PGP DSBA

# TIME SERIES FORECASING

JOTINDER SINGH MATTA

# ROSE WINE SALES

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.

In this section we will first focus on the Rose Wine data set. Subsequent section will be dedicated to Sparkling wine data set from the same wine company.

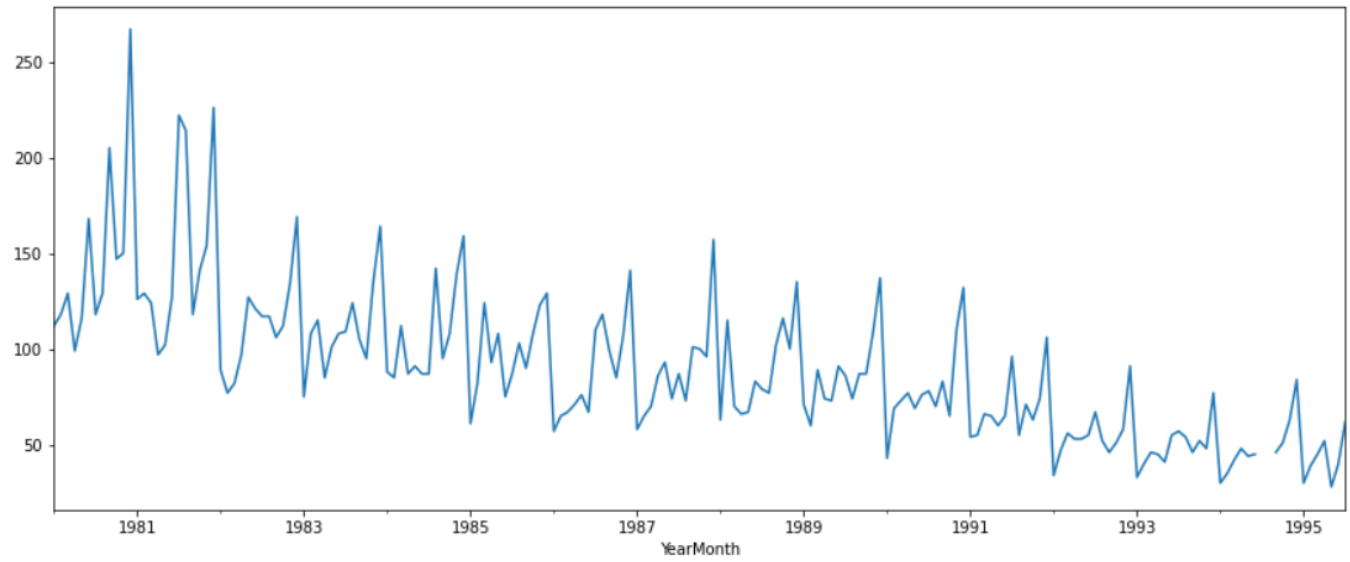
The report has been divided into two parts to keep things easy to understand, as the same set of questions need to be answered twice, each time with a different data set.

# 1. READ THE DATA AS AN APPROPRIATE TIME SERIES DATA AND PLOT THE DATA.

The data set was read using pandas library's read_csv function. Parse date feature was used to read this time series data. index_col was selected as the 'YearMonth' column from the given csv file. Data was hence ingested.

Post ingesting the data, we renamed the column 'Rose' to 'Sales' to better indicate its intended purpose. Along with it, we also extracted the Month and Years as columns from the given index values. Below is the head of the ingested data after above operations were performed. Also we plotted the data using matplotlib python library, this too has been shown below.

| YearMonth | Sales | Year | Month |
|---|---|---|---|
| **1980-01-01** | 112.0 | 1980 | 1 |
| **1980-02-01** | 118.0 | 1980 | 2 |
| **1980-03-01** | 129.0 | 1980 | 3 |
| **1980-04-01** | 99.0 | 1980 | 4 |
| **1980-05-01** | 116.0 | 1980 | 5 |



# 2. PERFORM APPROPRIATE EXPLORATORY DATA ANALYSIS TO UNDERSTAND THE DATA AND ALSO PERFORM DECOMPOSITION.

We set off to perform Exploratory data analysis (EDA) once the data was properly ingested. However immediately upon looking at the data plot, we could see there were certain missing values in the series. We found the values for the months of July & August were missing for the year 1994.

| YearMonth | Sales | Year | Month |
|---|---|---|---|
| 1994-07-01 | NaN | 1994 | 7 |
| 1994-08-01 | NaN | 1994 | 8 |

Since this was time series analysis, and we could not do proper analysis with missing data, we decided to first impute the missing values and then proceed with the rest of the EDA.

We tried multiple approaches to impute the data, these were as below.

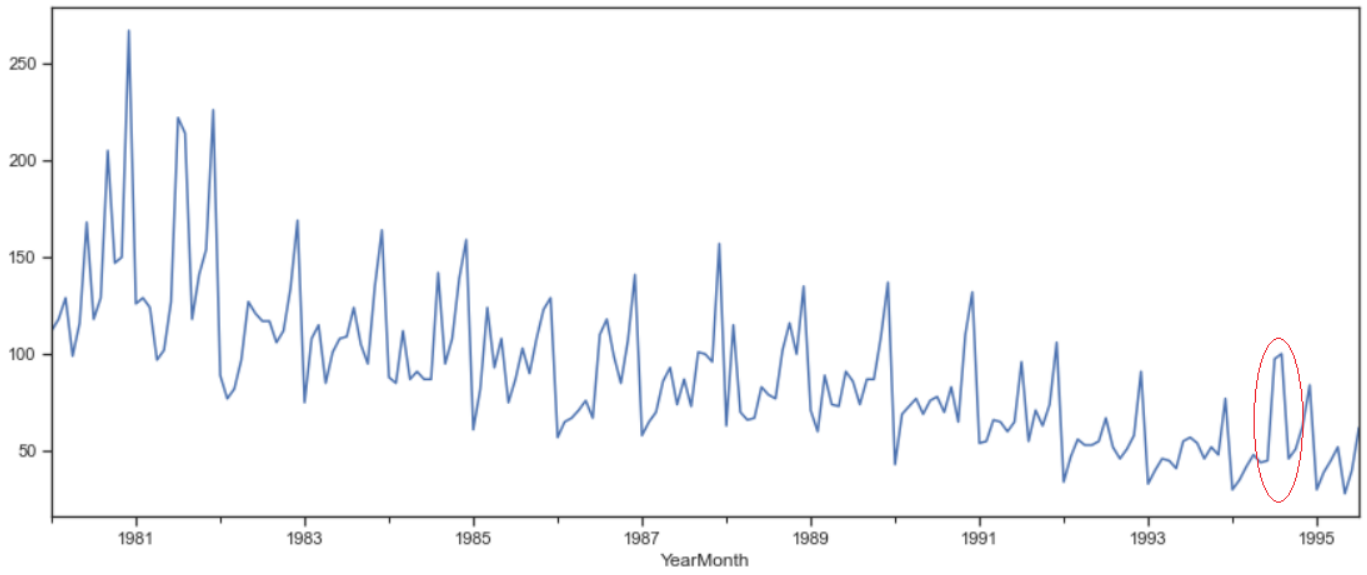
1) Full Mean
2) Interpolate - Linear
3) Interpolate - Spline
4) Mean - Before & After

We will discuss each of these approaches and then decide on one of the approaches to actually impute the values. We will further elaborate on our reasoning to select the choosen technique.

# 1) Full Mean

In this approach we added the values for all the 7th months for all the years and took its mean to impute the 7th month values for 1994. Similar activity was done for 8th month. This gave us below values, Also we plotted the series after interpolation.

| YearMonth | Sales | Year | Month | Sales_Full_Mean | Sales_Linear | Sales_Spline | Sales_Before_After_Mean |
|---|---|---|---|---|---|---|---|
| **1994-07-01** | NaN | 1994 | 7 | 97.466667 | NaN | NaN | NaN |
| **1994-08-01** | NaN | 1994 | 8 | 100.142857 | NaN | NaN | NaN |



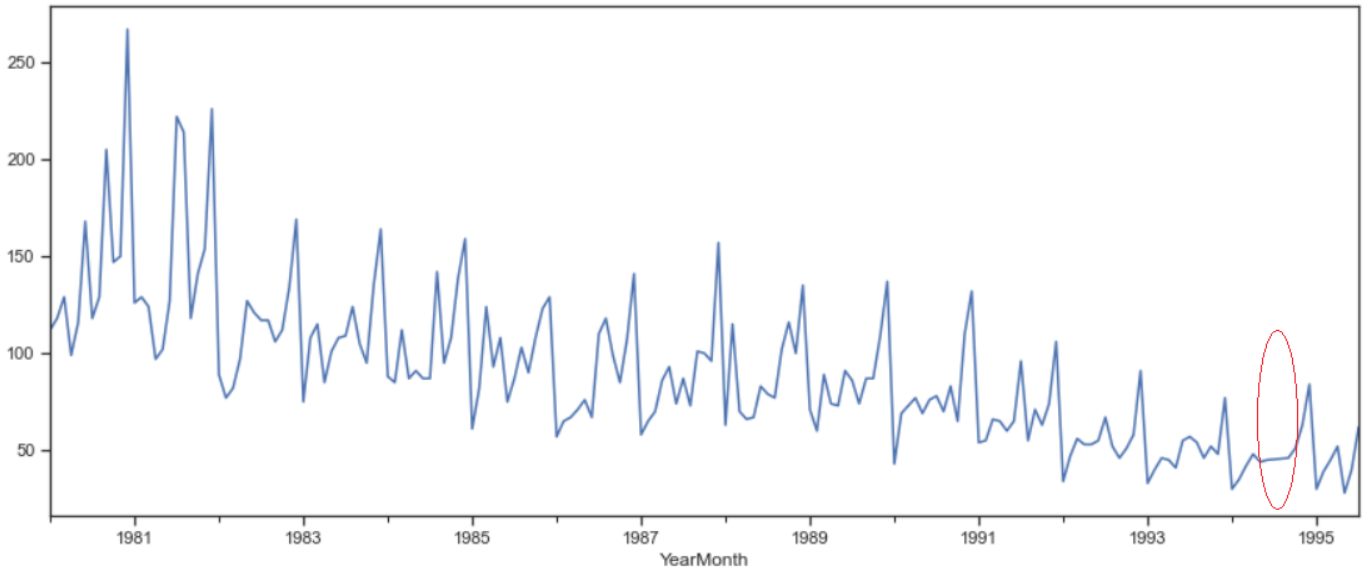

From the above plot, we can see the abnormally high spike being created, which has been highlighted with by the red circle. This seems to be out of line from the recent trends, as no such high peak has been observed in recent past, at this time of the year.

Hence we reject this interpolation method, due to its huge divergence from the recent trends.

## 2) Interpolate - Linear

In this approach we used pandas interpolate function, which tried to impute the missing values based on the selected method. We first selected the method to be 'Linear'. This basically tries to make a straight line across the missing values in the series and assign value to data points based on the mathematical equation of the line thus obtained. This gave us below values. Series was also plotted after interpolation, to see its effect on the series.

| YearMonth | Sales | Year | Month | Sales_Full_Mean | Sales_Linear | Sales_Spline | Sales_Before_After_Mean |
|---|---|---|---|---|---|---|---|
| **1994-07-01** | NaN | 1994 | 7 | 97.466667 | 45.333333 | NaN | NaN |
| **1994-08-01** | NaN | 1994 | 8 | 100.142857 | 45.666667 | NaN | NaN |



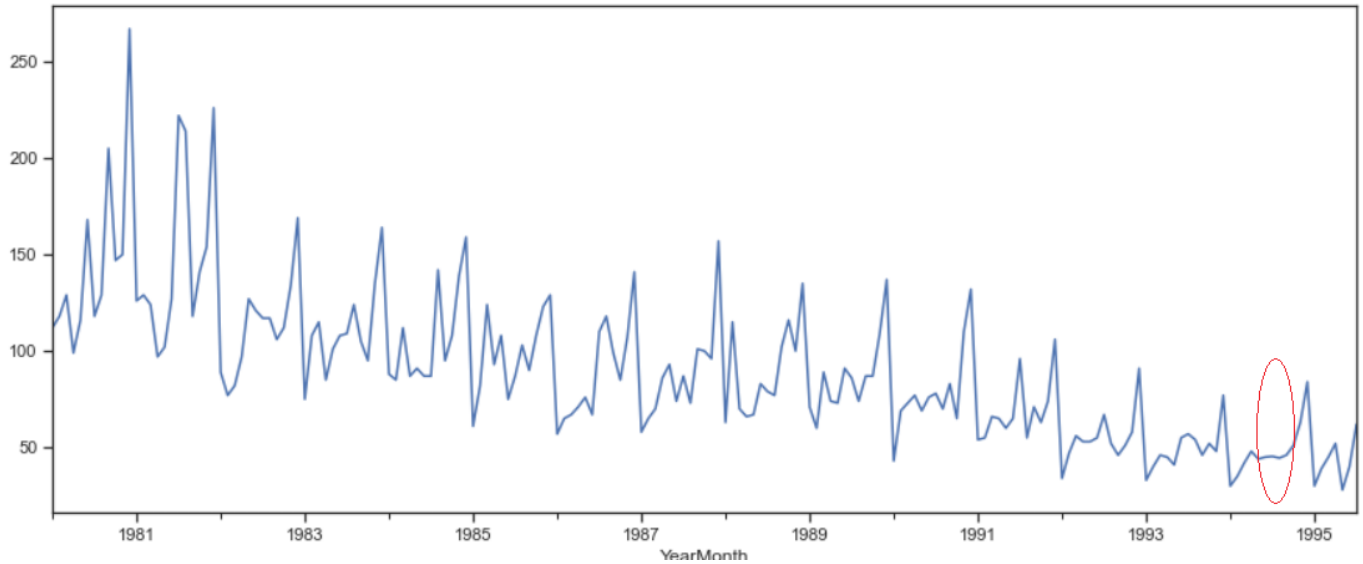

Here we can see a flat line has been drawn across the missing points. Though this is much better than the first approach, but comparing the values to the previous years sales figures from the months of July and August, we can see the values are on the lower side. This is a good contender for the imputation, however we will select this only if other approaches are no good.

Hence these values were kept as backup at this juncture in our analysis.

## 3) Interpolate - Spline

Here too we used interpolate function from pandas, however method chosen this time was spline of order 2. Spline is a peicewise polynomial function. Below were the values obtained using this approach. Again the series was also plotted to get a better idea.

| YearMonth | Sales | Year | Month | Sales_Full_Mean | Sales_Linear | Sales_Spline | Sales_Before_After_Mean |
|---|---|---|---|---|---|---|---|
| **1994-07-01** | NaN | 1994 | 7 | 97.466667 | 45.333333 | 45.34978 | NaN |
| **1994-08-01** | NaN | 1994 | 8 | 100.142857 | 45.666667 | 44.51237 | NaN |



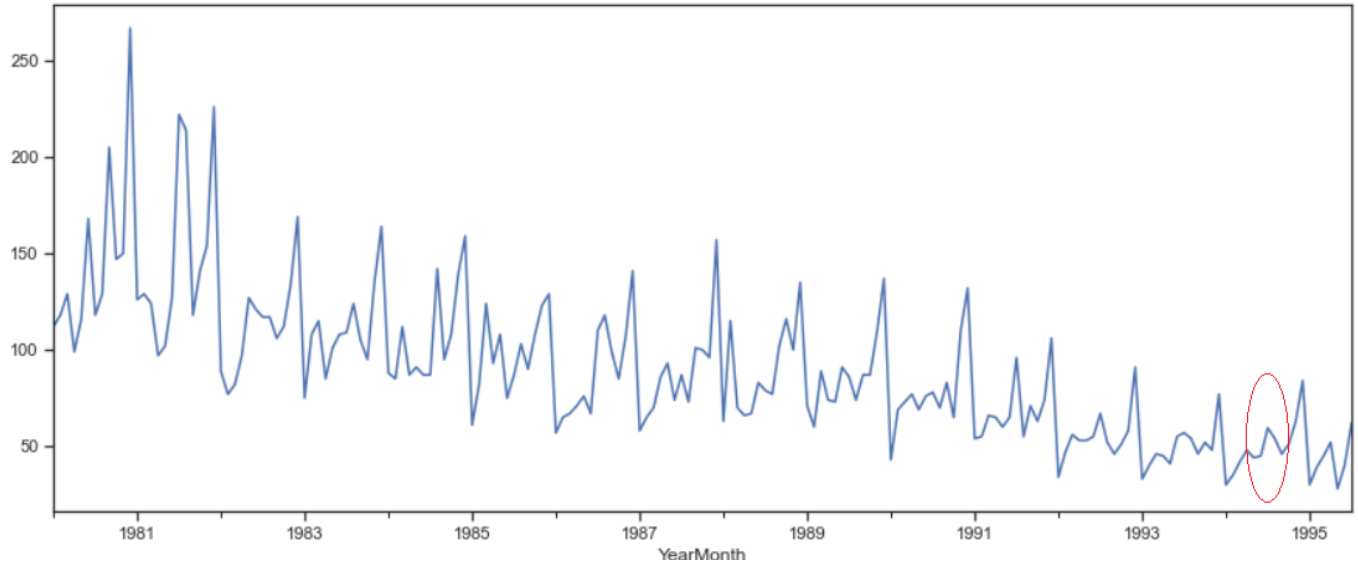

Here too we can see the values are very similar to the second approach. The interpolated line is not an exact straight line, but very similar to it. Looking at the values, again they look at little on the lower side compared to previous years July and August sales figures.

Again, at this point in our analysis, we kept it as a backup option along with second method, these would be used only in case the last method does not yield better results.

## 4) Mean - Before & After

In this approach, instead of taking means for the 7th months across all the years, we just took mean of the 7th months values from a year before and a year after the missing value. Similar steps were taken for 8th month. This allowed us to get figures, which were more in line with the current trends and were not influenced highly by the values from past. Below were the values obtained. Also the series was plotted to gain more insights.

| YearMonth | Sales | Year | Month | Sales_Full_Mean | Sales_Linear | Sales_Spline | Sales_Before_After_Mean |
|---|---|---|---|---|---|---|---|
| **1994-07-01** | NaN | 1994 | 7 | 97.466667 | 45.333333 | 45.34978 | 59.5 |
| **1994-08-01** | NaN | 1994 | 8 | 100.142857 | 45.666667 | 44.51237 | 54.0 |



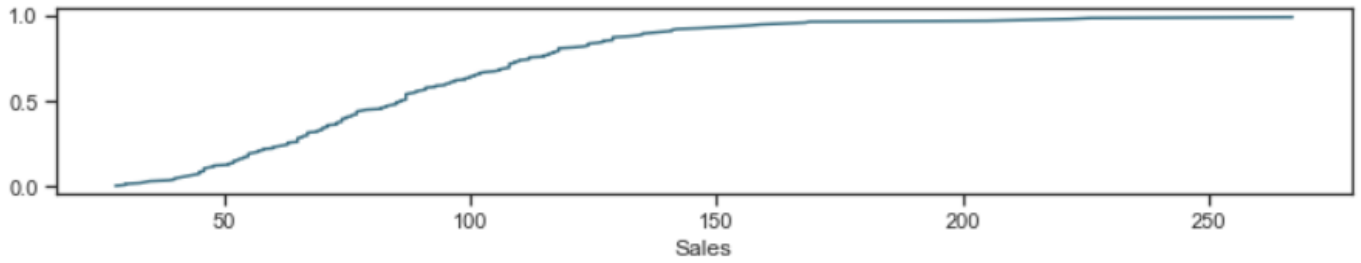

Here we can see a small peak is being formed just before the bigger peak for the year (December Peak). Looking at the recent trends, we could observe there have indeed been a smaller peak occurring in the months of July and August before the seasons peak in December.

Also comparing the figures with the July and August figures from recent years, we could see the values made much more sense than any of the approaches used above.

Hence we used this technique to impute the missing data and then proceeded with the remaining EDA.

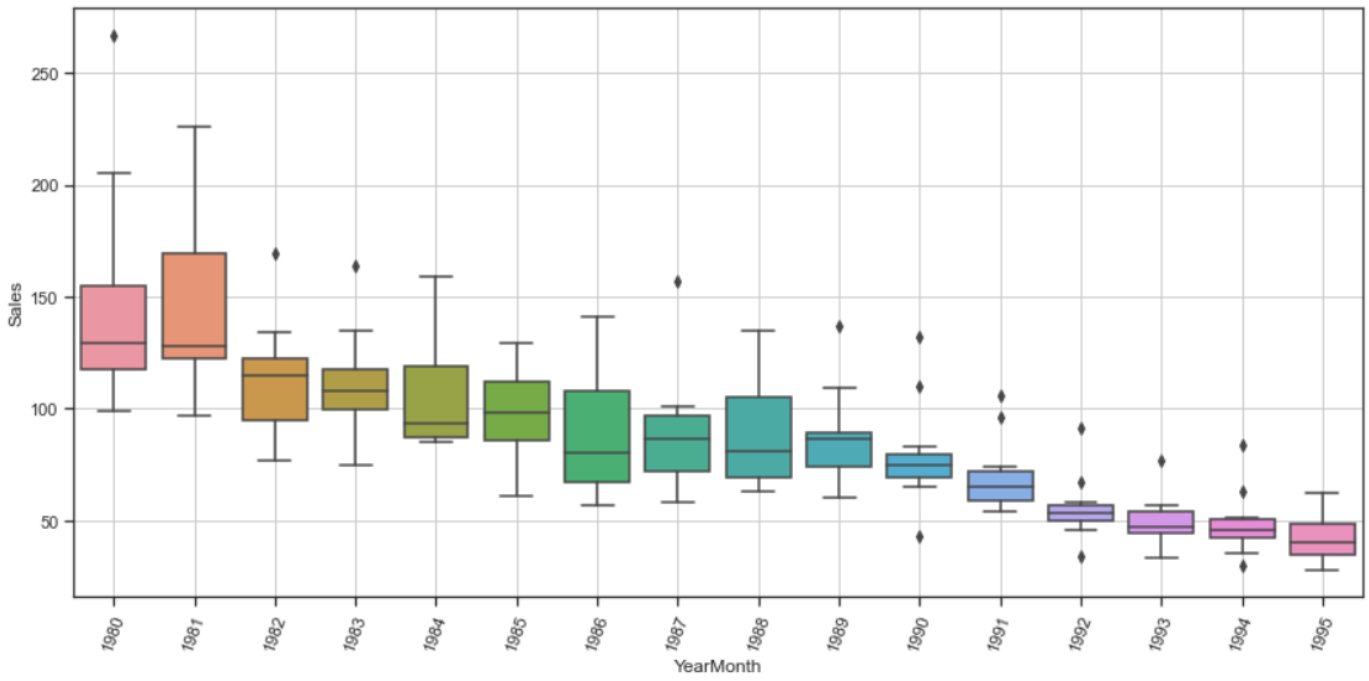
## ECF Plot

An ECF plot was made for the given series. This plot shows that for the most months i.e. more than 50% of the months the sale has been less than 100. Peak value being around 250, however almost 90% of the times the value has been less 150. This graph shows how the data is distributed.



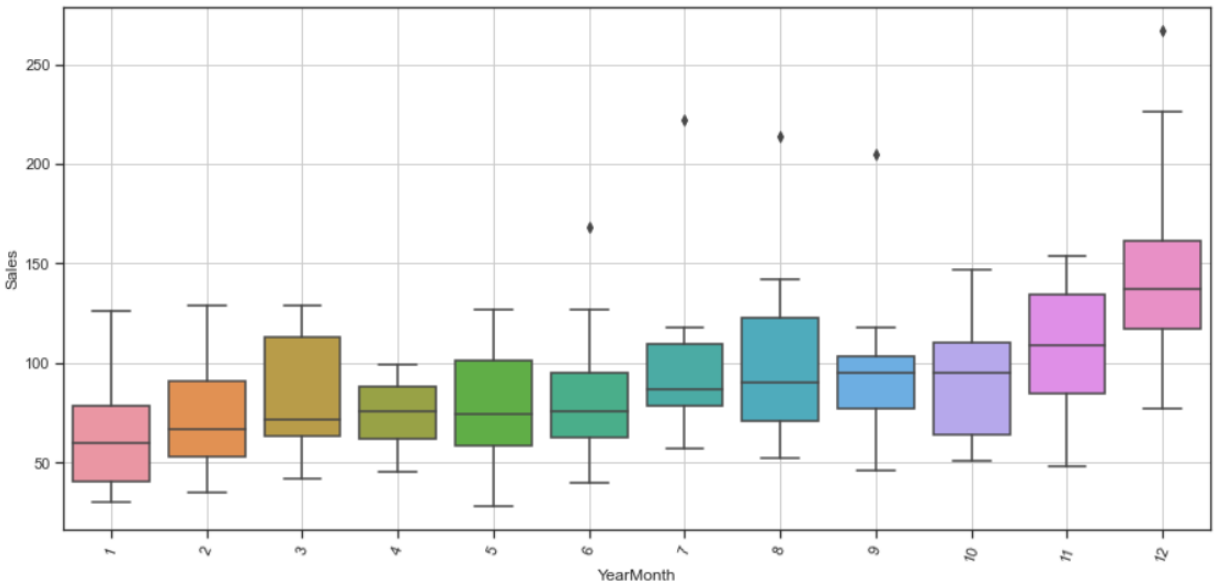

## Box Plot - Yearly

An yearly box plot, clearly indicates a downward trend in the sales of Rose wine for the company. The sales peaked out in 1981 and since then are on a steady decline. Most of the outlier being show in the graph are for the month of December when peak sales are observed. However these are genuine values and do not need to be treated in any way.



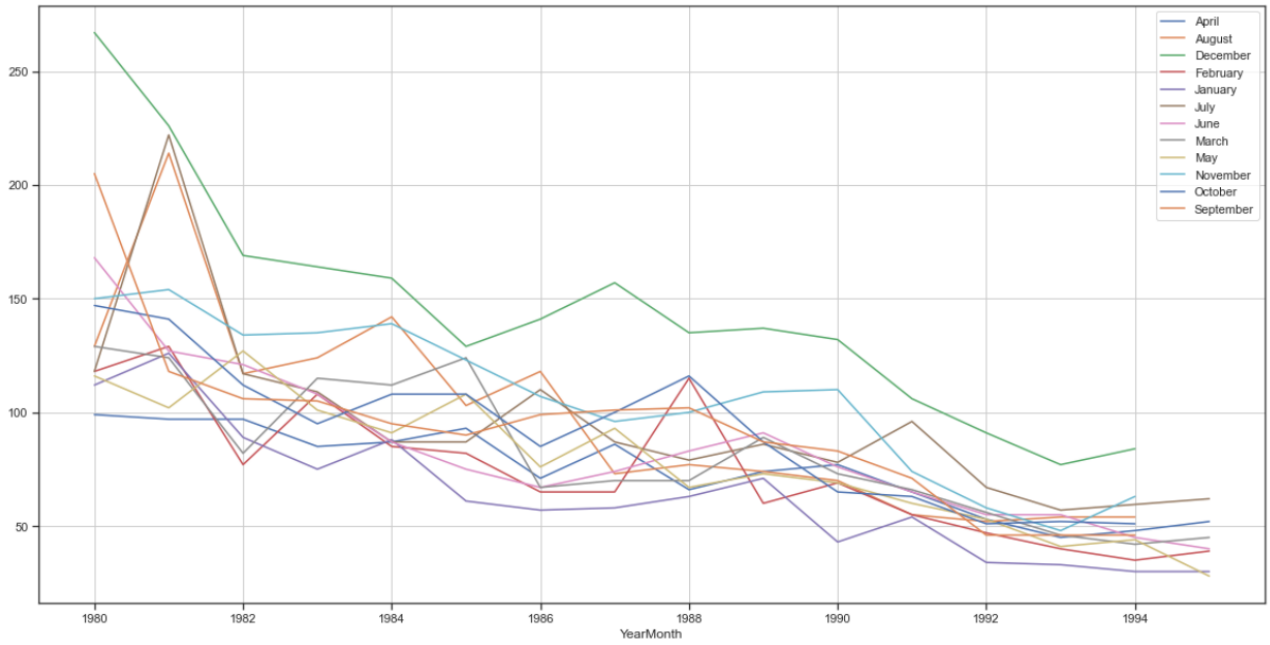
## Box Plot - Monthly

We also plotted a box plot, depicting the sales pattern for each month over the years. It is very clear that peak sales are observed in the month of December, while Sales plummet in the month of January to its lowest level. There has been a slight increase in the sales in the months of July and August, before dipping in Sepetember at times and then picking up again in subsequent months.



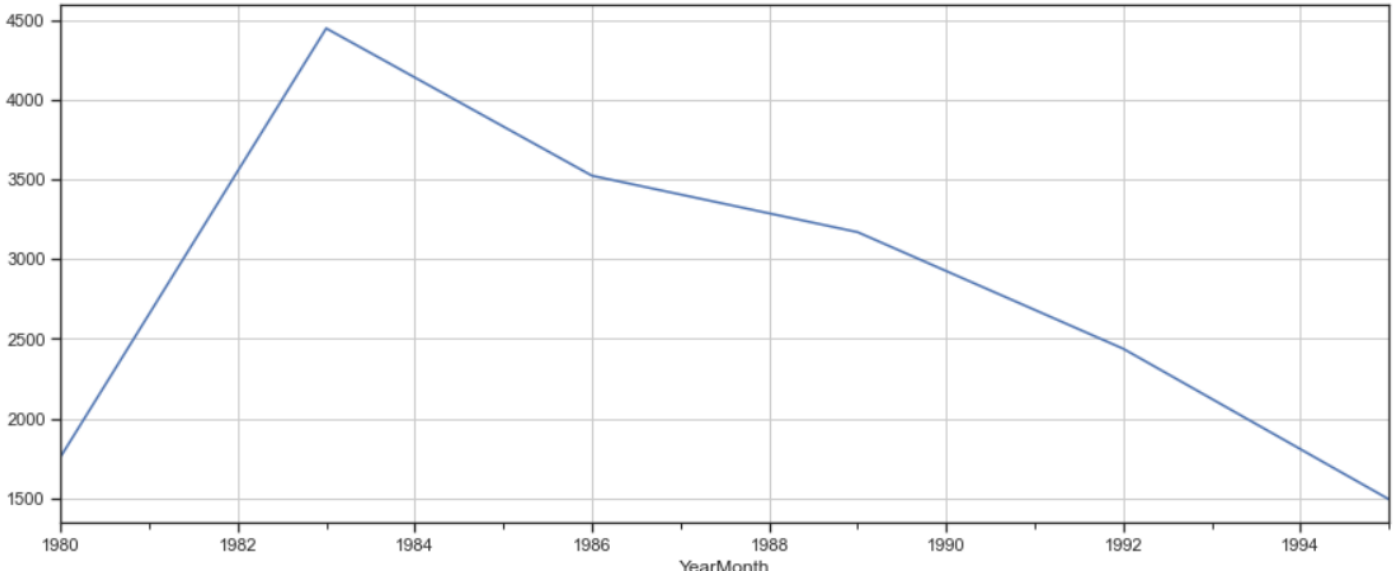

## Monthly Sales Across Years

A chart was plotted to further corroborate the above findings. This gives a clearer view of sales across different months of the year over the 15 years period i.e. 1980 to 1995.



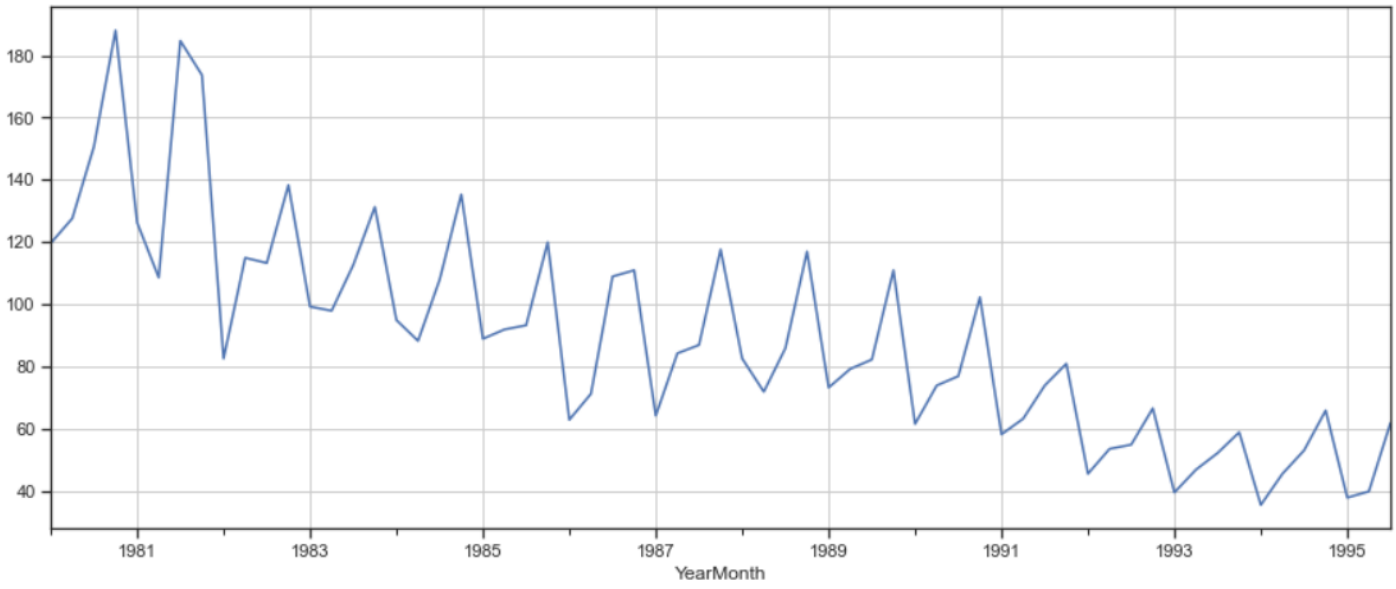
## 3 - Year Sales Graph

Using the up-sampling techniques of pandas library we up-sampled the yearly data to 3 year data and plotted the below graph to understand if there is any broader trend that we might be missing. Looking at below graph, we can infer that the sales peaked at the beginning, however post that it has been on a constant decline. This corroborates with our earlier observations.



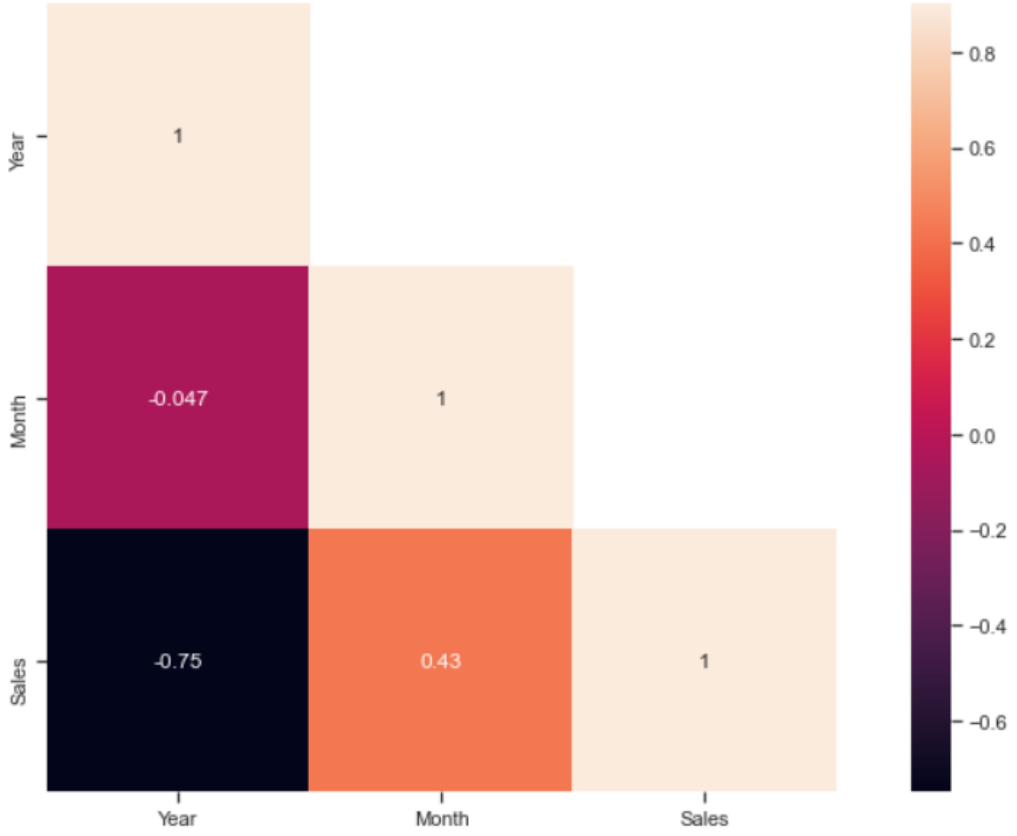

## Quarterly Sales Graph

Using down sampling techniques of pandas we down sampled the data to quarterly range and found that the peak quarter of sales is the last quarter of the year, which is again primarily due to the peak sales in December.



## Correlation Heat map

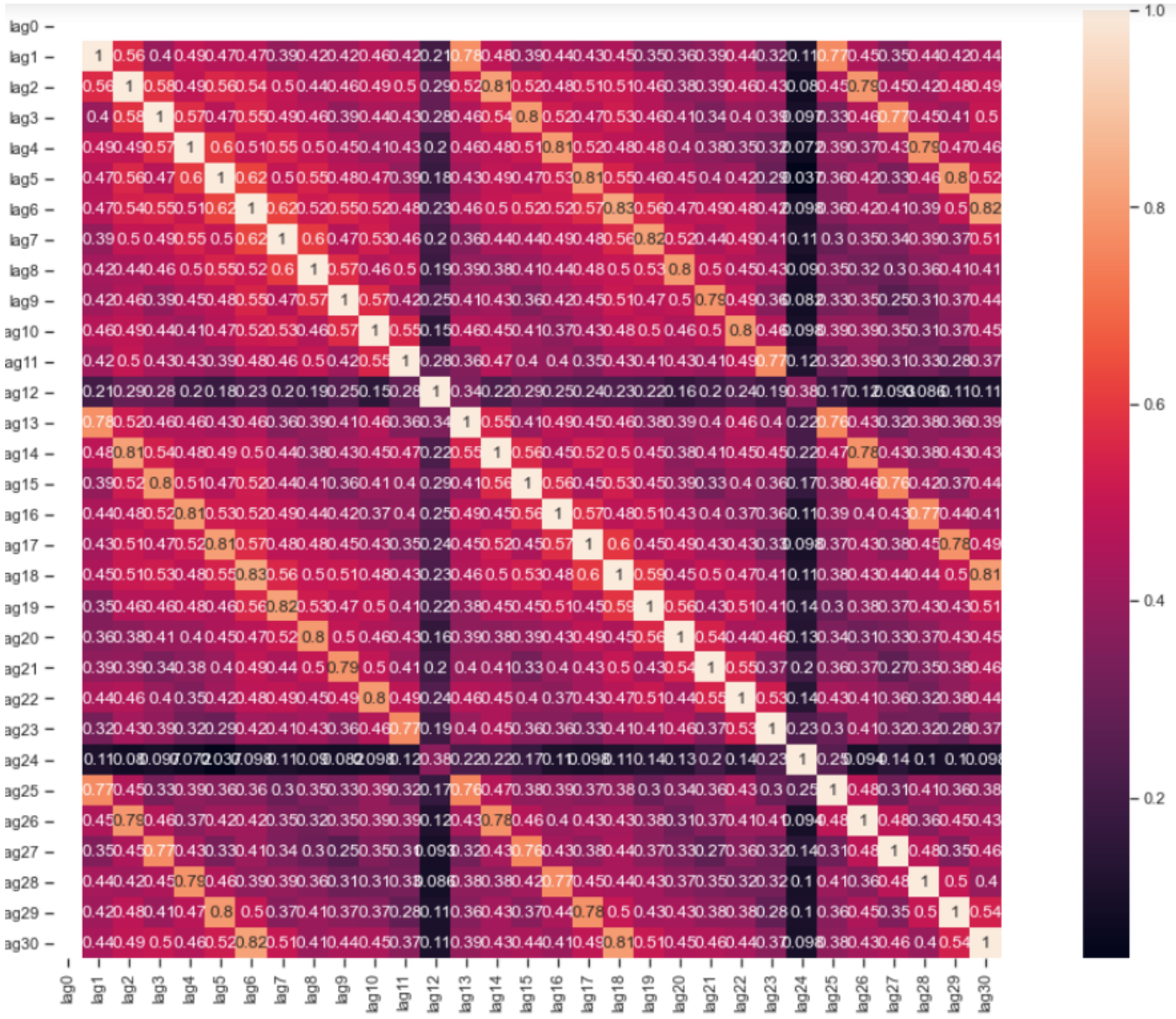
We plotted a heat map of sales with month and year columns. We could see while there was little correlation between Sales and the Years data, there significantly more correlation between the month and Sales columns. Clearly indicating a seasonal pattern in our Sales data. Certain months have higher sales, while certain months have lesser.




Correlation between months and Sales also makes sense. During December due to festive season, it is expected that wine sales in general would increase. Again during the month of January due to extreme winters, sales might be on the lower side. Hence the sales of wine are somewhat correlated to months.

## Correlation Heat map for Lags

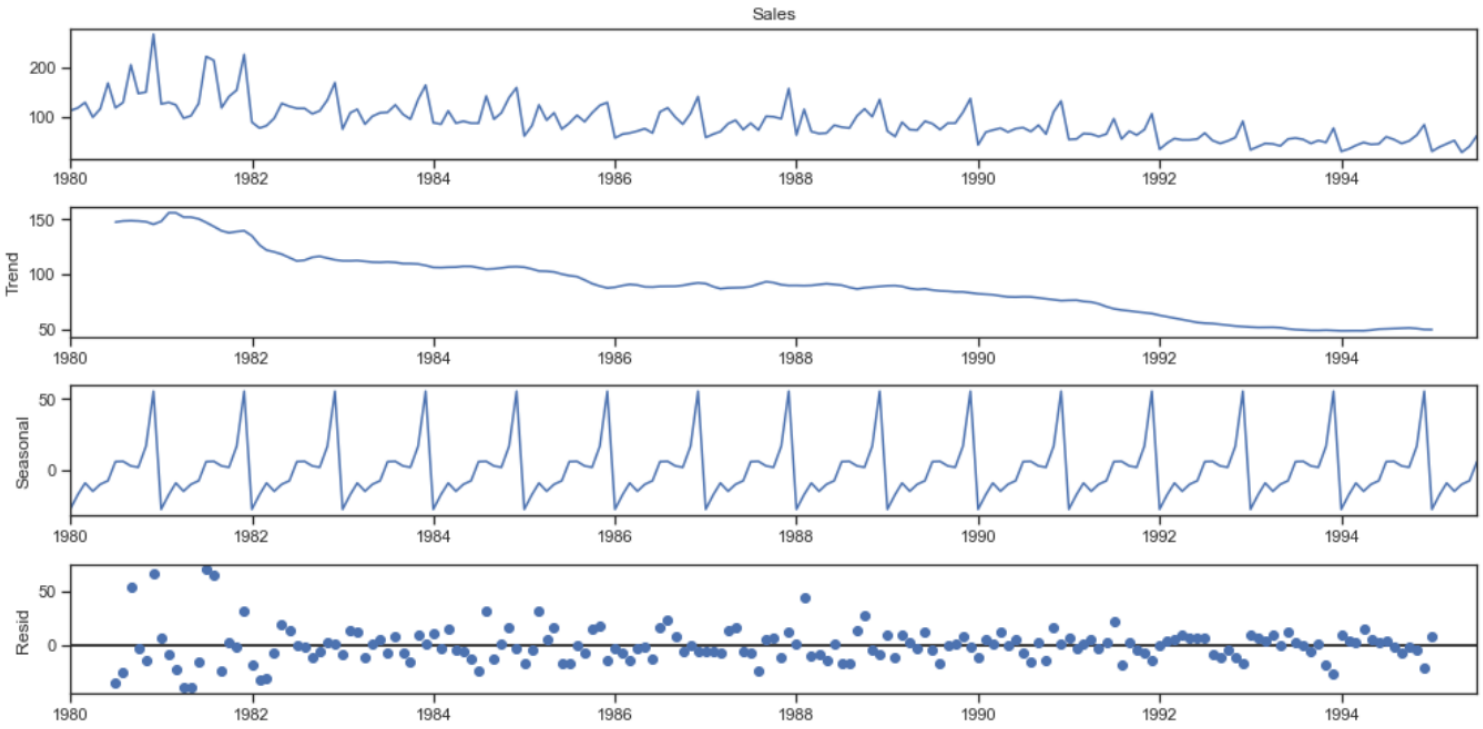
We calculated multiple lag values (30 values) and a heat map of the lags with each other were plotted. From the plot we can see a clear correlation between lag1 and lag13, a difference of 12. Again correlation between lag2 and lag14 is high, a difference of 12. This indicates a yearly seasonality or a seasonality of 12.




The black lines being observed in the heat map, indicate correlation between january and December months which are in stark comparison to each other. Sales peak in December while they plummet in January hence a very low correlation is present which is represented by a darker color.

## Decomposition - Additive

We used seasonal_decompose method from statsmodels library in python to decompose the given series into Trend, Seasonality and Residue. We first decomposed the time series using additive approach. Below are the decomposition plots.



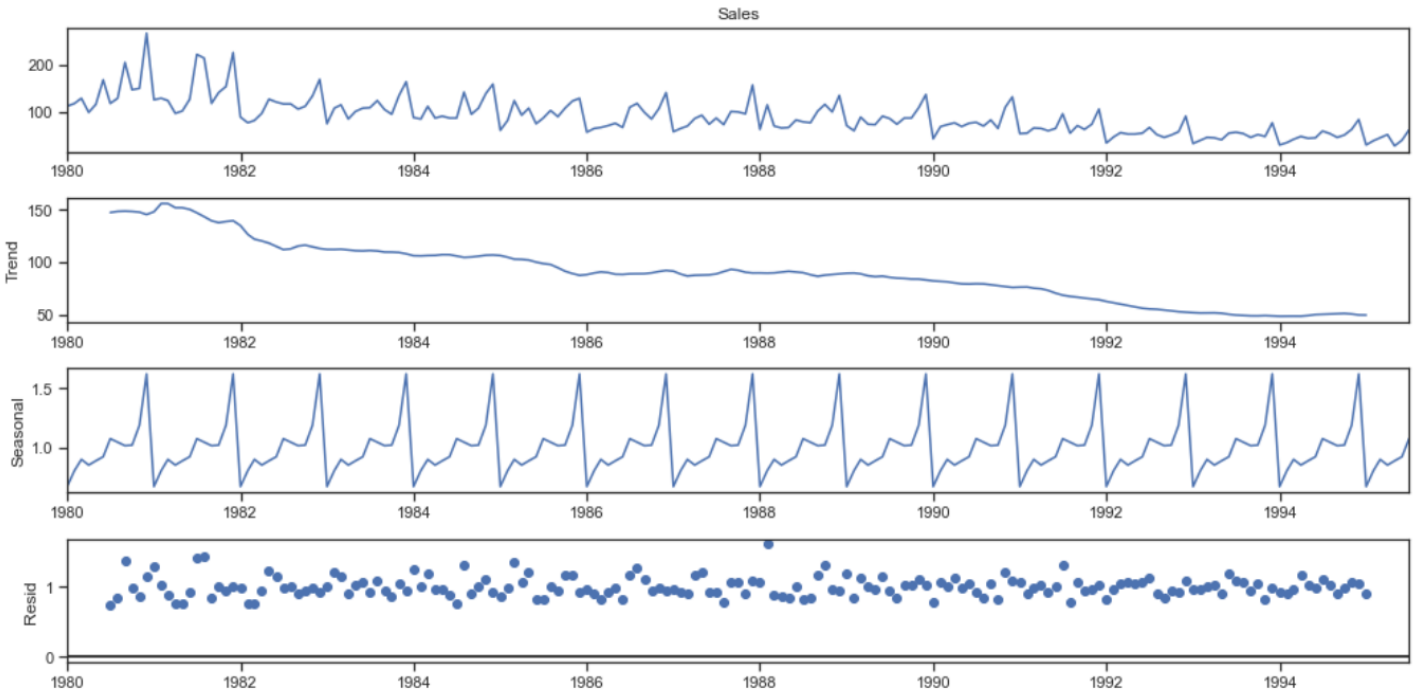

Looking at the above decomposition, we can clearly see a trend which is downward in nature. Also a clear seasonality is present. It seems the peak is reached towards the end of the year, which corroborates with our earlier findings.

The residue is quite spread out and is not forming a straight line. Hence we will further look for decomposition of the series using multiplicative approach.

There are clear trend and seasonality components to this time series data.

## Decomposition - Multipicative

We used seasonal_decompose method from statsmodels library in python to decompose the given series into Trend, Seasonality and Residue. We used multiplicative approach this time. Below are the plots.




Looking at the above decomposition we can once again see a clear trend component which is downward in nature. Also a very clear seasonality trend is noticed even with multiplicative approach.

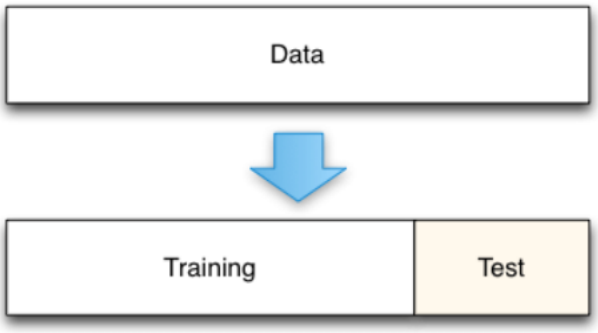
The residue in this case is also not forming a perfect straight line but it is better than additive model.

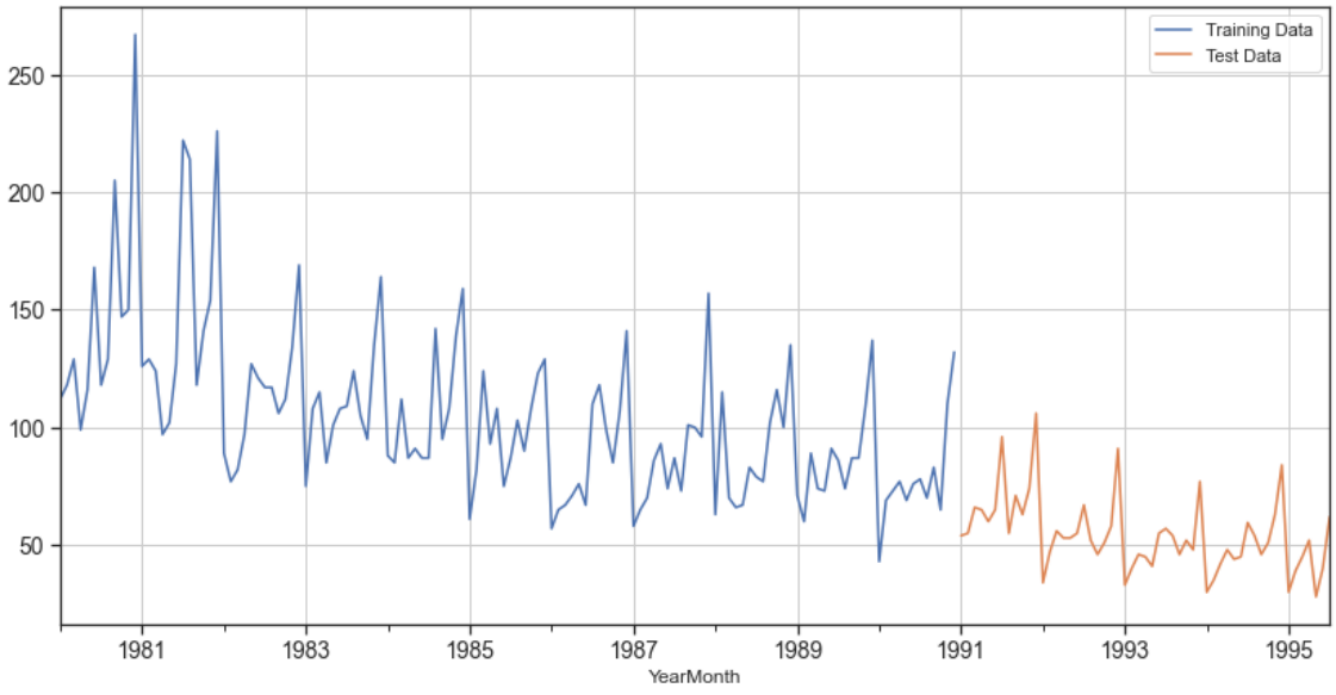
To decide between multiplicative and additive, we take into account lower range of residual, which in case of multiplicative is 0 to 1, while for additive is 0 to 50. Hence we select the multiplicative model owing to a more stable residual plot and lower range of residuals.

# 3. SPLIT THE DATA INTO TRAINING AND TEST. THE TEST DATA SHOULD START IN 1991.

The data was split into train and test data sets, so that the machine learning models could be trained on the training set and the models could be further evaluated using the test data set.

As per the instructions given in the project we have split the data, around 1991. With training data from 1980 to 1990 December. Test data starts from the first month of January 1991 till the end.



## 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.

We started with simple models and gradually moved towards more and more complex models. We created below models as part of this question.

**Models Built:-**

1) Linear Regression Model
2) Naive Bayes Model
3) Simple Average Model
4) Moving Average Model - Rolling Window 2
5) Moving Average Model - Rolling Window 4
6) Moving Average Model - Rolling Window 6
7) Moving Average Model - Rolling Window 9
8) Simple Exponential Smoothing - AutoFit
9) Simple Exponential Smoothing - Using For Loop
10) Double Exponential Smoothing (Holt's Model) - AutoFit
11) Double Exponential Smoothing (Holt's Model) - Using For Loop
12) Tripple Exponential Smoothing (Holts - Winter Model) - AutoFit
13) Tripple Exponential Smoothing (Holts - Winter Model) - Using For Loop.

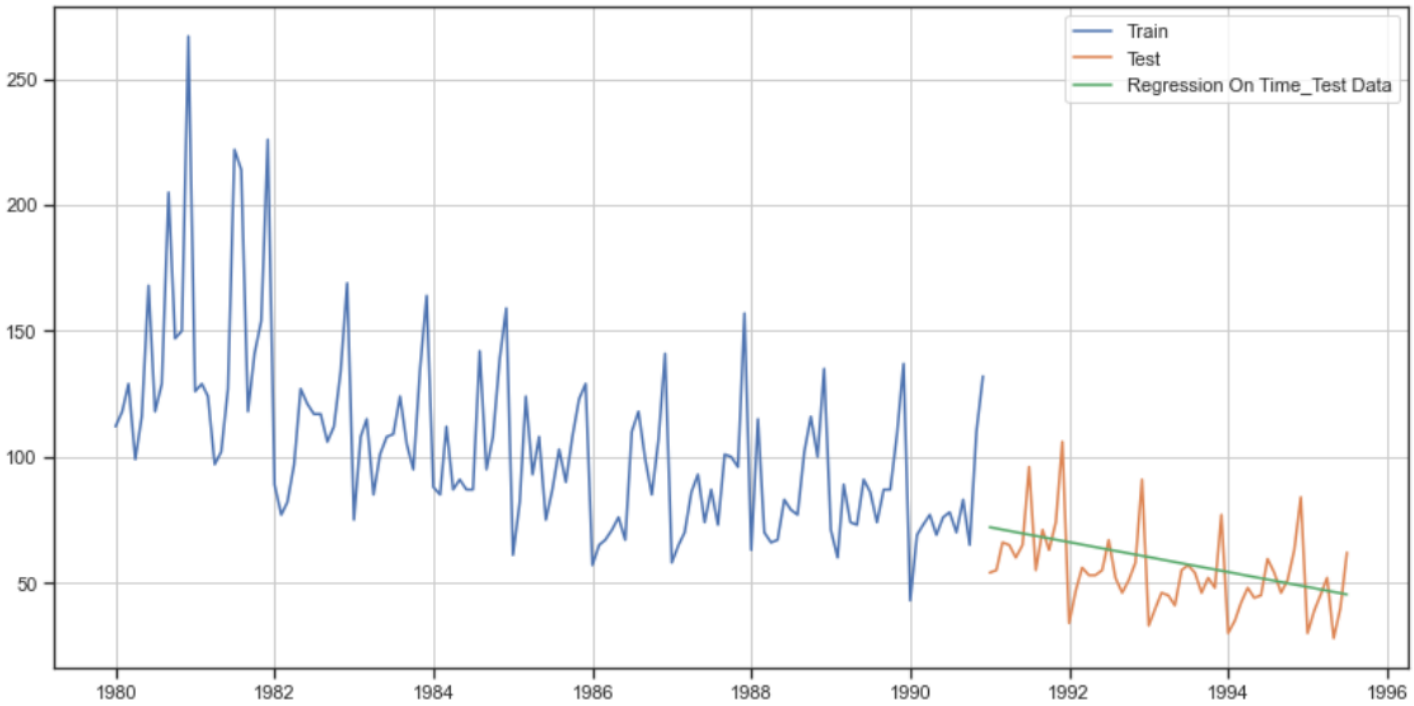
A total of 13 different models were created as part of this question and all of these were evaluated on the RMSE performance metric and compared at the end.

We also plotted prediction with actuals to visually see how accurate the results of each model were.

Each of these models will be discussed in subsequent pages.

## 1) Linear Regression Model

We started out with the simplest model, which was the Linear regression using sklearn library.This model tries to fit all the training points on a straight line and interpolate the line over the range of test vaues to predict the test values.



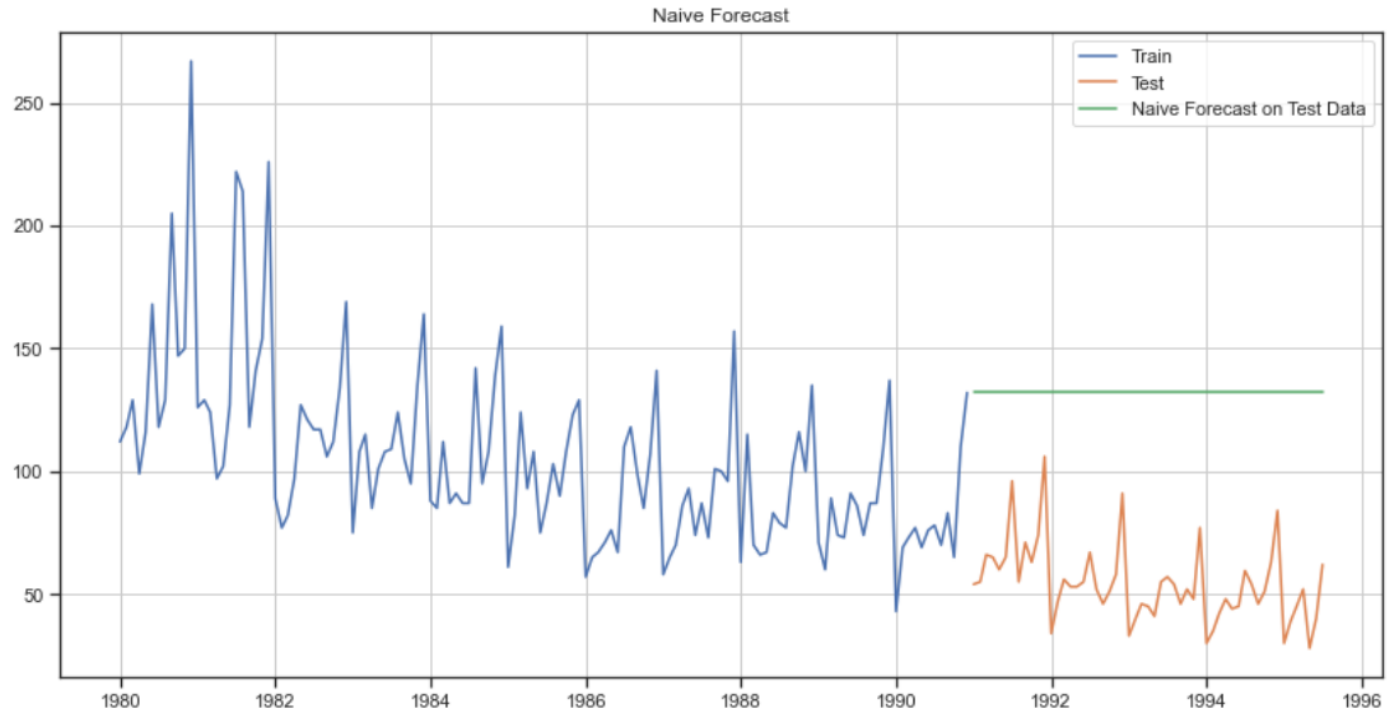

The green line indicates the predictions made by the model, while the orange values are the actual test values. It is clear the predicted values are very far off from the actual values, but this was expected as this is one of the simplest models.

Model was evaluated using the RMSE metric. Below is the RMSE calculated for this model.

| | Test RMSE |
|---|---|
| **RegressionOnTime** | 15.278158 |

## 2) Naive Bayes Model

Following the linear model, we created the naive bayes model, which as the name suggests is very naive in nature, as it assumes the last observed value to be the future values. All the future values will be the same as the last observed value.



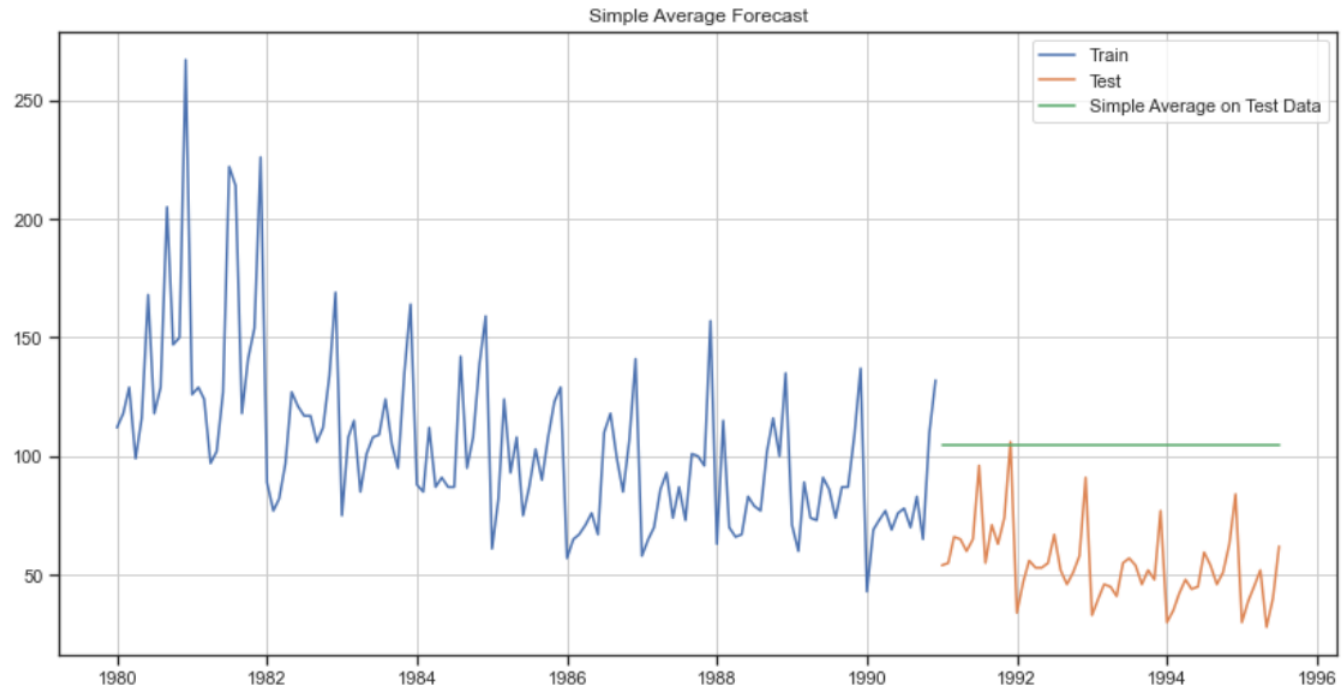

The green line indicates the predictions made by the model, while the orange values are the actual test values. It is clear the predicted values are very far off from the actual values, but once again this was expected from a naive model as it is just portraying the last observed value as the future values.

Model was evaluated using the RMSE metric. Below is the RMSE calculated for this model.

| | Test RMSE |
|---|---|
| NaiveModel | 79.304391 |

## 3) Simple Average Model

A simple average model takes the mean value across the different years and then plots a straight line for the future values. It is also very simple model and does not provide good predictions, however we still went ahead for the sake of exploration.



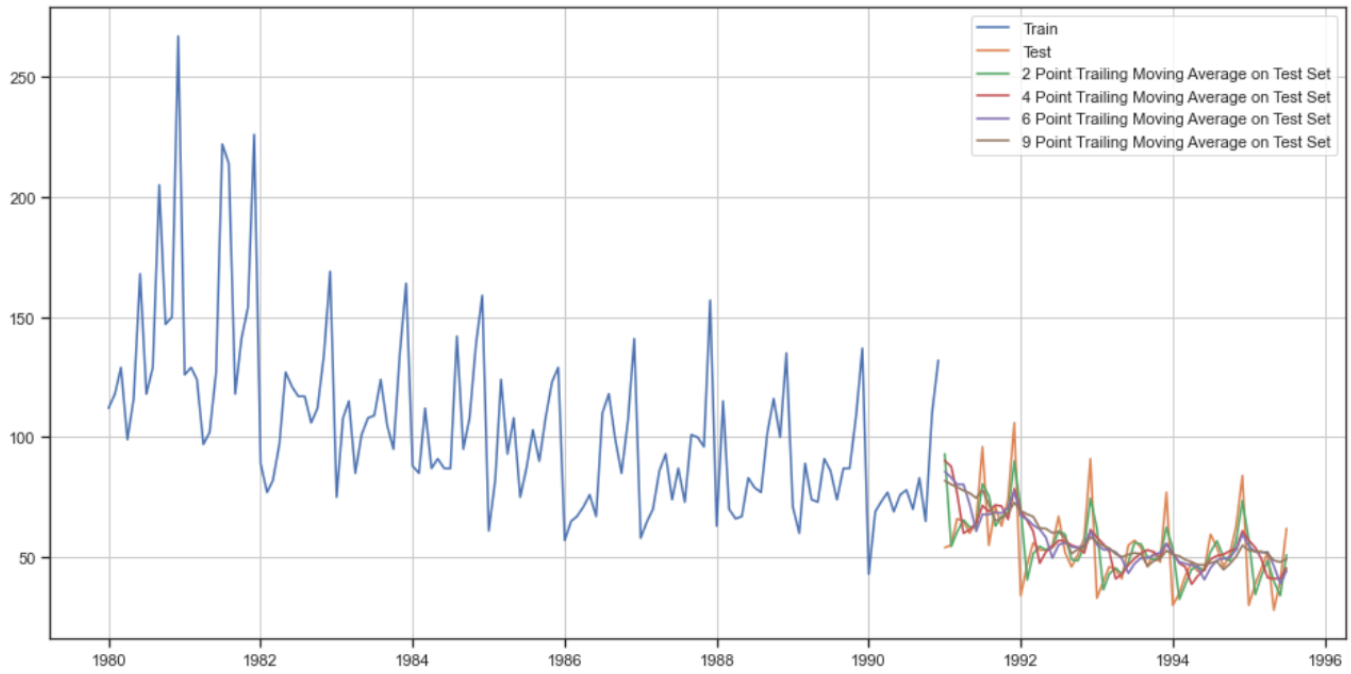

The green line indicates the predictions made by the model, while the orange values are the actual test values. It is clear the predicted values are very far off from the actual values, but once again this was expected as the predicted value is nothing but a simple mean of all the previous values.

Model was evaluated using the RMSE metric. Below is the RMSE calculated for this model.

| | Test RMSE |
|---|---|
| SimpleAverageModel | 53.049755 |

## 4 - 7) Moving Average Models

We created multiple moving average models with rolling windows varying from 2 to 9. Rolling average is a better method than simple average as it takes into account only the previous n values to make the prediction, where n is the rolling window defined. This takes into account the recent trends and is in general more accurate. Higher the rolling window, smoother will be its curve, since more values are being taken into account.



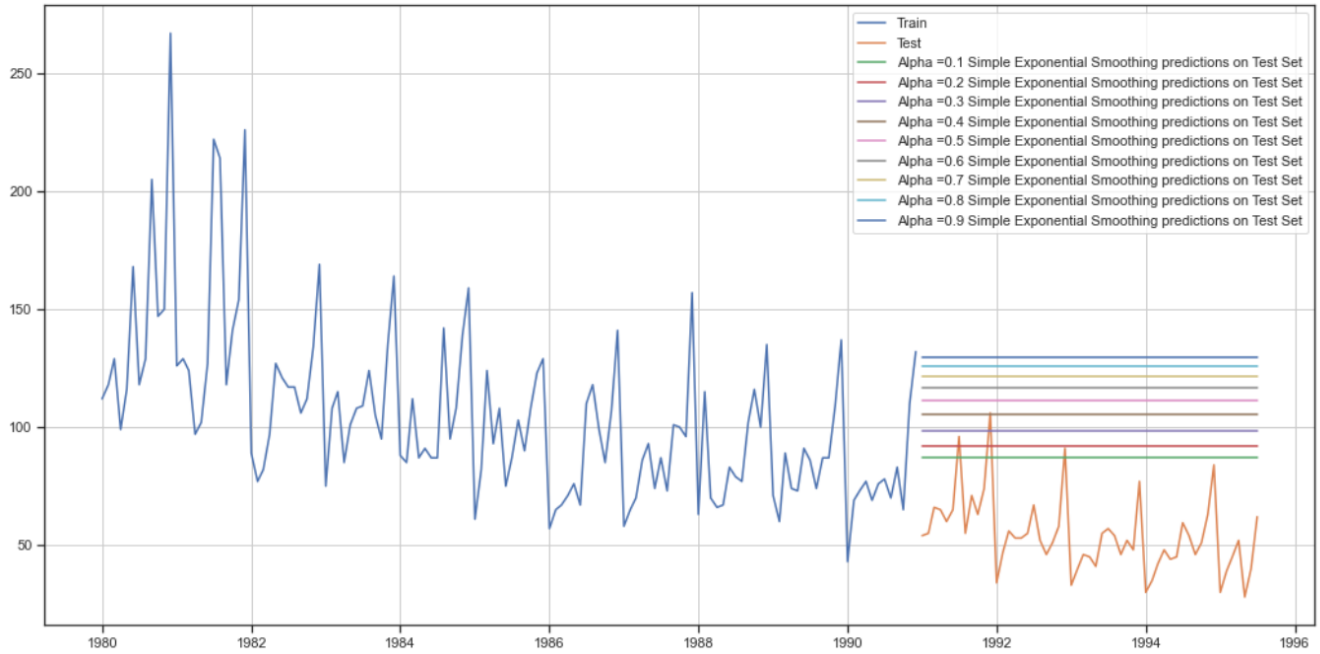

Output of different moving averages are show in the plot above. 2 point moving average lines moves the closest to the actual test values.

Model was evaluated using the RMSE metric. Below is the RMSE calculated for this model.

| | Test RMSE |
|---|---|
| 2pointTrailingMovingAverage | 11.589082 |
| 4pointTrailingMovingAverage | 14.506190 |
| 6pointTrailingMovingAverage | 14.558008 |
| 9pointTrailingMovingAverage | 14.797139 |

## 8-9) Simple Exponential Smoothing

Simple exponential smoothing model was then bought into the picture, this model is good at explaining the level. We first used .fit() function without mentioning explicit value for alpha, letting the algorithm decide an optimum value on its own. After this we used.fit() giving smoothing_level / alpha in the range of 0.1 to 1 with a step size of 0.1. Below was the output plot for various smoothing_level values.




Output for various alpha values is shown by different color lines in the above plot.

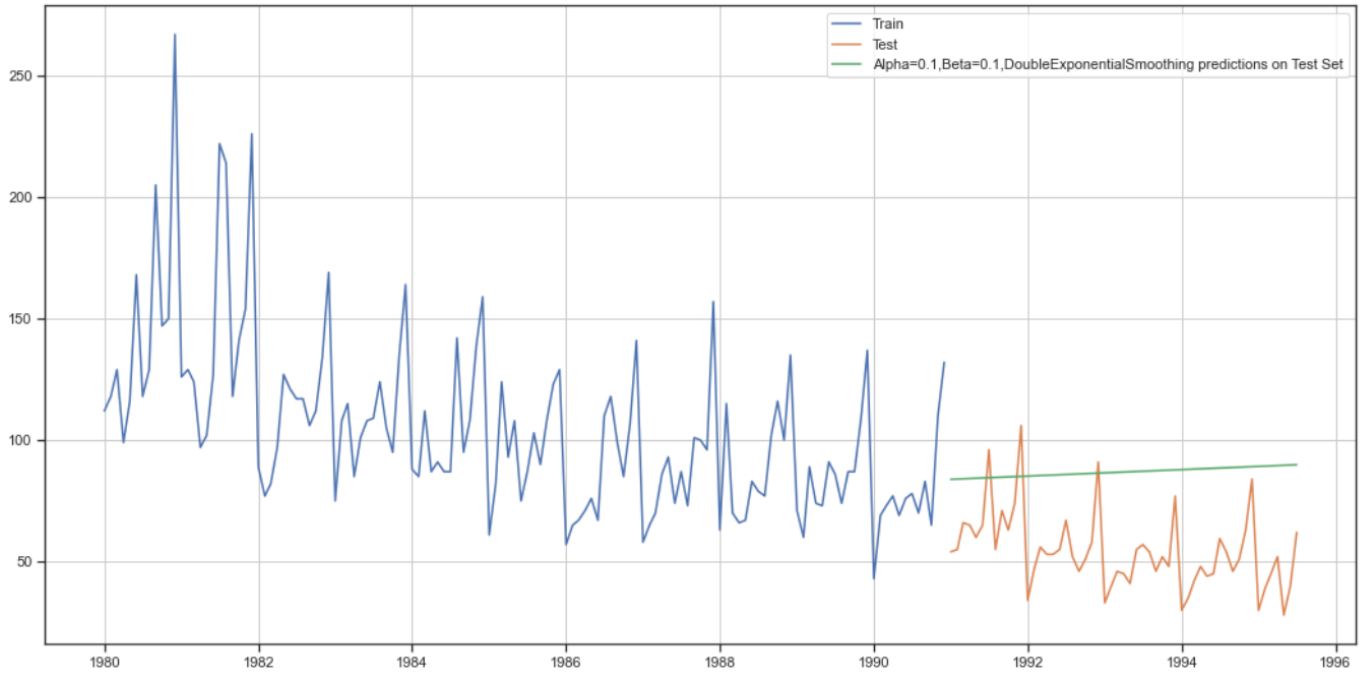
Models were evaluated using the RMSE metric. Below are the RMSE calculated for these models. Also final entry with least RMSE has been shown.

| | Alpha Values | Train RMSE | Test RMSE |
|---|---|---|---|
| **0** | 0.1 | 31.815610 | 36.429535 |
| **1** | 0.2 | 31.979391 | 40.957988 |
| **2** | 0.3 | 32.470164 | 47.096522 |
| **3** | 0.4 | 33.035130 | 53.356493 |
| **4** | 0.5 | 33.682839 | 59.229384 |
| **5** | 0.6 | 34.441171 | 64.558022 |
| **6** | 0.7 | 35.323261 | 69.284383 |
| **7** | 0.8 | 36.334596 | 73.359904 |
| **8** | 0.9 | 37.482782 | 76.725002 |

| | Test RMSE |
|---|---|
| **Alpha=0.09874,SimpleExponentialSmoothing_Auto_Fit** | 36.397792 |
| **Alpha=0.1,SimpleExponentialSmoothing** | 36.429535 |

## 10-11) Double Exponential Smoothing (Holts Model)

After performing Simple Exponential smoothing, we also performed double exponential smoothing, which apart from level also takes into account the trend of the series. Here again we used .fit() first without explicitly mentioning smoothing_level (alpha) and smoothing_slope (beta). After this we used a for loop with range (0.1,1.1) with step size of 0.1. Below is the output plot for the best value observed out of all the attempts, which was alpha = 0.1 and beta = 0.1.




Output for best alpha and beta values is shown by the green color line in the above plot.

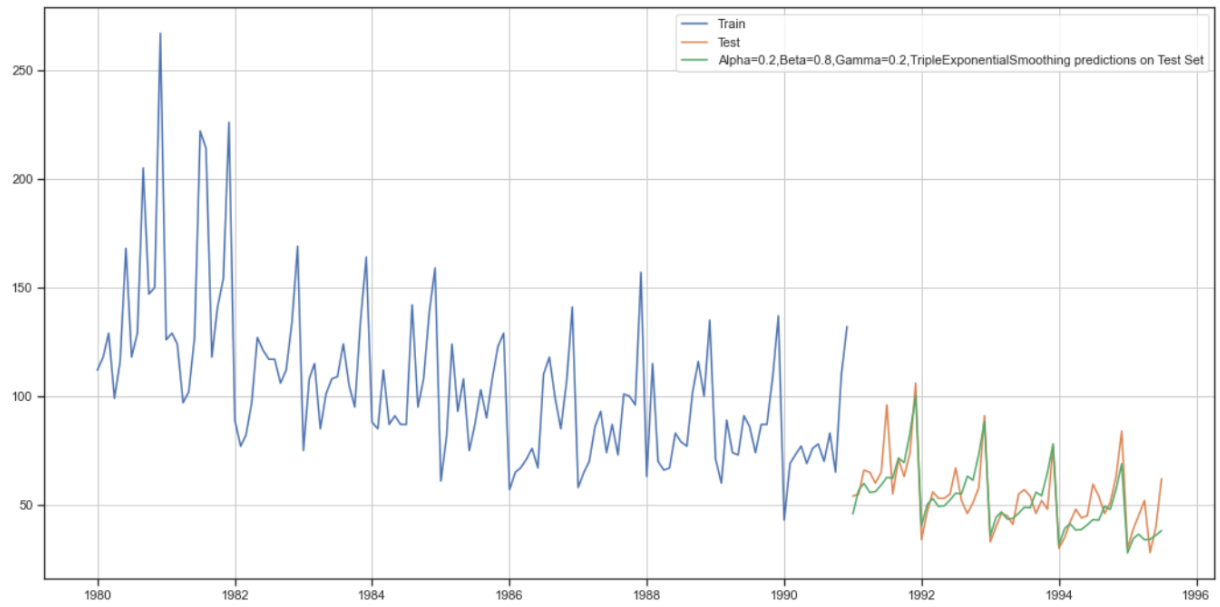
Models were evaluated using the RMSE metric. Below are the RMSE calculated for these models. Also final entry with least RMSE has been shown.

| | Alpha Values | Beta Values | Train RMSE | Test RMSE |
|---|---|---|---|---|
| **0** | 0.1 | 0.1 | 34.439111 | 36.510010 |
| **1** | 0.1 | 0.2 | 33.450729 | 48.221436 |
| **2** | 0.1 | 0.3 | 33.145789 | 77.649847 |
| **3** | 0.1 | 0.4 | 33.262191 | 99.064536 |
| **4** | 0.1 | 0.5 | 33.688415 | 123.742433 |
| **...** | ... | ... | ... | ... |
| **95** | 1.0 | 0.6 | 51.831610 | 801.137173 |
| **96** | 1.0 | 0.7 | 54.497039 | 841.349112 |
| **97** | 1.0 | 0.8 | 57.365879 | 853.421959 |
| **98** | 1.0 | 0.9 | 60.474309 | 834.167545 |
| **99** | 1.0 | 1.0 | 63.873454 | 779.536777 |

| | Test RMSE |
|---|---|
| **Alpha=0.1578,Beta=0.1578,DoubleExponentialSmoothing_Auto_Fit** | 36.397792 |
| **Alpha=0.1,Beta=0.1,DoubleExponentialSmoothing** | 36.510010 |

## 12-13) Triple Exponential Smoothing (Holts-Winter Model)

After performing double exponential smoothing which takes into account only the level and the trends, we moved ahead and tried triple exponential smoothing or the Holts Winter model which not only takes into account level and trends but also takes into account the seasonality present in the series. Again in the same way an auto fit approach and a for loop approach was used here. Here while using the for loop we also took into account both additive and multiplicative trends and seasonality and various permutation and combinations were tried and finally the best model out of all was plotted below.




Output for best alpha, beta and gamma values is shown by the green color line in the above plot. Best model had both multiplicative trend as well as seasonality.

Models were evaluated using the RMSE metric. Below are the RMSE calculated for these models. Also final entry with least RMSE has been shown.

| | Alpha Values | Beta Values | Gamma Values | Train RMSE | Test RMSE | Method |
|---|---|---|---|---|---|---|
| **2145** | 0.2 | 0.8 | 0.2 | 31.099271 | 9.454552 | tm_sm |
| **2136** | 0.2 | 0.7 | 0.2 | 30.330487 | 9.549856 | tm_sm |
| **1011** | 0.1 | 0.2 | 0.2 | 24.365597 | 9.733811 | ta_sm |
| **1012** | 0.1 | 0.2 | 0.3 | 23.969166 | 10.006929 | ta_sm |
| **1010** | 0.1 | 0.2 | 0.1 | 25.529854 | 10.141461 | ta_sm |

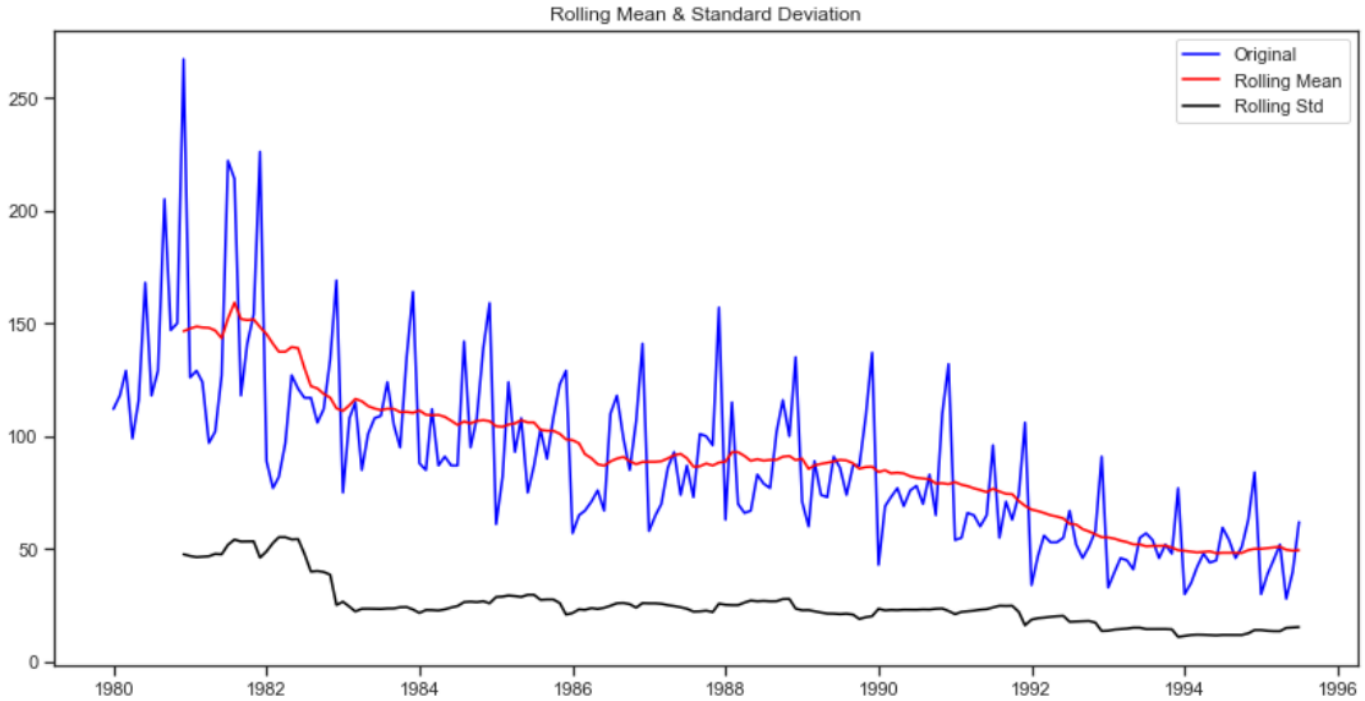
| | Test RMSE |
|---|---|
| **Alpha=0.07003,Beta=3.2222,Gamma=0.0,TrippleExponentialSmoothing_Auto_Fit** | 36.397792 |
| **Alpha=0.2,Beta=0.8,Gamma=0.2,TripleExponentialSmoothing** | 9.454552 |

## 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.

We made use of Dickey-Fuller hypothesis test to determine if the given series is stationary or not. We also plotted the series along with its rolling mean and rolling standard deviations.

Hypothesis for our test:-

Null Hypothesis --> H0 --> Series is not stationary
Alternate Hypothesis --> Ha --> Series is stationary

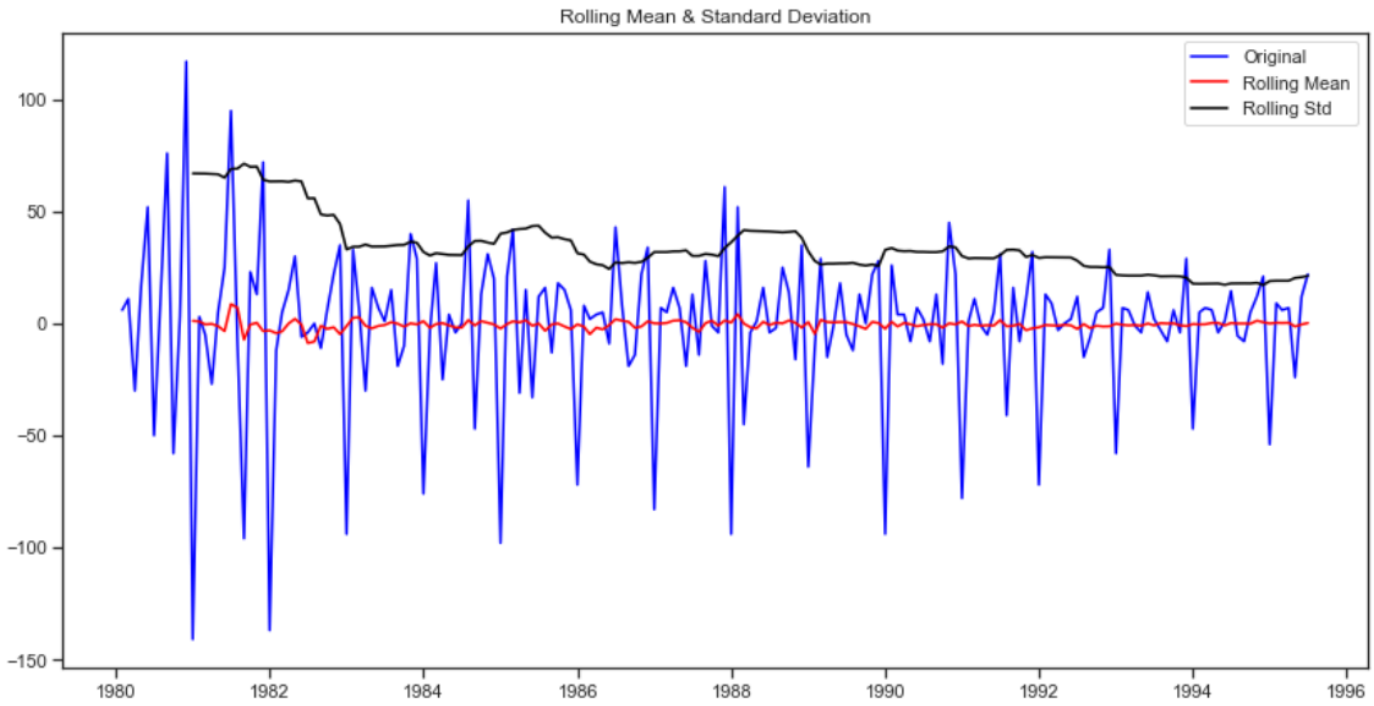



We could see after performing the hypothesis test that the p-values was 0.335674 which was not less than 0.05 (our alpha value) hence we failed to reject the null hypothesis, which implies the Series is not stationary in nature.

Also the rolling means was not a straight line. Now we needed to make the series stationary.

In order to try and make the series stationary we used the differencing approach. We used .diff() function on the existing series without any argument, implying the default diff value of 1 and also dropped the NaN values, since differencing of order 1 would generate the first value as NaN which need to be dropped.

Post doing this, we once again ran the Dickey-Fuller test with the same hypothesis as earlier, to see if differencing made our series stationary.




This time the series became stationary in nature and the p-value obtained from Dickey - Fuller test was just 1.938803e-12, which is way less than 0.05.

Hence the null hypothesis that the series is not stationary at difference = 1 was rejected, which implied that the series has indeed become stationary after we performed the differencing.

Null hypothesis was rejected since the p-value was less than alpha i.e. 0.05. Also the rolling mean plot was a straight line this time around. Also the series looked more or less the same from both the directions, indicating stationarity.

We could now proceed ahead with ARIMA/ SARIMA models, since we had made the series stationary.

## 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.

# Auto - ARIMA Model

Starting with ARIMA model, we made use of ARIMA function from statsmodel library in Python. We employed a for loop for determining the optimum values of p,d,q, where p is the order of the AR (Auto-Regressive) part of the model, while q is the order of the MA (Moving Average) part of the model. d is the differencing that is required to make the series stationary.

p,q values in the range of (0,4) were given to the for loop, while a fixed value of 1 was given for d, since we had already determined d to be 1, while checking for stationarity using ADF test.

Below were the different models that for loop evaluated.

```
Some parameter combinations for the Model...
Model: (0, 1, 1)
Model: (0, 1, 2)
Model: (0, 1, 3)
Model: (1, 1, 0)
Model: (1, 1, 1)
Model: (1, 1, 2)
Model: (1, 1, 3)
Model: (2, 1, 0)
Model: (2, 1, 1)
Model: (2, 1, 2)
Model: (2, 1, 3)
Model: (3, 1, 0)
Model: (3, 1, 1)
Model: (3, 1, 2)
Model: (3, 1, 3)
```

Akaike information criterion (AIC) value was evaluated for each of these models and the model with least AIC value was selected.

| | param | AIC |
|---|---|---|
| **15** | (3, 1, 3) | 1273.194114 |
| **2** | (0, 1, 2) | 1276.835372 |
| **6** | (1, 1, 2) | 1277.359223 |
| **5** | (1, 1, 1) | 1277.775749 |
| **3** | (0, 1, 3) | 1278.074254 |
| **9** | (2, 1, 1) | 1279.045689 |
| **10** | (2, 1, 2) | 1279.298694 |
| **7** | (1, 1, 3) | 1279.312642 |
| **13** | (3, 1, 1) | 1279.605967 |
| **1** | (0, 1, 1) | 1280.726183 |
| **14** | (3, 1, 2) | 1280.969245 |
| **11** | (2, 1, 3) | 1281.196226 |
| **12** | (3, 1, 0) | 1299.478739 |
| **8** | (2, 1, 0) | 1300.609261 |
| **4** | (1, 1, 0) | 1319.348311 |
| **0** | (0, 1, 0) | 1335.152658 |

Model with p=3, d=1, q=3 was found be the optimum model in this scenario with the lowest AIC value of 1273.194114.

Full report was generated for this particular model and RMSE was also calculated for this selection, so that it can then be compared with other models later.

PFB the summary report for the ARIMA model with values (p=3,d=1,q=3).

ARIMA Model Results

| | | | |
|---|---|---|---|
| Dep. Variable: | D.Sales | No. Observations: | 131 |
| Model: | ARIMA(3, 1, 3) | Log Likelihood | -628.597 |
| Method: | css-mle | S.D. of innovations | 28.355 |
| Date: | Sun, 08 Nov 2020 | AIC | 1273.194 |
| Time: | 11:14:36 | 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 | -5.549 | 0.000 | -0.664 | -0.317 |
| ar.L1.D.Sales | -0.7244 | 0.086 | -8.417 | 0.000 | -0.893 | -0.556 |
| ar.L2.D.Sales | -0.7219 | 0.086 | -8.349 | 0.000 | -0.891 | -0.552 |
| ar.L3.D.Sales | 0.2762 | 0.085 | 3.235 | 0.001 | 0.109 | 0.444 |
| ma.L1.D.Sales | -0.0150 | 0.044 | -0.338 | 0.736 | -0.102 | 0.072 |
| ma.L2.D.Sales | 0.0150 | 0.044 | 0.339 | 0.734 | -0.072 | 0.102 |
| ma.L3.D.Sales | -1.0000 | 0.046 | -21.920 | 0.000 | -1.089 | -0.911 |

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 | 3.6157 | -0.0000j | 3.6157 | -0.0000 |
| MA.1 | 1.0000 | -0.0000j | 1.0000 | -0.0000 |
| MA.2 | -0.4925 | -0.8703j | 1.0000 | -0.3320 |
| MA.3 | -0.4925 | +0.8703j | 1.0000 | 0.3320 |

AIC value - 1273.194
BIC value - 1296.196

Forecast was done and compared to the actual test values to determine the RMSE.

RMSE values are as below.

| | Test RMSE |
|---|---|
| p=3,d=1,q=3,Auto_ARIMA | 16.161570 |

# Auto - SARIMA Model

Moving for SARIMA, we now also take into account the seasonality of the series, to enable better predictions. Looking at the lag heat map which we had plotted earlier, we were able to determine the seasonality to be 12. This was further verified by plotting ACF and PACF graphs which showed significant spikes at 12,14,26,48 etc lags indicating a seasonality of 12 i.e. yearly seasonality. ACF and PACF graphs will be shown later when manual ARIMA and SARIMA models are discussed.

A similar for loop with below values was employed, resulting in the models shown below.

```
p = q = range(0, 4)
d= range(0,2)
D = range(0,2)
pdq = list(itertools.product(p, d, q))
model_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, D, q))]
```

```
Examples of some parameter combinations for Model...
Model: (0, 0, 1)(0, 0, 1, 12)
Model: (0, 0, 2)(0, 0, 2, 12)
Model: (0, 0, 3)(0, 0, 3, 12)
Model: (0, 1, 0)(0, 1, 0, 12)
Model: (0, 1, 1)(0, 1, 1, 12)
Model: (0, 1, 2)(0, 1, 2, 12)
Model: (0, 1, 3)(0, 1, 3, 12)
Model: (1, 0, 0)(1, 0, 0, 12)
Model: (1, 0, 1)(1, 0, 1, 12)
Model: (1, 0, 2)(1, 0, 2, 12)
Model: (1, 0, 3)(1, 0, 3, 12)
Model: (1, 1, 0)(1, 1, 0, 12)
Model: (1, 1, 1)(1, 1, 1, 12)
Model: (1, 1, 2)(1, 1, 2, 12)
Model: (1, 1, 3)(1, 1, 3, 12)
Model: (2, 0, 0)(2, 0, 0, 12)
Model: (2, 0, 1)(2, 0, 1, 12)
Model: (2, 0, 2)(2, 0, 2, 12)
Model: (2, 0, 3)(2, 0, 3, 12)
Model: (2, 1, 0)(2, 1, 0, 12)
Model: (2, 1, 1)(2, 1, 1, 12)
Model: (2, 1, 2)(2, 1, 2, 12)
Model: (2, 1, 3)(2, 1, 3, 12)
Model: (3, 0, 0)(3, 0, 0, 12)
Model: (3, 0, 1)(3, 0, 1, 12)
Model: (3, 0, 2)(3, 0, 2, 12)
Model: (3, 0, 3)(3, 0, 3, 12)
Model: (3, 1, 0)(3, 1, 0, 12)
Model: (3, 1, 1)(3, 1, 1, 12)
Model: (3, 1, 2)(3, 1, 2, 12)
Model: (3, 1, 3)(3, 1, 3, 12)
```

Akaike information criterion (AIC) value was evaluated for each of these models and the model with least AIC value was selected. Here only the top 5 models are shown.

| | param | seasonal | AIC |
|---|---|---|---|
| **957** | (3, 1, 1) | (3, 1, 1, 12) | 681.362811 |
| **1021** | (3, 1, 3) | (3, 1, 1, 12) | 681.607517 |
| **1022** | (3, 1, 3) | (3, 1, 2, 12) | 681.983927 |
| **958** | (3, 1, 1) | (3, 1, 2, 12) | 682.320704 |
| **989** | (3, 1, 2) | (3, 1, 1, 12) | 683.211699 |

PFB the summary report for the best SARIMA model with values (3,1,1)(3,1,1,12).

SARIMAX Results

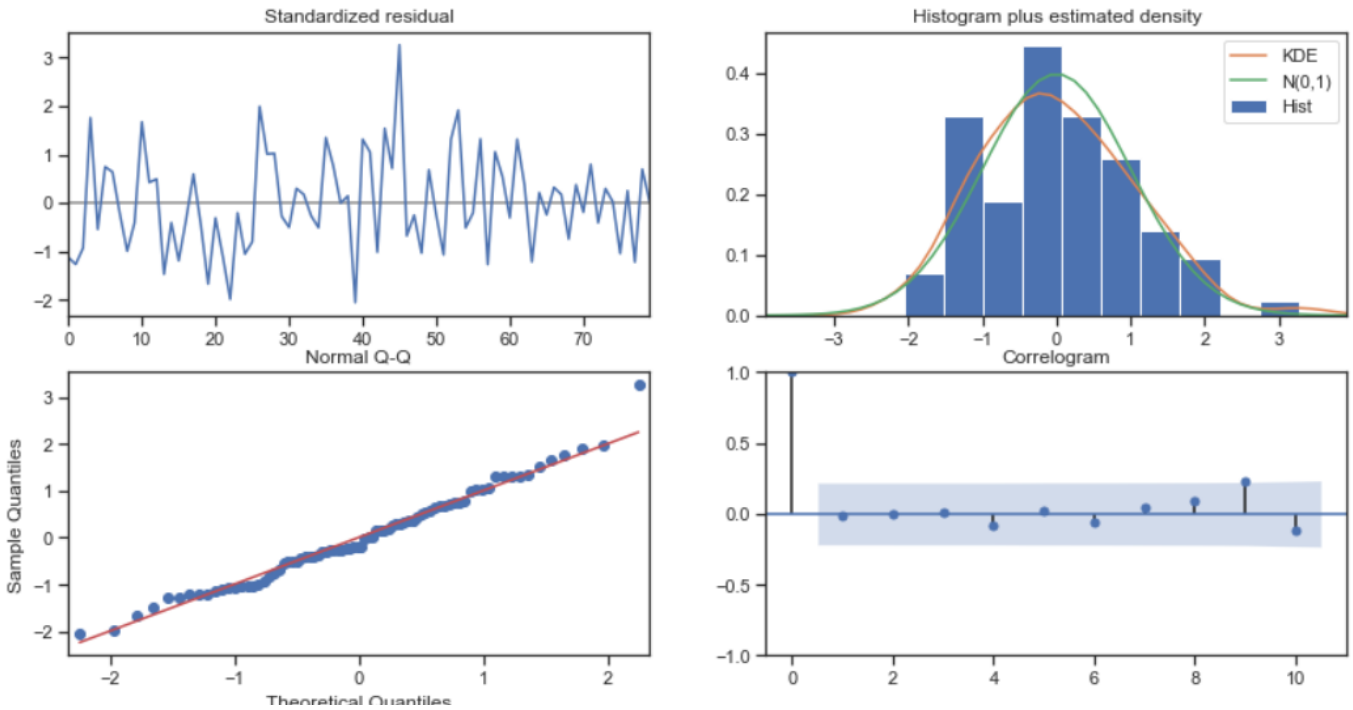
| | | | |
|---|---|---|---|
| Dep. Variable: | y | No. Observations: | 132 |
| Model: | SARIMAX(3, 1, 1)x(3, 1, 1, 12) | Log Likelihood | -331.681 |
| Date: | Sun, 08 Nov 2020 | AIC | 681.363 |
| Time: | 12:22:03 | BIC | 702.801 |
| Sample: | 0 | HQIC | 689.958 |
| | - 132 | | |
| Covariance Type: | opg | | |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| ar.L1 | 0.0173 | 0.151 | 0.114 | 0.909 | -0.279 | 0.314 |
| ar.L2 | -0.0427 | 0.141 | -0.303 | 0.762 | -0.319 | 0.234 |
| ar.L3 | -0.0573 | 0.119 | -0.483 | 0.629 | -0.290 | 0.175 |
| ma.L1 | -0.9388 | 0.085 | -11.103 | 0.000 | -1.105 | -0.773 |
| ar.S.L12 | 0.0907 | 0.126 | 0.720 | 0.471 | -0.156 | 0.337 |
| ar.S.L24 | -0.0438 | 0.108 | -0.407 | 0.684 | -0.255 | 0.167 |
| ar.S.L36 | -3.505e-05 | 0.053 | -0.001 | 0.999 | -0.104 | 0.104 |
| ma.S.L12 | -0.9995 | 94.583 | -0.011 | 0.992 | -186.379 | 184.380 |
| sigma2 | 185.4849 | 1.75e+04 | 0.011 | 0.992 | -3.42e+04 | 3.45e+04 |

| | | | |
|---|---|---|---|
| Ljung-Box (Q): | 42.97 | Jarque-Bera (JB): | 2.56 |
| Prob(Q): | 0.35 | Prob(JB): | 0.28 |
| Heteroskedasticity (H): | 0.56 | Skew: | 0.42 |
| Prob(H) (two-sided): | 0.13 | Kurtosis: | 3.22 |

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

We also plotted the graphs for the residual to determine if any further information can be extracted or all the usable information has already been extracted. Below were the plots for the best auto SARIMA model.




Looking at above plots we can see the residuals are now forming an almost normal distribution ranging withing the empirical range of (-3,3) for a normal distribution.

Also for the Q-Q plot we can see the points are almost overlapping the x=y straight line.

Correlogram also does not indicate that there are any pattern left to be extracted.

Hence looking at the above, we can safely assume that all the actionable information has been extracted by our model and it is a good model in this respect.

Below is the RMSE.

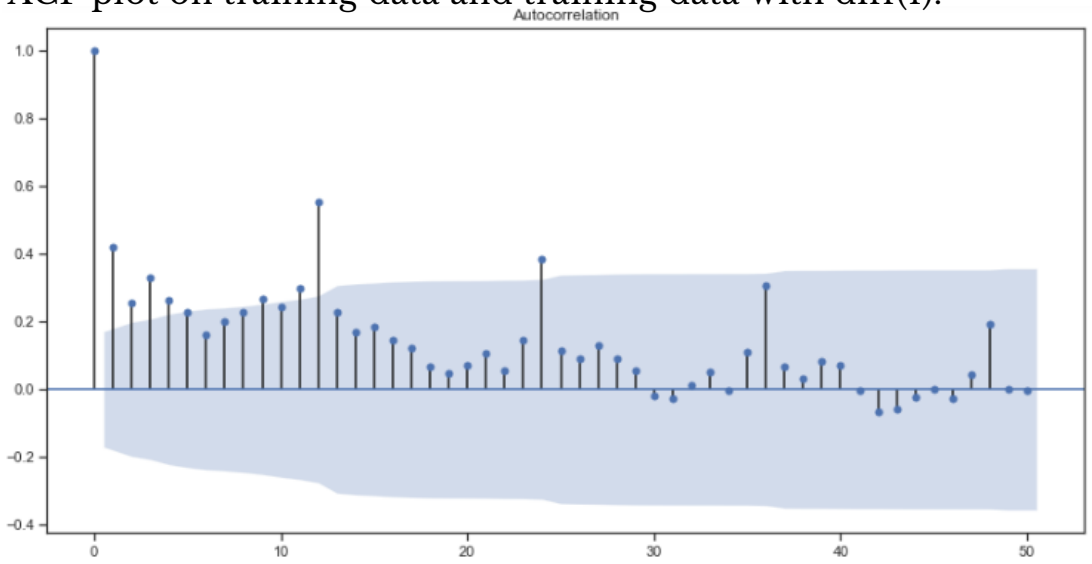
| | Test RMSE |
|---|---|
| (3,1,1),(3,1,1,12),Auto_SARIMA | 16.575833 |

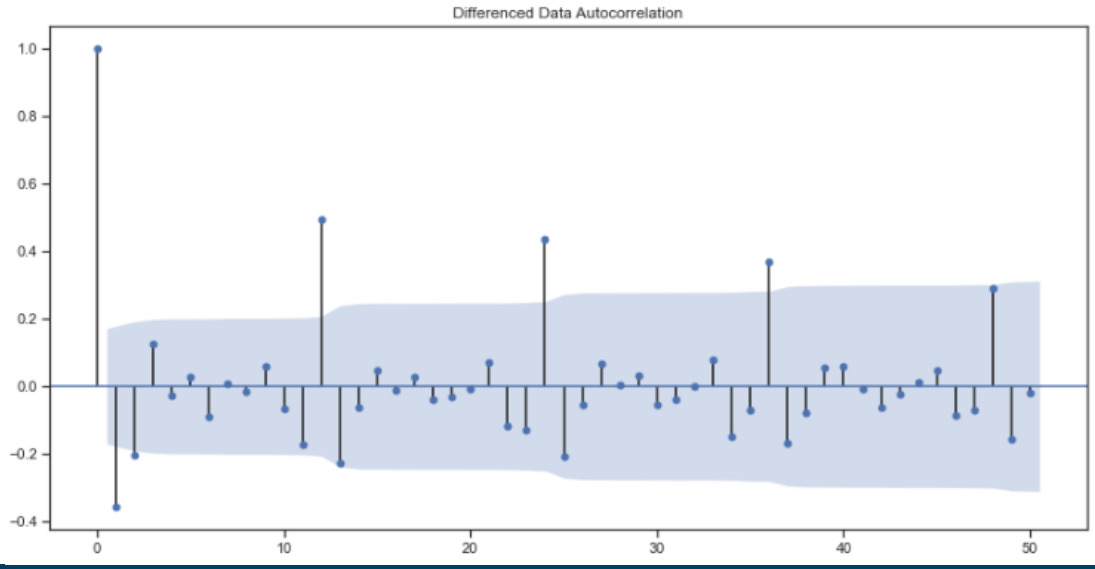
## 7. BUILD ARIMA/SARIMA MODELS BASED ON THE CUT-OFF POINTS OF ACF AND PACF ON THE TRAINING DATA AND EVALUATE THIS MODEL ON THE TEST DATA USING RMSE.

# Manual- ARIMA Model

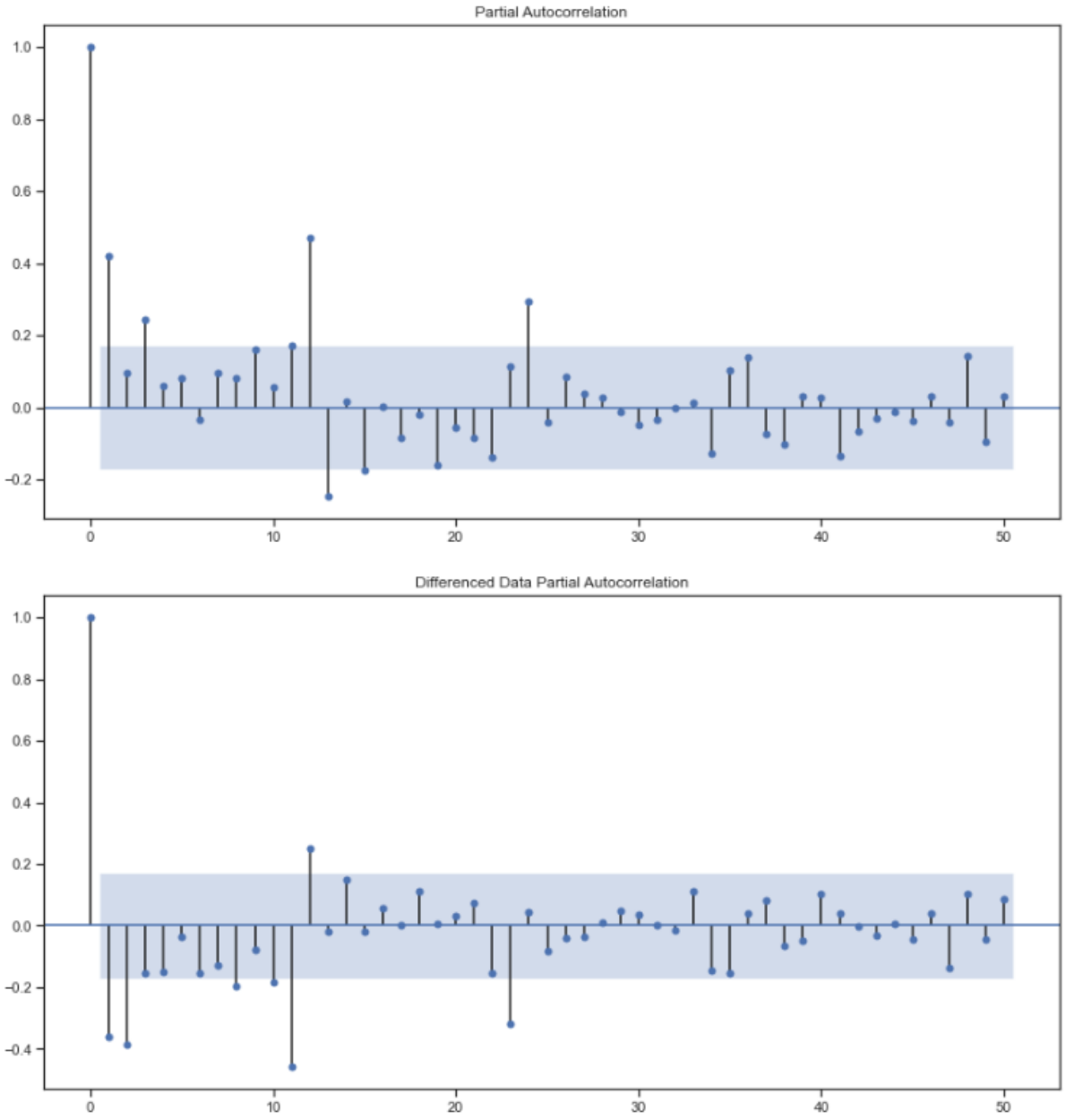
In order to make a manual ARIMA model, we have to first plot the ACF and PACF plots on the training data, based on which we will be able to determine the p,d,q values for the model.

PFB the ACF plot on training data and training data with diff(1).



Now plotting the PACF graph for the training data.




We will use ACF plot to determine order of MA i.e. value of q.
We will use PACF plot to determine order of AR i.e value of p.

Looking at ACF plot we can see a shard decay after lag 2 for original as well as differenced data. hence we select the q value to be 2. i.e. q=2.

Looking at PACF plot we can again see significant bars till lag 2 for differenced series which is stationary in nature, post 2 the decay is large enough. Hence we choose p value to be 2. i.e. p=2.

d values will be 1, since we had seen earlier that the series is stationary with lag1.

Hence the values selected for manual ARIMA:-
p=2,
d=1,
q=2

A ARIMA model was built with above params and AIC value observed for this model was 1279.299. Below is the summary from this manual ARIMA model.

ARIMA Model Results

| | | | |
|---|---|---|---|
| Dep. Variable: | D.y | No. Observations: | 131 |
| Model: | ARIMA(2, 1, 2) | Log Likelihood | -633.649 |
| Method: | css-mle | S.D. of innovations | 29.975 |
| Date: | Sun, 08 Nov 2020 | AIC | 1279.299 |
| Time: | 22:00:28 | BIC | 1296.550 |
| Sample: | 1 | HQIC | 1286.309 |

| | 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.y | -0.4383 | 0.218 | -2.015 | 0.044 | -0.865 | -0.012 |
| ar.L2.D.y | 0.0269 | 0.109 | 0.246 | 0.806 | -0.188 | 0.241 |
| ma.L1.D.y | -0.3316 | 0.203 | -1.633 | 0.102 | -0.729 | 0.066 |
| ma.L2.D.y | -0.6684 | 0.201 | -3.332 | 0.001 | -1.062 | -0.275 |

Roots

| | Real | Imaginary | Modulus | Frequency |
|---|---|---|---|---|
| AR.1 | -2.0289 | +0.0000j | 2.0289 | 0.5000 |
| AR.2 | 18.3387 | +0.0000j | 18.3387 | 0.0000 |
| MA.1 | 1.0000 | +0.0000j | 1.0000 | 0.0000 |
| MA.2 | -1.4960 | +0.0000j | 1.4960 | 0.5000 |

RMSE for manual ARIMA is as below

| | Test RMSE |
|---|---|
| (2,1,2),Manual_ARIMA | 15.369329 |

# Manual SARIMA Model

Looking at the ACF and PACF plots for training data, we can clearly see significant spikes at lags 12,24,36,48 etc, indicating a seasonality of 12. The parameters used for manual SARIMA model are as below.

SARIMAX(2, 1, 2)x(2, 1, 2, 12)

Below is the summary of the manual SARIMA model, having the AIC value of 776.996.

SARIMAX Results

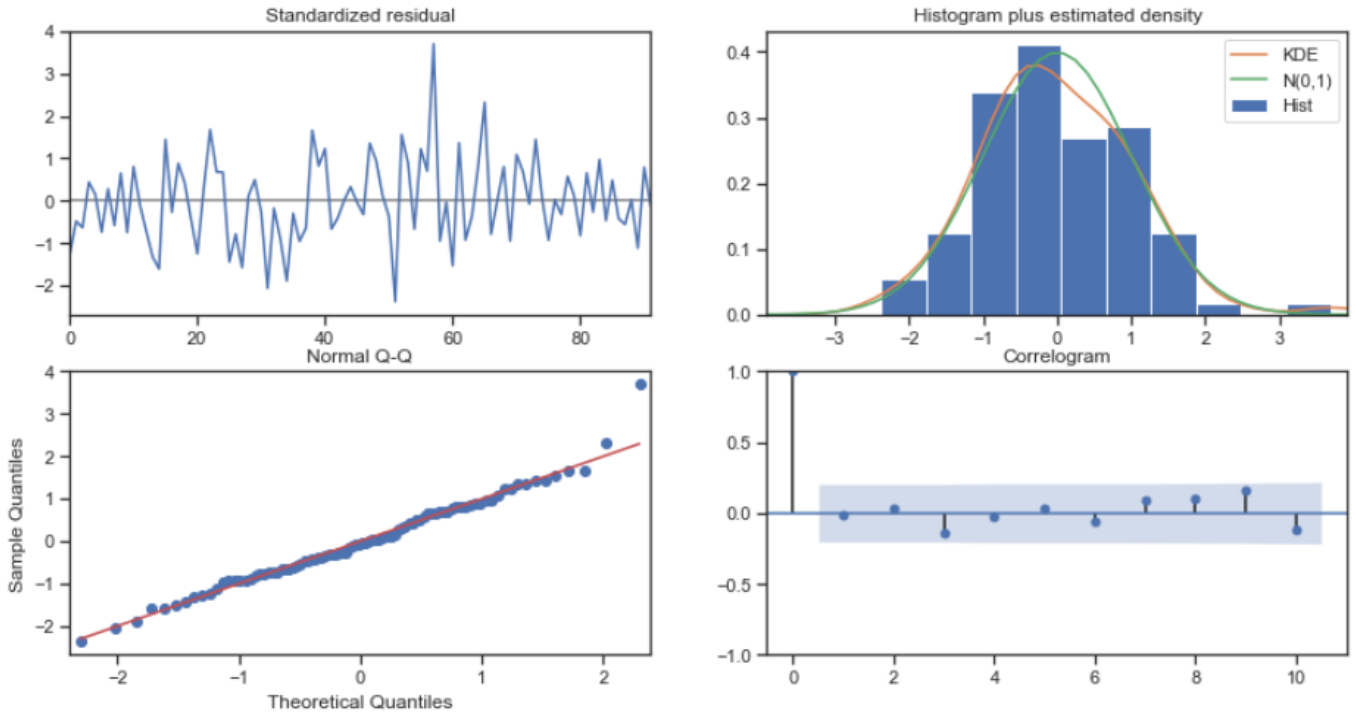
| | | | |
|---|---|---|---|
| Dep. Variable: | y | No. Observations: | 132 |
| Model: | SARIMAX(2, 1, 2)x(2, 1, 2, 12) | Log Likelihood | -379.498 |
| Date: | Sun, 08 Nov 2020 | AIC | 776.996 |
| Time: | 22:05:19 | BIC | 799.692 |
| Sample: | 0 | HQIC | 786.156 |
| | - 132 | | |
| Covariance Type: | opg | | |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| ar.L1 | -0.8551 | 0.146 | -5.838 | 0.000 | -1.142 | -0.568 |
| ar.L2 | -0.0022 | 0.125 | -0.017 | 0.986 | -0.247 | 0.242 |
| ma.L1 | -0.0586 | 0.157 | -0.373 | 0.709 | -0.366 | 0.249 |
| ma.L2 | -1.0125 | 0.191 | -5.305 | 0.000 | -1.387 | -0.638 |
| ar.S.L12 | 0.0347 | 0.185 | 0.188 | 0.851 | -0.328 | 0.397 |
| ar.S.L24 | -0.0459 | 0.029 | -1.598 | 0.110 | -0.102 | 0.010 |
| ma.S.L12 | -0.7223 | 0.333 | -2.172 | 0.030 | -1.374 | -0.071 |
| ma.S.L24 | -0.0772 | 0.212 | -0.364 | 0.716 | -0.493 | 0.339 |
| sigma2 | 179.0764 | 46.023 | 3.891 | 0.000 | 88.873 | 269.280 |

| | | | |
|---|---|---|---|
| Ljung-Box (Q): | 29.82 | Jarque-Bera (JB): | 7.06 |
| Prob(Q): | 0.88 | Prob(JB): | 0.03 |
| Heteroskedasticity (H): | 0.87 | Skew: | 0.45 |
| Prob(H) (two-sided): | 0.71 | Kurtosis: | 4.01 |

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

We also plotted the graphs for the residual to determine if any further information can be extracted or all the usable information has already been extracted. Below were the plots for the manual SARIMA model.




Looking at above plots we can see the residuals are now forming an almost normal distribution ranging withing the empirical range of (-3,3) for a normal distribution.

Also for the Q-Q plot we can see the points are almost overlapping the x=y straight line.

Correlogram also does not indicate that there are any pattern left to be extracted.

Hence looking at the above, we can safely assume that all the actionable information has been extracted by our model and it is a good model in this respect.

Below is the RMSE.

| | Test RMSE |
|---|---|
| (2,1,2)(2,1,2,12),Manual_SARIMA | 16.295095 |

## 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.

PFB the dataframe containing multiple models that we have created along with their RMSE values. The values have been sorted in ascending order, starting with the lowest RMSE, hence the best model is on the top while the worst performing model is at the bottom.

| | Test RMSE |
|---|---|
| Alpha=0.2,Beta=0.8,Gamma=0.2,TripleExponentialSmoothing | 9.454552 |
| 2pointTrailingMovingAverage | 11.589082 |
| 4pointTrailingMovingAverage | 14.506190 |
| 6pointTrailingMovingAverage | 14.558008 |
| 9pointTrailingMovingAverage | 14.797139 |
| RegressionOnTime | 15.278158 |
| (2,1,2),Manual_ARIMA | 15.369329 |
| p=3,d=1,q=3,Auto_ARIMA | 16.161570 |
| (2,1,2)(2,1,2,12),Manual_SARIMA | 16.295095 |
| (3,1,1),(3,1,1,12),Auto_SARIMA | 16.575833 |
| Alpha=0.1578,Beta=0.1578,DoubleExponentialSmoothing_Auto_Fit | 36.397792 |
| Alpha=0.07003,Beta=3.2222,Gamma=0.0,TrippleExponentialSmoothing_Auto_Fit | 36.397792 |
| Alpha=0.09874,SimpleExponentialSmoothing_Auto_Fit | 36.397792 |
| Alpha=0.1,SimpleExponentialSmoothing | 36.429535 |
| Alpha=0.1,Beta=0.1,DoubleExponentialSmoothing | 36.510010 |
| SimpleAverageModel | 53.049755 |
| NaiveModel | 79.304391 |

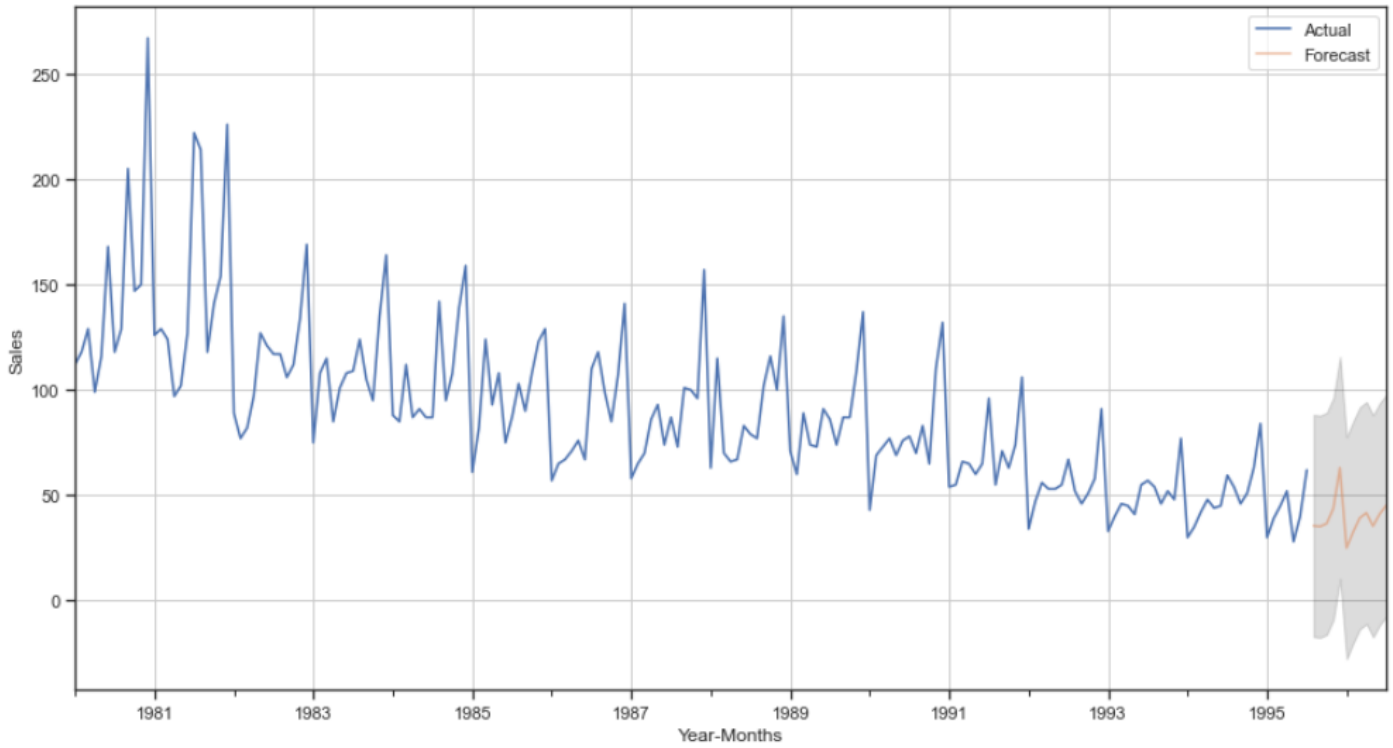
## 9 BASED ON THE MODEL-BUILDING EXERCISE, BUILD THE MOST OPTIMUM MODEL(S) ON THE COMPLETE DATA AND PREDICT 12 MONTHS INTO THE FUTURE WITH APPROPRIATE CONFIDENCE INTERVALS/BANDS.

Based on the above comparison of all the various models that we had built, we can conclude that the triple exponential smoothing or the Holts-Winter model is giving us the lowest RMSE, hence it would be the most optimum model and would give the most accurate prediction at alpha = 0.2 , beta = 0.8 and gamma = 0.2.

Hence we went ahead and built the most optimal model and made prediction 1 year into the future i.e. from 01-08-1995 to 01-07-1996. Below were the sales predictions made by this best optimum model.

| | Sales_Predictions |
|---|---|
| 1995-08-01 | 35.553959 |
| 1995-09-01 | 35.179772 |
| 1995-10-01 | 36.588317 |
| 1995-11-01 | 43.635943 |
| 1995-12-01 | 63.073500 |
| 1996-01-01 | 25.031723 |
| 1996-02-01 | 32.583791 |
| 1996-03-01 | 39.229308 |
| 1996-04-01 | 41.675073 |
| 1996-05-01 | 35.397941 |
| 1996-06-01 | 41.110059 |
| 1996-07-01 | 45.203036 |

We also plotted the sales prediction on the graph along with the confidence intervals. PFB the graph.




Predictions, 1 year into the future are shown in orange color, while the confidence interval has been shown in grey color.

## 10. COMMENT ON THE MODEL THUS BUILT AND REPORT YOUR FINDINGS AND SUGGEST THE MEASURES THAT THE COMPANY SHOULD BE TAKING FOR FUTURE SALES.

Looking at the most optimal model and its prediction, we can see a clear downward trend for the Rose wine for the company and trend seems to be carried forward in the future also.

This variety of wine has been constantly declining in popularity since the last more than a decade.

Also the wine sales are highly impacted by the seasonal changes, with wine sales picking up in the festival season, and dropping during the peak winter time i.e. January.

It would be recommended for the company to run some campaigns which would boost the consumption of the wine during the rest of the year.

Campaigns during lean periods are not recommended since it seems people are not purchasing wine due to climatic reasons, running campaigns won't change people's opinion at this time of the year.

Campaigns during peak might also not be that useful, since the sales are already high during this time of the year and adding campaigns might not generate that much bang for the buck.

Running campaigns during the April to June period might yield maximum results for the company, as the sales are subdued during this period, which if boosted will increase the overall performance of the wine in the market across the year.

# SPARKLING WINE SALES

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.

In this section we will focus on the Sparkling Wine data set.

The report has been divided into two parts to keep things easy to understand, as the same set of questions need to be answered twice, each time with a different data set.

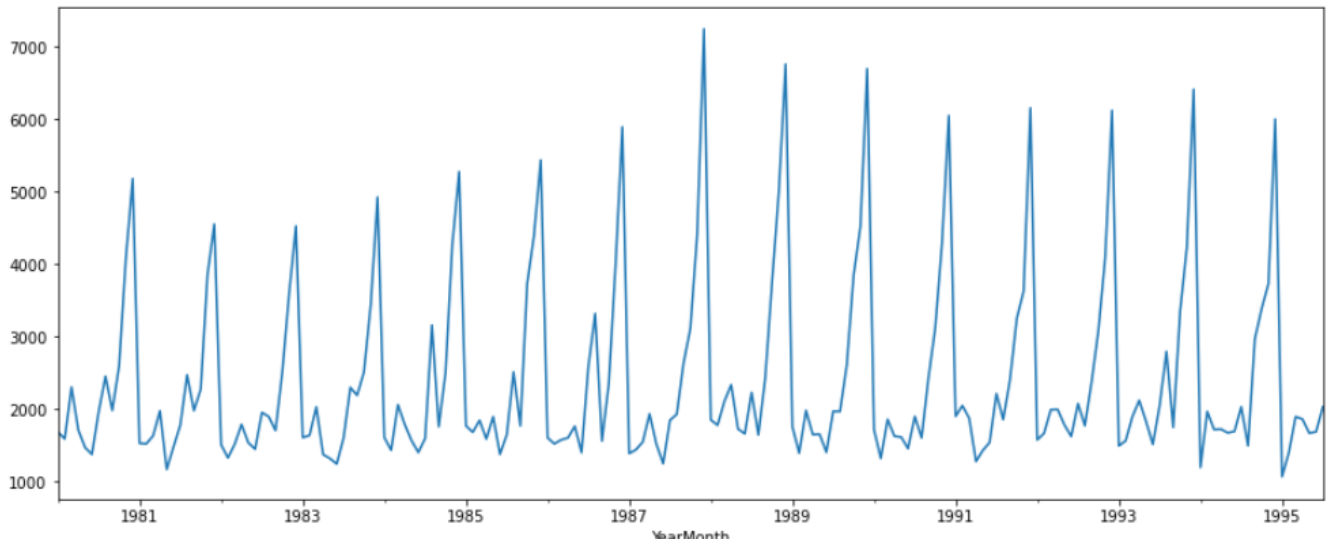
This section is for Sparkling Wine.

# 1. READ THE DATA AS AN APPROPRIATE TIME SERIES DATA AND PLOT THE DATA.

The data set was read using pandas library's read_csv function. Parse date feature was used to read this time series data. index_col was selected as the 'YearMonth' column from the given csv file. Data was hence ingested.

Post ingesting the data, we renamed the column 'Sparkling' to 'Sales' to better indicate its intended purpose. Along with it, we also extracted the Month and Years as columns from the given index values. Below is the head of the ingested data after above operations were performed. Also we plotted the data using matplotlib python library, this too has been shown below.

| YearMonth | Sales | Year | Month |
|---|---|---|---|
| 1980-01-01 | 1686 | 1980 | 1 |
| 1980-02-01 | 1591 | 1980 | 2 |
| 1980-03-01 | 2304 | 1980 | 3 |
| 1980-04-01 | 1712 | 1980 | 4 |
| 1980-05-01 | 1471 | 1980 | 5 |



# 2. PERFORM APPROPRIATE EXPLORATORY DATA ANALYSIS TO UNDERSTAND THE DATA AND ALSO PERFORM DECOMPOSITION.

We performed various steps to extract information from the given data set.

Below are the EDA techniques that we made use of on this sparkling wine data set.

**Techniques used:-**

1) ECF Plot

2) Box plot - Yearly

3) Box Plot - Monthly

4) Monthly sales across years

5) 3 - Year sales graph

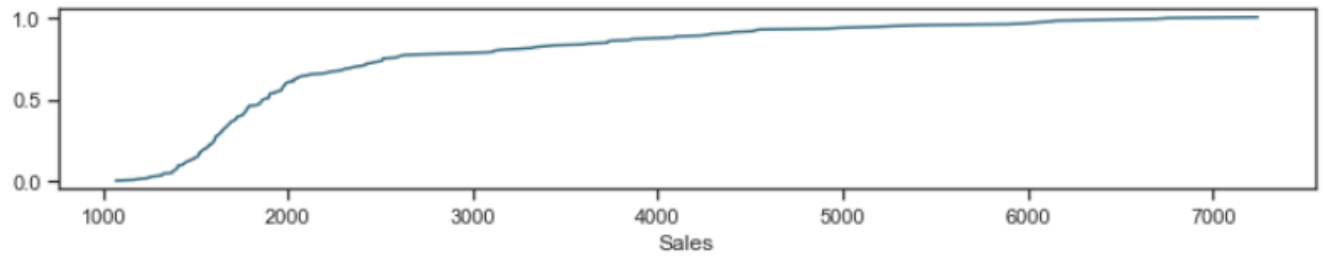
6) Quaterly sales graph

7) Correlation heat map

8) Lag correlation heal map.

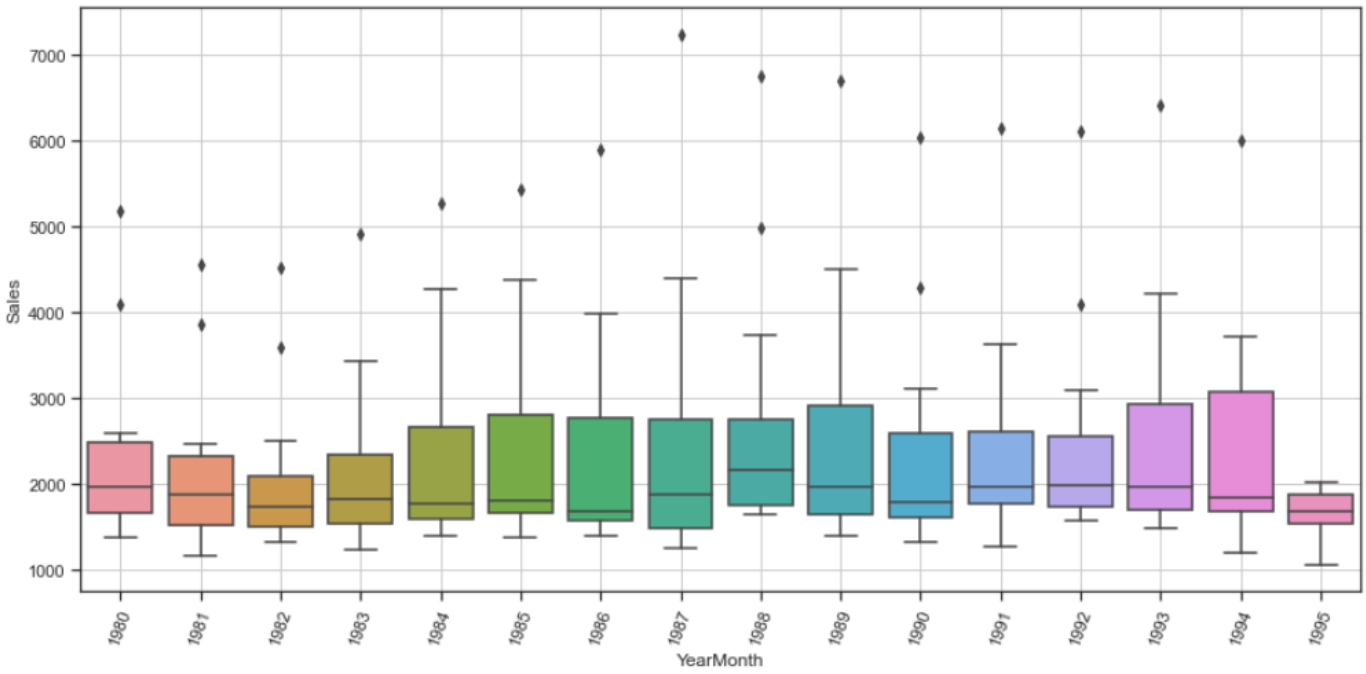
## ECF Plot

An ECF plot was made for the given series. This plot shows that for the most months i.e. more than 50% of the months the sale has been less than 2000. Peak value being around 7000, however almost 80% of the times the value has been less 3000. This graph shows how the data is distributed.



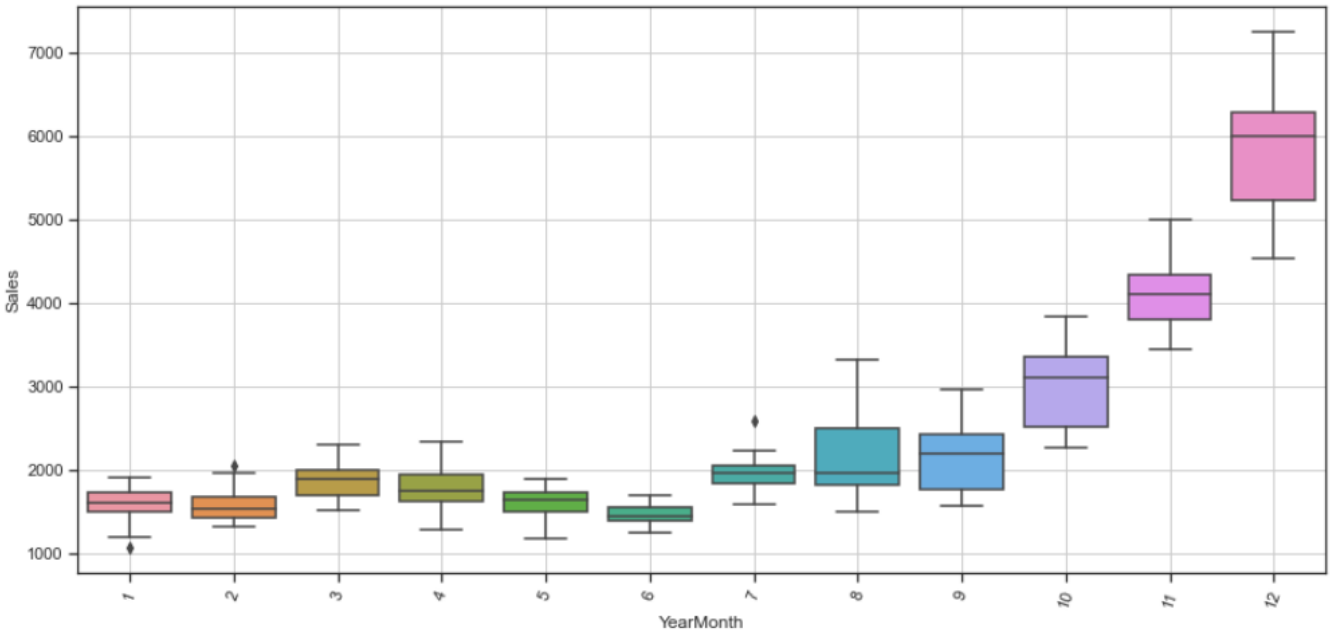

## Box Plot - Yearly

An yearly box plot, clearly shows that the sales trends for Sparkling wine are more or less constant over the last few years. The sales peaked around 1988 - 1989 period after which it has remained steady. Most of the outlier being show in the graph are for the month of December when peak sales are observed. However these are genuine values and do not need to be treated in any way.



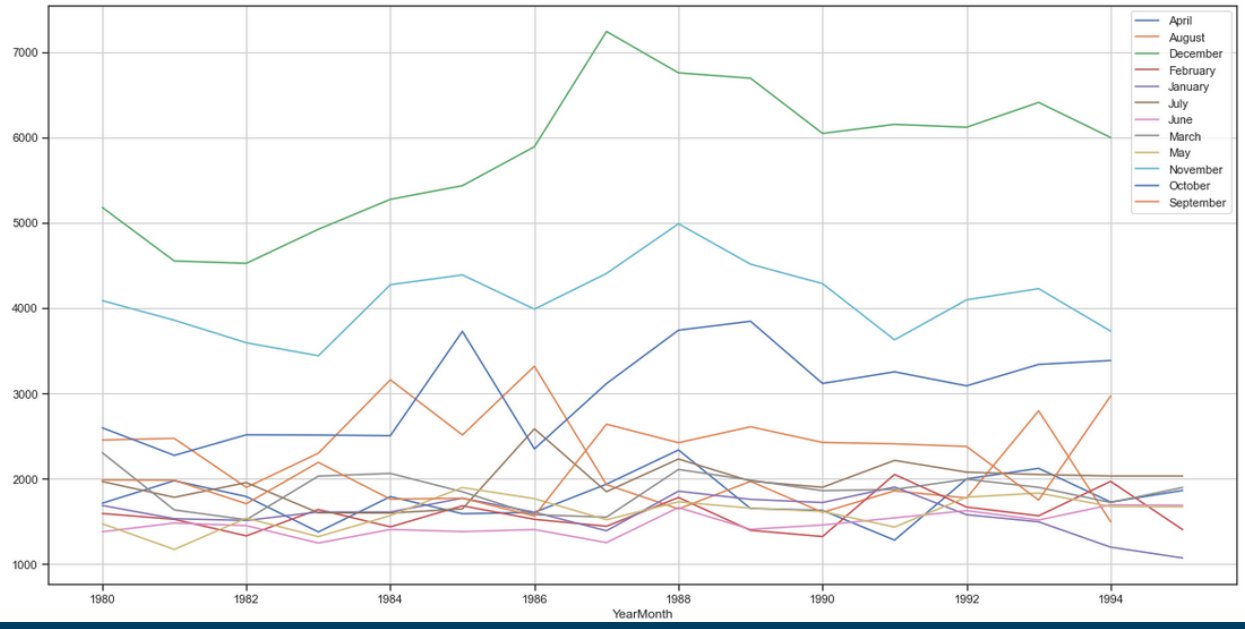
## Box Plot - Monthly

We also plotted a box plot, depicting the sales pattern for each month over the years. It is very clear that peak sales are observed in the month of December, while Sales plummet in the month of January to its lowest level. Sale remains tipid during the initial part of the year and picks up in the latter part starting from the month of August.



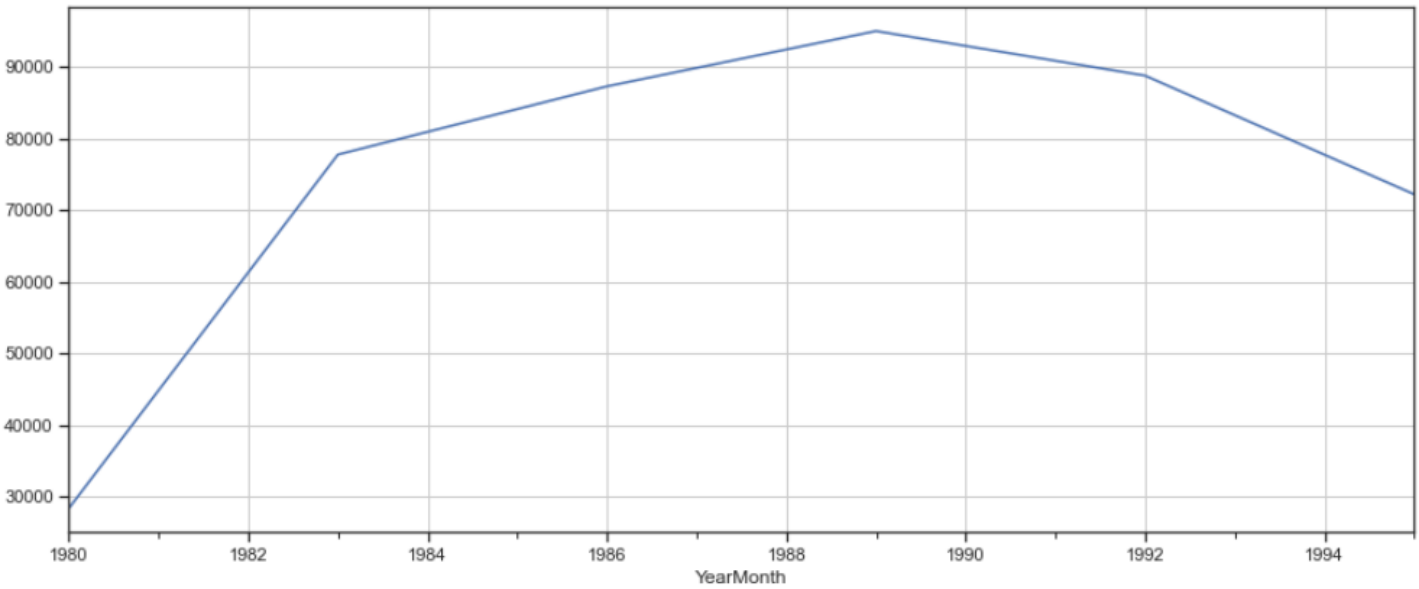

## Monthly Sales Across Years

A chart was plotted to further corroborate the above findings. This gives a clearer view of sales across different months of the year over the 15 years period i.e. 1980 to 1995. Clearly sales peak in the month of December.



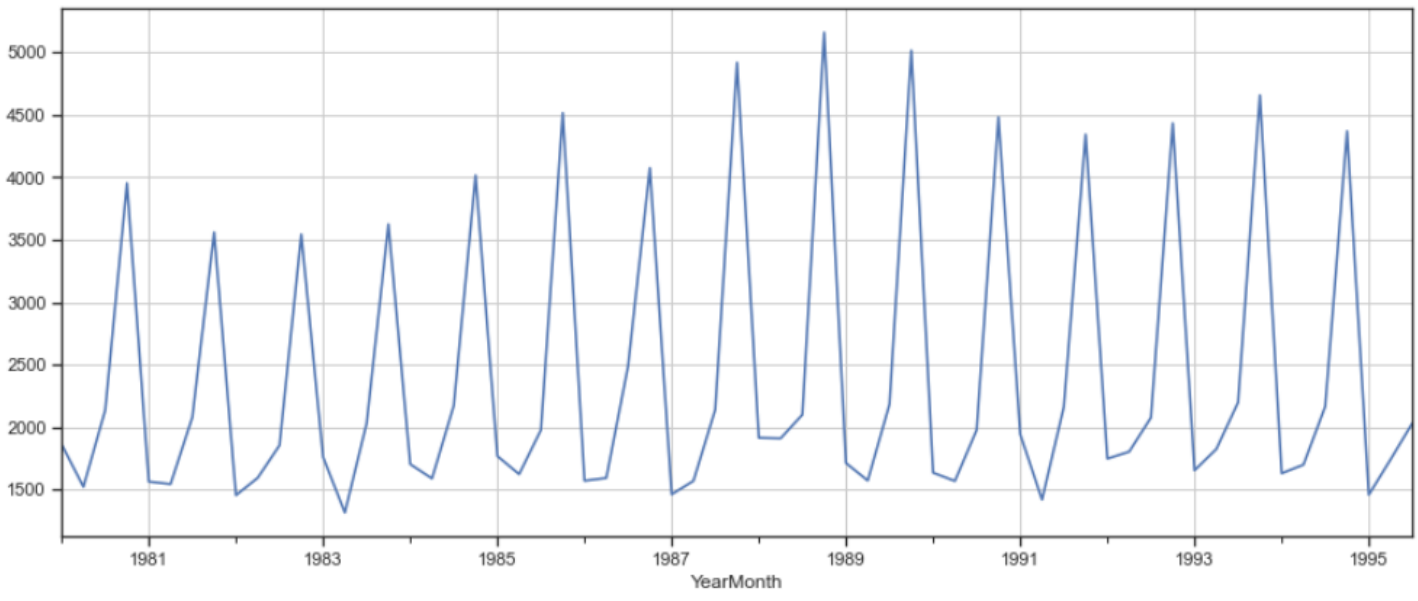
## 3 - Year Sales Graph

Using the up-sampling techniques of pandas library we up-sampled the yearly data to 3 year data and plotted the below graph to understand if there is any broader trend that we might be missing. Looking at below graph, we can infer that the sales peaked around 1988 - 1989, however post that there has been a slight decline.



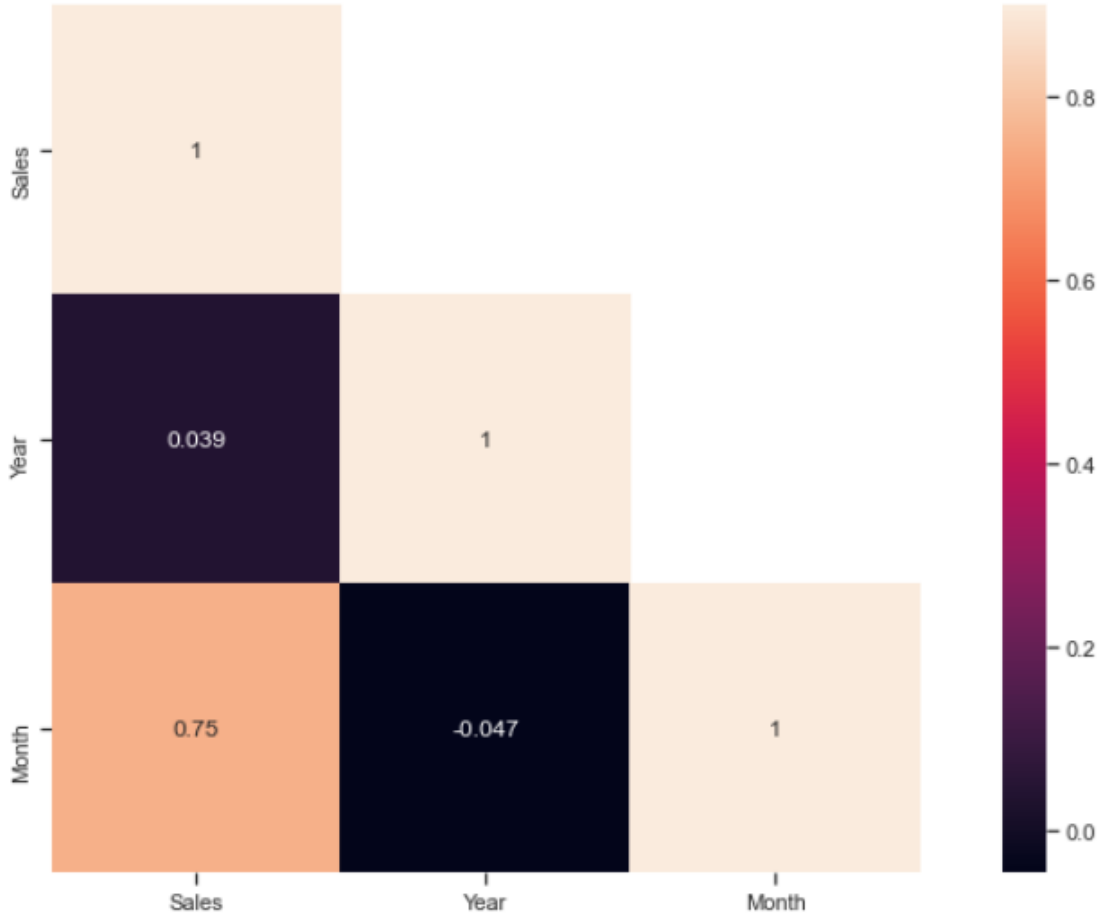

## Quarterly Sales Graph

Using down sampling techniques of pandas we down sampled the data to quarterly range and found that the peak quarter of sales is the last quarter of the year, which is again primarily due to the peak sales in December.



## Correlation Heat map

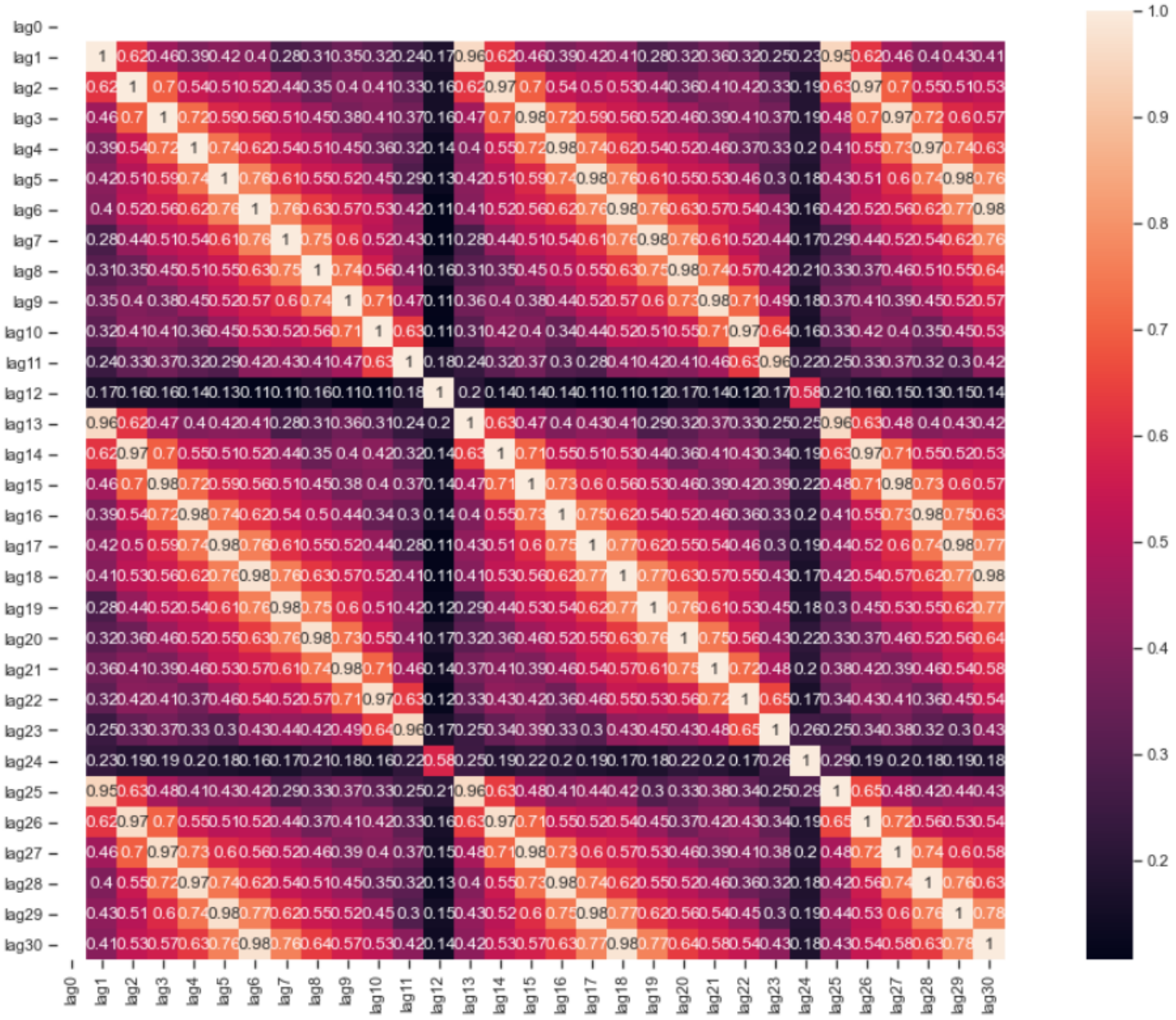
We plotted a heat map of sales with month and year columns. We could see while there was little correlation between Sales and the Years data, there significantly more correlation between the month and Sales columns. Clearly indicating a seasonal pattern in our Sales data. Certain months have higher sales, while certain months have lesser.




Correlation between months and Sales also makes sense. During December due to festive season, it is expected that wine sales in general would increase. Again during the month of January due to extreme winters, sales might be on the lower side. Hence the sales of wine are somewhat correlated to months.

## Correlation Heat map for Lags

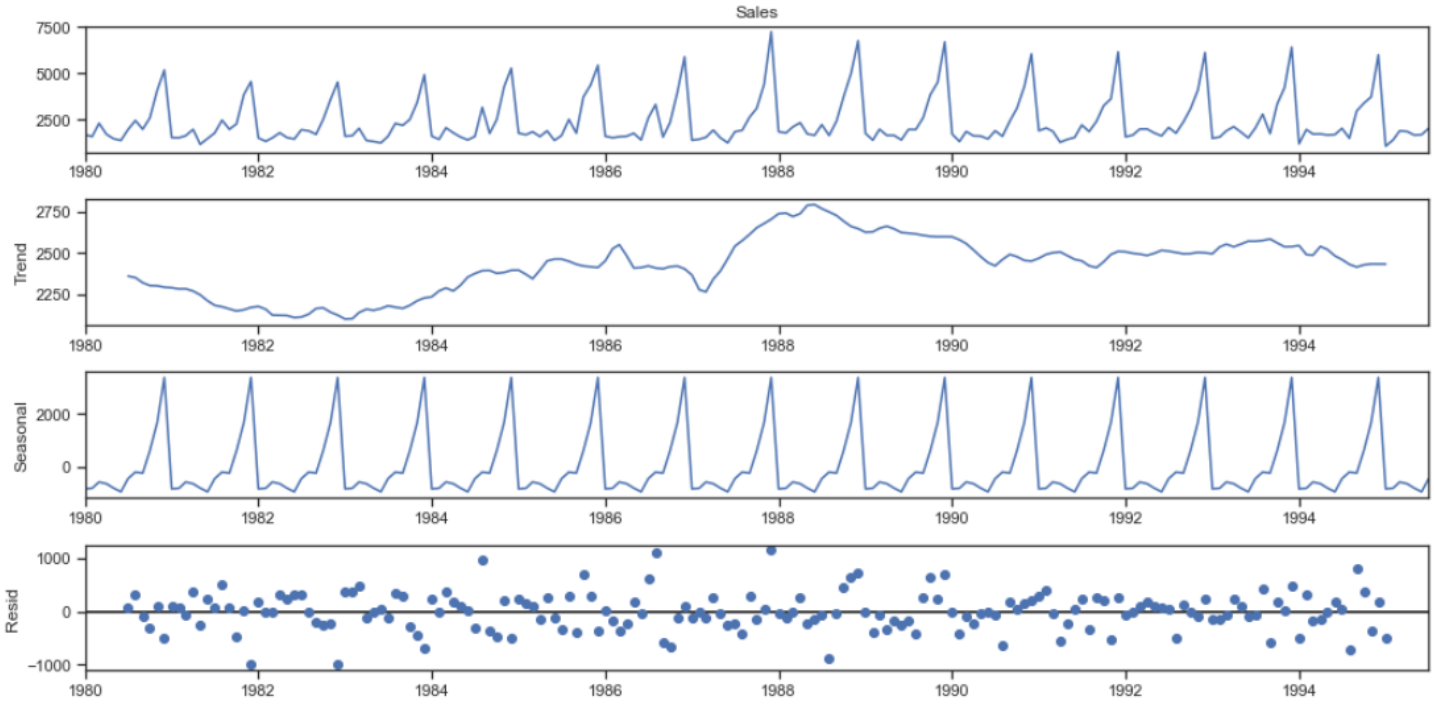
We calculated multiple lag values (30 values) and a heat map of the lags with each other were plotted. From the plot we can see a clear correlation between lag1 and lag13, a difference of 12. Again correlation between lag2 and lag14 is high, a difference of 12. This indicates a yearly seasonality or a seasonality of 12.




The black lines being observed in the heat map, indicate correlation between january and December months which are in stark comparison to each other. Sales peak in December while they plummet in January hence a very low correlation is present which is represented by a darker color.

## Decomposition - Additive

We used seasonal_decompose method from statsmodels library in python to decompose the given series into Trend, Seasonality and Residue. We first decomposed the time series using additive approach. Below are the decomposition plots.



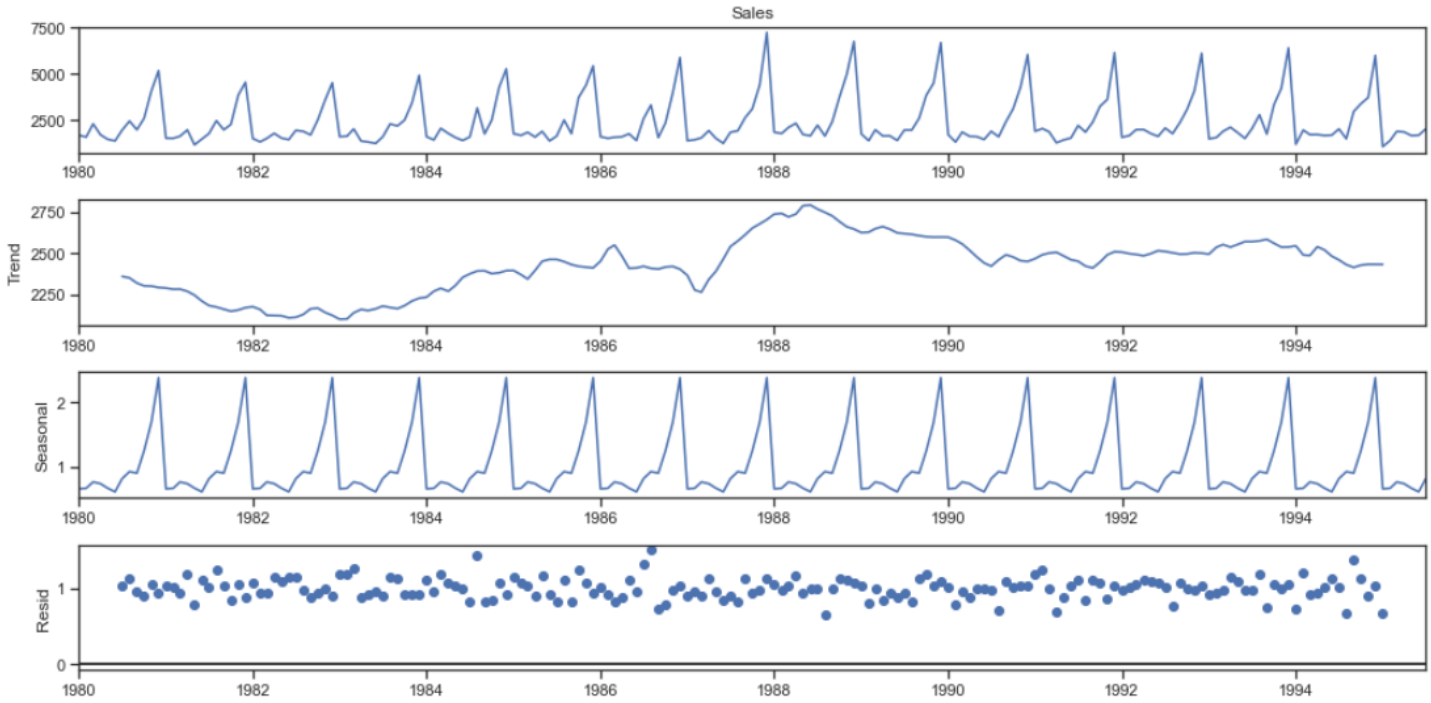

Looking at the above decomposition, we can clearly see a trend which peaked in 1988 - 1989 and then has remained more or less constant after declining shortly. Also a clear seasonality is present. It seems the peak is reached towards the end of the year, which corroborates with our earlier findings.

The residue is quite spread out and is not forming a straight line. Hence we will further look for decomposition of the series using multiplicative approach.

There are clear trend and seasonality components to this time series data.

## Decomposition - Multipicative

We used seasonal_decompose method from statsmodels library in python to decompose the given series into Trend, Seasonality and Residue. We used multiplicative approach this time. Below are the plots.




Looking at the above decomposition we can once again see a clear trend component which indicates sales reaching peak around 1988 - 1989 and then remaining the same more or less. Also a very clear seasonality trend is noticed even with multiplicative approach.

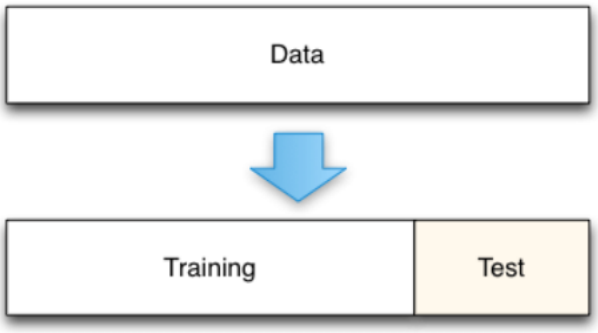
The residue in this case is also not forming a perfect straight line but it is better than additive model.

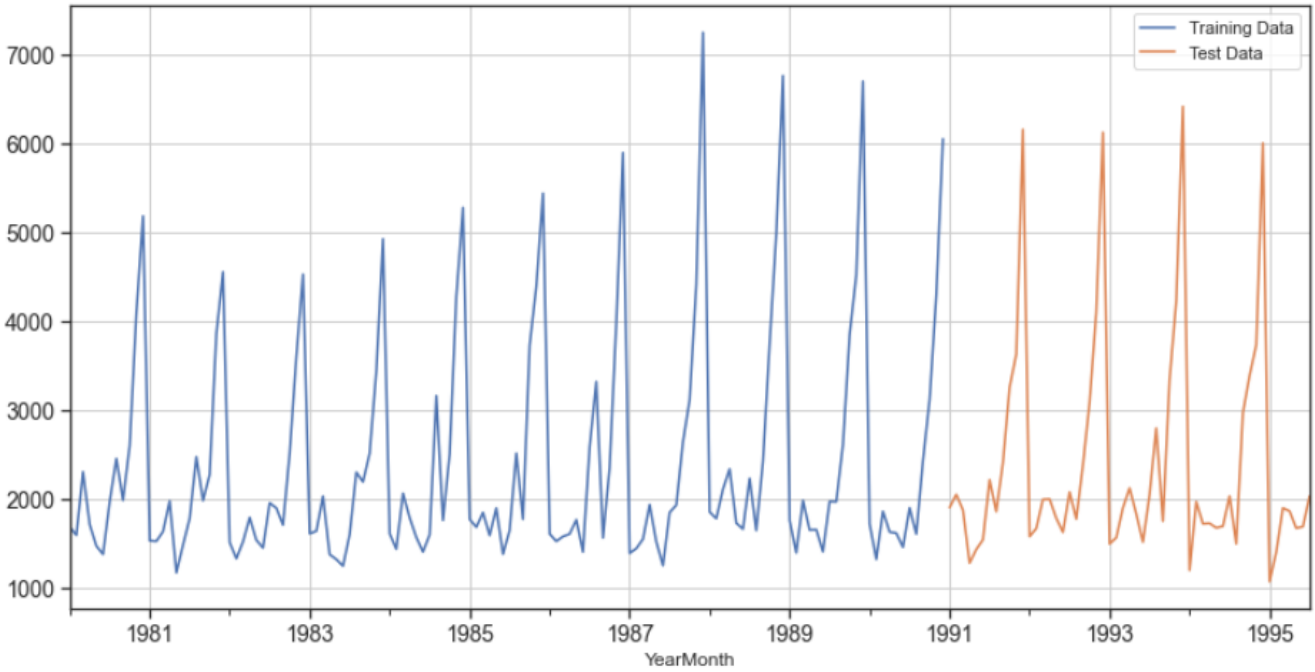
To decide between multiplicative and additive, we take into account lower range of residual, which in case of multiplicative is 0 to 1, while for additive is 0 to 1000. Hence we select the multiplicative model owing to a more stable residual plot and lower range of residuals.

# 3. SPLIT THE DATA INTO TRAINING AND TEST. THE TEST DATA SHOULD START IN 1991.

The data was split into train and test data sets, so that the machine learning models could be trained on the training set and the models could be further evaluated using the test data set.

As per the instructions given in the project we have split the data, around 1991. With training data from 1980 to 1990 December. Test data starts from the first month of January 1991 till the end.



## 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.

We started with simple models and gradually moved towards more and more complex models. We created below models as part of this question.

**Models Built:-**

1) Linear Regression Model
2) Naive Bayes Model
3) Simple Average Model
4) Moving Average Model - Rolling Window 2
5) Moving Average Model - Rolling Window 4
6) Moving Average Model - Rolling Window 6
7) Moving Average Model - Rolling Window 9
8) Simple Exponential Smoothing - AutoFit
9) Simple Exponential Smoothing - Using For Loop
10) Double Exponential Smoothing (Holt's Model) - AutoFit
11) Double Exponential Smoothing (Holt's Model) - Using For Loop
12) Tripple Exponential Smoothing (Holts - Winter Model) - AutoFit
13) Tripple Exponential Smoothing (Holts - Winter Model) - Using For Loop.

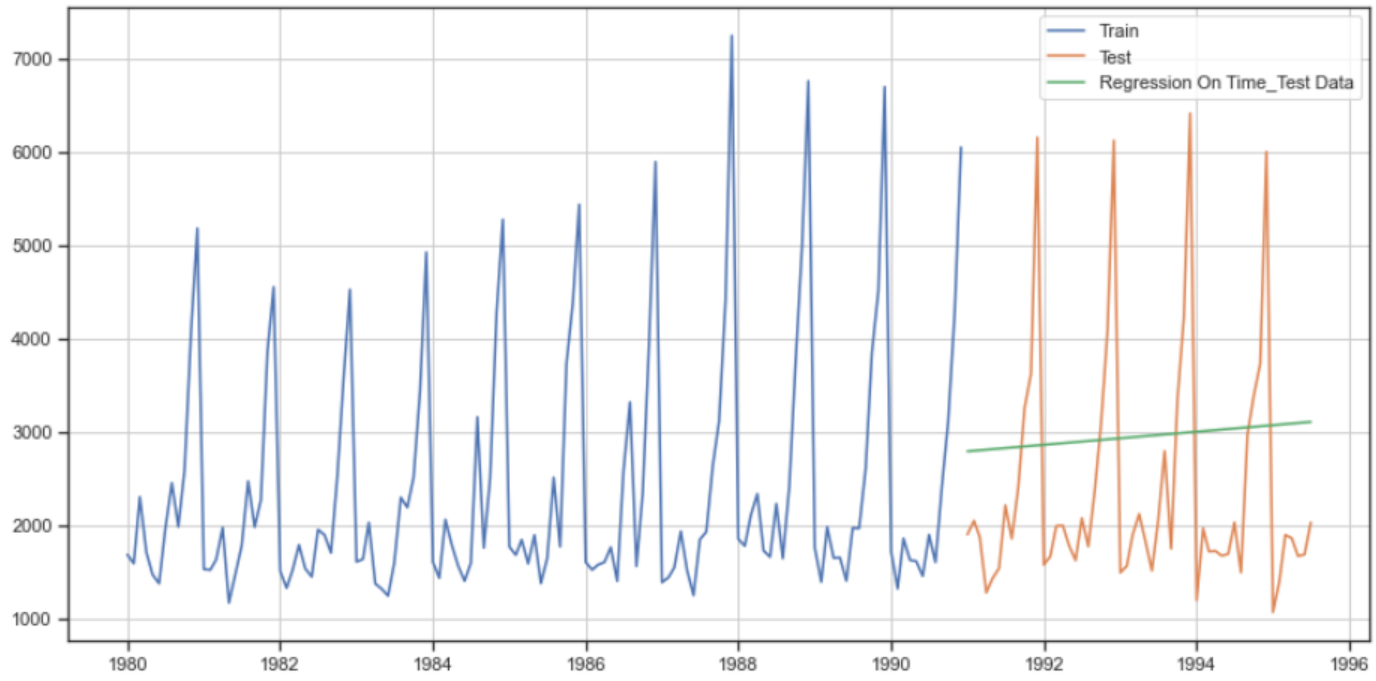
A total of 13 different models were created as part of this question and all of these were evaluated on the RMSE performance metric and compared at the end.

We also plotted prediction with actuals to visually see how accurate the results of each model were.

Each of these models will be discussed in subsequent pages.

## 1) Linear Regression Model

We started out with the simplest model, which was the Linear regression using sklearn library.This model tries to fit all the training points on a straight line and interpolate the line over the range of test vaues to predict the test values.



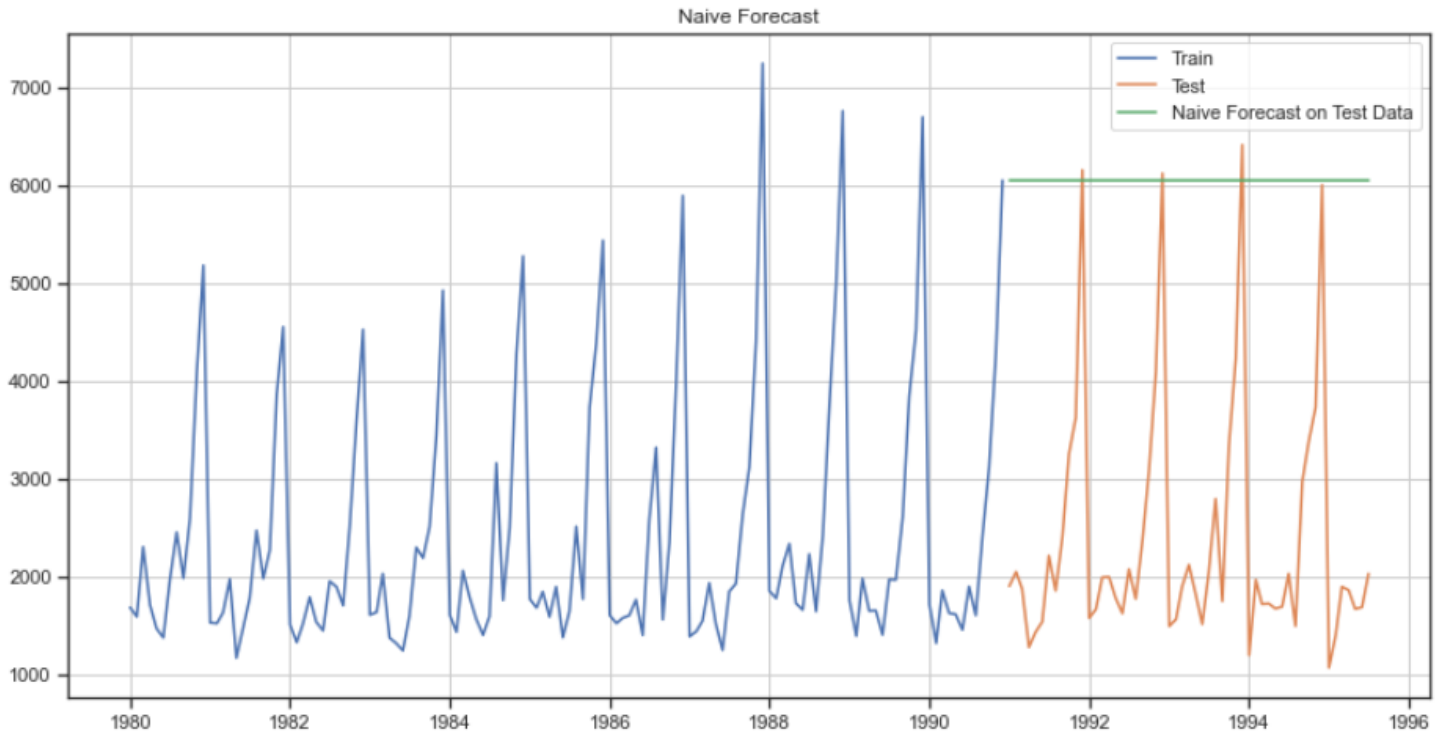

The green line indicates the predictions made by the model, while the orange values are the actual test values. It is clear the predicted values are very far off from the actual values, but this was expected as this is one of the simplest models.

Model was evaluated using the RMSE metric. Below is the RMSE calculated for this model.

| | Test RMSE |
|---|---|
| RegressionOnTime | 1389.135175 |

## 2) Naive Bayes Model

Following the linear model, we created the naive bayes model, which as the name suggests is very naive in nature, as it assumes the last observed value to be the future values. All the future values will be the same as the last observed value.



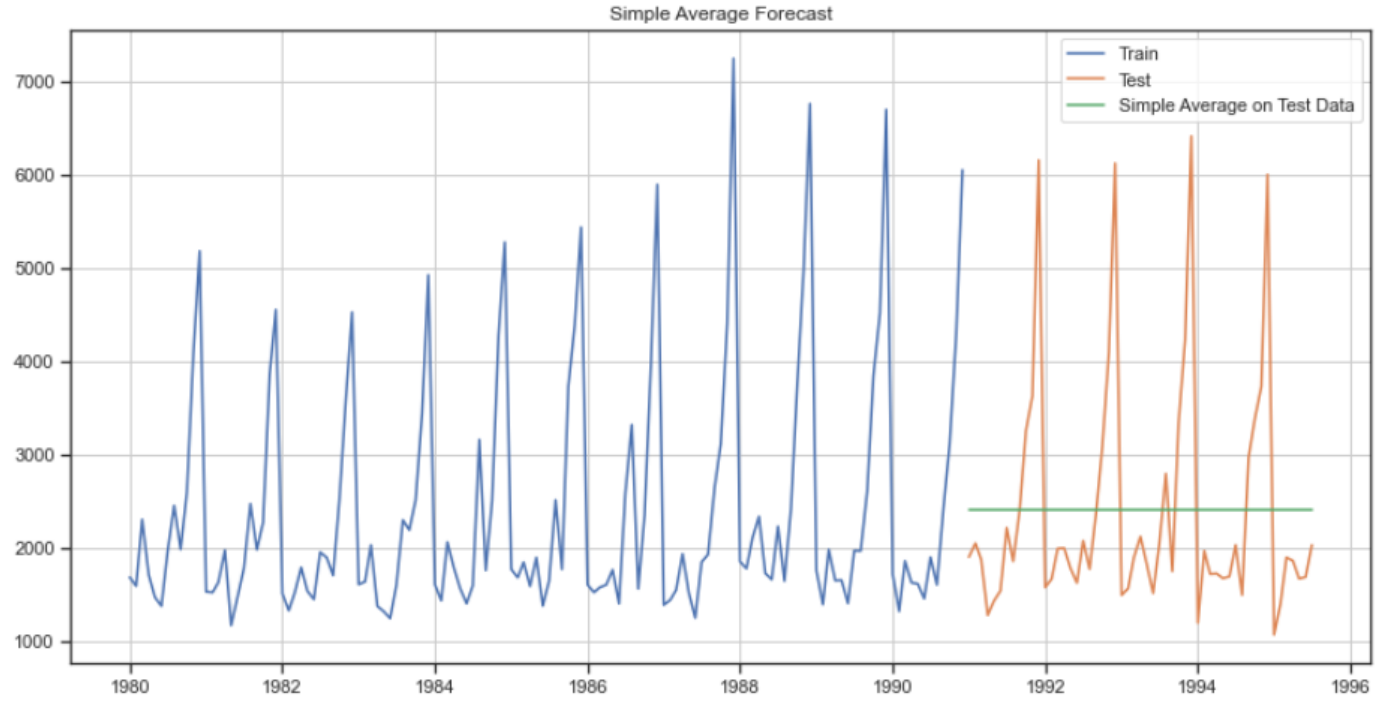

The green line indicates the predictions made by the model, while the orange values are the actual test values. It is clear the predicted values are very far off from the actual values, but once again this was expected from a naive model as it is just portraying the last observed value as the future values.

Model was evaluated using the RMSE metric. Below is the RMSE calculated for this model.

| | Test RMSE |
|---|---|
| NaiveModel | 3864.279352 |

## 3) Simple Average Model

A simple average model takes the mean value across the different years and then plots a straight line for the future values. It is also very simple model and does not provide good predictions, however we still went ahead for the sake of exploration.



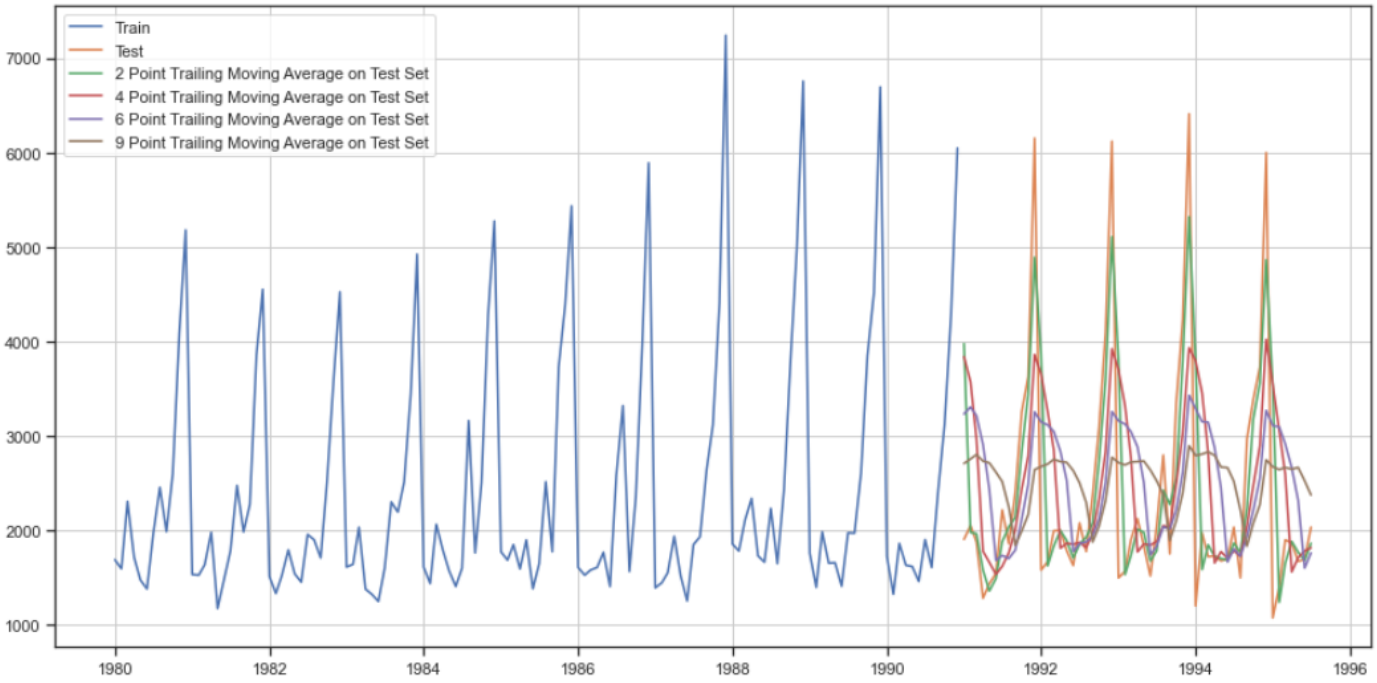

The green line indicates the predictions made by the model, while the orange values are the actual test values. It is clear the predicted values are very far off from the actual values, but once again this was expected as the predicted value is nothing but a simple mean of all the previous values.

Model was evaluated using the RMSE metric. Below is the RMSE calculated for this model.

| | Test RMSE |
|---|---|
| SimpleAverageModel | 1275.081804 |

## 4 - 7) Moving Average Models

We created multiple moving average models with rolling windows varying from 2 to 9. Rolling average is a better method than simple average as it takes into account only the previous n values to make the prediction, where n is the rolling window defined. This takes into account the recent trends and is in general more accurate. Higher the rolling window, smoother will be its curve, since more values are being taken into account.



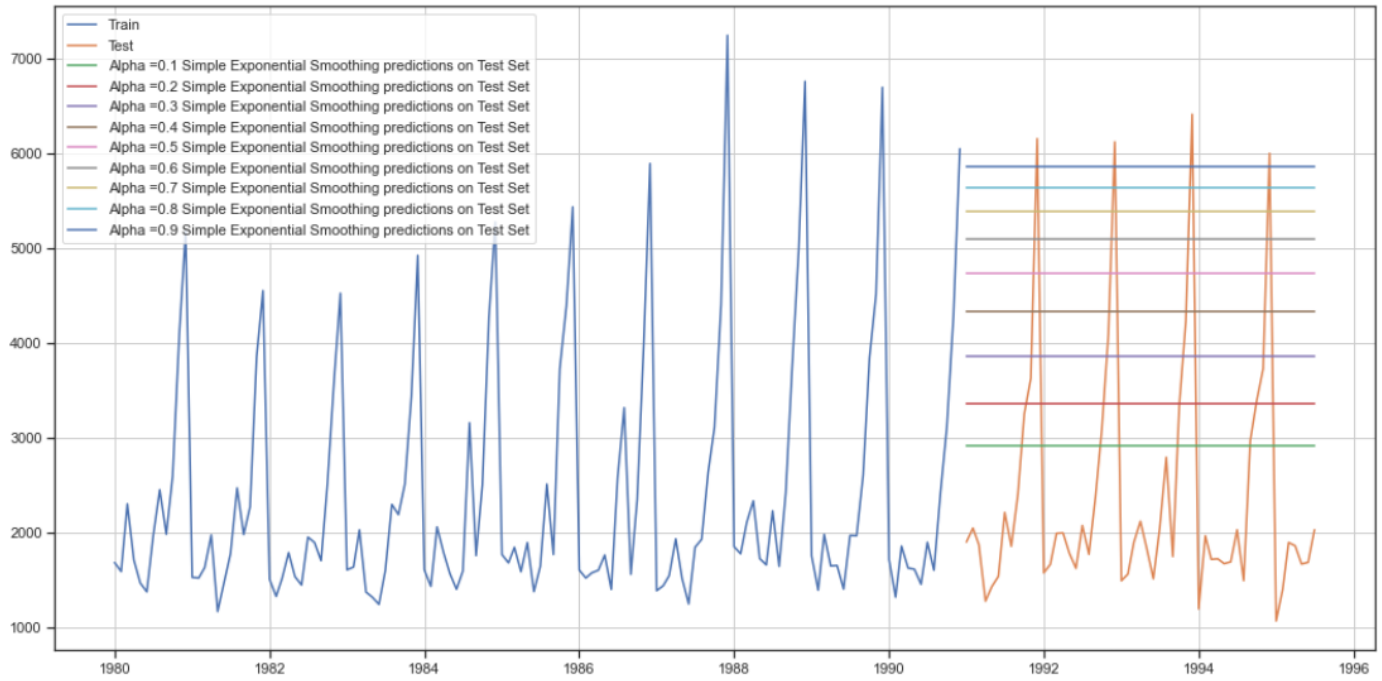

Output of different moving averages are show in the plot above. 2 point moving average lines moves the closest to the actual test values.

Model was evaluated using the RMSE metric. Below is the RMSE calculated for this model.

| | Test RMSE |
|---|---|
| 2pointTrailingMovingAverage | 813.400684 |
| 4pointTrailingMovingAverage | 1156.589694 |
| 6pointTrailingMovingAverage | 1283.927428 |
| 9pointTrailingMovingAverage | 1346.278315 |

## 8-9) Simple Exponential Smoothing

Simple exponential smoothing model was then bought into the picture, this model is good at explaining the level. We first used .fit() function without mentioning explicit value for alpha, letting the algorithm decide an optimum value on its own. After this we used.fit() giving smoothing_level / alpha in the range of 0.1 to 1 with a step size of 0.1. Below was the output plot for various smoothing_level values.




Output for various alpha values is shown by different color lines in the above plot.

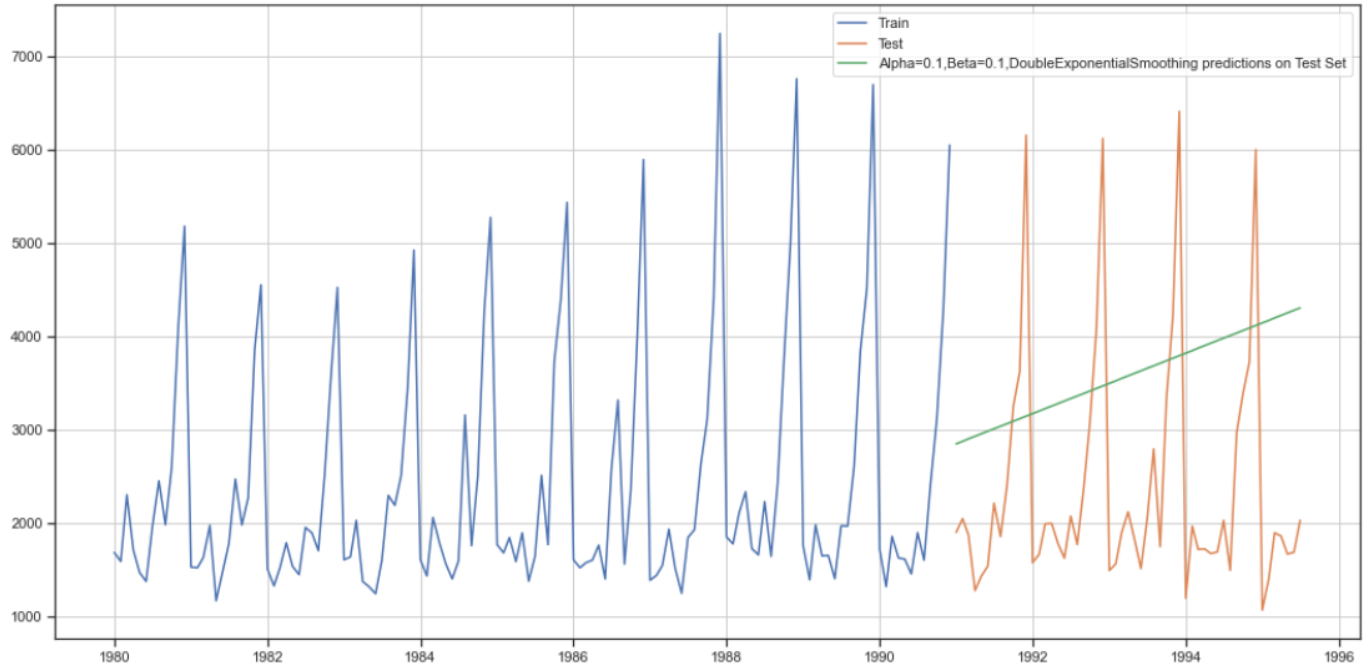
Models were evaluated using the RMSE metric. Below are the RMSE calculated for these models. Also final entry with least RMSE has been shown.

| | Alpha Values | Train RMSE | Test RMSE |
|---|---|---|---|
| 0 | 0.1 | 1333.873836 | 1375.393398 |
| 1 | 0.2 | 1356.042987 | 1595.206839 |
| 2 | 0.3 | 1359.511747 | 1935.507132 |
| 3 | 0.4 | 1352.588879 | 2311.919615 |
| 4 | 0.5 | 1344.004369 | 2666.351413 |
| 5 | 0.6 | 1338.805381 | 2979.204388 |
| 6 | 0.7 | 1338.844308 | 3249.944092 |
| 7 | 0.8 | 1344.462091 | 3483.801006 |
| 8 | 0.9 | 1355.723518 | 3686.794285 |

| | Test RMSE |
|---|---|
| Alpha=0.0,SimpleExponentialSmoothing_Auto_Fit | 1275.081823 |
| Alpha=0.1,SimpleExponentialSmoothing | 1375.393398 |

## 10-11) Double Exponential Smoothing (Holts Model)

After performing Simple Exponential smoothing, we also performed double exponential smoothing, which apart from level also takes into account the trend of the series. Here again we used .fit() first without explicitly mentioning smoothing_level (alpha) and smoothing_slope (beta). After this we used a for loop with range (0.1,1.1) with step size of 0.1. Below is the output plot for the best value observed out of all the attempts, which was alpha = 0.1 and beta = 0.1.




Output for best alpha and beta values is shown by the green color line in the above plot.

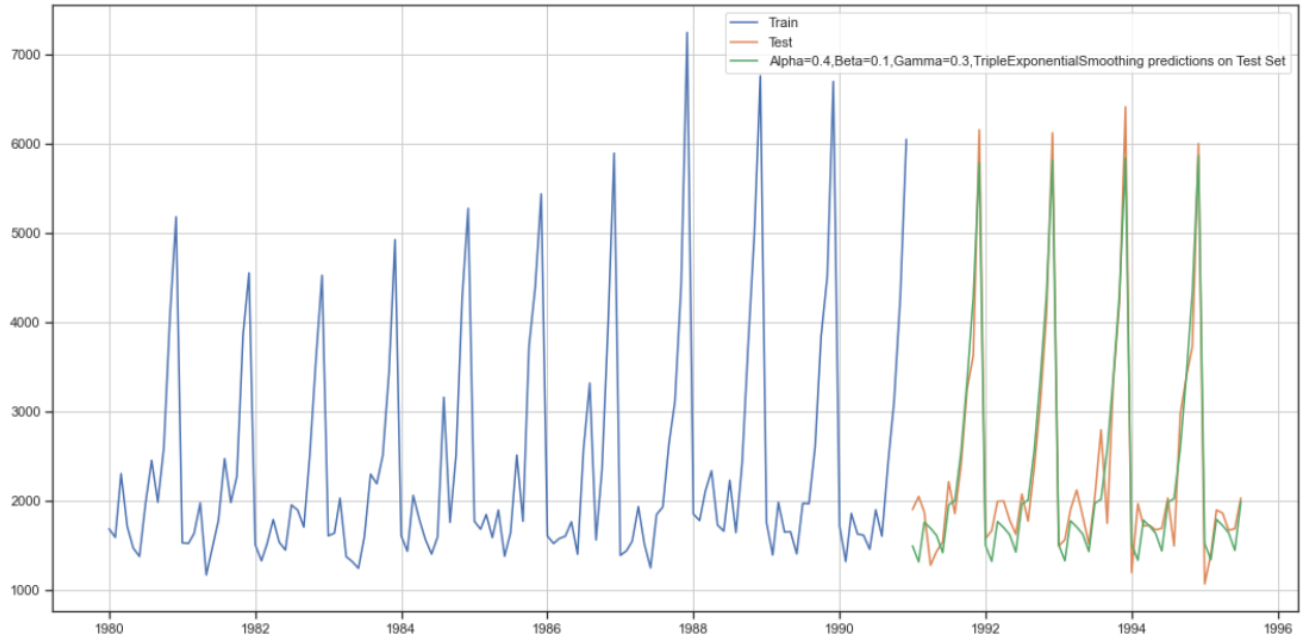
Models were evaluated using the RMSE metric. Below are the RMSE calculated for these models. Also final entry with least RMSE has been shown.

| | Alpha Values | Beta Values | Train RMSE | Test RMSE |
|---|---|---|---|---|
| 0 | 0.1 | 0.1 | 1382.520870 | 1778.564670 |
| 1 | 0.1 | 0.2 | 1413.598835 | 2599.439986 |
| 2 | 0.1 | 0.3 | 1445.762015 | 4293.084674 |
| 3 | 0.1 | 0.4 | 1480.897776 | 6039.537339 |
| 4 | 0.1 | 0.5 | 1521.108657 | 7390.522201 |
| ... | ... | ... | ... | ... |
| 95 | 1.0 | 0.6 | 1753.402326 | 49327.087977 |
| 96 | 1.0 | 0.7 | 1825.187155 | 52655.765663 |
| 97 | 1.0 | 0.8 | 1902.013709 | 55442.273880 |
| 98 | 1.0 | 0.9 | 1985.368445 | 57823.177011 |
| 99 | 1.0 | 1.0 | 2077.672157 | 59877.076519 |

| | Test RMSE |
|---|---|
| Alpha=0.6477,Beta=0.0,DoubleExponentialSmoothing_Auto_Fit | 1275.081823 |
| Alpha=0.1,Beta=0.1,DoubleExponentialSmoothing | 1778.564670 |

## 12-13) Triple Exponential Smoothing (Holts-Winter Model)

After performing double exponential smoothing which takes into account only the level and the trends, we moved ahead and tried triple exponential smoothing or the Holts Winter model which not only takes into account level and trends but also takes into account the seasonality present in the series. Again in the same way an auto fit approach and a for loop approach was used here. Here while using the for loop we also took into account both additive and multiplicative trends and seasonality and various permutation and combinations were tried and finally the best model out of all was plotted below.




Output for best alpha, beta and gamma values is shown by the green color line in the above plot. Best model had both multiplicative trend as well as seasonality.

Models were evaluated using the RMSE metric. Below are the RMSE calculated for these models. Also final entry with least RMSE has been shown.

| | Alpha Values | Beta Values | Gamma Values | Train RMSE | Test RMSE | Method |
|---|---|---|---|---|---|---|
| **2245** | 0.4 | 0.1 | 0.3 | 386.113240 | 326.869742 | tm_sm |
| **2163** | 0.3 | 0.1 | 0.2 | 381.218292 | 329.977234 | tm_sm |
| **1301** | 0.4 | 0.1 | 0.2 | 389.772245 | 336.715250 | ta_sm |
| **85** | 0.1 | 0.9 | 0.6 | 438.525019 | 338.458417 | ta_sa |
| **139** | 0.2 | 0.4 | 1.0 | 483.673805 | 338.840640 | ta_sa |

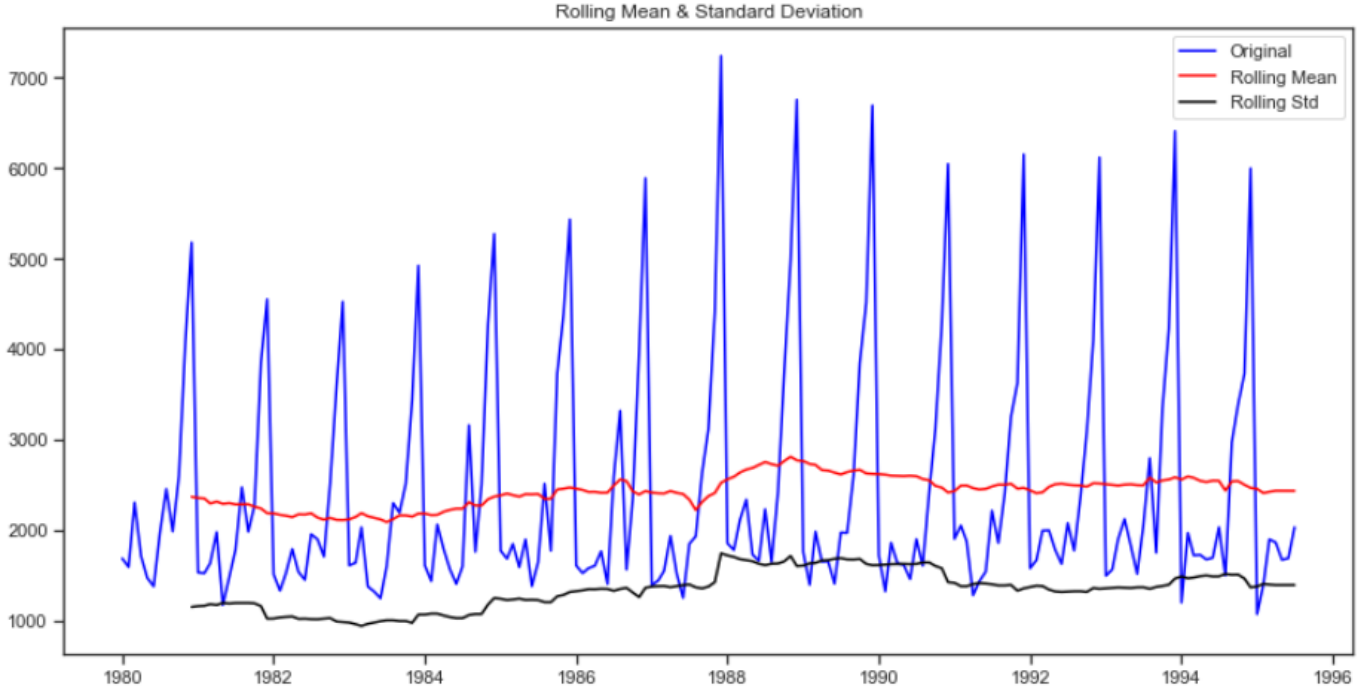
| | Test RMSE |
|---|---|
| **Alpha=0.08621,Beta=1.3722,Gamma=0.4763,TrippleExponentialSmoothing_Auto_Fit** | 1275.081823 |
| **Alpha=0.4,Beta=0.1,Gamma=0.3,TripleExponentialSmoothing** | 326.869742 |

## 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.

We made use of Dickey-Fuller hypothesis test to determine if the given series is stationary or not. We also plotted the series along with its rolling mean and rolling standard deviations.

Hypothesis for our test:-

Null Hypothesis --> H0 --> Series is not stationary
Alternate Hypothesis --> Ha --> Series is stationary

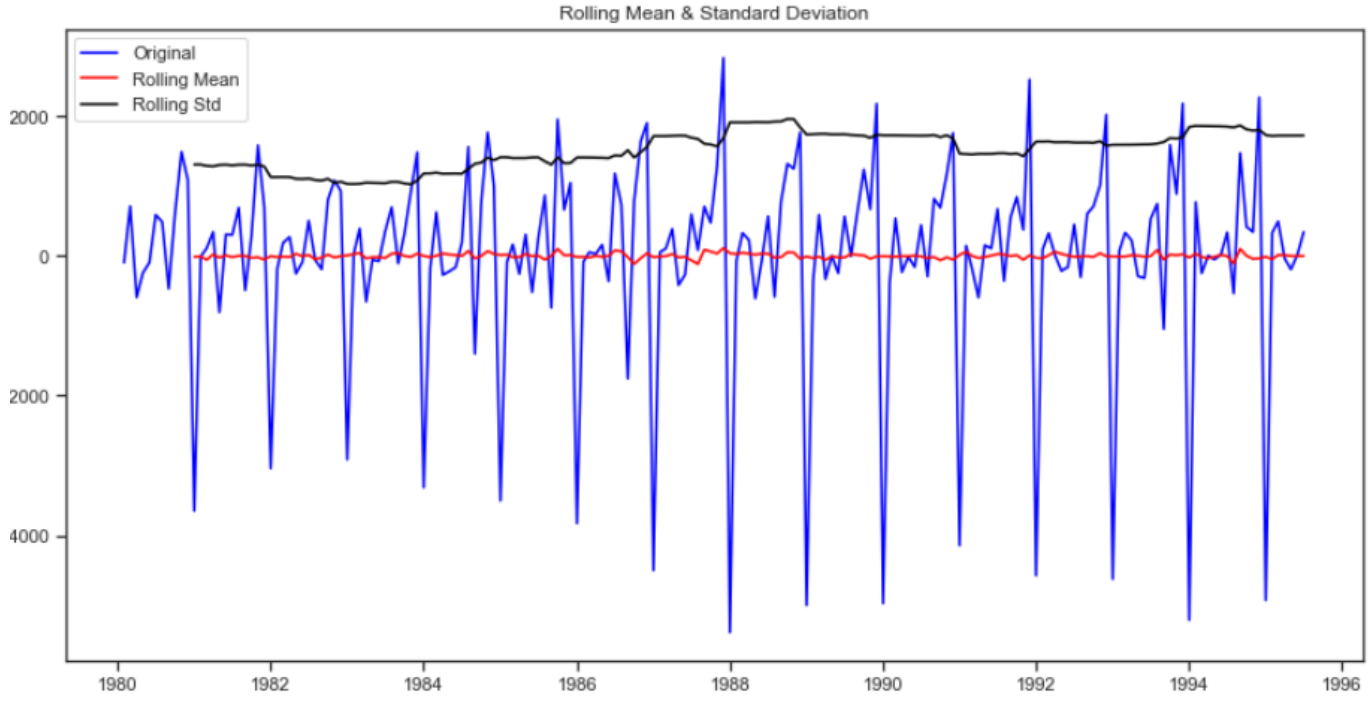



We could see after performing the hypothesis test that the p-values was 0.601061 which was not less than 0.05 (our alpha value) hence we failed to reject the null hypothesis, which implies the Series is not stationary in nature.

Also the rolling means was not a straight line. Now we needed to make the series stationary.

In order to try and make the series stationary we used the differencing approach. We used .diff() function on the existing series without any argument, implying the default diff value of 1 and also dropped the NaN values, since differencing of order 1 would generate the first value as NaN which need to be dropped.

Post doing this, we once again ran the Dickey-Fuller test with the same hypothesis as earlier, to see if differencing made our series stationary.




This time the series became stationary in nature and the p-value obtained from Dickey - Fuller test was 0.000, which is obviously less than 0.05.

Hence the null hypothesis that the series is not stationary at difference = 1 was rejected, which implied that the series has indeed become stationary after we performed the differencing.

Null hypothesis was rejected since the p-value was less than alpha i.e. 0.05. Also the rolling mean plot was a straight line this time around. Also the series looked more or less the same from both the directions, indicating stationarity.

We could now proceed ahead with ARIMA/ SARIMA models, since we had made the series stationary.

## 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.

# Auto - ARIMA Model

Starting with ARIMA model, we made use of ARIMA function from statsmodel library in Python. We employed a for loop for determining the optimum values of p,d,q, where p is the order of the AR (Auto-Regressive) part of the model, while q is the order of the MA (Moving Average) part of the model. d is the differencing that is required to make the series stationary.

p,q values in the range of (0,4) were given to the for loop, while a fixed value of 1 was given for d, since we had already determined d to be 1, while checking for stationarity using ADF test.

Below were the different models that for loop evaluated.

```
Some parameter combinations for the Model...
Model: (0, 1, 1)
Model: (0, 1, 2)
Model: (0, 1, 3)
Model: (1, 1, 0)
Model: (1, 1, 1)
Model: (1, 1, 2)
Model: (1, 1, 3)
Model: (2, 1, 0)
Model: (2, 1, 1)
Model: (2, 1, 2)
Model: (2, 1, 3)
Model: (3, 1, 0)
Model: (3, 1, 1)
Model: (3, 1, 2)
Model: (3, 1, 3)
```

Akaike information criterion (AIC) value was evaluated for each of these models and the model with least AIC value was selected.

| | param | AIC |
|---|---|---|
| **10** | (2, 1, 2) | 2210.618988 |
| **15** | (3, 1, 3) | 2225.661559 |
| **14** | (3, 1, 2) | 2228.928726 |
| **11** | (2, 1, 3) | 2229.358094 |
| **9** | (2, 1, 1) | 2232.360490 |
| **2** | (0, 1, 2) | 2232.783098 |
| **3** | (0, 1, 3) | 2233.016605 |
| **6** | (1, 1, 2) | 2233.597647 |
| **13** | (3, 1, 1) | 2233.921755 |
| **7** | (1, 1, 3) | 2234.574142 |
| **5** | (1, 1, 1) | 2235.013945 |
| **12** | (3, 1, 0) | 2259.471555 |
| **8** | (2, 1, 0) | 2262.035600 |
| **1** | (0, 1, 1) | 2264.906437 |
| **4** | (1, 1, 0) | 2268.528061 |
| **0** | (0, 1, 0) | 2269.582796 |

Model with p=2, d=1, q=2 was found be the optimum model in this scenario with the lowest AIC value of 2210.618988.

Full report was generated for this particular model and RMSE was also calculated for this selection, so that it can then be compared with other models later.

PFB the summary report for the ARIMA model with values (p=2,d=1,q=2).

ARIMA Model Results

| | | | |
|---|---|---|---|
| Dep. Variable: | D.Sales | No. Observations: | 131 |
| Model: | ARIMA(2, 1, 2) | Log Likelihood | -1099.309 |
| Method: | css-mle | S.D. of innovations | 1012.929 |
| Date: | Sun, 08 Nov 2020 | AIC | 2210.619 |
| Time: | 11:50:59 | BIC | 2227.870 |
| Sample: | 02-01-1980 - 12-01-1990 | HQIC | 2217.629 |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| const | 5.5854 | 0.517 | 10.806 | 0.000 | 4.572 | 6.598 |
| ar.L1.D.Sales | 1.2699 | 0.075 | 17.046 | 0.000 | 1.124 | 1.416 |
| ar.L2.D.Sales | -0.5602 | 0.074 | -7.618 | 0.000 | -0.704 | -0.416 |
| ma.L1.D.Sales | -1.9974 | 0.042 | -47.112 | 0.000 | -2.080 | -1.914 |
| ma.L2.D.Sales | 0.9974 | 0.042 | 23.497 | 0.000 | 0.914 | 1.081 |

Roots

| | Real | Imaginary | Modulus | Frequency |
|---|---|---|---|---|
| AR.1 | 1.1334 | -0.7074j | 1.3361 | -0.0888 |
| AR.2 | 1.1334 | +0.7074j | 1.3361 | 0.0888 |
| MA.1 | 1.0006 | +0.0000j | 1.0006 | 0.0000 |
| MA.2 | 1.0020 | +0.0000j | 1.0020 | 0.0000 |

AIC value - 2210.619
BIC value - 227.870

Forecast was done and compared to the actual test values to determine the RMSE.

RMSE values are as below.

| | Test RMSE |
|---|---|
| p=2,d=1,q=2,Auto_ARIMA | 1374.484105 |

# Auto - SARIMA Model

Moving for SARIMA, we now also take into account the seasonality of the series, to enable better predictions. Looking at the lag heat map which we had plotted earlier, we were able to determine the seasonality to be 12. This was further verified by plotting ACF and PACF graphs which showed significant spikes at 12,14,26,48 etc lags indicating a seasonality of 12 i.e. yearly seasonality. ACF and PACF graphs will be shown later when manual ARIMA and SARIMA models are discussed.

A similar for loop with below values was employed, resulting in the models shown below.

```
p = q = range(0, 4)
d= range(0,2)
D = range(0,2)
pdq = list(itertools.product(p, d, q))
model_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, D, q))]
```

```
Examples of some parameter combinations for Model...
Model: (0, 0, 1)(0, 0, 1, 12)
Model: (0, 0, 2)(0, 0, 2, 12)
Model: (0, 0, 3)(0, 0, 3, 12)
Model: (0, 1, 0)(0, 1, 0, 12)
Model: (0, 1, 1)(0, 1, 1, 12)
Model: (0, 1, 2)(0, 1, 2, 12)
Model: (0, 1, 3)(0, 1, 3, 12)
Model: (1, 0, 0)(1, 0, 0, 12)
Model: (1, 0, 1)(1, 0, 1, 12)
Model: (1, 0, 2)(1, 0, 2, 12)
Model: (1, 0, 3)(1, 0, 3, 12)
Model: (1, 1, 0)(1, 1, 0, 12)
Model: (1, 1, 1)(1, 1, 1, 12)
Model: (1, 1, 2)(1, 1, 2, 12)
Model: (1, 1, 3)(1, 1, 3, 12)
Model: (2, 0, 0)(2, 0, 0, 12)
Model: (2, 0, 1)(2, 0, 1, 12)
Model: (2, 0, 2)(2, 0, 2, 12)
Model: (2, 0, 3)(2, 0, 3, 12)
Model: (2, 1, 0)(2, 1, 0, 12)
Model: (2, 1, 1)(2, 1, 1, 12)
Model: (2, 1, 2)(2, 1, 2, 12)
Model: (2, 1, 3)(2, 1, 3, 12)
Model: (3, 0, 0)(3, 0, 0, 12)
Model: (3, 0, 1)(3, 0, 1, 12)
Model: (3, 0, 2)(3, 0, 2, 12)
Model: (3, 0, 3)(3, 0, 3, 12)
Model: (3, 1, 0)(3, 1, 0, 12)
Model: (3, 1, 1)(3, 1, 1, 12)
Model: (3, 1, 2)(3, 1, 2, 12)
Model: (3, 1, 3)(3, 1, 3, 12)
```

Akaike information criterion (AIC) value was evaluated for each of these models and the model with least AIC value was selected. Here only the top 5 models are shown.

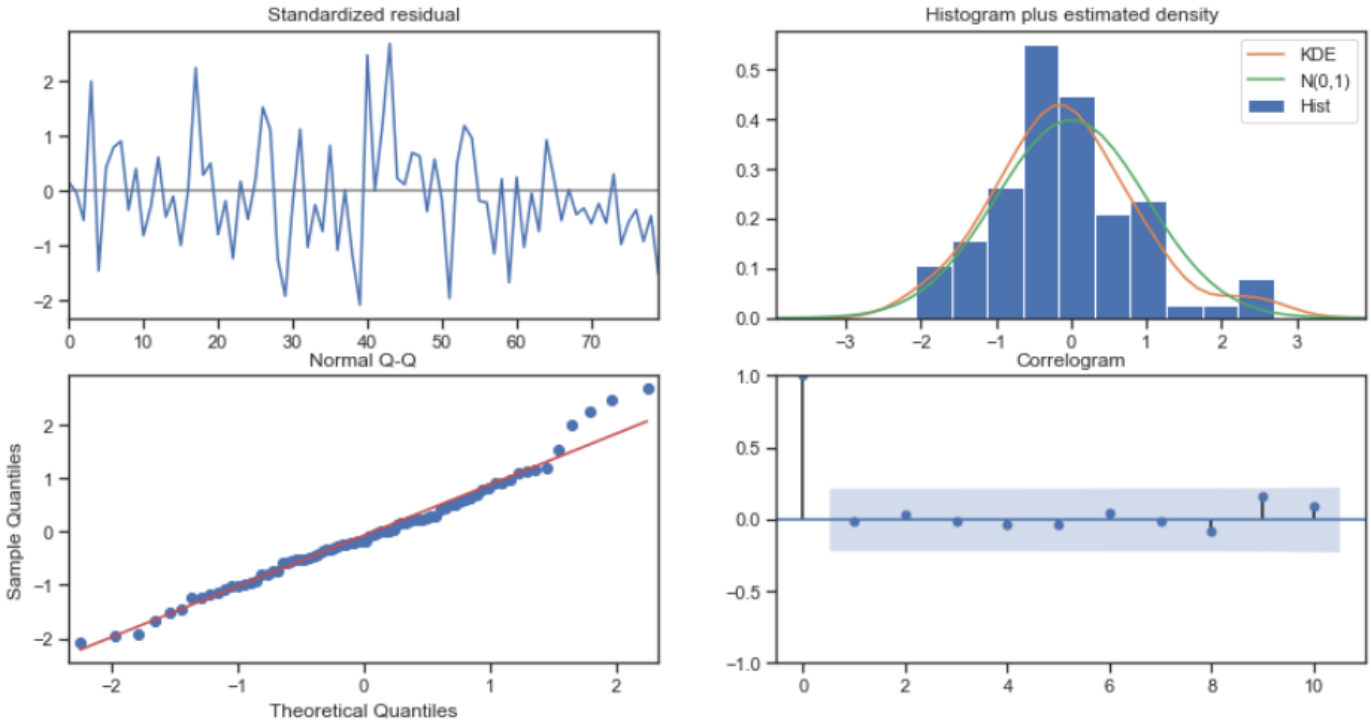
| | param | seasonal | AIC |
|---|---|---|---|
| **1020** | (3, 1, 3) | (3, 1, 0, 12) | 1213.282561 |
| **1021** | (3, 1, 3) | (3, 1, 1, 12) | 1215.213168 |
| **956** | (3, 1, 1) | (3, 1, 0, 12) | 1215.898777 |
| **1022** | (3, 1, 3) | (3, 1, 2, 12) | 1216.480002 |
| **988** | (3, 1, 2) | (3, 1, 0, 12) | 1216.859180 |

PFB the summary report for the best SARIMA model with values (3,1,3)(3,1,0,12).

```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                                   y   No. Observations:                  132
Model:             SARIMAX(3, 1, 3)x(3, 1, [], 12)   Log Likelihood                -596.641
Date:                             Sun, 08 Nov 2020   AIC                           1213.283
Time:                                     13:27:29   BIC                           1237.103
Sample:                                          0   HQIC                          1222.833
                                               - 132
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -1.6140      0.176     -9.177      0.000      -1.959      -1.269
ar.L2         -0.6120      0.299     -2.047      0.041      -1.198      -0.026
ar.L3          0.0863      0.161      0.538      0.591      -0.228       0.401
ma.L1          0.9854      0.469      2.102      0.036       0.067       1.904
ma.L2         -0.8739      0.166     -5.268      0.000      -1.199      -0.549
ma.L3         -0.9465      0.486     -1.947      0.052      -1.899       0.006
ar.S.L12      -0.4518      0.142     -3.191      0.001      -0.729      -0.174
ar.S.L24      -0.2343      0.144     -1.624      0.104      -0.517       0.049
ar.S.L36      -0.1006      0.122     -0.828      0.408      -0.339       0.138
sigma2      1.839e+05   8.91e+04      2.063      0.039    9149.728    3.59e+05
==========================================================================================
Ljung-Box (Q):                       23.20   Jarque-Bera (JB):                 4.06
Prob(Q):                              0.98   Prob(JB):                         0.13
Heteroskedasticity (H):               0.73   Skew:                             0.48
Prob(H) (two-sided):                  0.42   Kurtosis:                         3.54
==========================================================================================

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

We also plotted the graphs for the residual to determine if any further information can be extracted or all the usable information has already been extracted. Below were the plots for the best auto SARIMA model.




Looking at above plots we can see the residuals are now forming an almost normal distribution ranging withing the empirical range of (-3,3) for a normal distribution.

Also for the Q-Q plot we can see the points are almost overlapping the x=y straight line.

Correlogram also does not indicate that there are any pattern left to be extracted.

Hence looking at the above, we can safely assume that all the actionable information has been extracted by our model and it is a good model in this respect.

Below is the RMSE

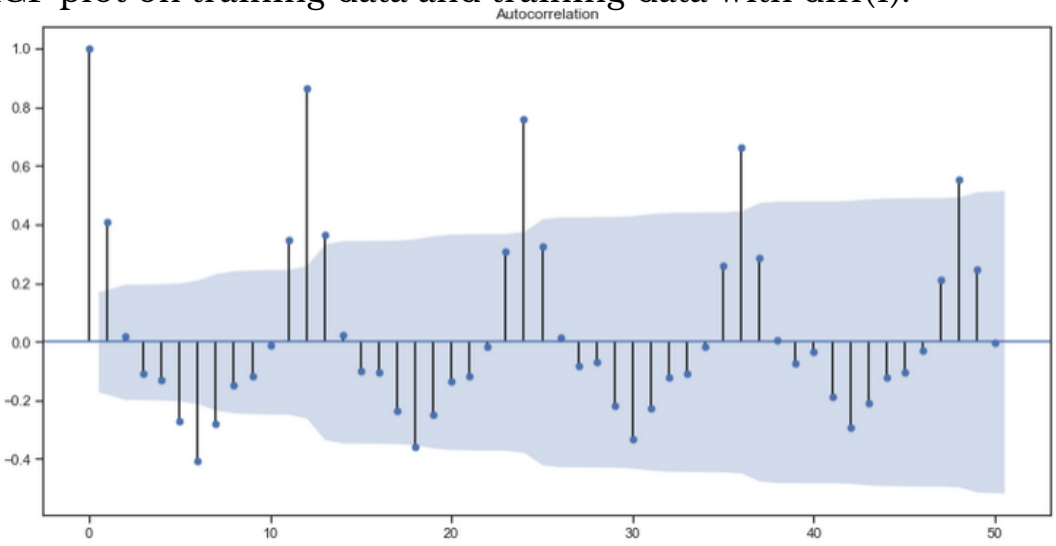
| | Test RMSE |
|---|---|
| (3,1,3),(3,1,0,12),Auto_SARIMA | 331.632012 |

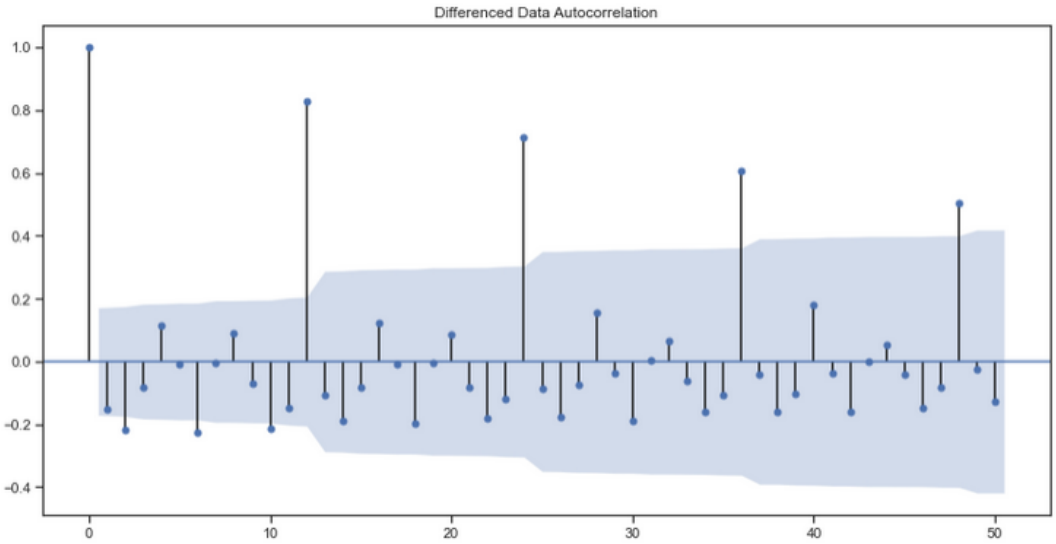
## 7. BUILD ARIMA/SARIMA MODELS BASED ON THE CUT-OFF POINTS OF ACF AND PACF ON THE TRAINING DATA AND EVALUATE THIS MODEL ON THE TEST DATA USING RMSE.

# Manual- ARIMA Model

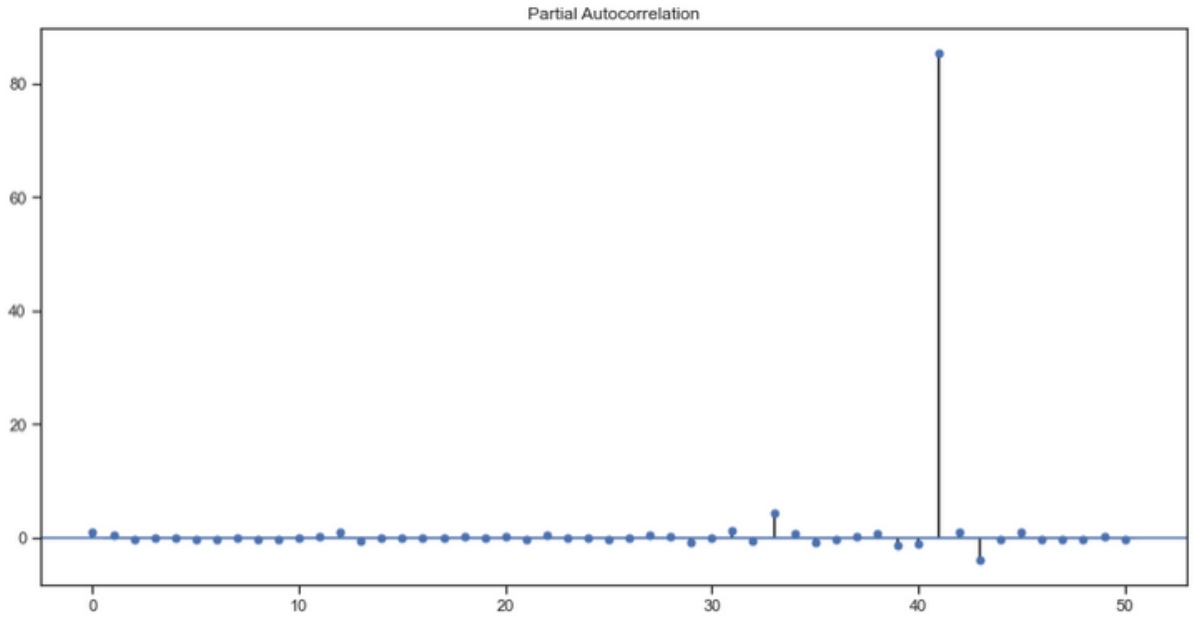
In order to make a manual ARIMA model, we have to first plot the ACF and PACF plots on the training data, based on which we will be able to determine the p,d,q values for the model.

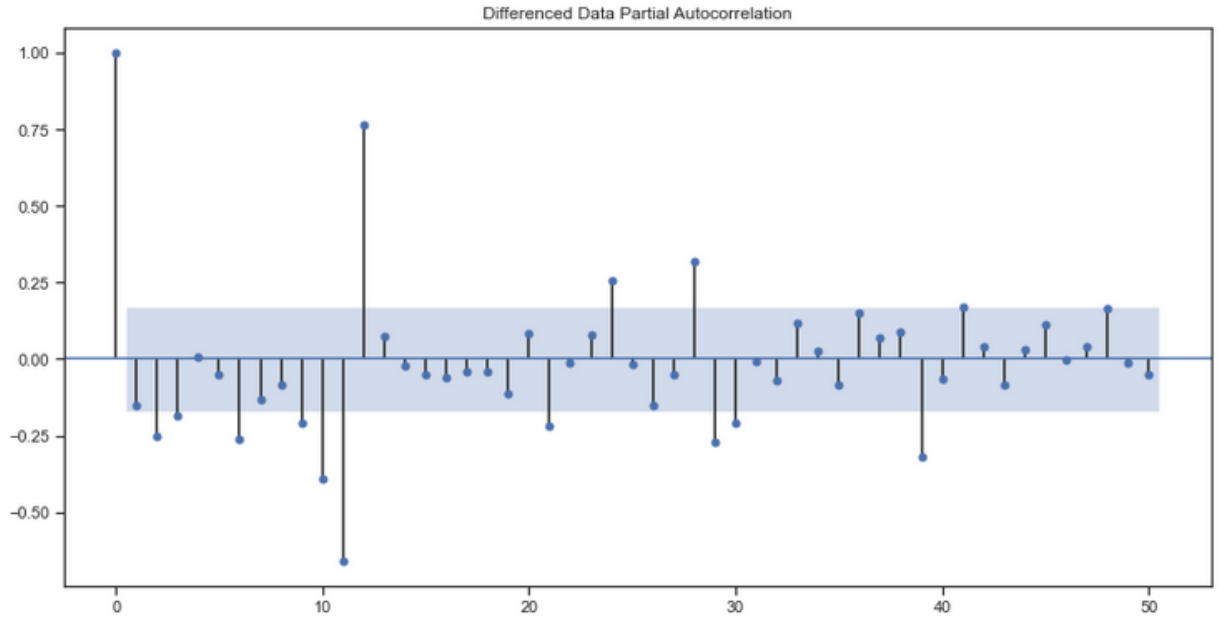
PFB the ACF plot on training data and training data with diff(1).



Now plotting the PACF graph for the training data.




We will use ACF plot to determine order of MA i.e. value of q.
We will use PACF plot to determine order of AR i.e value of p.

Looking at ACF plot we can see a shard decay after lag 1 for original as well as differenced data. hence we select the q value to be 1. i.e. q=1.

Looking at PACF plot we can again see significant bars till lag 1 for differenced series which is stationary in nature, post 1 the decay is large enough. Hence we choose p value to be 1. i.e. p=1.

d values will be 1, since we had seen earlier that the series is stationary with lag1.

Hence the values selected for manual ARIMA:-
p=1,
d=1,
q=1

A ARIMA model was built with above params and AIC value observed for this model was 2235.014. Below is the summary from this manual ARIMA model.

ARIMA Model Results

| | | | |
|---|---|---|---|
| Dep. Variable: | D.y | No. Observations: | 131 |
| Model: | ARIMA(1, 1, 1) | Log Likelihood | -1113.507 |
| Method: | css-mle | S.D. of innovations | 1171.377 |
| Date: | Sun, 08 Nov 2020 | AIC | 2235.014 |
| Time: | 23:32:17 | BIC | 2246.515 |
| Sample: | 1 | HQIC | 2239.687 |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| const | 6.7490 | 4.616 | 1.462 | 0.144 | -2.299 | 15.797 |
| ar.L1.D.y | 0.4289 | 0.082 | 5.221 | 0.000 | 0.268 | 0.590 |
| ma.L1.D.y | -1.0000 | 0.019 | -51.962 | 0.000 | -1.038 | -0.962 |

Roots

| | Real | Imaginary | Modulus | Frequency |
|---|---|---|---|---|
| AR.1 | 2.3313 | +0.0000j | 2.3313 | 0.0000 |
| MA.1 | 1.0000 | +0.0000j | 1.0000 | 0.0000 |

RMSE for manual ARIMA is as below

| | Test RMSE |
|---|---|
| (1,1,1),Manual_ARIMA | 1461.668081 |

# Manual SARIMA Model

Looking at the ACF and PACF plots for training data, we can clearly see significant spikes at lags 12,24,36,48 etc, indicating a seasonality of 12. The parameters used for manual SARIMA model are as below.

SARIMAX(1, 1, 1)x(1, 1, 1, 12)

Below is the summary of the manual SARIMA model, having the AIC value of 1570.672.

SARIMAX Results

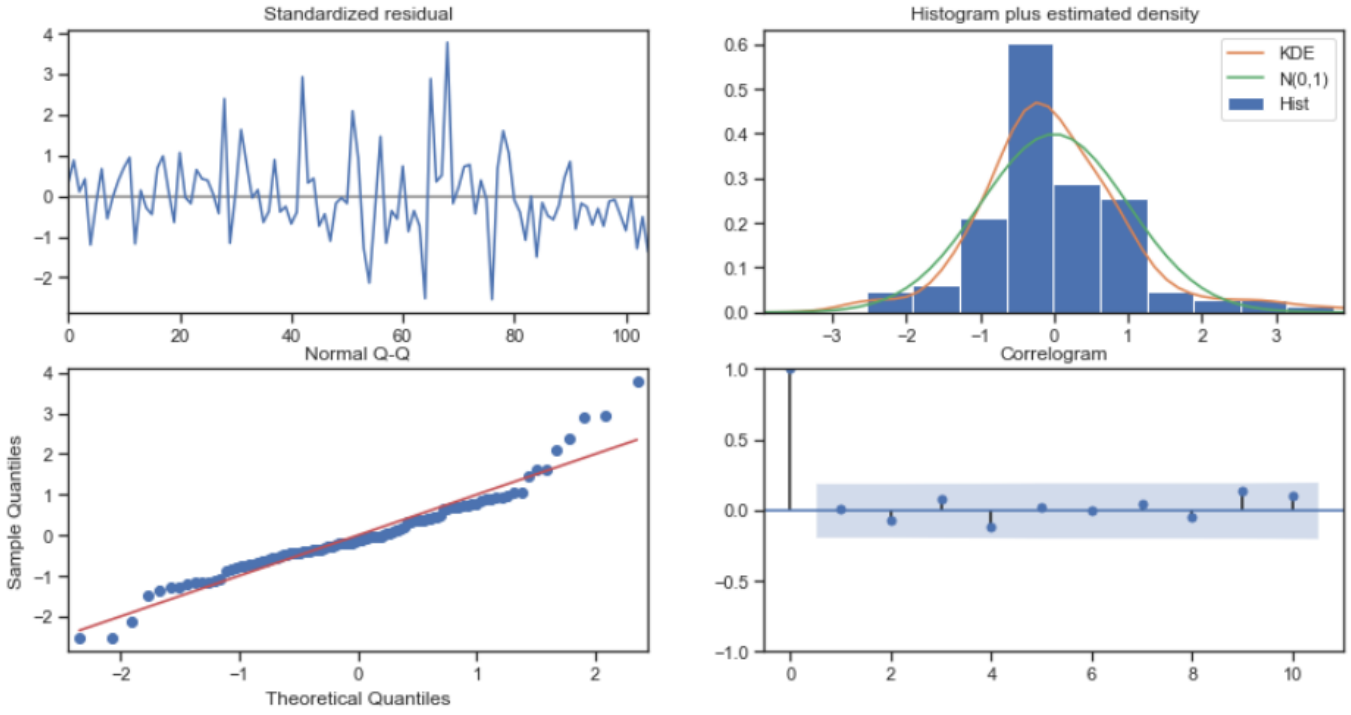
| | | | |
|---|---|---|---|
| Dep. Variable: | y | No. Observations: | 132 |
| Model: | SARIMAX(1, 1, 1)x(1, 1, 1, 12) | Log Likelihood | -780.336 |
| Date: | Sun, 08 Nov 2020 | AIC | 1570.672 |
| Time: | 14:27:01 | BIC | 1583.942 |
| Sample: | 0 | HQIC | 1576.050 |
| | - 132 | | |
| Covariance Type: | opg | | |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| ar.L1 | 0.1662 | 0.120 | 1.388 | 0.165 | -0.068 | 0.401 |
| ma.L1 | -0.9398 | 0.057 | -16.511 | 0.000 | -1.051 | -0.828 |
| ar.S.L12 | -0.0917 | 0.205 | -0.447 | 0.655 | -0.494 | 0.311 |
| ma.S.L12 | -0.3907 | 0.203 | -1.925 | 0.054 | -0.788 | 0.007 |
| sigma2 | 1.657e+05 | 1.65e+04 | 10.061 | 0.000 | 1.33e+05 | 1.98e+05 |

| | | | |
|---|---|---|---|
| Ljung-Box (Q): | 24.90 | Jarque-Bera (JB): | 33.96 |
| Prob(Q): | 0.97 | Prob(JB): | 0.00 |
| Heteroskedasticity (H): | 1.15 | Skew: | 0.77 |
| Prob(H) (two-sided): | 0.68 | Kurtosis: | 5.33 |

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

We also plotted the graphs for the residual to determine if any further information can be extracted or all the usable information has already been extracted. Below were the plots for the manual SARIMA model.




Looking at above plots we can see the residuals are now forming an almost normal distribution ranging withing the empirical range of (-3,3) for a normal distribution.

Also for the Q-Q plot we can see the points are almost overlapping the x=y straight line.

Correlogram also does not indicate that there are any pattern left to be extracted.

Hence looking at the above, we can safely assume that all the actionable information has been extracted by our model and it is a good model in this respect.

Below is the RMSE

| | Test RMSE |
|---|---|
| (1,1,1)(1,1,1,12),Manual_SARIMA | 375.614011 |

## 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.

PFB the dataframe containing multiple models that we have created along with their RMSE values. The values have been sorted in ascending order, starting with the lowest RMSE, hence the best model is on the top while the worst performing model is at the bottom.

| | Test RMSE |
|---|---|
| Alpha=0.4,Beta=0.1,Gamma=0.3,TripleExponentialSmoothing | 326.869742 |
| (3,1,3),(3,1,0,12),Auto_SARIMA | 331.632012 |
| (1,1,1)(1,1,1,12),Manual_SARIMA | 375.614011 |
| 2pointTrailingMovingAverage | 813.400684 |
| 4pointTrailingMovingAverage | 1156.589694 |
| SimpleAverageModel | 1275.081804 |
| Alpha=0.0,SimpleExponentialSmoothing_Auto_Fit | 1275.081823 |
| Alpha=0.6477,Beta=0.0,DoubleExponentialSmoothing_Auto_Fit | 1275.081823 |
| Alpha=0.08621,Beta=1.3722,Gamma=0.4763,TrippleExponentialSmoothing_Auto_Fit | 1275.081823 |
| 6pointTrailingMovingAverage | 1283.927428 |
| 9pointTrailingMovingAverage | 1346.278315 |
| p=2,d=1,q=2,Auto_ARIMA | 1374.484105 |
| Alpha=0.1,SimpleExponentialSmoothing | 1375.393398 |
| RegressionOnTime | 1389.135175 |
| (1,1,1),Manual_ARIMA | 1461.668081 |
| Alpha=0.1,Beta=0.1,DoubleExponentialSmoothing | 1778.564670 |
| NaiveModel | 3864.279352 |

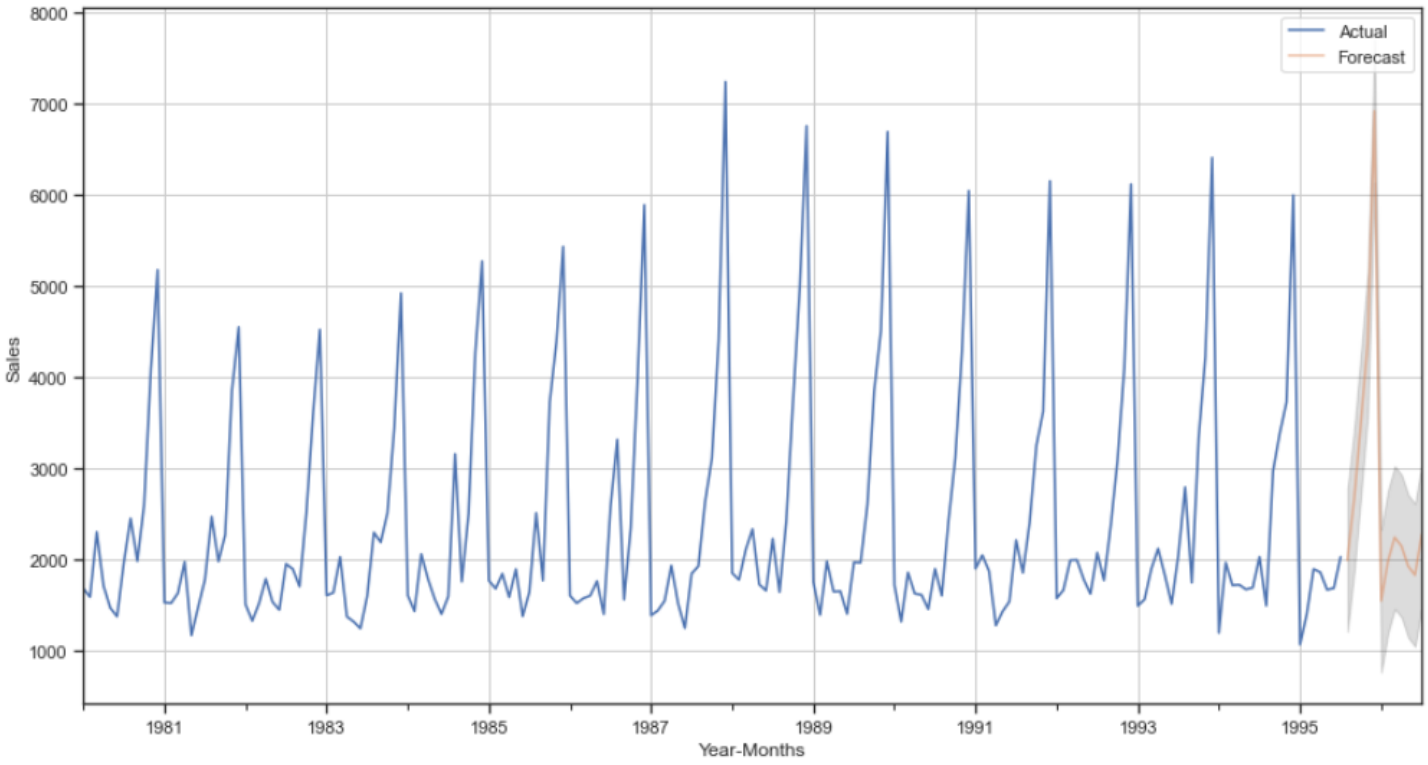
## 9 BASED ON THE MODEL-BUILDING EXERCISE, BUILD THE MOST OPTIMUM MODEL(S) ON THE COMPLETE DATA AND PREDICT 12 MONTHS INTO THE FUTURE WITH APPROPRIATE CONFIDENCE INTERVALS/BANDS.

Based on the above comparison of all the various models that we had built, we can conclude that the triple exponential smoothing or the Holts-Winter model is giving us the lowest RMSE, hence it would be the most optimum model and would give the most accurate prediction at alpha = 0.4 , beta = 0.1 and gamma = 0.3.

Hence we went ahead and built the most optimal model and made prediction 1 year into the future i.e. from 01-08-1995 to 01-07-1996. Below were the sales predictions made by this best optimum model.

| | Predictions |
|---|---|
| 1995-08-01 | 1994.197570 |
| 1995-09-01 | 2661.748348 |
| 1995-10-01 | 3497.632630 |
| 1995-11-01 | 4370.320057 |
| 1995-12-01 | 6919.218345 |
| 1996-01-01 | 1548.592493 |
| 1996-02-01 | 1982.230450 |
| 1996-03-01 | 2244.683691 |
| 1996-04-01 | 2149.422207 |
| 1996-05-01 | 1928.567787 |
| 1996-06-01 | 1831.097911 |
| 1996-07-01 | 2277.133204 |

We also plotted the sales prediction on the graph along with the confidence intervals. PFB the graph.




Predictions, 1 year into the future are shown in orange color, while the confidence interval has been shown in grey color.

## 10. COMMENT ON THE MODEL THUS BUILT AND REPORT YOUR FINDINGS AND SUGGEST THE MEASURES THAT THE COMPANY SHOULD BE TAKING FOR FUTURE SALES.

Looking at the most optimal model and its predictions, we can safely say the sales for Sparkling wine for the company will at the least be the same as last year, if not more. Peak sales for next year might be a little higher than this year.

Sparkling wine seems to be a very popular wine amongst the users and has hence enjoyed good amount of sales over the last few years with only a very marginal decline in sales.

Although the wine reached its peak popularity in 1988 - 1989 period, its popularity has not diminished much over the years.

Sparkling wine seems to highly impacted by the seasonality with the sales being very tipid in the first half of the year, and picking up only from August till December.

It would be recommended for the company to run some campaigns in the first half of the year where the sales are very slow.

Discounting for the weather conditions, middle of the year from March to July would be an ideal time to run campaigns for this type of wine. Being aggressive in this part of the year will drive the overall sales up.

May be combining promotions where a wine like "Rose wine" wine which is not so popular amongst the customers, is paired along with Sparkling wine under some special offer, will drive customers to try the under performing wine as well, which might also boost its sale. Which would be a win win for the company.

# Thank You !!