

AIR QUALITY ANALYSIS IN TAMILNADU

Objective:

The objective is to comprehensively evaluate and manage air quality in TamilNadu to ensure the health and well-being of its inhabitants while minimizing environmental impacts.

Design-Thinking:

Design thinking for air quality analysis in TamilNadu involves a user-centered approach to solving air quality-related problems. Here's a simplified outline of the process:

1. Empathize:

- Understand the needs and concerns of stakeholders, such as residents, environmental agencies, and healthcare professionals.
- Conduct interviews, surveys, and observations to gather insights into specific air quality issues.

2. Define:

- Clearly define the problem based on the insights gained during the empathize stage.
- Create a problem statement that focuses on a specific aspect of air quality, such as indoor pollution or urban outdoor air quality.

3. Ideate:

- Brainstorm and generate a wide range of ideas for addressing the defined problem.
- Encourage creative thinking among team members to come up with innovative solutions.

4. Prototype:

- Create low-fidelity prototypes of potential solutions. This could be in the form of data visualization tools, sensor networks, or apps.
- Test these prototypes internally and gather feedback.

5. Test:

- Test the prototypes with real users or in real-world environments to gather valuable feedback.
- Refine the prototypes based on user feedback and iterate on the design.

6. Implement:

- Develop a final, polished solution based on the feedback and improvements from the testing phase.
- Collaborate with experts in air quality analysis and technology to ensure accuracy and reliability.

7. Evaluate:

- Continuously monitor the effectiveness of the solution in addressing the air quality problem.
- Collect data on air quality parameters and user satisfaction to assess its impact.

8. Iterate:

- Use the evaluation results to make improvements and iterate on the solution.
- Keep refining and enhancing the system to adapt to changing air quality conditions and user needs.

Throughout this process, it's crucial to involve interdisciplinary teams with expertise in environmental science, data analysis, design, and technology. Collaboration and a user-centered approach are key to creating effective solutions for air quality analysis and improvement.

Innovative Process:

An innovative idea for air quality analysis is the use of drone swarms equipped with advanced sensors and AI algorithms. These drones can fly in coordinated patterns over urban areas to continuously monitor air quality in real-time. The data collected can be quickly analyzed to identify pollution sources, trends, and hotspots. This approach offers a dynamic and granular understanding of air quality, allowing for timely interventions and more effective pollution control measures.

Here are the steps involved in using drone swarms equipped with advanced sensors and AI algorithms for air quality analysis:

1. Sensor Deployment:

- Outfit a fleet of drones with advanced air quality sensors capable of measuring various pollutants such as particulate matter (PM2.5 and PM10), nitrogen dioxide (NO2), sulfur dioxide (SO2), carbon monoxide (CO), ozone (O3), and volatile organic compounds (VOCs).

2. Coordination and Communication:

- Implement a communication system to ensure drones can fly in coordinated patterns and share data in real-time. This can involve using technologies like 5G or satellite communication.

3. Flight Planning:

- Develop flight plans that cover specific urban areas efficiently. These plans should take into account factors such as wind patterns, weather conditions, and the locations of pollution sources.

4. Real-Time Data Collection:

- Deploy the drone swarm to fly over the designated areas, collecting air quality data continuously. Drones should follow the planned routes and altitudes while avoiding obstacles and other aircraft.

5. Data Transmission:

- Transmit the collected air quality data in real-time to a central server or cloud-based platform using the communication system. Ensure data security and integrity during transmission.

6. Data Analysis:

- Implement AI algorithms to process the incoming data. These algorithms can identify pollution sources, detect trends, and pinpoint pollution hotspots based on the sensor readings and historical data.

7. Visualization:

- Present the analyzed air quality data in user-friendly dashboards or maps accessible to relevant authorities, environmental agencies, and the public. Visualization tools can help stakeholders understand the air quality situation easily.

8. Alerting and Intervention:

- Set up automated alerting systems that trigger notifications when pollutant levels exceed acceptable thresholds or when unusual pollution patterns are detected. This allows for timely interventions, such as traffic rerouting or emission source inspections.

9. Data Storage and Reporting:

- Archive historical air quality data for trend analysis and reporting. This information can be valuable for long-term policymaking and urban planning.

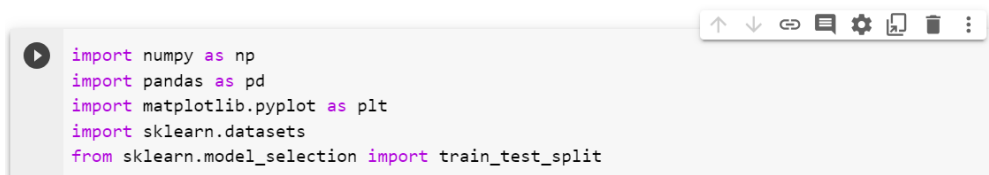
10. Feedback Loop:

- Continuously improve the drone swarm's performance and data analysis algorithms based on feedback and the evolving air quality situation.

By following these steps, a system using drone swarms, advanced sensors, and AI algorithms can provide real-time and comprehensive air quality analysis, aiding in the mitigation of air pollution and its associated health and environmental impacts.

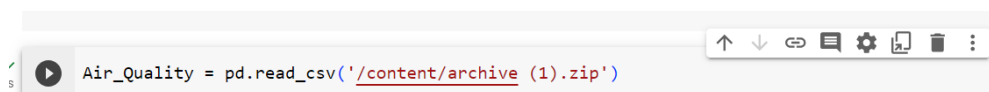
Data Analysis:

For Analyzing the Data collected from the various sources, we want to install and import the necessary libraries



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn.datasets
from sklearn.model_selection import train_test_split
```

After importing the Libraries we need to Load our Datasets



```
Air_Quality = pd.read_csv('/content/archive (1).zip')
```

The Above Picture represents the libraries that are required for the Datasets Operations.

Data Pre-processing:

0s

Air_Quality.head()

</

```
Air_Quality.tail()
```

	Date;Time;CO(GT);PT08.S1(CO);NMHC(GT);C6H6(GT);PT08.S2(NMHC);NOx(GT);PT08.S3(NOx);NO
.....	NaN NaN NaN NaN
	NaN
	NaN
	NaN
	NaN

```
Air_Quality.describe
```

```
<bound method NDFrame.describe of  
Date;Time;CO(GT);PT08.S1(CO);NMHC(GT);C6H6(GT);PT08.S2(NMHC);NOx(GT);PT08.S3(NOx);NO2(GT);PT08.S4(NO2);PT08.S5(O3);T;  
10/03/2004;18.00.00;2      6;1360;150;11      9;1046;166;1056;113;1692;1268;13  6;48  9;0  
7578;;  
10/03/2004;19.00.00;2;1292;112;9  4;955;103;1174;92;1559;972;13  3;47      7;0  7255;;  
NaN  
10/03/2004;20.00.00;2      2;1402;88;9      0;939;131;1140;114;1555;1074;11  9;54  0;0  
7502;;  
10/03/2004;21.00.00;2      2;1376;80;9      2;948;172;1092;122;1584;1203;11  0;60  0;0  
7867;;  
10/03/2004;22.00.00;1      6;1272;51;6      5;836;131;1205;116;1490;1110;11  2;59  6;0  
7888;;  
...  
...  
;;;;;;;;;;;;; NaN      NaN      NaN  NaN  
NaN  
  
NaN  
  
NaN  
  
NaN  
  
NaN
```

```
Air_Quality.info()

<class 'pandas.core.frame.DataFrame'>
MultiIndex: 9471 entries, ('10/03/2004;18.00.00;2', '6;1360;150;11', '9;1046;166;1056;113;1692;1268;13', '6;48', '9;0'
Data columns (total 1 columns):
#    Column
---  ---
0    Date;Time;CO(GT);PT08.S1(CO);NMHC(GT);C6H6(GT);PT08.S2(NMHC);NOx(GT);PT08.S3(NOx);NO2(GT);PT08.S4(NO2);PT08.S5(O3)
dtypes: object(1)
memory usage: 1.2+ MB
```

```
Air_Quality.info()

<class 'pandas.core.frame.DataFrame'>
MultiIndex: 9471 entries, ('10/03/2004;18.00.00;2', '6;1360;150;11', '9;1046;166;1056;113;1692;1268;13', '6;48', '9;0'
Data columns (total 1 columns):
#    Column
---  ---
0    Date;Time;CO(GT);PT08.S1(CO);NMHC(GT);C6H6(GT);PT08.S2(NMHC);NOx(GT);PT08.S3(NOx);NO2(GT);PT08.S4(NO2);PT08.S5(O3)
dtypes: object(1)
memory usage: 1.2+ MB
```

```
[7] Air_Quality.isnull().sum()

Date;Time;CO(GT);PT08.S1(CO);NMHC(GT);C6H6(GT);PT08.S2(NMHC);NOx(GT);PT08.S3(NOx);NO2(GT);PT08.S4(NO2);PT08.S5(O3);T;RI
2556
dtype: int64

Air_Quality.shape

(9471, 1)
```

from sklearn.preprocessing import StandardScaler

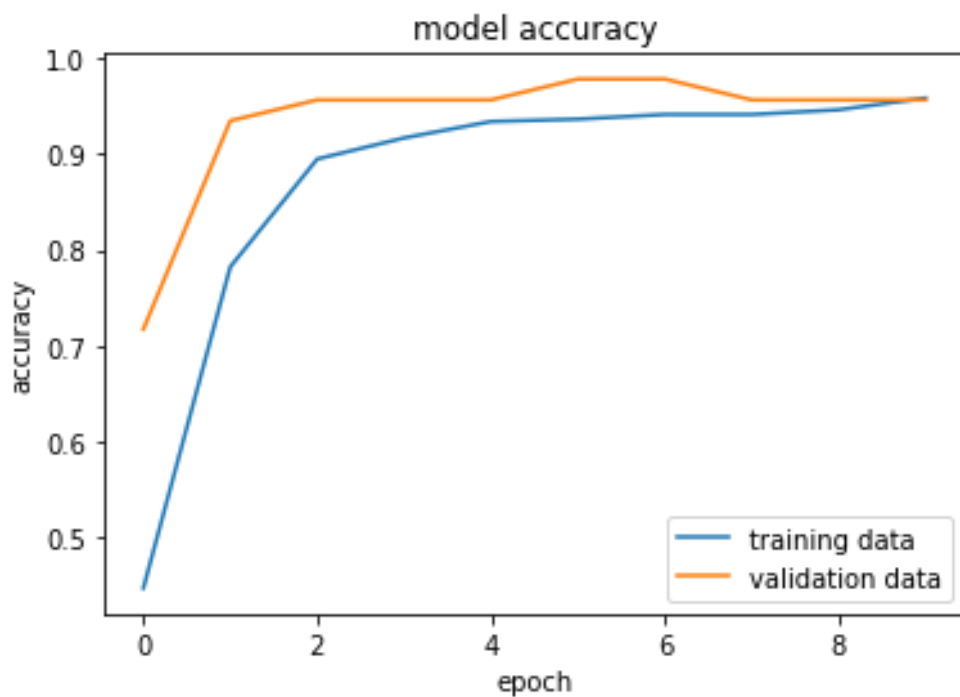
scaler = StandardScaler()

X_train_std = scaler.fit_transform(X_train)

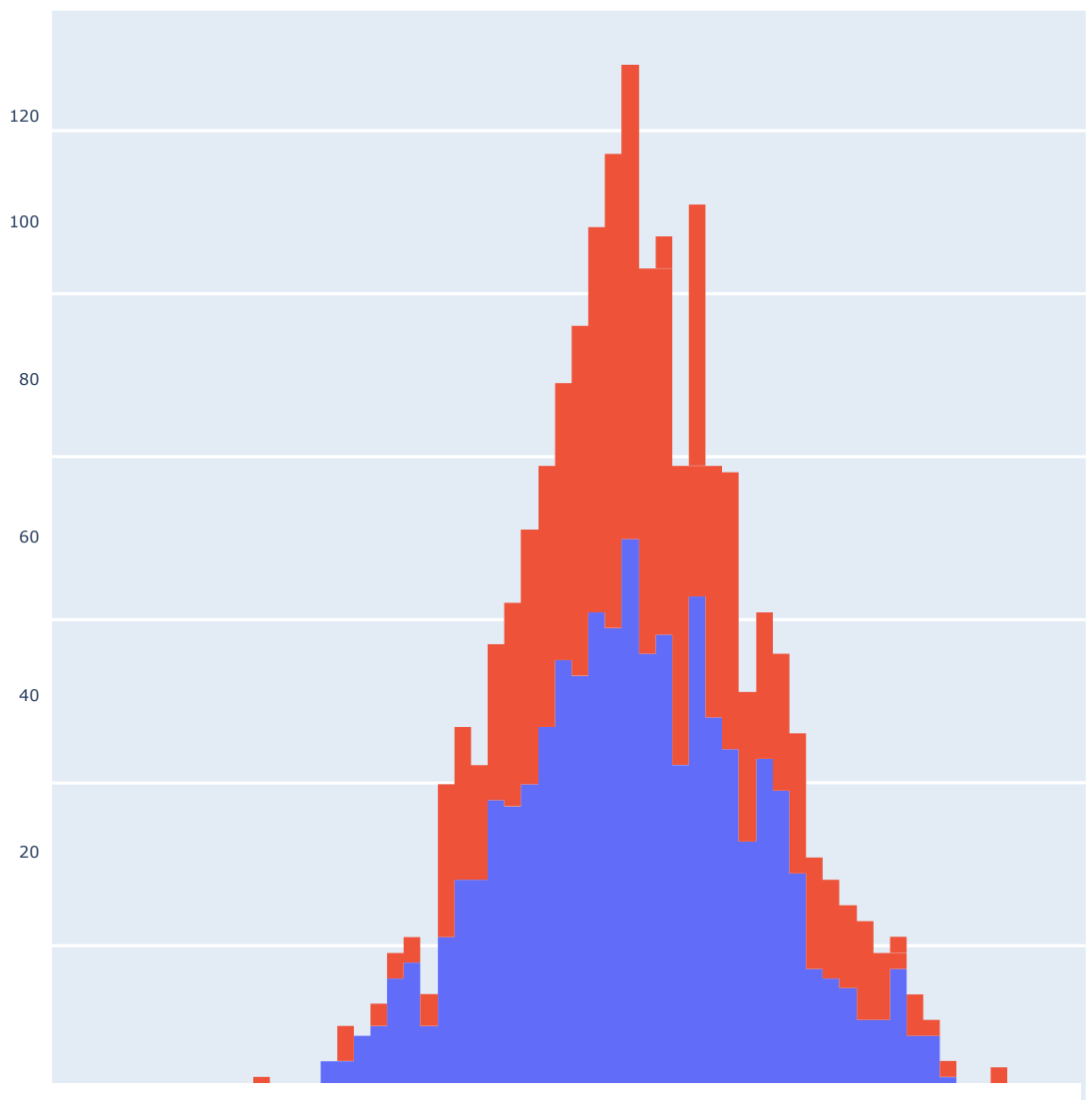
X_test_std = scaler.transform(X_test)

#Accuracy and Loss Visualization

```
plt.plot(history.history['accuracy'])  
plt.plot(history.history['val_accuracy'])  
plt.title('model accuracy')  
plt.ylabel('accuracy')  
plt.xlabel('epoch')  
plt.legend(['training data', 'validation data'], loc = 'lower right')
```



Factors Affecting Air Quality:

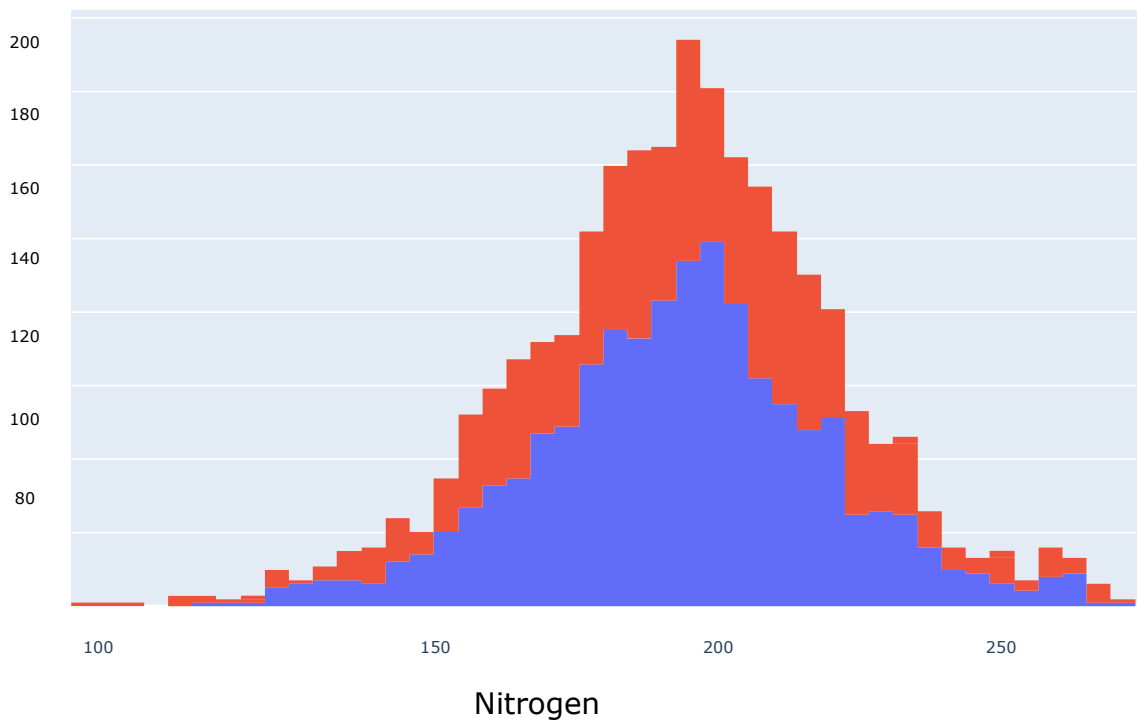


```
import plotly.express as px

data = data

figure = px.histogram(data, x = "NO2",
                      color = "Potability",
                      title= "Factors Affecting Air Quality: NO2")

figure.show()
```



```
import plotly.express as px

data = data

figure = px.histogram(data, x = "Nitrogen",

color = "Potability",

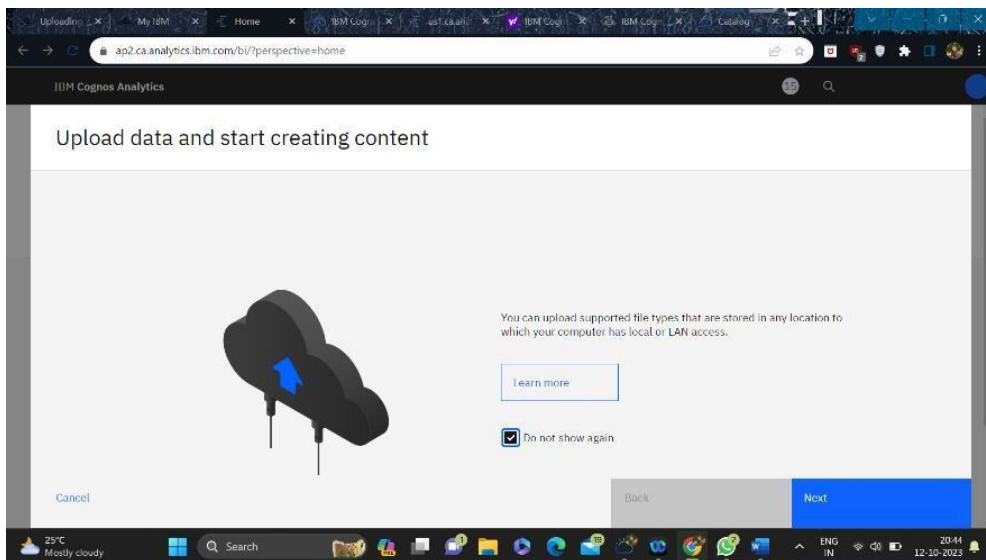
title= "Factors Affecting Air Quality: Nitrogen")

figure.show()
```

Visualization using IBM Cognos:

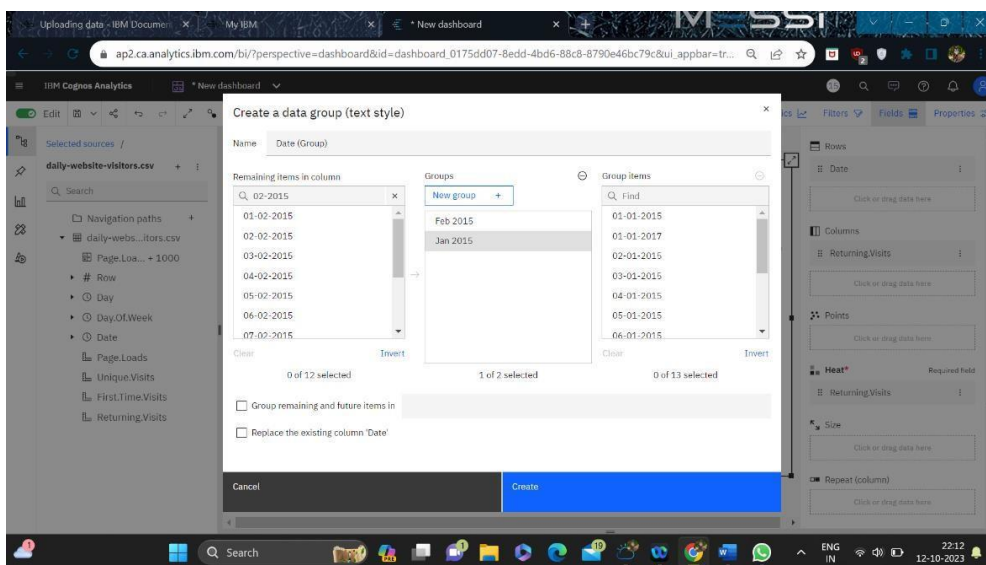
For using IBM Cognos we need to create an IBM Account first then using that IBM Account we can login IBM Cognos

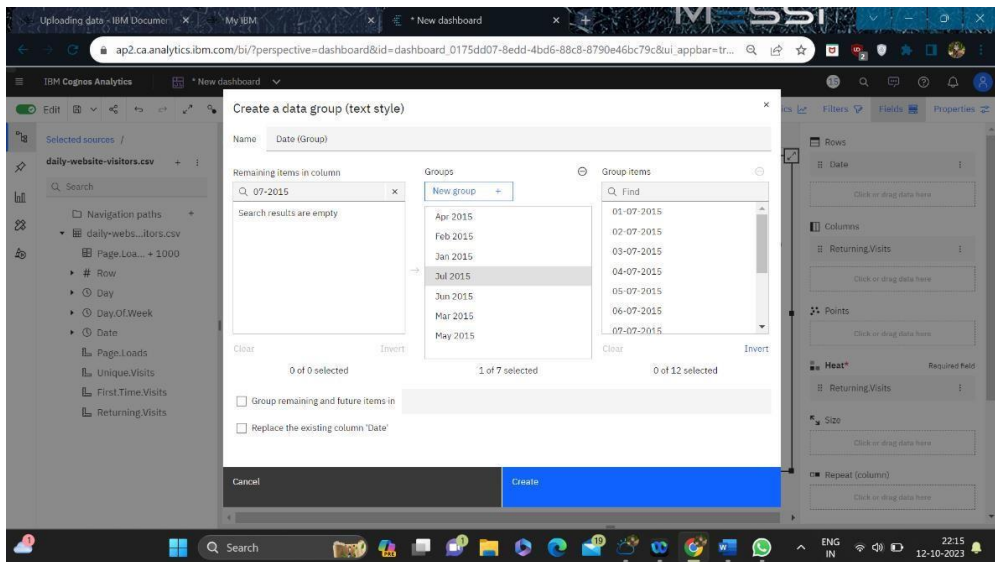
For visualization we need to upload our Dataset in IBM Cloud



Preprocessing Dataset:

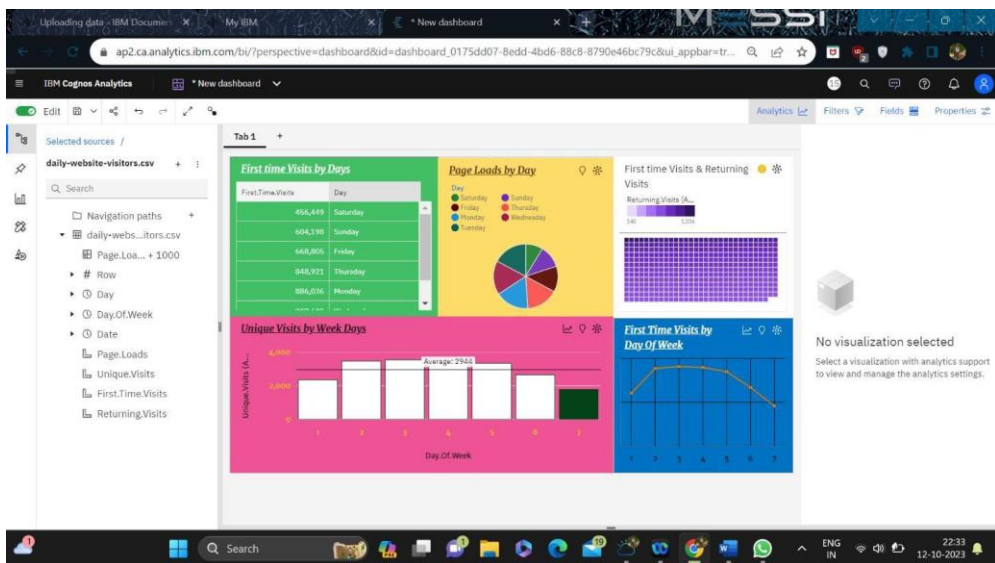
1. Clean the data by handling missing values, outliers, and data quality issues.
2. Transform and reshape the data as needed. This might include feature engineering, data aggregation, or other data preparation.



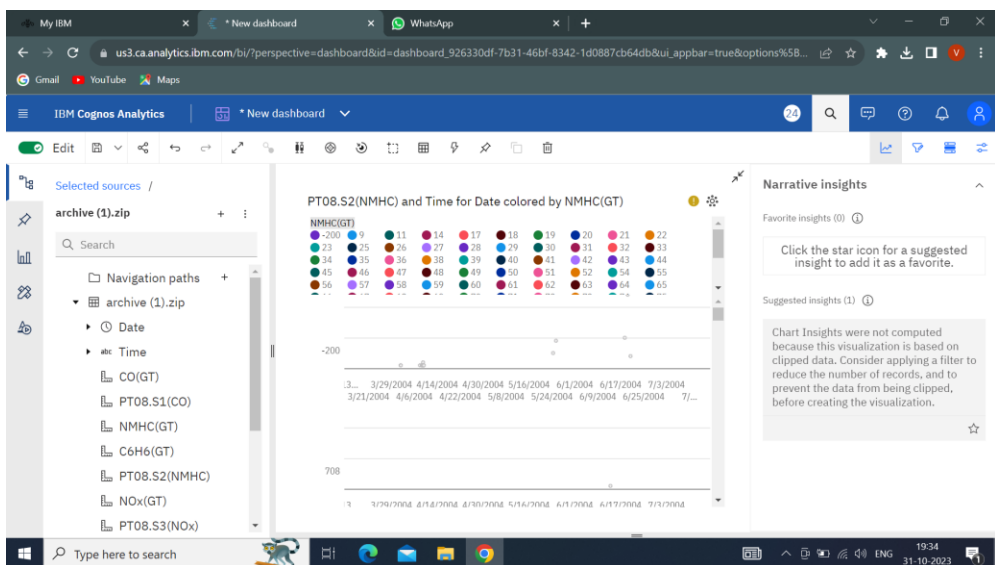
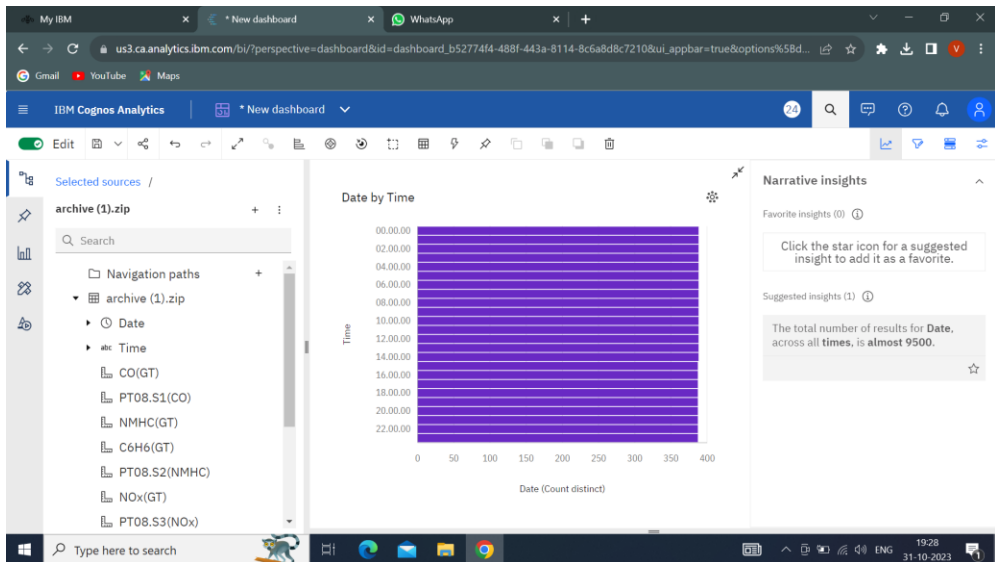


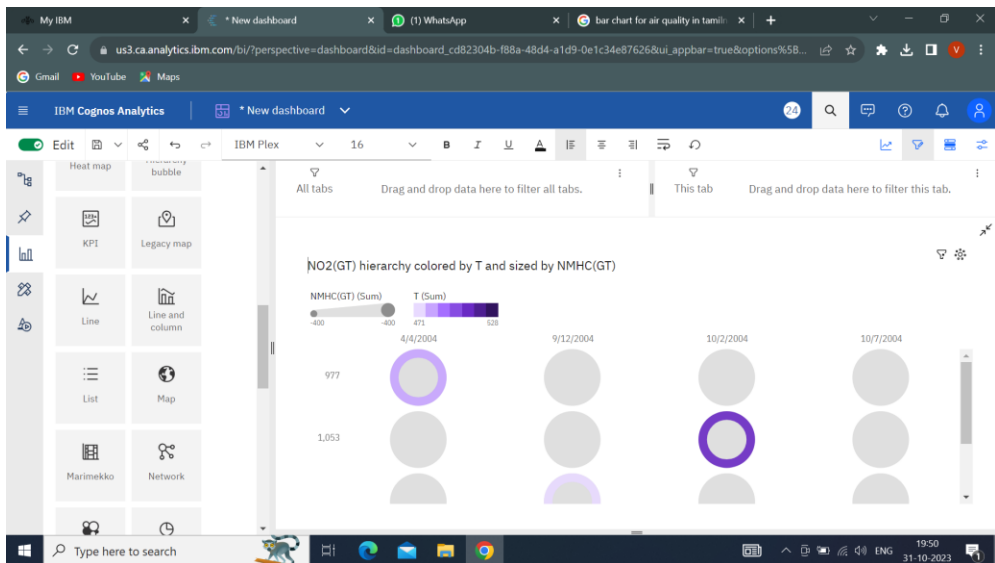
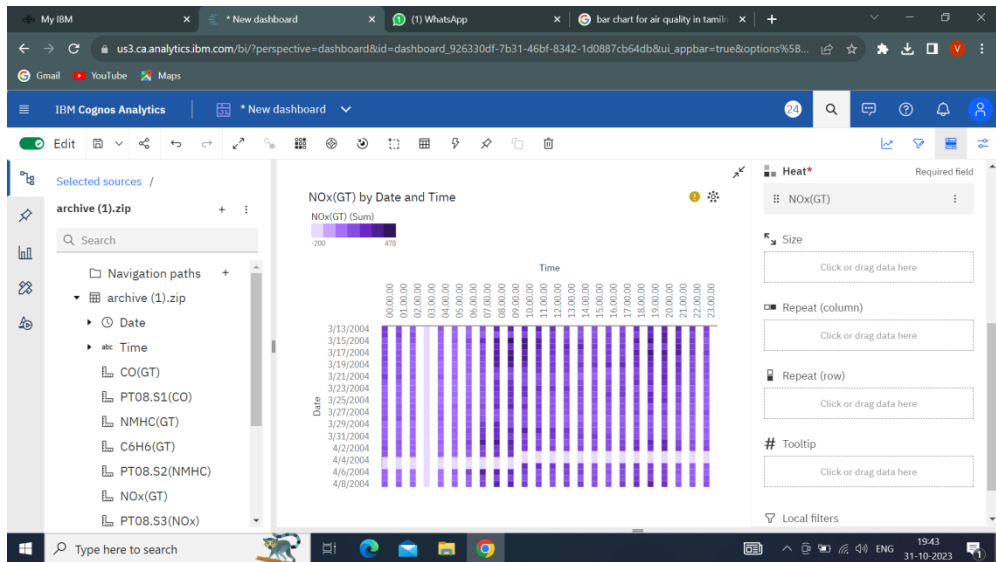
Visualization:

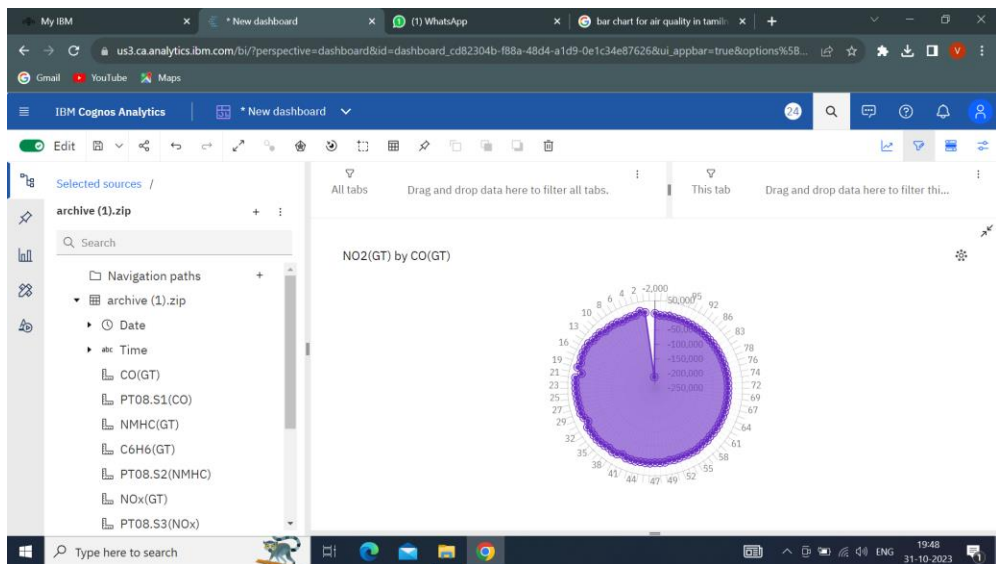
- Create visualizations using IBM Cognos to present your findings. This might include:
- Histograms, scatter plots, and box plots to visualize data distributions and relationships.
- Line charts or time series visualizations to explore trends over time.
- Dashboards that summarize key insights in a user-friendly manner.



Visualization of Air Quality using IBM Cognos







Conclusion:

The data collected can be quickly analyzed to identify pollution sources, trends, and hotspots. This approach offers a dynamic and granular understanding of air quality, allowing for timely interventions and more effective pollution control measures.