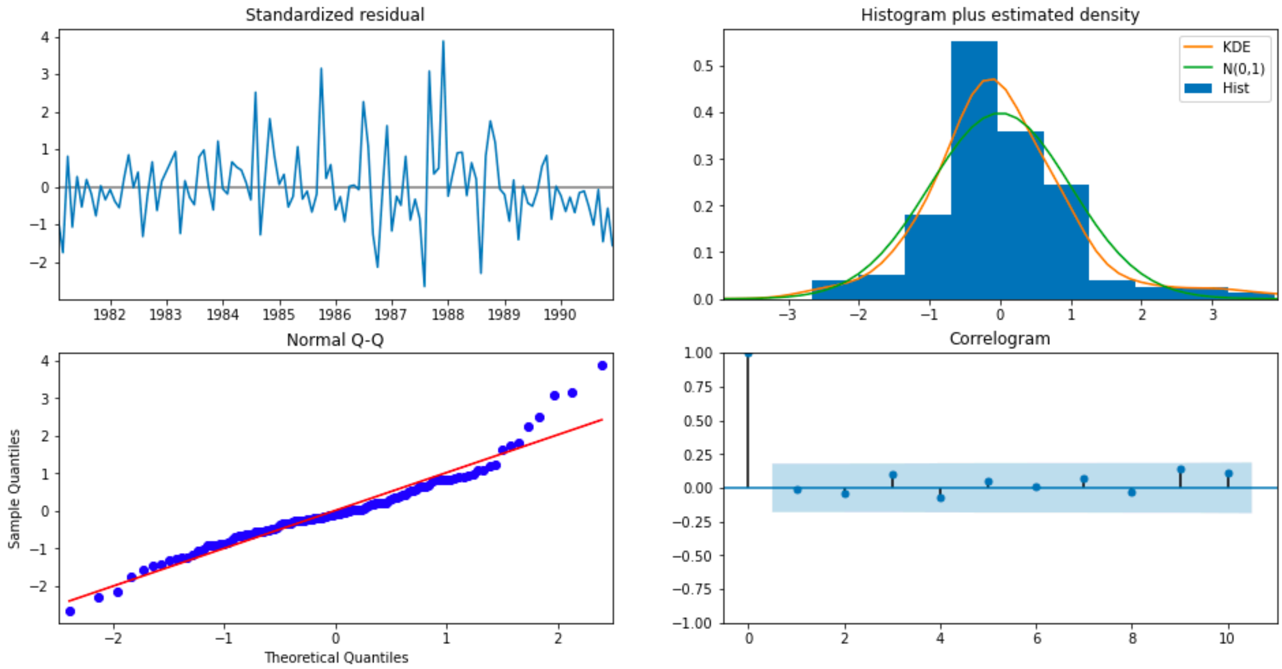
```
AICs 2179.939363178083 (0, 1, 0) (0, 0, 1, 12)
AICs 1837.436763242784 (0, 1, 0) (0, 1, 0, 12)
AICs 1820.8690065108165 (0, 1, 0) (0, 1, 1, 12)
AICs 2061.3806922605127 (0, 1, 0) (1, 0, 0, 12)
AICs 2049.8777358244074 (0, 1, 0) (1, 0, 1, 12)
AICs 1826.0909414867165 (0, 1, 0) (1, 1, 0, 12)
AICs 1822.4059517568458 (0, 1, 0) (1, 1, 1, 12)
AICs 2263.0600155831944 (0, 1, 1) (0, 0, 0, 12)
AICs 2171.8628995286795 (0, 1, 1) (0, 0, 1, 12)
AICs 1795.5874917947513 (0, 1, 1) (0, 1, 0, 12)
AICs 1773.6337244354402 (0, 1, 1) (0, 1, 1, 12)
AICs 2012.0689418156394 (0, 1, 1) (1, 0, 0, 12)
AICs 2019.3987991314848 (0, 1, 1) (1, 0, 1, 12)
AICs 1778.9512471714872 (0, 1, 1) (1, 1, 0, 12)
AICs 1775.633565173928 (0, 1, 1) (1, 1, 1, 12)
AICs 2281.18624152835 (1, 0, 0) (0, 0, 0, 12)
AICs 2185.3516909887016 (1, 0, 0) (0, 0, 1, 12)
AICs 1796.9354187578729 (1, 0, 0) (0, 1, 0, 12)
AICs 1778.5396578338296 (1, 0, 0) (0, 1, 1, 12)
AICs 2014.4331578178476 (1, 0, 0) (1, 0, 0, 12)
AICs 2000.627855813489 (1, 0, 0) (1, 0, 1, 12)
AICs 1783.7829038980728 (1, 0, 0) (1, 1, 0, 12)
AICs 1780.385302566101 (1, 0, 0) (1, 1, 1, 12)
AICs 2280.580892325116 (1, 0, 1) (0, 0, 0, 12)
AICs 2183.985325081195 (1, 0, 1) (0, 0, 1, 12)
AICs 1797.4680437388697 (1, 0, 1) (0, 1, 0, 12)
AICs 1779.9939863658458 (1, 0, 1) (0, 1, 1, 12)
AICs 2015.1051011891163 (1, 0, 1) (1, 0, 0, 12)
AICs 1996.2325294406974 (1, 0, 1) (1, 0, 1, 12)
AICs 1784.9897010542581 (1, 0, 1) (1, 1, 0, 12)
AICs 1781.8320579764181 (1, 0, 1) (1, 1, 1, 12)
AICs 2266.608539319009 (1, 1, 0) (0, 0, 0, 12)
AICs 2177.955568925301 (1, 1, 0) (0, 0, 1, 12)
AICs 1825.9888034585597 (1, 1, 0) (0, 1, 0, 12)
AICs 1804.2190675211223 (1, 1, 0) (0, 1, 1, 12)
AICs 2050.439331534795 (1, 1, 0) (1, 0, 0, 12)
AICs 2033.6175539254127 (1, 1, 0) (1, 0, 1, 12)
AICs 1810.2270422633364 (1, 1, 0) (1, 1, 0, 12)
AICs 1806.2003804649391 (1, 1, 0) (1, 1, 1, 12)
AICs 2235.755093031724 (1, 1, 1) (0, 0, 0, 12)
AICs 2142.3868326403667 (1, 1, 1) (0, 0, 1, 12)
AICs 1790.3825028612412 (1, 1, 1) (0, 1, 0, 12)
```

```
AICs 1772.71020608695 (1, 1, 1) (0, 1, 1, 12)
AICs 2005.7801370294824 (1, 1, 1) (1, 0, 0, 12)
AICs 1778.0802505918882 (1, 1, 1) (1, 1, 0, 12)
AICs 1774.4149655838733 (1, 1, 1) (1, 1, 1, 12)


# So the best params are:
# p = 1, d = 1, q = 1
# P = 0, D = 1, q =1
# with a seasonal parameter of 12
# and best AIC of 1772.71020608695
```

Statistical Summary SARIMA model -

```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                            Sparkling   No. Observations:                  132
Model:             SARIMAX(1, 1, 1)x(0, 1, 1, 12)   Log Likelihood                -882.355
Date:                              Fri, 06 Nov 2020   AIC                           1772.710
Time:                                      07:17:01   BIC                           1783.827
Sample:                                  01-01-1980   HQIC                          1777.224
                                       - 12-01-1990
Covariance Type:                                opg
==========================================================================================
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.2355      0.089      2.652      0.008       0.061       0.410
ma.L1         -0.9990      0.705     -1.417      0.156      -2.381       0.383
ma.S.L12      -0.4418      0.070     -6.288      0.000      -0.579      -0.304
sigma2      1.491e+05   1.01e+05      1.481      0.139   -4.83e+04    3.46e+05
==============================================================================
```

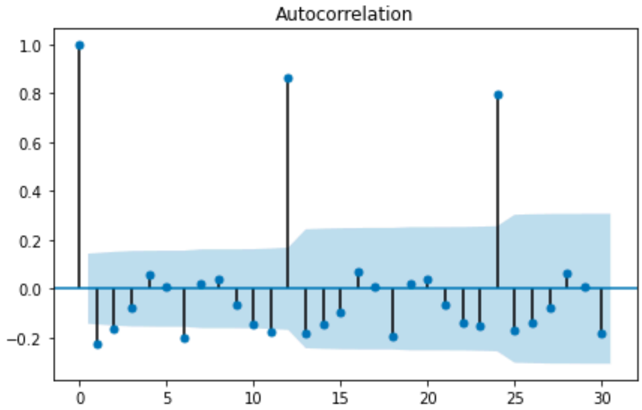
SARIMA Model Plots -

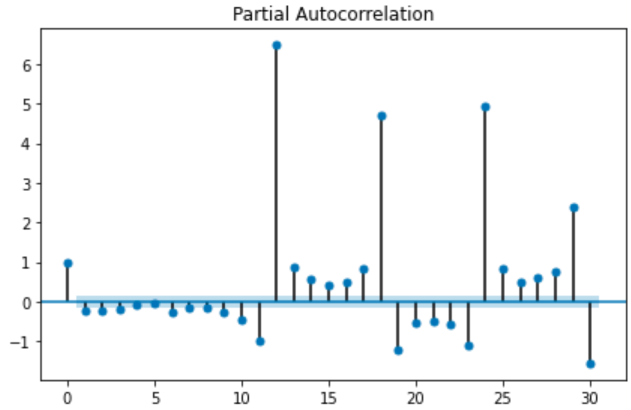


RMSE score is - 343.3116320860303

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.

Let's see ACF and PACF plots -





\# From ACF and PACF plots above we can notive few points -
\# 1. Significant lag after which the ACF cuts-off is 3 that is p = 2
\# 2. Significant lag after which the PACF cuts-off is 4 that is q = 4
\# 3. Searsonal lag is 12

1. ARIMA Model -

```
manual_arima_model = ARIMA(train,order=(2,1,4)).fit()
```

AIC = 2220.2205054806113

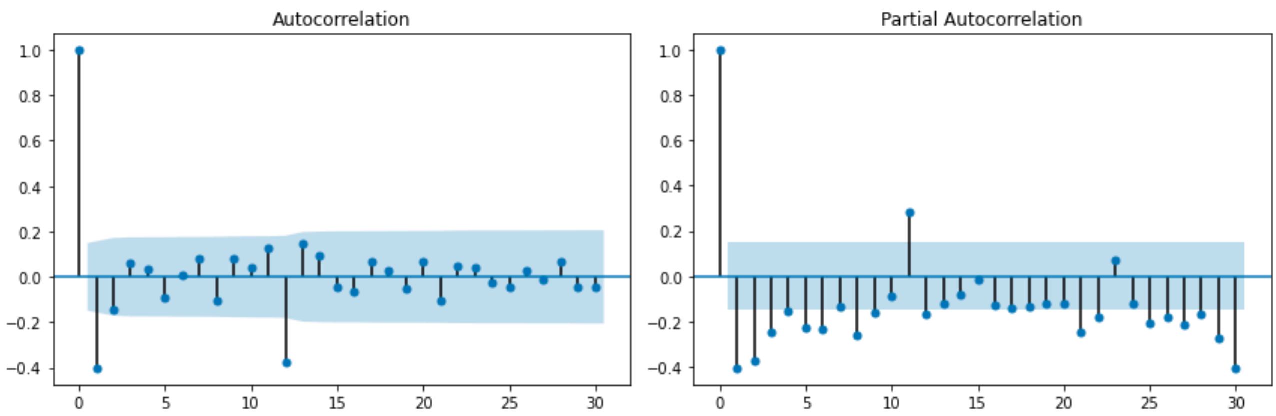
ARIMA Model Summary -

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:            D.Sparkling   No. Observations:                  131
Model:                 ARIMA(2, 1, 4)   Log Likelihood               -1102.110
Method:                       css-mle   S.D. of innovations           1050.995
Date:                Sat, 07 Nov 2020   AIC                           2220.221
Time:                        20:13:46   BIC                           2243.222
Sample:                    02-01-1980   HQIC                          2229.567
                         - 12-01-1990
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                 6.1440      3.925      1.565      0.117      -1.548      13.836
ar.L1.D.Sparkling    -1.5963      0.048    -33.499      0.000      -1.690      -1.503
ar.L2.D.Sparkling    -0.8909      0.043    -20.532      0.000      -0.976      -0.806
ma.L1.D.Sparkling     1.1864      0.077     15.309      0.000       1.035       1.338
ma.L2.D.Sparkling    -0.2269      0.065     -3.509      0.000      -0.354      -0.100
ma.L3.D.Sparkling    -1.3516      0.066    -20.452      0.000      -1.481      -1.222
ma.L4.D.Sparkling    -0.6078      0.072     -8.498      0.000      -0.748      -0.468
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           -0.8960           -0.5655j            1.0595           -0.4104
AR.2           -0.8960           +0.5655j            1.0595            0.4104
MA.1            1.0000           -0.0000j            1.0000           -0.0000
MA.2           -0.7893           -0.6140j            1.0000           -0.3948
MA.3           -0.7893           +0.6140j            1.0000            0.3948
MA.4           -1.6451           -0.0000j            1.6451           -0.5000
-----------------------------------------------------------------------------
```

RMSE Score -1399.0555785963588

2.SARIMA Model -

Let's see ACF and PACF plots -



From ACF and PACF plots above we can say:

- ● Significant lag after which the ACF cuts-off is 2 that is Q = 2

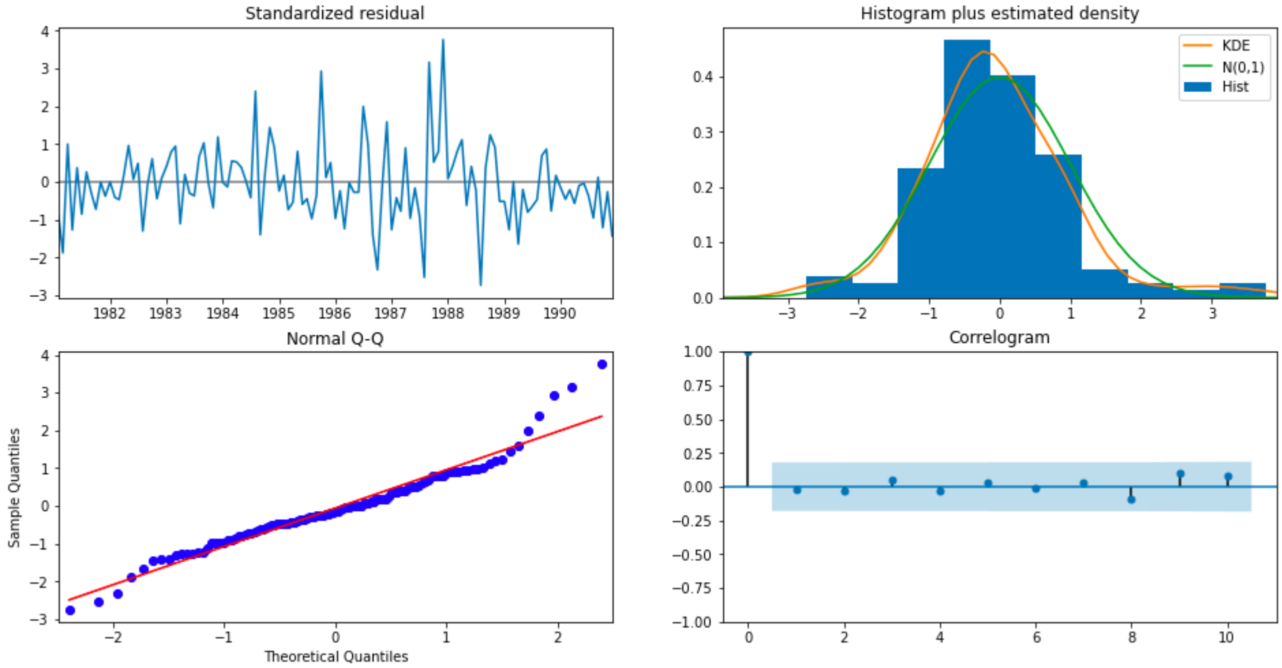
- ● Significant lag after which the PACF cuts-off is 3 that is P = 3 ● Seasonallagis12andD=1

```
sarima_manual_model = sm.tsa.statespace.SARIMAX(train,
                                order=(3, 1, 2),
```

Model Summary -

```
                                 SARIMAX Results
==========================================================================================
Dep. Variable:                         Sparkling   No. Observations:                  132
Model:             SARIMAX(6, 1, 1)x(2, 1, 1, 12)  Log Likelihood                -881.123
Date:                           Fri, 06 Nov 2020   AIC                           1784.246
Time:                                   07:34:04   BIC                           1814.816
Sample:                               01-01-1980   HQIC                          1796.659
                                    - 12-01-1990
Covariance Type:                             opg
==========================================================================================
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.1038      0.172      0.605      0.545      -0.233       0.440
ar.L2         -0.1224      0.128     -0.958      0.338      -0.373       0.128
ar.L3         -0.0208      0.123     -0.169      0.866      -0.263       0.221
ar.L4         -0.1430      0.131     -1.095      0.273      -0.399       0.113
ar.L5         -0.0142      0.132     -0.108      0.914      -0.272       0.244
ar.L6         -0.0630      0.128     -0.492      0.622      -0.314       0.188
ma.L1         -0.8370      0.118     -7.073      0.000      -1.069      -0.605
ar.S.L12      -0.0657      0.437     -0.150      0.880      -0.922       0.791
ar.S.L24      -0.0493      0.186     -0.265      0.791      -0.414       0.315
ma.S.L12      -0.3881      0.413     -0.939      0.348      -1.199       0.422
sigma2      1.483e+05   1.91e+04      7.767      0.000    1.11e+05    1.86e+05
==============================================================================
```

Plotting SARIMA model -



Some Inference from the Plots above -

1. Qq plots shows some linear trend. This shows that the residuals are normally distributed.

2. The KDE plot of the residuals is almost similar with the normal distribution.

RMSE on test data - 546.0119766985941

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.

After creating the data frame of all the models TEST RMSE -

	Test RMSE
SES	1275.081797
DES	3851.089353
TES	360.347809
TES2	383.169844
LR	1389.135175
NB	3864.279352
Simple Average	1275.081804
2pointTrailingMovingAverage	813.400684
4pointTrailingMovingAverage	1156.589694
6pointTrailingMovingAverage	1283.927428
9pointTrailingMovingAverage	1346.278315
ARIMA	1296.575101
SARIMA	343.311632
Manual ARIMA	1380.758457
Manual SARIMA	546.011977

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.

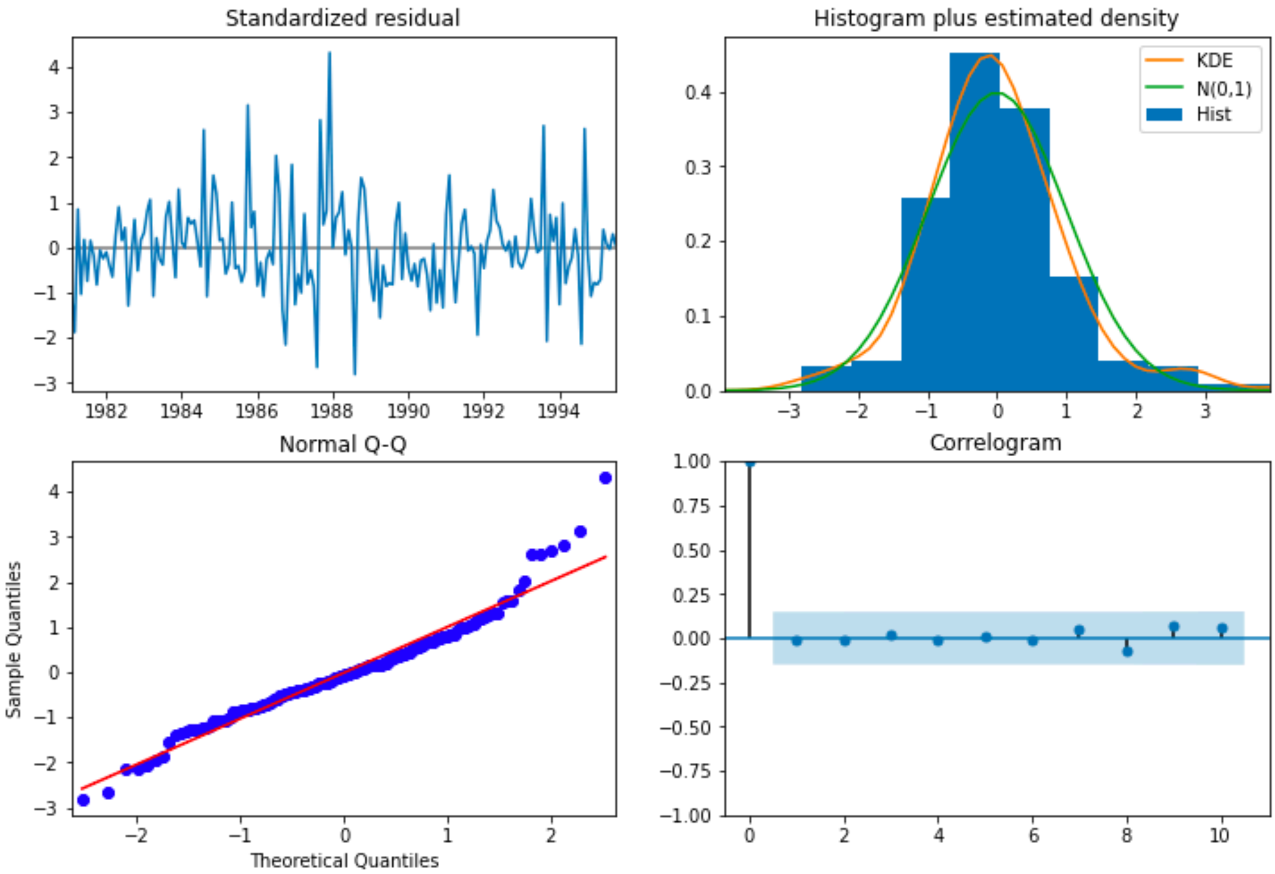
The lowest RMSE score is of Auto SARIMA

```
full_model = sm.tsa.statespace.SARIMAX(df,
                                        order=(6, 1, 1),
                                        seasonal_order=(2,
1, 1, 12),

enforce_stationarity=True).fit()
```

Statistical Summary of SARIMA -

```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                           Sparkling   No. Observations:                  187
Model:             SARIMAX(6, 1, 1)x(2, 1, 1, 12)   Log Likelihood               -1281.642
Date:                             Fri, 06 Nov 2020   AIC                           2585.283
Time:                                     07:45:17   BIC                           2620.033
Sample:                                 01-01-1980   HQIC                          2599.380
                                      - 07-01-1995
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.0870      0.088      0.987      0.324      -0.086       0.260
ar.L2         -0.0764      0.092     -0.827      0.408      -0.258       0.105
ar.L3         -0.0096      0.082     -0.117      0.907      -0.170       0.151
ar.L4         -0.0780      0.089     -0.876      0.381      -0.253       0.097
ar.L5         -0.0599      0.093     -0.642      0.521      -0.243       0.123
ar.L6         -0.0140      0.103     -0.137      0.891      -0.215       0.187
ma.L1         -0.9125      0.051    -17.785      0.000      -1.013      -0.812
ar.S.L12       0.0621      0.182      0.342      0.732      -0.294       0.418
ar.S.L24       0.0224      0.138      0.163      0.871      -0.247       0.292
ma.S.L12      -0.6108      0.172     -3.561      0.000      -0.947      -0.275
sigma2       1.37e+05   1.26e+04     10.848      0.000    1.12e+05    1.62e+05
===================================================================================
Ljung-Box (Q):                       18.87   Jarque-Bera (JB):                53.15
Prob(Q):                              1.00   Prob(JB):                         0.00
Heteroskedasticity (H):               1.08   Skew:                             0.68
Prob(H) (two-sided):                  0.78   Kurtosis:                         5.34
===================================================================================
```

Statistical Plots -



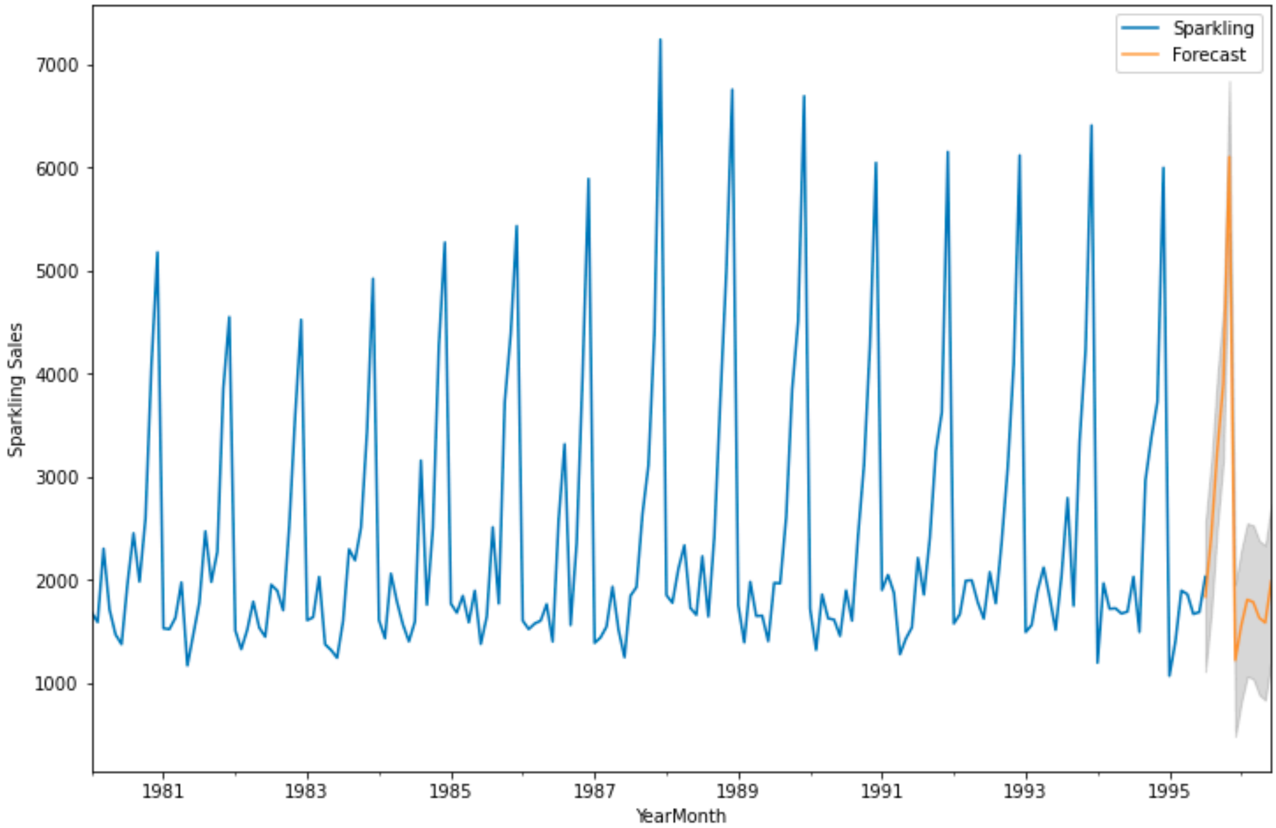
Some Inference from the Plots above -

1. Qq plots shows some linear trend. This shows that the residuals are normally distributed.

2. The KDE plot of the residuals is almost similar with the normal distribution.

RMSE on test data - 534.4678931903776

Plotting the forecast along with the confidence band -



1. It seems that the sales of the Sparkling wine is good as compared to previous year.
2. The upcoming sales are closely matching with the sales of the year 1986-1987.

10) Comment on the model thus built and report your findings and suggest the measures that the company should be taking for future sales.
Please explain and summarise the various steps performed in this project. There should be proper business interpretation and actionable insights present.

From the model build above we have following findings:

1. Moving Average part is quite significant than the Auto Regressive part.

2. The KDE plot of the residuals is almost similar with the normal distribution.

3. The qq-plot on the bottom left shows that the ordered distribution of residuals (blue dots) follows the linear trend of the samples taken from a standard normal distribution with N(0, 1). Again, this shows that the residuals are normally distributed.

4. The residuals over time (top left plot) don't display any obvious seasonality and appear to be white noise. This is confirmed by the autocorrelation (i.e. correlogram) plot on the bottom right, which shows that the time series residuals have low correlation with lagged versions of itself.

From this we can conclude that the residuals are random with no information and our model produces a satisfactory fit that could help us understand our time series data and forecast future values. It seems that our SARIMA model is working fine.

From business point of view we can see from the forecast plot above, the predicted sales of 'Sparkling' wine of future 12 months seem to be good, there is seasonal effect which can cause sales go to up as well.In order to boost more profit -

1. We can offer discounts to the customers in the festive season.

2. We can give free samples to the customers for tasting and in return hope for feedback which in return helps us to gain information about what is wrong with this wine.

3. Depending on the taste feedback we can add flavours.

4. As we know different wines are for different occasions we can also provide the same knowledge to the customers.

5. Social media is a powerful tool to promote our product.