EE798 ASSIGNMENT

AVINASH SHUKLA

210236

We are provided with a dataset of several pollutants in the form of a csv file in which we must perform time series analysis and implement proper ARIMA model for the pollutants.

I have plotted the time series model and implemented appropriate ARIMA model for the pollutant PM10 which can be extended to other pollutants also by using similar analysis on the software Google Collab.

Here is the implementation of code as well as outputs:

```
pip install pmdarima
import pandas as pd
import numpy as np

# reading our csv file into a dataframe df
df = pd.read_csv('/content/drive/MyDrive/FISA.csv',index_col='From'    ,parse_dates=True)

 #plotting the graph for pollutant PM10
df['PM10 (µg/m3)'].plot(figsize=(20,6), color = 'red')

#dropping na values from the dataframe df
df2=df.dropna()

#plotting the graph for df2 of pollutant PM10
df2['PM10 (µg/m3)'].plot(figsize=(20,6), color = 'red')

#implement ad_test function for checking the stationarity of the data
from statsmodels.tsa.stattools import adfuller
def ad_test(dataset):
    df2test = adfuller(dataset, autolag = 'AIC')
    print("1. ADF : ",df2test[0])
    print("2. P-Value : ", df2test[1])
    print("3. Num Of Lags : ", df2test[2])
    print("4. Num Of Observations Used For ADF Regression:",      df2test[3])
    print("5. Critical Values :")
    for key, val in df2test[4].items():
        print("\t",key, ": ", val)
ad_test(df2['PM10 (µg/m3)'])

#choosing best arima model for our dataframe
from pmdarima import auto_arima
stepwise_fit = auto_arima(df2['PM10 (µg/m3)'], trace=True,
suppress_warnings=True)
```

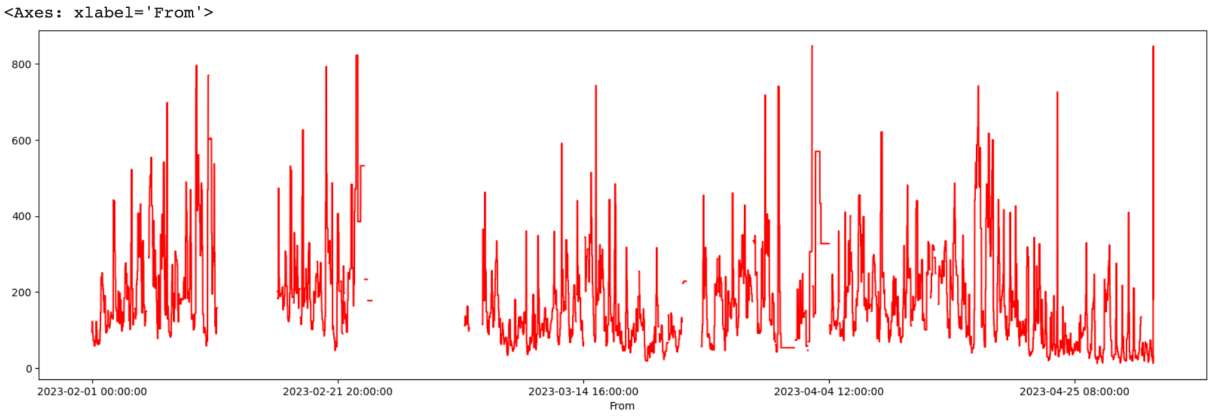
Below are the outputs that we get for the above code :

Plot of df dataset which has missing values:

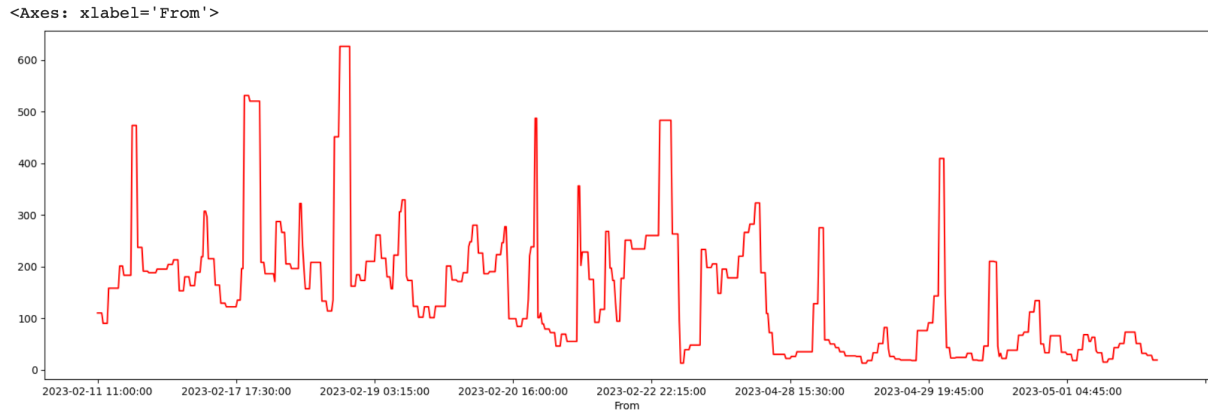
EE798 ASSIGNMENT

AVINASH SHUKLA

210236



Plot of df2 dataset which has no missing values:



Output of ad_test function which does stationary test for our pollutant data:

```
1. ADF :  -4.199738418758473
2. P-Value :  0.0006595723184032765
3. Num Of Lags :  12
4. Num Of Observations Used For ADF Regression: 753
5. Critical Values :
        1% :  -3.4390641198617864
        5% :  -2.8653859408474482
        10% :  -2.5688179819544312
```

Here we can infer from the P value that our dataset is stationary or not. If P value is less than 0.0005 our data is not stationary, and we need to implement ARIMA model of appropriate order for correct forecasting. If P value is greater than 0.005 our dataset is stationary.

As our dataset is not stationary, so we need to run auto_arima test which we imported from PMDARIMA library.

EE798 ASSIGNMENT

AVINASH SHUKLA

210236

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=inf, Time=4.87 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=8104.874, Time=0.11 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=8105.778, Time=0.08 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=8105.593, Time=0.62 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=8102.879, Time=0.08 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=8098.878, Time=0.76 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=8100.872, Time=2.32 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=8100.871, Time=1.53 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=8103.360, Time=0.98 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=8103.773, Time=0.44 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=8096.883, Time=1.00 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=8103.598, Time=0.69 sec
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=8103.782, Time=0.22 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=8098.876, Time=2.02 sec
 ARIMA(1,1,2)(0,0,0)[0]             : AIC=8098.875, Time=1.56 sec
 ARIMA(0,1,2)(0,0,0)[0]             : AIC=8101.365, Time=0.43 sec
 ARIMA(2,1,0)(0,0,0)[0]             : AIC=8101.778, Time=0.17 sec
 ARIMA(2,1,2)(0,0,0)[0]             : AIC=8065.548, Time=1.27 sec
 ARIMA(3,1,2)(0,0,0)[0]             : AIC=8065.189, Time=2.39 sec
 ARIMA(3,1,1)(0,0,0)[0]             : AIC=8099.732, Time=1.38 sec
 ARIMA(4,1,2)(0,0,0)[0]             : AIC=8066.827, Time=2.88 sec
 ARIMA(3,1,3)(0,0,0)[0]             : AIC=8067.651, Time=6.62 sec
 ARIMA(2,1,3)(0,0,0)[0]             : AIC=8065.319, Time=2.32 sec
 ARIMA(4,1,1)(0,0,0)[0]             : AIC=8101.393, Time=1.97 sec
 ARIMA(4,1,3)(0,0,0)[0]             : AIC=8068.919, Time=5.71 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=inf, Time=6.91 sec

Best model:  ARIMA(3,1,2)(0,0,0)[0]
Total fit time: 49.575 seconds
```

After running this test it shows that the best ARIMA model if of following order(3,1,2) which has the minimum AIC value. We will implement this ARIMA model which is shown in the next snippet of code.

```python
#splitting the dataset into train and test
print(df2.shape)
train=df2.iloc[:-30]
test=df2.iloc[-30:]
print(train.shape,test.shape)

#Implementing the ARIMA model for our dataset
model = ARIMA(train['PM10 (µg/m3)'], order=(3, 1, 2))
model = model.fit()
model.summary()

#training arima model on the df2 dataset
print(df.shape)
train = df2.iloc[:-0]
test = df2.iloc[-800:]
print(train.shape, test.shape)

#df2.iloc[] is used to split the dataset into train and test

#plotting the train and test data
start=len(train)
end=len(train)+len(test)-1
pred=model.predict(start=start,end=end,typ='levels').rename('ARIMA Predictions')
pred.plot(legend=True, color = 'red')
test['PM10 (µg/m3)'].plot(legend=True , figsize = (20,6))
```

EE798 ASSIGNMENT

AVINASH SHUKLA

210236

After splitting our df2 dataset into training set and testing set, we implement our ARIMA model based on the result we got earlier we can check the summary of our model.

Here is the snippet of the same:

SARIMAX Results

Dep. Variable:	PM10 (µg/m3)	**No. Observations:**	736
Model:	ARIMA(3, 1, 2)	**Log Likelihood**	-3883.041
Date:	Mon, 26 Jun 2023	**AIC**	7778.082
Time:	11:42:07	**BIC**	7805.681
Sample:	0	**HQIC**	7788.727
	- 736		

Covariance Type: opg

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.0388	0.087	0.444	0.657	-0.132	0.210
ar.L2	0.7939	0.060	13.317	0.000	0.677	0.911
ar.L3	-0.0597	0.064	-0.929	0.353	-0.186	0.066
ma.L1	-0.0377	0.050	-0.762	0.446	-0.135	0.059
ma.L2	-0.9402	0.045	-20.744	0.000	-1.029	-0.851
sigma2	2267.3806	45.754	49.556	0.000	2177.704	2357.057

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	21460.54
Prob(Q):	0.95	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.37	**Skew:**	0.33
Prob(H) (two-sided):	0.00	**Kurtosis:**	29.46

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

We can see a whole lot of information about our model over here. Also, we will be able to see the coefficients of each AR and MA term. Generally, a higher magnitude of this coefficients means that it has a larger impact on the output.

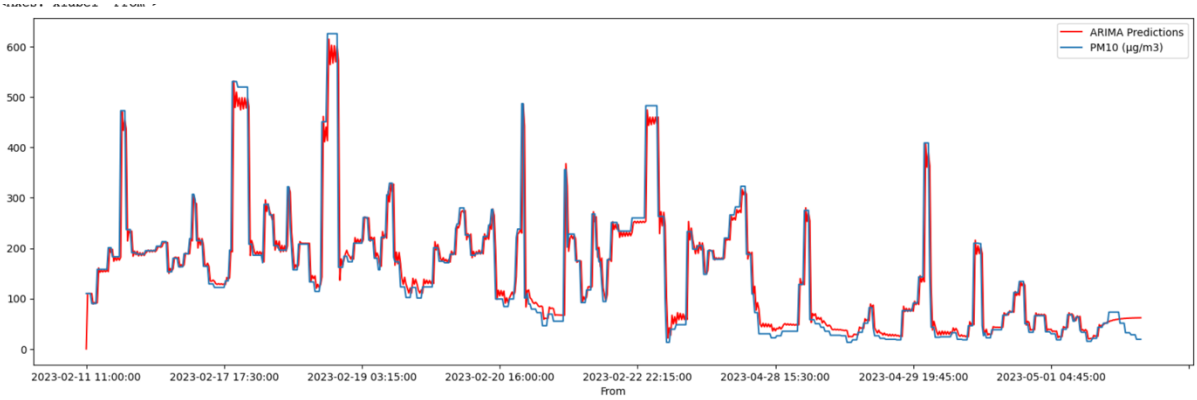
After that code from line 21-25 is used to plot our model and can be compared with the actual data set.

I've plotted the actual pollutant data (Blue) and as well as ARIMA predictions (Red). Since we want to start making predictions where the training data ends, that is what I have written in the start variable. We want to stop making predictions when the data set ends, which explains the end variable. If I want to make future predictions as well, I can just change that accordingly in the start and end variable to the indexes you want.

EE798 ASSIGNMENT

AVINASH SHUKLA

210236



As we can se our ARIMA predictor is approximately similar as our training dataset curve.

As we can see that there is somewhat constant ARIMA curve at the last part of plot which indicates that for that part our model analyzed and predicted that our model will get lower error value in forecasting around the mean value instead of fitting to the irregular trend.

To ascertain how good or bad your model is we find the root mean squared error for it. The following code snippet shows that:

```
df['PM10 (µg/m3)'].mean()
```

```
181.47994972708992
```

```
from sklearn.metrics import mean_squared_error
from math import sqrt
df2['PM10 (µg/m3)'].mean()
rmse=sqrt(mean_squared_error(pred,df2['PM10 (µg/m3)']))
print(rmse)
```

```
27.093524203356495
```

First, we check the mean value of the data set which comes out to be 181. And the root mean squared error for this model should come to around 27. Also, we should care about is that our root mean squared should be very smaller than the mean value of test set. In this case we can see the average error is going to be roughly 27/181*100=15% of the actual value.

Interference of NA values with the plots

- **Discontinuities**: When NA values are present in a time series, they create discontinuities in the plot. This can lead to visual interruptions in the line or curve representing the data, making it difficult to interpret the overall trend.
- **Impact on statistical measures**: NA values can affect the calculation of statistical measures, such as averages or correlations, used for analysis. These missing values can result in biased estimates or incorrect assessments of relationships between variables. It is important to handle NA values appropriately before plotting the time

series using techniques such as **interpolation**, where missing values are estimated based on the neighboring data points, or **forecasting** using time series analysis where missing values are calculated by the ARMA/ARIMA model which we have done earlier.

Can we just replace NA with 0 values?

No, we cannot simply replace NA with 0 values. By replacing NA values with 0s, you are essentially treating the missing data as if it has a value of 0.

- **Distorted representation**: By replacing NA values with 0s, the plot will include these 0 values, which may not accurately represent the true values of the missing data points. This can lead to misleading interpretations of the time series and its trends.

- **Impact on statistical measures**: Replacing NA values with 0s can significantly affect statistical measures such as averages, sums, or correlations. Since 0 is a specific value, it will influence these calculations and potentially skew the results.

- **Artificial patterns**: When NA values are replaced with 0s, it can introduce artificial patterns or trends in the data. These patterns may not reflect the true nature of the time series and can mislead the analysis or modeling process.

A better strategy to handle NA entries is using alternative approaches such as interpolation, Forward or Backward Filling, Mean or Median Substitution, Time Series Forecasting, or imputation which may be more appropriate to preserve the integrity of the time series data.

NOW COMING TO THE LINEAR INTERPOLATION OF THE MISSING NA VALUES.

The missing NA values in our data can be roughly estimated as a combination of the neighboring values. It can give us a pretty much handy representation of what final plot will look like.

Below is the code which we can use for linear interpolation of PM10 pollutant which we can extend to the other pollutants also.

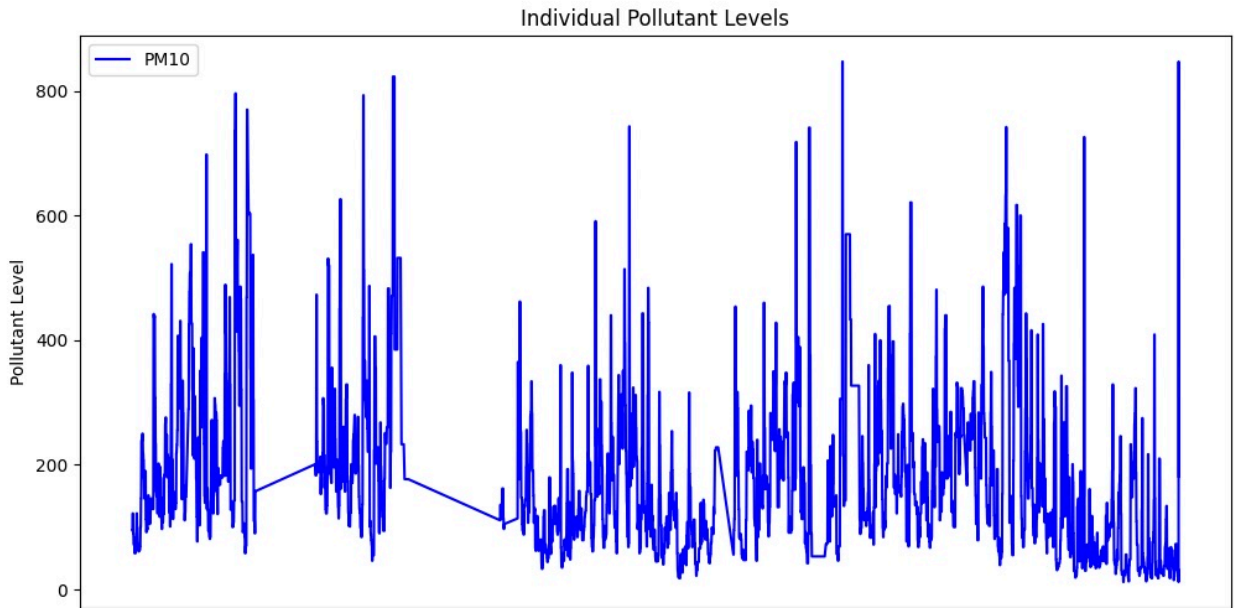
EE798 ASSIGNMENT

AVINASH SHUKLA

210236

```python
st.py > ...
import pandas as pd
import matplotlib.pyplot as plt
# Load the CSV file, parse date and time columns as datetime
data = pd.read_csv('ABCD.csv', parse_dates={'DateTime': ['From', 'To (Interval: 15M)']})

# Set the 'DateTime' column as the DataFrame index
data.set_index('DateTime', inplace=True)
# Perform linear interpolation
data.interpolate(method='linear', axis=0, inplace=True)

# Plot individual pollutant levels
plt.figure(figsize=(12, 6))
plt.plot(data['PM10 (µg/m3)'], color='blue', label='PM10')
plt.xlabel('Time')
plt.ylabel('Pollutant Level')
plt.title('PM10')
plt.legend()
plt.show()
```

The linear interpolated graph looks like:



EE798 ASSIGNMENT

AVINASH SHUKLA

210236

INTERPOLATION VS ARMA/ARIMA

Interpolation is useful for filling in missing values in a time series and can be a simple and straightforward method. Interpolation can be effective when the missing values are sporadic, and the time series exhibits a relatively stable pattern. However, Interpolation does not capture the underlying dynamics or trends in the data and may introduce artificial values that do not reflect the true behavior of the time series.

On the other hand, ARMA/ARIMA processes are used for modeling and forecasting the underlying patterns and trends in a time series. They consider the autocorrelation and stationarity of the data, allowing for more accurate modeling and forecasting. However, these models require the data to meet certain assumptions, such as stationarity, and the selection of appropriate model orders can be challenging.

If the goal is simply to fill in missing values, interpolation can be a quick and effective method. However, if the goal is to understand and model the underlying dynamics and make accurate forecasts, ARMA/ARIMA processes are generally more suitable.

NOW TO CHECK IF THERE IS INCREASE AT THE TIME OF BLASTING WHICH IS 13:45:00 EVERYDAY?

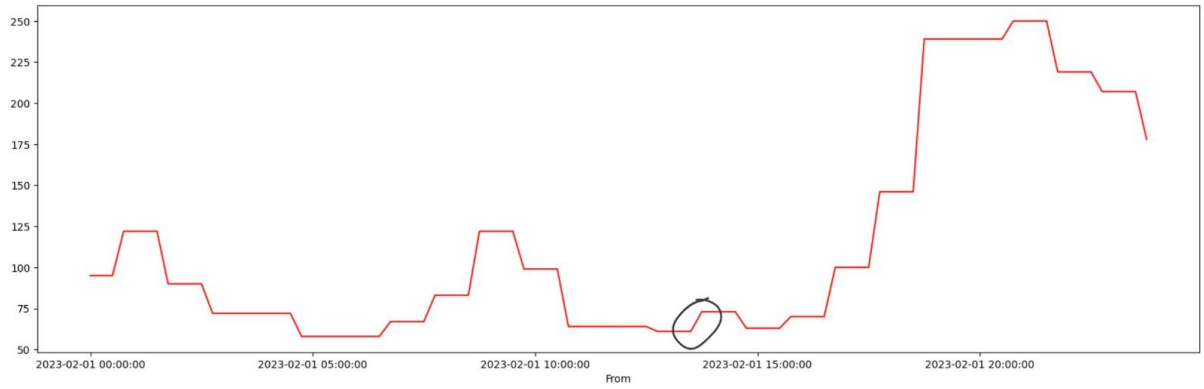
First, we need to extract the data of 1day so that we can check the surge in the level at the time of blasting more correctly. Here is the code snippet to that:

```
#To plot the graph of PM10 over 1 day
#And checking the sudden increase at time of blasing
#which is 13:45:00
import pandas as pd

# Define the starting row and ending row
starting_row = 1
ending_row = 96

# Extract the data from the specified rows
new_df = df.iloc[starting_row-1:ending_row]

new_df['PM10 (µg/m3)'].plot(figsize=(20,6), color = 'red')
```

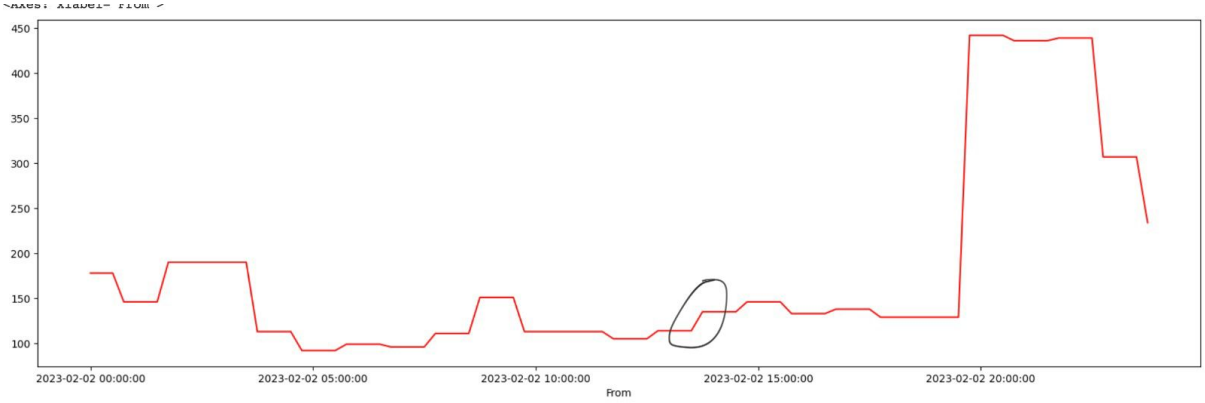
EE798 ASSIGNMENT

AVINASH SHUKLA

210236



As we can see in the plot in the highlighted region that there is a sudden increase in the PM10 level at 13:45:00 which later settles down after 14:45:00. Similarly, we can do for other days also and observe the same pattern. For e.g.: For 2nd Feb we change the starting row data to 97 and ending row to 192, then we get this plot which also has same insurgence at blasting time.



EE798 ASSIGNMENT

AVINASH SHUKLA

210236

NOW COMING TO THE QQ PLOTS OF THE POLLUTANTS. LETS SEE WHAT WE CAN INFER FROM IT.

QQ plots, also known as quantile-quantile plots, are used to assess if a dataset follows a specific distribution, typically comparing it to a theoretical distribution such as the normal distribution. The plot compares the quantiles of the observed data against the quantiles of the theoretical distribution.

QQ plots are particularly useful for comparing a dataset to a normal distribution. In a QQ plot comparing data to a normal distribution, if the points approximately fall along a straight line, it suggests that the data follows a normal distribution. Any deviations from the straight line indicate departures from normality.

Specifically, if the points in the plot deviate from the straight line towards the tails, it indicates heavier or lighter tails in the data compared to a normal distribution. If the points deviate from the straight line in the center, it suggests a deviation from normality in the central portion of the distribution.

First of all to draw the QQ plot we need to remove NA values from our dataset and then proceed.Here is the code which we can use to plot QQ plot.

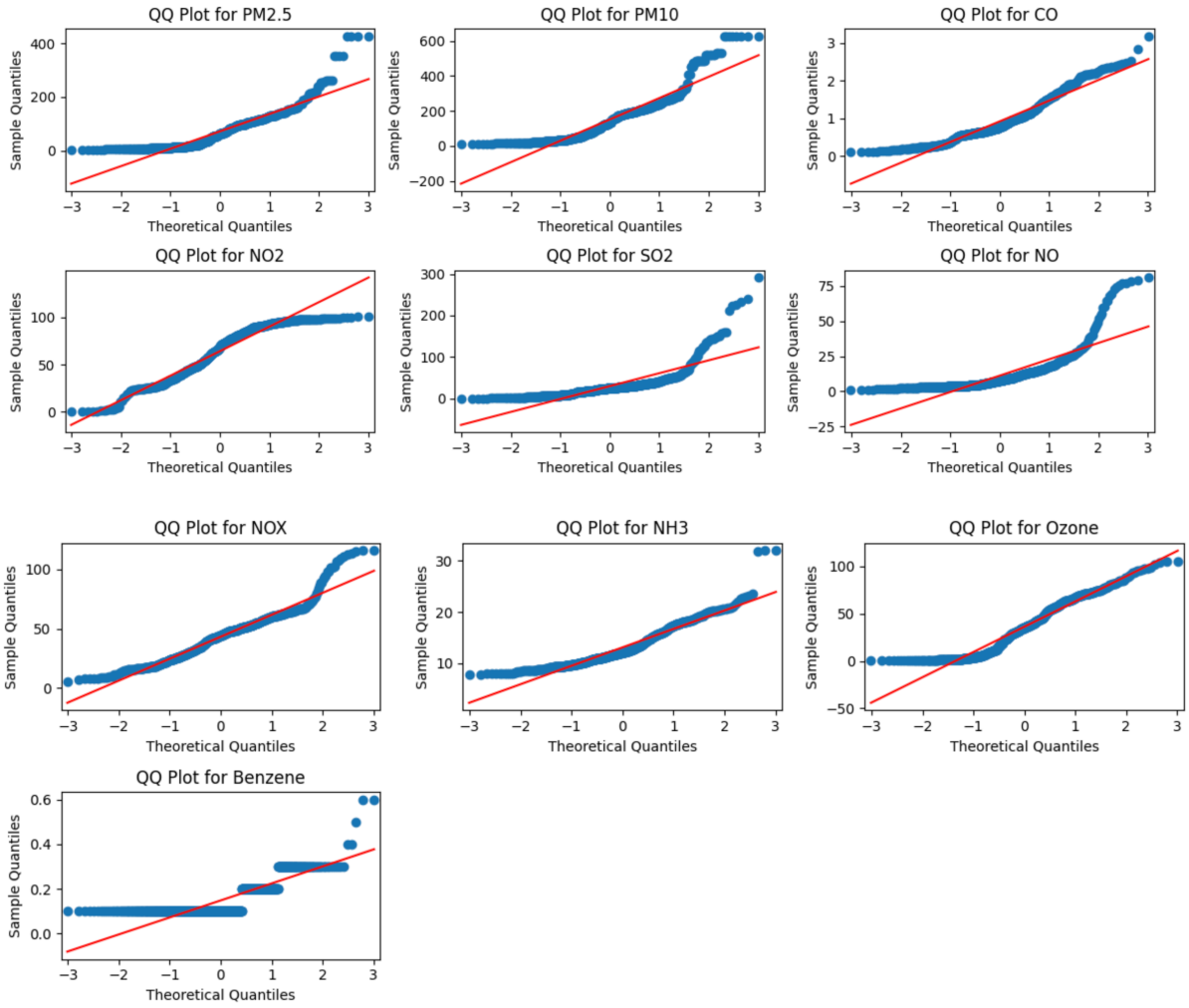
```python
import pandas as pd
import matplotlib.pyplot as plt

#we need to import qqplots from statsmodels.graphics.gofplots
from statsmodels.graphics.gofplots import qqplot

# Select the pollutant columns
pollutant_columns = ['PM2.5 (µg/m3)', 'PM10 (µg/m3)', 'CO (mg/m3)', 'NO2 (µg/m3)', 'SO2 (µg/m3)', 'NO (µg/m3)',
                     'NOX (ppb)', 'NH3 (µg/m3)', 'Ozone (µg/m3)', 'Benzene (µg/m3)']
pollutant_names = ['PM2.5', 'PM10', 'CO', 'NO2', 'SO2', 'NO', 'NOX', 'NH3', 'Ozone', 'Benzene']

# Set the figure size
plt.figure(figsize=(12, 10))

# Iterate over each pollutant and plot the QQ plot
for i, column in enumerate(pollutant_columns):
    plt.subplot(4, 3, i+1)
    qqplot(df2[column], line='s', ax=plt.gca())
    plt.title(f'QQ Plot for {pollutant_names[i]}')

# Adjust the plot layout
plt.tight_layout()

# Show the plots
plt.show()

```

EE798 ASSIGNMENT

AVINASH SHUKLA

210236

We need to import QQ plot from stats model library. And we can proceed in the similar manner. Here are the QQ plots of various pollutants.



More the plot aligns with the straight line more the similarity of it with normal distribution. For ex. Plot of Ozone, NOX and NH3 have pretty much same distribution as normal.

Now coming to the question,

Can you plot the histogram of this blast trigger times across all months of data?

To execute this, first we need to extract the blasting data which is from 13:45:00 to 14:45:00 across whole dataset and then we can plot the histogram of pollutants in that time frame.

We can use the iloc method to select rows from a Data Frame by row number. As we can see the pattern in our data set that the blasting time starts with the row 56 and occurs at interval of 96 rows. Here is the code for that.

EE798 ASSIGNMENT

AVINASH SHUKLA

210236

```python
import pandas as pd
# Define the starting row, ending row, and row gap
starting_row = 56
ending_row = 8600
row_gap = 96

# Calculate the total number of rows to extract
num_rows = (ending_row - starting_row) // row_gap + 1

# Create a list to store the selected rows
selected_rows = []

# Iterate through the rows at the specified intervals and select the rows
for i in range(num_rows):
    row_index = starting_row + i * row_gap
    selected_rows.append(df.iloc[row_index])

# Create a new DataFrame from the selected rows
dg = pd.DataFrame(selected_rows)

# Print the selected rows
print(dg)
```

We can see in the output that blasting data has been captured and stored in the dg dataset.

```
                        #   To (Interval: 15M)  PM10 (µg/m3)  PM2.5 (µg/m3)  \
2023-02-01 14:00:00    57  2023-02-01 14:15:00          73.0           18.0
2023-02-02 14:00:00   153  2023-02-02 14:15:00         135.0           30.0
2023-02-03 14:00:00   249  2023-02-03 14:15:00         147.0           24.0
2023-02-04 14:00:00   345  2023-02-04 14:15:00           NaN            NaN
2023-02-05 14:00:00   441  2023-02-05 14:15:00         149.0           23.0
...                   ...                  ...           ...            ...
2023-04-27 14:00:00  8217  2023-04-27 14:15:00          29.0           13.0
2023-04-28 14:00:00  8313  2023-04-28 14:15:00          30.0           12.0
2023-04-29 14:00:00  8409  2023-04-29 14:15:00          21.0            7.0
2023-04-30 14:00:00  8505  2023-04-30 14:15:00          32.0            4.0
2023-05-01 14:00:00  8601  2023-05-01 14:15:00          15.0            4.0
```

Finally, we can plot the histogram using the matplotlib library. Here is the code for that.

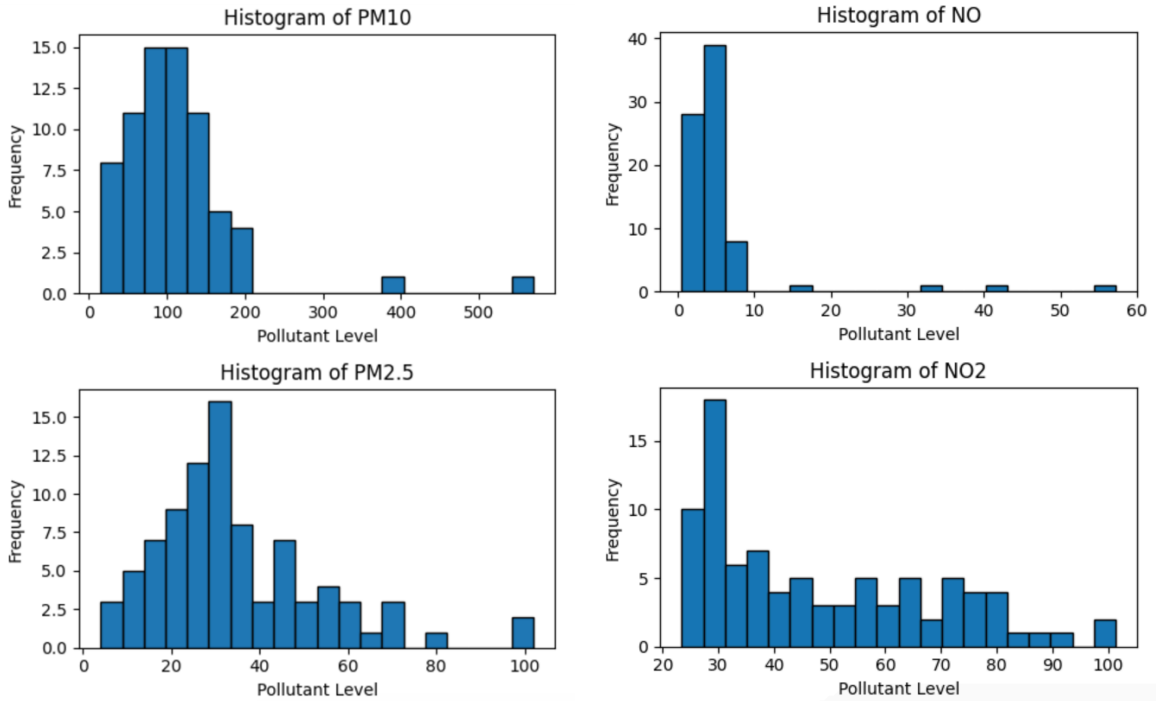
EE798 ASSIGNMENT

AVINASH SHUKLA

210236

```python
import matplotlib.pyplot as plt

# Define the pollutant columns and names
pollutant_columns = ['PM10 (µg/m3)', 'PM2.5 (µg/m3)', 'NO (µg/m3)', 'NO2 (µg/m3)', 'NOX (ppb)',
                     'CO (mg/m3)', 'SO2 (µg/m3)', 'NH3 (µg/m3)', 'Ozone (µg/m3)', 'Benzene (µg/m3)']
pollutant_names = ['PM10', 'PM2.5', 'NO', 'NO2', 'NOX', 'CO', 'SO2', 'NH3', 'Ozone', 'Benzene']

# Set up the subplots
fig, axs = plt.subplots(len(pollutant_columns), 1, figsize=(5, 25))

# Iterate through each pollutant column and plot the histogram
for i, column in enumerate(pollutant_columns):
    ax = axs[i]
    pollutant_name = pollutant_names[i]

    # Plot the histogram
    ax.hist(dg[column].dropna(), bins=20, edgecolor='black')

    # Set the title and labels
    ax.set_title(f'Histogram of {pollutant_name}')
    ax.set_xlabel('Pollutant Level')
    ax.set_ylabel('Frequency')

# Adjust the layout and spacing
plt.tight_layout()

# Show the plot
plt.show()
```

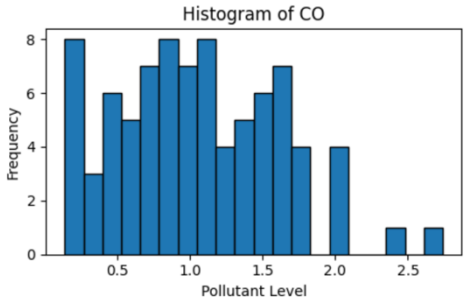
Here are the output histograms of the pollutants.

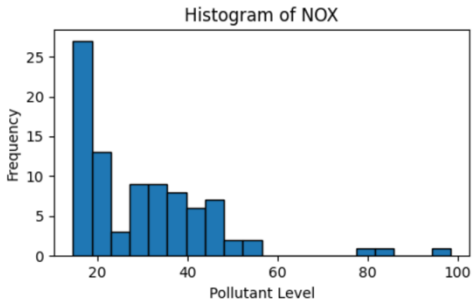


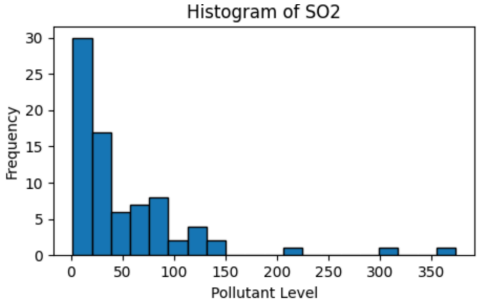
EE798 ASSIGNMENT

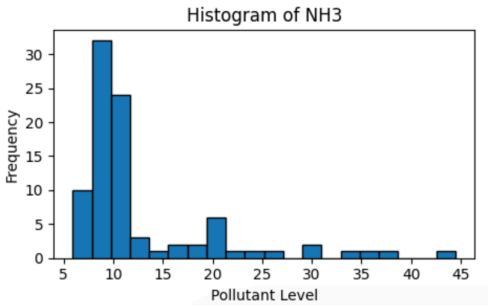
AVINASH SHUKLA

210236





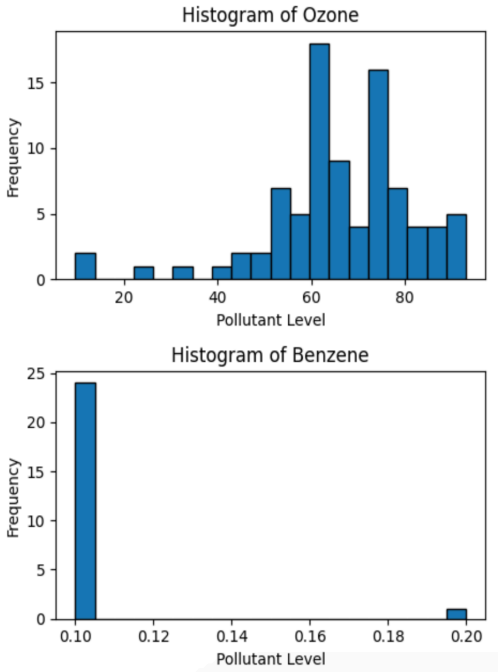




EE798 ASSIGNMENT

AVINASH SHUKLA

210236



Here are some observations from the Histogram plot of pollutants:

- PM 10 has most of the times its value around 100 at the time of blasting.
- NO has its value centered between 0-10 most of the times.
- PM2.5 has its value around 35-40 mostly.
- For NO2 its values are around 30 most of the times.
- Histogram of CO is spread out at many values, so can't really infer anything strongly from it.

Now coming to the last part, AQI CALCULATION FROM THE DATA.

- ## What is the Air Quality Index (AQI)?

  Air Quality Index (AQI) is a number used to convey the quality of air by the government to the general public. Air quality deteriorates with an increase in the concentration of pollutants. The Air Quality Index represents the severity of pollution for ordinary people.

- ## Indian (CPCB) AQI:

  According to the Indian Government (CPCB), Indian AQI range is from 0-500, from 0 being good and 500 being severe. There are eight major pollutants to be considered for AQI calculation, viz. particulate matter (PM 10 and PM 2.5), carbon monoxide (CO), ozone (O3), nitrogen dioxide (NO2), sulfur dioxide (SO2), ammonia (NH3), and lead (Pb). To calculate AQI,

data for a minimum of three pollutants must be present, of which one should be either PM10 or PM2.5, AQI ranging from 0-500 has different concentrations for each pollutant and has health effects accordingly.

- Indian AQI range & probable impacts:

0-50: This range defines air quality as good as it shows minimal or no impact on health.
51-100: This is a satisfactory air quality range, and it can show effects such as breathing difficulty in sensitive groups.
101-200: The range shows moderate air quality with impacts such as breathing discomfort for children and elderly people, and people already suffering from lung disorders and heart disease.
201-300: AQI falling in this range communicates that the air quality is poor and shows health effects on people when exposed for the long term. People already suffering from heart diseases can experience discomfort from short exposure.
301-400: This range shows very poor air quality and causes respiratory illness for a longer duration of exposure.
401-500: This is the severe range of AQI causing health impacts to normal and diseased people. It also causes severe health impacts on sensitive groups.

- How to calculate Air Quality Index?

The formula to calculate AQI is the same as per the Indian CPCB and US-EPA. The AQI is calculated using the equations separately for parameters. For example, if you wish to calculate AQI based on four parameters, use the equation four times, and the worst sub-index communicates the AQI. A subindex is a linear function (two different yet related notions) of the concentration of pollutants.

$$Ip = [IHi – ILo / BPHi – BPLo] (Cp – BPLo) + ILo$$

Were,
Ip = index of pollutant p
Cp = truncated concentration of pollutant p
BPHi = concentration breakpoint i.e., greater than or equal to Cp
BPLo = concentration breakpoint i.e. less than or equal to Cp
IHi = AQI value corresponding to BPHi
ILo = AQI value corresponding to BPLo

- INDIAN EQUATION FOR AQI

The Indian AQI range differs from that of US-EPA. To calculate AQI, a minimum of three parameters should be taken out of which one must be either PM10 or PM2.5.
To calculate sub-indices, 16 hours of data is needed.

EE798 ASSIGNMENT

AVINASH SHUKLA

210236

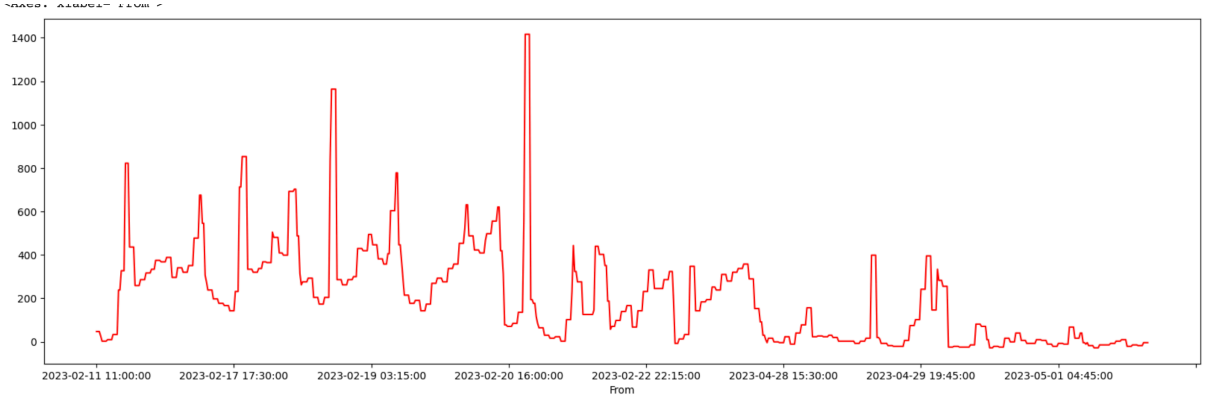
For example: If you wish to calculate AQI based on PM2.5, CO, and ozone, calculate the sub-index for each parameter separately.
If the current concentration of PM2.5 is 110 ug/m3, then referring to AQI range as per Indian standards BPHi = 120, BPLo = 91, IHi = 300 and ILo = 101.
Putting the values in equation and solving:
Sub Index= [(300-201)/ (120-91)] (110-91) + 201 = 265.86
Similarly, for other parameters, the sub-index can be calculated, and the worst sub-index shows the AQI.

We can calculate AQI based on PM2.5 as in our dataset it generally assumes higher value as compared to other pollutants.

On putting values of the constants, we get the formula as:
AQI_of_PM2.5 = 3.414*(x-91) + 201 where x = level of PM2.5
Here is the code snippet of the same:

```python
import pandas as pd
import matplotlib.pyplot as plt

# Alternatively, create a new dataset to store AQI values
new_df_AQI = pd.DataFrame()
new_df_AQI['AQI'] = (df2['PM2.5 (µg/m3)'] - 91) * 3.414 + 201

new_df_AQI['AQI'].plot(figsize=(20,6), color = 'red')
```

We get the following plot of AQI_based_on_PM2.5



From the plot we can infer that the AQI value based on the PM2.5 pollutant has its peak value around 20th February and follows a pretty low value in the starting of May.

EE798 ASSIGNMENT

AVINASH SHUKLA

210236

Here is the link of .py file:

https://colab.research.google.com/drive/1mragHxlmk9QnktCx3fOVE7qB0hsukSBz

Link to .py file

https://drive.google.com/file/d/1pApUS0iYwAxfqMi7TqrXaoX8Bbk_gfdh/view?usp=share_link