SUMMER PROJECT

TIME SERIES

INDIA's UPI TRANSECTION TIME SERIES FORECASTING

Guide: Prof. Ashish Das, Department of Mathematics, IIT Bombay, India

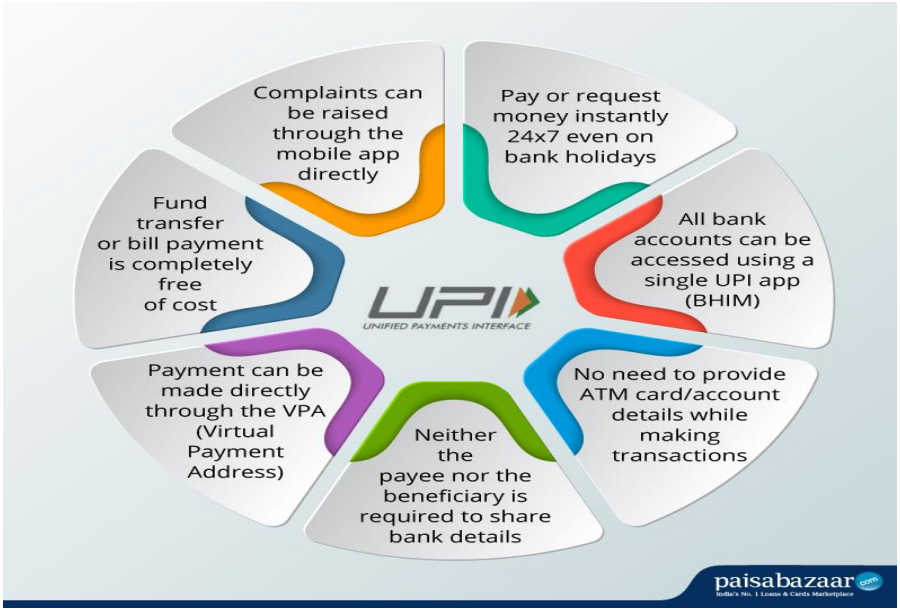
Prepared by:

Shiv Yadav (M.Sc. ASI, Roll no: 22N0064)

Sudipta Das (M.Sc. ASI, Roll no: 22N0075)

Gautam Patel (M.Sc. ASI, Roll no: 22N0067)

Table of Contents



1. Introduction:

In an era where technology permeates every aspect of our lives, the financial landscape of nations is witnessing an unprecedented transformation. In India, the advent of the Unified Payments Interface (UPI) has revolutionized the way transactions are conducted, propelling the nation towards a cashless economy. As data enthusiasts and aspiring analysts, we, Shiv Yadav, Sudipta Das and Gautam Patel, embarked on a journey to explore and forecast India's UPI transaction time series. The unparalleled growth of digital payment systems in India has been largely attributed to the visionary introduction of the Unified Payments Interface by the National Payments Corporation of India (NPCI). UPI has facilitated a seamless and secure payment ecosystem, democratizing financial access and empowering millions of Indians, from urban centres to rural hinterlands. The rapid adoption of UPI and its increasing significance in the economy have inspired us to delve into the troves of data and unlock valuable insights that can drive informed decision-making.

2. Objective:

This project aims to leverage data analysis and time series forecasting methodologies to gain comprehensive insights into the intricate patterns of UPI transactions. Through rigorous examination of historical transaction data, we seek to identify underlying trends, seasonality, and potential external influences on transaction volumes. Armed with this knowledge, we endeavour to create accurate and meaningful forecasts for future UPI transactions, enabling stakeholders to make informed decisions and embrace the transformative potential of digital payments.

Some Additional objectives of this project include:

- **In-Depth Analysis of Historical Transaction Data**
- **Identification of Trend and Seasonal Patterns**
- **Development of Robust Forecasting Models**
- **Evaluation of Forecast Accuracy**
- **Implications for India's Financial Ecosystem**

Through a rigorous and data-driven approach, we aspire to contribute to the ongoing dialogue surrounding the transformative role of digital payments in India's socio-economic development. By exploring the depths of UPI transaction data, we aim to present valuable insights that can guide the nation towards a more inclusive, efficient, and secure financial future.

3. Dataset:

The dataset used for this time series forecasting project is a time series dataset which contains about 3 years (Jun'20-May'23) of the daily volume and value of UPI transection and the size of dataset is 1076 rows × 3 columns, (one column for 'date', one for UPI volume (in crores) and the other one for the UPI Value (in crores).

Source of this dataset: The dataset was provided by our Guide Prof. Dr Ashish Das, Dept. of Mathematics, IIT Bombay.

The glimpse of this dataset looks like:

```
In [8]: df_copy
```

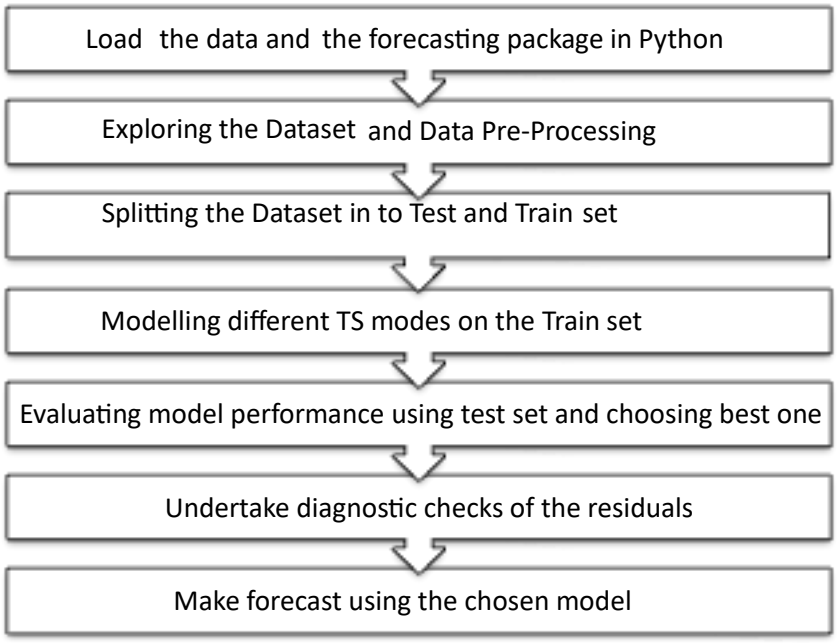
Out[8]:

	DATE	Vol	Val
0	2020-06-01	4.7697	10413.11
1	2020-06-02	4.7678	9951.30
2	2020-06-03	4.5626	9622.38
3	2020-06-04	4.6305	9639.50
4	2020-06-05	4.6479	9539.52
...	...	...	...
1071	2023-05-08	31.0363	54789.08
1072	2023-05-09	31.1973	53279.57
1073	2023-05-10	31.3276	52894.17
1074	2023-05-11	30.8546	50019.11
1075	2023-05-12	32.3425	50084.02

1076 rows × 3 columns

4. Methodology

The methodology which we are going to follows to carry this time series project is clearly depicted in the following flow-chart as follows:



5. Exploring the Dataset and Data-Preprocessing:

In this section, we are going to examine my dataset like is there any missing values and if there are any missing values then we are going to impute those missing values by any of the missing value imputation method and, I am going to check for any outlying values (outliers) and if there are any outliers then handle them.

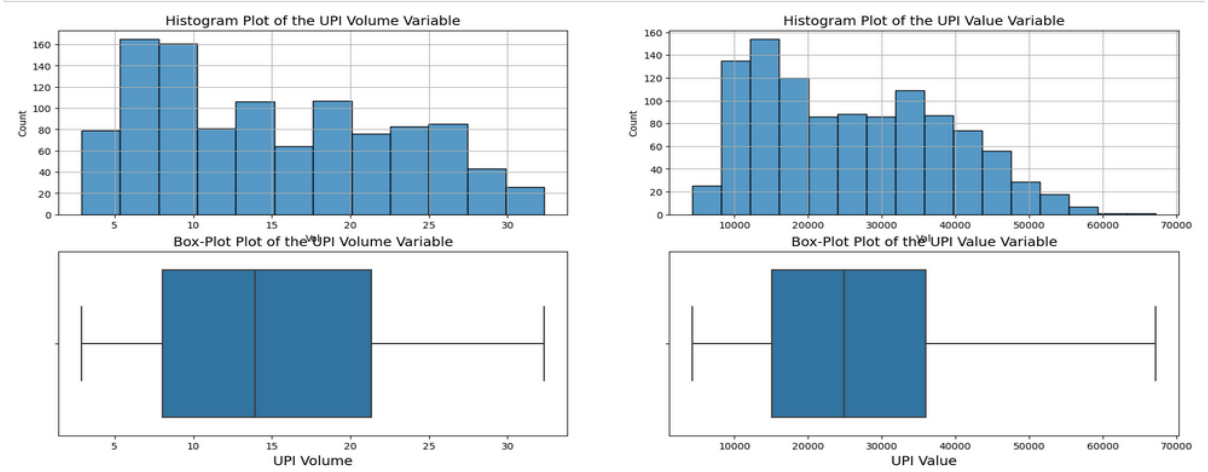
5.1. Checking and Handling the Missing values:

```
In [12]: df_copy.isnull().sum()

Out[12]: Vol    0
         Val    0
         dtype: int64
```

From above output of the python code, we have seen that fortunately we do not have **any missing values** in our time series data; therefore, we can proceed further in our project as follows:

5.2. Histogram of UPI Volume and UPI Value and Outlier detection Using the Box-Plot:

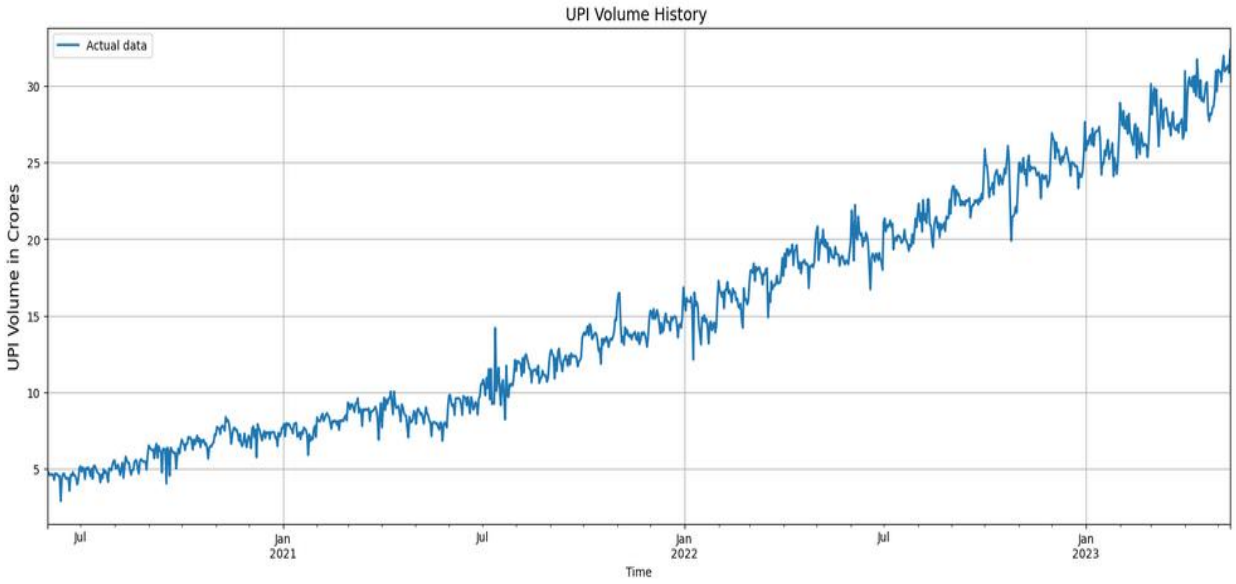


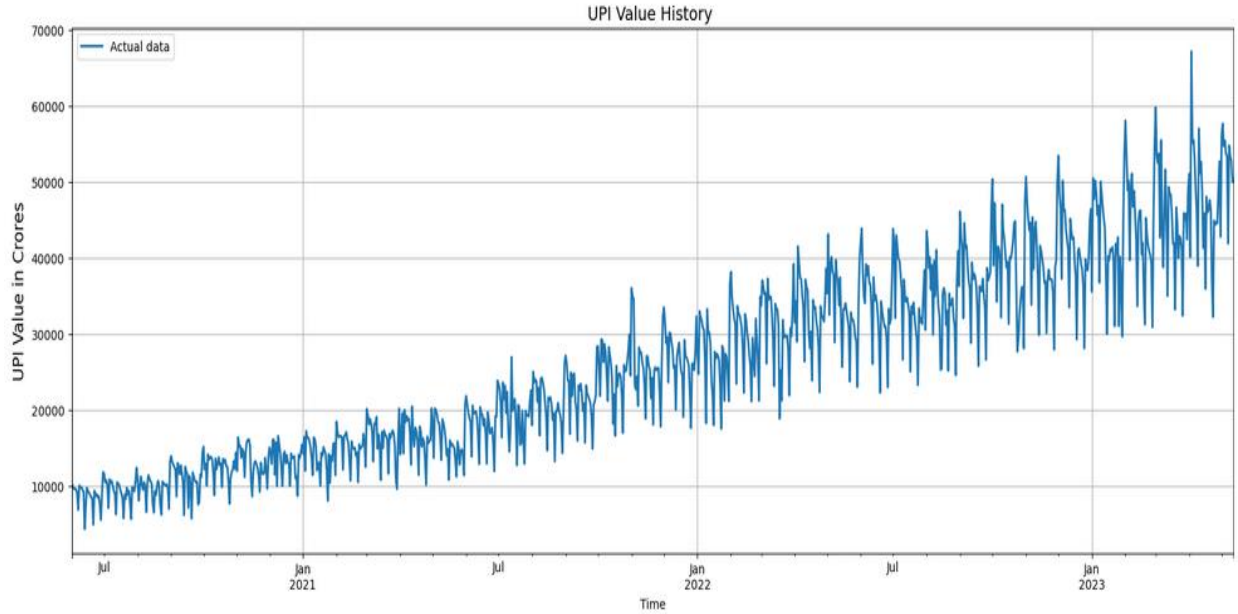
Observation:

- From the above histogram plots of the UPI Volume and the UPI Value variable, we observe that both variables have a slightly positively skewed distribution as the values are higher at the left side and getting lower at the right.
- Now from the Box-plots, we can easily see that none of the values lies beyond the upper and lower whisker, thus we can conclude that our UPI Volume and UPI Value variables have no any outliers.

5.3. Plotting the Time Series Data

Now let us look of our time series data over whole period to get some insights about our time series data:

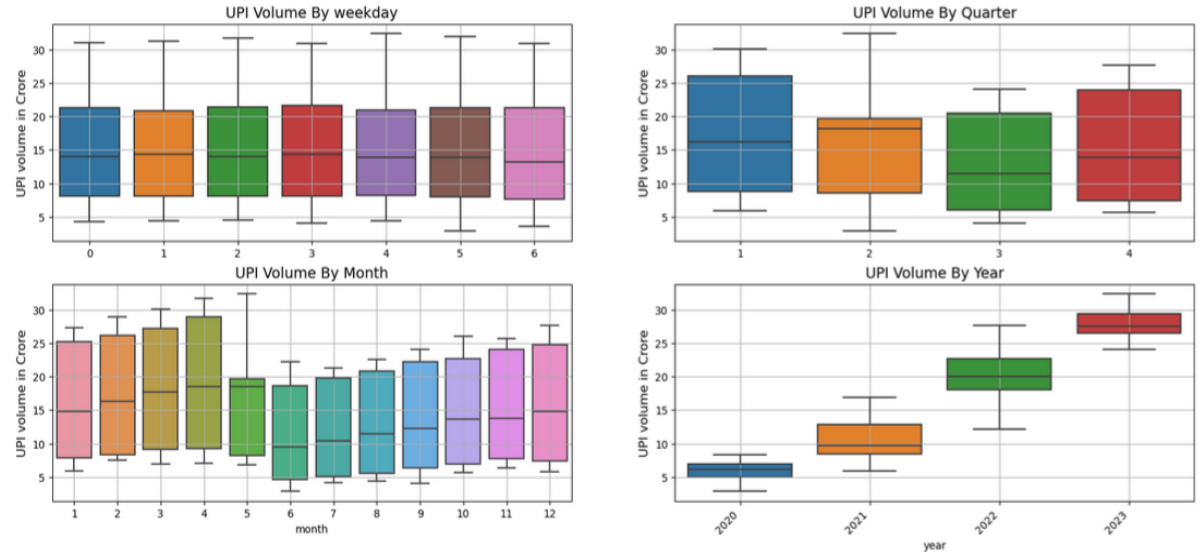


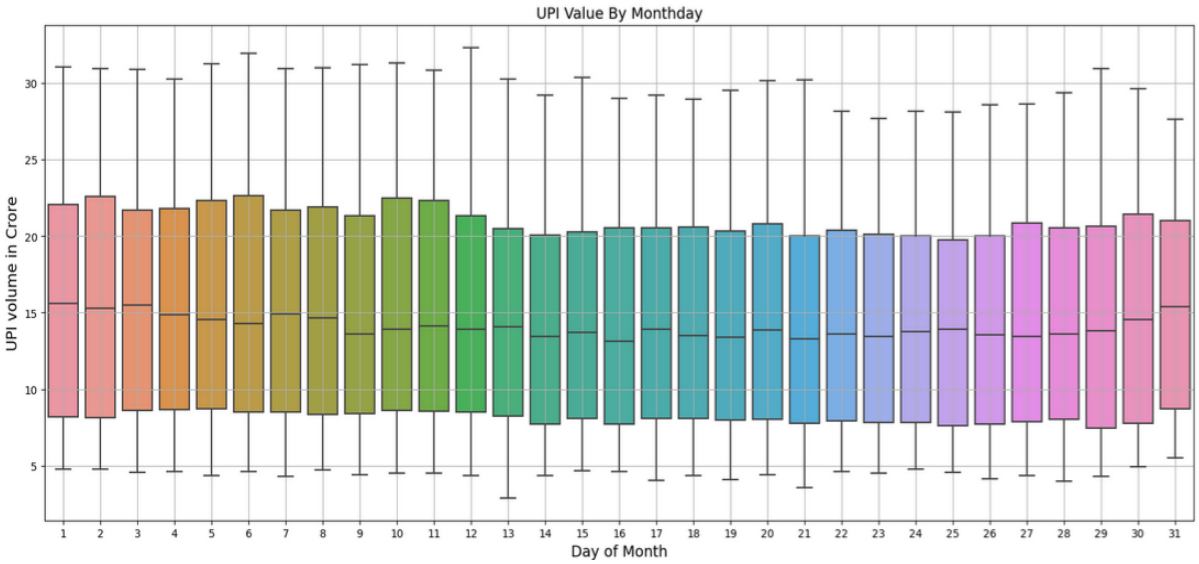


Comment: On the first look on our both time series, we can observe that our time series have some kind of increasing trend because as the time has gone our UPI Volume and UPI Value also get increased over the time.

5.4. Visualising the Relationship between our Target variables UPI Volume & UPI Value & and Various Time Features

- **Relation between UPI volume and Various Time features**

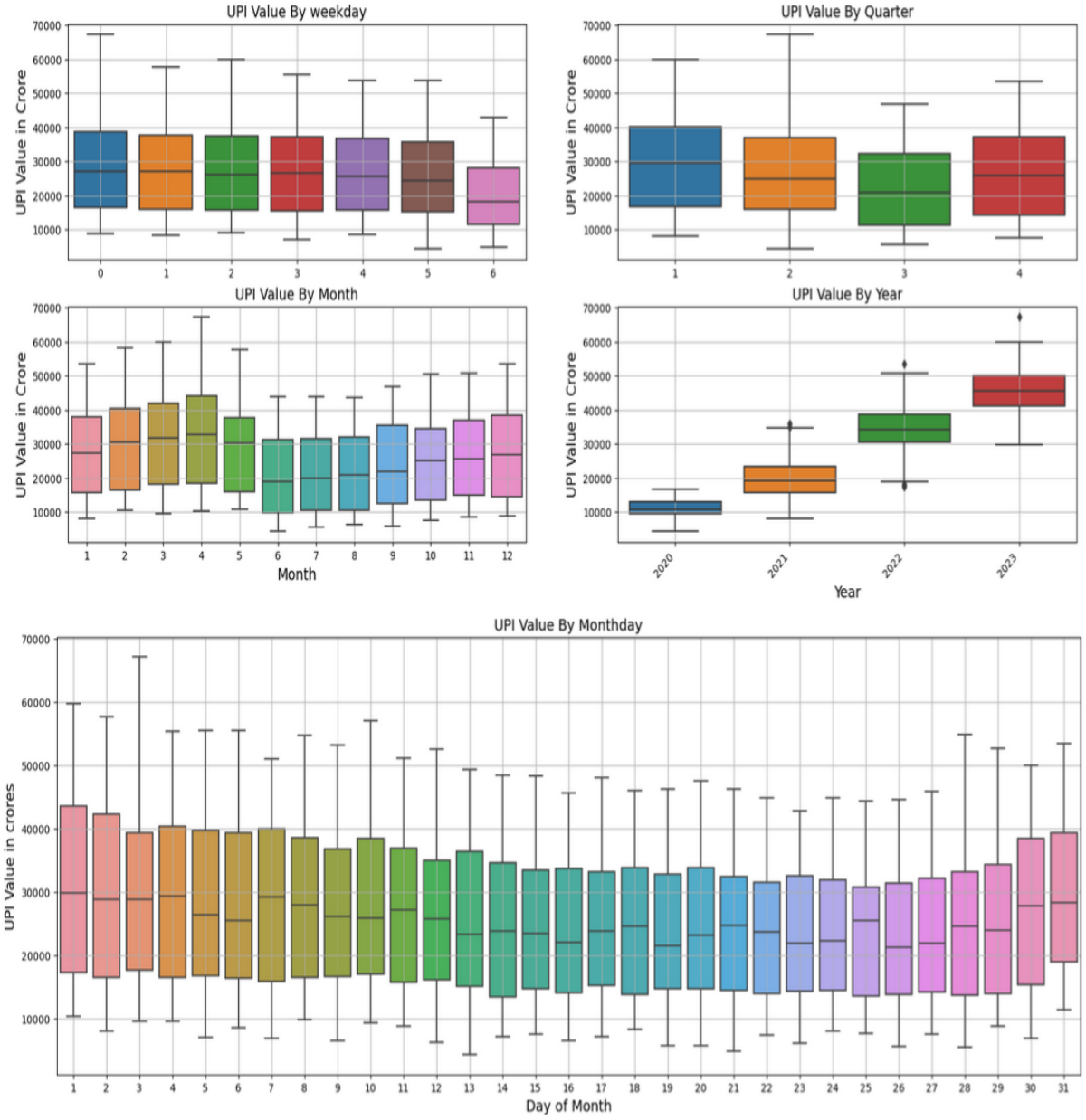




Observations: From the above plots showing the relationships between the UPI Volume and various time features we have following observations:

- **UPI Volume** is approximately independent of the day of week
- **UPI Volume** is slightly higher in the first and the last quarters while in the middle quarters it has lower values
- **UPI Volume** are getting increasing up to the April, then there is sudden drop in volume in the month of May and it is lowest in June, and after June it again starts increasing till December.
- **UPI Volume** has increased over the years, this may suggest of increasing trend
- **UPI volume** remains approximately unaffected by the day of a month except some of the small fluctuations in the middle days of a month

- **Relation between UPI Value and Various Time features**



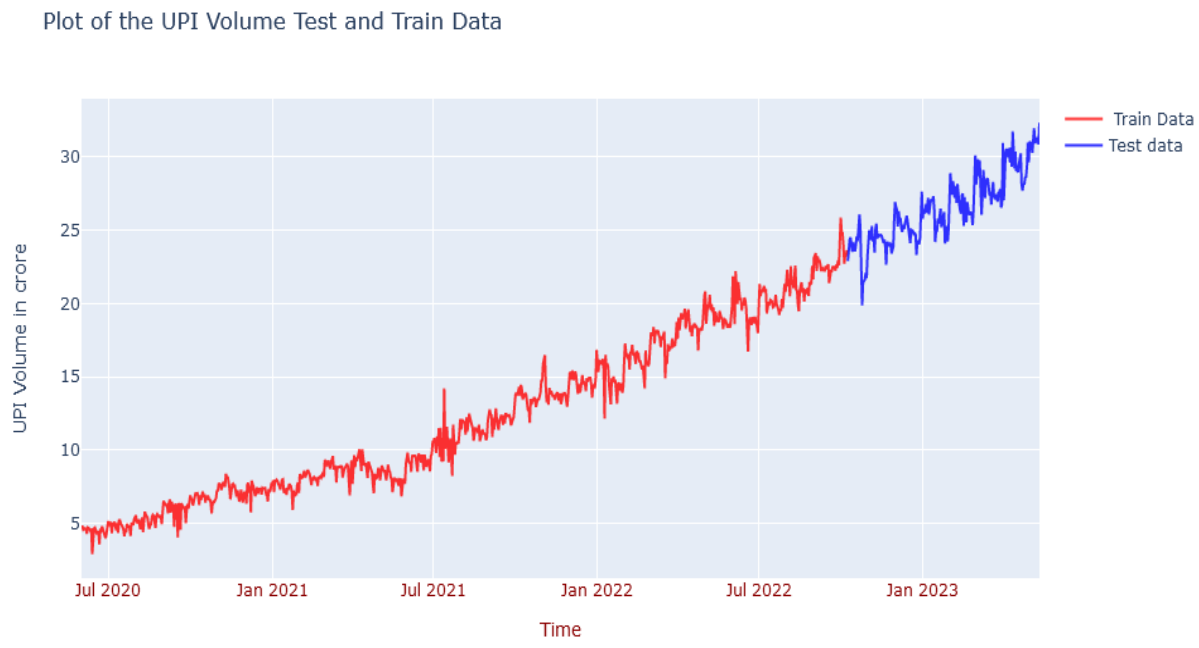
Observations: From the above plots showing the relationships between the UPI Value and various time features we have following observations:

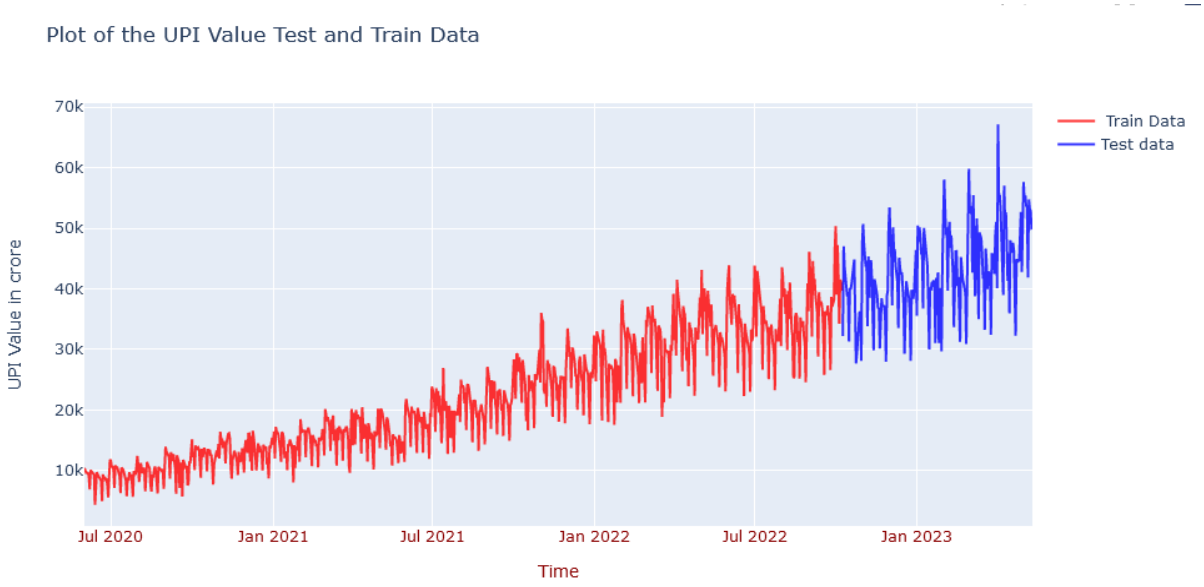
- **UPI Value** has a slightly decreasing up to Saturday and it suddenly drops on Sunday
- **UPI Value** is higher in the first and the last quarters while in the middle quarters it has lower values
- **UPI Value** are getting increasing up to the April, then there is sudden drop in volume in the month of May and it is lowest in June, and after June it again starts increasing till December.
- **UPI Value** has increased over the years, this may suggest of increasing trend
- **UPI Value** starts with a higher value in the starting days of month and gets decreased in the middle days of a month and it again starts increasing in the end days of a month, **this suggests a seasonal pattern in the UPI value series.**

6. Splitting the Dataset in to Train and Test Set:

In this section, we are going to split our whole time series data in to two parts one is called the Train set (The set of data used for model building/ training the models) and the other one is called the Test set (The set of the data used for model validation and evaluating performances metrics of the trained models). Usually in splitting the data set in to Train and Test set we take the 80% of the whole data as Train set and the remaining 20% as a Test set, so I have also followed this thumb rule of splitting the data set into **Train set (80%)** and the **Test set (20%)** for my analysis.

Now, let us look our Test and Train sets as follows:





Note: From now and onwards we are going to work with only the train set of the time series data to analyse our Time series.

7. Decomposition of the time series Data:

Decomposition: Decomposing a time series data refers to the process of breaking down the time series into its constituent components, namely the trend, the seasonality, and the Residual (also called noise). Time series decomposition is a common technique used in analysing and understanding time series data.

- **7.1 Naive Decomposition (Classical Decomposition) "*The Additive Model*"**
  The classical decomposition of a time series refers to a widely used method for decomposing a time series into its trend, seasonality, and residual components. It is also known as the additive decomposition method. The classical decomposition assumes that the time series can be expressed as the sum of following three components:

  1. $\boldsymbol{Trend(T)}$
  2. $\boldsymbol{Seasonality(S)}$
  3. $\boldsymbol{Residuals(R)}$

Thus, the *Additive Model* is given by:

$$\boldsymbol{X_t = T + S + R}$$

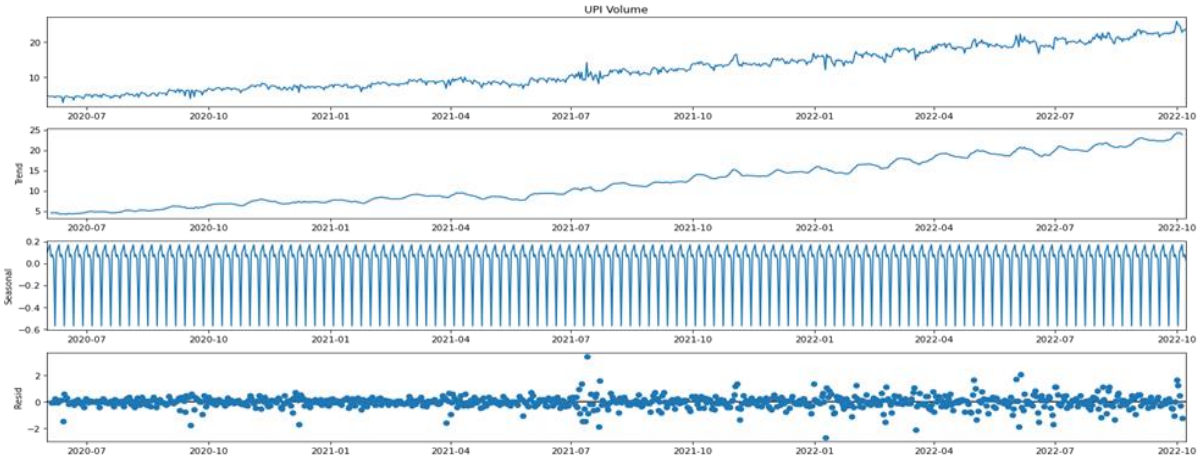
- **7.2 Multiplicative Decomposition of the Time Series Data:**
  This method of decomposition assumes that the time series can be expressed as the product of following three components:

  1. $\boldsymbol{Trend(T)}$
  2. $\boldsymbol{Seasonality(S)}$
  3. $\boldsymbol{Residuals(R)}$

And the *Multiplicative Model* is given by:

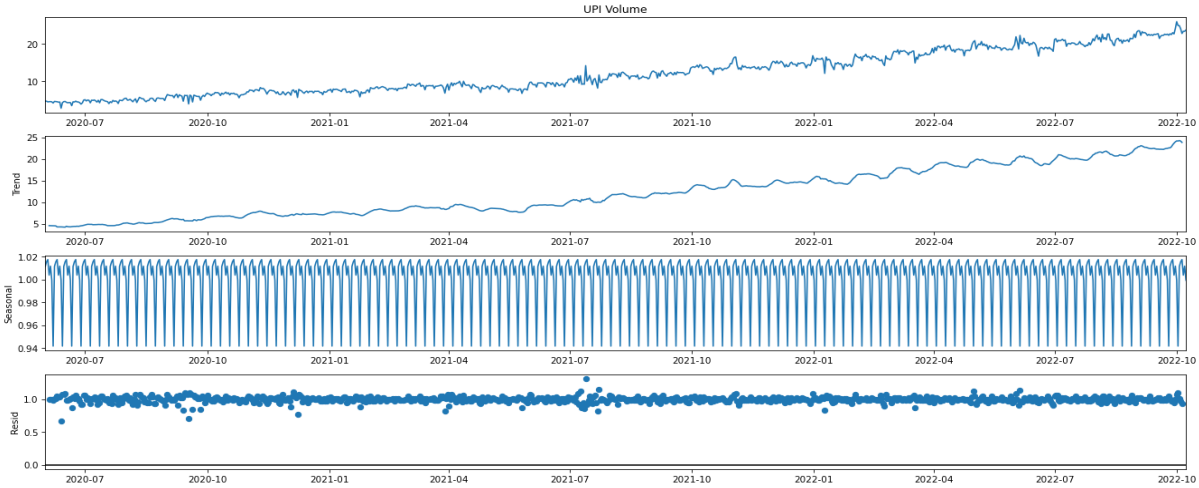
$$X_t = T * S * R$$

- **(a) Additive Decomposition for UPI Volume Data:**



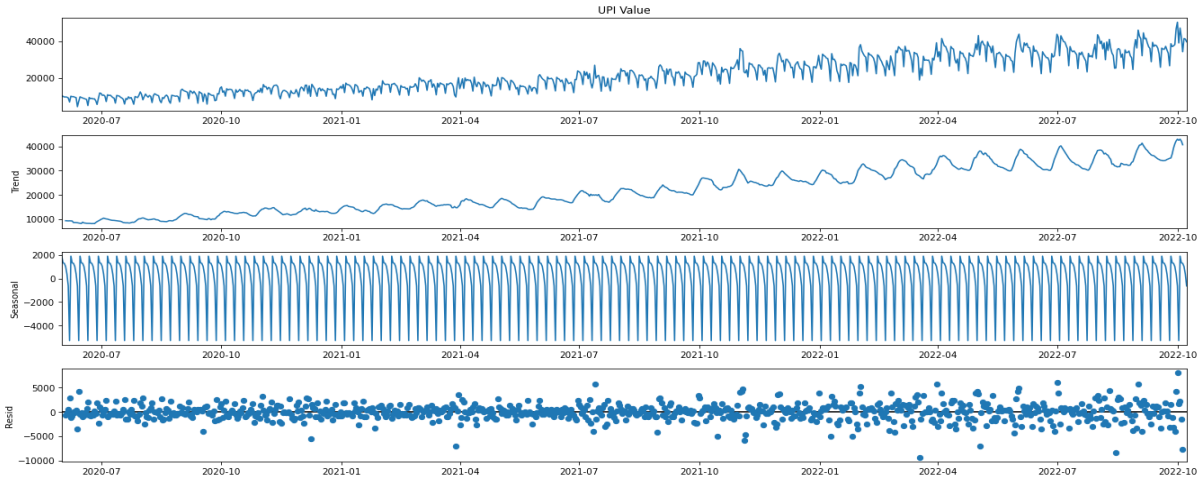
Comment: From the above plots it can be concluded that there is an **upward** trend and a seasonal component with very small period (maybe **7 days**) present in the **UPI Volume** data also from the residual plot it can be interpreted that before **July 2021** residuals are evenly distributed around the Zero line but after that residuals have more variation around the Zero line.

- **(b) Multiplicative Decomposition for UPI Volume Data:**



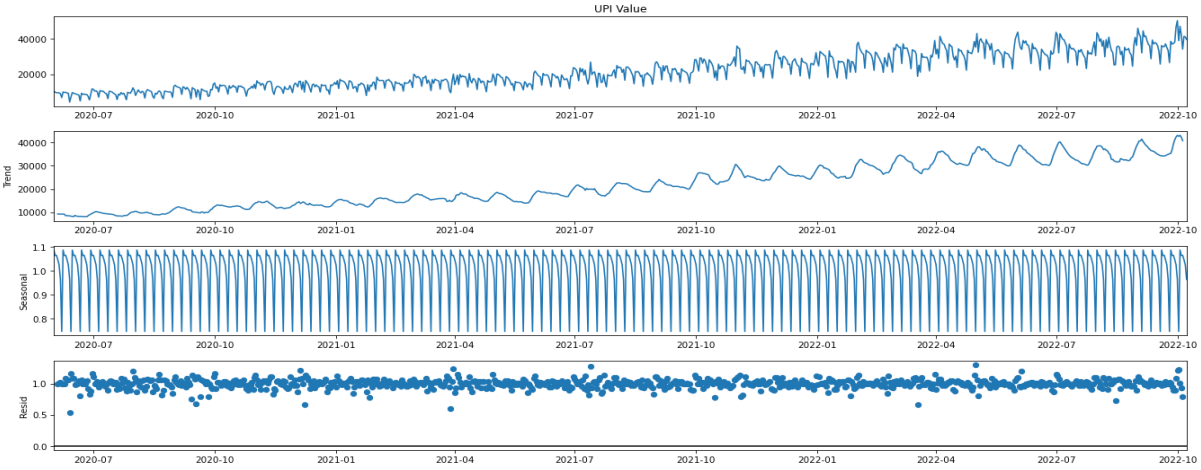
Comment: From the above multiplicative decomposition, the same can be concluded, which we have concluded from the Additive decomposition except the residuals terms, in the additive decomposition the residuals are centred around the Zero line(which can so, because in Additive decomposition, $\boldsymbol{Residual = Actual\ value - Predicted\ value}$) while in the multiplicative decomposition the residuals are centred around the $y = 1$line again which can so, because in multiplicative decomposition $\boldsymbol{Residual = \frac{Actual\ Value}{Predicted\ Value}}$

- **(c) Additive Decomposition for UPI Value Data:**



Comment: Again, from the above plots it can be concluded that there is an **upward** trend and a seasonal component with very small period (maybe **7 days**) present in the **UPI Value** data also from the residual plot it can be interpreted that before **July 2021** residuals are evenly distributed around the Zero line but after that residuals have more variation around the Zero line.

- **(d) Multiplicative Decomposition for UPI Value Data:**



Comment: Comparing the above multiplicative decomposition of the UPI Value data with its Additive decomposition, we can interpret that the residuals in **Additive** model are **more fluctuating** around the **Zero line** while in the **Multiplicative** model the residuals have **lesser fluctuation** around the $\boldsymbol{y = 1}$ line, this indicates that Multiplicative model is quite **preferable** over the additive model for the UPI value data.

8. Modelling our Time Series:

In this section we are going to build various models for our time series data as follows:

8.1. Building Some Base Line Models

- **Polynomial Regression:** In this section, we are going to use polynomial regression to train our model for the series as follows:
  - **Fitting a 2nd degree to the UPI volume data**

Model summary:

OLS Regression Results

Dep. Variable:	Vol	R-squared:	0.975
Model:	OLS	Adj. R-squared:	0.975
Method:	Least Squares	F-statistic:	1.683e+04
Date:	Sat, 29 Jul 2023	Prob (F-statistic):	0.00
Time:	13:55:54	Log-Likelihood:	-1109.9
No. Observations:	860	AIC:	2226.
Df Residuals:	857	BIC:	2240.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	4.6262	0.090	51.205	0.000	4.449	4.804
x1	0.0090	0.000	18.533	0.000	0.008	0.010
x2	1.506e-05	5.45e-07	27.632	0.000	1.4e-05	1.61e-05

Omnibus:	16.256	Durbin-Watson:	0.729
Prob(Omnibus):	0.000	Jarque-Bera (JB):	28.707
Skew:	-0.077	Prob(JB):	5.84e-07
Kurtosis:	3.882	Cond. No.	9.96e+05

Visualising the Result of Polynomial fitting on the UPI Volume data

- **Fitting a 2nd degree to the UPI Value data**

Model Summary:

```
In [130]: ## Fitting the Regression model
          reg_model_val = data_stat_model.OLS(y_train, X_train_lr).fit()
          print(reg_model_val.summary())
```

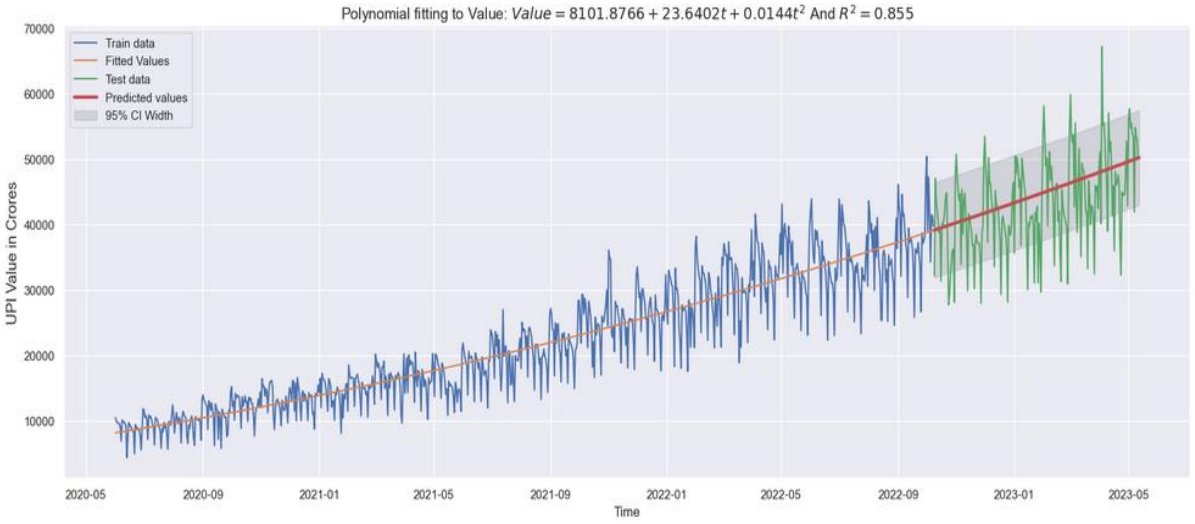
OLS Regression Results

Dep. Variable:	y	R-squared:	0.855
Model:	OLS	Adj. R-squared:	0.855
Method:	Least Squares	F-statistic:	2525.
Date:	Sat, 29 Jul 2023	Prob (F-statistic):	0.00
Time:	14:39:20	Log-Likelihood:	-8286.0
No. Observations:	860	AIC:	1.658e+04
Df Residuals:	857	BIC:	1.659e+04
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	8101.8766	379.979	21.322	0.000	7356.078	8847.675
time	23.6402	2.038	11.598	0.000	19.640	27.641
time_sq	0.0144	0.002	6.278	0.000	0.010	0.019

Omnibus:	29.897	Durbin-Watson:	1.202
Prob(Omnibus):	0.000	Jarque-Bera (JB):	55.501
Skew:	-0.233	Prob(JB):	8.87e-13
Kurtosis:	4.154	Cond. No.	9.96e+05

Visualising the Result of Polynomial fitting on the UPI Value data:



8.2. Modelling with some Classical Time Series Models

8.2.1. Auto Regressive Integrated Moving Average (ARIMA) Method

- Auto Regressive model is also a multiple linear regression model in which the independent variables are time-lagged versions of the dependent variable.

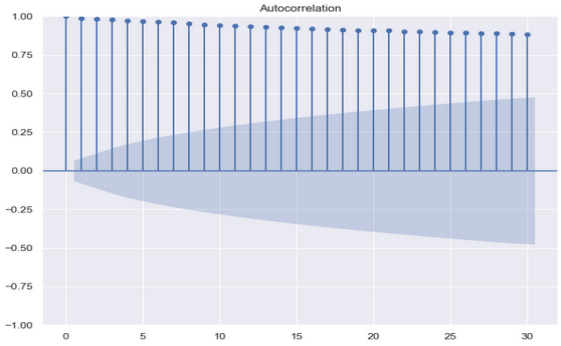
$y_t = \mu + \varphi_1 y_{(t-1)} + \cdots + \varphi_p y_{(t-p)} + \epsilon_t$; where $\epsilon_t \sim WN(0, \sigma^2)$ and $|\varphi_i| < 1$ for all $i = 1,2,..,p$

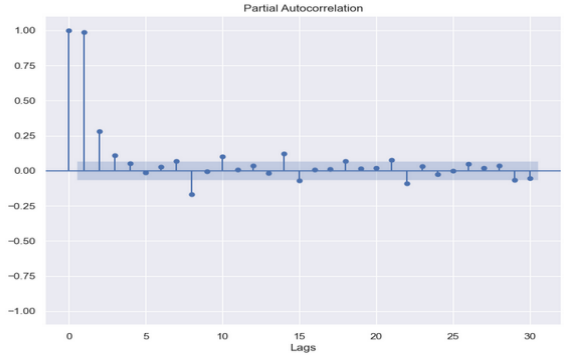
- y_t is the forecasted value and $y_{(t-1)}, y_{(t-2)}, \ldots, y_{(t-p)}$ are the historical values and p is the number of lags
- In order to build the model, we need to identify the following:
  - Seasonality
  - Trend

  To select the p : the number of lags

➢ Before going the Model Building first let us identify the lags by visualizing the plots of **Autocorrelation function (ACF**) and the **Partial Autocorrelation Function (PACF)** as:

- **ACF And PACF of UPI volume time series**

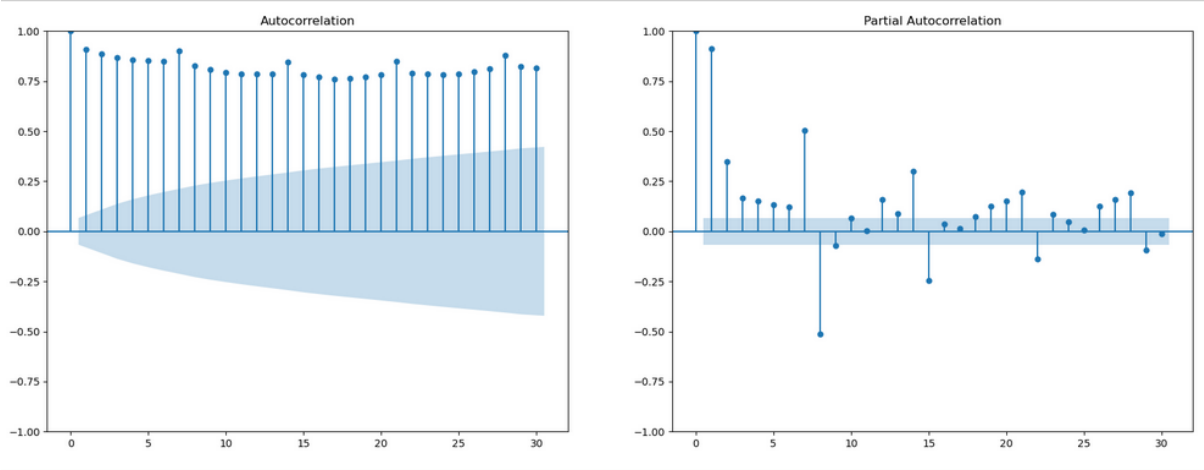




Interpretation:

- **ACF**: From the above ACF plot, it can be seen that the ACF is **very slowly decaying**, that means future values of the series are correlated / heavily affected by past values. That means the mean will change over time, which means that our series is **non-stationary**.
- From the PACF plot, it can be interpreted that **lag 1** correlation is very high and the **lag 2** correlation is moderately significant, **lag 3** correlation is slightly significant, while the next 4 lags are not significant but after that, i.e., the **lag 8** is the last significant correlation and after that all the correlation are statistically insignificant as they lie inside the 95% confidence Interval width.

- **ACF And PACF of UPI Value time series**



Interpretation:

- **ACF**: Again, from the above ACF plot for the UPI value, it can be seen that the ACF is **very slowly decaying**, that means future values of the series are correlated / heavily affected by past values. That means the mean will change over time, which means that our series is **non-stationary**.
- From the PACF plot, it can be interpreted that **lag 1** correlation is very high and the **lag 2** correlation is moderately significant, **lag 3** correlation is slightly significant, while the next 4 lags are not significant but after that, i.e., the **lag 8** is the last significant correlation and after that all the correlation are statistically insignificant as they lie inside the 95% confidence Interval width.

Observation: From the above discussion, it is seeming that both the time series UPI Volume and UPI Value are **NON-STATIONARY** in nature, but to be sure, we have performed a statistical test aka **The Augmented Dickey Fuller Test aka The Unit Root Test,** to identify whether a time series is stationary**.**

The Augmented Dickey Fuller Test for Stationarity:

Now to perform the ADF test for Stationarity, we define our Null and Alternative hypothesis as follows:

H_0: $\varphi(B)$ *contains a unit root implying the Non − Stationary Time series* Vs

H_1: $\varphi(B)$ *does NOT contains any unit root implying the Stationary Time series*

- **Decision Rule**: The decision rule of above test is given by:
  - Reject H_0 if the calculated value of test statistic is less than the tabulated value OR if the p-value <0.05
  - Otherwise, Do Not Reject H_0

Now, we have performed the ADF test in python for both the series and the results are as follows:

```
In [21]: ## AD-Fuller test for UPI Volume data
         import statsmodels.tsa.stattools as sts
         sts.adfuller(df_train_ar['Vol'])

Out[21]: (1.067322302967681,
          0.9949278521493288,
          20,
          839,
          {'1%': -3.438168140637663,
           '5%': -2.864991017121922,
           '10%': -2.5686076019468094},
          9316.518407183255)
```

```
In [20]: ## AD-Fuller test for UPI Value data
         import statsmodels.tsa.stattools as sts
         sts.adfuller(df_train_ar['Val'])

Out[20]: (-0.45767470253465653,
          0.9000147326771621,
          21,
          838,
          {'1%': -3.4381774989729816,
           '5%': -2.8649951426291,
           '10%': -2.568609799556849},
          15328.223783908474)
```

Observation: From the above snippet of Python code, showing the results of The ADF test on both data UPI Volume and UPI Value, and from **the p-value** for both the tests, it is clear that our series are **NON- STATIONARY** in nature, so to model our time series, we must have to make both the series stationary.

- **Method of Differencing to make a non-stationary series to a Stationary series**

The differencing transform is a very popular transform to make a time series stationary, or at least get rid of unit roots. The concept is simple: we transform the time series from the domain of observation to the domain of change in observations. The differencing transform subtracts subsequent observations from one another as follows:

$$z_t = y_t - y_{t-1}$$

Differencing helps us stabilize the mean of the time series and, with that, reduce or eliminate trend and seasonality. So, let us see how differencing can make a series stationary:

Now, to check whether the stationarity is achieved or not, we have performed the ADF test on the differenced series and the results are as follows:

```
In [23]: ## AD-Fuller test for UPI Volume difference data
         import statsmodels.tsa.stattools as sts
         sts.adfuller(diff_vol)

Out[23]: (-11.890522971376782,
          5.883463195393311e-22,
          21,
          837,
          {'1%': -3.4381868797392277,
           '5%': -2.864999278011895,
           '10%': -2.568612002429454},
          9299.031229750652)
```

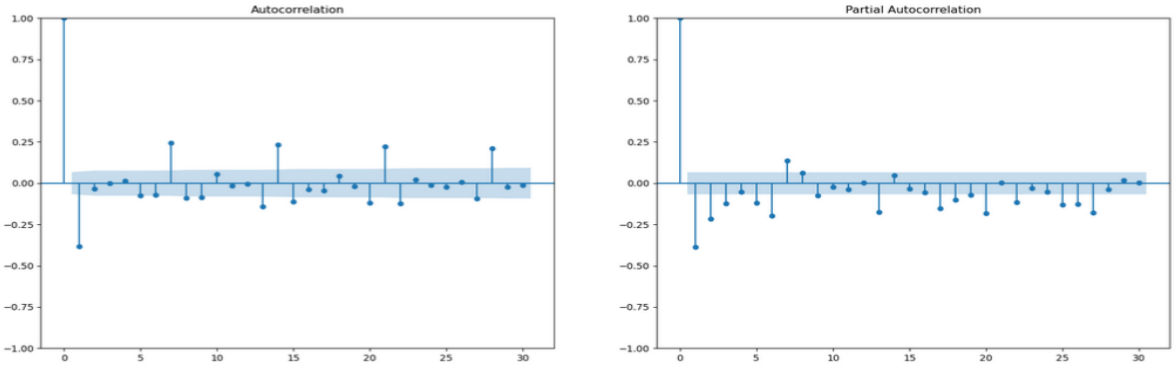
```
In [24]: ## AD-Fuller test for UPI Value difference data
         import statsmodels.tsa.stattools as sts
         sts.adfuller(diff_val)

Out[24]: (-11.979941171692266,
          3.7251526662613746e-22,
          21,
          837,
          {'1%': -3.4381868797392277,
           '5%': -2.864999278011895,
           '10%': -2.568612002429454},
          15301.818593565133)
```

Observation: Now, from the above results of the ADF test for the first order differenced series, we have seen that the p-values for both the test is too small, this implies that now, our series becomes stationary, thus we are ready to model the ARIMA time series model as follows:

But, before fitting the ARIMA model, first we have to find the appropriate values of the p, d and q of the ARIMA model, therefore to find the appropriate values of p, d, and q we have the ACF and PACF plot of the stationary (differenced) series, and the results are as follows:

- Identifying p, d, and q of ARIMA model for UPI Volume using ACF and PACF of the differenced UPI Volume series as follows:

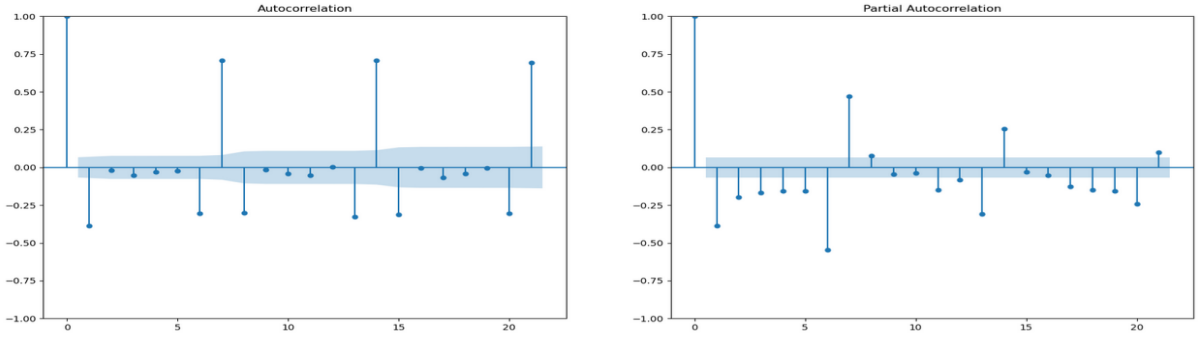


Observation: From the above ACF and PACF plots of the first order differenced series for UPI Volume, we have the following:

- p = 3 [as in the PACF plot after the 3rd lag correlation gets statistically insignificant]
- d =1 [as the first order differenced series id used to get the stationary series]
- q = 1[as there is a shutoff after lag 1 correlation in the ACF plot]

⇨ **Thus for UPI Volume we get the ARIMA model as: ARIMA(3,1,1)**

- Identifying p, d, and q of ARIMA model for UPI Value using ACF and PACF of the differenced UPI Volume series as follows:



Observation: From the above ACF and PACF plots of the first order differenced series for UPI Value, we have the following:

- p = 0 or 1
- d =1 [as the first order differenced series id used to get the stationary series]
- q = 1[as there is a shutoff after lag 1 correlation in the ACF plot]

⇨ **Thus for UPI Value we get the ARIMA model as: ARIMA(0,1,1) or ARIMA(1,1,1)**

Fitting the ARIMA model for our time series:

In this section we have fitted the ARIMA models for the UPI Volume and Value series with parameters obtained using the above discussion as follows:

- **Fitting the ARIMA (3,1,1) model for the UPI Volume Series:**

Model Summary:

```
### Model Building Method for UPI Volume
## ARIMA(3,1,1)
model_arima_vol = statsmodels.tsa.arima.ARIMA(diff_vol, order=(3,0, 1)).fit()
print(model_arima_vol.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                    Vol   No. Observations:                  859
Model:                 ARIMA(3, 0, 1)   Log Likelihood                -856.745
Date:                Thu, 03 Aug 2023   AIC                           1725.489
Time:                        14:21:25   BIC                           1754.024
Sample:                    06-02-2020   HQIC                          1736.414
                         - 10-08-2022
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0221      0.003      7.398      0.000       0.016       0.028
ar.L1          0.4103      0.028     14.493      0.000       0.355       0.466
ar.L2          0.1613      0.039      4.164      0.000       0.085       0.237
ar.L3          0.0982      0.033      2.932      0.003       0.033       0.164
ma.L1         -0.9575      0.017    -56.758      0.000      -0.991      -0.924
sigma2         0.4299      0.011     38.435      0.000       0.408       0.452
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):              1180.67
Prob(Q):                              0.91   Prob(JB):                         0.00
Heteroskedasticity (H):               2.97   Skew:                             0.33
Prob(H) (two-sided):                  0.00   Kurtosis:                         8.71
===================================================================================
```

- **Fitting the ARIMA (0,1,1) model for the UPI Value Series:**

Model Summary:

```
### Model Building Method for UPI Value (transforemd value)
## ARIMA(0,1,1)
model_arima_val = statsmodels.tsa.arima.ARIMA(diff_val_tr, order=(0,0, 1)).fit()
print(model_arima_val.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                    Val   No. Observations:                  859
Model:                 ARIMA(0, 0, 1)   Log Likelihood               -1100.710
Date:                Thu, 03 Aug 2023   AIC                           2207.419
Time:                        16:16:06   BIC                           2221.687
Sample:                    06-02-2020   HQIC                          2212.882
                         - 10-08-2022
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0004      0.009      0.046      0.963      -0.018       0.019
ma.L1         -0.6842      0.021    -33.374      0.000      -0.724      -0.644
sigma2         0.7589      0.030     25.198      0.000       0.700       0.818
===================================================================================
Ljung-Box (L1) (Q):                  10.63   Jarque-Bera (JB):                54.76
Prob(Q):                              0.00   Prob(JB):                         0.00
Heteroskedasticity (H):               5.34   Skew:                            -0.23
Prob(H) (two-sided):                  0.00   Kurtosis:                         4.15
-----------------------------------------------------------------------------------
```

- **Fitting The ARIMA (1,1,1) model for the UPI Value Series:**

Model Summary:

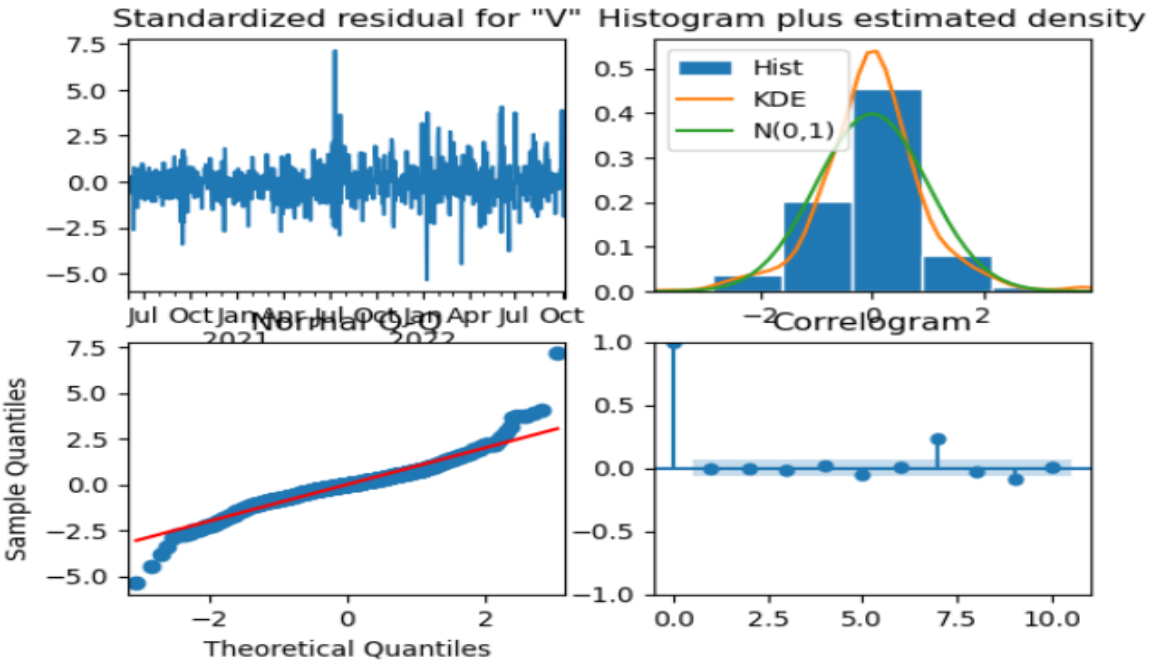
```
### Model Building Method for UPI Value (transforemd value)
## ARIMA(1,1,1)
model_arima_val = statsmodels.tsa.arima.ARIMA(diff_val_tr, order=(1,0, 1)).fit()
print(model_arima_val.summary())
```

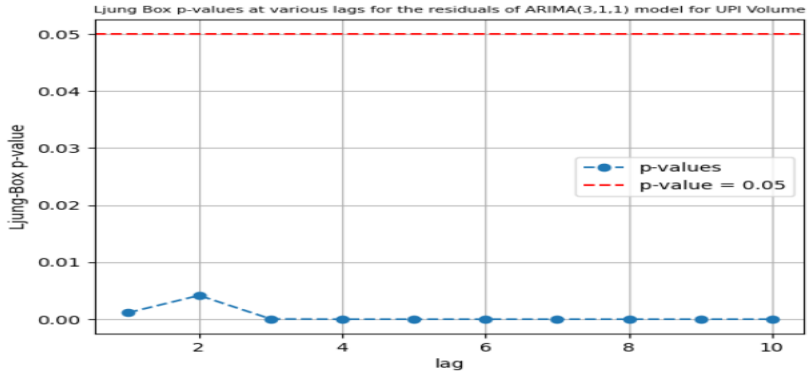
```
                               SARIMAX Results
==============================================================================
Dep. Variable:                    Val   No. Observations:                  859
Model:                 ARIMA(1, 0, 1)   Log Likelihood               -1068.415
Date:                Thu, 03 Aug 2023   AIC                           2144.831
Time:                        16:18:33   BIC                           2163.854
Sample:                    06-02-2020   HQIC                          2152.114
                         - 10-08-2022
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0002      0.001      0.207      0.836      -0.002       0.002
ar.L1          0.3815      0.027     13.943      0.000       0.328       0.435
ma.L1         -0.9814      0.007   -147.984      0.000      -0.994      -0.968
sigma2         0.7024      0.028     25.488      0.000       0.648       0.756
===================================================================================
Ljung-Box (L1) (Q):                   1.58   Jarque-Bera (JB):                41.49
Prob(Q):                              0.21   Prob(JB):                         0.00
Heteroskedasticity (H):               5.33   Skew:                             0.04
Prob(H) (two-sided):                  0.00   Kurtosis:                         4.07
===================================================================================
```

Model Diagnostics of ARIMA Models:

- **ARIMA(3,1,1) For UPI Volume**

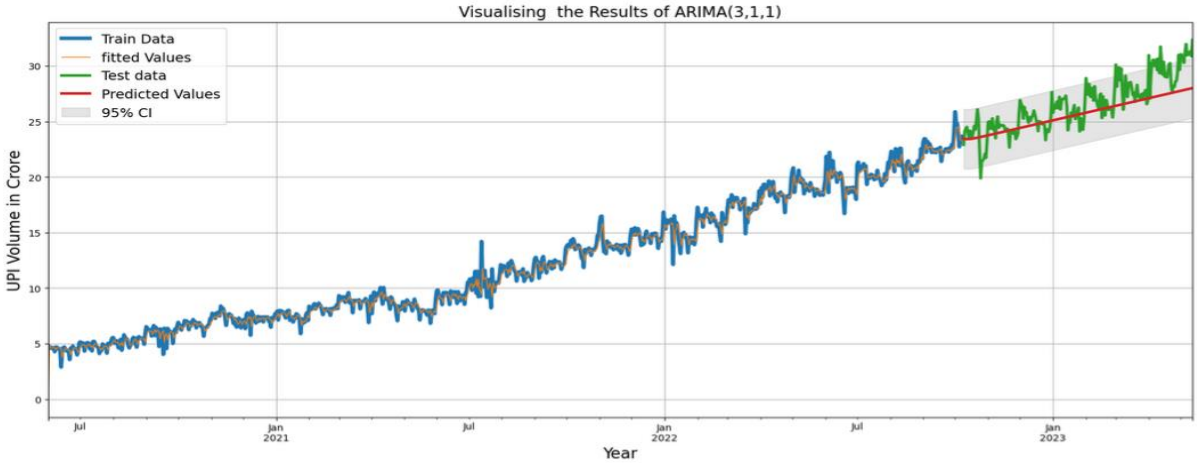
Residual Analysis Plot:





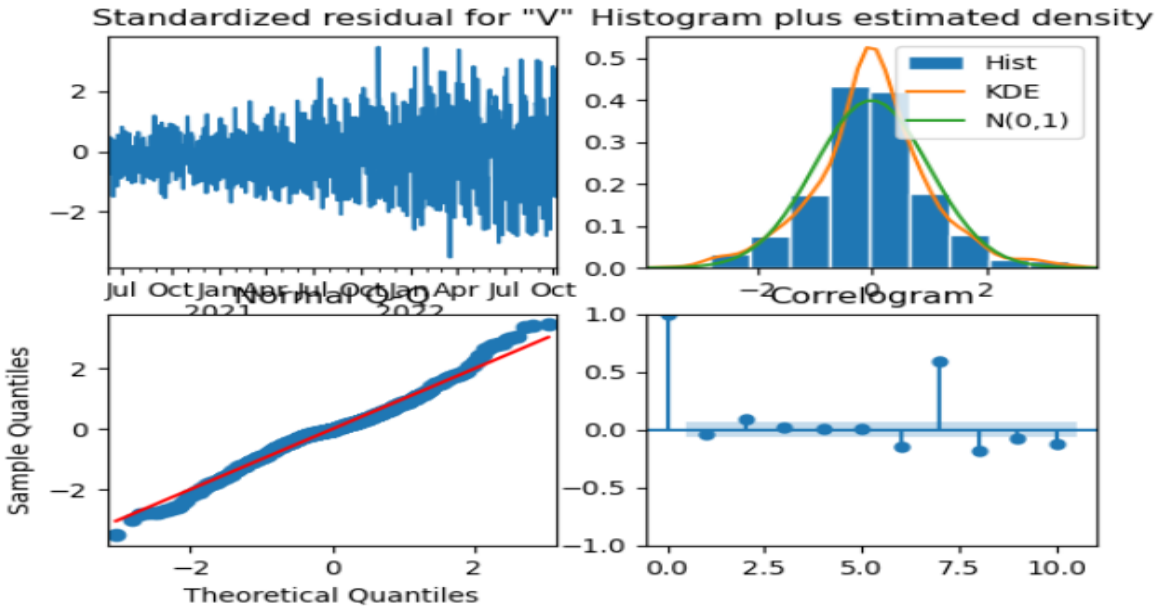
Comment: From the above Ljung-Box test p-values plot, it can be concluded that the residuals are serially correlated, thus they do not follow the white noise process.

Now, Since the assumption of TSA is not satisfied by the ARIMA(3,1,1) model for the UPI Volume Serie, but still let us visualise how, our model performance on train and test set:

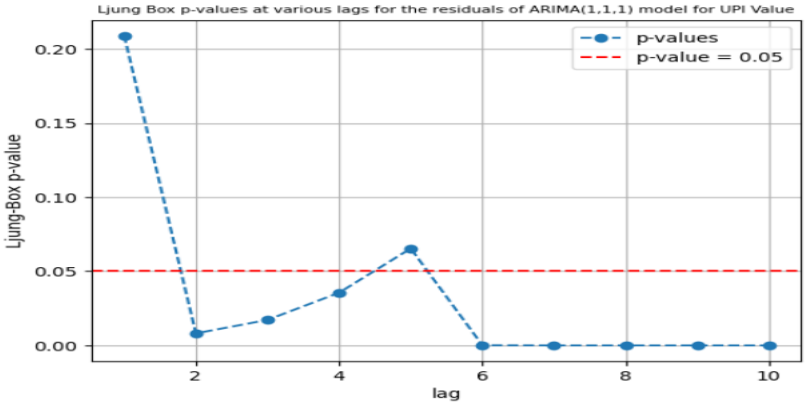


- **<u>ARIMA(1,1,1) for UPI Value:</u>**

<u>Residuals Analysis Plot:</u>

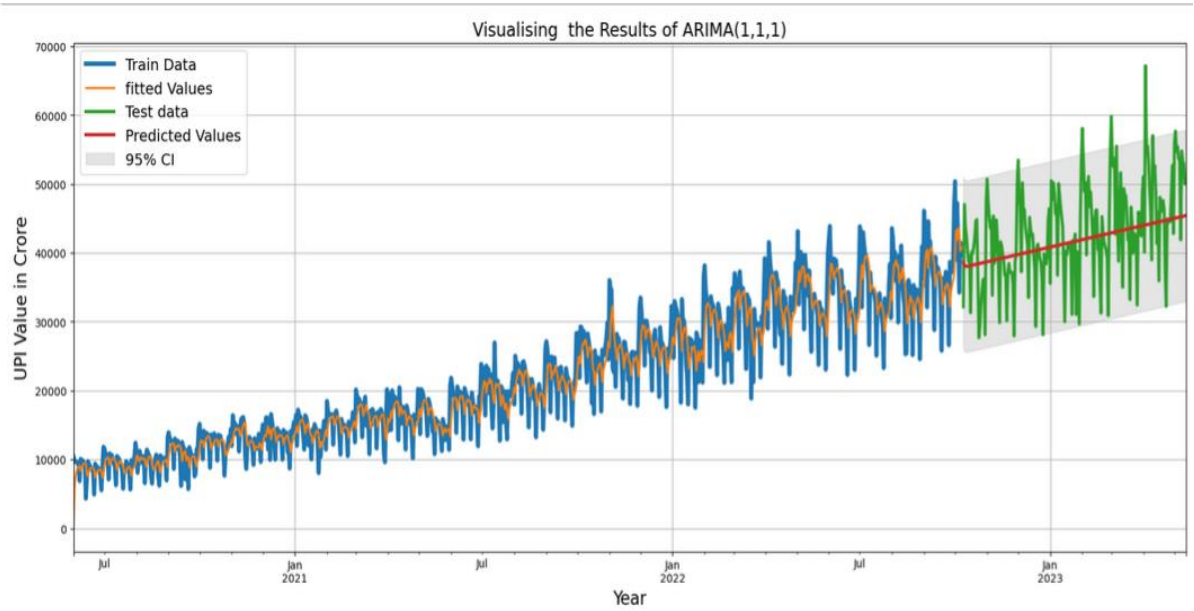


Ljung-Box Test p-values at different lags:



Comment: From the above Ljung-Box test p-values plot, it can be concluded that the residuals are serially correlated, and the plot of the standardised residuals over the time indicates that there is a pattern in the standardised residuals, violating the homoscedasticity assumption, thus they do not follow the white noise process.

Now, Since the assumption of TSA is not satisfied by the ARIMA(1,1,1) model for the UPI Value Serie, but still let us visualise how, our model performance on train and test set as follows:



NOTE: From the above analysis and Modelling, we have noticed that we are able to get the trend values in the series but there are still some variabilities in the prediction, that might be due to seasonality patterns, thus our next aim to introduce some models that considers the seasonality in data also, therefore Now, we are going to analyse and build the Seasonal Auto Regressive Integrated Moving Average SARIMA model as follows:

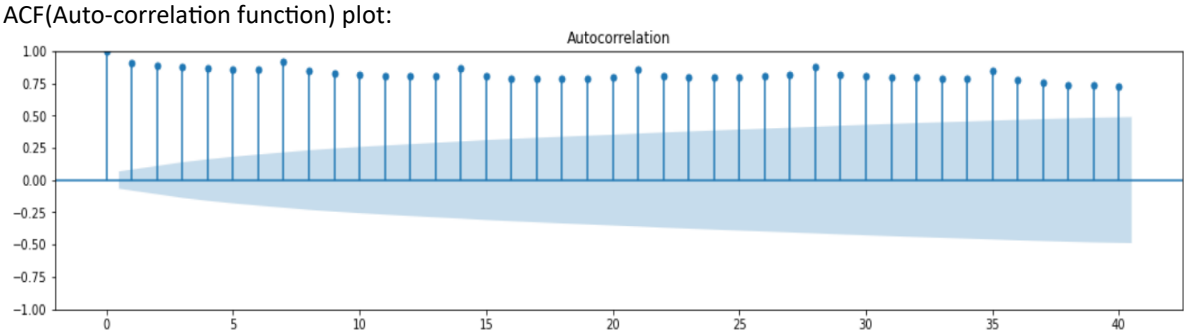
8.2.2. SARIMA Method

- **SEASONAL AUTOREGRESSIVE INTEGRATED MOVING AVERAGE (SARIMA) METHOD**

Stationarity:

In this part we are going to check stationarity of our data.

ACF(Auto-correlation function) plot:

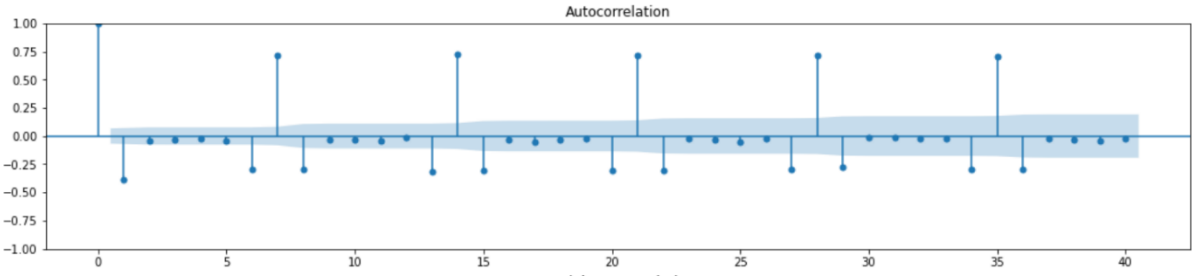


From the above plot it is clear that our data is non-stationary as several lags of ACF are significant. We should verify it using ADF test

```
ADF Statistic: -0.901215
p-value: 0.787572
Critical Values:
        1%: -3.438
        5%: -2.865
        10%: -2.569
```

It is confirmed from ADF test that our data is non-stationary as p-value of the test >0.05.

ACF plot after applying one lag of differencing once:

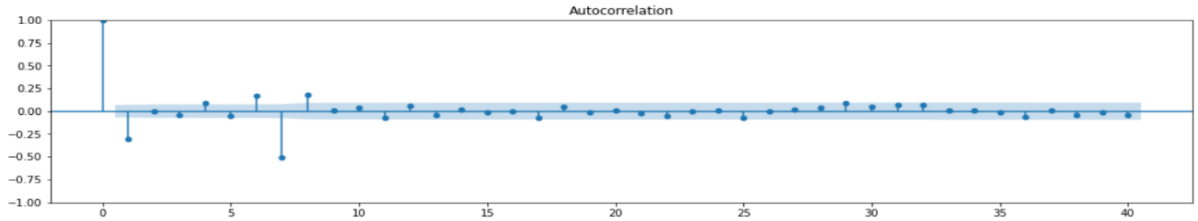


From the above plot we can infer that the data became stationary after applying one lag of differencing once as in the plot after lag 1 there is a cut-off. Also, we can observe that the multiple of 6^{th}, 7^{th} and 8^{th} lags are significant in which multiple of 7^{th} lags are most significant.

```
ADF Statistic: -11.890523
p-value: 0.000000
Critical Values:
        1%: -3.438
        5%: -2.865
        10%: -2.569
```

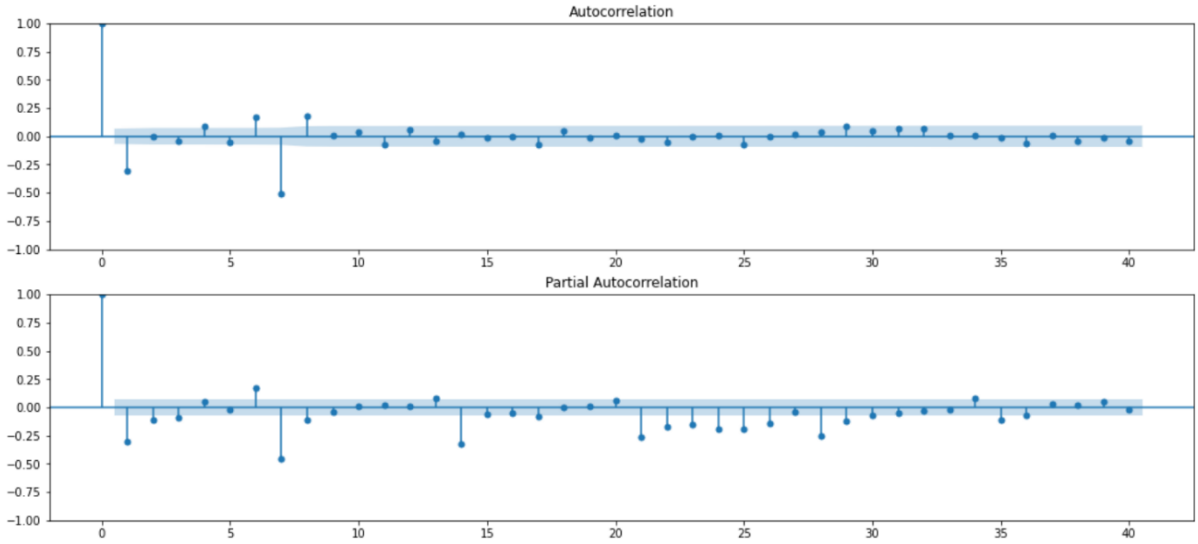
ADF test also confirmed that the data became stationary after applying one lag of differencing once.
Now, we should remove the seasonality by applying 7^{th} lag of differencing on top of one lag of differencing.

After applying 7^{th} lag of differencing once the ACF plot:



Model parameters:

ACF and PACF plots of data after applying 7th lag of differencing once on top of one lag of differencing once:



Analyzing the above plots, we get the following models are suitable for our dataset.

1. SARIMA(1,1,1)x(0,1,1,7)
2. SARIMA(2,1,1)x(0,1,1,7)
3. SARIMA(2,1,1)x((4,1,1,7)
4. SARIMA(1,1,1)x(4,1,1,7)

Model fitting:

$$SARIMA(1,1,1) \times (0,1,1,7)$$

SARIMAX Results

Dep. Variable:	Transformed	No. Observations:	860
Model:	SARIMAX(1, 1, 1)x(0, 1, 1, 7)	Log Likelihood	-2332.103
Date:	Tue, 01 Aug 2023	AIC	4672.205
Time:	01:39:04	BIC	4691.195
Sample:	06-01-2020	HQIC	4679.479
	- 10-08-2022		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.2258	0.069	3.282	0.001	0.091	0.361
ma.L1	-0.5867	0.063	-9.252	0.000	-0.711	-0.462
ma.S.L7	-0.9684	0.011	-85.427	0.000	-0.991	-0.946
sigma2	13.6460	0.377	36.168	0.000	12.907	14.385

Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	660.43
Prob(Q):	0.95	Prob(JB):	0.00
Heteroskedasticity (H):	1.39	Skew:	-0.14
Prob(H) (two-sided):	0.01	Kurtosis:	7.30

$SARIMA(2,1,1) \times (0,1,1,7)$

SARIMAX Results

Dep. Variable:	Transformed	No. Observations:	860
Model:	SARIMAX(2, 1, 1)x(0, 1, 1, 7)	Log Likelihood	-2305.466
Date:	Tue, 01 Aug 2023	AIC	4620.933
Time:	00:24:29	BIC	4644.671
Sample:	06-01-2020 - 10-08-2022	HQIC	4630.025
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.5696	0.024	23.795	0.000	0.523	0.616
ar.L2	0.1570	0.028	5.642	0.000	0.102	0.212
ma.L1	-0.9993	0.032	-31.456	0.000	-1.062	-0.937
ma.S.L7	-0.9624	0.013	-72.602	0.000	-0.988	-0.936
sigma2	12.6957	0.535	23.728	0.000	11.647	13.744

Ljung-Box (L1) (Q):	0.12	Jarque-Bera (JB):	610.11
Prob(Q):	0.73	Prob(JB):	0.00
Heteroskedasticity (H):	1.38	Skew:	-0.22
Prob(H) (two-sided):	0.01	Kurtosis:	7.12

$SARIMA(2,1,1) \times (4,1,1,7)$

SARIMAX Results

Dep. Variable:	Transformed	No. Observations:	860
Model:	SARIMAX(2, 1, 1)x(4, 1, 1, 7)	Log Likelihood	-2296.572
Date:	Tue, 01 Aug 2023	AIC	4611.144
Time:	00:38:40	BIC	4653.872
Sample:	06-01-2020 - 10-08-2022	HQIC	4627.509
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.5099	0.026	19.510	0.000	0.459	0.561
ar.L2	0.1411	0.030	4.648	0.000	0.082	0.201
ma.L1	-0.9878	0.008	-125.596	0.000	-1.003	-0.972
ar.S.L7	0.0044	0.047	0.092	0.926	-0.088	0.097
ar.S.L14	-0.0716	0.040	-1.801	0.072	-0.149	0.006
ar.S.L21	-0.0422	0.046	-0.923	0.356	-0.132	0.047
ar.S.L28	0.1437	0.037	3.910	0.000	0.072	0.216
ma.S.L7	-0.9674	0.015	-66.311	0.000	-0.996	-0.939
sigma2	12.4622	0.361	34.499	0.000	11.754	13.170

Ljung-Box (L1) (Q):	0.07	Jarque-Bera (JB):	630.46
Prob(Q):	0.78	Prob(JB):	0.00
Heteroskedasticity (H):	1.35	Skew:	-0.30
Prob(H) (two-sided):	0.01	Kurtosis:	7.17

$SARIMA(1,1,1) \times (4,1,1,7)$

SARIMAX Results

Dep. Variable:	Transformed	No. Observations:	860
Model:	SARIMAX(1, 1, 1)x(4, 1, 1, 7)	Log Likelihood	-2304.501
Date:	Tue, 01 Aug 2023	AIC	4625.001
Time:	01:06:14	BIC	4662.982
Sample:	06-01-2020 - 10-08-2022	HQIC	4639.548
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.5667	0.026	21.868	0.000	0.516	0.617
ma.L1	-0.9795	0.008	-115.723	0.000	-0.996	-0.963
ar.S.L7	-0.0012	0.047	-0.026	0.979	-0.093	0.090
ar.S.L14	-0.0979	0.040	-2.466	0.014	-0.176	-0.020
ar.S.L21	-0.0568	0.045	-1.262	0.207	-0.145	0.031
ar.S.L28	0.1594	0.036	4.380	0.000	0.088	0.231
ma.S.L7	-0.9649	0.014	-66.865	0.000	-0.993	-0.937
sigma2	12.7048	0.378	33.638	0.000	11.965	13.445

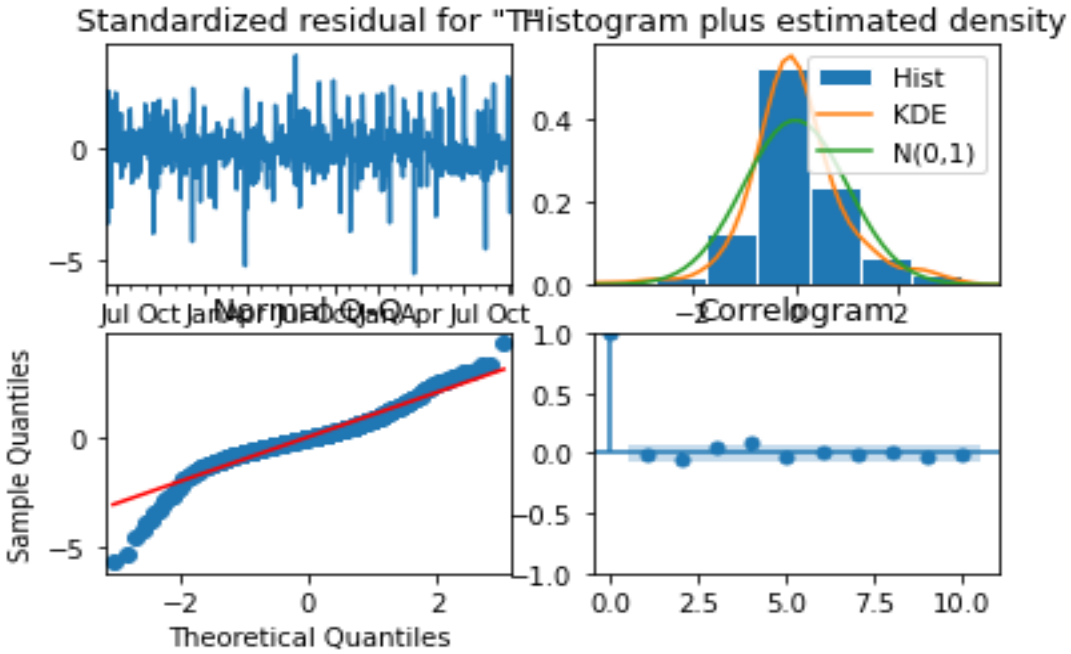
Ljung-Box (L1) (Q):	4.62	Jarque-Bera (JB):	520.35
Prob(Q):	0.03	Prob(JB):	0.00
Heteroskedasticity (H):	1.30	Skew:	-0.29
Prob(H) (two-sided):	0.03	Kurtosis:	6.79

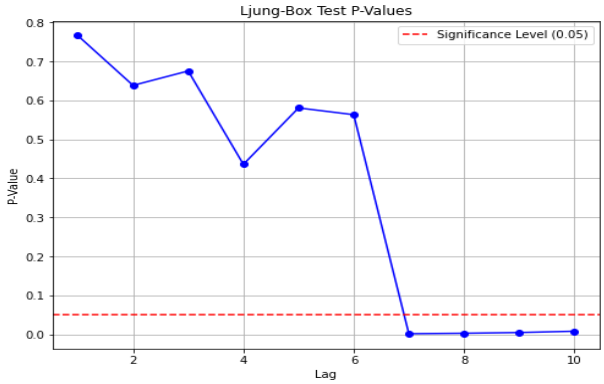
AIC scores of the models:

Model	AIC score
$SARIMA(1,1,1) \times (0,1,1,7)$	4672.205
$SARIMA(2,1,1) \times (0,1,1,7)$	4620.933
$SARIMA(1,1,1) \times (4,1,1,7)$	4611.144
$SARIMA(2,1,1) \times (4,1,1,7)$	4625.001

Based on AIC score we have chosen the model $SARIMA(1,1,1) \times (4,1,1,7)$

Residual analysis:

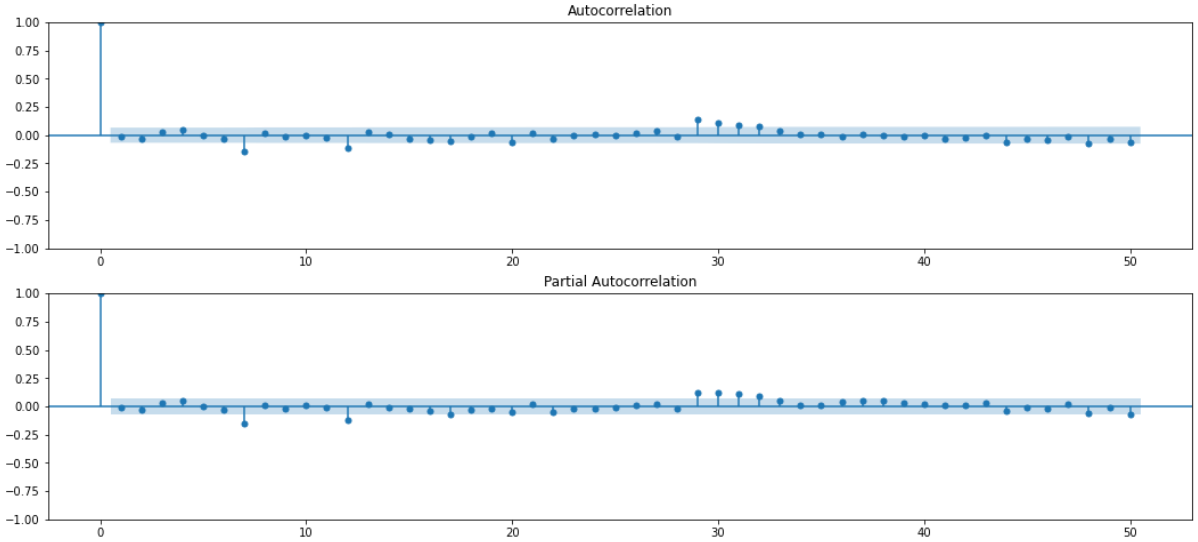




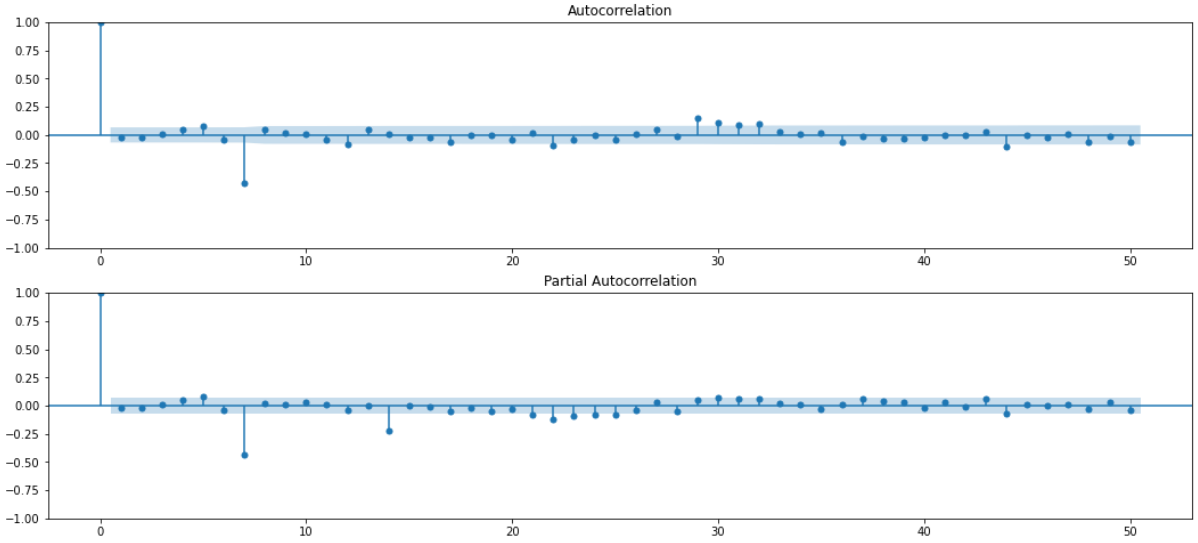
In Ljung-Box test there is strong evidence to reject the null hypothesis as p-value of the test after lag 6 is <0.05. Therefore, the residuals are correlated.

Now, we should model the residual of $SARIMA(1,1,1) \times (4,1,1,7)$ to extract more information or time dependent structure.

ACF and PACF plot of Residuals:



ACF and PACF plots of residuals after applying 7[th] lag of differencing once:



Preferable model for residuals is $SARIMA(0,0,0) \times (2,1,1,7)$

Fitting $\boldsymbol{SARIMA(0,0,0) \times (2,1,1,7)}$ to the residuals

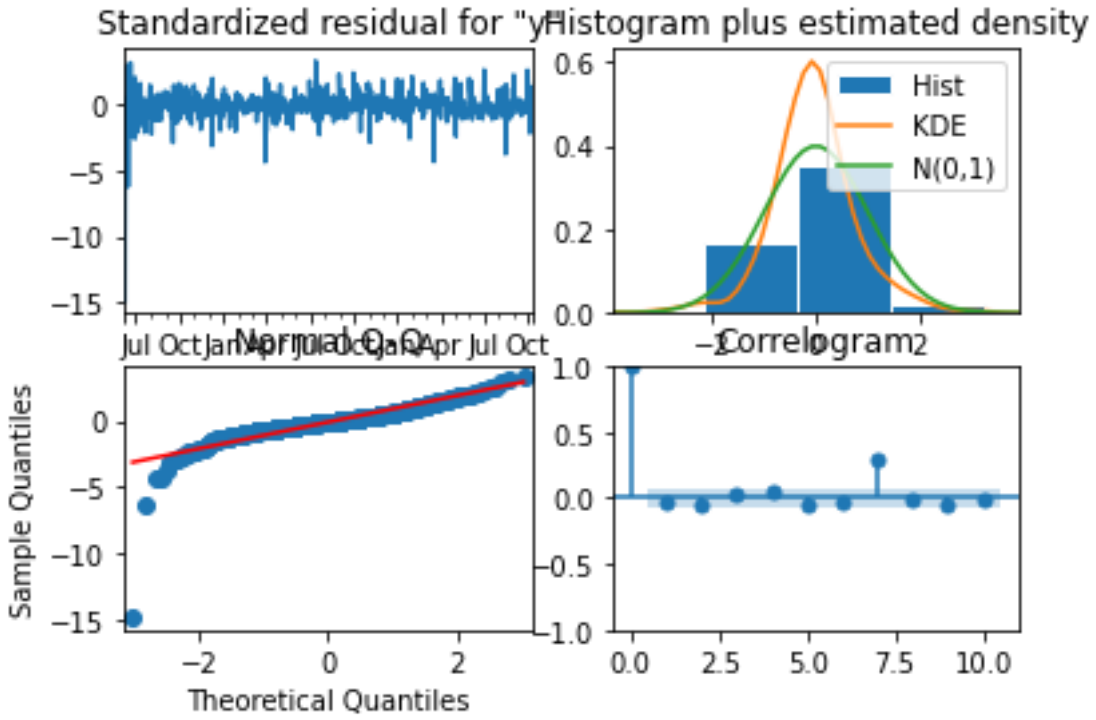
SARIMAX Results

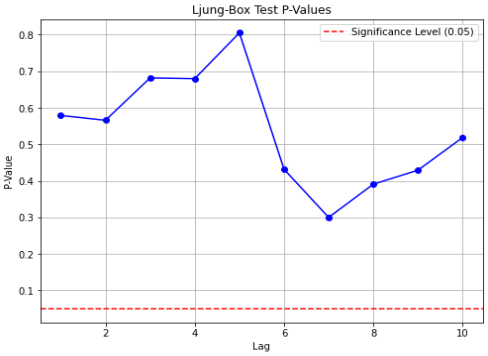
Dep. Variable:	y	No. Observations:	860
Model:	SARIMAX(2, 1, [1], 7)	Log Likelihood	-2519.320
Date:	Tue, 01 Aug 2023	AIC	5046.639
Time:	01:14:01	BIC	5065.634
Sample:	06-01-2020	HQIC	5053.914
	- 10-08-2022		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.S.L7	-0.2219	0.027	-8.353	0.000	-0.274	-0.170
ar.S.L14	0.0110	0.026	0.421	0.674	-0.040	0.062
ma.S.L7	-0.9998	1.178	-0.848	0.396	-3.310	1.310
sigma2	20.6184	24.120	0.855	0.393	-26.656	67.893

Ljung-Box (L1) (Q):	0.78	Jarque-Bera (JB):	124656.32
Prob(Q):	0.38	Prob(JB):	0.00
Heteroskedasticity (H):	0.44	Skew:	-4.21
Prob(H) (two-sided):	0.00	Kurtosis:	61.62

Residual analysis:






```
# Breusch Pagan test
_, p_value, _, _ = het_breuschpagan(res_resid, df_train)
# Test the null hypothesis at a certain significance level (e.g., 0.05)
alpha = 0.05
if p_value < alpha:
    print("The residuals are likely heteroscedastic (reject null hypothesis).")
else:
    print("The residuals are likely homoscedastic (fail to reject null hypothesis).")
```

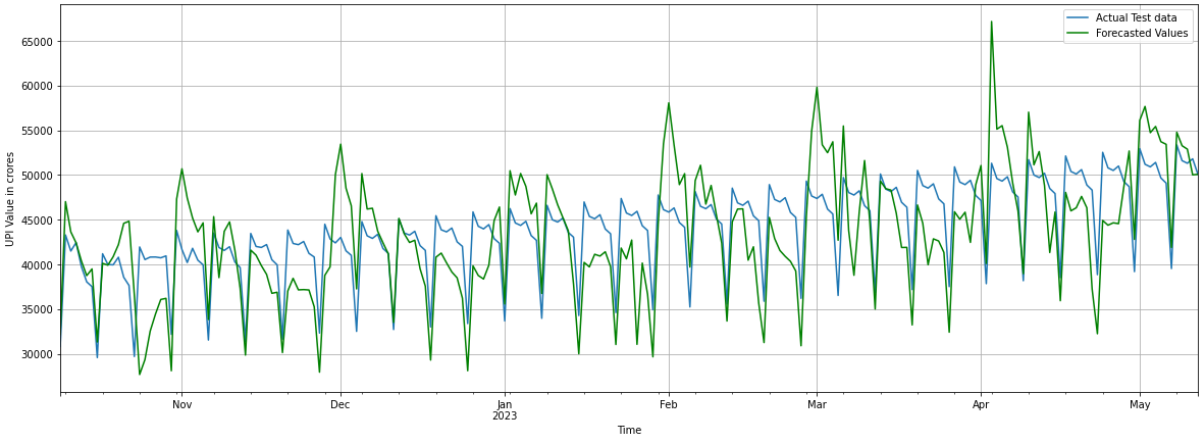
```
The residuals are likely homoscedastic (fail to reject null hypothesis).
```

Now, we can conclude that the residuals of the residuals are follows white noise process.

Forecast:

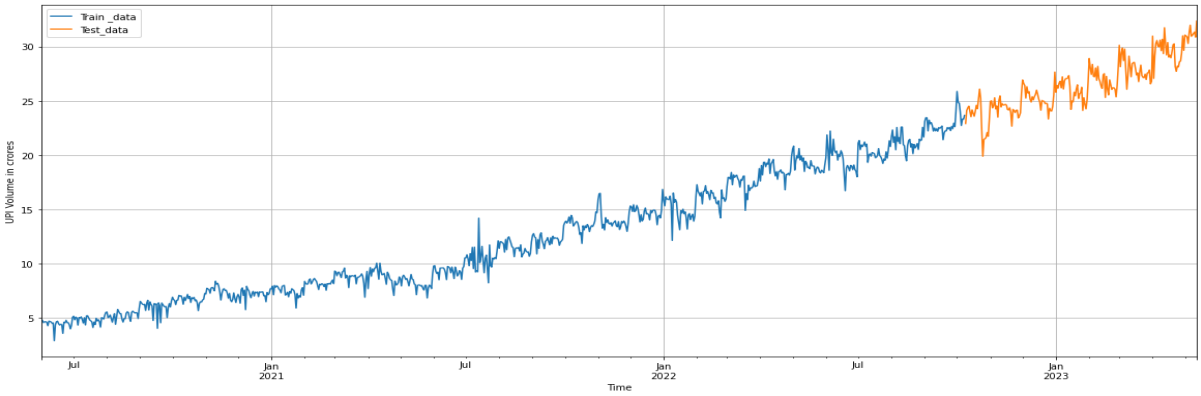
Now, we are going to forecast using $SARIMA(2,1,1) \times (4,1,1,7) + SARIMA(0,0,0) \times (2,1,1,7)[of\ residual]$:

DATE	Forecasted	Value
2022-10-09	30828.612130	32190.94
2022-10-10	43299.278225	47027.29
2022-10-11	41500.901290	43642.01
2022-10-12	42449.355779	42412.68
2022-10-13	39754.925722	40387.93
2022-10-14	38054.803432	38748.95
2022-10-15	37516.232282	39499.74
2022-10-16	29583.996144	31327.83
2022-10-17	41198.335471	40125.10
2022-10-18	39996.752857	39917.14



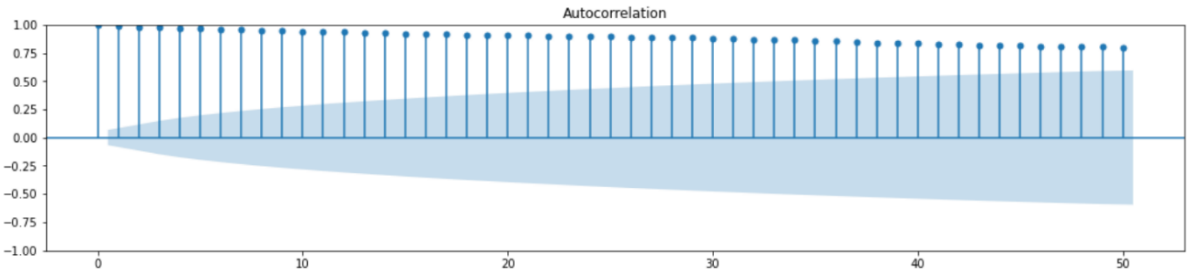
UPI Volume:

In this part we did not use any transformations.
Here also we used initial 80% data for training the model and rest 20% for testing



Stationarity:

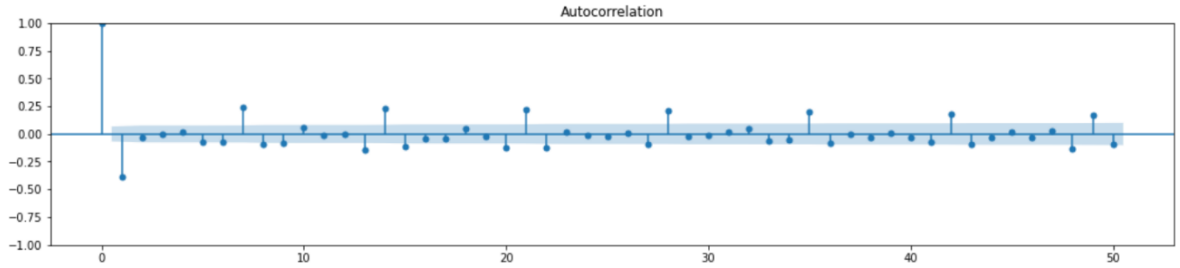
Let's plot the ACF plot of the data



From above plot suggests that the data is non-stationary. We should verify it using ADF test

```
ADF Statistic: 1.067322
p-value: 0.994928
Critical Values:
        1%: -3.438
        5%: -2.865
        10%: -2.569
```

ACF plot of the data after applying one lag of differencing once



There is a shut off after lag one. It suggests that the data may became stationary after applying one lag of differencing once.

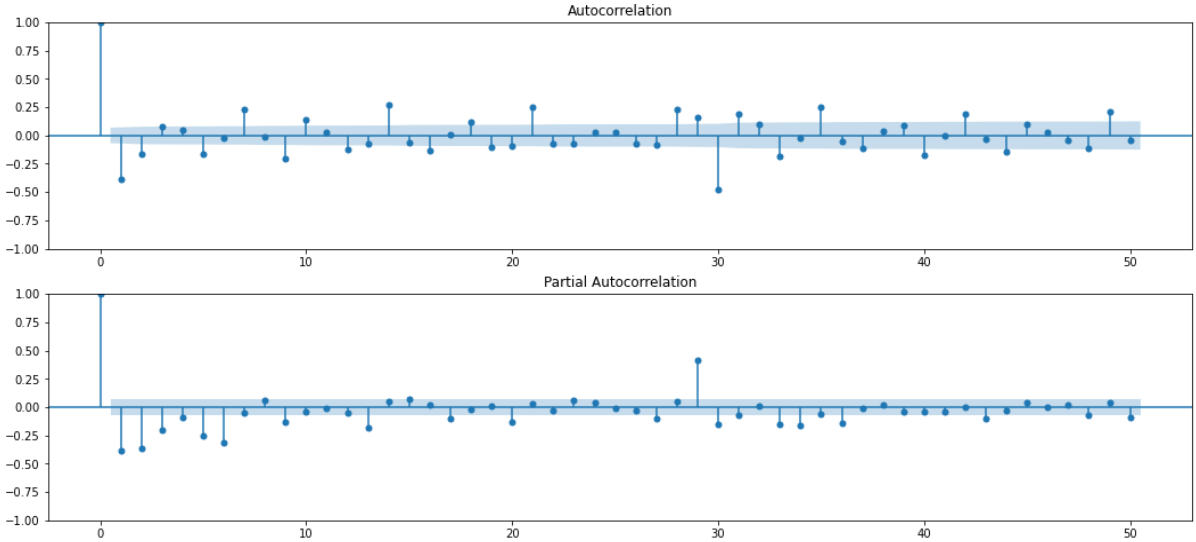
```
ADF Statistic: -11.890523
p-value: 0.000000
Critical Values:
        1%: -3.438
        5%: -2.865
        10%: -2.569
```

As p-value of the ADF test of data after applying one lag of differencing is less than 0.05, we can conclude that the data became stationary. So, we eliminate the trend from the data.

We have seen that the appropriate period of UPI Volume data is 30.

To eliminate seasonal pattern from data we applied 30^{th} lag of differencing once on top of one lag of differencing.

ACF and PACF plots after applying 30^{th} lag of differencing:



Suggested SARIMA models are:

1. $SARIMA(3,1,1) \times (0,1,1,30)$
2. $SARIMA(2,1,1) \times (1,1,1,30)$
3. $SARIMA(2,1,1) \times (0,1,1,30)$

Model building:

$$SARIMA(3,1,1) \times (0,1,1,30)$$

SARIMAX Results

Dep. Variable:	Vol	No. Observations:	860
Model:	SARIMAX(3, 1, 1)x(0, 1, 1, 30)	Log Likelihood	-844.757
Date:	Thu, 03 Aug 2023	AIC	1701.514
Time:	10:10:27	BIC	1729.835
Sample:	06-01-2020 - 10-08-2022	HQIC	1712.375
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.2087	0.033	6.316	0.000	0.144	0.273
ar.L2	0.0268	0.039	0.690	0.490	-0.049	0.103
ar.L3	0.0037	0.032	0.117	0.907	-0.058	0.066
ma.L1	-0.8832	0.023	-39.200	0.000	-0.927	-0.839
ma.S.L30	-0.8521	0.021	-40.168	0.000	-0.894	-0.811
sigma2	0.4282	0.012	34.393	0.000	0.404	0.453

Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	909.92
Prob(Q):	0.96	Prob(JB):	0.00
Heteroskedasticity (H):	2.56	Skew:	0.08
Prob(H) (two-sided):	0.00	Kurtosis:	8.13

$$SARIMA(2,1,1) \times (0,1,1,30)$$

SARIMAX Results

Dep. Variable:	Vol	No. Observations:	860
Model:	SARIMAX(2, 1, 1)x(0, 1, 1, 30)	Log Likelihood	-844.761
Date:	Thu, 03 Aug 2023	AIC	1699.522
Time:	10:11:53	BIC	1723.123
Sample:	06-01-2020 - 10-08-2022	HQIC	1708.573
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.2073	0.031	6.699	0.000	0.147	0.268
ar.L2	0.0264	0.039	0.684	0.494	-0.049	0.102
ma.L1	-0.8819	0.021	-41.955	0.000	-0.923	-0.841
ma.S.L30	-0.8517	0.021	-40.535	0.000	-0.893	-0.811
sigma2	0.4283	0.012	34.729	0.000	0.404	0.452

Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	906.13
Prob(Q):	0.96	Prob(JB):	0.00
Heteroskedasticity (H):	2.56	Skew:	0.08
Prob(H) (two-sided):	0.00	Kurtosis:	8.12

$$SARIMA(2,1,1) \times (1,1,1,30)$$

SARIMAX Results

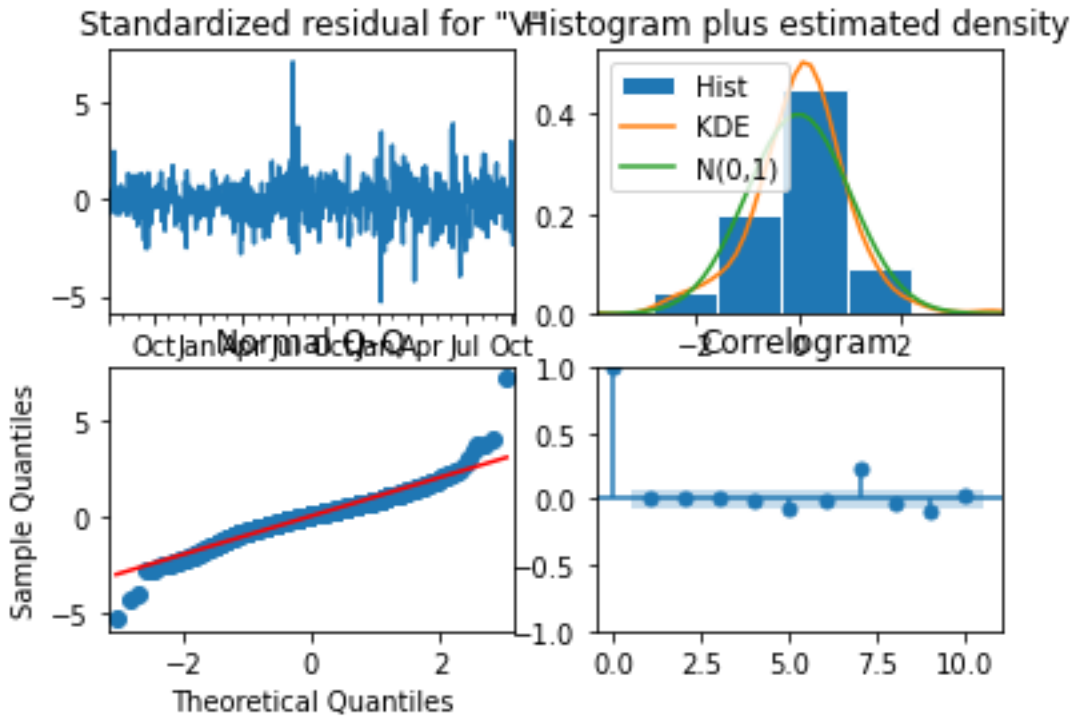
Dep. Variable:	Vol	No. Observations:	860
Model:	SARIMAX(2, 1, 1)x(1, 1, 1, 30)	Log Likelihood	-844.760
Date:	Thu, 03 Aug 2023	AIC	1701.519
Time:	10:13:21	BIC	1729.840
Sample:	06-01-2020 - 10-08-2022	HQIC	1712.381
Covariance Type:	opg		

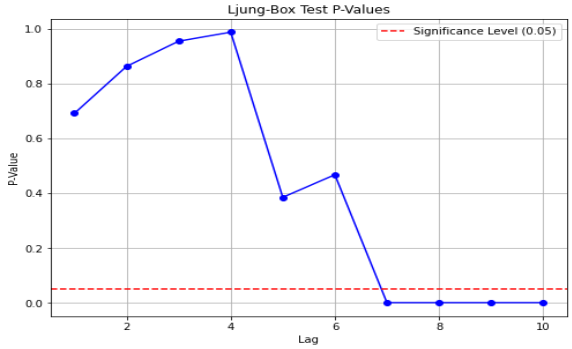
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.2073	0.031	6.698	0.000	0.147	0.268
ar.L2	0.0266	0.039	0.691	0.490	-0.049	0.102
ma.L1	-0.8819	0.021	-41.540	0.000	-0.924	-0.840
ar.S.L30	-0.0022	0.040	-0.055	0.956	-0.081	0.077
ma.S.L30	-0.8511	0.025	-33.662	0.000	-0.901	-0.802
sigma2	0.4283	0.012	34.591	0.000	0.404	0.453

Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	906.40
Prob(Q):	0.96	Prob(JB):	0.00
Heteroskedasticity (H):	2.56	Skew:	0.08
Prob(H) (two-sided):	0.00	Kurtosis:	8.12

Based on AIC score we are going with the model $SARIMA(2,1,1) \times (0,1,1,30)$ for further analysis.

Residual analysis:



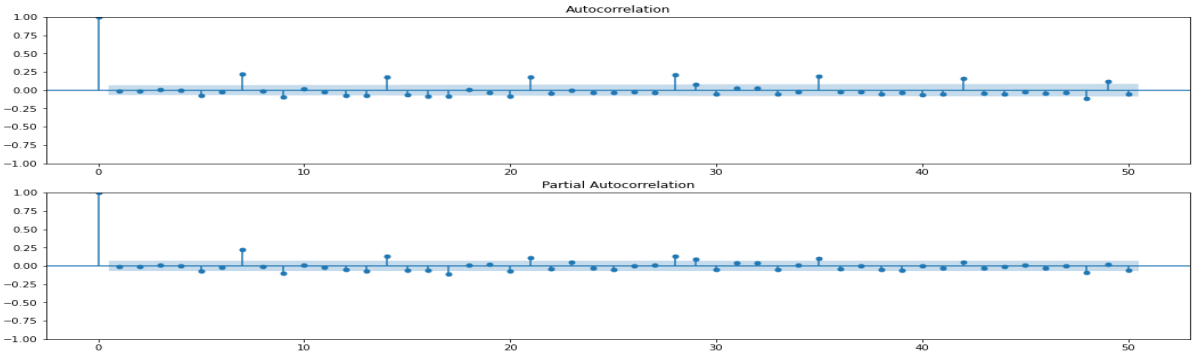


Ljung Box test suggests that the residuals are correlated as p-value after lag 6 is less than 0.05 it means there is strong evidence to reject null hypothesis which is “residuals are not correlated”.

We should model the residuals of $SARIMA(2,1,1) \times (0,1,1,30)$ to extract more information or time dependent structure.

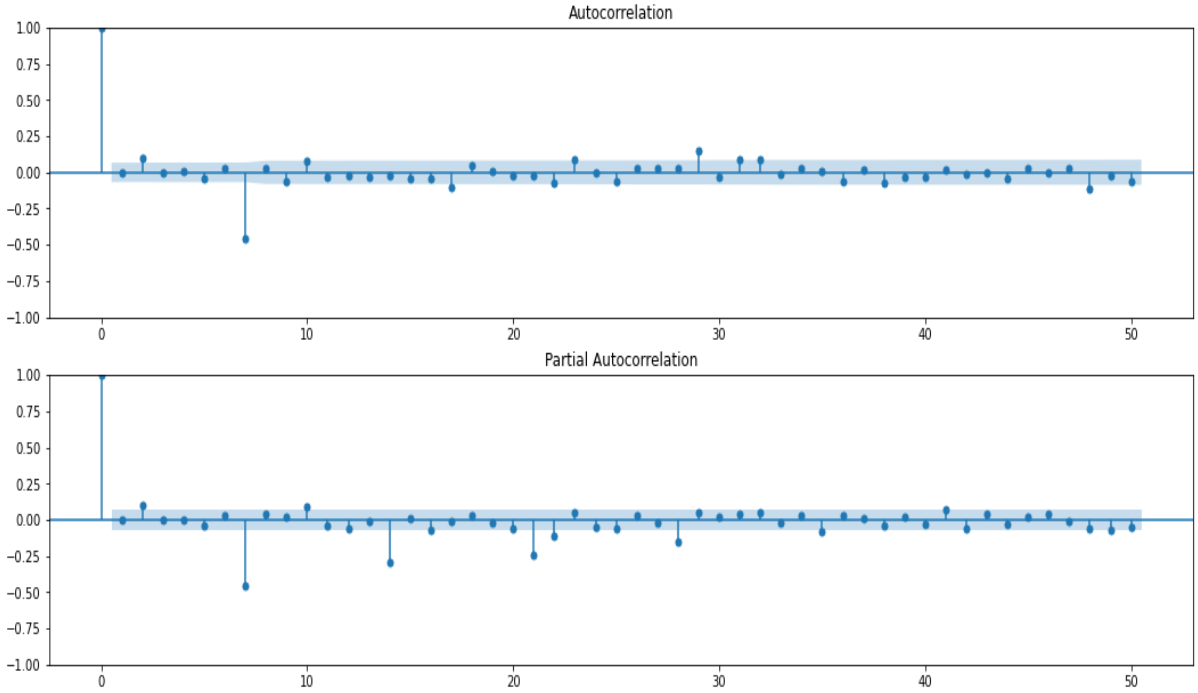
Modelling of residuals of $\boldsymbol{SARIMA(2,1,1) \times (0,1,1,30)}$:

ACF and PACF plots of residuals:



We can observe that lags at multiple of 7 are quite significant. We should apply 7^{th} lag of differencing to eliminate 7 period seasonal pattern from residuals.

ACF and PACF plots after 7^{th} lag of differencing once:



It suggests a suitable model for residuals is $SARIMA(0,0,0) \times (0,1,1,7)$

Fitting $SARIMA(0,0,0) \times (0,1,1,7)$ model to the residuals

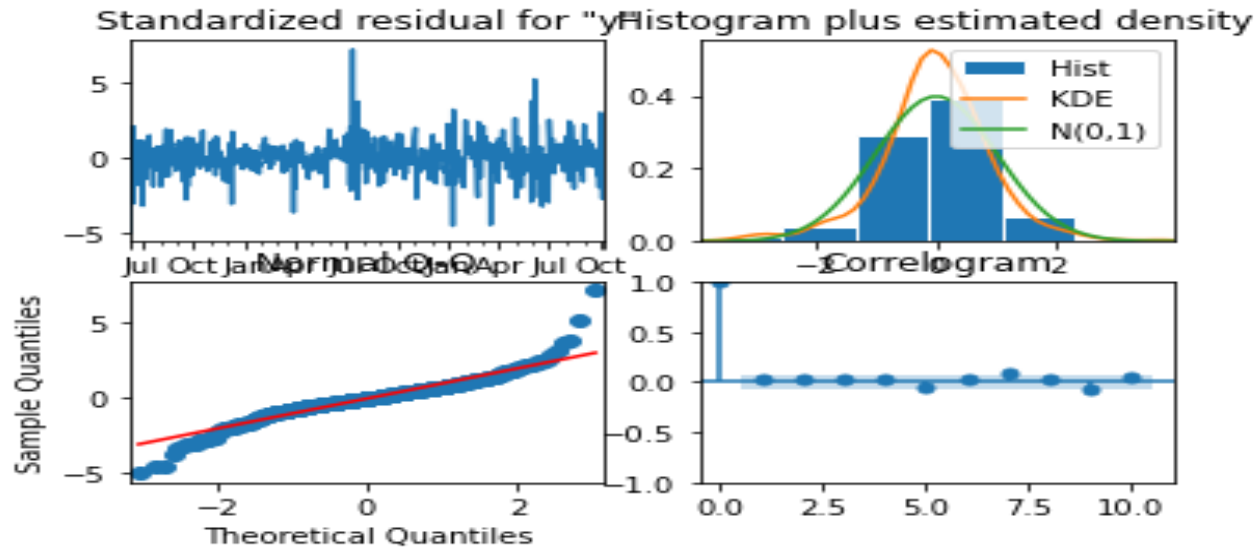
SARIMAX Results

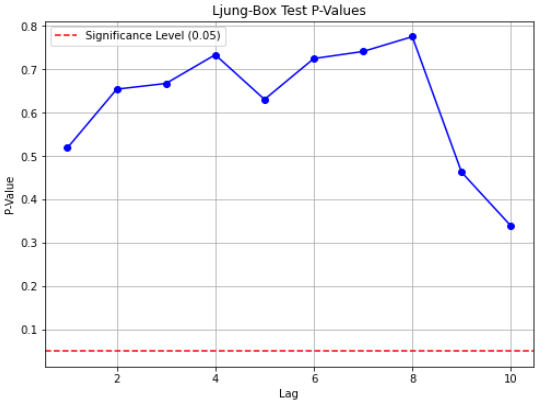
Dep. Variable:	y	No. Observations:	860
Model:	SARIMAX(0, 1, [1], 7)	Log Likelihood	-819.437
Date:	Thu, 03 Aug 2023	AIC	1642.873
Time:	10:28:53	BIC	1652.371
Sample:	06-01-2020	HQIC	1646.510
	- 10-08-2022		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ma.S.L7	-0.9666	0.013	-72.246	0.000	-0.993	-0.940
sigma2	0.3911	0.009	41.411	0.000	0.373	0.410

Ljung-Box (L1) (Q):	0.83	Jarque-Bera (JB):	1356.41
Prob(Q):	0.36	Prob(JB):	0.00
Heteroskedasticity (H):	1.91	Skew:	0.06
Prob(H) (two-sided):	0.00	Kurtosis:	9.18

Residual analysis of residuals of residual:





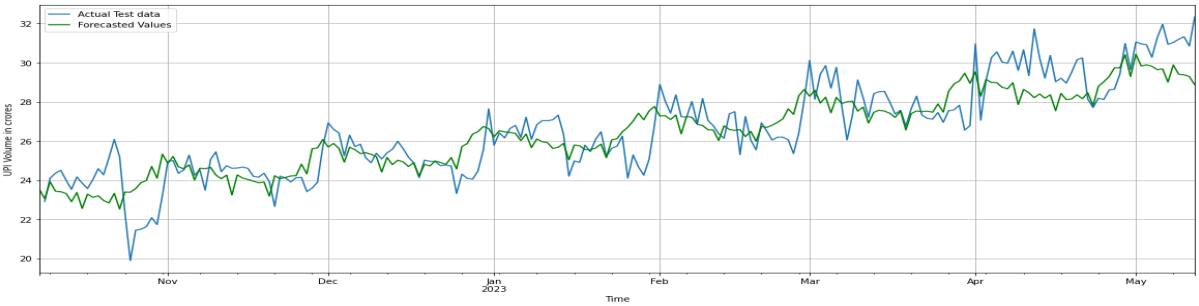

```
# Breusch Pagan test
_, p_value, _, _ = het_breuschpagan(res_resid30, df_train)
# Test the null hypothesis at a certain significance level (e.g., 0.05)
alpha = 0.05
if p_value < alpha:
    print("The residuals are likely heteroscedastic (reject null hypothesis).")
else:
    print("The residuals are likely homoscedastic (fail to reject null hypothesis).")
```

```
The residuals are likely homoscedastic (fail to reject null hypothesis).
```

From above analysis we can conclude that the residuals of residual follows white noise process.

Forecast:

Now, we are going to forecast using $SARIMA(2,1,1) \times (0,1,1,30) + SARIMA(0,0,0) \times (0,1,1,7)[for\ residual]$



First 10 days forecast and actual value of UPI Volume:

DATE	Vol	forecasted
2022-10-09	22.9061	23.048701
2022-10-10	24.1002	23.914540
2022-10-11	24.3491	23.437936
2022-10-12	24.5081	23.408932
2022-10-13	23.9986	23.316457
2022-10-14	23.5274	22.898704
2022-10-15	24.1634	23.368645
2022-10-16	23.8485	22.553453
2022-10-17	23.5707	23.285577
2022-10-18	24.0264	23.120702

8.3. Modelling With Some Smoothing Techniques

In this section, We, are going to model our time series data using some of the smoothing techniques like moving averages and Exponential techniques as follows:

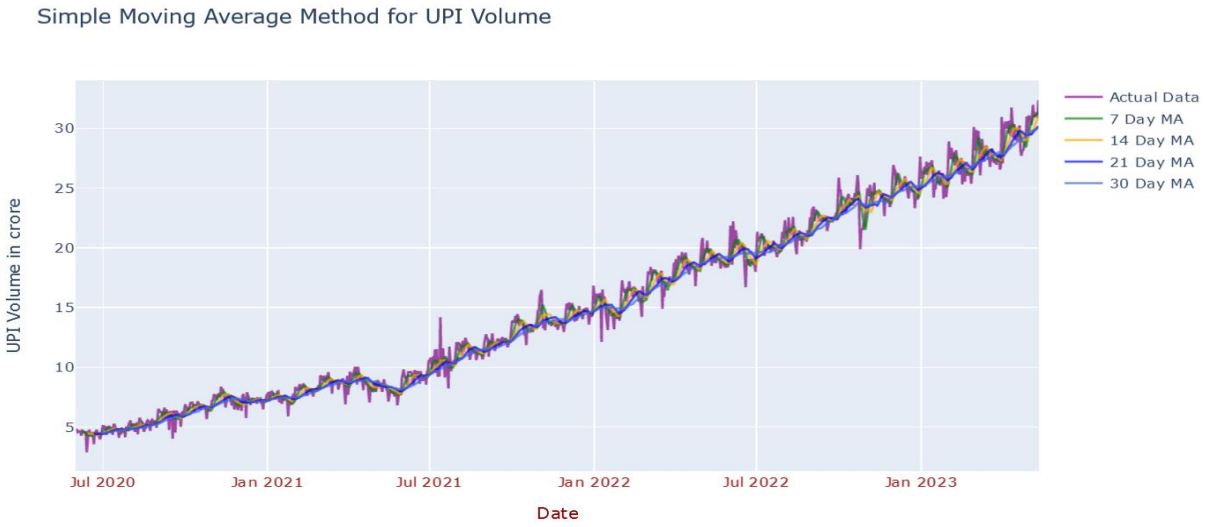
8.3.1. Simple Moving Average method:

A simple moving average forecast is another simple method. Instead of taking the average of all past values like in Naïve Method, it takes the mean of the latest k steps as the forecast**.** The forecast equation is given by:

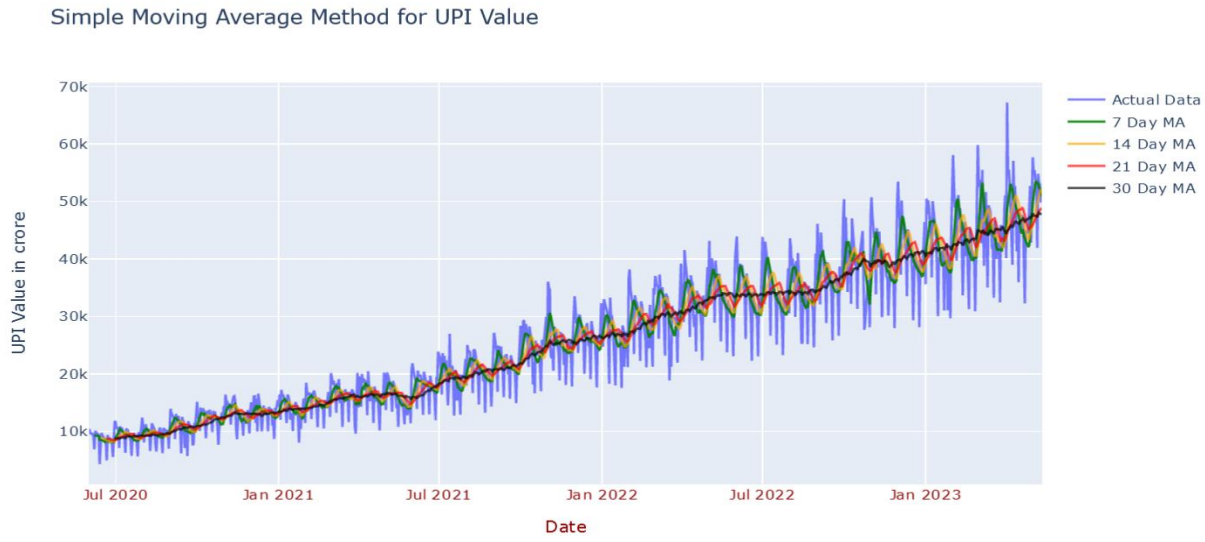
$$f_t = \left(\frac{1}{k}\right) * (y_{t-1} + y_{t-2} + \cdots + y_{t-k})$$

Note: The Forecast Equation may not be a straight line.

- **Modelling the Moving Average method to our data:**
  Moving Average for UPI Volume data:



Moving Average for UPI Value data:



8.3.1. Exponential Smoothing Techniques (ETS)

What is Exponential Smoothing?

Exponential smoothing is basically a weighted moving average technique. We know that in the moving average smoothing the past observations are weighted equally, but in this case, smoothing is done by assigning exponentially decreasing weights to the past observation. There are a few different variants of ETS – **Simple exponential smoothing**, **double exponential smoothing, Triple Exponential OR The Holt-Winter's Exponential smoothing.**

General form of the Exponential Smoothing Model:

$$f_t = \alpha * Y_{t-1} + \alpha(1-\alpha) * Y_{t-2} + \alpha(1-\alpha)^2 * Y_{t-3} + \cdots$$

Here, $0 \leq \alpha \leq 1$ is the smoothing parameter.

8.3.1.1. **Simple Exponential Smoothing (SES):** This method of smoothing is most suited when there is no trends or seasonality, and forecasting is going is to be a flat line. The forecast is generated using the following formula:

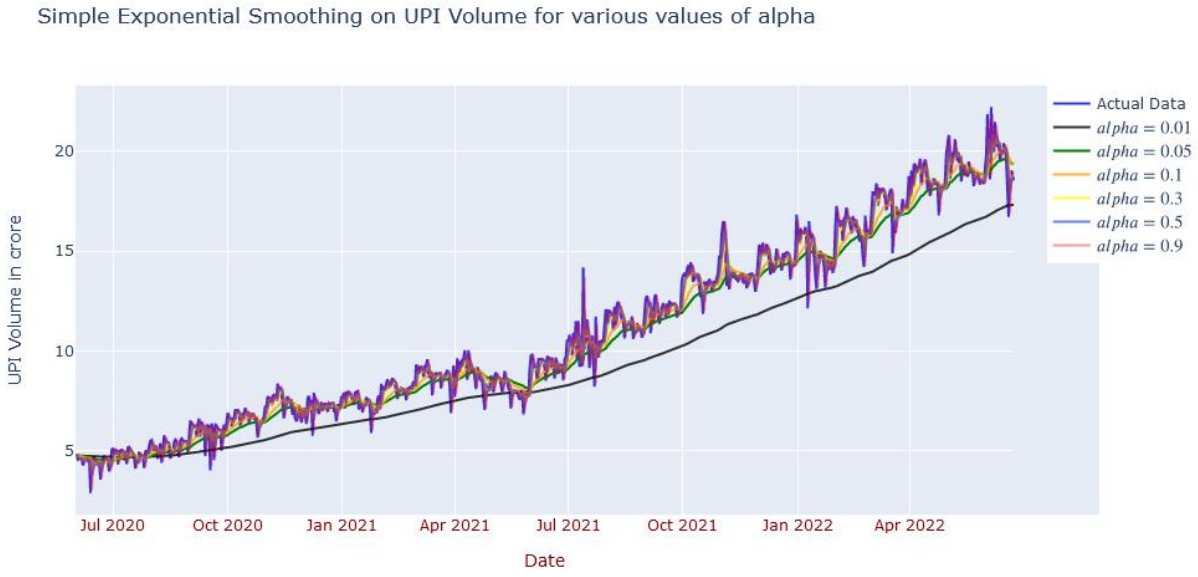
$$f_0 = y_0$$

$$f_t = \alpha * y_{t-1} + (1-\alpha) * f_{t-1} \qquad ; t > 0$$

Where $\boldsymbol{0 < \alpha < 1}$ is called the **data Smoothing parameter**.

Applying the Simple Exponential Smoothing Method to our Time series data:

- **SES on UPI Volume data**



Now, we must find the best Value of the smoothing parameter i.e., the optimum value of the Smoothing Parameter, So, we have used the **Grid-Search Method** based on minimising the Root Mean Square Error and The Mean Absolute Percentage Error to find the best value of the smoothing parameter and the results are as follows:

```
In [309]: for i in np.arange(0.0,0.2,0.01):
              model_fit = model.fit(smoothing_level=i,optimized=False)
              y_hat_avg = DF_vd.copy()
              y_hat_avg['SES'] = model_fit.forecast(len(DF_vd['Vol']))
              rmse = np.sqrt(mean_squared_error(DF_vd.Vol, y_hat_avg.SES))
              rmse = round(rmse, 4)
              abs_error = np.abs(DF_vd['Vol']-y_hat_avg.SES)
              actual = DF_vd['Vol']
              mape = np.round(np.mean(abs_error/actual),6)*100
              print('smoothing_level : ',np.round(i,3),'RMSE : ' ,np.round(rmse,4) ,'MAPE : ',mape)
```

```
smoothing_level :  0.0 RMSE :  17.5937 MAPE :  78.4239
smoothing_level :  0.01 RMSE :  5.2576 MAPE :  21.5221
smoothing_level :  0.02 RMSE :  4.0207 MAPE :  15.4672
smoothing_level :  0.03 RMSE :  3.6667 MAPE :  13.7575
smoothing_level :  0.04 RMSE :  3.5356 MAPE :  13.1374
smoothing_level :  0.05 RMSE :  3.4867 MAPE :  12.906899999999998
smoothing_level :  0.06 RMSE :  3.4747 MAPE :  12.850700000000002
smoothing_level :  0.07 RMSE :  3.4816 MAPE :  12.8831
smoothing_level :  0.08 RMSE :  3.4993 MAPE :  12.966700000000001
smoothing_level :  0.09 RMSE :  3.5235 MAPE :  13.0806
smoothing_level :  0.1 RMSE :  3.5516 MAPE :  13.2128
smoothing_level :  0.11 RMSE :  3.5821 MAPE :  13.3567
smoothing_level :  0.12 RMSE :  3.6137 MAPE :  13.5069
smoothing_level :  0.13 RMSE :  3.6457 MAPE :  13.6583
smoothing_level :  0.14 RMSE :  3.6775 MAPE :  13.808000000000002
smoothing_level :  0.15 RMSE :  3.7085 MAPE :  13.9556
smoothing_level :  0.16 RMSE :  3.7384 MAPE :  14.099800000000002
smoothing_level :  0.17 RMSE :  3.767 MAPE :  14.2379
smoothing_level :  0.18 RMSE :  3.7941 MAPE :  14.3697
smoothing_level :  0.19 RMSE :  3.8196 MAPE :  14.4942
```

$$Best\ Smoothing\ parameter = 0.06$$

From the above result, it can be concluded that, the best value of the Data Smoothing Parameter $(\alpha)\ is\ 0.06$

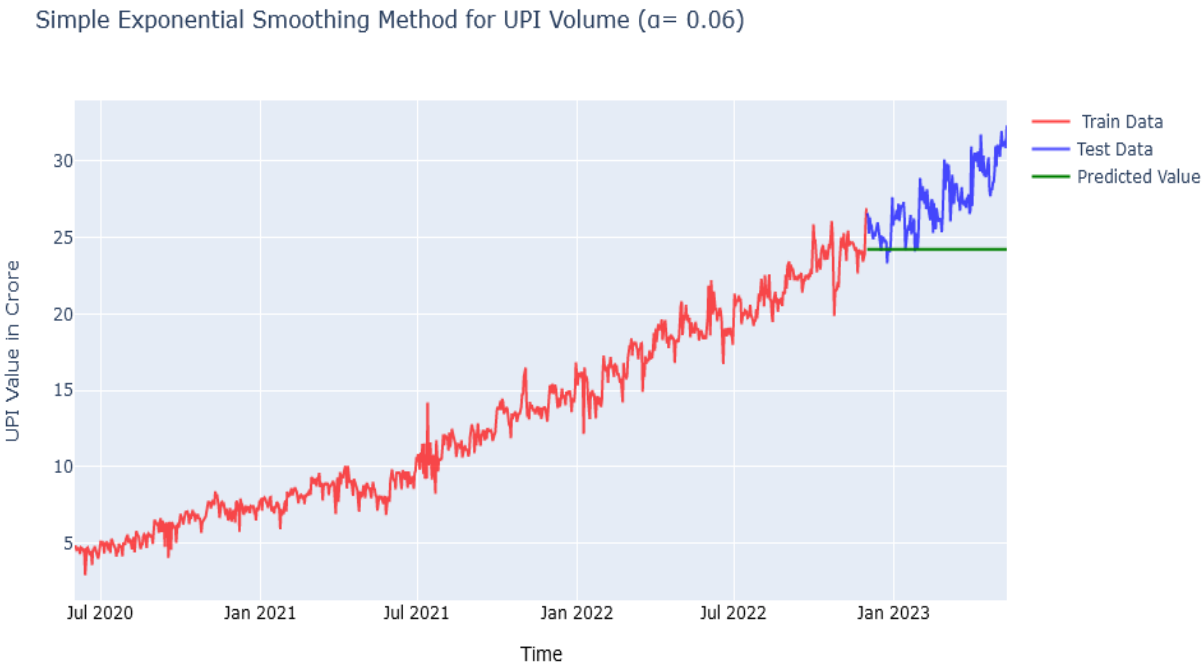
Fitting the Simple Exponential Smoothing With $\alpha = 0.06$:

SimpleExpSmoothing Model Results

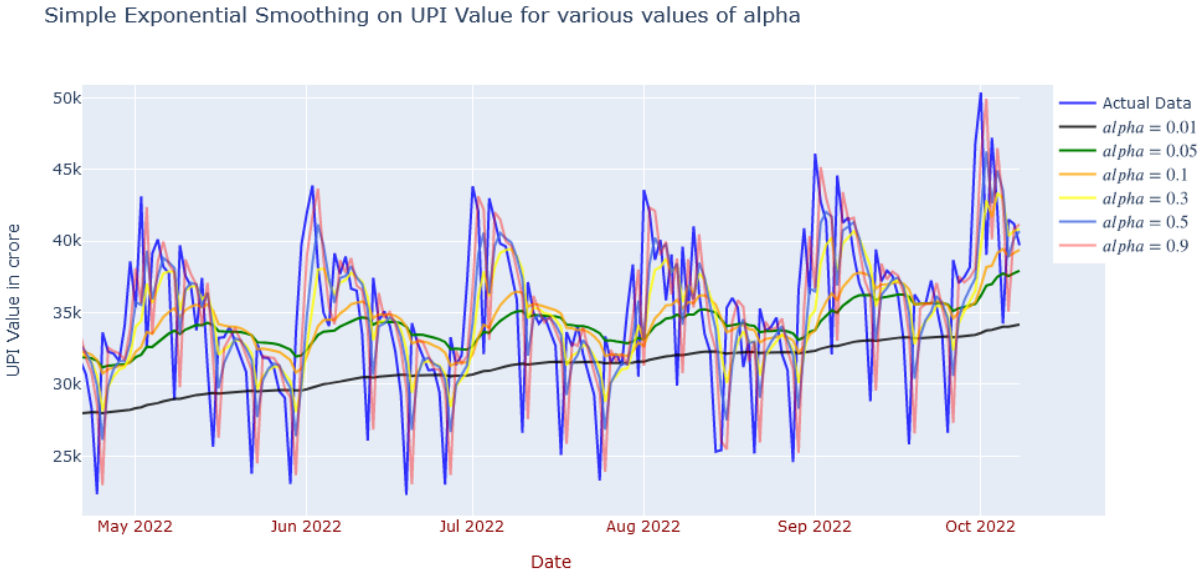
Dep. Variable:	Vol	**No. Observations:**	914
Model:	SimpleExpSmoothing	**SSE**	718.430
Optimized:	False	**AIC**	-216.056
Trend:	None	**BIC**	-206.420
Seasonal:	None	**AICC**	-216.012
Seasonal Periods:	None	**Date:**	Sun, 30 Jul 2023
Box-Cox:	False	**Time:**	12:27:13
Box-Cox Coeff.:	None		

	coeff	code	optimized
smoothing_level	0.0600000	alpha	False
initial_level	4.7697000	l.0	False

Visualization of SES on Train and Test data:



- **SES on UPI Value data**



Now, again for the UPI value data, we must find the best Value of the smoothing parameter i.e., the optimum value of the Smoothing Parameter, So, we have used the **Grid-Search Method** based on minimising the Root Mean Square Error and The Mean Absolute Percentage Error to find the best value of the smoothing parameter and the results are as follows:

```
In [328]: for i in np.arange(0.0,0.2,0.01):
              model_fit = model.fit(smoothing_level=i,optimized=False)
              y_hat_avg = DF_vd.copy()
              y_hat_avg['SES'] = model_fit.forecast(len(DF_vd['Val']))
              rmse = np.sqrt(mean_squared_error(DF_vd.Val, y_hat_avg.SES))
              rmse = round(rmse, 4)
              abs_error = np.abs(DF_vd['Val']-y_hat_avg.SES)
              actual = DF_vd['Val']
              mape = np.round(np.mean(abs_error/actual),6)*100
              print('smoothing_level : ',np.round(i,3),'RMSE : ' ,np.round(rmse,4) ,'MAPE : ',mape)
```

```
smoothing_level :  0.0 RMSE :  27027.6281 MAPE :  70.9501
smoothing_level :  0.01 RMSE :  8467.6089 MAPE :  17.9627
smoothing_level :  0.02 RMSE :  7139.048 MAPE :  14.933399999999999
smoothing_level :  0.03 RMSE :  6865.1062 MAPE :  14.4038
smoothing_level :  0.04 RMSE :  6805.768 MAPE :  14.2896
smoothing_level :  0.05 RMSE :  6818.0414 MAPE :  14.313400000000001
smoothing_level :  0.06 RMSE :  6860.7236 MAPE :  14.3954
smoothing_level :  0.07 RMSE :  6918.3043 MAPE :  14.505199999999999
smoothing_level :  0.08 RMSE :  6983.7658 MAPE :  14.635000000000002
smoothing_level :  0.09 RMSE :  7053.359 MAPE :  14.77
smoothing_level :  0.1 RMSE :  7124.7366 MAPE :  14.9056
smoothing_level :  0.11 RMSE :  7196.2288 MAPE :  15.0467
smoothing_level :  0.12 RMSE :  7266.5522 MAPE :  15.187800000000001
smoothing_level :  0.13 RMSE :  7334.6933 MAPE :  15.323
smoothing_level :  0.14 RMSE :  7399.8587 MAPE :  15.451400000000001
smoothing_level :  0.15 RMSE :  7461.4505 MAPE :  15.575800000000001
smoothing_level :  0.16 RMSE :  7519.0464 MAPE :  15.703700000000001
smoothing_level :  0.17 RMSE :  7572.3804 MAPE :  15.8289
smoothing_level :  0.18 RMSE :  7621.3215 MAPE :  15.9437
smoothing_level :  0.19 RMSE :  7665.8514 MAPE :  16.0472
```

$$Best\ Smoothing\ parameter = 0.04$$

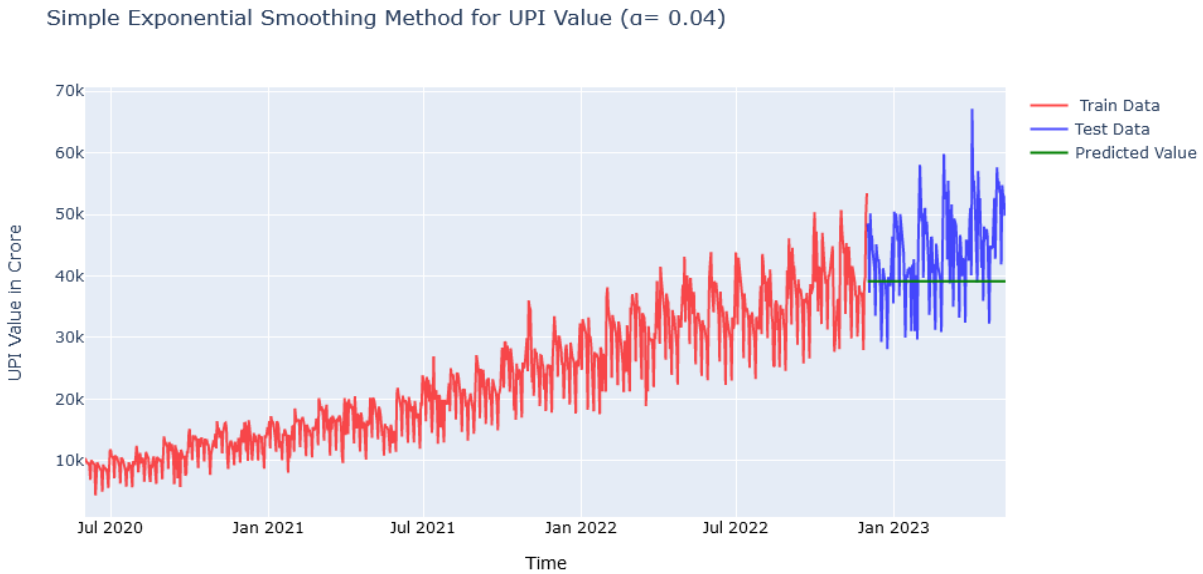
Fitting the Simple Exponential Smoothing With $\alpha = 0.04$:

SimpleExpSmoothing Model Results

Dep. Variable:	Val	**No. Observations:**	914
Model:	SimpleExpSmoothing	**SSE**	14135661095.095
Optimized:	False	**AIC**	15134.480
Trend:	None	**BIC**	15144.116
Seasonal:	None	**AICC**	15134.524
Seasonal Periods:	None	**Date:**	Sun, 30 Jul 2023
Box-Cox:	False	**Time:**	12:41:31
Box-Cox Coeff.:	None		

	coeff	code	optimized
smoothing_level	0.0400000	alpha	False
initial_level	10413.110	l.0	False

Visualization of SES on Train and Test data:



8.3.1.2. Double Exponential Smoothing (DES) Method

- DES extends the smoothing idea to model the trend as well, that is why DES is also known as the Exponential Smoothing with Trend
- It has two smoothing equations, one for the level and the other one for the trend
- Once we have the estimate of the level function and the trend function, then we can combine them to get the whole forecasting equation
- This forecast is not necessarily flat, because the estimated trend is used to extrapolate the future

The Forecast Equation id given by:

$$f_0 = y_0$$

$$b_0 = y_1 - y_0$$

$$and\ for\ t \geq 1, we\ have$$

$$f_t = \alpha * y_t + (1 - \alpha) * (f_{t-1} + b_{t-1})$$

$$b_t = \beta * (f_t - f_{t-1}) + (1 - \beta) * b_{t-1}$$

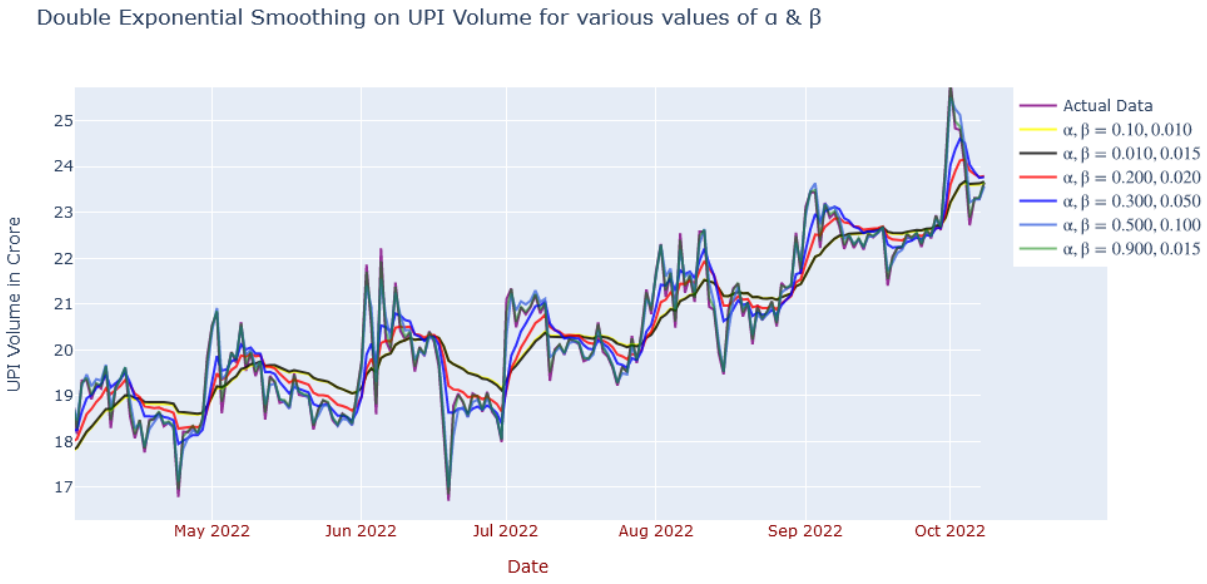
Where;

$$b_t = estimate\ of\ the\ trend\ line\ at\ time\ t\ and$$

$$0 < \beta < 1\ is\ the\ trend\ smoothing\ OR\ paremeter$$

Applying the Double Exponential Smoothing Method to our Time series data:

- **DES on UPI Volume data**



Now, again for the Double Exponential Smoothing method, we must find the best Value of the trend smoothing parameter i.e., the optimum value of the Trend Smoothing Parameter, So, we have used the **Grid-Search Method** based on minimising the Root Mean Square Error and The Mean Absolute Percentage Error to find the best value of the Trend smoothing parameter and the results are as follows:

```
In [348]: for i in np.arange(0.0,0.03,0.001):
              model_fit = model_double_exp.fit(smoothing_level = 0.06, smoothing_slope = i, optimized = False)
              y_hat_avg = DF_vd.copy()
              y_hat_avg['DES'] = model_fit.forecast(len(DF_vd['Vol']))
              rmse = np.sqrt(mean_squared_error(DF_vd.Vol, y_hat_avg.DES))
              rmse = round(rmse, 4)
              abs_error = np.abs(DF_vd['Vol']-y_hat_avg.DES)
              actual = DF_vd['Vol']
              mape = np.round(np.mean(abs_error/actual),6)*100
              print('smoothing_trend : ',np.round(i,3),'RMSE : ' ,np.round(rmse,4) ,'MAPE : ',mape)
```

```
smoothing_trend :  0.0 RMSE :  8.8522 MAPE :  34.605999999999995
smoothing_trend :  0.001 RMSE :  4.5094 MAPE :  16.9816
smoothing_trend :  0.002 RMSE :  1.9665 MAPE :  6.9962
smoothing_trend :  0.003 RMSE :  1.0538 MAPE :  3.6928
smoothing_trend :  0.004 RMSE :  1.0562 MAPE :  3.7711
smoothing_trend :  0.005 RMSE :  1.1334 MAPE :  4.1088
smoothing_trend :  0.006 RMSE :  1.1408 MAPE :  4.1418
smoothing_trend :  0.007 RMSE :  1.1112 MAPE :  4.023800000000005
smoothing_trend :  0.008 RMSE :  1.0733 MAPE :  3.8627000000000002
smoothing_trend :  0.009 RMSE :  1.0414 MAPE :  3.7122
smoothing_trend :  0.01 RMSE :  1.0211 MAPE :  3.6304000000000003
smoothing_trend :  0.011 RMSE :  1.014 MAPE :  3.6186000000000003
smoothing_trend :  0.012 RMSE :  1.0198 MAPE :  3.6216
smoothing_trend :  0.013 RMSE :  1.0373 MAPE :  3.6639999999999997
smoothing_trend :  0.014 RMSE :  1.0653 MAPE :  3.7499
smoothing_trend :  0.015 RMSE :  1.1025 MAPE :  3.8871999999999995
smoothing_trend :  0.016 RMSE :  1.1479 MAPE :  4.0671
smoothing_trend :  0.017 RMSE :  1.2003 MAPE :  4.266699999999999
smoothing_trend :  0.018 RMSE :  1.2589 MAPE :  4.4763
smoothing_trend :  0.019 RMSE :  1.3229 MAPE :  4.7012
smoothing_trend :  0.02 RMSE :  1.3916 MAPE :  4.9352
smoothing_trend :  0.021 RMSE :  1.4641 MAPE :  5.1817
smoothing_trend :  0.022 RMSE :  1.5398 MAPE :  5.437
smoothing_trend :  0.023 RMSE :  1.6181 MAPE :  5.705
smoothing_trend :  0.024 RMSE :  1.6982 MAPE :  5.985399999999999
smoothing_trend :  0.025 RMSE :  1.7795 MAPE :  6.2734
smoothing_trend :  0.026 RMSE :  1.8616 MAPE :  6.5639
smoothing_trend :  0.027 RMSE :  1.9438 MAPE :  6.864299999999999
smoothing_trend :  0.028 RMSE :  2.0257 MAPE :  7.173499999999999
smoothing_trend :  0.029 RMSE :  2.1068 MAPE :  7.479900000000001
```

Best Trend Smoothing parameter $(beta) = 0.011$

Fitting the Double Exponential Smoothing With $\alpha = 0.06$ and $\beta = 0.011$:

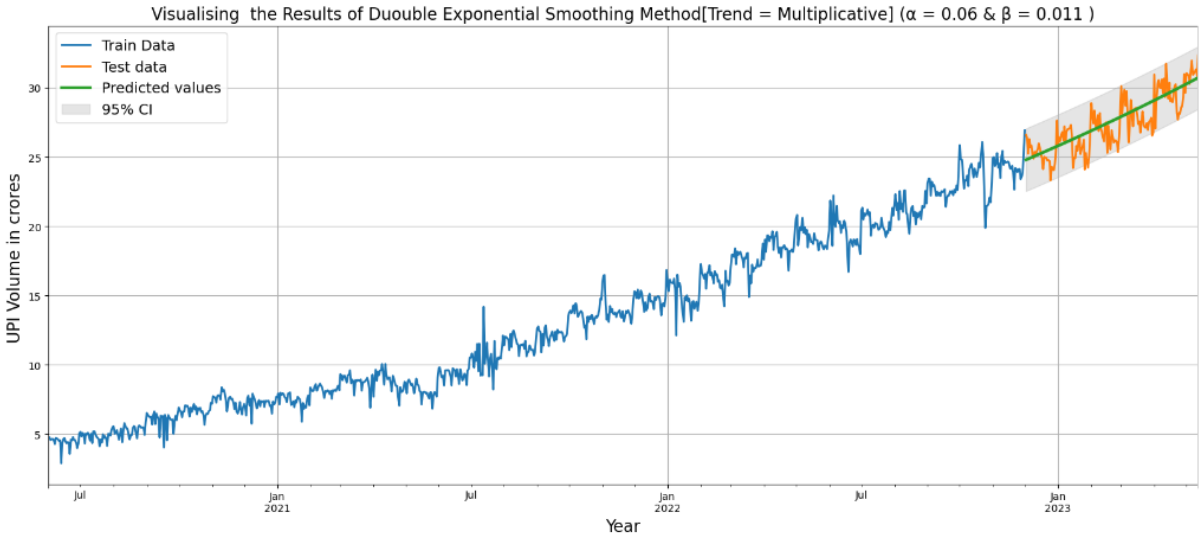
Model Summary:

ExponentialSmoothing Model Results

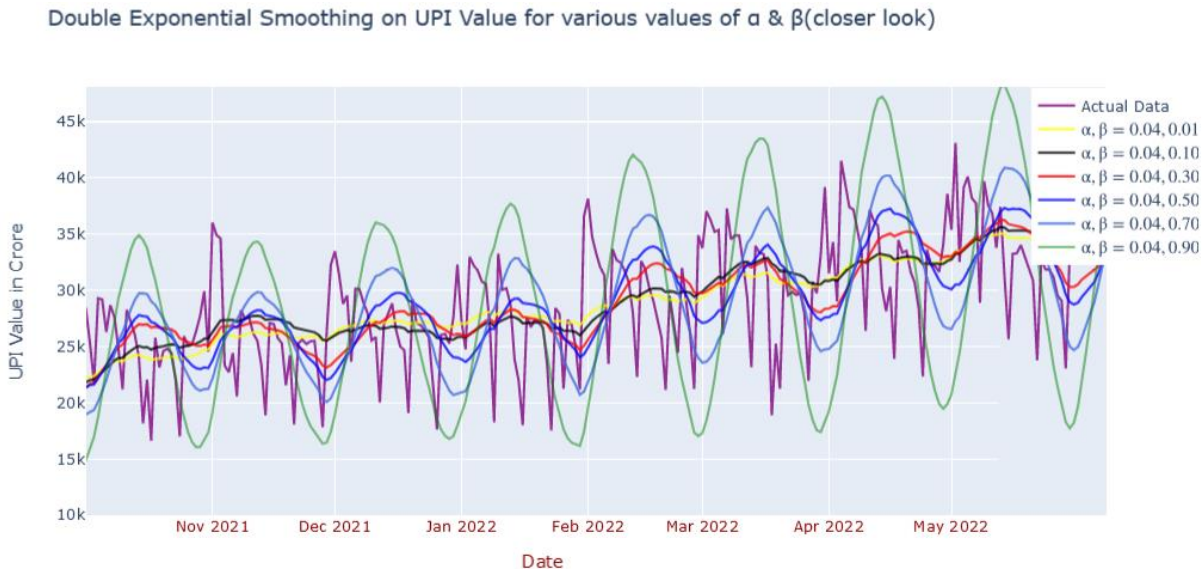
Dep. Variable:	endog	**No. Observations:**	914
Model:	ExponentialSmoothing	**SSE**	618.919
Optimized:	False	**AIC**	-348.329
Trend:	Multiplicative	**BIC**	-329.058
Seasonal:	None	**AICC**	-348.236
Seasonal Periods:	None	**Date:**	Sun, 30 Jul 2023
Box-Cox:	False	**Time:**	13:51:02
Box-Cox Coeff.:	None		

	coeff	code	optimized
smoothing_level	0.0600000	alpha	False
smoothing_trend	0.0110000	beta	False
initial_level	4.7063000	l.0	False
initial_trend	0.9967552	b.0	False

Now, let us visualise the results of DES on our Test data as follows:



- **DES on UPI Value Data**



Now, we have found out the best trend smoothing parameter (i.e., beta) using Grid Search method by minimizing the RMSE and MAPE and the results are as follows:

```
In [564]: for i in np.arange(0.0,0.1,0.005):
              model_fit = model_double_exp.fit(smoothing_level = 0.04, smoothing_slope = i, optimized = False)
              y_hat_avg = DF_vd.copy()
              y_hat_avg['DES'] = model_fit.forecast(len(DF_vd['Val']))
              rmse = np.sqrt(mean_squared_error(DF_vd.Val, y_hat_avg.DES))
              rmse = round(rmse, 4)
              abs_error = np.abs(DF_vd['Val']-y_hat_avg.DES)
              actual = DF_vd['Val']
              mape = np.round(np.mean(abs_error/actual),6)*100
              print('smoothing_trend : ',np.round(i,3),'RMSE : ' ,np.round(rmse,4) ,'MAPE : ',mape)
```

```
smoothing_trend :  0.0 RMSE :  14101.1985 MAPE :  30.921300000000002
smoothing_trend :  0.005 RMSE :  5458.8884 MAPE :  12.3498
smoothing_trend :  0.01 RMSE :  5464.5553 MAPE :  12.3743
smoothing_trend :  0.015 RMSE :  5477.7862 MAPE :  12.1664
smoothing_trend :  0.02 RMSE :  5565.7924 MAPE :  12.1304
smoothing_trend :  0.025 RMSE :  5752.7931 MAPE :  12.347800000000001
smoothing_trend :  0.03 RMSE :  6053.05 MAPE :  12.9604
smoothing_trend :  0.035 RMSE :  6451.8999 MAPE :  13.7019
smoothing_trend :  0.04 RMSE :  6909.9742 MAPE :  14.5484
smoothing_trend :  0.045 RMSE :  7382.9712 MAPE :  15.420800000000002
smoothing_trend :  0.05 RMSE :  7836.8117 MAPE :  16.2821
smoothing_trend :  0.055 RMSE :  8252.824 MAPE :  17.0957
smoothing_trend :  0.06 RMSE :  8627.1234 MAPE :  17.869799999999998
smoothing_trend :  0.065 RMSE :  8967.5709 MAPE :  18.5697
smoothing_trend :  0.07 RMSE :  9289.2804 MAPE :  19.261200000000002
smoothing_trend :  0.075 RMSE :  9609.339 MAPE :  19.9546
smoothing_trend :  0.08 RMSE :  9941.9222 MAPE :  20.701800000000002
smoothing_trend :  0.085 RMSE :  10294.983 MAPE :  21.5129
smoothing_trend :  0.09 RMSE :  10669.0924 MAPE :  22.3607
smoothing_trend :  0.095 RMSE :  11058.2731 MAPE :  23.2437
```

Best Trend Smoothing parameter $(beta) = 0.020$

Fitting the Double Exponential Smoothing model for UPI Value With $\alpha = 0.04 and \beta = 0.020$**:**

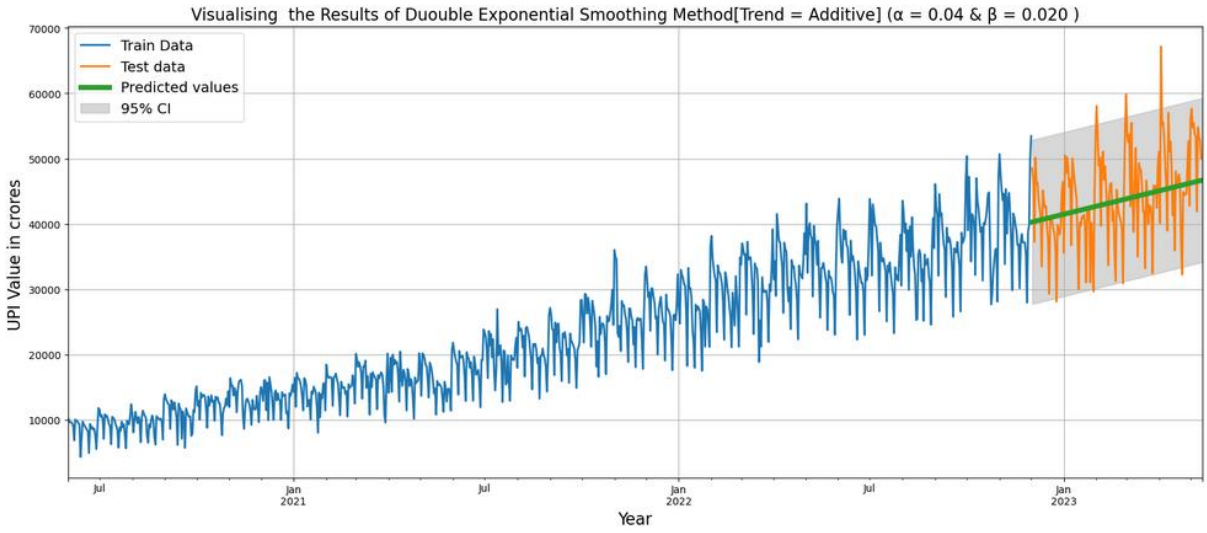
Model Summary:

ExponentialSmoothing Model Results

Dep. Variable:	endog	**No. Observations:**	914
Model:	ExponentialSmoothing	**SSE**	13856545483.458
Optimized:	False	**AIC**	15120.252
Trend:	Additive	**BIC**	15139.524
Seasonal:	None	**AICC**	15120.345
Seasonal Periods:	None	**Date:**	Mon, 31 Jul 2023
Box-Cox:	False	**Time:**	12:07:24
Box-Cox Coeff.:	None		

	coeff	code	optimized
smoothing_level	0.0400000	alpha	False
smoothing_trend	0.0200000	beta	False
initial_level	9912.4867	l.0	False
initial_trend	-77.753030	b.0	False

Now, let us visualise the results of DES on our UPI value Test data as follows:



Interpretation: Looking to the fitted values using the Double Exponential Smoothing and Comparing with the Simple Exponential Method, it can be concluded that for our time series data, Double Exponential Smoothing performs better, which is not as unlikely, this was going to happen, because we know that our time series contains trend, thus the Double Exponential Smoothing Method performs better than the Simple Exponential Method.

8.3.1.3. Triple Exponential Smoothing (TES) OR Holt and Winter's Exponential Method:

- It is used to handle the time series data containing trend as well as a seasonal component
- Double Exponential Smoothing will not work well in case of the data containing seasonal
- So, that for smoothing the seasonality a third equation is introduced

The Forecast Equation id given by:

$$f_0 = y_0$$

$$F_0 = y_0$$

$$b_0 = \frac{\sum_{i=0}^{l-1}(y_{l+i} - y_i)}{l^2}$$

$$f_t = \alpha * (y_t - C_{t\%l}) + (1 - \alpha) * (f_{t-1} + \varphi * b_{t-1})$$

$$b_t = \beta * (f_t - f_{t-1}) + (1 - \beta) * \varphi * b_{t-1}$$

$$C_{t\%l} = \gamma * (y_t - f_t) + (1 - \gamma) * C_{t\%l}$$

$$F_{t+m} = f_t + b_t * \sum_{i=1}^{m} \varphi^i + C_{t\%l}$$

Where;

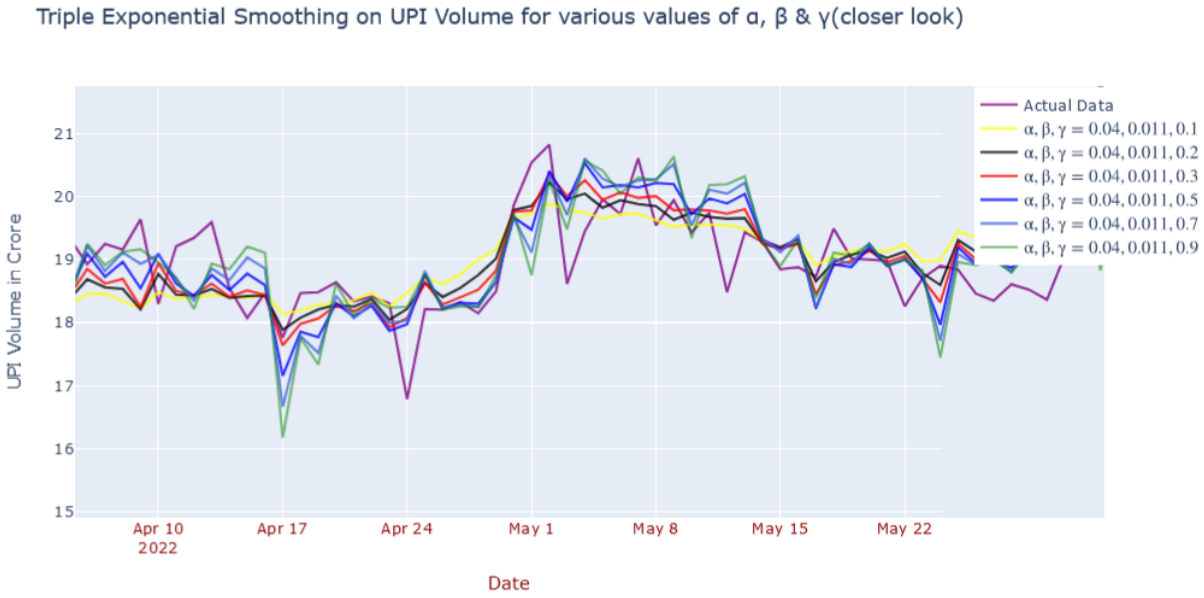
$$0 < \alpha, \beta, \gamma < 1$$

$$\gamma \text{ is called the Seasonal Smoothing } factor \text{ and}$$

$$\varphi \text{ is called the Damped Smoothing Factor such that } 0 < \varphi < 1$$

Applying Triple Exponential Smoothing OR Holt & Winter's Method to our Time series:

- **TES on UPI Volume data**



Now, we have found out the best Seasonal smoothing parameter (i.e., gamma) using Grid Search method by minimizing the RMSE and MAPE and the results are as follows:

```
In [420]: for i in np.arange(0.1,0.3,0.01):
              alpha = 0.06
              beta = 0.011
              gamma = i
              seasonal_period = 30  # Assuming monthly seasonality
              triple_exp_model = ExponentialSmoothing(DF_train['Vol'], trend='mul', seasonal='add', seasonal_periods=seasonal_period)
              triple_exp_fit = triple_exp_model.fit(smoothing_level=alpha, smoothing_slope=beta, smoothing_seasonal=gamma)
              y_hat_avg = DF_vd.copy()
              y_hat_avg['TES'] = triple_exp_fit.forecast(len(DF_vd['Vol']))
              rmse = np.sqrt(mean_squared_error(DF_vd.Vol, y_hat_avg.TES))
              rmse = round(rmse, 4)
              abs_error = np.abs(DF_vd['Vol']-y_hat_avg.TES)
              actual = DF_vd['Vol']
              mape = np.round(np.mean(abs_error/actual),6)*100
              print('smoothing_seasonal : ',np.round(i,3),'RMSE : ' ,np.round(rmse,4) ,'MAPE : ',mape)
```

```
smoothing_seasonal :  0.1 RMSE :  1.071 MAPE :  3.6809000000000003
smoothing_seasonal :  0.11 RMSE :  1.0673 MAPE :  3.6540000000000004
smoothing_seasonal :  0.12 RMSE :  1.0719 MAPE :  3.6754000000000002
smoothing_seasonal :  0.13 RMSE :  1.0677 MAPE :  3.6498000000000004
smoothing_seasonal :  0.14 RMSE :  1.0634 MAPE :  3.6192
smoothing_seasonal :  0.15 RMSE :  1.0682 MAPE :  3.6424
smoothing_seasonal :  0.16 RMSE :  1.0704 MAPE :  3.6454
smoothing_seasonal :  0.17 RMSE :  1.0619 MAPE :  3.5993999999999997
smoothing_seasonal :  0.18 RMSE :  1.0637 MAPE :  3.6043
smoothing_seasonal :  0.19 RMSE :  1.0753 MAPE :  3.653
smoothing_seasonal :  0.2 RMSE :  1.0753 MAPE :  3.6484
smoothing_seasonal :  0.21 RMSE :  1.0738 MAPE :  3.6408000000000005
smoothing_seasonal :  0.22 RMSE :  1.068 MAPE :  3.6126
smoothing_seasonal :  0.23 RMSE :  1.0609 MAPE :  3.5732
smoothing_seasonal :  0.24 RMSE :  1.0635 MAPE :  3.5864
smoothing_seasonal :  0.25 RMSE :  1.0701 MAPE :  3.6189
smoothing_seasonal :  0.26 RMSE :  1.0653 MAPE :  3.5949
smoothing_seasonal :  0.27 RMSE :  1.0659 MAPE :  3.5985000000000005
smoothing_seasonal :  0.28 RMSE :  1.0529 MAPE :  3.5324
smoothing_seasonal :  0.29 RMSE :  1.0662 MAPE :  3.6019
```

Best Seasonal Smoothing paraeter $(\gamma) = 0.17$

Fitting the Triple Exponential Smoothing With $\alpha = 0.06, \beta = 0.011, \& \gamma = 0.17$:

Model Summary:

ExponentialSmoothing Model Results

Dep. Variable:	Vol	**No. Observations:**	914
Model:	ExponentialSmoothing	**SSE**	499.746
Optimized:	True	**AIC**	-483.810
Trend:	Multiplicative	**BIC**	-320.004
Seasonal:	Additive	**AICC**	-480.772
Seasonal Periods:	30	**Date:**	Sun, 30 Jul 2023
Box-Cox:	False	**Time:**	16:43:39
Box-Cox Coeff.:	None		

	coeff	code	optimized
smoothing_level	0.0600000	alpha	False
smoothing_trend	0.0110000	beta	False
smoothing_seasonal	0.1700000	gamma	False
initial_level	40.519760	l.0	True
initial_trend	1.0003677	b.0	True
initial_seasons.0	-36.315807	s.0	True
initial_seasons.1	-36.293151	s.1	True
initial_seasons.2	-36.104277	s.2	True
initial_seasons.3	-36.020553	s.3	True
initial_seasons.4	-36.126398	s.4	True
initial_seasons.5	-36.058939	s.5	True
initial_seasons.6	-36.049058	s.6	True
initial_seasons.7	-35.959360	s.7	True
initial_seasons.8	-35.858183	s.8	True
initial_seasons.9	-35.982149	s.9	True
initial_seasons.10	-36.048361	s.10	True
initial_seasons.11	-35.955886	s.11	True
initial_seasons.12	-36.191944	s.12	True
initial_seasons.13	-36.008521	s.13	True
initial_seasons.14	-36.157110	s.14	True
initial_seasons.15	-36.103189	s.15	True
initial_seasons.16	-36.286712	s.16	True
initial_seasons.17	-36.382076	s.17	True
initial_seasons.18	-36.703091	s.18	True
initial_seasons.19	-36.351226	s.19	True
initial_seasons.20	-36.547540	s.20	True
initial_seasons.21	-36.486156	s.21	True
initial_seasons.22	-36.442283	s.22	True
initial_seasons.23	-36.491194	s.23	True
initial_seasons.24	-36.489947	s.24	True
initial_seasons.25	-36.612961	s.25	True
initial_seasons.26	-36.636059	s.26	True
initial_seasons.27	-36.850512	s.27	True
initial_seasons.28	-36.669573	s.28	True
initial_seasons.29	-36.459876	s.29	True

NOTE: Now, since we already know that our UPI Volume time series data have Trend & some seasonal pattern and we know that, Triple Exponential Smoothing method considers both Trend and the Seasonality, therefore, we are going to take the TES model for model diagnostics on residuals and the results are as follows:

9. Model Diagnostics

- **Checking for Auto-correlation in the residuals Using the Ljung-Box test:**

To test for autocorrelation in residuals, we can use various statistical tests that specifically check for the presence of autocorrelation in the residual series. One of the commonly used tests is the **Ljung-Box test,** which is described as follows:

The Ljung-Box Test: The Ljung-Box test is used to assess the presence of autocorrelation in a time series or the residuals of a time series model. The test determines whether or not residuals are iid (i.e., white noise) or whether there is something more behind them; whether or not the autocorrelations for the errors are residuals are non-zero. Essentially, it is a test of lack of fit: if the autocorrelations of the residuals are very small, we say that model does not show 'significant lack of fit.'

Null and Alternative Hypothesis:

- $\boldsymbol{Null\ Hypothesis\ (H_0)}$ **: The residuals are independently distributed vs**
- $\boldsymbol{Alternative\ Hypothesis\ (H_1)}$**: The residuals are not independently distributed; they exhibit serial correlation.**

Ideally, we would like to fail to reject the null hypothesis. That is, we would like to see the p-value of the test be greater than 0.05 because this means the residuals for our time series model are independent, which is often an assumption we make when creating a model.

The test statistic is given by:

$$\boldsymbol{Q = n(n+2)\sum_{k=1}^{h}\frac{\widehat{\rho_k}^2}{(n-k)}}$$

Where, n is the sample size, $\widehat{\rho_k}$ is the sample autocorrelation at lag k, and h is the number of lags being tested. Under H_0 the statistic Q asymptotically follows a $\chi^2_{(h)}$.Note: A higher test statistic value indicates stronger evidence of autocorrelation at the lag.

Decision Rule:

- The p-value measures the probability of obtaining the observed test statistics under the assumption that the null hypothesis (no autocorrelation) is true.
- A small p-value (0.05 or less) => Reject H0 => The residuals is NOT independent i.e.; they exhibit serial correlation.

Conversely, A large p-value (more than 0.05) => Fail to reject H0 => The residuals are uncorrelated i.e., they are independent.

Performing the Ljung-Box test on the residuals of the TES model for UPI Volume:

```
In [504]: # Performing the Ljung-Box test on residuals of TES model
          import statsmodels.api as sm
          lags = 5  # Number of lags to test
          results2 = sm.stats.acorr_ljungbox(DF_train_new['error_tes'], lags=lags)
          print(results2)
```

```
      lb_stat     lb_pvalue
1  135.839084  2.163740e-31
2  180.984097  5.009595e-40
3  201.284836  2.226079e-43
4  209.543785  3.330547e-44
5  210.325318  1.752966e-43
```

Interpretation:

- For lags 1 and 2, the p-values are all less than 0.05. This suggests that there is significant evidence of autocorrelation at these lags, and the null hypothesis of no autocorrelation is rejected.
- For lags 3 & 5, the p-values are still less than 0.05, indicating that there is significant evidence of autocorrelation at these lags as well.
- For lag 5, the p-value is once again less than 0.05, confirming that there is significant evidence of autocorrelation at this lag.

In summary, based on the provided results, we do have significant evidence to reject the null hypothesis of no autocorrelation in the data for any of the lags tested, THUS, WE CAN SAY THAT RESIDUALS OF TES MODEL FOR UPI VALUE IS AUTOCORRELATED.

NOTE: Since, our residuals are autocorrelated, therefore the residuals are NOT White Noise, thus, the assumption of Time Series Analysis gets Violated, therefore, we must model the residuals also, So, we have modelled the residuals also as follows:

MODELLING THE RESIDUALS OF TES model for UPI Volume data:

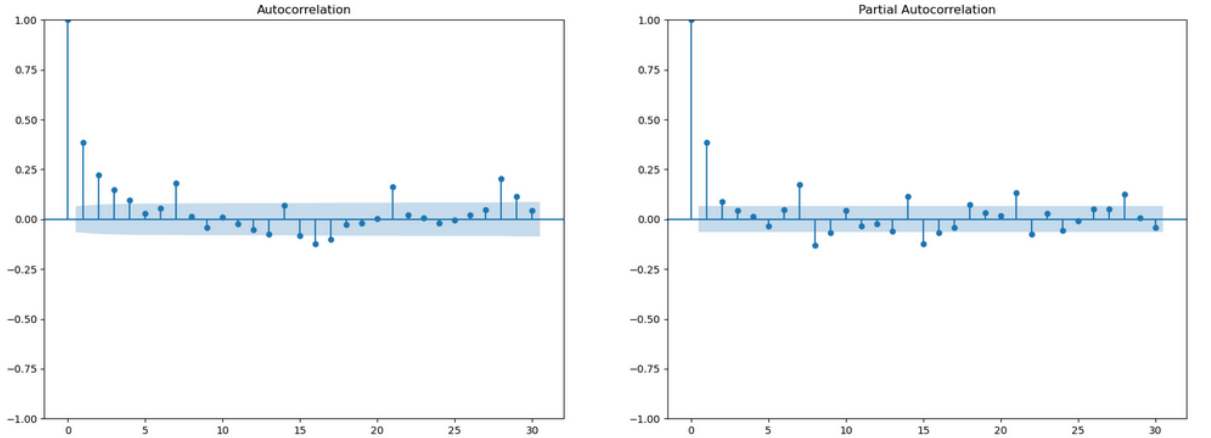
Checking for stationarity of residuals:

```
In [478]: ## test for stationary on the Residuals of the TES model for UPI Volume
          import statsmodels.tsa.stattools as sts
          sts.adfuller(DF_train_new['error_tes'])

Out[478]: (-5.868522194681913,
           3.281856980969873e-07,
           21,
           892,
           {'1%': -3.4377022625762232,
            '5%': -2.8647856243940817,
            '10%': -2.568498194061815},
           1785.2455916729566)
```

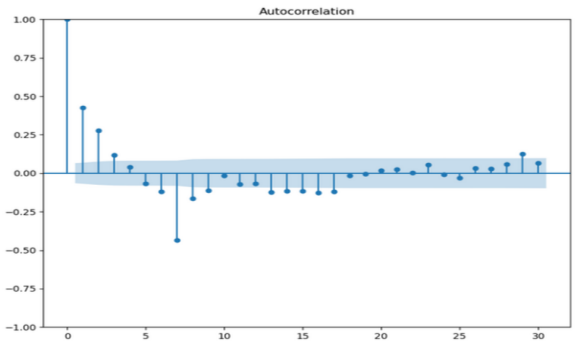
Interpretation: Now one can see that the first value which denotes the value of the calculated t-statistics which is -5.86852219 and which is less than all critical values of the t-distribution at each significance level viz. (1%,5%, and 10%) at 892 degrees of freedom, thus this time our Null hypothesis get Rejected i.e., ϕ(B) does not contain any unit root, thus our residuals are Stationary. Also, from the p-value, which is approximately zero, thus there is a ZERO probability that our **residuals are non-stationary in nature.**

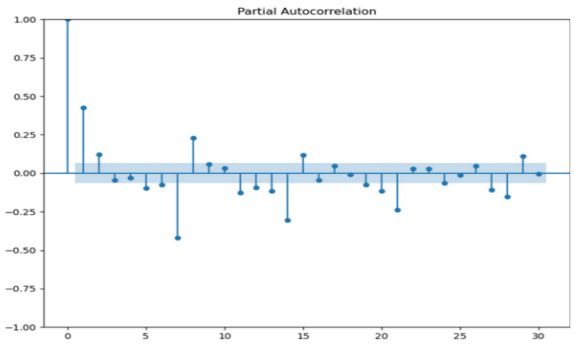
ACF AND PACF plot for identifying (p,d,q) of the ARIMA model for residual series:



Conclusion: From the above ACF and PACF plot, one can conclude that the parameters there is a seasonality pattern and the period of seasonality is around 7 days in the residuals of the TES model for UPI Volume, so our next aim is to model the residuals as a SARIMA model as follows:

Identifying the parameters of SARIMA model for Residuals using the ACF and PACF of the 7^{th} lagged differenced data as follows:





Thus, From the above plots of ACF and PACF we conclude:

- **p = 2**
- **d = 0 (as the residuals are already stationary)**
- **q = 3**
- **P = 0**
- **D = 1**
- **Q = 1**
- **S = 7**

Modelling the residuals as SARIMA (2,0,3)(0,1,1)[7]:

SARIMAX Results

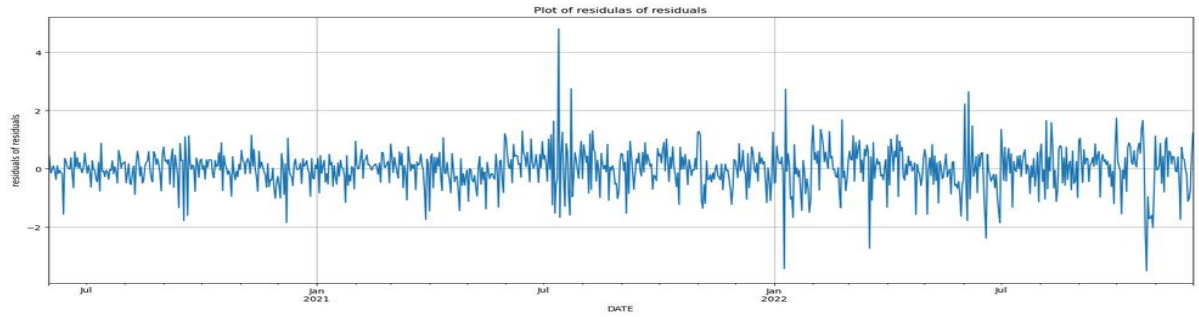
Dep. Variable:	error_tes	**No. Observations:**	914
Model:	SARIMAX(2, 0, 3)x(0, 1, [1], 7)	**Log Likelihood**	-868.323
Date:	Sun, 06 Aug 2023	**AIC**	1750.647
Time:	20:38:56	**BIC**	1784.318
Sample:	06-01-2020	**HQIC**	1763.504
	- 12-01-2022		
Covariance Type:	opg		

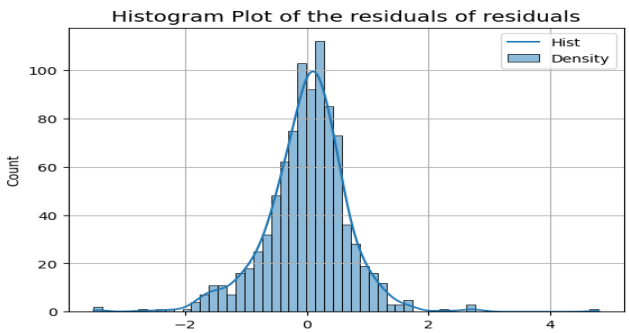
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.1954	0.135	-1.449	0.147	-0.460	0.069
ar.L2	0.6499	0.090	7.201	0.000	0.473	0.827
ma.L1	0.5933	0.132	4.478	0.000	0.334	0.853
ma.L2	-0.3258	0.097	-3.355	0.001	-0.516	-0.135
ma.L3	-0.0594	0.043	-1.373	0.170	-0.144	0.025
ma.S.L7	-0.9570	0.012	-82.119	0.000	-0.980	-0.934
sigma2	0.3896	0.009	42.187	0.000	0.372	0.408

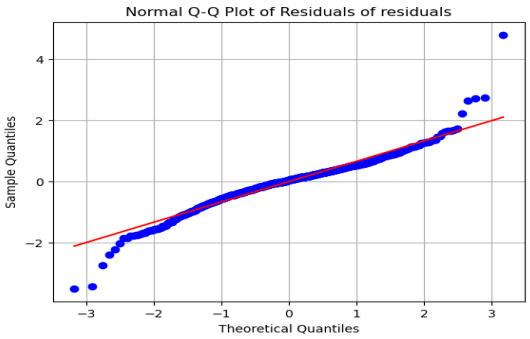
Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	2341.96
Prob(Q):	0.97	**Prob(JB):**	0.00
Heteroskedasticity (H):	3.66	**Skew:**	-0.05
Prob(H) (two-sided):	0.00	**Kurtosis:**	10.87

Residual Analysis of residuals of SARIMA model for Residuals

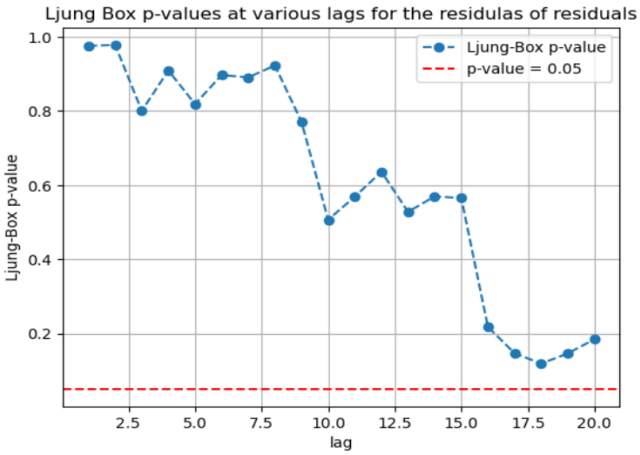
Homoscedasticity and Normality of residuals of residuals:

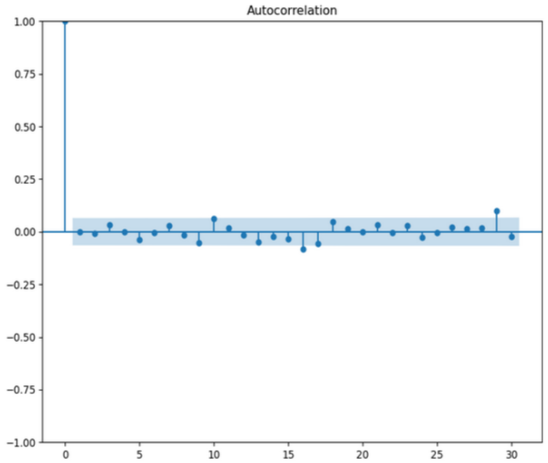






Breusch Pagan Test for Homoscedasticity of residuals of residuals:






```
In [644]: # Perform the Breusch-Pagan test
_, p_value, _, _ = het_breuschpagan(err2, Df_train_new)

# Test the null hypothesis at a certain significance level (e.g., 0.05)
alpha = 0.05
if p_value < alpha:
    print("The residuals are likely heteroscedastic (reject null hypothesis).")
else:
    print("The residuals are likely homoscedastic (fail to reject null hypothesis).")

The residuals are likely homoscedastic (fail to reject null hypothesis).
```

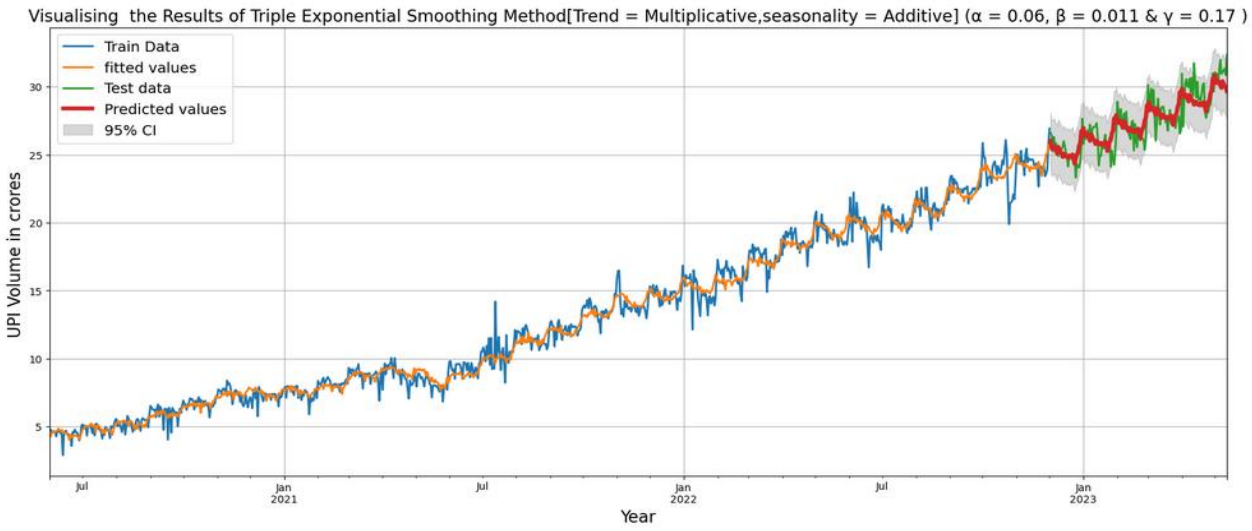
Now, from the above TES model for UPI Volume diagnostics, we infer that residuals of residuals and hence the actual residuals are:

- **Independent (i.e., No Autocorrelation)**
- **Homoscedastic in nature (i.e., constant Variance)**
- **Residuals are approximately Normally distributed**
  - ⇨ **Residuals follows White Noise Process**

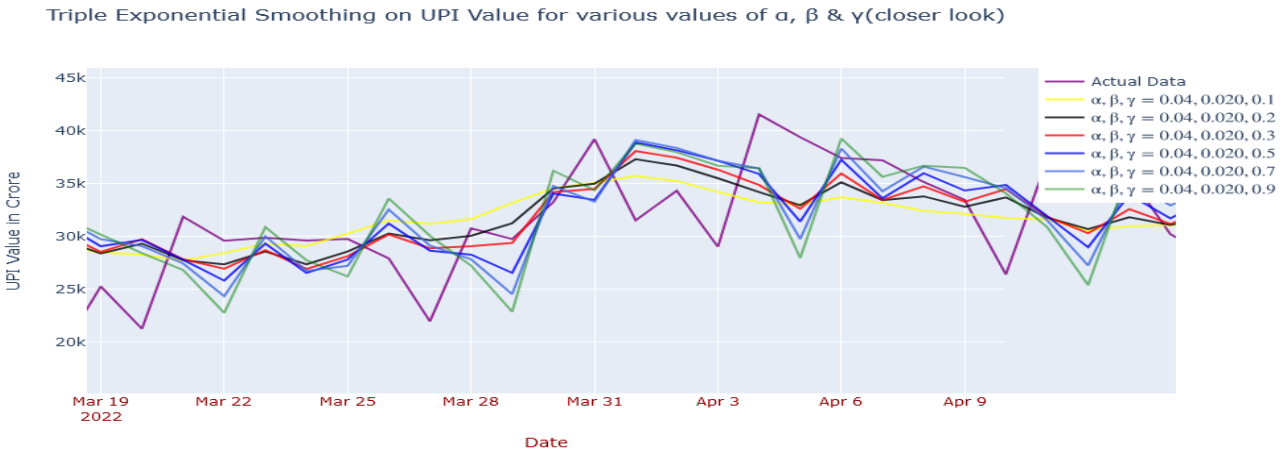
Thus, the Assumption of Time series modelling is achieved, therefore we are ready to test our model performance on our test data, as follows:

Now, We, have added the predicted error to the predicted values through the TES model to get the final predicted values for the test data, and the results are depicted as follows:

Now, let us visualise the results of 'TES + SARIMA (2,0,3)(0,1,1)[7]: [for residuals]' on our UPI Volume Test data as follows:



- **TES on UPI Value data**



Grid-Search to find the best Seasonal Smoothing parameter (γ) for UPI Value data:

In [1017]:
```
for i in np.arange(0.3,0.5,0.01):
    alpha = 0.04
    beta = 0.020
    gamma = i
    seasonal_period = 30  # Assuming monthly seasonality
    triple_exp_model = ExponentialSmoothing(DF_train['Val'], trend='add', seasonal='mul', seasonal_periods=seasonal_period)
    triple_exp_fit = triple_exp_model.fit(smoothing_level=alpha, smoothing_slope=beta, smoothing_seasonal=gamma)
    y_hat_avg = DF_vd.copy()
    y_hat_avg['TES'] = triple_exp_fit.forecast(len(DF_vd['Val']))
    rmse = np.sqrt(mean_squared_error(DF_vd.Val, y_hat_avg.TES))
    rmse = round(rmse, 4)
    abs_error = np.abs(DF_vd['Val']-y_hat_avg.TES)
    actual = DF_vd['Val']
    mape = np.round(np.mean(abs_error/actual),6)*100
    print('smoothing_seasonal : ',np.round(i,3),'RMSE : ' ,np.round(rmse,4) ,'MAPE : ',mape)
```

```
smoothing_seasonal :  0.3 RMSE :  5380.3231 MAPE :  11.8302
smoothing_seasonal :  0.31 RMSE :  5379.3876 MAPE :  11.821900000000001
smoothing_seasonal :  0.32 RMSE :  5379.6498 MAPE :  11.8159
smoothing_seasonal :  0.33 RMSE :  5380.9878 MAPE :  11.8111
smoothing_seasonal :  0.34 RMSE :  5383.397 MAPE :  11.8076
smoothing_seasonal :  0.35 RMSE :  5386.8224 MAPE :  11.8051
smoothing_seasonal :  0.36 RMSE :  5391.2206 MAPE :  11.8042
smoothing_seasonal :  0.37 RMSE :  5396.5556 MAPE :  11.8107
smoothing_seasonal :  0.38 RMSE :  5402.792 MAPE :  11.8218
smoothing_seasonal :  0.39 RMSE :  5409.8976 MAPE :  11.833499999999999
smoothing_seasonal :  0.4 RMSE :  5417.8429 MAPE :  11.8469
smoothing_seasonal :  0.41 RMSE :  5426.5995 MAPE :  11.860999999999999
smoothing_seasonal :  0.42 RMSE :  5436.1425 MAPE :  11.8824
smoothing_seasonal :  0.43 RMSE :  5446.4488 MAPE :  11.906600000000001
smoothing_seasonal :  0.44 RMSE :  5457.4966 MAPE :  11.9327
smoothing_seasonal :  0.45 RMSE :  5469.2665 MAPE :  11.9601
smoothing_seasonal :  0.46 RMSE :  5481.7404 MAPE :  11.9892
smoothing_seasonal :  0.47 RMSE :  5494.902 MAPE :  12.0185
smoothing_seasonal :  0.48 RMSE :  5508.7364 MAPE :  12.0479
smoothing_seasonal :  0.49 RMSE :  5523.23 MAPE :  12.0816
```

Best Seasonal Smoothing paraeter $(\gamma) = 0.36$

Fitting Triple Exponential Smoothing on Value With $\alpha = 0.04, \beta = 0.010, \& \gamma = 0.36$:

Model Summary:

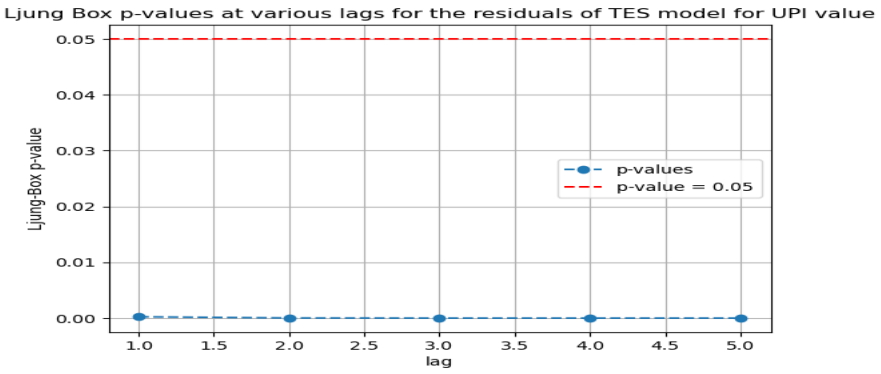
ExponentialSmoothing Model Results

Dep. Variable:	Val	**No. Observations:**	914
Model:	ExponentialSmoothing	**SSE**	12357899385.435
Optimized:	True	**AIC**	15075.634
Trend:	Additive	**BIC**	15239.440
Seasonal:	Multiplicative	**AICC**	15078.672
Seasonal Periods:	30	**Date:**	Mon, 31 Jul 2023
Box-Cox:	False	**Time:**	19:23:48
Box-Cox Coeff.:	None		

	coeff	code	optimized
smoothing_level	0.0400000	alpha	False
smoothing_trend	0.0200000	beta	False
smoothing_seasonal	0.3600000	gamma	False
initial_level	8715.9501	l.0	True
initial_trend	23.995401	b.0	True
initial_seasons.0	1.1484940	s.0	True
initial_seasons.1	1.2225191	s.1	True
initial_seasons.2	1.1727067	s.2	True
initial_seasons.3	1.1462284	s.3	True
initial_seasons.4	1.1056012	s.4	True
initial_seasons.5	1.1290193	s.5	True
initial_seasons.6	1.1290603	s.6	True
initial_seasons.7	1.1335029	s.7	True
initial_seasons.8	1.1463368	s.8	True
initial_seasons.9	1.0792843	s.9	True
initial_seasons.10	1.0743836	s.10	True
initial_seasons.11	1.0431205	s.11	True
initial_seasons.12	1.0315199	s.12	True
initial_seasons.13	1.0773474	s.13	True
initial_seasons.14	1.0443210	s.14	True
initial_seasons.15	1.0385860	s.15	True
initial_seasons.16	0.9787845	s.16	True
initial_seasons.17	0.9863016	s.17	True
initial_seasons.18	0.8558838	s.18	True
initial_seasons.19	0.9921724	s.19	True
initial_seasons.20	0.8700936	s.20	True
initial_seasons.21	0.9721424	s.21	True
initial_seasons.22	0.9492710	s.22	True
initial_seasons.23	0.9494505	s.23	True
initial_seasons.24	0.9537260	s.24	True
initial_seasons.25	0.8887440	s.25	True
initial_seasons.26	0.9690881	s.26	True
initial_seasons.27	0.8232038	s.27	True
initial_seasons.28	0.9409395	s.28	True
initial_seasons.29	1.0981950	s.29	True

NOTE: Again, we already know that our UPI Value time series data have Trend & some seasonal pattern and we know that, Triple Exponential Smoothing method considers both Trend and the Seasonality, therefore, we are going to take this TES model for model diagnostics on residuals and the results are as follows:

- **Checking for Auto-correlation in the residuals of TES model Using the Ljung-Box test:**



Interpretation: From the above plot, it is very clear that all the p-values for Ljung-Box test are less than 0.05, thus there is sufficient evidence for rejecting the null hypothesis of No Autocorrelation in the residuals, thus, we can say that our residuals of TES model for UPI Value data are NOT independent i.e., they are serially correlated, hence the assumption of the Time series modelling gets Violated, thus we can NOT use this model for forecasting until we can model the residuals also, therefore our next aim is to model the residuals of this model as follows:

MODELLING THE RESIDUALS OF TES model for UPI Value data:

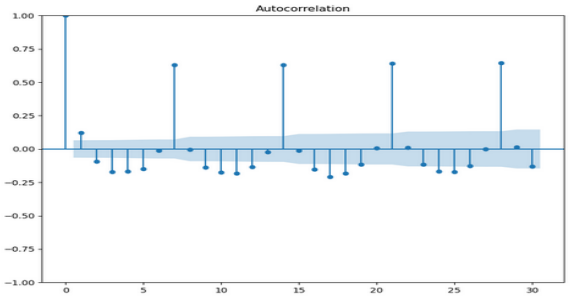
Checking for stationarity in the residual's series:

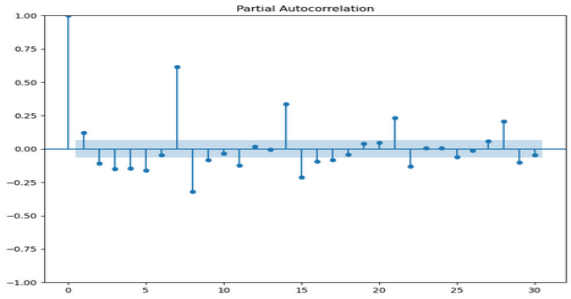
```
In [864]: ## test for stationary on the Residuals of the TES model for UPI Value
          import statsmodels.tsa.stattools as sts
          sts.adfuller(DF_train_new['error_tes'])

Out[864]: (-6.049787391271769,
           1.2857936031151327e-07,
           21,
           892,
           {'1%': -3.4377022625762232,
            '5%': -2.8647856243940817,
            '10%': -2.568498194061815},
           16382.838981632694)
```

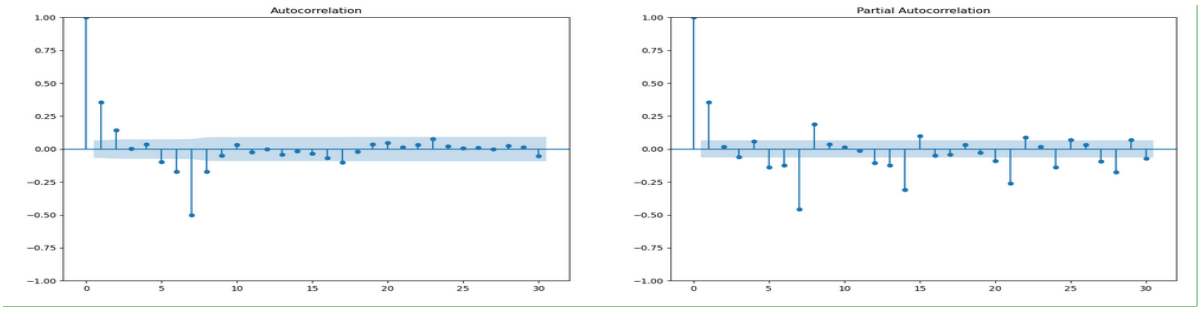
Interpretation: Now one can see that the first value which denotes the value of the calculated t-statistics which is -6.049787 and which is less than all critical values of the t-distribution at each significance level viz. (1%,5%, and 10%) at 892 degrees of freedom, thus this time our Null hypothesis get Rejected i.e., ϕ(B) does not contain any unit root, thus our residuals are Stationary. Also, from the p-value, which is approximately zero, thus there is a ZERO probability that our **residuals are non-stationary in nature.**

ACF AND PACF plot for identifying (p,d,q) of the ARIMA model for residual series:





Observation: From the above ACF plot of the residuals, we have seen that, the residuals have sort of seasonality pattern and the period of the seasonality is about 7 days, therefore, we are going to model the residuals by a SARIMA model, and to identify the parameters of SARIMA model, we looked the following ACF and PACF plot of the 7th lag difference series of the residuals as follows:



Interpretating the above ACF and PACF plot of the seasonal difference error

- p = 1
- d = 0
- q = 2
- P = 0
- D = 1
- Q = 1
- s = 7

hence the required SARIMA model for the residuals is : $SARIMA(1,0,2)[0,1,1,7]$

Modelling the residuals of TES model for UPI Value as SARIMA (1,0,2)(0,1,1)[7] :

In [886]:
```
### Model Building for residuals
model_error_ar1 = SARIMAX(DF_train_new['error_tes_std'], order = (1,0,2),seasonal_order = (0, 1, 1, 7))
model_fit_x = model_error_ar1.fit()
model_fit_x.summary()
```

Out[886]:

SARIMAX Results

Dep. Variable:	error_tes_std	**No. Observations:**	914
Model:	SARIMAX(1, 0, 2)x(0, 1, [1], 7)	**Log Likelihood**	-793.165
Date:	Mon, 31 Jul 2023	**AIC**	1596.330
Time:	15:58:12	**BIC**	1620.381
Sample:	06-01-2020	**HQIC**	1605.514
	- 12-01-2022		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.7314	0.108	-6.759	0.000	-0.943	-0.519
ma.L1	1.0910	0.107	10.222	0.000	0.882	1.300
ma.L2	0.3356	0.033	10.101	0.000	0.270	0.401
ma.S.L7	-0.9085	0.014	-64.643	0.000	-0.936	-0.881
sigma2	0.3319	0.009	38.313	0.000	0.315	0.349

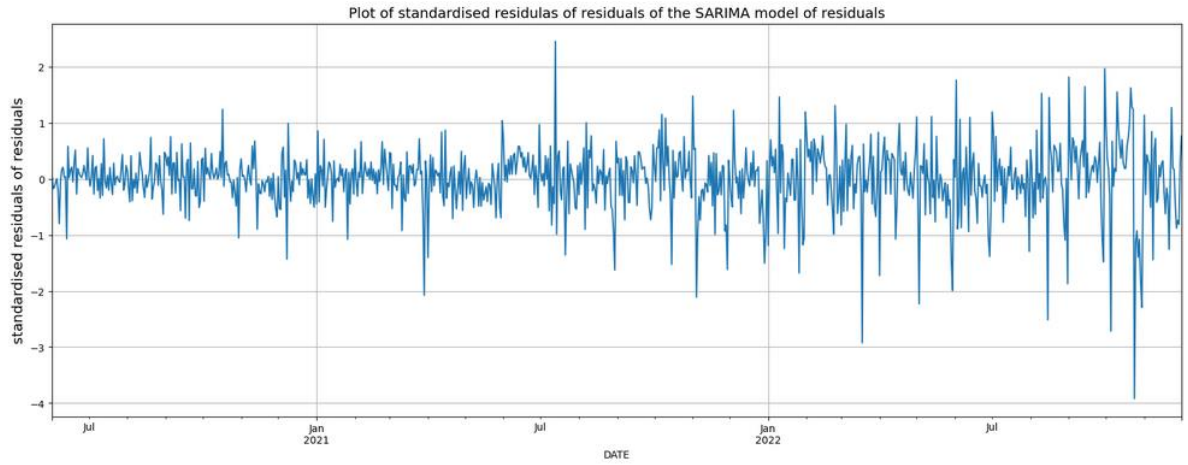
Ljung-Box (L1) (Q):	0.10	**Jarque-Bera (JB):**	1202.93
Prob(Q):	0.75	**Prob(JB):**	0.00
Heteroskedasticity (H):	4.16	**Skew:**	-0.95
Prob(H) (two-sided):	0.00	**Kurtosis:**	8.31

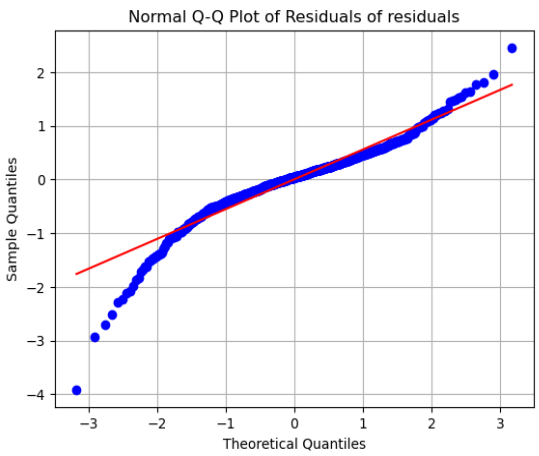
Predicting the Residuals through SARIMA (1,0,2)(0,1,1)[7]:

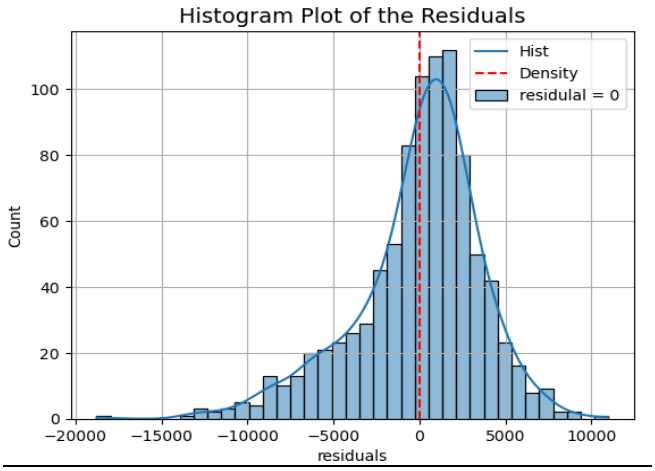
In [948]:
```
DF_test.predicted_error_val
```

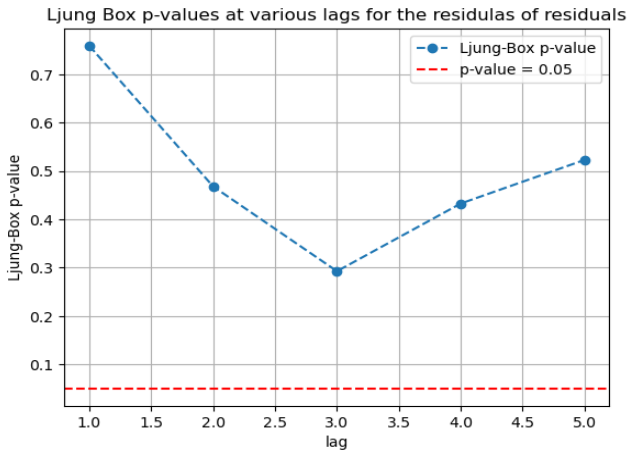
Out[948]:
```
DATE
2022-12-02    1013.260710
2022-12-03    -127.346339
2022-12-04   -9185.350490
2022-12-05    1562.553684
2022-12-06    3026.131989
                  ...
2023-05-08    1523.758437
2023-05-09    3054.507080
2023-05-10     470.845479
2023-05-11    3041.386303
2023-05-12     -71.415452
Name: predicted_error_val, Length: 162, dtype: float64
```

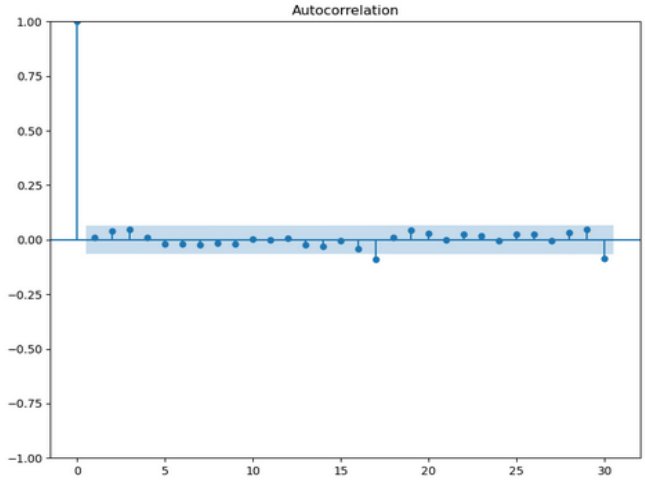
Diagnostics of residuals of residuals of the TES model for UPI Value data:











In [969]:
```
# Perform the Breusch-Pagan test on residuals of residulas to test for Homoscadisticity
_, p_value, _, _ = het_breuschpagan(DF_train_new['error_tes'], Df_train_new)

# Test the null hypothesis at a certain significance level (e.g., 0.05)
alpha = 0.05
if p_value < alpha:
    print("The residuals are likely heteroscedastic (reject null hypothesis).")
else:
    print("The residuals are likely homoscedastic (fail to reject null hypothesis).")
```

```
The residuals are likely homoscedastic (fail to reject null hypothesis).
```

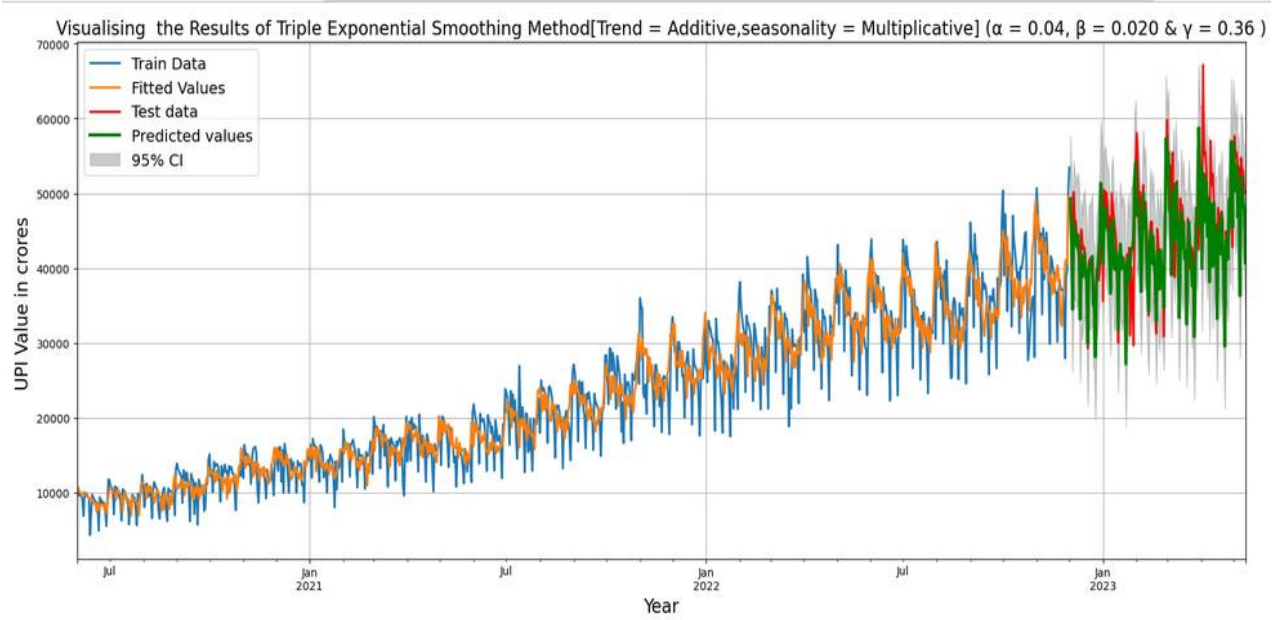
Now, from the above TES model for UPI Value diagnostics, we infer that residuals of residuals and hence the actual residuals are:

- **Independent (i.e., No Autocorrelation)**
- **Homoscedastic in nature (i.e., constant Variance)**
- **Residuals are approximately Normally distributed**
  - ⇨ **Residuals follows White Noise Process**

Thus, the Assumption of Time series modelling is achieved, therefore we are ready to test our model performance on our test data, as follows:

Now, We, have added the predicted error to the predicted values through the TES model to get the final predicted values for the test data, and the results are depicted as follows:

Now, let us visualise the results of 'TES + SARIMA(1,0,2)(0,1,1)[7]: [for residuals]' on our UPI Volume Test data as follows:



10. Model Adequacy and Performance on the Test set

Evaluating the Models Adequacy based on Evaluating Metric (RMSE, MAE and MAPE):

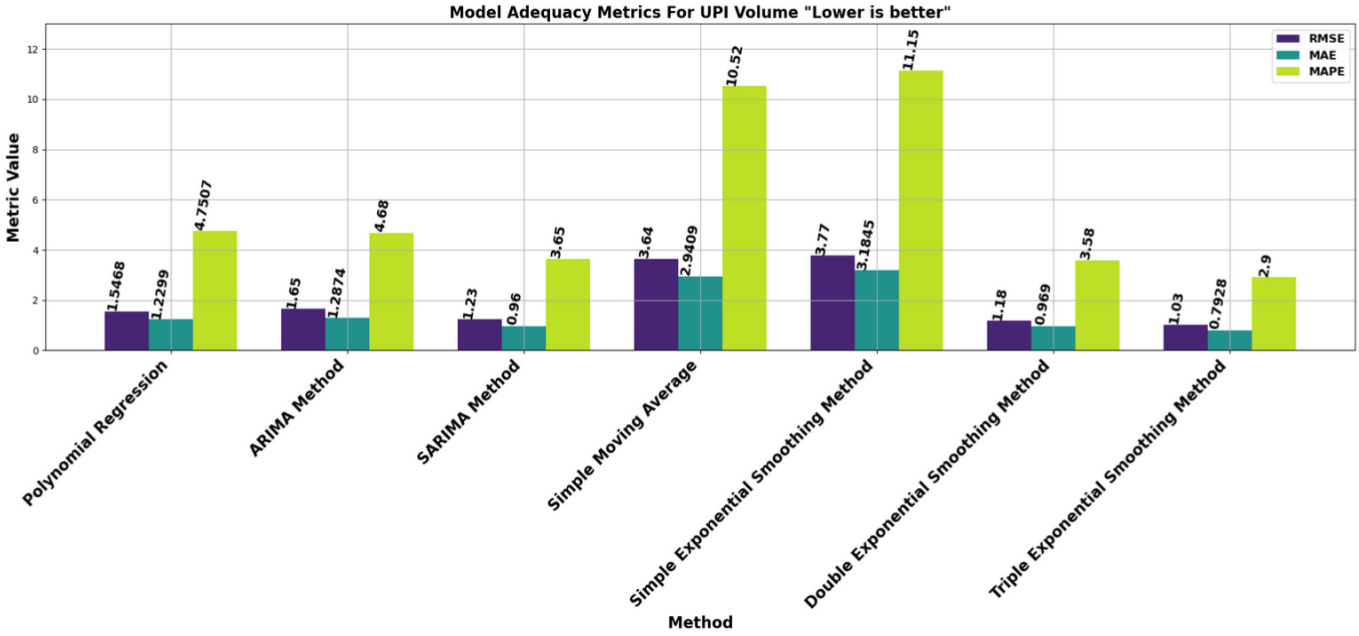
- $\boldsymbol{Root\ Mean\ Square\ Error}: \boldsymbol{RMSE} = \sqrt{\sum_{t=1}^{n} \frac{(y_t - \widehat{y_t})^2}{n}}$
- $\boldsymbol{Mean\ Absolute\ Error}: \boldsymbol{MAE} = \frac{\sum_{t=1}^{n} |y_t - \widehat{y_t}|}{\boldsymbol{n}}$

- $\boldsymbol{Mean\ Absolute\ Percentage\ Error: MAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|y_t - \hat{y}_t|}{y_t}}$

Where;

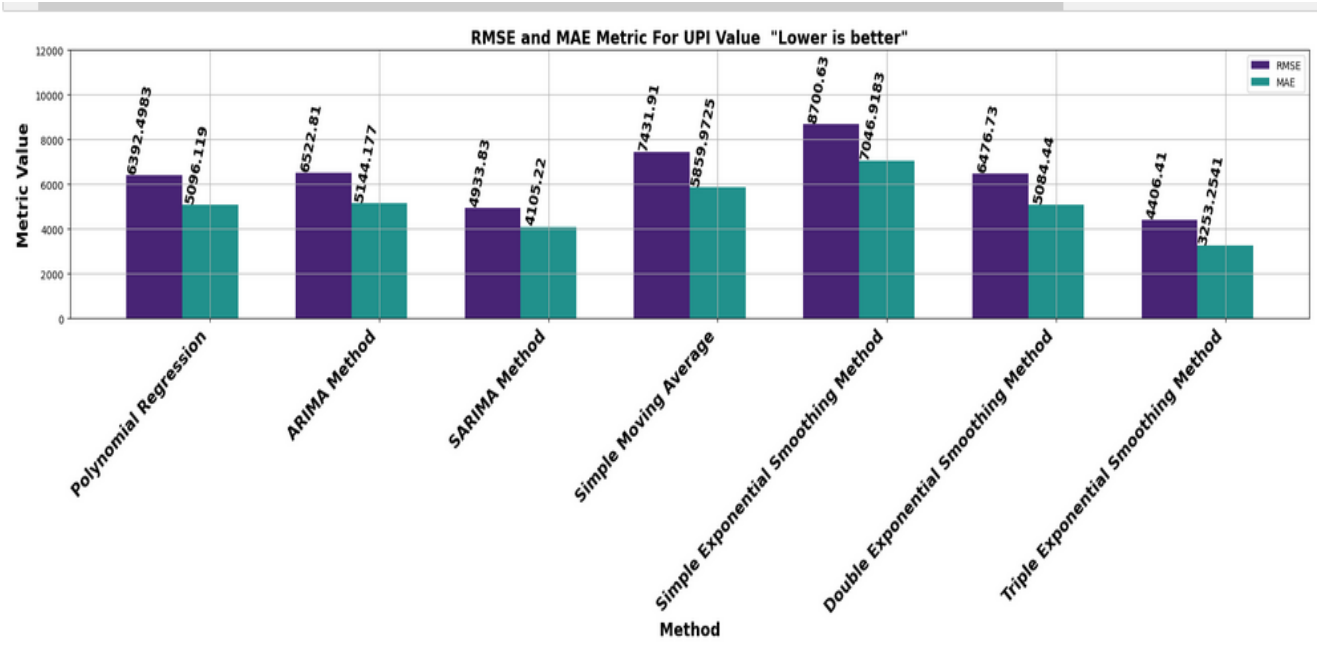
- $\boldsymbol{y_t: Actual\ Value\ of\ the\ series}$
- $\boldsymbol{\hat{y}_t: Predicted\ Value\ of\ the\ series\ and}$
- $\boldsymbol{n: is\ the\ total\ number\ of\ observation}$

- **Model Adequacy Checking of fitted Models for UPI Volume Series:**

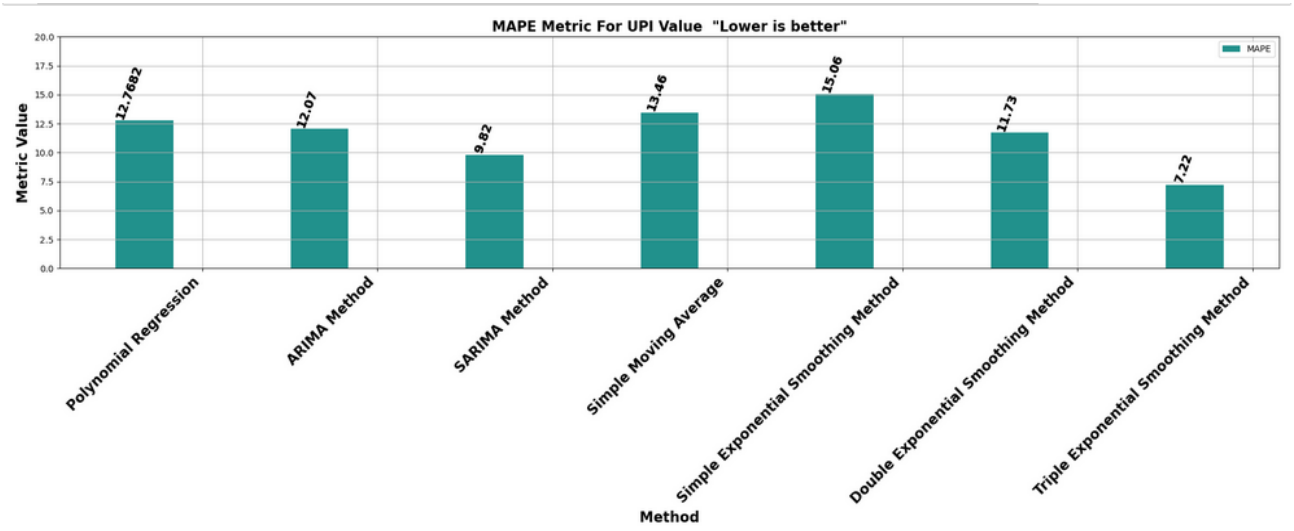


Comment: From the above bar plot showing the values of each model adequacy metrics for each of the fitted model, and from the plot we conclude that '**Triple Exponential Smoothing'** is the best model, while the second-best id Double Exponential and the 3rd best the ***SARIMA*** model, therefore for forecasting purpose, we will use the Triple Exponential Smoothing model to predict the future values of UPI Volume.

- **Model Adequacy Checking of fitted Models for UPI Value Series:**



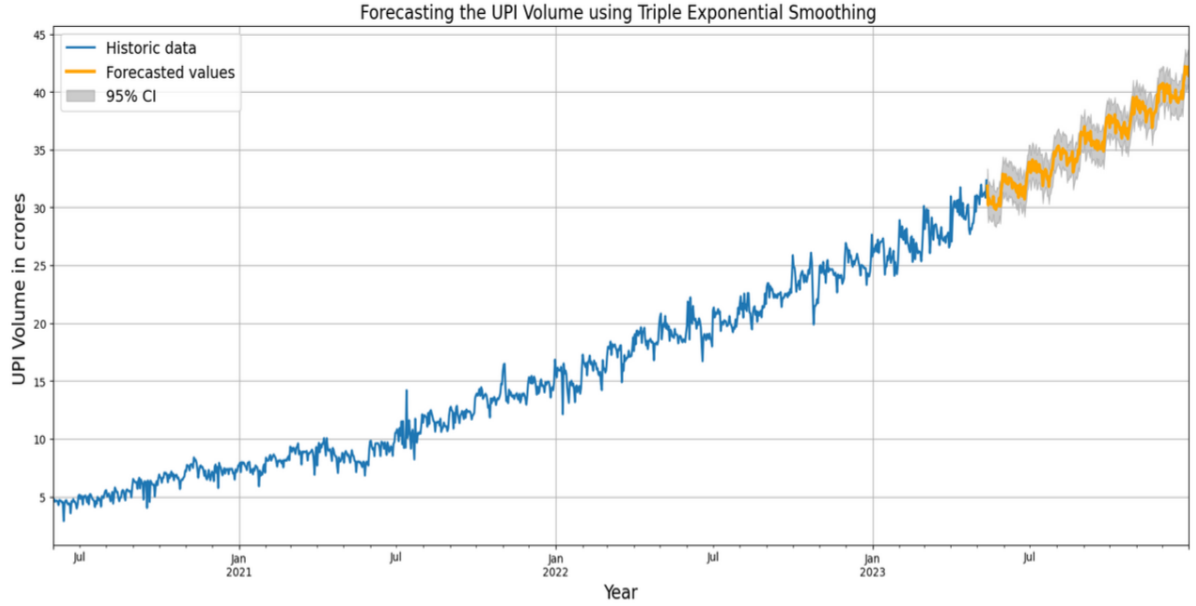
Now, let us see the MAPE of each model for the UPI volume series as follows:



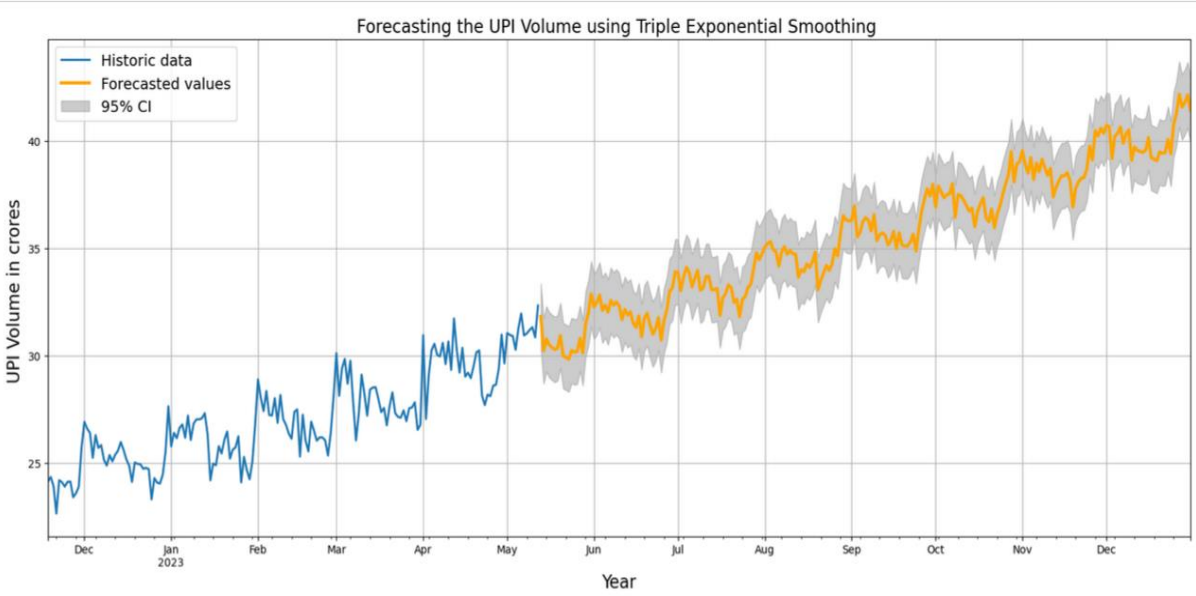
Comment: Again, looking to the **MAPE** plot for various models fitted for the UPI Value series, we have concluded that the **Triple Exponential Smoothing** model is the best among the fitted models, while the 2nd best model is the Seasonal Auto-Regressive Integrated Moving Average (**SARIMA**) model for the UPI Value series, therefore for forecasting purpose we will use the Triple Exponential Smoothing Model to forecast the Future values of the UPI value data.

11. FORECASTING FUTURE VALUES OF UPI VOLUME AND UPI VALUE UP TO NEXT 6 MONTHS:

- **UPI VOLUME FORECASTING**



Let us take a closer look to the forecasted values of UPI Volume:

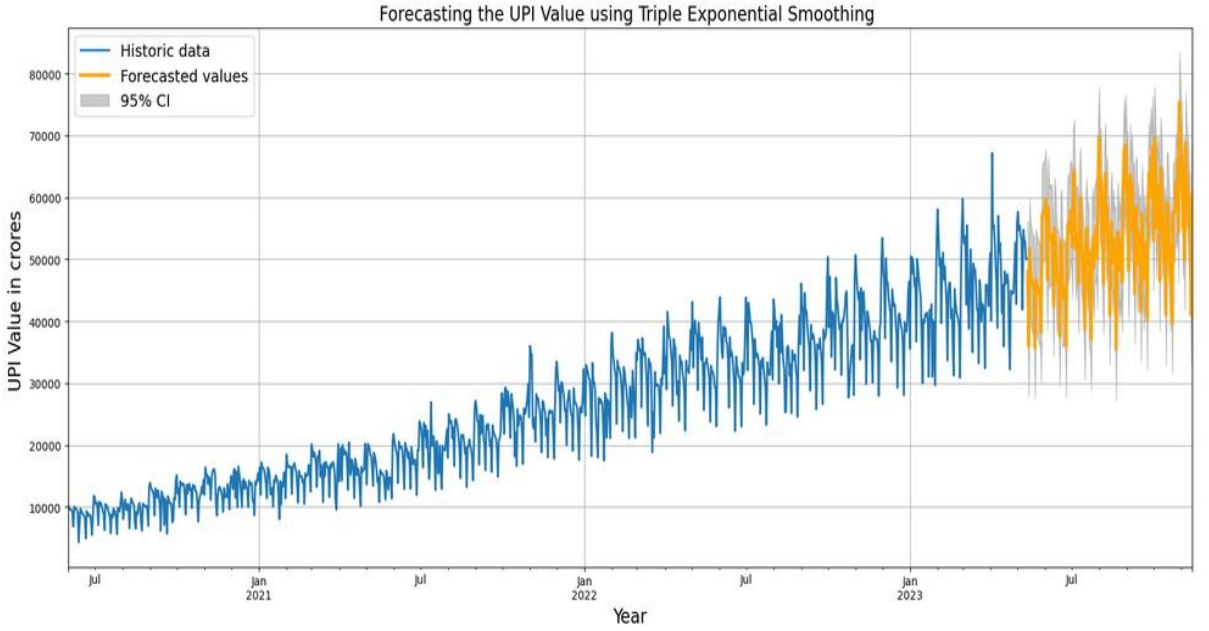


UPI VOLUME FORECASTED VALUES:

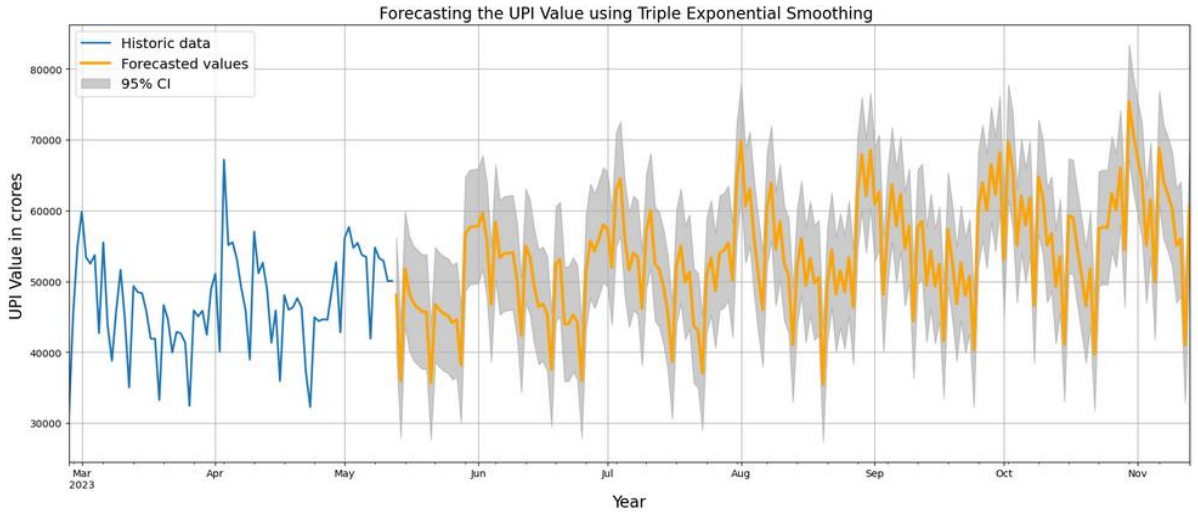
Out[140]:

Date	Final_forecast_vol	Lower_bound_95%	Upper_bound_95%
2023-05-13	31.831459	30.307516	33.355402
2023-05-14	30.225392	28.701449	31.749334
2023-05-15	30.782618	29.258675	32.306560
2023-05-16	30.506887	28.982944	32.030829
2023-05-17	30.383615	28.859673	31.907558
2023-05-18	30.289150	28.765207	31.813092
2023-05-19	30.365143	28.841200	31.889085
2023-05-20	30.937486	29.413543	32.461429
2023-05-21	29.992799	28.468856	31.516742
2023-05-22	29.925723	28.401781	31.449666
2023-05-23	29.842312	28.318370	31.366255
2023-05-24	30.238262	28.714319	31.762204
2023-05-25	30.170733	28.646790	31.694675
2023-05-26	30.197438	28.673495	31.721380
2023-05-27	30.807756	29.283814	32.331699
2023-05-28	30.131336	28.607393	31.655279
2023-05-29	31.478622	29.954680	33.002565
2023-05-30	31.991185	30.467242	33.515128
2023-05-31	32.877599	31.353656	34.401541
2023-06-01	32.275971	30.752028	33.799914
2023-06-02	32.488572	30.964630	34.012515
2023-06-03	32.825514	31.301571	34.349456
2023-06-04	32.115677	30.591735	33.639620
2023-06-05	32.351457	30.827514	33.875400
2023-06-06	32.029287	30.505345	33.553230
2023-06-07	32.572801	31.048859	34.096744
2023-06-08	32.357340	30.833397	33.881283
2023-06-09	32.499038	30.975095	34.022981
2023-06-10	32.322065	30.798123	33.846008
2023-06-11	31.684010	30.160067	33.207953

- **UPI VALUE FORECASTING**



Again, Let us take a closer look to the forecasted values of UPI Value:



UPI VALUE FORECASTED VALUES:

Out[146]:

Date	Final_forecast_val	Lower_bound_95%	Upper_bound_95%
2023-05-13	48054.733565	39964.805993	56144.661136
2023-05-14	35931.992391	27842.064819	44021.919962
2023-05-15	51794.634849	43704.707278	59884.562421
2023-05-16	48257.968214	40168.040642	56347.895785
2023-05-17	46891.510923	38801.583351	54981.438494
2023-05-18	46254.015880	38164.088308	54343.943451
2023-05-19	45702.009043	37612.081471	53791.936614
2023-05-20	45713.129594	37623.202022	53803.057165
2023-05-21	35674.355011	27584.427439	43764.282582
2023-05-22	46742.017845	38652.090273	54831.945416
2023-05-23	45966.488635	37876.561063	54056.416206
2023-05-24	45431.009248	37341.081677	53520.936820
2023-05-25	45123.653473	37033.725901	53213.581044
2023-05-26	44183.515350	36093.587778	52273.442921
2023-05-27	44610.448903	36520.521331	52700.376474
2023-05-28	38193.773376	30103.845805	46283.700948
2023-05-29	56738.472208	48648.544636	64828.399780
2023-05-30	57623.833318	49533.905746	65713.760889
2023-05-31	57739.868115	49649.940544	65829.795687
2023-06-01	57816.294981	49726.367409	65906.222552
2023-06-02	59676.788026	51586.860454	67766.715597
2023-06-03	55908.745461	47818.817890	63998.673033
2023-06-04	46710.848900	38620.921329	54800.776472
2023-06-05	58409.980974	50320.053403	66499.908546
2023-06-06	53349.296482	45259.368910	61439.224053
2023-06-07	53907.850200	45817.922629	61997.777772
2023-06-08	53997.351941	45907.424369	62087.279512
2023-06-09	54082.175384	45992.247812	62172.102955
2023-06-10	50204.007740	42114.080168	58293.935311
2023-06-11	42390.419797	34300.492225	50480.347368

!!!!!!!!!!

12. References:

Along with our dedicated and expert professor Dr. Ashish Das we have taken heip from the following online resources also:

- https://chat.openai.com/
- https://github.com/jiwidi/time-series-forecasting-with-python
- https://github.com/sakusuma/AirlinePassengerPrediction/blob/main/AirlinePassengerPrediction_TimeSeries.ipynb
- https://www.udemy.com/course/complete-practical-time-series-forecasting-in-python/learn/lecture/24043500?start=15
- https://www.youtube.com/watch?v=k9jOeUKRGSs&list=PLtIY5kwXKny91_IbkqcIXuv6t1prQwFhO