Major Project
on
Time Series Analysis of Climate of Delhi

Submitted to the Amity University Uttar Pradesh In partial fulfillment of requirements for the award of the Degree of

BACHELOR OF STATISTICS

By

Arpit Saxena
Enrollment No: A4479119002

Under the Supervision of:

Supervisor

Dr. Dheeraj Pawar

Department of Statistics
Amity Institute of Applied Science

Amity Institute of Applied Sciences, Amity University Uttar Pradesh, Sector 125, Noida – 201303 (India)

AMITY UNIVERSITY

UTTAR PRADESH

AMITY INSTITUTE OF APPLIED SCIENCES

Synopsis of Major Project:

Title: Time Series Analysis of Climate of Delhi

Name of Guide: Dr. Dheeraj Pawar

Programme:- B.Stats.		**Semester:- 6th**	
S.No.	**Enrollment No.**	**Name**	**Signature**
1	**A4479119002**	**Arpit Saxena**	

Summary:- Time Series Analysis on Delhi Climate from the year 2013-2017 and Sarima model is conducted for it. The Accuracy of the Model is checked and forecasting is done for the year 2018.

Schedule of work completion:- 3 January 20222 – 22 April 2022

Signature of Student

Signature of Guide

Signature of Programme Leader

Approval by Board of Faculty

Member	**Signature**	**Remark (Approved / Not Approved)**

DECLARATION

I, Arpit Saxena, student of Bachelor Of Statistics hereby declare that the Major project titled "Time Series Analysis of Climate of Delhi" which is submitted by me to Department of Statistics, Amity Institute of Applied Sciences, Amity University, Uttar Pradesh, Noida, in partial fulfillment of requirement for the award of the degree of Bachelor Of Statistics, has not been previously formed the basis for the award of any degree, diploma or other similar title or recognition.

Noida

Date: 22 April 2022

Arpit Saxena

CERTIFICATE

On the basis of declaration submitted by Arpit Saxena ,student of Bachelor Of Statistics, I hereby certify that the Major project titled "Time Series Analysis of Climate of Delhi" which is submitted to Department of Statistics, Amity Institute of Applied Sciences, Amity University, Uttar Pradesh, Noida, in partial fulfillment of requirement for the award of the degree of Bachelor Of Statistics, is an original contribution with existing knowledge and faithful record of work carried out by him under my guidance and supervision.

To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Noida

Date: 22 April 2022

Dr. Dheeraj Pawar

Department of Statistics
Amity Institute of Applied Sciences
Amity University, Uttar Pradesh, Noida

ACKNOWLEDGEMENT

I, Arpit Saxena, would like to express my deep sense of gratitude towards my faculty guide Dr.Dheeraj Pawar, for guiding me from the inception till the completion of the project. The experience of working under my faculty guide has been a value addition to the learning during my course of bachelors of statistics. I would like to express my heartfelt gratitude for Dr. Dheeraj Pawar with whose guidance and valuable suggestions I was able to maximize on the learning curve during the completion of my project. His timely responses to any issues that came along and his promptness helped me to successfully complete the project.

I am thankful to him that he provided me an opportunity to work under his guidance and sharing his valuable experience

Also, I am thankful to Amity Institute of Applied Science (AIAS) and Department of Statistics for providing me with such an opportunity to gain analytical knowledge and skills, imparted as a part of curriculum.

Arpit Saxena

CONTENTS

1. **ABSTRACT**
2. **INTRODUCTION**
    i. **TIME SERIES ANALYSIS**
    ii. **ARIMA MODELLING**
    iii. **SARIMA MODELLING**
3. **OBJECTIVES OF THE PROJECT**
4. **METHODOLOGY**
5. **SAMPLE OF THE DATASET**
6. **LOADING LIBRARIES**
7. **DATA PREPARATION AND CLEANING**
    i. **DETECTION OF OUTLIERS**
    ii. **REMOVAL OF OUTLIERS**
8. **EXPLORATION OF DATA**
    i. **DESCRIPTIVE STATISTICS**
    ii. **DISTRIBUTION OF THE DATA**
9. **RESAMPLING OF MONTLY TEMPERATURES**
10. **VISUALIZE THE DATA**
11. **DECOMPOSITION**
12. **CHECKING OF STATIONARITY**
    i. **ROLLING STATISTICS**
    ii. **AUGMENTED DICKEY-FULLER TEST**
13. **STATIONARIZE THE DATA**
14. **SARIMA MODELLING**
    i. **AUTOCORRELATION PLOTS**
    ii. **PARAMETER SELECTION FOR THE SARIMA MODEL**
    iii. **FITTING OF THE MODEL**
    iv. **MODEL DIAGNOSTICS**
    v. **CHECKING ACCURACY OF THE MODEL**
    vi. **PRODUCING AND VISUALIZING FORECASTS**
15. **CONCLUSION**
16. **REFERENCES**

Abstract

This project report includes the change in pattern of Delhi Climate Weather for the time period 2013-2017 which is further used for forecasting weather for the year 2018 with the help of SARIMA model in python. Since the temperature of 365 days is seasonal so ARIMA model cannot be used, therefore it required the use of its extension which is Seasonal ARIMA model (SARIMA). Today's air temperature depends on yesterday's air temperature which further depends on the day before and so on. This assumption is used to predict the values of air temperature for the future. The model explained in this report is performed using pandas, numpy, statsmodel libraries, matplotlib and pdmarima in Python to predict the values.

Introduction

- **Time Series**

  A group of observations which are kept at regular time intervals are known as time series and analyzing those observations to predict future values invokes time series analysis. It is used for data which is stationary and are utilized for analysis purpose in stock marketing, weather data, temperature readings, etc. If it is not stationary then it has to be made stationary for further analysis i.e. a series not having a predictable pattern in a long term. It requires plenty of observations which are collected at a consistent time interval and not randomly which show the nature of the change in variables over time and also helps in prediction of the forthcoming events with the help of past values.

- **ARIMA Modelling**

  Auto Regressive Integrated Moving Average, known as ARIMA, is a time series forecasting model in which the previous qualities are utilized to foresee the future estimations of the information. It is basically an extension of ARMA model having additional integration component which is further merged by two components as follows:

  - AR(Auto Regressive) is utilizing past qualities to anticipate future qualities and is indicated by ‘p’
  - MA (Moving Average) is utilizing the previous blunders to foresee the future estimations of the details and is signified by 'q'. The level of differencing is signified by’d’.

  It has three request boundaries (p, d, q) and demonstrated as ARIMA (p,d,q) where

  p = AR model’s order
  d = differencing’s order to get stationary series
  q = MA model’s order

- **SARIMA Modelling**

  SARIMA or Seasonal ARIMA is basically an extension of ARIMA model having additional component i.e. it upholds the seasonal component of a univariate time series data which is not supported by ARIMA. It includes both seasonal as well as non-seasonal factors and are denoted as ARIMA (p,d,q)*(P,D,Q)S where

  Trend Elements -

  p = AR model’s order

  d = differencing’s order to get stationary series

  q = MA model’s order

Seasonal Elements -

P: Order containing Seasonal autoregressive.

D: Order containing Seasonal difference.

Q: Order containing Seasonal moving average.

m: Time steps of a single seasonal period.

Objectives of the Project

- To conduct a time series analysis on Climate of Delhi from the year 2013-2017.
- To construct a SARIMA model using Temperature from the year 2013-2017.
- To check the accuracy of the model.
- To forecast values of the year 2018.

Methodology

This project includes the time series analysis on the change Delhi Climate from the year 2013-2017 and construction of a Sarima model (seasonal arima model) using temperature for the year 2013-2017 on Python. The project includes the explanation of time series, arima modelling and Sarima modelling. Then we will set objective of our project and methodology applied. Python is used in our project and includes the libraries loaded, preparation of data and cleaning it with the help of boxplot to detect outliers in order to remove them. It also includes data exploration. Then a filtered data for visualization and decomposing a time series data to check trend and seasonality. The data was assessed for stationarity using the rolling statistics method and Augmented dickey-fuller test (ADF Test). But it was not stationary, so it was stationarized using differencing and it was again tested for stationarity. Now, it is stationary so the PACF and ACF plots are plotted to check the presence of MA and AR term in model and further, the different possible combinations of parameters are preferred based on their least AIC value and then the SARIMA model is fitted. Afterwards, the model diagnostics are represented and furthermore, the accuracy of the model is checked using MSE and RMSE and finally forecasting is done.

Sample of the Dataset

Date	Temperature	Humidity	Wind Speed	Pressure
01-01-2013	10	84.5	0	1015.667
02-01-2013	7.4	92	2.98	1017.8
03-01-2013	7.166667	87	4.633333	1018.667
04-01-2013	8.666667	71.33333	1.233333	1017.167
05-01-2013	6	86.83333	3.7	1016.5

06-01-2013	7	82.8	1.48	1018
07-01-2013	7	78.6	6.3	1020
08-01-2013	8.857143	63.71429	7.142857	1018.714
09-01-2013	14	51.25	12.5	1017
10-01-2013	11	62	7.4	1015.667
11-01-2013	15.71429	51.28571	10.57143	1016.143
12-01-2013	14	74	13.22857	1015.571
13-01-2013	15.83333	75.16667	4.633333	1013.333
14-01-2013	12.83333	88.16667	0.616667	1015.167
15-01-2013	14.71429	71.85714	0.528571	1015.857
16-01-2013	13.83333	86.66667	0	1016.667
17-01-2013	16.5	80.83333	5.25	1015.833
18-01-2013	13.83333	92.16667	8.95	1014.5
19-01-2013	12.5	76.66667	5.883333	1021.667
20-01-2013	11.28571	75.28571	8.471429	1020.286
21-01-2013	11.2	77	2.22	1021
22-01-2013	9.5	79.66667	3.083333	1021.8
23-01-2013	14	60.16667	4.016667	1020.5
24-01-2013	13.83333	60.66667	6.166667	1020.5
25-01-2013	12.25	67	5.55	1020.75
26-01-2013	12.66667	64.16667	6.8	1019.667
27-01-2013	12.85714	65.57143	5.557143	1018.143
28-01-2013	14.83333	56	3.7	1017.833
29-01-2013	14.125	65.5	3.2375	1016.625
30-01-2013	14.71429	70.42857	1.057143	1017.857
31-01-2013	16.2	65.6	2.96	1018.4
01-02-2013	16	73	2.22	1016
02-02-2013	16.28571	77.57143	1.328571	1017.143
03-02-2013	18	65.57143	1.857143	1015.286
04-02-2013	17.42857	74.28571	11.11429	1014.571
05-02-2013	16.625	92.375	9.725	1016.375
06-02-2013	16.66667	71.33333	8.633333	1018.667
07-02-2013	15.6	59.4	10.74	1018.6
08-02-2013	14	70.42857	9.257143	1017.143
09-02-2013	15.42857	61.28571	9.257143	1016.857
10-02-2013	15.25	71.5	3.475	1017.125
11-02-2013	15.875	70.5	5.325	1016.5
12-02-2013	15.33333	70.33333	7.416667	1017.5
13-02-2013	16.28571	70.14286	6.085714	1016
14-02-2013	17.33333	63.83333	4.333333	1014.167
15-02-2013	19.16667	65.33333	10.18333	1011.667
16-02-2013	14.42857	92.71429	8.485714	1008
17-02-2013	13.66667	90	0	1012.667
18-02-2013	15.6	78.4	12.22	1016.2
19-02-2013	15.85714	82	5.814286	1015.714

20-02-2013	17.71429	74.71429	5.814286	1017	`11
21-02-2013	20	67.28571	6.614286	1015.429	
22-02-2013	20.5	65.625	10.875	1016	
23-02-2013	17.42857	74.85714	9.257143	1017	
24-02-2013	16.85714	78.85714	7.4	1018.857	
25-02-2013	16.875	72.875	4.6375	1018.25	
26-02-2013	17.85714	70	17.5875	1015.143	
27-02-2013	20.8	57.2	6.66	1015.2	
28-02-2013	19.42857	52.85714	12.95714	1017.429	
01-03-2013	17.33333	49.33333	24.06667	1016.333	
02-03-2013	19	54	15.725	1016.25	
03-03-2013	19.33333	62.83333	8.633333	1016.167	
04-03-2013	17.6	71	5.56	1015.8	
05-03-2013	20.875	61.875	4.1625	1016.375	
06-03-2013	20.85714	65.28571	6.871429	1015.714	
07-03-2013	23.42857	57.14286	8.728571	1015.286	
08-03-2013	24.16667	44.83333	7.1	1014.833	
09-03-2013	25.42857	49.71429	5.285714	1009.286	
10-03-2013	23.14286	57.57143	4.228571	1008	
11-03-2013	24	66.33333	0.933333	1011.167	
12-03-2013	23.5	62.5	4.933333	1011.5	
13-03-2013	21.5	70.5	5.55	1009	
14-03-2013	22.33333	61.16667	12.03333	1013.167	
15-03-2013	24.16667	45.83333	7.716667	1016.167	
16-03-2013	20.33333	67.66667	3.7	1016.167	
17-03-2013	22.66667	58.66667	8.95	1015	
18-03-2013	23.42857	58.14286	17.45714	1009.429	
19-03-2013	22.5	73.66667	10.48333	1011	
20-03-2013	29.16667	36.33333	6.8	1009.5	
21-03-2013	23.83333	58.5	10.5	1008.333	
22-03-2013	25.25	50.25	9.7125	1006.375	
23-03-2013	27.375	50.125	9.95	1007.625	
24-03-2013	27	48.75	10.8875	1010.25	
25-03-2013	23.5	45.5	15.9625	1010.625	
26-03-2013	24.14286	44.57143	12.95714	1008.857	
27-03-2013	21	62	1.85	1009	
28-03-2013	22.42857	62.71429	7.414286	1009.571	
29-03-2013	21.25	70.375	5.55	1009.75	
30-03-2013	23.5	54.75	11.1125	1008.625	
31-03-2013	23.2	58	6.66	1008.6	
01-04-2013	25.375	45.5	4.4	1008.5	
02-04-2013	25.16667	51	8.65	1009.5	
03-04-2013	26.2	45.6	8.14	1009	
04-04-2013	24.6	41.8	11.12	1007.8	
05-04-2013	25.6	31	15.56	1007	

06-04-2013	25.85714	29.85714	11.9	1006.143
07-04-2013	29.14286	23.28571	10.31429	1005
08-04-2013	28.71429	33.85714	5.3	1006
09-04-2013	30.16667	30.5	8.65	1005.333
10-04-2013	30	28	6.1	1006.714
11-04-2013	30	24.2	7.78	1006.4
12-04-2013	28.85714	32.57143	6.342857	1007.571
13-04-2013	30.2	29.2	10	1008.4
14-04-2013	28.25	39.375	6.4875	1007
15-04-2013	28.25	41.375	6.25	1003.875
16-04-2013	32.125	24.625	10.425	1000
17-04-2013	29.2	24.2	6.66	1002.2
18-04-2013	30.28571	30.28571	4.757143	1002.857
19-04-2013	28.28571	31.28571	3.971429	1002.571
20-04-2013	30.625	29	7.6375	1003.375
21-04-2013	27.66667	38.66667	13.88333	1006.833
22-04-2013	27.375	45.375	7.65	1010
23-04-2013	28.625	44.125	4.625	1009.875
24-04-2013	30.28571	41.71429	2.114286	1008.571
25-04-2013	31.14286	38.28571	3.7	1007.714
26-04-2013	29.875	45.875	6.7125	1008.375
27-04-2013	31.14286	31.42857	13.48571	1007
28-04-2013	30.57143	28	12.97143	1005.429
29-04-2013	32.125	26.375	7.875	1004.875
30-04-2013	31.14286	32	7.928571	1004.857
01-05-2013	31.85714	15.85714	12.68571	1002.833
02-05-2013	29.83333	22.16667	11.43333	1001.2
03-05-2013	28.57143	31.57143	9	999.4286
04-05-2013	32.85714	31.42857	2.914286	999
05-05-2013	32.625	31.125	3.0125	1000.625
06-05-2013	32.75	39.25	3.7	1001.625
07-05-2013	32.875	33.25	7.175	1002
08-05-2013	34.5	23	9.25	1001.167
09-05-2013	34.28571	26	8.985714	1000.857
10-05-2013	34	27.71429	9.528571	1000.143
11-05-2013	30.75	30.375	14.8125	999.875
12-05-2013	29.85714	40.14286	5.828571	1003.714
13-05-2013	31.71429	34	3.971429	1003.571
14-05-2013	32.28571	34.28571	4.771429	1001.571
15-05-2013	33	33	3.2375	999.8571
16-05-2013	33	34.75	7.175	998.5
17-05-2013	32.83333	28.16667	9.866667	1000.333
18-05-2013	31.4	42.2	12.96667	999
19-05-2013	35.33333	22.33333	15.75	999.6667
20-05-2013	36.4	24.2	7.4	998.4

21-05-2013	36	19	11.37143	998.6667
22-05-2013	36.75	22.125	17.5875	998.625
23-05-2013	37.5	23.33333	13.56667	997.1667
24-05-2013	38.42857	27.42857	11.38571	996.4286
25-05-2013	38.71429	22.42857	10.31429	998.1429
26-05-2013	37.8	21.2	10.74	998.2
27-05-2013	35.85714	31.57143	8.728571	999.1429
28-05-2013	35.33333	29	8.333333	1000.167
29-05-2013	34.14286	27.85714	5.557143	999.4286
30-05-2013	32.2	28.2	5.56	999.6
31-05-2013	33.625	40.125	10.6375	998.7143
01-06-2013	32	54	13.4375	998.75
02-06-2013	32.4	56.6	14.2	1001
03-06-2013	35.6	47	12.24	999.6
04-06-2013	35.85714	42.57143	5.028571	999.8571
05-06-2013	37.16667	36.5	9.866667	998
06-06-2013	31.28571	61.71429	10.04286	997.2857
07-06-2013	34	49.25	7.4125	998
08-06-2013	34.2	57.4	13.34	997
09-06-2013	36.16667	40	11.41667	995.1667
10-06-2013	36.625	42.75	8.1	995.125
11-06-2013	30.16667	61.66667	16.35	1000
12-06-2013	34.14286	54.14286	7.414286	998
13-06-2013	29.83333	68.66667	9.866667	998.6667
14-06-2013	30.14286	68.14286	7.942857	997.4286
15-06-2013	30.71429	65	13.22857	997.8571
16-06-2013	27	88.85714	8.485714	996.4286
17-06-2013	26.875	87.375	8.3375	994.75
18-06-2013	28.4	83.6	5.18	996.4
19-06-2013	29.85714	72.85714	4.228571	999
20-06-2013	33	52.71429	4.771429	997.1667
21-06-2013	34.83333	47.33333	5.866667	995.3333
22-06-2013	35.6	39.4	10.74	996.2
23-06-2013	35.16667	53	8.35	995.6667
24-06-2013	33.14286	55	13.22857	996.2857
25-06-2013	30.57143	70.42857	11.11429	999.1667
26-06-2013	30.66667	71.5	6.483333	999
27-06-2013	31.42857	61.85714	7.957143	996.7143
28-06-2013	31.5	64.25	8.8125	996
29-06-2013	33.25	57	13.875	995.25
30-06-2013	32.83333	52.16667	10.5	997.1667
01-07-2013	33.85714	54	8.728571	996.4286
02-07-2013	33.14286	58.28571	9.8	996
03-07-2013	31.57143	70	6.357143	997.5714
04-07-2013	32.375	67.5	9.25	997

05-07-2013	32.8	62.2	12.96	997.8
06-07-2013	31	73.14286	11.37143	1000.571
07-07-2013	31.16667	77.33333	7.1	1000.167
08-07-2013	29.83333	75.16667	6.483333	997.5
09-07-2013	29	78.83333	8.016667	996.8333
10-07-2013	29.75	75.375	6.4875	995.75
11-07-2013	32.5	65.5	7.116667	997.3333
12-07-2013	28.875	86.625	5.5625	1000
13-07-2013	31.75	70.25	12.0625	998.875
14-07-2013	30.75	71.625	4.8125	997.375
15-07-2013	31.75	73.25	13.5375	998.375
16-07-2013	29.875	83.875	9	999
17-07-2013	31.125	78.375	8.65	998
18-07-2013	31.75	76	5.55	995
19-07-2013	30.5	73.5	9.266667	995.3333
20-07-2013	26.83333	90.66667	4.933333	999.1667
21-07-2013	28.2	88	2.24	996.4
22-07-2013	29.5	80.33333	4.333333	996.8333
23-07-2013	31.85714	70.71429	3.971429	998
24-07-2013	29.71429	84.14286	3.971429	997.1429
25-07-2013	28.33333	84.5	8.016667	995.1667
26-07-2013	30	73.85714	5.571429	995.4286
27-07-2013	30.14286	79	4.228571	997.2857
28-07-2013	32.28571	67.42857	6.085714	998.1429
29-07-2013	31	77.42857	4.771429	997.1429
30-07-2013	30.5	75.66667	8.016667	996.5
31-07-2013	28.83333	78.5	9.866667	996.6667
01-08-2013	30	73.5	5.55	998
02-08-2013	31.57143	66.42857	11.11429	999.4286
03-08-2013	32.5	62.33333	10.81667	1000.833
04-08-2013	32.5	62.5	5.1	993.25
05-08-2013	29.5	77	6.783333	1001

Table 1

Loading Libraries

First, we import libraries in python as follows:

```
import pandas as pd
from datetime import datetime, timedelta
import statsmodels.api as sm
import matplotlib.pyplot as plt
import numpy as np

#Seasonal Decomposition
from statsmodels.tsa.seasonal import seasonal_decompose
import seaborn as sns

#Augmented Dicker Fulley Test
from statsmodels.tsa.stattools import adfuller
%matplotlib inline

#Autocorrelation and Partial Autocorrelation Plots
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

#Grid Search
import itertools

#Root Mean Square Error
from statsmodels.tools.eval_measures import rmse

#Specify to ignore warning messages
import warnings
warnings.filterwarnings('ignore')
```

Data Cleaning and Preparation

- **Detection of Outliers**
  Boxplots are presented for the dataset of each column to check outliers as follows:

  - **Temperature**

```
#Checking of Outliers
plt.boxplot(df_raw.Temperature)
plt.xlabel("Temperature")
```

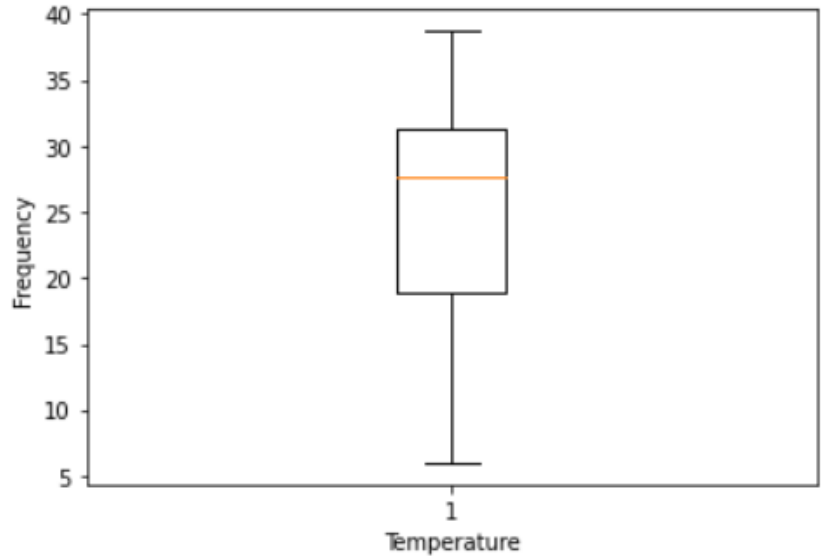

Figure 1

Here, it is clear that there are no outliers present in temperature data so there is no need to clean it.

- **Wind Speed**

```
#Checking of Outliers
plt.boxplot(df_raw.WindSpeed)
plt.xlabel("WindSpeed")
```

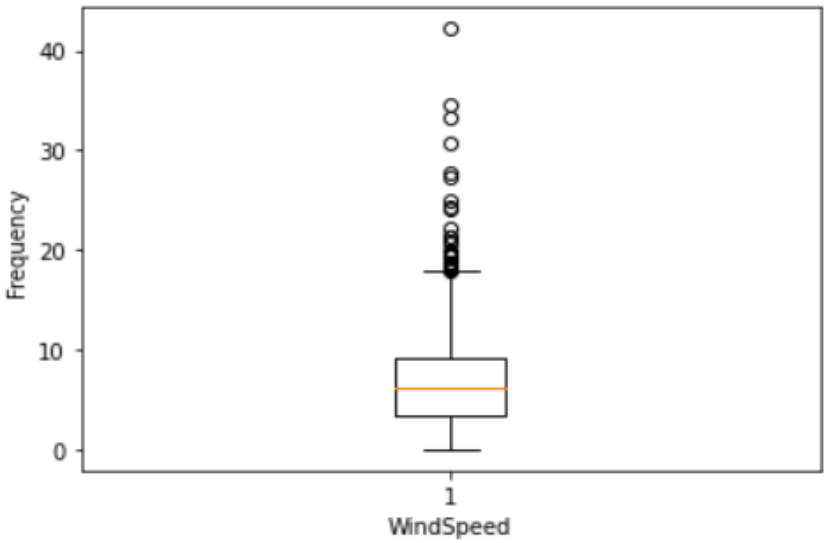

Figure 2

Here, it is clear that most values are above the whisker line which is around 18 in this case, so these values are considered as outliers which should be removed.

- **Humidity**

```
#Checking of Outliers
plt.boxplot(df_raw.Humidity)
plt.xlabel("Humidity")
```

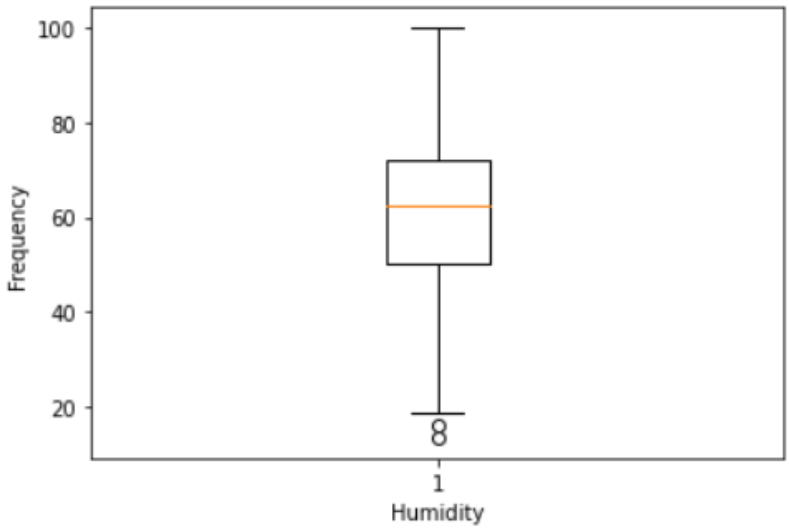

Figure 3

Here, it is clear that two values are below the whisker line which is around 20 in this case, so these values are considered as outliers which should be removed.

- **Pressure**

```
#Checking of Outliers
plt.boxplot(df_raw.Pressure)
plt.xlabel("Pressure")
```

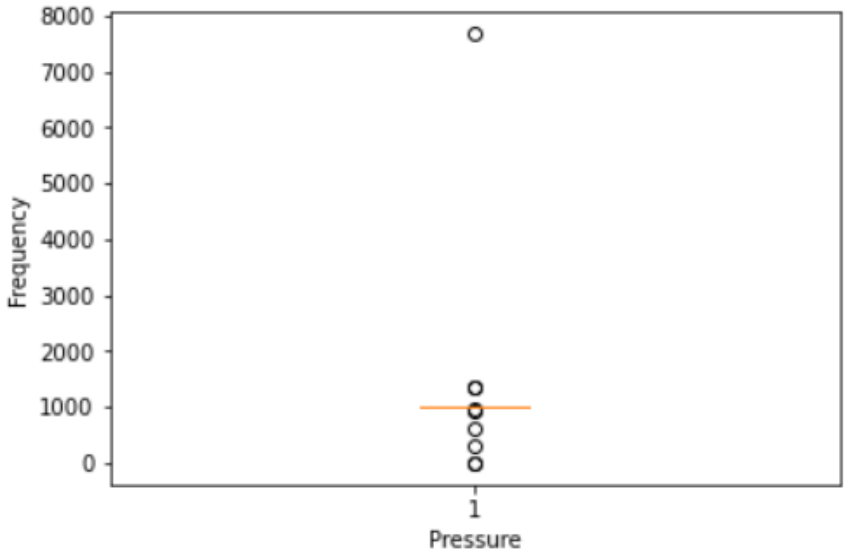

Figure 4

Here, it is clear that plenty of values lie around 1000 and one observation which is extremely high around 7000-8000 is an outlier which should be removes to obtain a clearer boxplot since it is burdensome to interpret in this boxplot.

- **Removal of Outliers**

```
#Outliers Detection
def outliers(df1, ft):
    Q1 = df1[ft].quantile(0.25)
    Q3 = df1[ft].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q1 + 1.5 * IQR

    Is = df1.index[ (df1[ft] < lower_bound) | (df1[ft] > upper_bound) ]

    return Is

index_list= []
for climate in ['Humidity']:
    index_list.extend(outliers(df_raw, climate))

index_list

#Outliers Removal
def remove(df1, Is):
   Is = sorted(set(Is))
   df1 = df1.drop(Is)
   return df1

df = remove(df_raw, index_list)
```

Following code removed all the outliers which are present in each column and now the cleaned data is utilized for data exploration and for further analysis.

- **Temperature**

```
#After Removal of Outliers
plt.boxplot(df.Temperature)
plt.xlabel("Temperature")
```

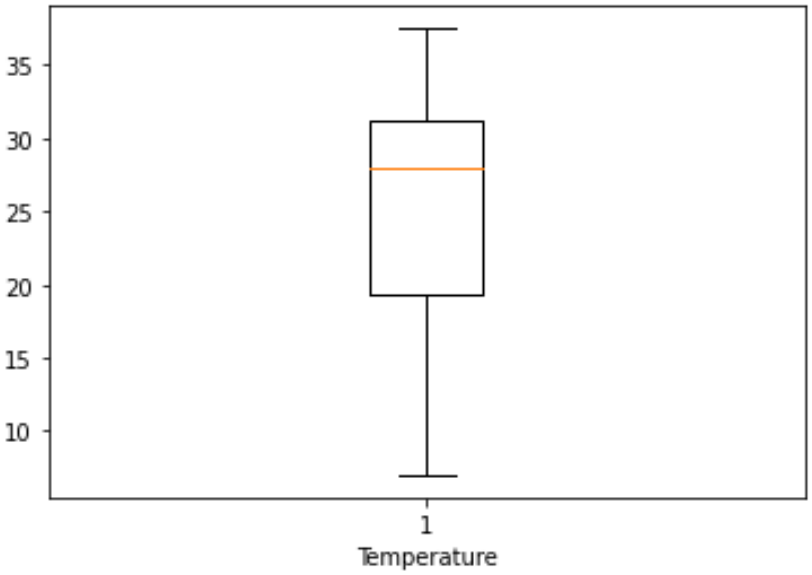

Figure 5

- Since the median is closer to the top of the box, and the whisker is also short on the upper end of the box, so the distribution is negatively skewed or skewed left.
- As there are no data points that are falling outside the whisker so it is verified that the outliers are removed.
- Since IQR (i.e. box length) is more, it indicates a more scattered data.

- **Wind Speed**

```
#After Removal of Outliers
plt.boxplot(df.WindSpeed)
plt.xlabel("WindSpeed")
```

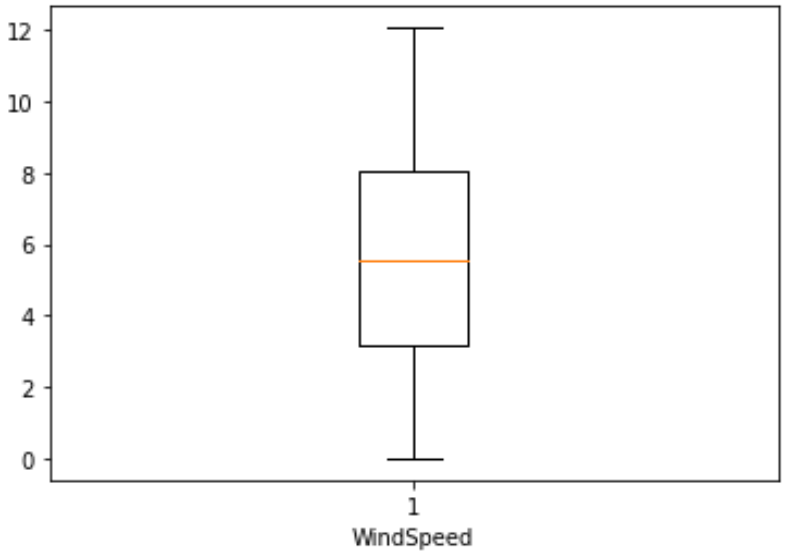

Figure 6

- Since the median is at the bottom of the box, and the whisker is also shorter on lower end of the box, so the distribution is positively skewed or skewed right.
- As there are no data points that are falling outside the whisker so it is verified that the outliers are removed.
- Since IQR (i.e. box length) is more, it indicates a more scattered data.

- **Humidity**

```
#After Removal of Outliers
plt.boxplot(df.Humidity)
plt.xlabel("Humidity")
df.drop(df[df['Humidity']<25].index, inplace=True)
```

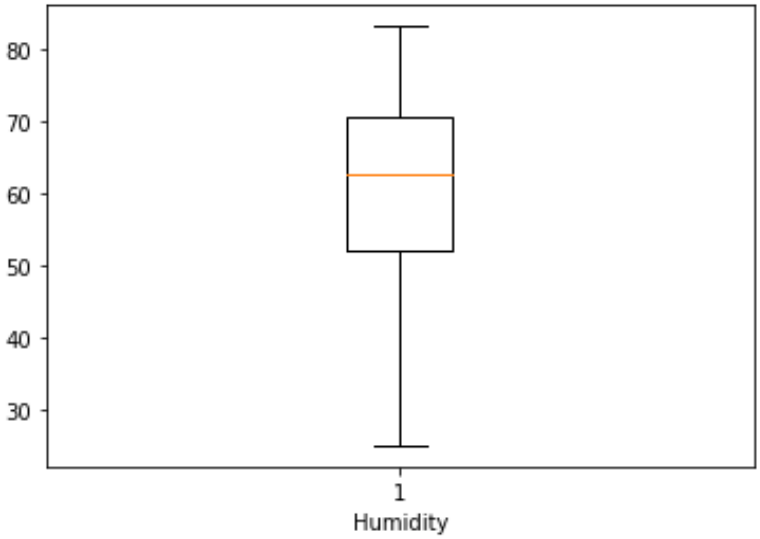

Figure 7

- Since the median is closer to the top of the box, and the whisker is also short on the upper end of the box, so the distribution is said to be negatively skewed or skewed left.

- As there are no data points that are falling outside the whisker so it is verified that the outliers are removed.
- Since IQR (i.e. box length) is less, it indicates a less scattered data.

- **Pressure**

```
#After Removal of Outliers
plt.boxplot(df.Pressure)
plt.xlabel("Pressure")
```

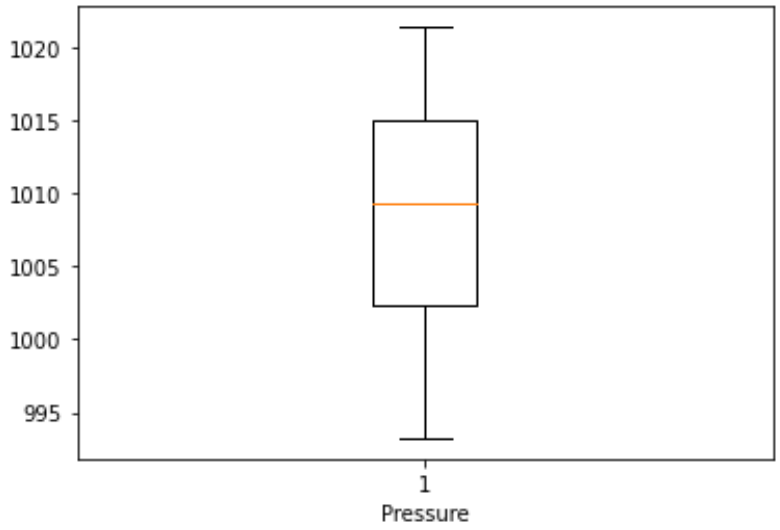

Figure 8

- Since the median is closer to the top of the box, and the whisker is also short on the upper end of the box, so the distribution is said to be negatively skewed or skewed left.
- As there are no data points that are falling outside the whisker so it is verified that the outliers are removed.
- Since IQR (i.e. box length) is more, it indicates a more scattered data.

Exploration of Data

Descriptive Statistics

Now, several descriptive statistics have to be computed with presence and absence of outliers respectively.

- **Presence of Outliers**

```
#Descriptive Statistics
df.describe()
```

	Temperature	Humidity	Wind Speed	Pressure
count	1462.000000	1462.000000	1462.000000	1462.000000
mean	25.495521	60.771702	6.802209	1011.104548
std	7.348103	16.769652	4.561602	180.231668
min	6.000000	13.428571	0.000000	-3.041667
25%	18.857143	50.375000	3.475000	1001.580357
50%	27.714286	62.625000	6.221667	1008.563492
75%	31.305804	72.218750	9.238235	1014.944901
max	38.714286	100.000000	42.220000	7679.333333

Table 2

- **Absence of Outliers**

	Temperature	Humidity	WindSpeed	Pressure
count	1154.00000000	1154.00000000	1154.00000000	1154.00000000
mean	25.61770158	59.65533821	5.77142389	1008.50277763
std	7.01697085	14.32864060	3.08309131	7.19637372
min	7.00000000	19.00000000	0.00000000	993.25000000
25%	19.37500000	51.33238636	3.23750000	1002.25000000
50%	27.93750000	62.24592391	5.68333333	1009.00000000
75%	31.25000000	70.25000000	8.11250000	1014.87500000
max	37.50000000	83.13333333	12.06250000	1021.37500000

Table 3

The summary function provided us the descriptive statistics of each column of our data. The numbers of observations were 1462 in presence of outliers but now it is 1154 in the absence of outliers. Also, we can observe the minimum, maximum, 1st quartile, median, 3rd quartile, standard deviation and mean of each variable and interquartile range can also be calculated using the 1st and 3rd quartile values respectively.

➢ Distribution of the data

```
#Distributtion of the Data
sns.histplot(data=df,x="Temperature",kde=True)
plt.xlabel("Temperature")
```

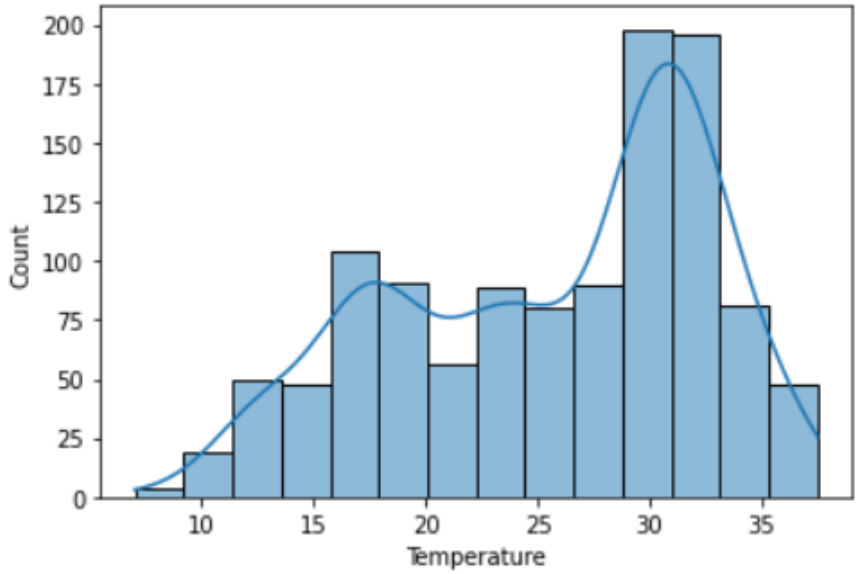

Figure 9

```
sns.histplot(data=df,x="Humidity",kde=True)
plt.xlabel("Humidity")
```

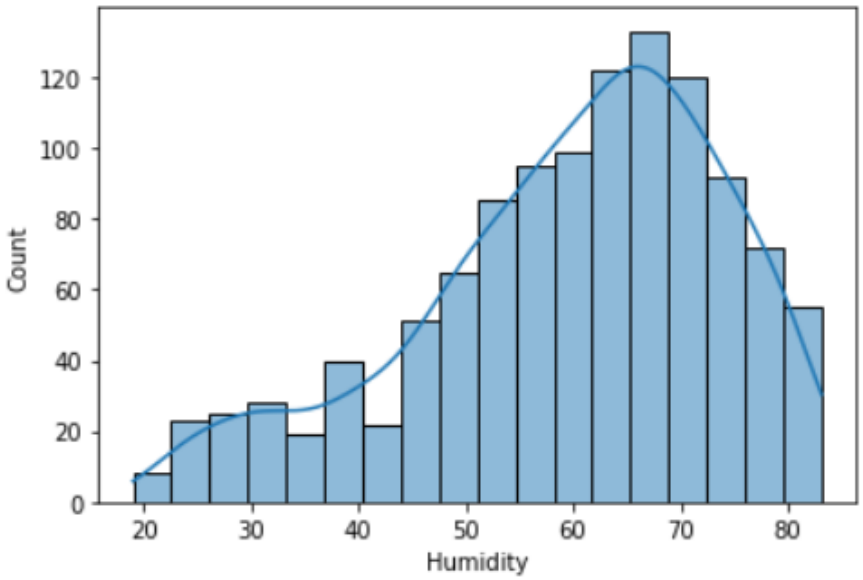

Figure 10

```
sns.histplot(data=df,x="WindSpeed",kde=True)
plt.xlabel("WindSpeed")
```

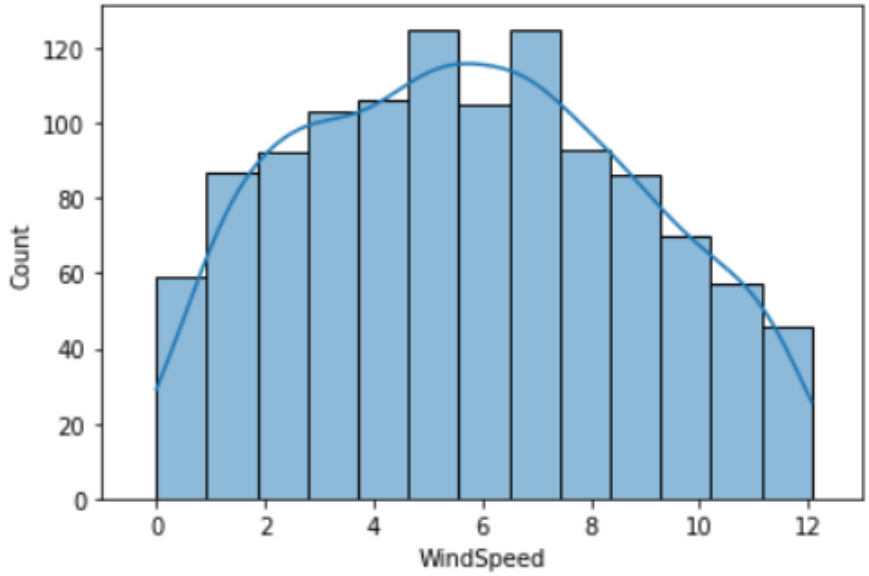

Figure 11

```
sns.histplot(data=df,x="Pressure",kde=True)
plt.xlabel("Pressure")
```

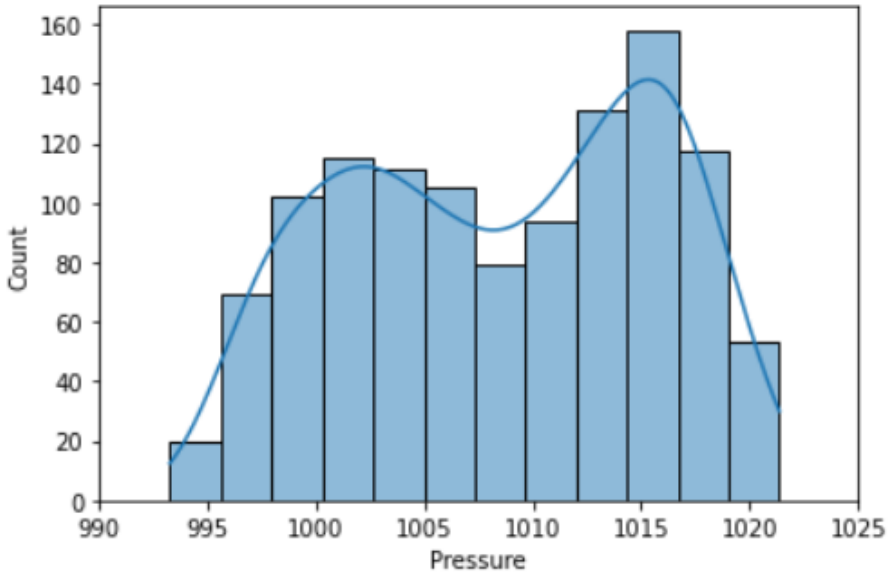

Figure 12

From the following histograms, it can be stated that the above distributions are negatively skewed i.e. leftly skewed except the Wind Speed histogram which is positively skewed i.e. rightly skewed invoking that temperature, Humidity and Pressure have plenty of data points having high values i.e. Delhi has higher Temperature, Humidity and Pressure whereas Wind Speed has higher number of data points having low values i.e. Delhi has low Wind Speed. The graphs also indicates the direction of the outliers which are absent since the data is cleaned.

Resampling of Monthly Temperatures

Since, it is burdensome to interpret the graphs for such a large data, so this problem can be solved by resampling into monthly temperatures from daily temperatures to get a more clear insights of the pictorial representations.

It can be done using the following code:

```
#Visualize the Data
resample_monthly_temp = df['Temperature'].resample('MS').mean()
resample_monthly_temp.plot(figsize=(12,6))
```

Visualize the Data

```
#Visualize the Data
resample_monthly_temp = df['Temperature'].resample('MS').mean()
resample_monthly_temp.plot(figsize=(12,6))
plt.ylabel("Temperature")
```

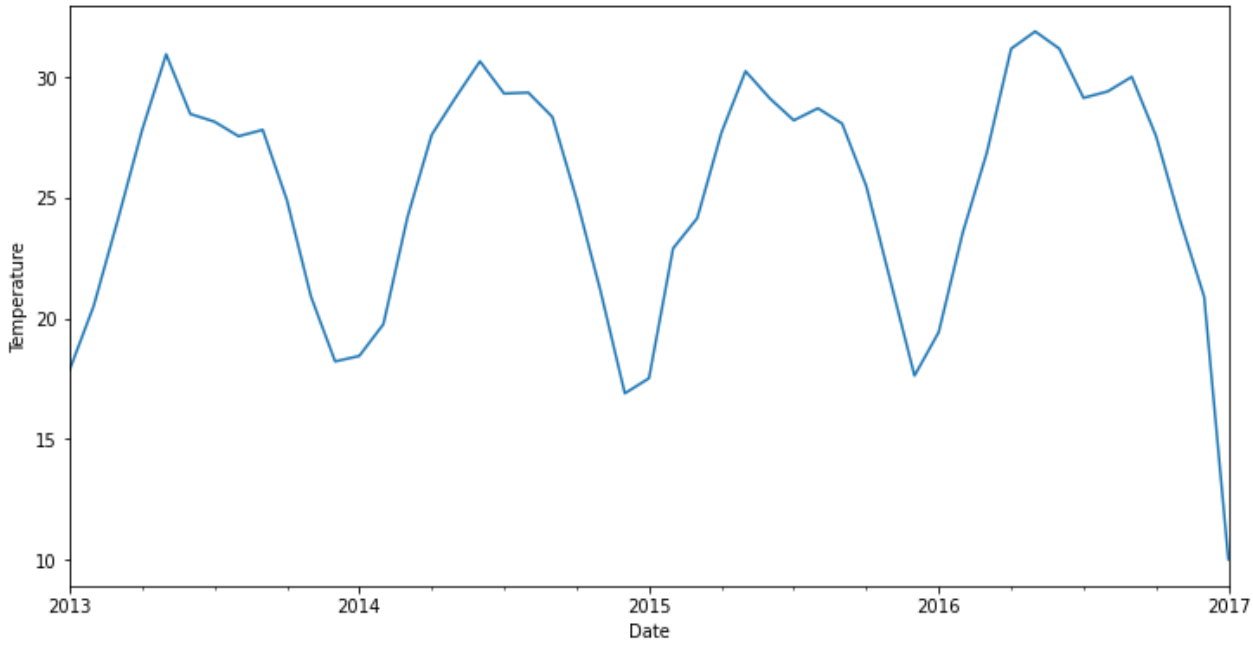

Figure 13

The given plot visualises the average temperature of Delhi from 2013-2017. The temperature shows an up and down pattern every year indicating the influence of the seasons which affects the temperature every year. The maximum temperature and minimum temperature are about 36 and close to 10 degrees respectively as depicted in the plot. It is clear in the graph about the seasonal patterns of the temperature in Delhi which occurs repeatedly and highlights that the temperature has seasonality and does not represent a trend.

Decomposition

```
from statsmodels.tsa.seasonal import seasonal_decompose
from datetime import datetime
seasonal_decompose(resample_monthly_temp).plot();
```

Decomposition is the separation of data of time series into its constituent components.

Here, the average daily temperatures in Delhi are used and its trend is visualized using ETS Decomposition which shows whether it's a seasonal data or not.

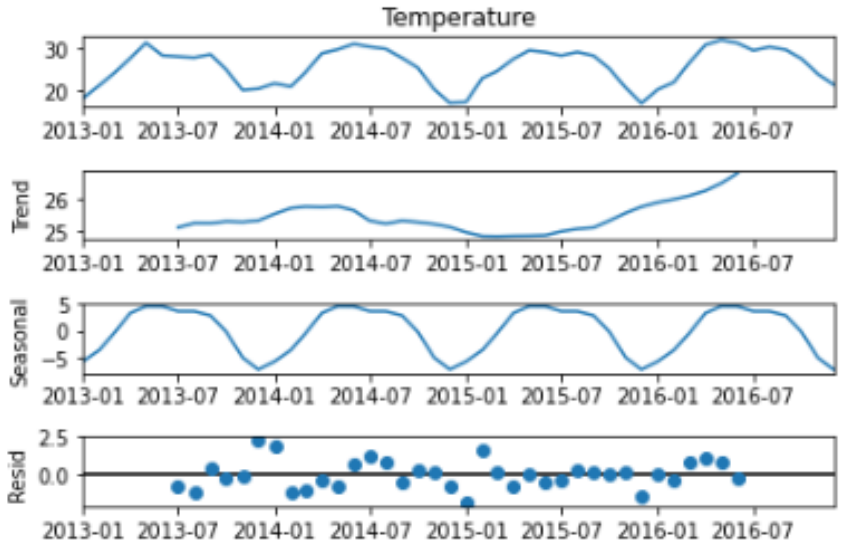

Figure 14

Clearly, from the graph shown above, we observe the following:

- The observed frequencies show same pattern from starting of 2014 year onwards.
- The trend first decreases in mid-year of 2015 but then it increases showing an uptrend.
- The data is yearly seasonal which means that it repeats its pattern after 1 year or 12 months ,thereby showing seasonality.
- When trend and seasonality are removed from the data then the leftover part is the remainder component which are the residuals.

Checking of Stationarity

Now, the next move is to check stationarity of the time series. A time series data will be stationary if its properties like mean, variance and autocorrelation do not change with respect to time. In other words, it has to be made clear that the time series data is independent of time. In general, it does not illustrate any pattern which can be predicted in the long-term.

A time series data having trend and seasonality will not be stationarity since they affect its values at different time intervals. So if trend and seasonality are present in the time series then it has to be removed before using forecasting models to make predictions.

In business activities, the time series data are not stationary since usage of various non-stationary elements such as trend and economic cycles are present but since in order to use forecasting models and to make predictions, it has to be made stationary using some methods.

Following two methods can be utilized to check stationarity of the data:

Rolling Statistics

This test is used to assess stationarity of the data using its pictorial distribution. First, a rolling standard deviation and rolling mean which are basically moving standard deviation and moving average is plotted taking a moving window of time which is 12 in our case. Then it is checked if it varies with time or not.

This method represents at a glance about the rolling statistics (mean and variance) changing over time as depicted below:

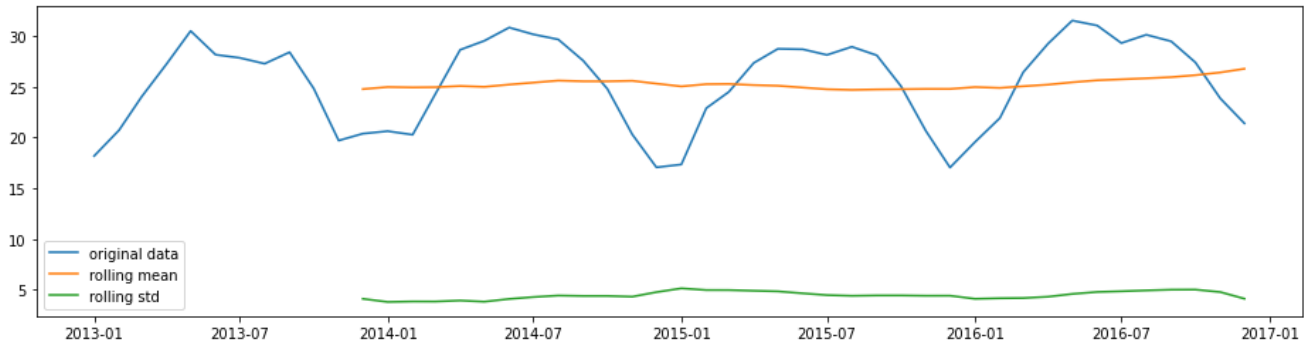

Figure 15

Although, in this case, both the rolling standard deviation and rolling mean do not illustrate any change over time but we cannot conclude its stationarity by just simply looking at the graph so this method is just for visualization purpose and the final conclusion cannot be confidently drawn from it. So, to be more sure, we will conduct Augmented Dickey-Fuller Test (ADF) test.

Augmented Dickey-Fuller Test

Augmented Dickey-Fuller test is used to check absence of serial correlation,up to a specified lag k. Now, we will set up the hypothesis-

H_o: Time-series data is non-stationary.

H_1: Time-series data is stationary.

```
def test_adfuller(df,title):

    result = adfuller(df)
    labels = ['adf value', 'p-value', 'lags', 'observations']
    out = pd.Series(result[:4], index=labels)

    for key,val in result[4].items():
        out['critical '+key]=val
    print(out)

    if result[1] > 0.5:
        print("Weak evidence against the null hypothesis")
        print("Fail to reject the null hypothesis")
        print("Data has no unit root and is non-stationary")
    elif 0 < result[1] <= 0.5:
        print("Strong evidence against the null hypothesis")
        print("Reject the null hypothesis")
        print("Data has a unit root and is stationary")
    elif result[1] == 0:
        print("P-value is 0")

test_adfuller(resample_monthly_temp,'Original Data')
```

```
adf value      -0.58286068
p-value         0.87480797
lags           10.00000000
observations   37.00000000
critical 1%    -3.62091752
critical 5%    -2.94353946
critical 10%   -2.61040024
dtype: float64
Weak evidence against the null hypothesis
Fail to reject the null hypothesis
Data has no unit root and is non-stationary
```

Presently, using the adfuller function, we acquired a dickey-fuller esteem = - 0.5829 at slack order=10. Also, as the p-value=0.8748 is more than the 0.05 degree of importance, we

acknowledge the invalid speculation. In this way, we can close both from the chart and the ADF test that the time arrangement information is non-fixed. Presently, first we need to stationarize the time arrangement information.

Stationarize the Data

Differencing is a common solution which removes the seasonal patterns in time series used to stationarize the variable. We will perform differencing using python function diff.

```
#Stationarize the Data
firstdiff = resample_monthly_temp.diff().dropna()
firstdiff.plot(figsize=(15,5))
```

Now, after differencing, we will obtain its desirable output as follows:

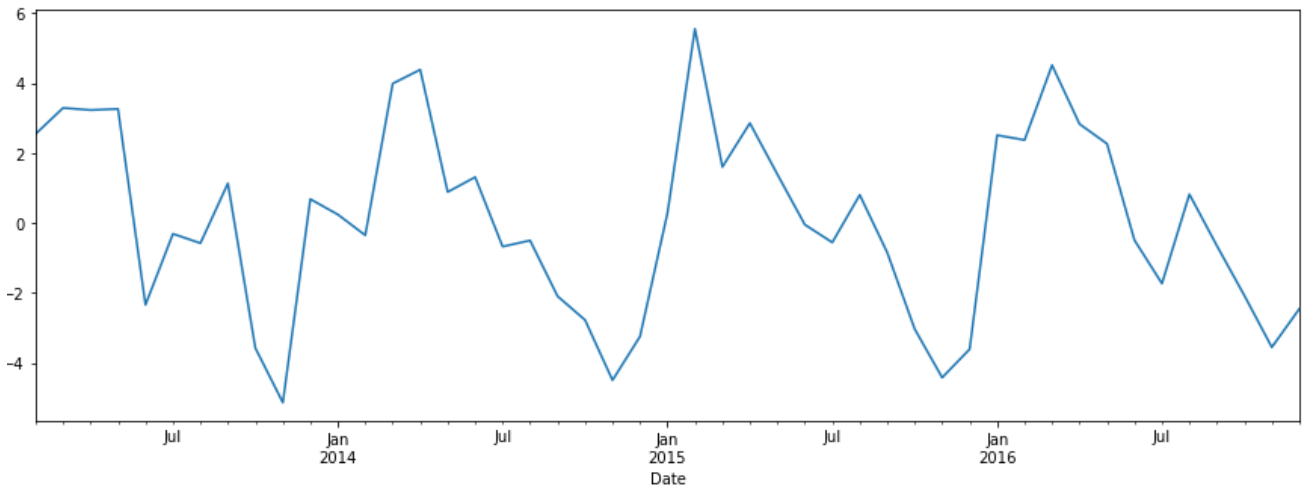

Figure 16

Though, it's difficult to interpret our data as stationary but it can be verified by again conducting the Augmented Dickey-Fuller Test (ADF) test.

Now, we need to test again for the stationarity of the differenced time series data.

We will again set up the hypothesis-

H_o: Time-series data is non-stationary.

H_1: Time-series data is stationary.

```
#Applying Again Augmented Dickey Fuller Test
test_adfuller(firstdiff,'differencing')
```

```
adf value      -6.70551367
p-value         0.00000000
lags            9.00000000
observations   37.00000000
critical 1%    -3.62091752
critical 5%    -2.94353946
critical 10%   -2.61040024
dtype: float64
Strong evidence against the null hypothesis
Reject the null hypothesis
Data has a unit root and is stationary
```

Presently, in the wake of applying the adf.test, we acquired a dickey-fuller esteem = - 6.7055 at slack order=9. Also, as the p-value=0.0000(close to 0) is underneath 0.05 degree of importance, we reject the invalid speculation. In this way, we can finish up from the ADF test that the differenced time arrangement information is fixed. Henceforth, we would now be able to move to the model assessment step.

SARIMA Modelling

1. Autocorrelation plots

Autocorrelation plots (also known as ACF or the auto correlation function) can help to get the order parameters for ARIMA model. It helps to see the correlation between points, up to and including the lag unit. So, in short, it will acknowledge us about the correlation of time series with itself. In ACF, the correlation coefficient is in y-axis whereas the x-axis illustrates the number of lags. Now, the rule is that:

- We make usage of the AR model if positive autocorrelations exists at lag 1.
- We make usage of the MA model if negative autocorrelations exists at lag 1.

Partial autocorrelation plots (PACF) displays connection between a variable and its slacks that isn't clarified by past slacks so, in short, the pacf at lag k illustrates the correlation which resulted due to the terms at shorter lags after removing the effect of any correlations. Now, here the rule is that:

- We make usage of AR(n) model if the PACF plot drops off at lag n.
- We make usage of MA model if the drop in PACF is more gradual.

Now, Applying the acf and pacf function, we will plot the graphs.

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

lag_acf = plot_acf(firstdiff, lags=15)
plt.plot()
plt.title('Autocorrelation Function')
```

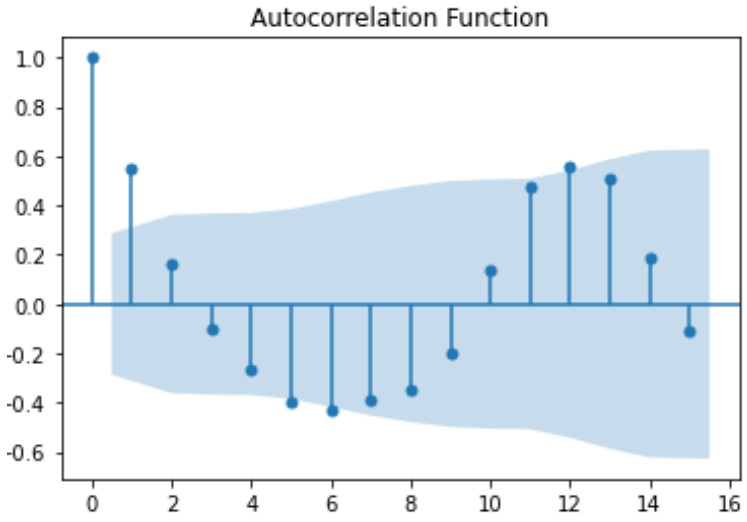

Figure 17

As at lag=1, it can be stated that positive correlation exists, so there is a AR term in the model. Also, we can spot at lag 0, the acf=1 because the value has correlation with itself only.

```
lag_pacf = plot_pacf(firstdiff, lags=15)
plt.plot()
plt.title('Partial Autocorrelation Function')
```

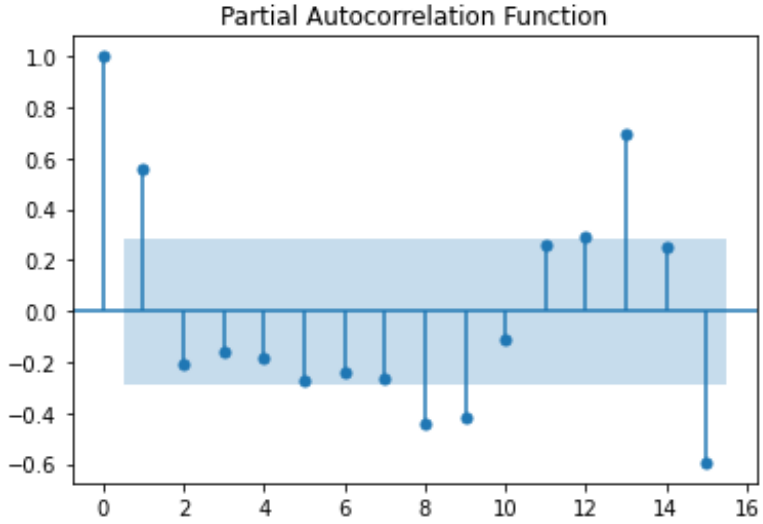

Figure 18

As the PACF plot doesn't show a gradual decrease in the pacf values pertaining to the number of lags, we can say that AR term is present in the model.

2. Parameter Selection for the SARIMA Time Series Model

When time series data is encountered in which Seasonal ARIMA has to be fitted then the first goal is to evaluate the values of ARIMA(p,d,q)(P,D,Q,S) is to make the best use of it. Here, optimal parameter values have to be selected programmatically for the model by using grid search for exploring different possible combinations of parameters. Using SARIMAX () Function from the statsmodels module, a new seasonal ARIMA model has to be fitted for each possible combination of parameters and the set of parameters which results in giving the best performance for the model will be the optimal set of parameters. The different possible combinations of parameters are generated as follows:

```
import itertools

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

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

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

print("Examples of parameter combinations for Seasonal ARIMA")
print('SARIMAX: {} X {}'.format(pdq[1],seasonal_pdq[1]))
print('SARIMAX: {} X {}'.format(pdq[1],seasonal_pdq[2]))
print('SARIMAX: {} X {}'.format(pdq[2],seasonal_pdq[3]))
print('SARIMAX: {} X {}'.format(pdq[2],seasonal_pdq[4]))
print('SARIMAX: {} X {}'.format(pdq[3],seasonal_pdq[5]))
```

The above code yields the output as:

```
Examples of parameter combinations for Seasonal ARIMA
SARIMAX: (0, 0, 1) X (0, 0, 1, 12)
SARIMAX: (0, 0, 1) X (0, 1, 0, 12)
SARIMAX: (0, 1, 0) X (0, 1, 1, 12)
SARIMAX: (0, 1, 0) X (1, 0, 0, 12)
SARIMAX: (0, 1, 1) X (1, 0, 1, 12)
```

Now, the triplets of parameters given above can be utilized for evaluation of ARIMA models on different combinations which is grid search that is used for model selection.

While evaluating and comparing models with different parameters that are fitted, they are ranked against one another based on the ability to predict future data points with more accuracy and how well it fits our data which, in turn, is checked using AIC (Akaike Information Criteria) value which, in turn, is returned with ARIMA Models fitted with usage of statsmodel. A model gives a larger AIC value to the one which fits the data very well using a lot of features and a lower AIC value to that one which utilizes less features and executes the same goodness of fit. So, the combination of parameters will be selected for the model having the least AIC value. If two or more models give the same least value then the model having fewer parameters will be selected for better fit.

In the code given below, iteration through different combinations of parameters is done and with the help of SARIMAX function in python, the corresponding Seasonal ARIMA is fitted. The parameters (p,d,q) represents the order argument and (P, D, Q, S) represents seasonal components of the SARIMA model. When each SARIMAX() model is fitted, then their respective AIC values are generated.

```
warnings.filterwarnings("ignore") # specify to ignore warning messages

for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(train,
                                            order=param,
                                            seasonal_order=param_seasonal,
                                            enforce_stationarity=False,
                                            enforce_invertibility=False)
            results = mod.fit()
            print('SARIMAX{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
        except:
            continue
```

Since this code yields a plenty of different parameter combinations, so it often yields some parameter combinations which may results in numerical misspecifications which can produce plenty of warning messages and also may leads to error. Therefore, these warning messages are disabled to steer clear of such error messages and ignore those parameter combinations that causes such problems.
Since there are large number of different combinations, so this code may delay to generate the output.

```
SARIMAX(0, 0, 0)x(0, 0, 0, 12)12 - AIC:10482.453598824479
SARIMAX(0, 0, 0)x(0, 0, 1, 12)12 - AIC:9066.687679182636
SARIMAX(0, 0, 0)x(0, 0, 2, 12)12 - AIC:8168.512797683074
SARIMAX(0, 0, 0)x(0, 1, 0, 12)12 - AIC:5808.237003079088
SARIMAX(0, 0, 0)x(0, 1, 1, 12)12 - AIC:5742.454880649962
                    .
                    .
                    .
                    .
                    .
SARIMAX(2, 1, 2)x(2, 0, 1, 12)12 - AIC:4418.309759706845
SARIMAX(2, 1, 2)x(2, 0, 2, 12)12 - AIC:4396.635543265751
SARIMAX(2, 1, 2)x(2, 1, 0, 12)12 - AIC:4325.9973304941395
SARIMAX(2, 1, 2)x(2, 1, 1, 12)12 - AIC:4423.113213729766
SARIMAX(2, 1, 2)x(2, 1, 2, 12)12 - AIC:4487.371206165735
```

The above output highlights that SARIMAX(1,1,1)x(2,1,0,12) gives the lowest AIC value of 4317.5489. Therefore, this model is selected out of all the other models.

3. Fitting SARIMA Time Series Model

With the help of grid search, the set of parameters which produces the best fit has been successfully selected for the model. Now, the next approach is to analyzation of this model.

```
model = sm.tsa.SARIMAX(train,order=(1,1,1), seasonal_order=(2,1,0,12), m=12)
results = model.fit()
results.summary()
```

SARIMAX Results

Dep. Variable:	Temperature	**No. Observations:**	1117
Model:	SARIMAX(1, 1, 1)x(2, 1, [], 12)	**Log Likelihood**	-2210.023
Date:	Sat, 16 Apr 2022	**AIC**	4430.047
Time:	14:25:23	**BIC**	4455.080

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.5278	0.043	12.249	0.000	0.443	0.612
ma.L1	-0.8281	0.032	-25.842	0.000	-0.891	-0.765
ar.S.L12	-0.6580	0.026	-25.166	0.000	-0.709	-0.607
ar.S.L24	-0.3365	0.026	-13.162	0.000	-0.387	-0.286
sigma2	3.1891	0.097	32.908	0.000	2.999	3.379

Table 4

The result summary of Seasonal ARIMA displays many information, some of which are self-explanatory as follows:

- It shows the dependent variable which we have to predict.
- It shows which type of model we are using.
- It shows the date of running the model.
- It shows the time of running the model.
- It shows the count of observations in the data.

Here, the dependent variable is the temperature which has to be predicted, ar.L1 and ma.L1 are the lag variables, ar.S.L12 and as.S.L24 are the seasonal lag variables and sigma2 or epsilon is the error term.

The models can compared with another model with the help of Log-Likelihood, AIC and BIC as Log-Likelihood helps to identify which distribution will best fit the data, AIC and BIC helps in determining the strength and complexity of the model.

The weights of each picture is highlighted above in the coef column and also since the p-values of all the lag variables are less than level of significance i.e. 0.05 so they are statistically significant.

4. Model Diagnostics

Now, the model diagnostics has to be performed to investigate any unusual behaviour if it is there.

```
model_aic.plot_diagnostics(figsize=(16,8))
plt.show()
```

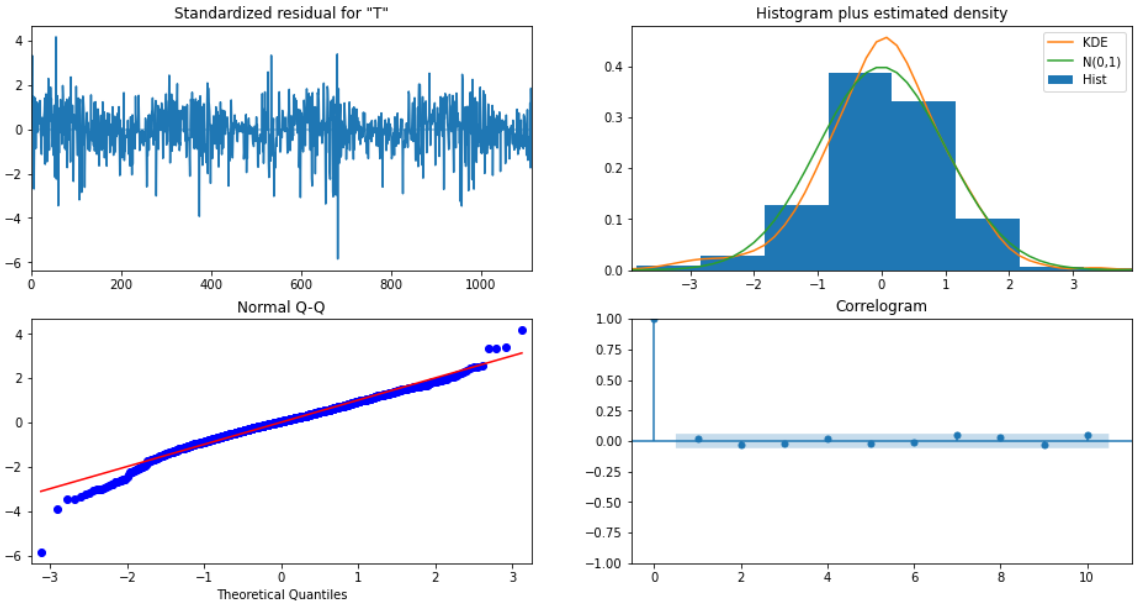

Figure 19

Our main goal is to check whether no correlation exist in the residuals of our model or not and whether it is normally distributed with zero mean. If, in a case, these properties are not satisfied then it will give an indication that it should be further improved.

In this demonstration, the model diagnostic represents the following:

- **Top left:** Here, it depicts the fluctuations of residual errors around a mean of zero but do not depicts any seasonality but represents a variance which is uniform in a range of -4 to 4.

- **Top Right:** It highlights mainly the red coloured KDE line which follows closely with the normal distribution having mean and variance as 0 and 1 respectively. Normal distribution having mean 0 is represented by the density plot.

- **Bottom left:** The distribution is very low skewed because many blue dots are over the red line.

- **Bottom Right:** The ACF plot, here, depicts that the residual errors are not autocorrelated which shows that residuals in time series have low correlation with lagged versions of itself.

These observations, therefore, proves that our model fit which is satisfactory enough to understand our data of time series and further forecast values of the future.

Using these values, the prediction is done with usage of the SARIMA model of the year 2016.

```
test.plot(legend=True,figsize=(12,6))
predictions.plot(legend=True)
```

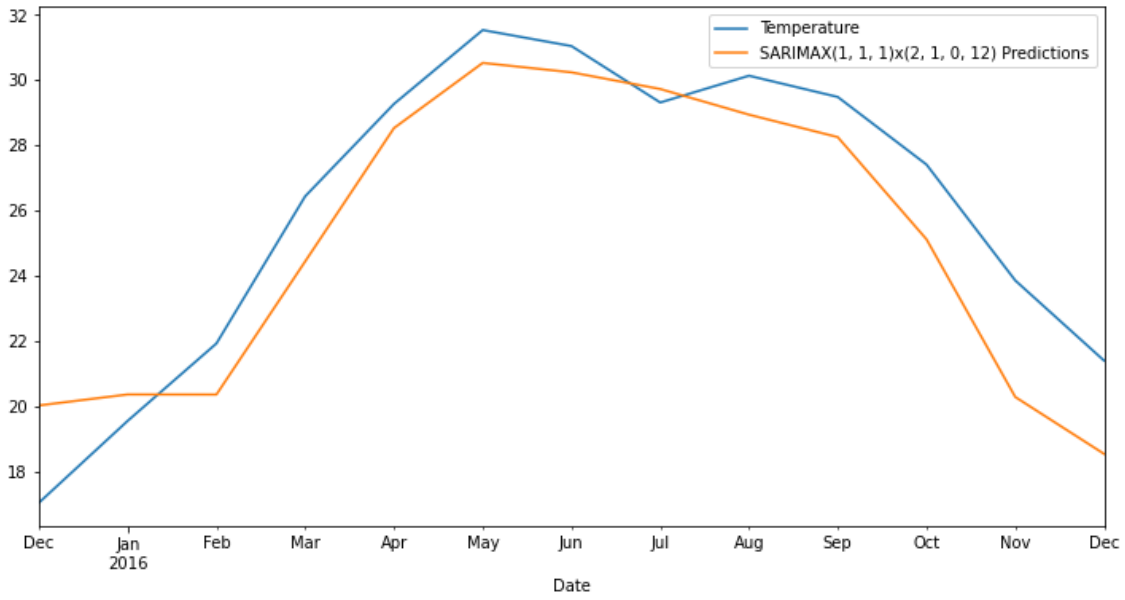

Figure 20

- It can be observed that the predicted values are near the obseved values of the Temperature.
- It is clear that the predicted values exceeds the observed values in december 2015 to mid of january 2016 and at the starting of the month of july 2016.
- It does not depict any kind of trend or seasonality.

5. Checking Accuracy of the Model

In order to check the accuracy of the model, MSE(Mean Squared Error) which helps in summarizing the average of the error which are present in the model for the forecasts. It is found by taking the distance from each predicted to the true values and then it has to be squared to steer clear of the negative sign since they will cancel out the positive signs when overall mean is computed.

It is also useful to quantify the accuracy of our forecasts. We will use the MSE (Mean Squared Error), which summarizes the average error of our forecasts. For each predicted value, we compute its distance to the true value and square the result. The results need to be squared so that positive/negative differences do not cancel each other out when we compute the overall mean.

```
start=len(train)
end=len(train)+len(test)-1
predictions = results.predict(start=start, end=end).rename('SARIMAX(1, 1, 1)x(2, 1, 0, 12) Predictions')

from sklearn.metrics import mean_squared_error

error = mean_squared_error(test, predictions)
print('Mean Squared Error ',error)
```

```
Mean Squared Error  3.6550202169899517
```

The MSE comes out to be 3.6550 which is not very close to 0 so it does not represent a perfect accuracy but since it is highly biased for higher values so a better error rate has to be computed which is RMSE (Root Mean Squared Error) which is better when large error values are present.

```
from statsmodels.tools.eval_measures import rmse

error = rmse(test, predictions)
print('Root Mean Squared Error ',error)
```

```
Root Mean Squared Error  1.9118107168310234
```

Here, the RMSE comes out to be 1.9118 which can be accepted since when dealing with the Weather Forecasting, there are many attributed which can affect the weather such as wind speed, humidity, atmospheric pressure, etc.)

6. Producing and Visualizing Forecasts

Since the RMSE is acceptable, it indicates that the Seasonal ARIMA model is used to forecast future values. The forecasted values can be found using the code as follows:

```
fcast = results.predict(len(resample_monthly_temp),len(resample_monthly_temp)+12,typ='levels').rename('SARIMAX(1,1,1)(2,1,0,12) Forecast')
fcast
```

```
2017-01-01   18.79263195
2017-02-01   22.39281546
2017-03-01   24.73921133
2017-04-01   28.14020944
2017-05-01   29.16061826
2017-06-01   29.85100732
2017-07-01   29.22622619
2017-08-01   29.66811034
2017-09-01   28.15781589
2017-10-01   25.27559532
2017-11-01   20.89489846
2017-12-01   19.14780243
2018-01-01   19.47279986
Freq: MS, Name: SARIMAX(1,1,1)(2,1,0,12) Forecast, dtype: float64
```

Now, these value can be plotted in the graph as follows:

```
resample_monthly_temp.plot(legend=True,figsize=(12,6))
fcast.plot(legend=True)
```

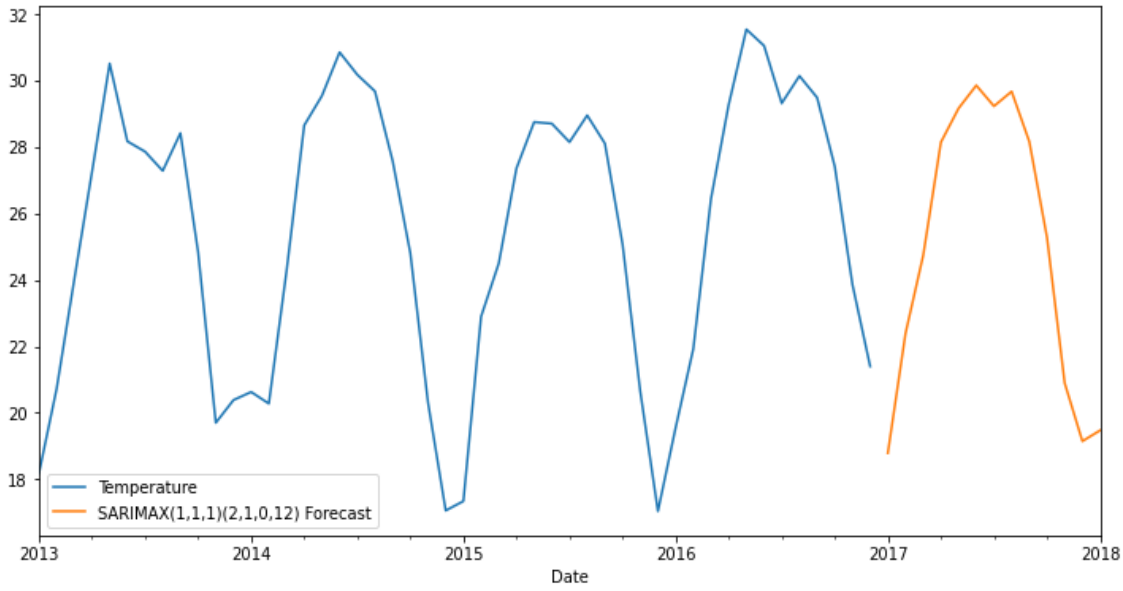

Figure 21

Here, the forecasted values for the year 2018 are depicted in the figure above. Now, both the observed and forecasted values can be used further to assimilate the time series and can be foreseen what to expect. Our forecasted values represents that the temperature increases first from the starting of the year, reach its peak and gradually decreases at the year-end just like the previous observed year values.

CONCLUSION

Therefore, we observed some useful assimilations from the time series analysis. The SARIMA model is used since seasonality is present in the model. The Temperature increases at the beginning of the year, reaches its peak in the middle of the year and then gradually decreases at the year-end. The ARIMA model cannot be used since it does not hold seasonal data and this time series contains a repetitive cycle. The optimal seasonal arima model for the time series data is ARIMA (1, 1, 1) x (2, 1, 0, 12). The RMSE of the model is 1.9118 which is acceptable since when dealing with the Weather Forecasting, there are many attributes which can affect

the weather such as wind speed, humidity, atmospheric pressure, etc. Also, the SARIMA model does seem to provide forecasted values for the year 2018 whose temperature increases or decreases just like the previous observed year values.

REFERENCES

1. An Introduction to Statistical Learning – By Gareth James
2. Fundamentals of Applied Statistics – By Gupta and Kapoor
3. https://www.kaggle.com/andreshg/timeseries-analysis-a-complete-guide/notebook
4. https://towardsdatascience.com/time-series-forecasting-arima-models-7f221e9eee06
5. https://medium.com/@cmukesh8688/why-is-augmented-dickey-fuller-test-adf-test-so-important-in-time-series-analysis-6fc97c6be2f0