**AIR QUALITY ANALYZE**

Project Submission Part 5: Project Documentation & Submission

**Devendhiran [812121106007]**

**Objective:**

The main objective of the air quality analyze at developing a monitoring system for SO2 levels and any influencing environmental factors such as air temperature, humidity and wind speed, using wireless sensor networks. The design will also enable delivery of near real-time data and information that can be accessed from personal computers and smartphones.

**Introduction:**

Data Analysis is the process of bringing order and structure to collected data. It turns data into information teams can use. Data visualization is the process of putting data into a chart, graph, or other visual format that helps inform analysis and interpretation. Analysis and Visualization of datasets has always been a helpful for various reasons whether it’s for improvement of customer experience or business plans, etc. These all aspects require the analysis of the data. In 2020, the world has seen a paradigm shift across many industries, businesses, climate and to human life itself due to the COVID pandemic. The Government and many private organizations need to know the damage caused by the pandemic for reasons ranging from public welfare to business strategies. These calculations are very important for the growth and robustness of the National economy. To calculate and analyze the effects, we need data regarding the damage. Data is available as clusters in the many nooks and crannies of the internet. This data is then collected as a whole and then merged into a data-set. Even when data is amassed into data sets, it is still an enormous task to sort and make meaning out of it. This data can be simplified and visualized using various Python libraries like matplotlib, NumPy, pandas, etc. In this project the main goal is to implement the Python tools to simplify, analyse, visualize and predict different aspects under the banner “Impact of COVID - 19 on industries, climate and population.”

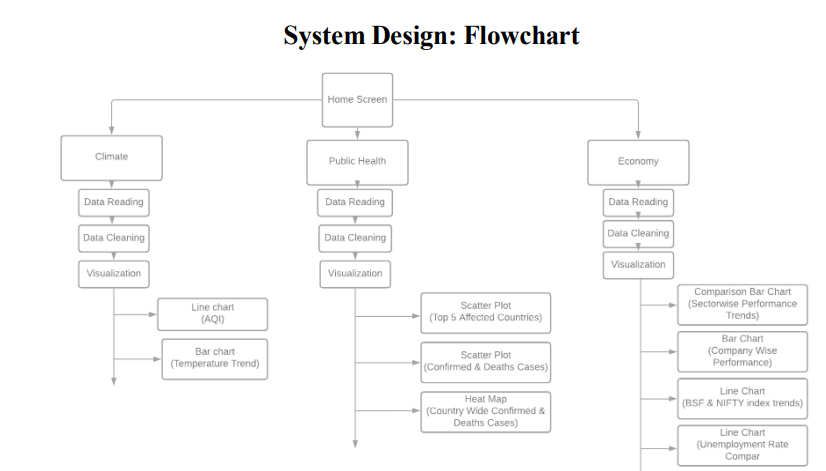
To analysis the impact of the pandemic across the globe in fields such as public health, economic effects, climate changes.- • Visualize the data by using various visualization tools available with python • Cleaning the data to improve the data quality and overall productivity • Use machine learning to predict stock market • The trained LSTM model will help us visualize how the stock market is affected due to the pandemic and based on past results will also be able to predict where the market is heading

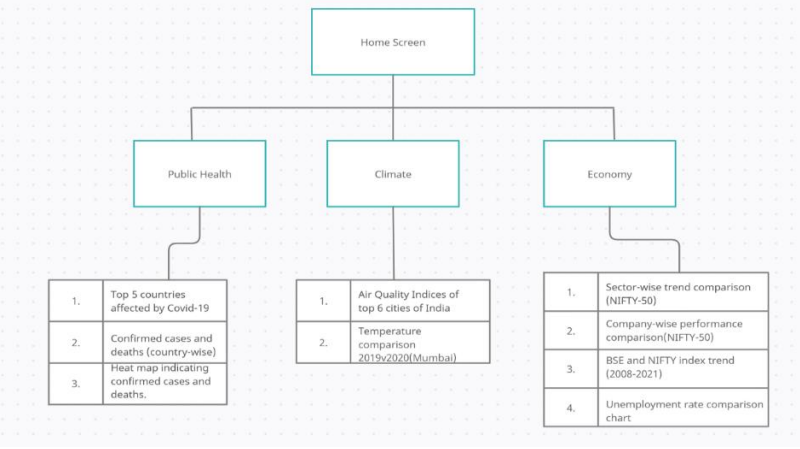
**Requirement Analysis:**

• Visualization Modules: Stream-lit, Matplotlib, Plotly, Folium.

• Data Reading & Storage Modules: Pandas, Pandas Data Reader, MS Excel.

• Computation Modules: NumPy, Date-Time, SK-learn, TensorFlow.





**System Requirements:**

**Hardware requirements:**

Processor: Pentium(R)Dual Core CPU RAM: 2 GB

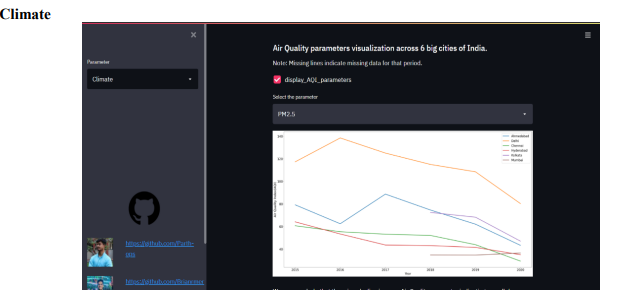
**Software requirements:**

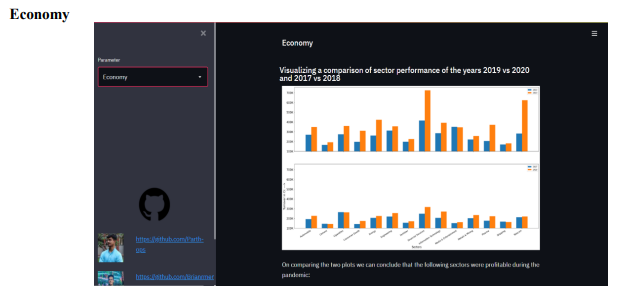
Operating system: Windows 7/8/10 Environment:Streamlit and Jupyter Notebook Python Version: 3.7+ The following libraries and modules are required for project implementation: o NumPy o Streamlit o Pandas o Matplotlib o Plotly o Sci-kit learn o TensorFlow o Folium o Pandas data-reader o Datetime

**Solution Methodology**

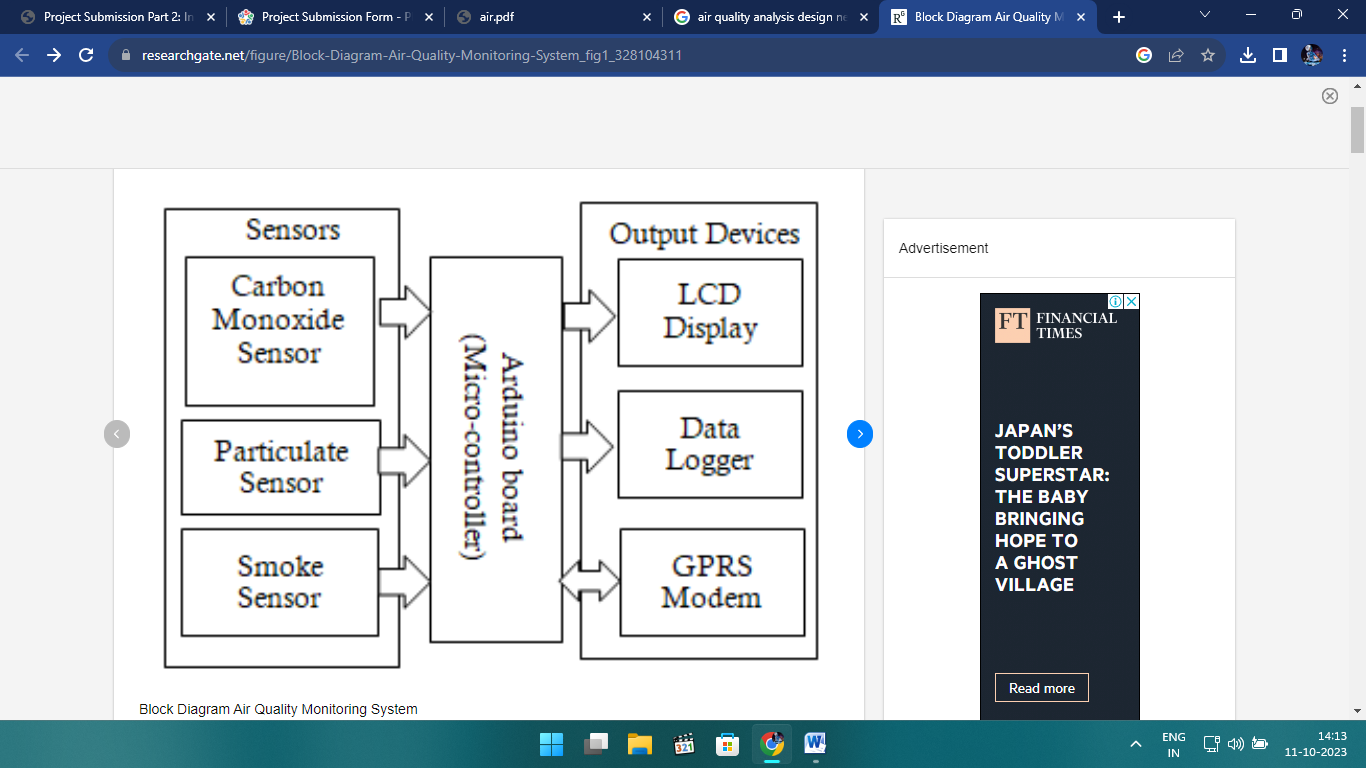
**Climate Screen:**

Data used is static and is read into a Pandas data-frame. • For the first graph, the data is segregated into 6 data frames of 6 different cities. Then the AQI indices are given as parameters to the plot function which are selected from a drop-down list in the Streamlit app. • For the second graph, the data is cleaned using dropna and fillna functions, segregated into different time intervals(2019 and 2020) and visualized in the form a comparison bar chart.





**Development phase**



**MODULES**

**CARBON MONOXIDE SENSOR:**

A carbon monoxide (CO) sensor is a device designed to detect and measure the presence of carbon monoxide gas in the environment. Carbon monoxide is a colorless, odorless, and tasteless gas that is produced as a byproduct of incomplete combustion of carbon-containing fuels, such as natural gas, gasoline, wood, and coal. It is highly toxic and can be life-threatening if it accumulates in enclosed spaces.

**PARTICULATE SENSOR:**

A particulate sensor, also known as a particle sensor or particle counter, is a device used to measure and monitor the concentration of particulate matter (PM) in the air. Particulate matter refers to tiny solid or liquid particles suspended in the air, which can vary in size and composition. These particles can come from various sources, including industrial processes, vehicle emissions, construction, and natural sources like dust and pollen.

**SMOKE SENSOR:**

A smoke sensor, also known as a smoke detector or smoke alarm, is a device that is designed to detect the presence of smoke in the air. It is an important component of fire detection and safety systems. Smoke sensors are commonly found in homes, businesses, and industrial settings to provide early warning of a fire, allowing people to evacuate and take appropriate action to mitigate the danger.

**LCD DISPLAY:**

An LCD (Liquid Crystal Display) is a type of flat-panel display technology that is commonly used in a wide range of electronic devices, including computer monitors, television screens, smartphones, tablets, and more. It works by using the properties of liquid crystals to control the passage of light through the display.

LCD displays have several advantages, including their thin profile, energy efficiency, and the ability to produce high-resolution images. However, they also have some limitations, such as limited viewing angles, potential motion blur in fast-moving scenes, and lower contrast ratios compared to other display technologies like OLED (Organic Light Emitting Diode) displays.

OLED displays have become increasingly popular in recent years due to their improved contrast ratios and faster response times, but LCDs are still widely used in many devices due to their cost-effectiveness and versatility.

**DATA LOGGER:**

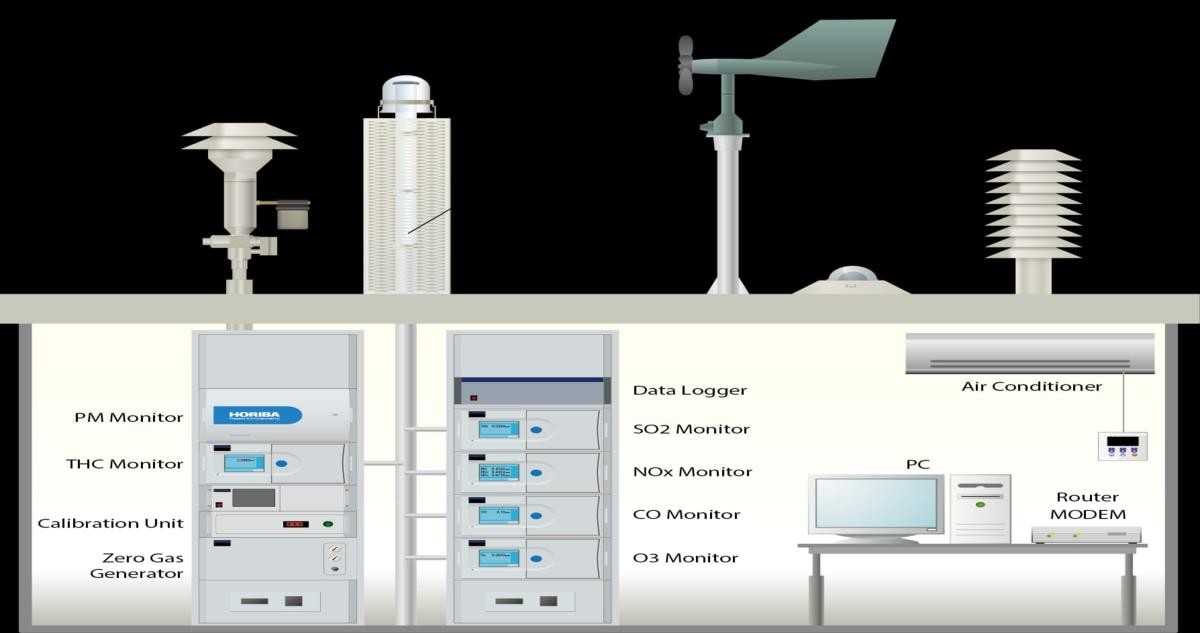
A data logger is an electronic device or system designed to record and store data over time. It is commonly used in various applications to monitor and collect information from sensors, instruments, or other sources. Data loggers are especially useful in situations where manual data collection is impractical, labor-intensive, or prone to errors.

Data loggers come in various forms, from small, portable devices to larger, stationary systems, depending on the specific needs of the application. The choice of a data logger depends on the type of data to be collected, the environmental conditions, and the required level of accuracy and reliability.

**GPRS MODEM:**

A GPRS (General Packet Radio Service) modem is a device that enables communication over the GPRS network, which is a 2G mobile data service. GPRS is a packet-switched technology that allows for data transmission over mobile networks, making it possible to send and receive data over a cellular connection, including text messages, email, and internet browsing.

GPRS modems were more commonly used in the early 2000s when faster data technologies were less widespread. Today, for most data-intensive applications, users tend to prefer 3G, 4G, or 5G modems for faster and more reliable data connectivity.



AQMS, the doctor for "Human Health"

* + An Air Quality Monitoring Station (AQMS) is a system that measures metrological parameters such as wind speed, wind direction, rainfall, radiation, temperature, barometric pressure and ambient parameters. The AQMS also integrates a series of ambient analyzers to monitor the concentration of air.
  + pollutants (such as SO2, NOx, CO, O3, THC, PM, etc.), continuously. HORIBA also provides mobile monitoring stations that can be used to monitor ambient conditions at multiple sites.
  + HORIBA has more than 50 years experience providing ambient monitoring solutions, recognized around the world. HORIBA has supplied over 15,000 units with the major share in many regions. The monitoring station is tailor-made according to the



* + customer's request. HORIBA can provide several types of stations, calibration equipment and more to meet your challenging monitoring requirements.
  + The measured data can be remotely monitored and exported in various formats to the local central authorities. The data can be published via the Internet for easy public access to raise awareness on current air pollution levels. This way, the public can prevent outdoor activities and reduce health impacts during heavy polluted days.

# PROGRAM:

from tkinter import \*

import requests

from bs4 import BeautifulSoup

# link for extract html data

def getdata(url):

r = requests.get(url)

return r.text

def airinfo():

soup = BeautifulSoup(htmldata, 'html.parser')

res\_data = soup.find(class\_="DonutChart--innerValue--2rO41 AirQuality--extendedDialText--2AsJa").text

air\_data = soup.find\_all(class\_="DonutChart--innerValue--2rO41 AirQuality--pollutantDialText--3Y7DJ")

air\_data=[data.text for data in air\_data]

ar.set(res\_data)

o3.set(air\_data[0])

no2.set(air\_data[1])

so2.set(air\_data[2])

pm.set(air\_data[3])

pml.set(air\_data[4])

co.set(air\_data[5])

res = int(res\_data)

if res <= 50:

remark = "Good"

impact = "Minimal impact"

elif res <= 100 and res > 51:

remark = "Satisfactory"

impact = "Minor breathing discomfort to sensitive people"

elif res <= 200 and res >= 101:

remark = "Moderate"

impact = "Breathing discomfort to the people with lungs, asthma and heart diseases"

elif res <= 400 and res >= 201:

remark = "Very Poor"

impact = "Breathing discomfort to most people on prolonged exposure"

elif res <= 500 and res >= 401:

remark = "Severe"

impact = "Affects healthy people and seriously impacts those with existing diseases"

res\_remark.set(remark)

res\_imp.set(impact)

# object of tkinter

# and background set to grey

master = Tk()

master.configure(bg

'light grey')

# Variable Classes in tkinter

air\_data = StringVar()

ar = StringVar()

o3 = StringVar

no2 = StringVar()

so2 = StringVar()

pm = StringVar()

pml = StringVar()

co = StringVar()

res\_remark = StringVar()

res\_imp = StringVar()

# Creating label for each information # name using widget Label

Label(master, text="Air Quality : ",

bg="light grey").grid(row=0, sticky=W)

Label(master, text="O3 (

μg/3) :"

bg="light grey").grid(row=1, sticky=W)

Label(master, text="NO2 (μg/m3) :",

bg="light grey").grid(row=2, sticky=W)

Label(master, text="SO2 (μg/m3) :",

bg="light grey").grid(row=3, sticky=W)

Label(master, text="PM2.5 (μg/m3) :",

bg="light grey").grid(row=4, sticky=W)

Label(master, text="PM10 (μg/m3) :", bg="light grey").grid(row=5, sticky=W)

Label(master, text="CO (μg/m3) :",

bg="light grey").grid(row=6, sticky=WLabel(master, text="Remark :",

bg="light grey").grid(row=7, sticky=W)

Label(master, text="Possible Health Impacts :",

bg="light grey").grid(row=8, sticky=W)

# Creating label for class variable # name using widget Entry

Label(master, text="", textvariable=ar,

bg="light grey").grid(

row=0, column=1, sticky=W)

Label(master, text="", textvariable=o3,

bg="light grey").grid(

row=1, column=1, sticky Label(master, text="", textvariable=no2,

bg="light grey").grid(

row=2, column=1, sticky=W)

Label(master, text="", textvariable=so2,

bg="light grey").grid(

row=3, column=1, sticky=W)

Label(master, text="", textvariable=pm,

bg="light grey").grid(

row=4, column=1, sticky=W)

Label(master, text="", textvariable=pml,

bg="light grey").grid(

row=5, column=1, sticky= Label(master, text="", textvariable=co,

bg="light grey").grid(

row=6, column=1, sticky=W)

Label(master, text="", textvariable=res\_remark,

bg="light grey").grid(row=7, column=1, sticky=W)

Label(master, text="", textvariable=res\_imp,

bg="light grey").grid(row=8, column=1, sticky=W)

# creating a button using the widget

b = Button(master, text="Check",

command=airinfo, bg="Blue")

b.grid(row=0, column=2, columnspan=2,

rowspan=2, padx=5, pady=5,)

mainloop

**Model training:**

Training the model with Prophet is really easy. The team copied the mechanism used is scikit packages: fit() and predict()

1. Creating a model: introduce to basic setup of folder, install pandas , matplotlib , seaborn (using pip for Python package), Anaconda is a good choice if you are using Windows (or even Mac, Linux). ...
2. basic use of those tools (clean, explore, plot, interpret)
3. work with a CSV file from Airnow.gov.

**Feature engineering:**

Feature engineering is a crucial step in any data analysis or machine learning project, including air quality analysis. It involves creating new features or transforming existing ones to better represent the underlying patterns in the data. In the context of air quality analysis, the goal is to extract relevant information from various data sources, such as sensors, weather data, and geographical information, to improve the accuracy of predictive models or gain better insights. Here are some feature engineering ideas for air quality analysis:

**Temporal Features:**

Time of Day: Extract hour and day of the week information from the timestamp to account for diurnal and weekly variations.

Time Lags: Create lag features to capture historical trends and autocorrelation in air quality data.

**Meteorological Features:**

Weather Data: Include weather conditions like temperature, humidity, wind speed, and precipitation, as these can influence air quality.

Wind Direction: Convert wind direction to categorical variables (e.g., N, S, E, W) or create polar coordinates.

Seasonal Indicators: Encode seasons or climatic seasons (e.g., summer, winter) based on the calendar date.

**Geospatial Features:**

Location Information: If you have data from multiple monitoring stations, encode station locations or distances to specific landmarks.

Geographical Clusters: Group stations or areas into clusters based on proximity, and use cluster labels as features.

**Historical Features:**

Rolling Statistics: Compute rolling mean, median, standard deviation, or other statistical measures over a specific time window to capture short-term trends.

Historical Max and Min: Track the historical maximum and minimum values of air quality indicators.

**Categorical Features:**

Day of the Week: Convert the day of the week to a categorical feature. Public Holidays: Include a binary variable indicating whether a particular day is a public holiday.

**Interactions:**

Create interaction terms between relevant features. For example, the interaction between temperature and humidity can affect air quality.

**Cyclical Encoding:**

For cyclical features like time, use techniques like circular encoding (e.g., sine and cosine transforms) to handle periodic patterns.

**Outlier Detection:**

Create binary indicators for extreme values or outliers in the air quality data. These can be used to capture unusual events.

**Feature Scaling and Normalization:**

Standardize or normalize features to ensure that they are on a similar scale, especially if you are using algorithms sensitive to feature scaling, like k-means or SVM.

**Dimensionality Reduction:**

Use dimensionality reduction techniques such as Principal Component Analysis (PCA) or t-SNE to reduce the number of features while preserving essential information.

**Textual Features:**

If you have textual data related to air quality events or reports, consider using natural language processing techniques to extract relevant information from text.

**External Data:**

Integrate external data sources like traffic data, population density, industrial activities, or land use data if they are known to impact air quality.

Remember that the choice of features and feature engineering techniques depends on the specific characteristics of your dataset and the goals of your air quality analysis. Careful feature selection and engineering can significantly enhance the performance and interpretability of your models.

**Model evaluation:**

Evaluating a model for air quality analysis is crucial to ensure that it provides accurate and reliable predictions or insights. Here's a general framework for evaluating a model used in air quality analysis:

1. Data Preparation and Splitting:

- Collect and preprocess your air quality data. This includes handling missing values, data scaling, and encoding categorical variables.

- Split your data into training, validation, and test sets to evaluate the model's performance.

2. Choose Evaluation Metrics:

- Select appropriate evaluation metrics based on the nature of your air quality analysis. Common metrics include Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and R-squared (R2).

- Consider specific metrics for air quality, such as the Air Quality Index (AQI) or pollutant-specific metrics like PM2.5 levels.

3. Model Selection:

- Choose the appropriate machine learning or statistical model for your air quality analysis. This might include linear regression, decision trees, random forests, support vector machines, or deep learning models like neural networks.

4. Training and Validation:

- Train your model on the training data.

- Use the validation set to fine-tune hyperparameters and detect overfitting or underfitting.

5. Cross-Validation:

- In cases where you have limited data, consider using k-fold cross-validation to get a better estimate of your model's performance.

6. Model Evaluation:

- Evaluate your mel's performance on the test set using the chosen evaluation metrics.

- Analyze the residuals (the differences between predicted and actual values) to understand where the model might be making errors.

7. Visualizations:

- Create visualizations, such as scatter plots, time series plots, or histograms, to visualize the model's predictions against actual data.

8. Feature Importance:

- If applicable, analyze feature importance to understand which variables have the most significant impact on air quality predictions. This can help in feature selection or data interpretation.

9. Domain Expert Review:

- Involve domain experts to assess the model's outputs and understand the practical implications of the results. They can provide valuable insights and help identify any model biases. 10. Model Interpretability:

- Depending on the model used, consider using techniques like SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) to interpret the model's predictions.

11. Model Deployment and Monitoring:

- If your model is for real-time air quality monitoring, deploy it in a production environment and set up monitoring to ensure it continues to perform well over time. Make necessary updates as needed.

12. Documentation and Reporting:

- Document the entire modeling process, including data sources, preprocessing, model selection, and evaluation results. Share these findings and insights with stakeholders and the wider community.

13. Continuous Improvement:

- Air quality models may need regular updates to adapt to changing environmental conditions. Continuously gather new data, retrain the model, and refine the analysis process.

Remember that the specific evaluation process may vary based on the objectives of your air quality analysis, the data available, and the complexity of the models used. It's important to choose an appropriate model, fine-tune it, and thoroughly validate its performance to ensure reliable air quality predictions.

**Visualization.**

Visualizing air quality analysis is essential for understanding and communicating data effectively. Here are some common ways to visualize air quality data:

1.Time Series Plots: Time series plots are useful for displaying changes in air quality over time. You can use line charts to show the variations in pollutants (e.g., PM2.5, PM10, NO2) over hours, days, or months.

2.Heatmaps: Heatmaps can be used to show spatial variations in air quality. Color-coding the map based on pollutant concentration levels can quickly highlight areas with poor air quality.

3.Bar Charts: Bar charts are useful for comparing air quality parameters between different locations or over different time periods. You can use horizontal or vertical bars to represent pollutant concentrations.

4. Pie Charts: Pie charts are useful for showing the composition of different pollutants in the air. You can create a pie chart to represent the percentage of each pollutant in the total air quality.

5. Box Plots: Box plots provide a summary of the distribution of air quality data, including median, quartiles, and outliers. They are useful for identifying variations and anomalies.

6.Scatter Plots: Scatter plots are valuable for examining relationships between air quality variables. For example, you can create a scatter plot to show the correlation between temperature and ozone levels.

7. Histograms: Histograms can be used to visualize the distribution of a single air quality parameter. They are especially helpful for understanding the frequency of different concentration ranges.

8. Geospatial Maps: Use geospatial maps to display air quality data over a geographic area. Geographic Information Systems (GIS) can help create interactive maps that allow users to explore air quality across different locations.

9. Animated Visualizations: Animation can be used to show how air quality changes over time. For instance, you can create an animation that illustrates the movement of pollutants over the course of a day or a year.

10. Radar Charts: Radar charts can be useful for comparing air quality parameters across different criteria. For example, you can use a radar chart to compare air quality at different monitoring stations based on multiple pollutants.

11.3D Plots: In cases where you have data in three dimensions (e.g., time, location, and pollutant concentration), 3D plots can provide a comprehensive view of the data.

12. Dashboard: Interactive dashboards, often created with tools like Tableau or Power BI, can combine multiple visualization types to provide a comprehensive view of air quality data. Users can filter, explore, and gain insights from the data interactively.

13. Word Clouds: Word clouds can be used to highlight frequently mentioned words or phrases related to air quality in textual data, such as social media comments or news articles.

14. Color-Coded Maps: Use color gradients on maps to represent different levels of air quality. For example, green can indicate good air quality, while red can indicate poor air quality.

15. Violin Plots: Violin plots combine aspects of box plots and kernel density estimation to provide a more detailed view of the distribution of air quality data.

The choice of visualization depends on your data, objectives, and audience. It's often useful to use a combination of these visualization techniques to provide a comprehensive understanding of air quality analysis. Additionally, you can use various data visualization software and programming libraries (e.g., Matplotlib, ggplot2, D3.js) to create these visualizations.

**PROGRAM:**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load your pollution data into a Pandas DataFrame (assuming it's in a CSV file)

data = pd.read\_csv("pollution\_data.csv")

# Data preprocessing (cleaning and grouping)

# Assuming the DataFrame has columns: 'City', 'SO2', 'NO2', 'RSPM\_PM10'

# Replace with your actual column names

data = data.dropna() # Remove rows with missing values

grouped\_data = data.groupby('City').mean() # Calculate average values for each city

# Data visualization

plt.figure(figsize=(12, 6))

# Bar chart for SO2 levels

plt.subplot(131)

sns.barplot(x=grouped\_data.index, y=grouped\_data['SO2'])

plt.title("Average SO2 Levels")

# Bar chart for NO2 levels

plt.subplot(132)

sns.barplot(x=grouped\_data.index, y=grouped\_data['NO2'])

plt.title("Average NO2 Levels")

# Bar chart for RSPM/PM10 levels

plt.subplot(133)

sns.barplot(x=grouped\_data.index, y=grouped\_data['RSPM\_PM10'])

plt.title("Average RSPM/PM10 Levels")

plt.tight\_layout()

plt.show()

**Conclusion:**

Drawing a conclusion from an air quality analysis requires careful consideration of the data and the context in which it was collected. Here's a general outline of how to draw conclusions from an air quality analysis.

()

)

=W)