Delhi Weather Analysis and Temperature Prediction

Weather analysis and Temperature prediction is a very important part that needs to be taken control of for a better understanding of our environment and proper prediction, which can greatly help industries like agriculture and many others for a more informed and effective decision making.

Data Source: https://www.kaggle.com/mahirkukreja/delhi-weather-data

Initial Data looked something like the table below:

datetime_utc	_conds	_dewptm	_fog	_hail	_heatindexm	_hum	_precipm	_pressurem	_rain	_snow	_tempm	_thunder	_tornado	_vism	_wdird	_wdire
1996-11-01 11:00:00	Smoke	9.0	0	0	NaN	27.0	NaN	1010.0	0	0	30.0	0	0	5.0	280.0	West
1996-11-01 12:00:00	Smoke	10.0	0	0	NaN	32.0	NaN	-9999.0	0	0	28.0	0	0	NaN	0.0	North
1996-11-01 13:00:00	Smoke	11.0	0	0	NaN	44.0	NaN	-9999.0	0	0	24.0	0	0	NaN	0.0	North
1996-11-01 14:00:00	Smoke	10.0	0	0	NaN	41.0	NaN	1010.0	0	0	24.0	0	0	2.0	0.0	North
1996-11-01 16:00:00	Smoke	11.0	0	0	NaN	47.0	NaN	1011.0	0	0	23.0	0	0	1.2	0.0	North

Now, we needed to extract columns of our concern and check for data integrity of those columns, which is done as given below:

datetime_utc	condition	humidity	temperature	_pressurem	dew
1996-11-01 11:00:00	Smoke	27.0	30.0	NaN	NaN
1996-11-01 12:00:00	Smoke	32.0	28.0	NaN	NaN
1996-11-01 13:00:00	Smoke	44.0	24.0	NaN	NaN
1996-11-01 14:00:00	Smoke	41.0	24.0	NaN	NaN
1996-11-01 16:00:00	Smoke	47.0	23.0	NaN	NaN

We find number of rows with no values in each column and remove the columns that have more than 70% of values as NaN.

```
weather_df.isnull().sum()
```

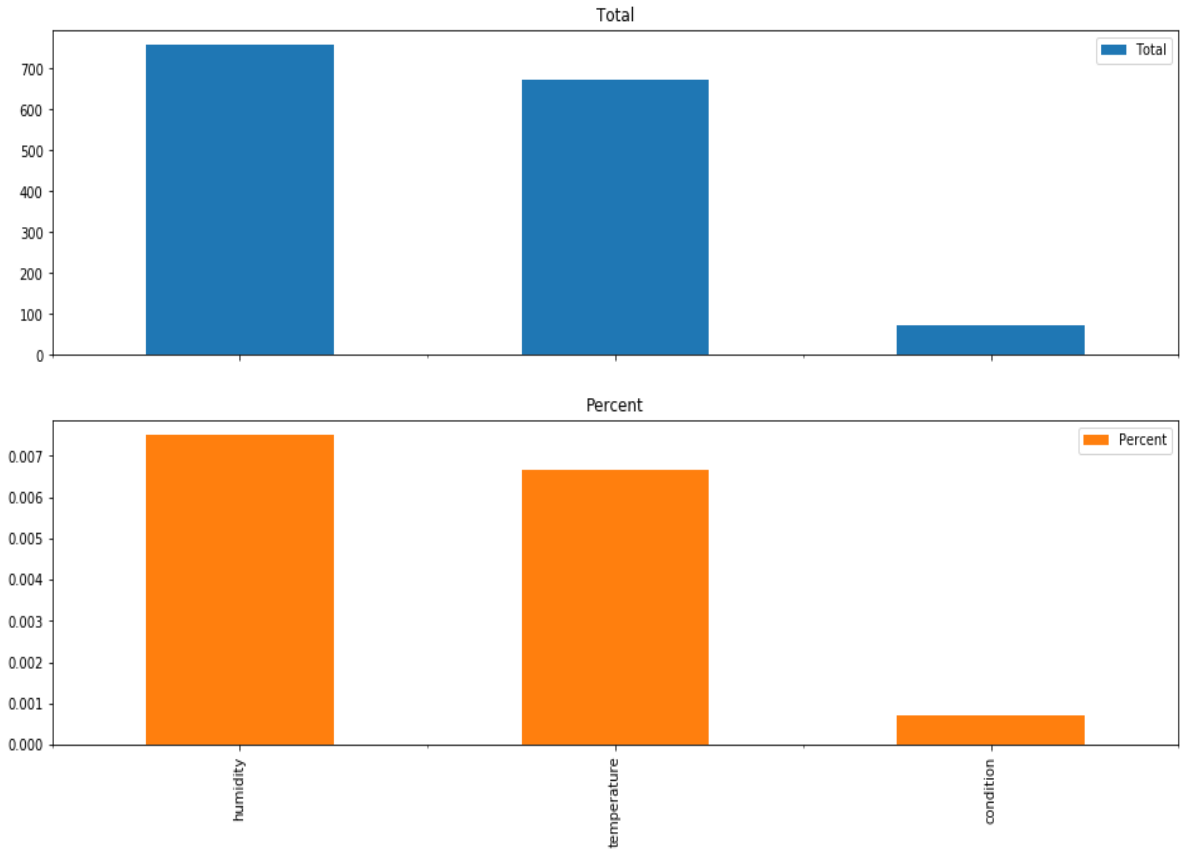
```
condition          72
humidity          757
temperature       673
_pressurem     100990
dew            100990
dtype: int64
```

```
weather_df.count()
```

```
condition      100918
humidity       100233
temperature    100317
_pressurem          0
dew                 0
dtype: int64
```

Data quality assessment was performed using the isnull().sum() and count() functions to make sure the data we need is present there, and remove any columns that do not have sufficient data. As we can see, pressure and dew columns do not consist of any data, thus, they are removed from the table.

Next, we check for the columns available, i.e. , condition, humidity and temperature and find how much % of the data is missing in those columns. This can be seen in below graph:



As can be seen, **very less amount of data is missing**, thus **data imputation method** of replacing last known valid value is replace with missing value and table then looks something like as shown below:

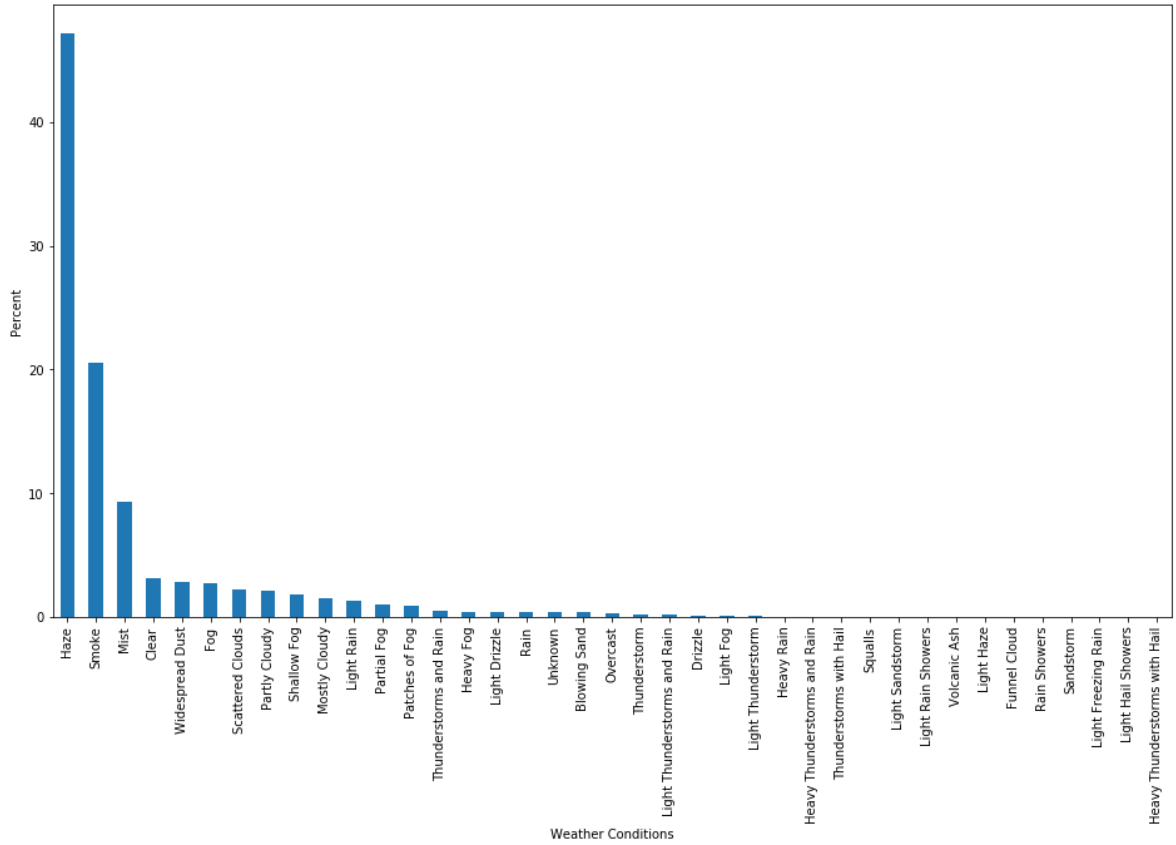
	humidity	temperature
count	100990.000000	100990.000000
mean	57.957422	25.438222
std	23.821218	8.487994
min	4.000000	1.000000
25%	39.000000	19.000000
50%	59.000000	27.000000
75%	78.000000	32.000000
max	243.000000	90.000000

Now, we find remove the rows with temperature values >= 55°C and with humidity values > 100. New table looks something as shown below:

	humidity	temperature
count	100983.000000	100983.000000
mean	57.955309	25.436361
std	23.805467	8.482944
min	4.000000	1.000000
25%	39.000000	19.000000
50%	59.000000	27.000000
75%	78.000000	32.000000
max	100.000000	47.000000

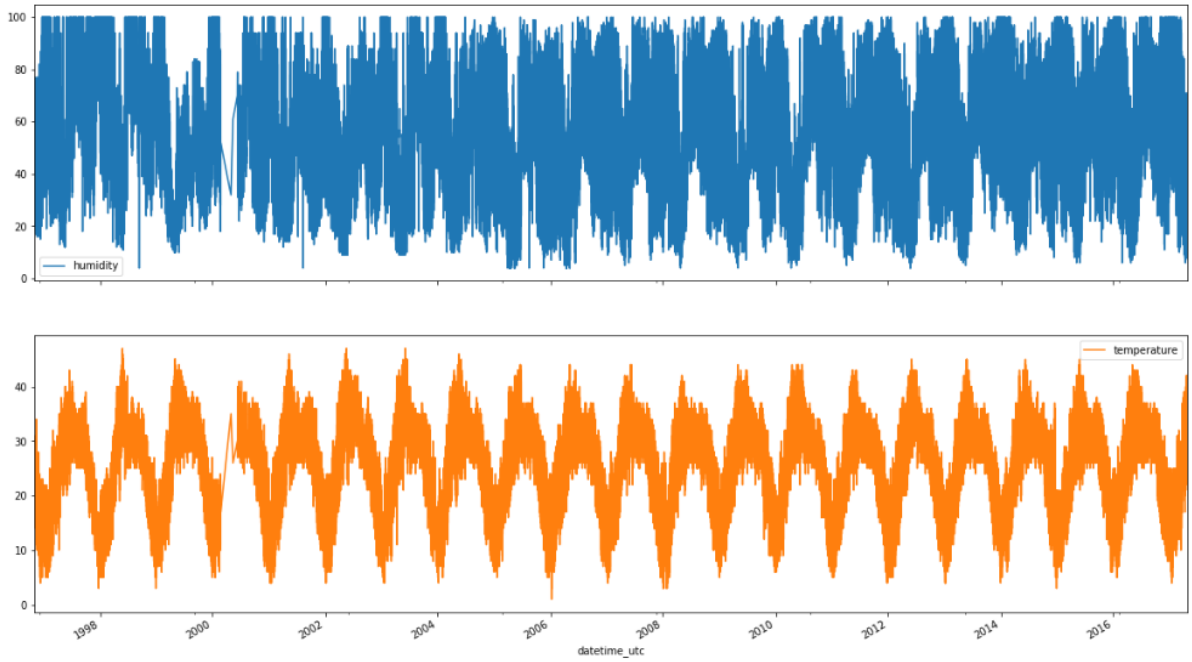
Next, **exploratory data analysis** is performed, whose results are shown below:

1. Finding most common weather condition:



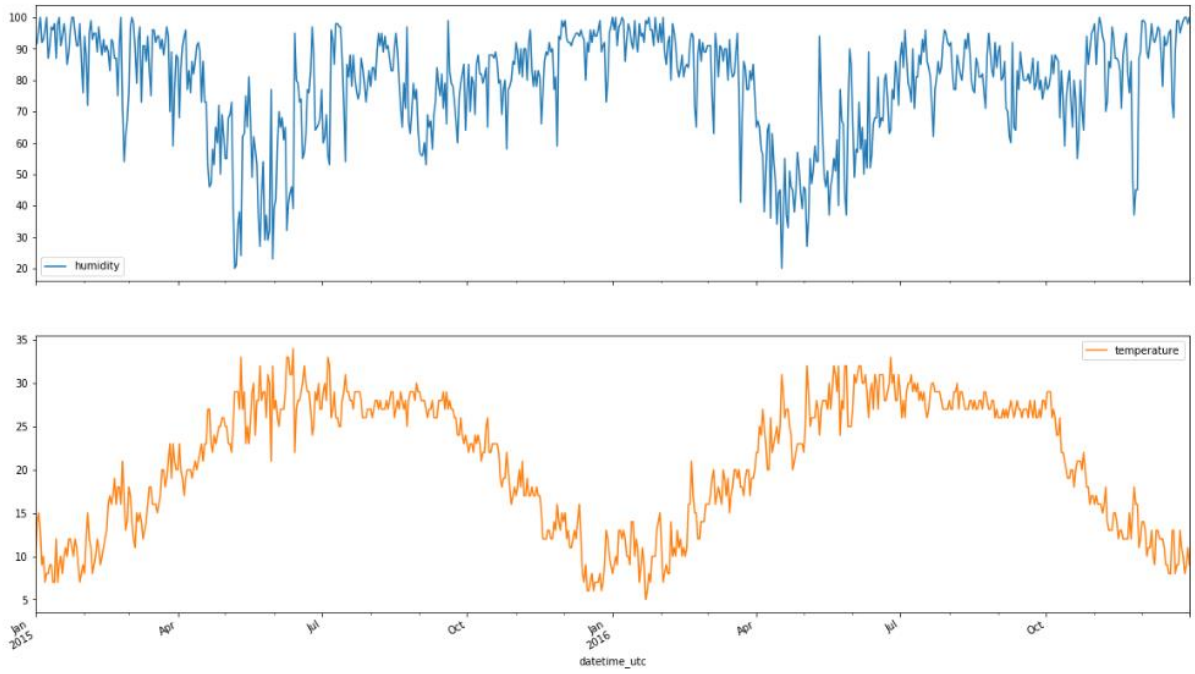
As we can see, since hazy and Smoky weather is the most common, we can clearly and undoubtedly infer the high pollution levels in Delhi

2. Temperature and Humidity logs of all time:



We can clearly see that the temperature and humidity have a particular pattern they follow every year. This is called seasonal behaviour.

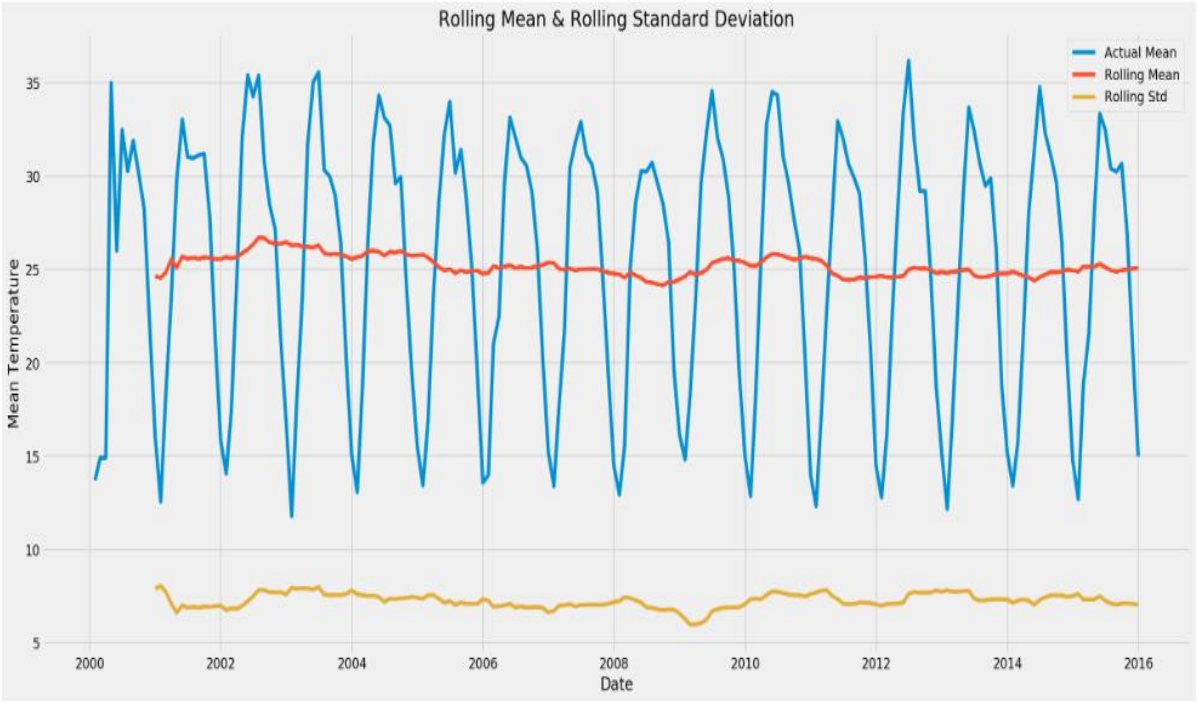
3. Next we focus onto 2 recent years data (2015 and 2016) to understand the monthly trend of temperature and humidity:



We can clearly see that temperature rises from start of the year till the mid-year and then falls back to lowest point towards the end of the year.

Humidity follows the exact opposite trend, it falls to the lowest point till the mid-year and then rises back to highest value by the end of the year.

In the next step, we **check for the stationarity of the data** by checking the rolling mean and rolling standard graphs and performing Augmented Dicky-Fuller test:

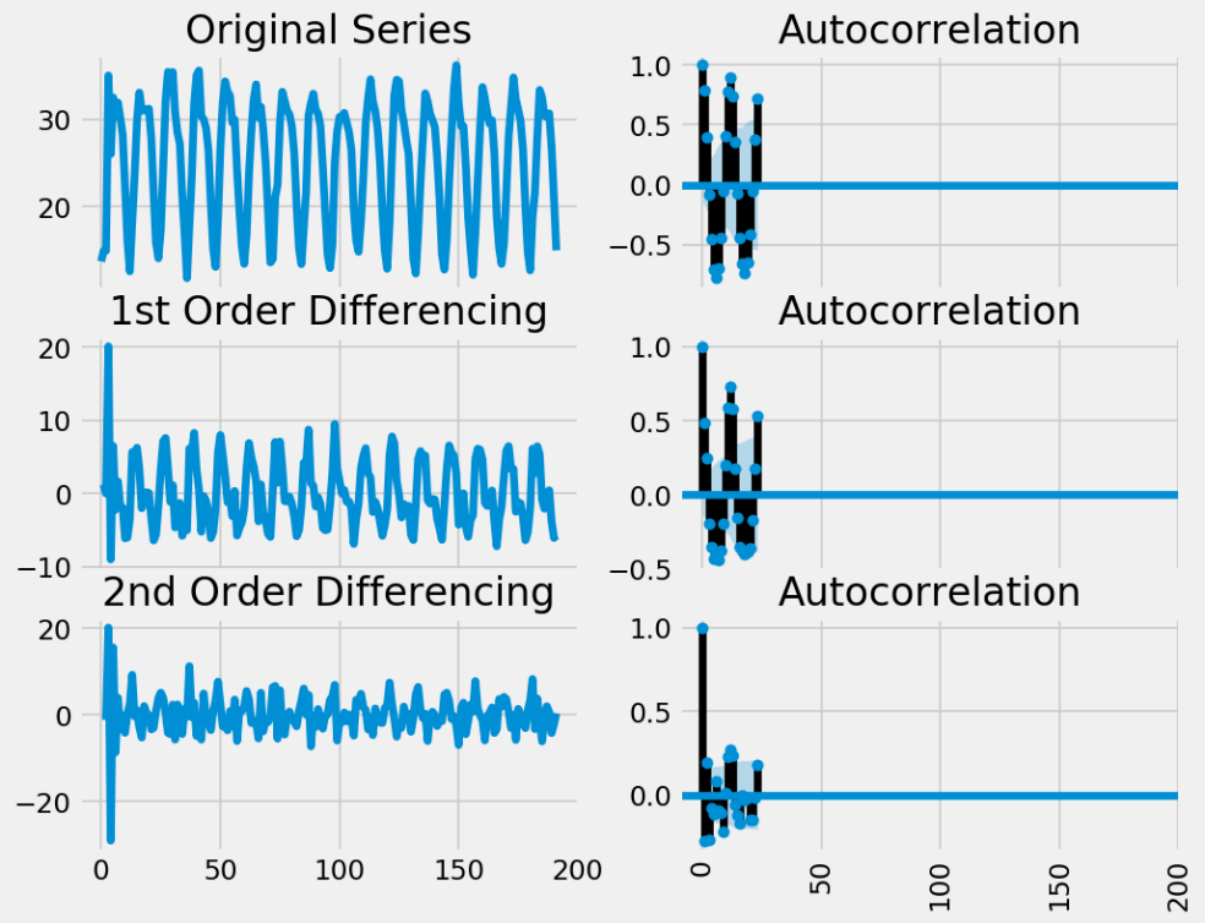



```
Test statistic:  -2.126995250298034
Critical Values: {'1%': -3.4674201432469816, '5%': -2.877826051844538, '10%': -2.575452082332012}
```

From the graph, it is evident that the rolling mean and rolling standard graphs are almost stationary and the test-statistic value is larger than the critical values, thus we can keep the value of “d” as zero (0) in the **ARIMA model**, which we will be using for model development and prediction of temperature values.

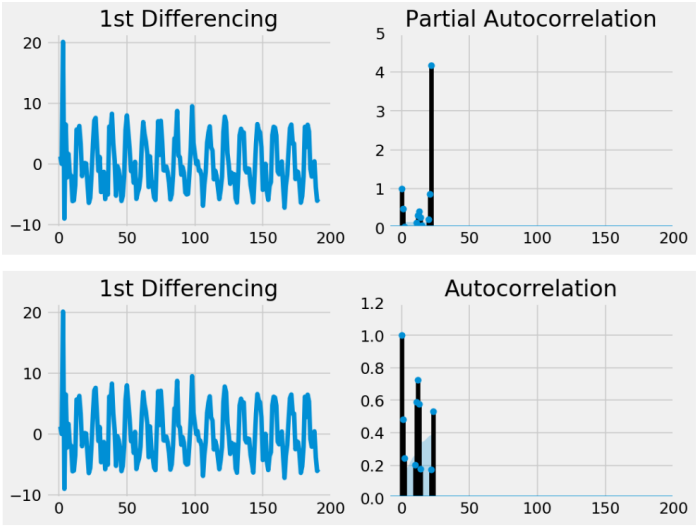
We use the **ARIMA model** because it is one of the best models available to perform time-series analysis and gives very accurate results.

In case of non-stationarity results of the rolling mean and rolling standard deviation, or the test-statistic value, we could have gone for either **Decomposing** or **Differencing** methods to make them stationary. For ease of demonstration, we will be showcasing the **Differencing** method as it is quite straight-forward.



Above image demonstrates how the **Differencing** method is performed. Since we get very accurate results in the original series, we won't be performing the differencing method.

Next we plot the Auto-Correlation and Partial Auto-Correlation graphs for the 1st Differencing of original series. These terms are explained later.



Now, we will explain the model we have decided to deploy for predicting the temperatures.

ARIMA Model Deployment

Timeseries Analysis (ARIMA Model)

For prediction we are going to use one of the most popular model for time series, Autoregressive Integrated Moving Average (ARIMA) which is a standard statistical model for time series forecast and analysis. An ARIMA model can be understood by outlining each of its components as follows:

Autoregression (AR) - refers to a model that shows a changing variable that regresses on its own lagged, or prior, values. The notation AR(p) indicates an autoregressive model of order p.

Example—If p is 3 the predictor for X(t) will be

$$\mathbf{X(t) = \mu + X(t-1) + X(t-2) + X(t-3) + \varepsilon t}$$

Where ε is error term.

Integrated (I) - represents the differencing of raw observations to allow for the time series to become stationary, i.e., data values are replaced by the difference between the data values and the previous values. Moving average (MA) - incorporates the dependency between an observation and a residual error from a moving average model applied to lagged observations.

Example—If q is 3 the predictor for X(t) will be

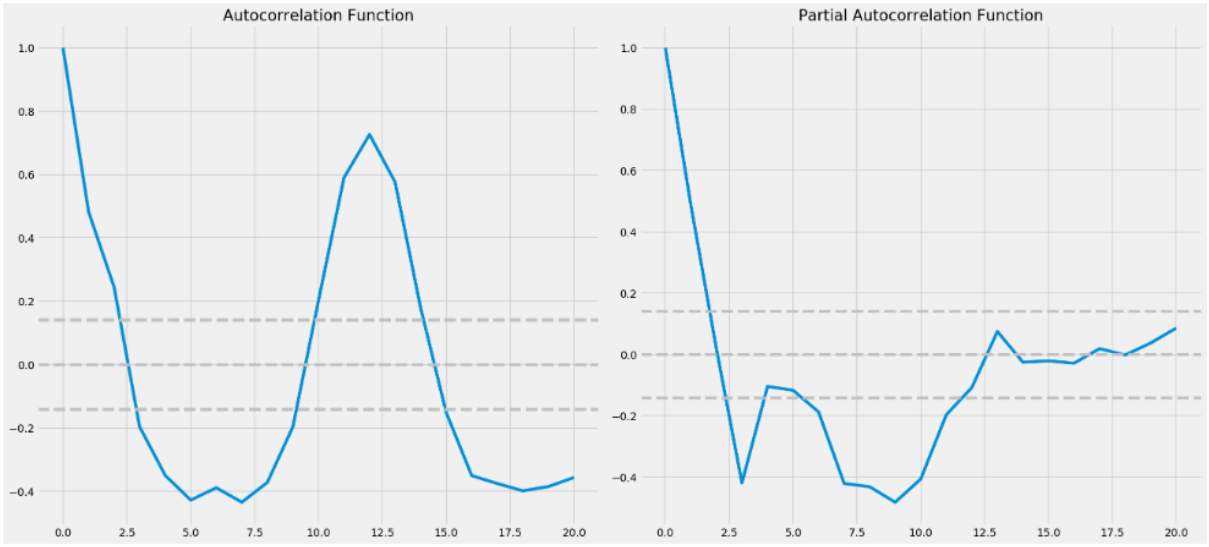
$$\mathbf{X(t) = \mu + \varepsilon t + \theta 1.\varepsilon(t-1) + \theta 2.\varepsilon(t-2) + \theta 3.\varepsilon(t-3)}$$

Here instead of difference from previous term, we take errer term (ε) obtained from the difference from past term Now we need to figure out the values of p and q which are parameters of ARIMA model. We use below two methods to figure out these values -

Autocorrelation Function (ACF): It just measures the correlation between two consecutive (lagged version). example at lag 4, ACF will compare series at time instance t1…t2 with series at instance t1–4…t2–4

Partial Autocorrelation Function (PACF): is used to measure the degree of association between X(t) and X(t-p).

Next, we plot the correlation function and the auto-correlation functions.



In this graph,

The grey dotted line are confidence intervals which we are going to use to find out the value of p and q.

p - The point where PACF crosses the upper confidence level. In our case it seems to be 2. So we will take p = 2.

q - The point where ACF crosses the upper confidence level. In our case it seems to be 2. So we will take q = 2.

d - Number of non-seasonal differences needed for stationarity. In this case we are going to take it as 0, since this series is already stationary.

Now we are going fit time series for **ARIMA** Models. We will **compare performance on the basis of RSS score and at last prefer the best one** as the best RSS score indicates the best fitted model which is neither under-fitted not over-fitted.

ARMA Model Results

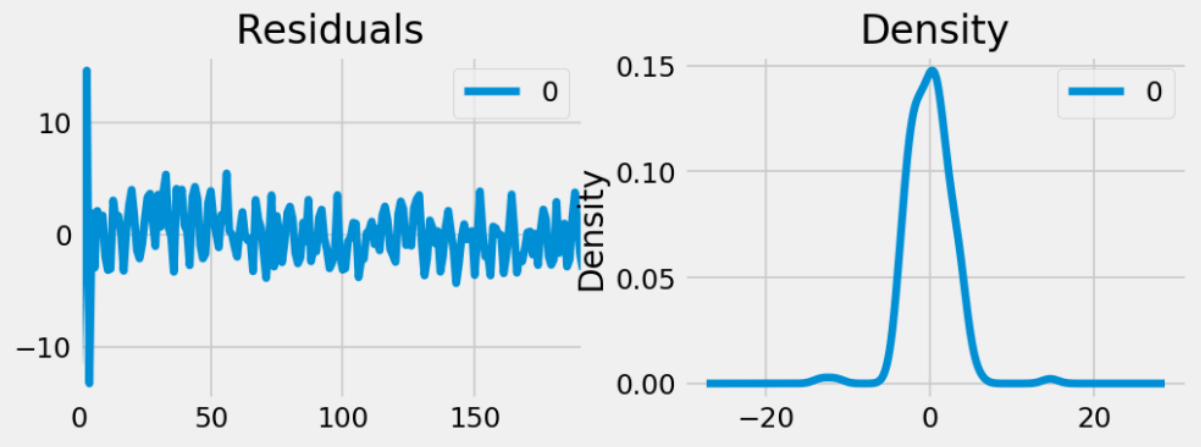
Dep. Variable:	y	No. Observations:	192
Model:	ARMA(2, 2)	Log Likelihood	-454.355
Method:	css-mle	S.D. of innovations	2.552
Date:	Mon, 29 Jun 2020	AIC	920.709
Time:	22:12:50	BIC	940.254
Sample:	0	HQIC	928.625

	coef	std err	z	P>|z|	[0.025	0.975]
const	25.1917	0.119	211.045	0.000	24.958	25.426
ar.L1.y	1.6785	0.024	69.834	0.000	1.631	1.726
ar.L2.y	-0.9519	0.023	-41.163	0.000	-0.997	-0.907
ma.L1.y	-0.9726	0.098	-9.919	0.000	-1.165	-0.780
ma.L2.y	0.1453	0.090	1.618	0.107	-0.031	0.321

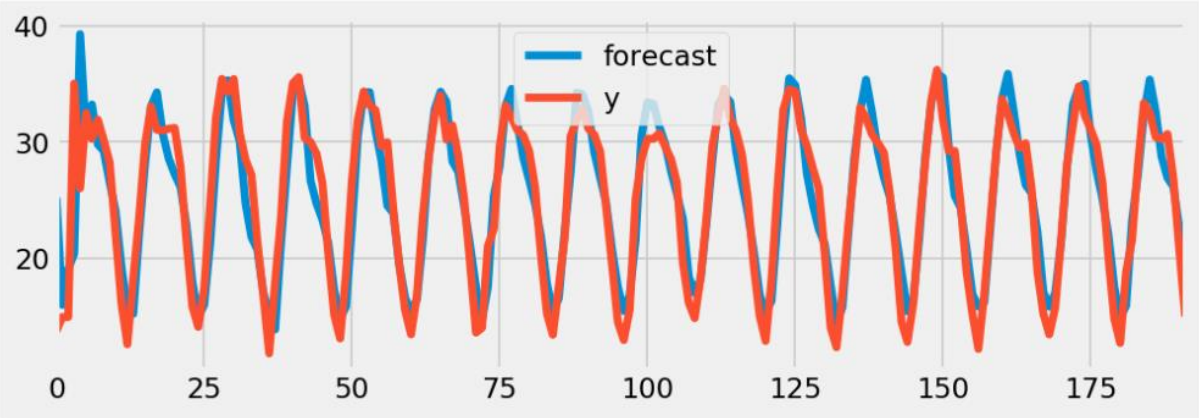
Roots

	Real	Imaginary	Modulus	Frequency
AR.1	0.8816	-0.5227j	1.0250	-0.0852
AR.2	0.8816	+0.5227j	1.0250	0.0852
MA.1	1.2685	+0.0000j	1.2685	0.0000
MA.2	5.4265	+0.0000j	5.4265	0.0000

Next, we plot the residuals of the ARIMA model as shown below:



Actual vs Fitted Graph:



Forecast Temperatures vs Actual Temperatures: