I started with Time Series Analysis of the data given to us and slowly proceeded to answer all the question asked.
I have answered all the question in the pdf only after the visualizion of that particular question.
We cannot replace Nan values with 0 as it gives high mape and replacing NaN values with mean is little bit better strategy than replacing them with 0 so we did time series analysis for that part.
Nan values were interferring with the plotting as we could not plot the acf and pacf plot without replacing Nan values.
Yes we can establish ARMA/ARIMA process for the dataset and I did it by per-column as we needed to find the time serie for particular pollutant. I have done the process for some columns and also did the descriptive analysis of the pollutants below.

For the interpolation part I have done all the three interpolations i.e., linear, cubic and spline and deduced the most suitable among all three.
Also I saw seasonality in density of some pollutants and replaced the nan values using seasonality then I compared the interpolated data and the data which i got through ARMA/ARIMA processes.

After this I did statistical inference in which we concluded that actual blasting time did not match our estimated blasting time. I computed the probability, did combined weighted time series analysis using ARMA/ARIMA and also did Q-Q plot.

At last I did the problem setting and answered them.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pandas.plotting
from pandas.plotting.mpl_timeseries import plot_acf, plot_pacf
%matplotlib inline
```

In [2]:
```
file_loc = r'C:\Users\khuma\Downloads\GE 7982\Open pit blasting.csv'
air_pollution_data = pd.read_csv(file_loc, index_col=0,  header = 0)
```

In [3]:
```
air_pollution_data.head()
```

Out[3]:

	From (Internal 1MB)	To	Smgrad, Surya Kiran Bhawan Dudhiwala pm10 (µg/m3)	Smgrad, Surya Kiran Bhawan Dudhiwala pm25 (µg/m3)	Smgrad, Surya Kiran Bhawan Dudhiwala NO (µg/m3)	Smgrad, Surya Kiran Bhawan Dudhiwala NO2 (µg/m3)	Smgrad, Surya Kiran Bhawan Dudhiwala NOX (ppb)	Smgrad, Surya Kiran Bhawan Dudhiwala NOX (µg/m3)	Smgrad, Surya Kiran Bhawan Dudhiwala CO (mg/m3)	Smgrad, Surya Kiran Bhawan Dudhiwala SO2 (µg/m3)	Smgrad, Surya Kiran Bhawan Dudhiwala O3 (µg/m3)	Smgrad, Surya Kiran Bhawan Dudhiwala Ozone (µg/m3)	Smgrad, Surya Kiran Bhawan Dudhiwala Benzene (µg/m3)
#													
1	2023-01-01 00:00:00	2023-01-01 00:15:00	95.0	35.0	NaN	90.1	56.2	0.31	NaN	17.7	28.1	0.4
2	2023-01-01 00:15:00	2023-01-01 00:30:00	95.0	35.0	NaN	88.0	55.1	0.33	NaN	18.3	27.1	0.4
3	2023-01-01 00:30:00	2023-01-01 00:45:00	95.0	35.0	NaN	87.7	55.2	0.38	NaN	19.7	24.9	0.4
4	2023-01-01 00:45:00	2023-01-01 01:00:00	122.0	34.0	NaN	88.9	55.7	0.38	NaN	21.3	21.9	0.4
5	2023-01-01 01:00:00	2023-01-01 01:15:00	122.0	34.0	NaN	90.0	55.8	0.38	NaN	22.3	16.7	0.4

In [5]:
```
data = pd.DataFrame(air_pollution_data)
```

In [6]:
```
data.columns = ['From','To','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)']
```

In [7]:
```
data.head(1)
```

Out[7]:

	From	To	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)
#												
1	2023-01-01 00:00:00	2023-01-01 00:15:00	95.0	35.0	NaN	90.1	56.2	0.31	NaN	17.7	28.1	0.4

In [8]:
```
# data['Time Interval'] = data['From'].* - *data['To']
data.fillna(value = 0, inplace = True)
data.head(1)
```

Out[8]:

	From	To	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)
#												
1	2023-01-01 00:00:00	2023-01-01 00:15:00	95.0	35.0	0.0	90.1	56.2	0.31	0.0	17.7	28.1	0.4

In [9]:
```
data = data[:-1]
# data.head()
inform = data['From']
# PM = data['To']
#data['Total Pollution'] = PM10+PM2.5+NO+NO2+CO+SO2+NH3+OZONE+BENZENE+(NOX*44.0131)/24.45
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8640 entries, 1 to 8640
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   From          8640 non-null   object
 1   To            8640 non-null   object
 2   PM10(µg/m3)   8640 non-null   float64
 3   PM2.5(µg/m3)  8640 non-null   float64
 4   NO(µg/m3)     8640 non-null   float64
 5   NO2(µg/m3)    8640 non-null   float64
 6   NOX(ppb)      8640 non-null   float64
 7   CO(mg/m3)     8640 non-null   float64
 8   SO2(µg/m3)    8640 non-null   float64
 9   NH3(µg/m3)    8640 non-null   float64
 10  Ozone(µg/m3)  8640 non-null   float64
 11  Benzene(µg/m3)  8640 non-null   float64
dtypes: object(2), float64(10)
memory usage: 877.5 KB
```

In [10]:
```
data['To'] = pd.to_datetime(data['To'],format='%Y-%m-%d %H:%M:%S')
data['From'] = pd.to_datetime(data['From'], format = '%Y-%m-%d %H:%M:%S')
s = data['To'][0#40]
s
```

```
Timestamp('2023-04-25 18:00:00')
```

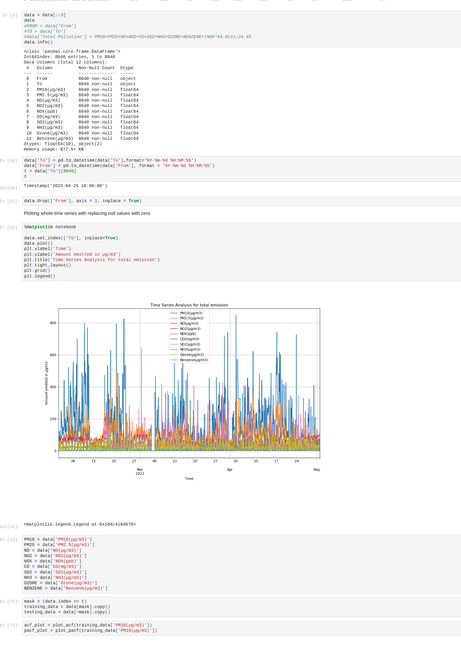
Out[10]:

In [11]:
```
data.drop(['From'], axis = 1, inplace = True)
```

Plotting whole time series with replacing null values with zero

In [12]:
```
%matplotlib notebook

data.set_index(['To'], inplace=True)
plt.figure()
plt.xlabel('Time')
plt.ylabel('Amount emitted in µg/m3')
plt.title('Time Series Analysis for total emission')
plt.tight_layout()
plt.grid()
plt.legend()
```

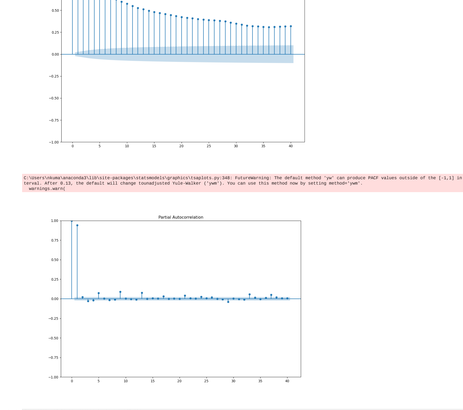
Out[12]:
```
<matplotlib.legend.Legend at 0x104c4160870>
```



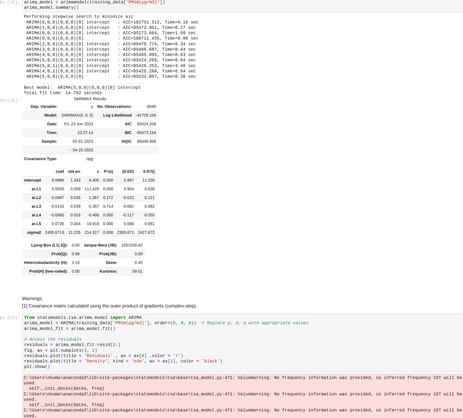
In [13]:
```
PM10 = data['PM10(µg/m3)']
PM25 = data['PM2.5(µg/m3)']
NO = data['NO(µg/m3)']
NO2 = data['NO2(µg/m3)']
NOX = data['NOX(ppb)']
CO = data['CO(mg/m3)']
SO2 = data['SO2(µg/m3)']
NH3 = data['NH3(µg/m3)']
OZONE = data['Ozone(µg/m3)']
BENZENE = data['Benzene(µg/m3)']
```

In [14]:
```
mask = (data.index <= s)
training_data = data[mask].copy()
testing_data = data[~mask].copy()
```

In [15]:
```
acf_plot = plot_acf(training_data['PM10(µg/m3)'])
pacf_plot = plot_pacf(training_data['PM10(µg/m3)'])
```




```
C:\Users\khuma\anaconda3\lib\site-packages\statsmodels\graphics\tsaplots.py:348: FutureWarning: The default method 'yw' can produce PACF values outside of the [-1,1] in
terval. After 0.13, the default will change tounadjusted Yule-Walker ('ywm'). You can use this method now by setting method='ywm'.
  warnings.warn(
```



In [16]:
```
from statsmodels.tsa.stattools import adfuller
adf_test = adfuller(training_data['PM10(µg/m3)'])
print(f'p-value: {adf_test[1]}')
```

```
p-value: 5.970874397215685e-12
```

In [18]:
```
# we set d = 0
```

In [19]:
```
#conda install pmdarima
```

In [20]:
```
from pmdarima.arima import auto_arima
```

In [21]:
```
def arimamodel(timeseriesarray):
    autoarima_model = auto_arima(timeseriesarray, start_p=0, start_q=0, max_p=5, max_q=2, d=0, test='adf', trace = True)
    return autoarima_model
```

In [22]:
```
arima_model = arimamodel(training_data['PM10(µg/m3)'])
arima_model.summary()
```

```
Performing stepwise search to minimize aic
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=102753.322, Time=0.10 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=95427.951, Time=0.27 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=98273.664, Time=1.50 sec
 ARIMA(2,0,0)(0,0,0)[0] intercept   : AIC=95171.435, Time=0.36 sec
 ARIMA(3,0,0)(0,0,0)[0] intercept   : AIC=95047.192, Time=0.48 sec
 ARIMA(4,0,0)(0,0,0)[0] intercept   : AIC=94946.687, Time=0.44 sec
 ARIMA(4,0,1)(0,0,0)[0] intercept   : AIC=95485.995, Time=0.63 sec
 ARIMA(3,0,1)(0,0,0)[0] intercept   : AIC=95253.255, Time=0.49 sec
 ARIMA(4,0,0)(0,0,0)[0]             : AIC=95428.253, Time=0.40 sec
 ARIMA(4,0,1)(0,0,0)[0] intercept   : AIC=95425.298, Time=0.94 sec
 ARIMA(3,0,1)(0,0,0)[0] intercept   : AIC=95252.897, Time=0.38 sec

Best model:  ARIMA(5,0,0)(0,0,0)[0] intercept
Total fit time: 14.782 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 8640
Model:               SARIMAX(5, 0, 0)   Log Likelihood              -47705.104
Date:                Fri, 23 Jun 2023   AIC                          95424.209
Time:                        23:37:14   BIC                          95475.154
Sample:                    02-01-2023   HQIC                         95440.958
                         - 04-25-2023
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      8.5988      1.343      6.405      0.000      5.967      11.230
ar.L1          0.9206      0.016     57.429      0.000      0.904       0.936
ar.L2          0.0497      0.016      3.367      0.172      -0.022       0.121
ar.L3          -0.0143      0.039     -0.367      0.714      -0.091       0.062
ar.L4          -0.0860      0.016     -5.486      0.000      -0.117      -0.055
ar.L5          0.0736      0.004     18.916      0.000      0.066       0.081
sigma2       2405.6713     12.225    196.737      0.000    2381.671    2427.672
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):            105026.42
Prob(Q):                              0.99   Prob(JB):                         0.00
Heteroskedasticity (H):               1.15   Skew:                            0.49
Prob(H) (two-sided):                  0.00   Kurtosis:                       18.61
===================================================================================
```

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

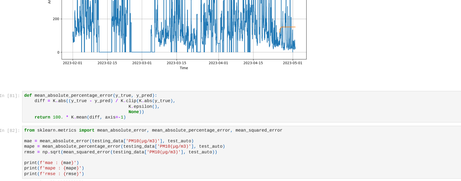
In [23]:
```
from statsmodels.tsa.arima.model import ARIMA
arima_model = ARIMA(training_data['PM10(µg/m3)'], order=(5, 0, 0))  # Replace p, d, q with appropriate values
arima_model_fit = arima_model.fit()

# Access the residuals
residuals = arima_model_fit.resid[1:]
fig, ax = plt.subplots(1, 2)
residuals.plot(title='Residuals', ax=ax[0],color='r')
residuals.plot(title='Density', kind='kde', ax=ax[1], color='black')
plt.show()
```

```
C:\Users\khuma\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning: No frequency information was provided, so inferred frequency 15T will be used.
  self._init_dates(dates, freq)
C:\Users\khuma\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning: No frequency information was provided, so inferred frequency 15T will be used.
  self._init_dates(dates, freq)
C:\Users\khuma\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning: No frequency information was provided, so inferred frequency 15T will be used.
  self._init_dates(dates, freq)
```

As seen above residue is white noise with mean zero and constant variance

In [24]:
```
test_auto = arima_model.predict(n_periods = len(testing_data['PM10(µg/m3)']))
data['PM10 Forecast'] = (Name:)*len(training_data['PM10(µg/m3)']) + list(test_auto)
```

In [25]:
```
%matplotlib notebook
# data.set_index(['To'], inplace = True)
# plt.figure(figsize=(15,6))
plt.plot(data['PM10 Forecast'], label='Forecast data')
plt.xlabel('Time')
plt.ylabel('Amount emitted in µg/m3')
plt.title('Actual PM10 emission v/s Predicted emission of PM10')
plt.tight_layout()
plt.legend()
plt.show()
```



In [26]:
```
def mean_absolute_percentage_error(y_true, y_pred):
    diff = np.abs(y_true - y_pred) / np.clip(np.abs(y_true),
                                             k_epsilon),
    return '{:.*}'.format(diff, axis=1)
```

In [27]:
```
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, mean_squared_error

mae = mean_absolute_error(testing_data['PM10(µg/m3)'], test_auto)
mape = mean_absolute_percentage_error(testing_data['PM10(µg/m3)'], test_auto)
rmse = np.sqrt(mean_squared_error(testing_data['PM10(µg/m3)'], test_auto))

print("mae : {mae}")
print("mape : {mape}")
print("rmse : {rmse}")
```

```
mae : 6.4284773886758
mape : 1.1339802918588E13-0
rmse : 143.1198918616226
```

Since we filled all the NaN values with zero so the mean absolute percentage error is very high so we will not replace NaN values with zero instead we will replace them with the mean of that column as it will maintain stationarity and not disturb the distribution much

```
Shape: (9358, 8)
<class'pandas.core.frame.DataFrame'>
DatetimeIndex: 9358 entries, 2014-03-10 18:00:00 to 2015-12-31 23:00:00
Data columns (total 8 columns):
 PT08.S1(CO)    9357 non-null   float64
 PT08.S2(NMHC)  9357 non-null   float64
 PT08.S3(NOx)   9357 non-null   float64
 PT08.S4(NO2)   9357 non-null   float64
 PT08.S5(O3)    9357 non-null   float64
 T              9357 non-null   float64
 RH             9357 non-null   float64
 AH             9357 non-null   float64
dtype: float64
```

```
[1] "Sample size: 1000"
[1] "Sample mean: 0.0"
[1] "Sample sd: 1.0"
[1] "Sample skewness: 0.0"
[1] "Sample kurtosis: 0.0"
```

```
[1] "Sample quantiles:"
  0.05   0.10   0.15   0.20   0.25   0.30   0.35   0.40   0.45   0.50   0.55   0.60   0.65   0.70   0.75   0.80   0.85   0.90   0.95
-1.64  -1.28  -1.04  -0.84  -0.67  -0.52  -0.39  -0.25  -0.13   0.00   0.13   0.25   0.39   0.52   0.67   0.84   1.04   1.28   1.64
```

```
[1] "Sample table:"
  -3.9   -3.8   -3.7   -3.6   -3.5   -3.4   -3.3   -3.2   -3.1   -3.0   -2.9   -2.8   -2.7   -2.6   -2.5   -2.4   -2.3   -2.2   -2.1
    1      1      1      1      1      1      1      1      1      1      1      1      1      1      1      1      1      1      1
   -2.0   -1.9   -1.8   -1.7   -1.6   -1.5   -1.4   -1.3   -1.2   -1.1   -1.0   -0.9   -0.8   -0.7   -0.6   -0.5   -0.4   -0.3   -0.2
    1      1      1      1      1      1      1      1      1      1      1      1      1      1      1      1      1      1      1
   -0.1    0.0    0.1    0.2    0.3    0.4    0.5    0.6    0.7    0.8    0.9    1.0    1.1    1.2    1.3    1.4    1.5    1.6    1.7
    1      1      1      1      1      1      1      1      1      1      1      1      1      1      1      1      1      1      1
    1.8    1.9    2.0    2.1    2.2    2.3    2.4    2.5    2.6    2.7    2.8    2.9    3.0    3.1    3.2    3.3    3.4    3.5    3.6
    1      1      1      1      1      1      1      1      1      1      1      1      1      1      1      1      1      1      1
    3.7    3.8    3.9
    1      1      1
```

Problem Statement

Air pollution data given to us is a stock time series data or a flow time series data? Can air pollution data be both? If you are given two air pollution data what are the factors which allow you to differentiate them as stock time series data or flow time series data?

Answer

The air pollution data given to us in particuar is a stock time series data as it refers to measuring characteristics at a specific moment, much like a static image of the data as it was. Because we were given the density of the pollutants at some particular timestamp and not the flow of variation in density of the pollutants or some timely emission of any pollutant.

In general, air pollution data can be considered as both stock time series data and flow time series data, depending on the specific context and variables being measured.

Stock Time Series Data: Air pollution data can be treated as stock time series data when measuring the concentration or level of pollutants at a specific point in time. For example, measurements of particulate matter (PM2.5), nitrogen dioxide (NO2), or ozone (O3) levels taken at regular intervals (e.g., hourly, daily, monthly) represent stock data. Each data point represents the stock or quantity of the pollutant at that particular time.

As the data given to us is similar to above definition for stock time series data so we can say that the data given to us is stock time series data.

Flow Time Series Data: Air pollution data can also be considered as flow time series data when measuring the flow or rate of pollutants over time. This typically involves measurements of pollutant emissions or pollutant concentrations over a specific time period. For example, hourly emissions of carbon monoxide (CO) from a specific source or the daily average concentration of sulfur dioxide (SO2) in a particular region represent flow data. Each data point represents the flow or rate of the pollutant during that time interval.

In practice, air pollution data can include both stock and flow components. For example, pollutant concentration measurements at monitoring stations represent stock data, while emission inventories or modeled pollutant dispersion data represent flow data. If we are given two dataset representing air pollution then we can differentiate them as stock time series data or flow time series data depending upon the specific variables being measured and the context in which the data is analyzed.

Problem Statement

Do the explanatory analysis for the air pollution data given to you for each pollutant and explain the features in the data. Explain the reason for missing values in the data and give some methonds to find them. Also explain whether the actual blasting time follows the estimated blasting time, if not then state reasons why?

Answer

Air pollution data given to us is a stock time series data which represent the density of the pollutants in air at a particular time interval of 15 minutes

Pollutants released during open pit mining are :

Dust and Particulate Matter (PM): Open pit mining involves excavation and movement of large amounts of soil, rock, and ore. This activity generates dust and particulate matter, which can include both coarse particles (PM10) and fine particles (PM2.5). These particles can cause air pollution and have potential health impacts when inhaled.

Sulfur Dioxide (SO2): Mining operations involving sulfide ores, such as copper, lead, and zinc, can release sulfur dioxide into the air. Sulfur dioxide is produced when sulfide minerals are exposed to air and react with oxygen. It is a major contributor to acid rain and can have adverse effects on human health and the environment.

Nitrogen Oxides (NOx): Equipment and machinery used in open pit mining, such as diesel-powered vehicles and machinery, can emit nitrogen oxides. Nitrogen oxides are formed during the combustion process and contribute to air pollution. They are known to contribute to the formation of smog, respiratory issues, and environmental degradation.

Volatile Organic Compounds (VOCs): Various mining activities, including blasting, drilling, and processing of ores, can release volatile organic compounds. These compounds can include hydrocarbons and other chemicals that evaporate easily into the air. VOC emissions contribute to air pollution and can have adverse effects on human health and the environment.

Heavy Metals: Mining operations can release heavy metals such as mercury (Hg), lead (Pb), cadmium (Cd), arsenic (As), and others. These metals are naturally present in many ores and can be released into the environment through various mining processes, such as ore crushing, grinding, and ore processing. Heavy metals can persist in the environment, accumulate in the food chain, and pose significant health and ecological risks.

Now the data for benzene and ozone given to us comes uder VOCs released during open pit mining which are not particularly released but are formed by the reaction between other gases relaesed durig open pit mining

As inferred form the given data PM10 and PM2.5 gets the highest peak among all pollutants whereas ozone and benzene show some seasonality in the data.

So for the missing data we considered seasonality for benzene and ozone.

Now PM10 and PM2.5 are suspended particulate matter so one reason for them to be high is, during the time of blasting the gases released in air are more dense and heavier than normal so due to this reason the PM10 sensor might consider them as PM10 only but in reality there may be some other gases mixed with it such as NO, NO2, NOX or CO.

One of the main reason for missing values in the data maay be the failure of the sensors that take the reading other may be the sensors worked properly but the reading was not taken at that time due to some other failere. Also during the time of blasting gases are released at high temperature so due to high temperature sensors my malfunction and could not take the reading.

NOX formed is the mixture of NO and NO2 so we can find the correlation between NO and NO2 to find the missing values of NOX.

Interpolation is one of the best technique to find the missing data, checking cyclicity and seasonality in the data can also be helpful as done for benzene and ozone.

When we extimated the blasting time and compared it with the actual blasting time we saw major variation in them this could be due to various reasons :

The time it takes for pollutants to release after blasting in open pit mines can vary depending on several factors, including the type of blasting method used, the characteristics of the rock or ore being blasted, and the atmospheric conditions at the site. Here are some considerations:

Instantaneous Release: Some pollutants, such as dust and particulate matter (PM), are typically released instantaneously or within seconds after the blast. The explosion and subsequent fragmentation of rock or ore can generate a cloud of dust and particulates that disperses rapidly into the surrounding air.

Gaseous Emissions: Gaseous pollutants, such as sulfur dioxide (SO2), nitrogen oxides (NOx), and volatile organic compounds (VOCs), can also be released during blasting. The time it takes for these gases to be emitted and disperse depends on various factors, including the chemical composition of the rock or ore, the extent of mineral oxidation or reaction, and the ventilation conditions in the mining area.

Dispersion and Transport: Once the pollutants are released, their dispersion and transport in the air depend on atmospheric conditions, such as wind speed and direction. Pollutants can disperse and travel different distances from the source, influenced by factors like air stability, topography, and nearby structures.

Duration of Impact: The duration of pollutant impact after blasting can vary. Dust and particulate matter can settle relatively quickly, depending on particle size and environmental conditions. Gaseous emissions may persist for a longer duration, particularly if there are ongoing processes or activities at the mining site that continue to release pollutants.

Traffic : Due to traffic and movement of vehicles in the region may increase or decrease the density of the air pollutants as dust is being carried away and the transportation of coal from open pit mines in India can occur throughout the day, depending on various factors such as operational requirements, logistical considerations, and local regulations. Coal mining operations in India typically involve round-the-clock activities to meet the demand for coal. Therefore, coal transportation can take place at any time, including day and night shifts. This can also result in the wrongly estimated time for blasting

Actual Blasting Time : It may also be the case that the actual blasting time was not the preferred blasting time and the data given to us itself is faulty for some days. This could happed due to various reasons such as failure of blasting equipment like dynamite or non-avaivalibality of those equipment or delay by the workes or the whether condition which which were not suitable for blasting.

Problem Statement

Do the descriptive analysis of the data give to you find trends like seasonality and the distribution of the residue ,measures of frequency, central tendency, dispersion or variation, and position of air pollution data.

Answer

To conduct descriptive analysis for air pollution data classified as stock time series data, you can perform several techniques to gain insights into the data distribution, patterns, and trends. Here are some approaches to consider:

Data Visualization:

Plotting Time Series: Create line plots or area plots to visualize the pollution levels over time. This will help you observe the general trends, seasonality, and any irregular patterns in the data as done above. Heatmaps: Create heatmaps to display pollution levels across different time intervals or geographical locations. This can help identify any spatial or temporal patterns. We could not plot heatmaps as we were not povided any information about the particular location of blasting or mining but just the density of pollutants in the nearby region

Summary Statistics:

Measure Central Tendency: Calculate measures like mean, median, and mode to understand the typical pollution levels. Assess Dispersion: Compute measures such as standard deviation, variance, or range to evaluate the variability or spread of pollution levels. This is also done above in TSA with nan = 0 and nan = mean Identify Outliers: Detect any extreme values or outliers in the data that may require further investigation. Instead of further investigation we discarded the extreme data which we were getting due to interpolation as it was of no use to us.

Seasonal Decomposition:

Decompose the time series data into its constituent components: trend, seasonality, and residual. This can be done using techniques like seasonal decomposition of time series (e.g., STL decomposition) or moving averages. Analyzing these components will provide insights into long-term trends and seasonal variations in pollution levels. We checked for seasonality in the Interpolation and Seasonality part and estimated the missing data points for some pollutants which show seasonality. We also did the residue and density plot for various pollutants and estimated that most of the residue was white noise as it has mean 0 and constant variance. Whereas the density mostly followed normal and cauchy distribution.

Autocorrelation Analysis:

Calculate Autocorrelation: Measure autocorrelation to assess the relationship between past and current pollution levels. This can be done using autocorrelation plots (ACF) or partial autocorrelation plots (PACF). Significant autocorrelation at specific lags may indicate temporal dependencies or patterns in the data. This was also done above

Statistical Tests:

Stationarity Test: Perform tests for stationarity, such as the Augmented Dickey-Fuller (ADF) test, to determine if the pollution data exhibits a constant mean and variance over time. We conducted ADF test for all the pollutants separately and together also to test and train our ARIM model above.