# AIR QUALITY INDEX ANALYZER

AN INDUSTRY ORIENTED MINI REPORT

Submitted to

**JAWAHARLAL NEHRU TECNOLOGICAL UNIVERSITY, HYDERABAD**

In partial fulfillment of the requirements for the award of the degree of

**BACHELOR OF TECHNOLOGY**

**In**

**COMPUTER SCIENCE AND ENGINEERING(AI&ML)**

Submitted By


| | |
|---|---|
| **YESHWANTH EDLA** | **20UK1A6649** |
| **HARSHITH KUMAR SAMALA** | **20UK1A6624** |
| **RISHITHA GANGULA** | **20UK1A6651** |
| **SAI KUMAR JAKKULA** | **20UK1A6608** |


Under the guidance of

**Mr. T. Sanath kumar**

Assistant Professor



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**VAAGDEVI ENGINEERING COLLEGE**

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) –

506005

**DEPARTMENT OF**

**COMPUTER SCIENCE AND ENGINEERING(AI&ML)**

**VAAGDEVI ENGINEERING COLLEGE(WARANGAL)**


## CERTIFICATE OF COMPLETION
## INDUSTRY ORIENTED MINI PROJECT

This is to certify that the UG Project Phase-1 entitled "AIR QUALITY INDEX (AQI) ANALYZER" is being submitted by YESHWANTH EDLA(20UK1A6649),SAMALA HARSHITH KUMAR(20UK1A6624),RISHITHA GANGULA(20UK1A6651),SAI KUMAR JAKKULA(20UK1A6608) in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering to Jawaharlal Nehru Technological University Hyderabad during the academic year 2023-2024.

| | |
|---|---|
| **Project Guide** | **HOD** |
| **Mr.T.Sanath kumar** | **Dr. K. Sharmila** |
| (Assistant Professor) | (Professor) |

**External**

# ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr.P.PRASAD RAO,** Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG Project Phase-1 in the institute.

We extend our heartfelt thanks to **Dr.K.SHARMILA**, Head of the Department of CSE, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the UG Project Phase-1.

We express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the UG Project Phase-1 and for their support in completing the UG Project Phase-1.

We express heartfelt thanks to the guide, **T.SANATH KUMAR ,** Assistant professor, Department of CSE for his constant support and giving necessary guidance for completion of this UG Project Phase-1.

Finally, we express our sincere thanks and gratitude to my family members, friends for their encouragement and outpouring their knowledge and experience throughout the thesis.

**YESHWANTH EDLA (20UK1A6649)**
**HARSHITH KUMAR SAMALA (20UK1A6624)**
**RISHITHA GANGULA (20UK1A6651)**
**SAI KUMAR JAKKULA (20UK1A6608)**

# ABSTRACT

The Air Quality Index (AQI) is a crucial tool for assessing air quality based on pollutant concentrations. It categorizes air quality into six buckets: Good, Satisfactory, Moderate, Poor, Very Poor, and Severe, with specific AQI value ranges, associated symptoms, diseases, and precautions. Good signifies excellent air quality with no notable symptoms, while Severe indicates life-threatening pollution with severe health risks. Understanding the AQI allows individuals to make informed decisions about outdoor activities and take appropriate precautions to protect their health. It serves as a valuable resource in promoting awareness and mitigating the adverse effects of air pollution on human well-being.

# TABLE OF CONTENTS:-

# 1.INTRODUCTION

## 1.1.OVERVIEW

Clean and healthy air is a fundamental necessity for human well-being, and the Air Quality Index (AQI) serves as a critical tool for assessing and understanding the quality of the air we breathe. The AQI system categorizes air quality into six distinct buckets, each defined by specific AQI value ranges. These categories, ranging from "Good" to "Severe," provide essential information about the levels of pollutants in the atmosphere and their potential impacts on our health. This introduction explores the AQI classification, the associated symptoms and diseases, as well as the recommended precautions at each level, emphasizing the importance of AQI awareness in safeguarding public health. Understanding the AQI is vital in making informed decisions about outdoor activities and taking proactive measures to minimize the risks associated with varying air quality conditions.

In this overview, we delve into the AQI classification, the associated symptoms and diseases at each level, and the recommended precautions to protect public health. Recognizing the significance of AQI awareness is essential in making informed decisions about outdoor activities and implementing preventive measures to mitigate health risks associated with fluctuating air quality conditions. It is a fundamental aspect of safeguarding the well-being of individuals and communities in an increasingly polluted world.

## 1.2.PURPOSE

The Air Quality Index (AQI) serves several critical purposes in assessing and communicating air quality, with a primary focus on safeguarding public health and the environment. In detail, its purposes include:

1. **Informing the Public**: The AQI is designed to provide the general public with easily understandable information about air quality. It informs individuals about the safety of outdoor activities and helps them make informed decisions to protect their health.
2. **Health Protection**: One of its primary purposes is to protect public health. The AQI categorizes air quality into different levels, each associated with specific health risks. This enables individuals, particularly those with respiratory conditions, to take precautions based on the current air quality conditions.
3. **Government Regulation**: The AQI is used by regulatory agencies and governments to set air quality standards and policies. It helps in identifying areas with poor air quality and implementing measures to improve it, such as emission controls and pollution reduction strategies.
4. **Environmental Monitoring**: The AQI also plays a role in monitoring the impact of air pollution on the environment. It helps track changes in air quality over time and assess the effectiveness of pollution control measures.
5. **Economic Impact**: Understanding air quality is essential for evaluating the economic impact of pollution on industries, healthcare costs, and productivity. It can guide policy decisions that affect economic development.

# 2.LITERATURE SURVEY

## 2.1 EXISTING PROBLEM

- The existing problem of air quality degradation is a multifaceted global challenge that spans health, environmental, and economic domains. Poor air quality, characterized by elevated levels of pollutants, takes a devastating toll on public health, contributing to a wide array of respiratory and cardiovascular diseases while being a leading cause of premature mortality.
- Moreover, air pollution wreaks havoc on ecosystems, causing harm to both flora and fauna, and contributes to environmental issues like acid rain and smog. Economically, it translates into substantial costs related to healthcare expenses, lost workdays, and decreased labor productivity. Additionally, air pollution plays a significant role in climate change, with greenhouse gas emissions intensifying global warming. Vulnerable communities, often located near pollution sources, bear a disproportionate burden.
- Inadequate regulations, limited public awareness, and the variability of air quality further compound the issue. Addressing these problems necessitates stringent regulations, the transition to cleaner energy sources, robust public education, and international cooperation to combat the health, environmental, and economic consequences associated with poor air quality.

## 2.2 PROPOSED SOLLUTION

- Our innovative proposed solution leverages advanced predictive modeling to anticipate and communicate Air Quality Index (AQI) bucket categories accurately and efficiently. By forecasting AQI buckets, we enable proactive dissemination of information regarding air quality, accompanying health risks, and recommended precautions. This solution integrates technology, data analytics, and public health awareness to create a comprehensive system for managing air quality issues.

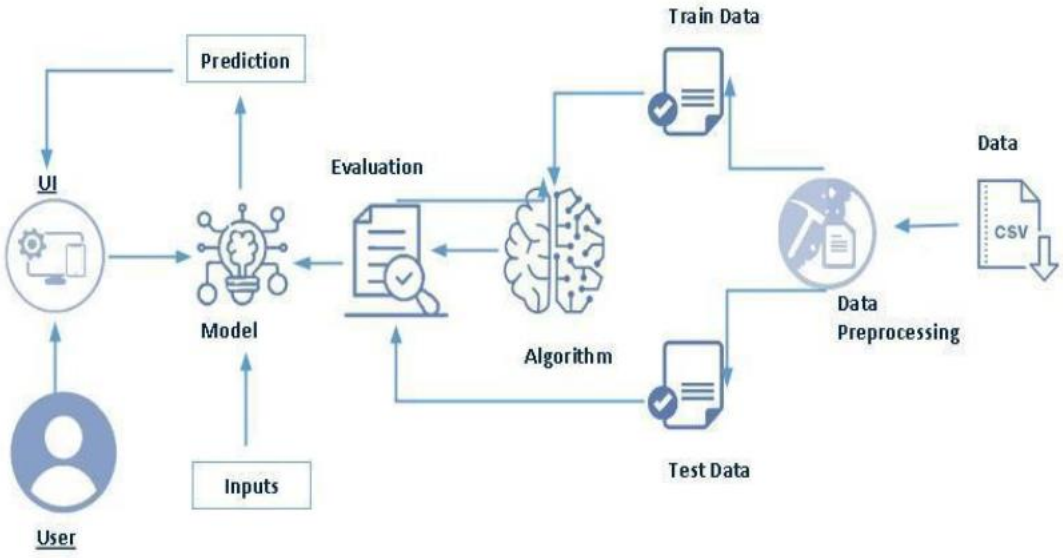
1. **Advanced Predictive Modeling**: We employ sophisticated predictive models that take into account various environmental factors, historical data, and real-time monitoring to forecast AQI buckets. This ensures timely and reliable information for the public.
2. **Customized Health Information**: For each AQI category, our system provides a tailored list of potential diseases and recommended precautions. This empowers individuals to take proactive steps to protect their health based on the forecasted air quality conditions.
3. **User-Friendly Interfaces**: Our solution offers user-friendly interfaces, making it easy for the public to access and understand AQI predictions and associated health information.
4. **Community Engagement**: We actively engage with communities to raise awareness about the AQI forecasting system, ensuring that the public is well-informed and prepared to take necessary precautions.

5. **Educational Initiatives**: Our solution includes educational initiatives to raise awareness about the importance of air quality and the impact of air pollution on public health, empowering individuals to make informed choices

- By accurately predicting AQI bucket categories and providing associated health information, our solution empowers individuals and communities to make informed decisions, protect their health, and reduce the adverse health effects of air pollution. It is a comprehensive approach to proactively address air quality issues and create healthier environments for all.
- By encompassing these elements, our proposed solution creates a comprehensive and adaptable framework for AQI prediction and management, with a strong emphasis on accuracy, user-friendliness, and community engagement.

# 3.THEORITICAL ANALYSIS

## 3.1. BLOCK DIAGRAM




## 3.2. SOFTWARE DESIGNING

The following is the Software required to complete this project:

- **Google Colab**: Google Colab will serve as the development and execution environment for your predictive modeling, data preprocessing, and model training tasks. It provides a cloud-based Jupyter Notebook environment with access to Python libraries and hardware acceleration.

- **Dataset (CSV File)**: The dataset in CSV format is essential for training and testing your predictive model. It should include historical air quality data, weather information, pollutant levels, and other relevant features.
- **Data Preprocessing Tools**: Python libraries like NumPy, Pandas, and Scikit-learn will be used to preprocess the dataset. This includes handling missing data, feature scaling, and data cleaning.

- **Feature Selection/Drop**: Feature selection or dropping unnecessary features from the dataset can be done using Scikit-learn or custom Python code to enhance the model's efficiency.
- **Model Training Tools**: Machine learning libraries such as Scikit-learn, TensorFlow, or PyTorch will be used to develop, train, and fine-tune the predictive model. Regression or classification models can be considered, depending on the nature of the AQI prediction task.
- **Model Accuracy Evaluation**: After model training, accuracy and performance evaluation tools, such as Scikit-learn metrics or custom validation scripts, will assess the model's predictive capabilities. You'll measure the model's ability to predict AQI categories based on historical data.
- **UI Based on Flask Environment**: Flask, a Python web framework, will be used to develop the user interface (UI) for the system. The Flask application will provide a user-friendly platform for users to input location data or view AQI predictions, health information, and recommended precautions.
- Google Colab will be the central hub for model development and training, while Flask will facilitate user interaction and data presentation. The dataset, along with data preprocessing, will ensure the quality of the training data, and feature selection will optimize the model. Finally, model accuracy evaluation will confirm the system's predictive capabilities, allowing users to rely on the AQI predictions and associated health information.

# 4.EXPERIMENTAL INVESTIGATION

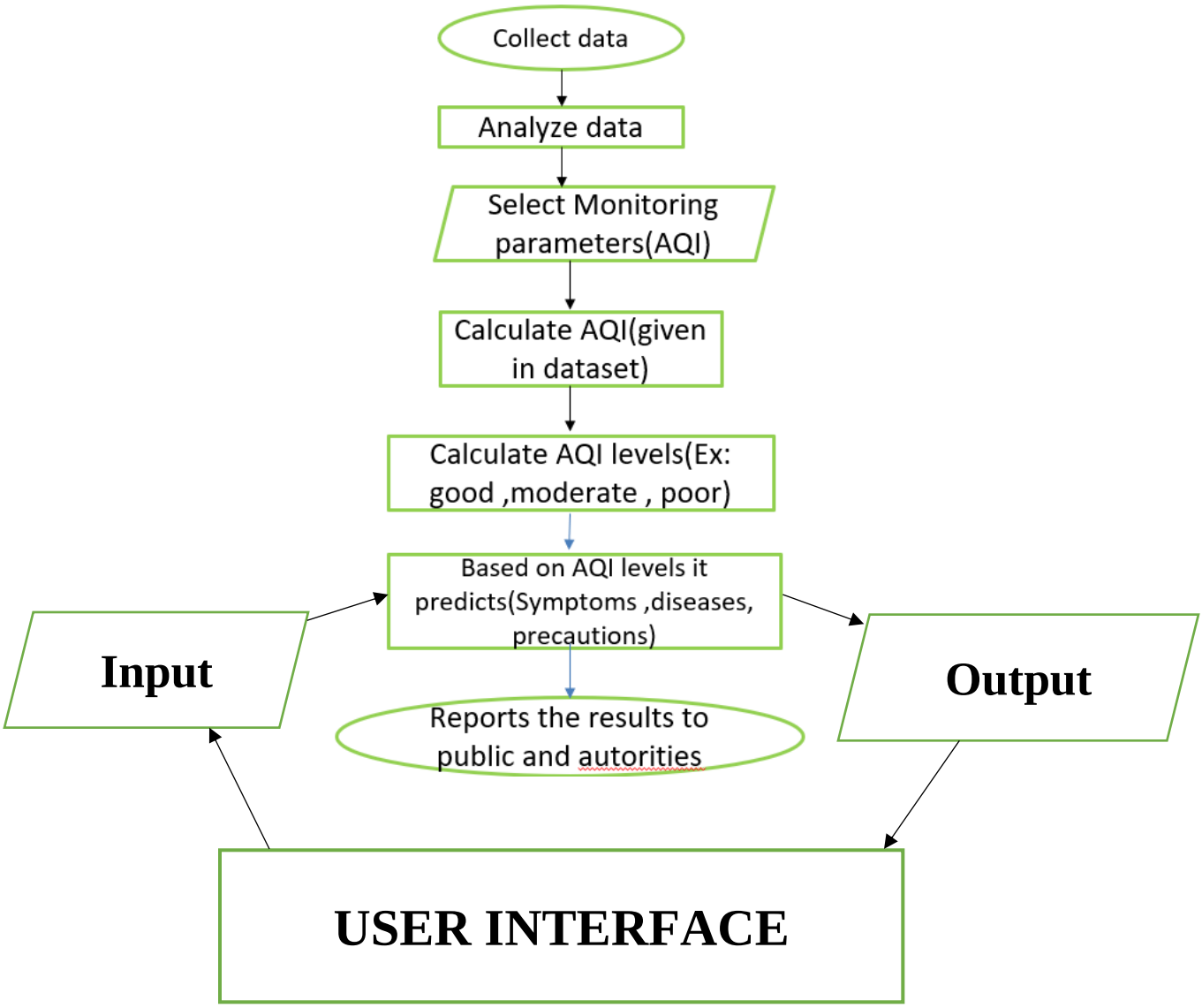
In this project, we have used Air Quality Dataset. This dataset is a csv file consisting of labelled data and having the following columns-

1. **City**: Location or urban area for which air quality data is recorded.
2. **Date**: The specific date on which air quality measurements were taken.
3. **PM2.5**: Particulate Matter with a diameter of 2.5 micrometers, a key air pollutant.
4. **PM10**: Particulate Matter with a diameter of 10 micrometers, another significant air pollutant.
5. **NO**: Nitric Oxide, a gaseous air pollutant.
6. **NO2**: Nitrogen Dioxide, a toxic gas often related to combustion processes.
7. **NOx**: Nitrogen Oxides, a group of nitrogen-containing air pollutants.
8. **NH3**: Ammonia, a compound that can contribute to air pollution.
9. **CO**: Carbon Monoxide, a colorless, odorless gas produced by incomplete combustion.
10. **SO2**: Sulfur Dioxide, a toxic gas often linked to industrial processes.
11. **O3**: Ozone, a secondary pollutant with varying health impacts.
12. **Benzene**: A volatile organic compound that can be harmful when inhaled.
13. **Toluene**: Another volatile organic compound commonly found in urban air.
14. **Xylene**: A group of volatile organic compounds, often associated with vehicle emissions.
15. **AQI**: Air Quality Index, a numerical value indicating overall air quality.
16. **AQI_Bucket**: The qualitative classification of air quality based on the AQI value.

For the dataset we selected, it consists of more than the columns we want to predict it . So, we have chosen the feature drop it contains the columns that we are going to predict the AQI value.

- Feature drop means it drops the columns that we don’t want in our dataset.
- Feature_drop = ['PM10','NH3','Benzene','Toluene','Xylene','index']

# 5.FLOWCHART


Collect data
Analyze data
Select Monitoring parameters(AQI)
Calculate AQI(given in dataset)
Calculate AQI levels(Ex: good ,moderate , poor)
Based on AQI levels it predicts(Symptoms ,diseases, precautions)
Reports the results to public and autorities
Input
Output
USER INTERFACE

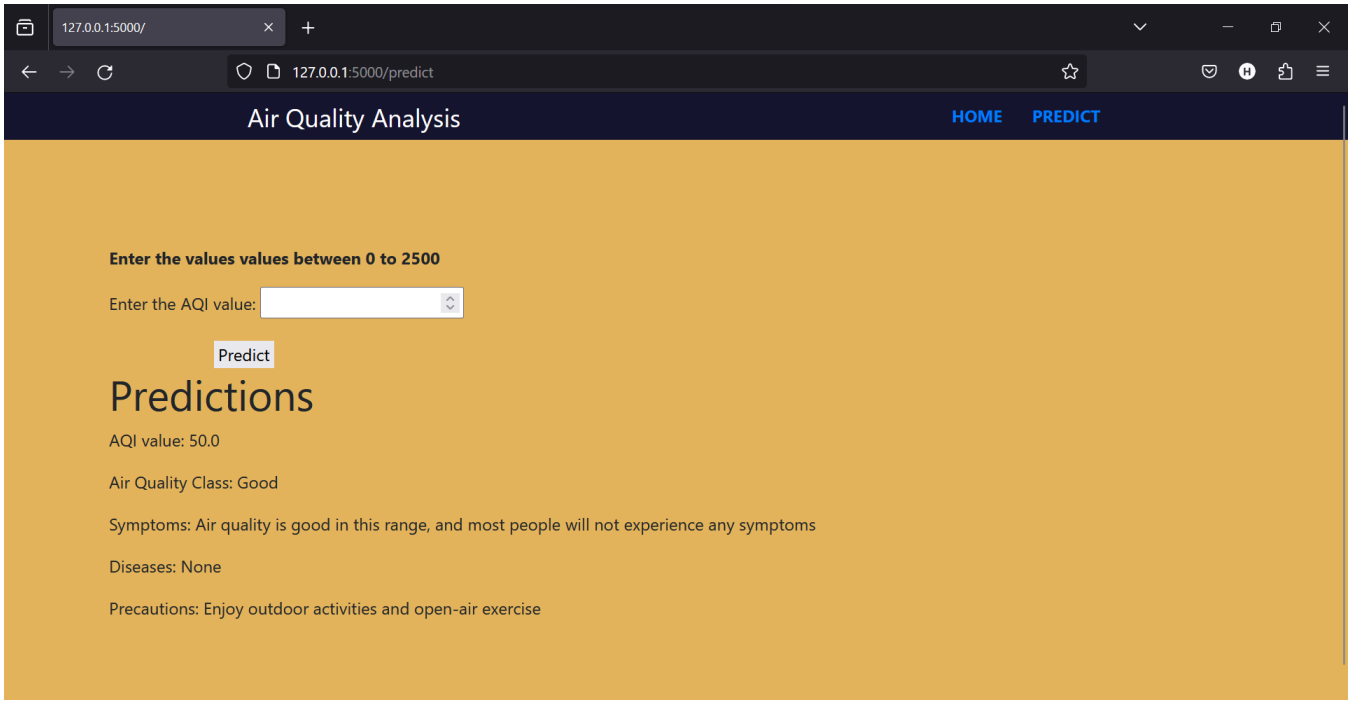
# 6.RESULT

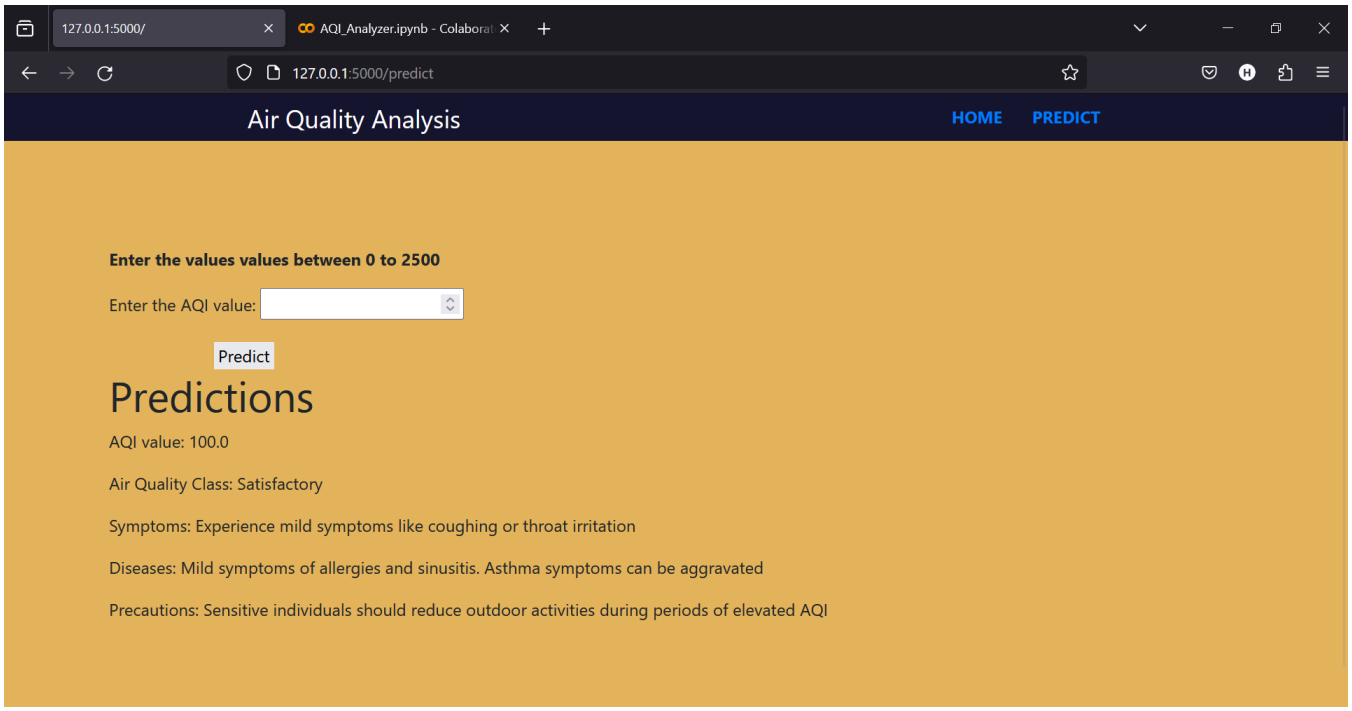
## HOME PAGE

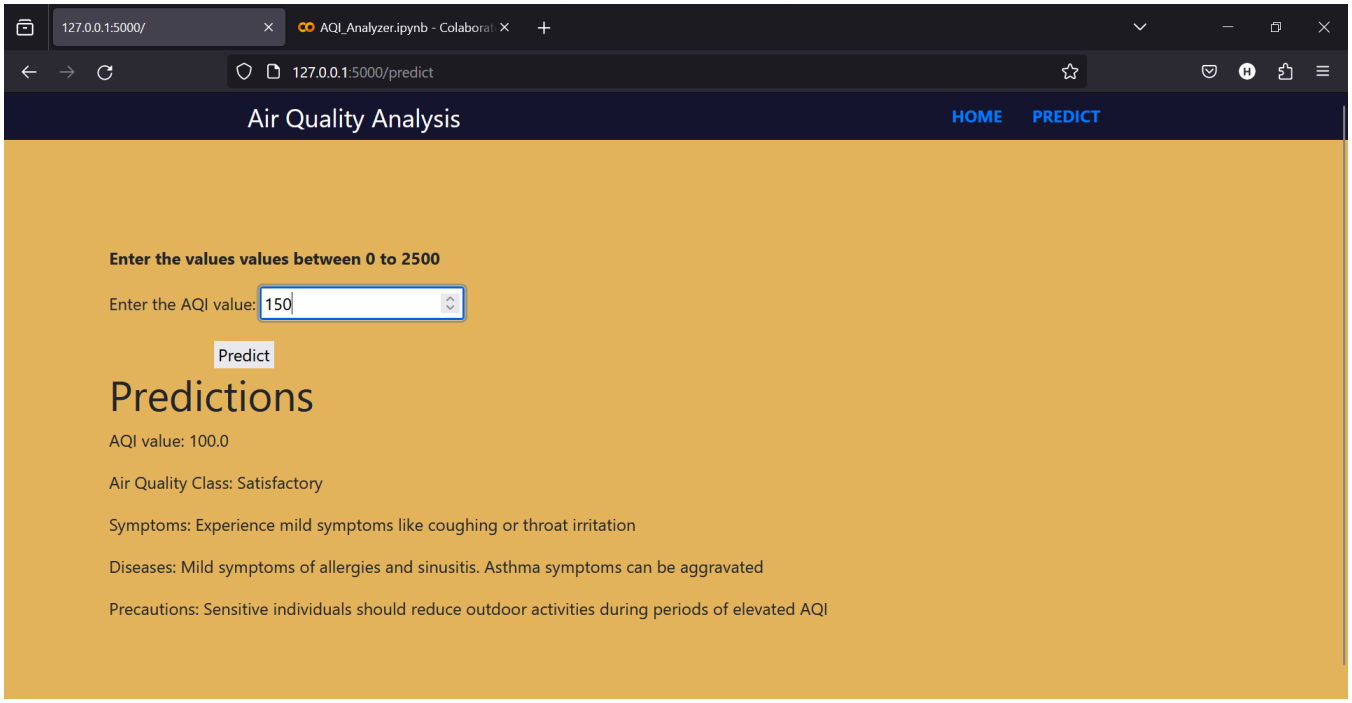

### What is AQI ?

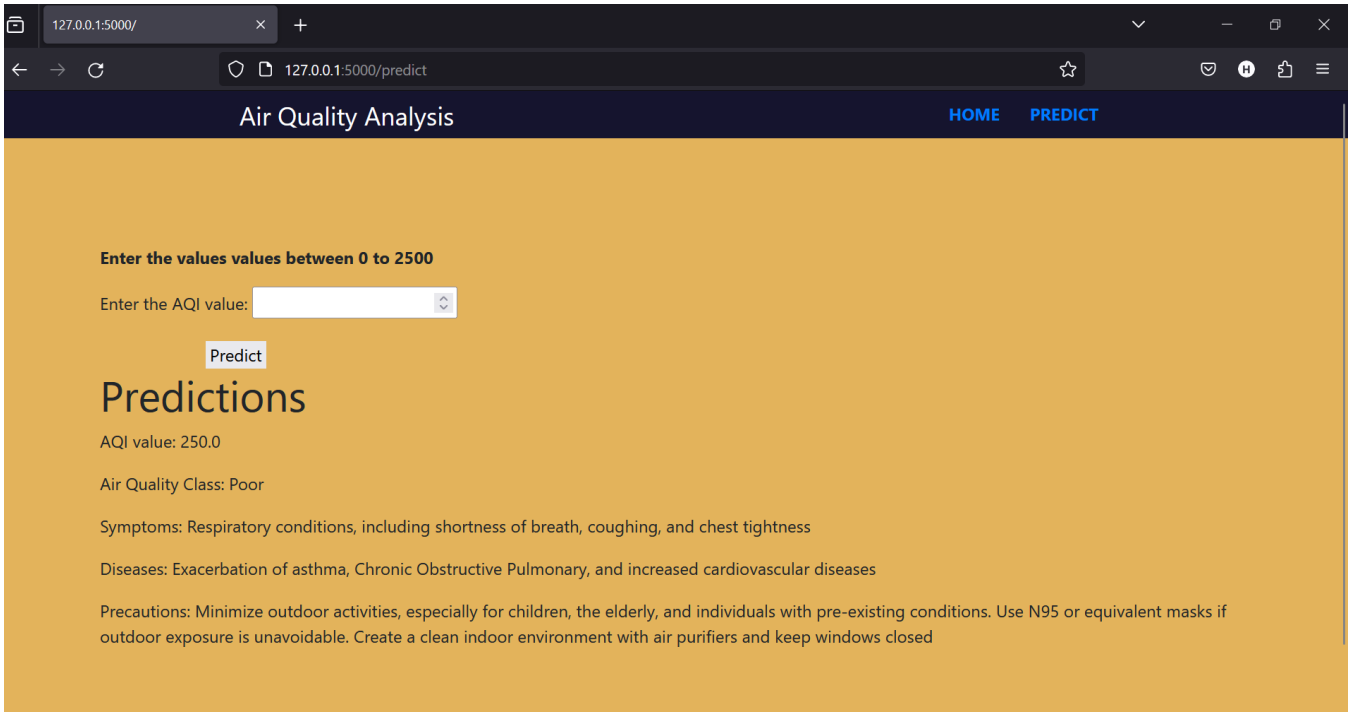
The air quality index (AQI) is an index for reporting air quality on a daily basis. It is a measure of how air pollution affects one's health within a short time period. The purpose of the AQI is to help people know how the local air quality impacts their health The higher the AQI value, the greater the level of air pollution and the greater the health concerns. The concept of AQI has been widely used in many developed countries for over the last three decades. AQI quickly disseminates air quality information in real time

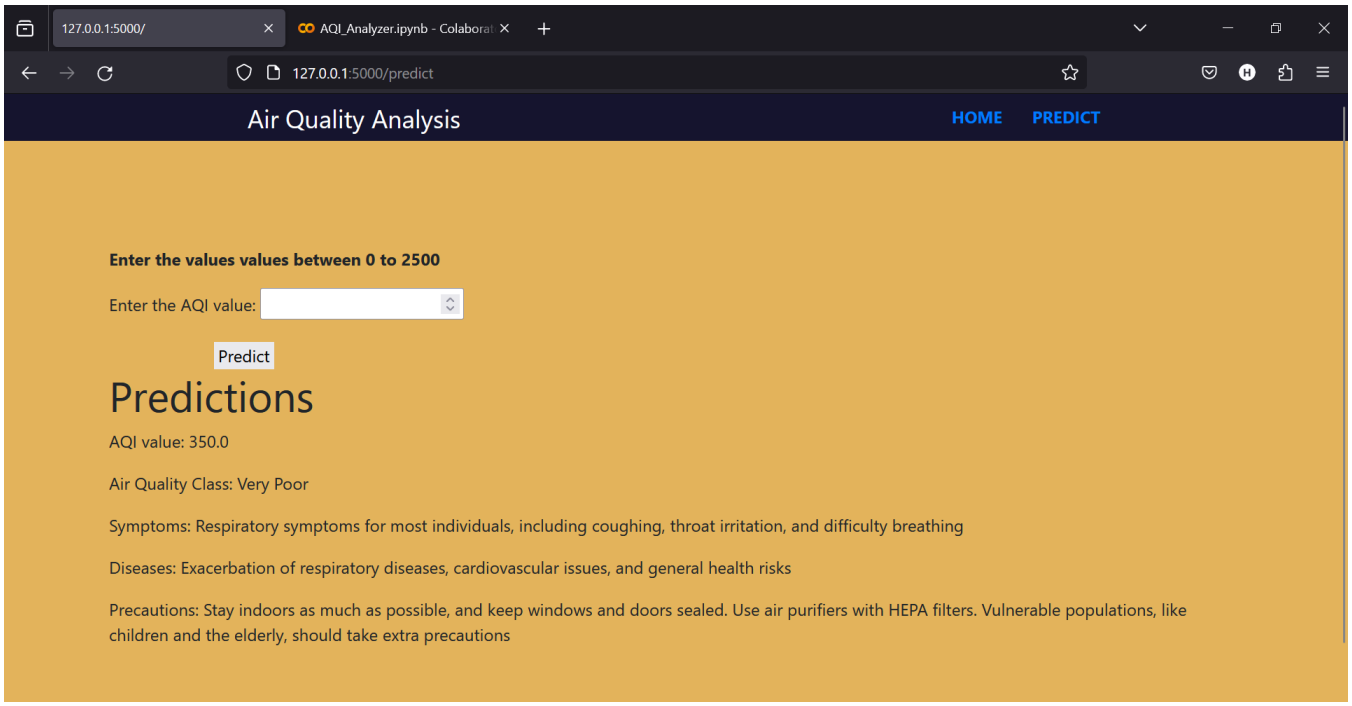
## PREDICTIONS

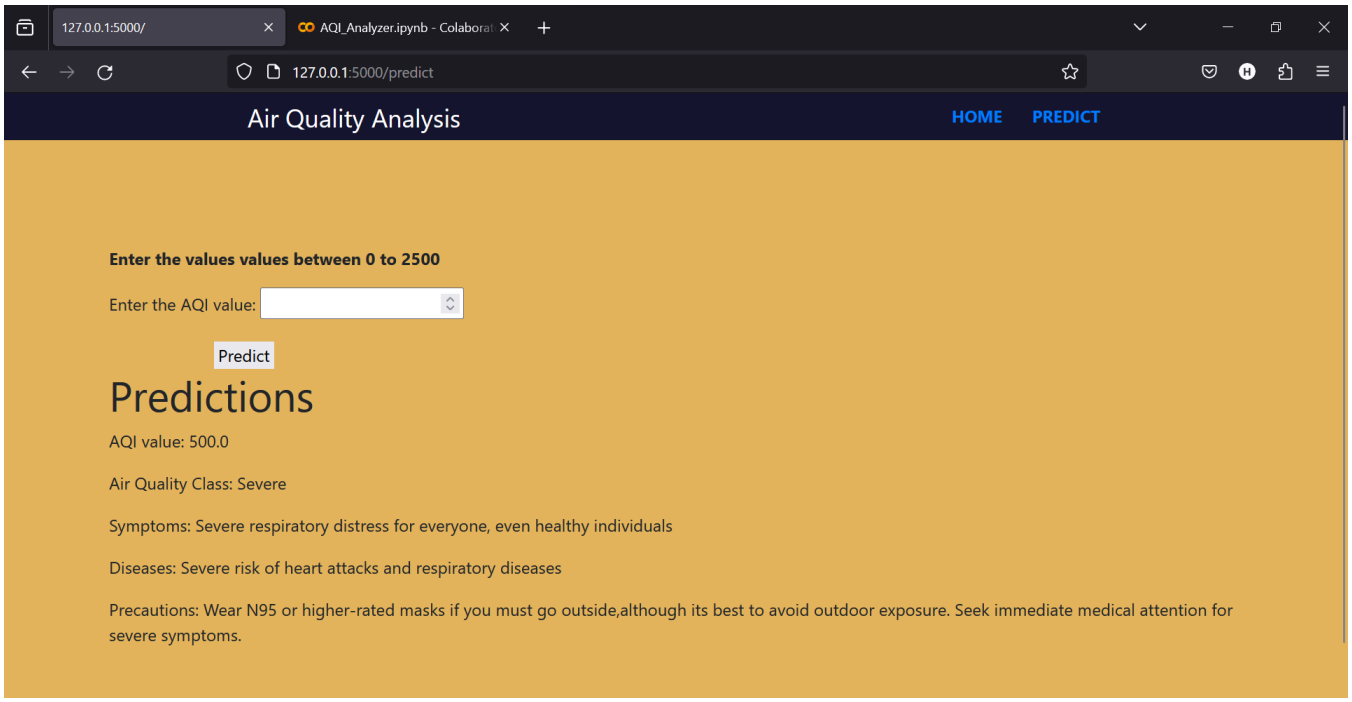



127.0.0.1:5000/
AQI_Analyzer.ipynb - Colaborat
127.0.0.1:5000/predict
Air Quality Analysis
HOME
PREDICT
Enter the values values between 0 to 2500
Enter the AQI value:
Predict
Predictions
AQI value: 100.0
Air Quality Class: Satisfactory
Symptoms: Experience mild symptoms like coughing or throat irritation
Diseases: Mild symptoms of allergies and sinusitis. Asthma symptoms can be aggravated
Precautions: Sensitive individuals should reduce outdoor activities during periods of elevated AQI


127.0.0.1:5000/
AQI_Analyzer.ipynb - Colaborat
127.0.0.1:5000/predict
Air Quality Analysis
HOME
PREDICT
Enter the values values between 0 to 2500
Enter the AQI value:
150
Predict
Predictions
AQI value: 100.0
Air Quality Class: Satisfactory
Symptoms: Experience mild symptoms like coughing or throat irritation
Diseases: Mild symptoms of allergies and sinusitis. Asthma symptoms can be aggravated
Precautions: Sensitive individuals should reduce outdoor activities during periods of elevated AQI


127.0.0.1:5000/
127.0.0.1:5000/predict
Air Quality Analysis
HOME
PREDICT
Enter the values values between 0 to 2500
Enter the AQI value:
Predict
Predictions
AQI value: 250.0
Air Quality Class: Poor
Symptoms: Respiratory conditions, including shortness of breath, coughing, and chest tightness
Diseases: Exacerbation of asthma, Chronic Obstructive Pulmonary, and increased cardiovascular diseases
Precautions: Minimize outdoor activities, especially for children, the elderly, and individuals with pre-existing conditions. Use N95 or equivalent masks if outdoor exposure is unavoidable. Create a clean indoor environment with air purifiers and keep windows closed


127.0.0.1:5000/
AQI_Analyzer.ipynb - Colabora
127.0.0.1:5000/predict
Air Quality Analysis
HOME
PREDICT
Enter the values values between 0 to 2500
Enter the AQI value:
Predict
Predictions
AQI value: 350.0
Air Quality Class: Very Poor
Symptoms: Respiratory symptoms for most individuals, including coughing, throat irritation, and difficulty breathing
Diseases: Exacerbation of respiratory diseases, cardiovascular issues, and general health risks
Precautions: Stay indoors as much as possible, and keep windows and doors sealed. Use air purifiers with HEPA filters. Vulnerable populations, like children and the elderly, should take extra precautions


127.0.0.1:5000/
AQI_Analyzer.ipynb - Colaborat
127.0.0.1:5000/predict
Air Quality Analysis
HOME
PREDICT
Enter the values values between 0 to 2500
Enter the AQI value:
Predict
Predictions
AQI value: 500.0
Air Quality Class: Severe
Symptoms: Severe respiratory distress for everyone, even healthy individuals
Diseases: Severe risk of heart attacks and respiratory diseases
Precautions: Wear N95 or higher-rated masks if you must go outside,although its best to avoid outdoor exposure. Seek immediate medical attention for severe symptoms.

# 7.ADVANTAGES AND DISADVANTAGES

## ADVANTAGES:

1. **Enhanced Public Health**: Provides timely air quality information, promoting health protection.
2. **Proactive Decision-Making**: Enables informed decisions regarding outdoor activities.
3. **Environmental Awareness**: Increases awareness of air pollution and its impacts.
4. **Community Engagement**: Engages communities in air quality management and feedback.
5. **Customized Health Information**: Tailors health recommendations to AQI categories.

## DISADVANTAGES:

1. **Data Reliance**: Relies on accurate and up-to-date air quality data for precise predictions.
2. **Resource-Intensive**: Requires substantial resources for data collection, modeling, and system maintenance.
3. **Technical Expertise**: Users may need some technical knowledge to interpret AQI and health information.
4. **Model Accuracy**: The system's accuracy depends on the quality of the predictive model.
5. **Privacy Concerns**: User data may raise privacy concerns and necessitate secure data handling.

# 8.APPLICATIONS

1. **Public Health Protection**: Empowering individuals to make informed decisions regarding outdoor activities, reducing exposure to poor air quality, and minimizing health risks.
2. **Environmental Monitoring**: Assessing the impact of air pollution on the environment, ecosystems, and natural habitats, aiding in conservation efforts.
3. **Government Policy**: Assisting governments and regulatory bodies in setting air quality standards, formulating pollution control policies, and conducting effective urban planning
4. **Public Awareness**: Raising public awareness about the importance of air quality and its impact on health, influencing behavior and lifestyle choices.

# 9.CONCLUSION

- In conclusion, the proposed Air Quality Index (AQI) prediction and management system presents a holistic solution to address the critical issues of air quality monitoring, public health protection, and community engagement. By integrating advanced predictive modeling with user-friendly interfaces, the system empowers individuals to make informed decisions, safeguard their health, and actively participate in air quality management. The project's key components, including data preprocessing, feature selection, model training, and user interface development, create a comprehensive framework for delivering real-time AQI predictions and associated health information.

- In the ever-evolving field of environmental science and technology, this project represents a significant step forward in ensuring cleaner air and healthier living environments. With ongoing research, innovation, and community involvement, the system has the potential to make a positive impact on public health, environmental conservation, and global collaboration in mitigating air pollution issues.

# 10.FUTURE SCOPE

Future Scope of the AQI Prediction and Management System:

1. **Global Expansion**: Extend the system's reach to more regions and countries, addressing air quality issues on a global scale.
2. **Advanced Technology Integration**: Integrate IoT sensor networks and smart city initiatives for real-time air quality monitoring and urban planning.
3. **Air Quality Forecasting**: Enhance the system's capabilities for short- and long-term air quality forecasting.
4. **Healthcare Integration**: Collaborate with healthcare providers to incorporate AQI information into patient care, particularly for those with respiratory conditions, improving public health outcomes.

# 11.BIBILOGRAPHY

[1] Anderson H. R., R.W. Atkinson, J. L. Peacock, M. J. Sweeting and L. Marston. Ambient Particulate matter and health effect;Publication bias in studies of short-term association. Epidemiol 16; 2005: 155-163.

[2] Analitis A.,K. Katsouyanni, E. Dimakopoulou, A. K. Samoli,Y.Nikolouopoulos, G. Petasakis, J. Touloumi, H. Schwartz, H. R.Anderson, K. Cambra, F. Forastiere, D. Zmirou, J. M. Vonk,L.clancy, B. Kriz, J. Bobvos and J. Pekkanen. Short-term effects of ambient paricles on cardiovascular and resipiratory mortilaity.

Epidemiol 17; 2006: 230-233.

[3] Kumar A.,Goyal P. Forecasting of air quality in delhi using principal componemt regression technique. Atmospheric Pollution Research. 2 . 2011: 436-444.

[4] Central Pollution Control Board (CPCB).Guidelines for National ambient air quality monitoring, Series:NAAQM/25/2003-04.Parivesh Bhavan,Delhi; 2009:

[5] Central pollution control board (CPCB). National air quality index , Series CUPS/82/2014-15; 2014:

[6] Ekpenyong E. C.,Eltebong E. O., Akpan E. E., Samson T. K.,Danierl E. N. Urban city transportation mode and respiratory health effect of an air pollution: a cross sectional study among transit and non transit worker in Nigeria. BMJ

open,doi:10.1136/bmjopen-2012-001253; 2012:

[7] Kaushik. C. P., Ravindra K., Yadav K., Mehta S. and Haritash A.K.. Assessment of ambient air quality in urban centres of haryana(india) in relation to different anthropogenic activities and health risks. Environment Monitoring and Assement. 122; 2006:27-40.

[8] Pipalatkar. P. P. , Gajghate. D.G and Khaparde V.V. Source identification of different size fraction of PM10 Using Factor analysis at residential cum commercial Area of Nagpur city. Bull.Envionment Contam Toxicol. 88;2012: 260-264.

[9] Pope C. A. III and D. W.Dockery Health effects of fine particulate air pollution; lines that connect. Journal of Air and Waste Management Association. 56; 2006: 709-742.

[10] Bhuyan P. K.,Samantray P., Rout S. P. Ambient air quality status in choudwar area of cuttack district.International Journal of Environmental Sciences. 1; 2010:

[11] Ravikumar,P., Prakash, K.L. and Somashekar,R.K.. Air quality Indices to understand the ambient air quality in the vicity of dam site of different irrigation projects in karanataka state, india.International

journal of science and nature. 5; 2014 : 531-541.
[12] U. S. Environmental Protection Agency (USEPA). Guidelines for reporting of daily air quality- air quality index (AQI), Series EPA-454/B-06-001. Research Trangle Park , North carolina.2006

# 12.APPENDIX

**Model building :**

1)Dataset
2)Google colab and VS code Application Building
   1. HTML file (Index file, Predict file )
   1. CSS file
   2. Models in pickle format

**SOURCE CODE:**

**INDEX.HTML**

```
<!DOCTYPE html>
<html>

<head>
   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
integrity="sha384-
JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
crossorigin="anonymous" />
   <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>
   <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
integrity="sha384-
9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"
crossorigin="anonymous"></script>
   <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"
integrity="sha384-B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV"
crossorigin="anonymous"></script>
   <script src="https://kit.fontawesome.com/961c028791.js" crossorigin="anonymous"></script>
   <link rel="stylesheet" href="/static/style.css">
</head>

<body>
   <div id="sectionone">
      <div class="main_1">
         <div>
            <h4 class="m-2 head1">Air Quality Analysis</h4>
         </div>
         <div class="m-2">
            <button class="style_but2 mr-3"><a href="index.html">HOME</a></button>
```

```
            <button class="style_but2"><a href="{{url_for('predict')}}">PREDICT</a></button>
        </div>
    </div>
    <div class="main_2"></div>

    <div class="main_3">
        <div>
            <img class="mr-5"
src="
BgYGBIYGBIYGBgYGBgYGRgZGhgYGBgcIS4lHB4rHxgYJjgmKy8xNTU1GiQ7QDs0Py40NTEB
DAwMEA8QHhISHzQrJSs0NDQ0NDQ0NDQ0NDQ0NDQ0NDY0NDQ0NDQ0NDQ0NDQ0NDQ0N
DQ0NDQ0NDQ0NDQ0NDQ0NP/AABEIALcBEwMBIgACEQEDEQH/xAAbAAABBQEBAAAAA
AAAAAAAAAAAAQIDBAUGB//EADoQAAEDAgQDBQcDAwMFAAAAAAEAAhEDIQQSMUEF
E1EGImFxgRSRobHB0fAyQlJikvEjgrIVM1PC0v/EABoBAQEAAwEBAAAAAAAAAAAAAAABA
gMEBQb/xAAlEQACAgICAgEFAQEAAAAAAAAAAQIRAxIhMQQTQRQiUWGRgXH/2gAMAw
EAAhEDEQA/AOvo01YDE9jE8NXFZ20RhiUMUgalDVbJRHlS5VIGpwaliiINTg1SQlASxQwNS5E8
BLCWKGZUoYnwiEsUNyoyqQBEJYoZlRlUkIhLFEeVI5qkR4pYopvw5J9dVG+hlILnen3V6m9rhLS
COoMqq6nmeXHTYeHgoVCGo4/pAtqT9lSrNObMDMWvp6K1iaZDMrd1Bg+GkiahJvIHS+/VLLRe
otMSfz1UxRltAVeoXbWKWSrGV8aynIeSIEz9Fx+M4malQOZmgOgAG5HgF1OI4cajSD+7UqnheC
AHTTTYKqSLRa4fxFr2gHukQO9Ek+S0adxdZI4N/qB3jJ6T5LaYyBClhpDYRCfCSEsxoYWoyp8JEs
UNhNcxSwmwliitkSqeEKWKIwEoCUBKFhZtoAEsJUBWyUJCITkJYoSEsJUsJYoSE4BASgK2ShIS
gJ0LI4nx+lQfyyZfbuzAA8SquSUawCWFFhsS2o0PabWUspYoCoX1gN1X4nXAYQHQ6P06kz4Kpgn
BsuqOs1sydIjXzUsqj8lzF8SpUh33AH+Op9wXEcV7Tl7y0yGDMAxpjNsMxWH2s4o2pWLmvzXmBo
NgB1sAucq4u9xI6rphitWzVKai6R1XDO0z6RIBM/tEkgeY3C6fs92lNSpy3kuzR3ogNK8qwteHLpOz+
NyVGySyHglwEmPqssmNU2hCbbpnr72AhQ1KkQ0g36KxSeHNDhoQCPIhGQTO/Vcdm1EIqNixv03
VXAFziSSYk2Kt16GYRJHiI+qr4OkKbsgJIMnMfkpZkqovQkywmvqBokn03XP8R7QNEta1wcNCBt9
lldkSOjCQrmOE9ps4LKjC03h/7TGxOxWXhuOvxFZwaSGgyY0I2Hl4q0xqdzKCVzGHxr3OzyWgdT
MgbwtZ3EWwL7CTFpWFl1NCUBZzK+U5psfHRWGV9Smw1LSIVf2geCma+QpsTUWEiMyFbJQw
JyYClla9jbQ9CbKAVdiUPQmynSlkFSpspZVsg5KmyllWyEHEMW2jTdUdo0eXgF5RxriQr4nOxoE6
kEkHbden8aZmoPb1abdfBeXPeylScZlx0/Nlvwtf6YyXBYxPH302ctsicuYtOUmPJXcH2veQ1pOgDcx
knzK4vFYx1Qjw0VrA1HNcHEWm8roeKNco1LK9uD0jDYx7253jvnQxqNpUHGMUTQqAgl2WIAP
r5CE/heKY1rczmh0HujoU/iDmNdzGuOctIDTGUW1jfey47qR11weTYlpJmFWldDiAA5096Zg9SVgv
ZDoXoRlZ5840zT4Pw41XgNuZAjReoYDs23K0kQ4ESevULkewtMOrgZQWwZna1l6pSEW+K4883t
qdWNJRslDYEBInSmkrRZkiMU4MlxPQKlWxoY8zAA1cTYK3VqARJAm1z8FicTc0SC7WZiJjp8Vj
ZnFX2ZvGO0VKCWGT/ID6lYY44x9RrRTIm2YEEAdSsbitZhqObYMB6XPgqWDLg4uGsGF1xxL
WzU8j2pHe4fhZqBxa8gW7rfjfoVcw3By1znthsgAiLk7ql2Rc9jOZU/SRDb+pPwXS4PGMqAgGDJsdf
Nc0m02rN6XF0ZHMyvywbD9EfVSuwzzDnDYSDEhbVPDNBnUp5pgrW2W0Yow7wNfFObTeRLpg
bG3wWxygoKjgLOBujZU7KDMQ4AtLTeO8PutGg9IKQUjWAKWR0PshIhWyUNBSgqIOTg5a9jKi
WUoKiDkuZWyUSyiVFmS5ksmpLmRKizJcybDUlBSyow5LKuxjqRcRaXUnAax/leNdpXOfWJDS0C
Gxt3QB9F7PVbmaWncEe8Lx7tERTquptdIYcs21GvxXX4r+415V9phNIGqf7RJuYVZ7pNlYZgHObI9
PHwXoOvk41b6Oh4C8Oq05dJkgzcAbLtOHUnOLnPaIBu7a9hA2XnfC8O7MIBDwRG0Gd163hMSyn
RbmBFhmi4mBJXB5PD4O7C3r0cB2k4E+gDUa6WA9Iieq4t7szpXtuLxtCpRc6p/2zIJcLWPT3Lx7iD
KfNfyycmY5ZsYmy3ePkbTTNOeCVNG32Urmm+YJ/pB1XrOAqZqbT4Cx1HmvG+BYkU6jHOBIm
4GsaW8V7AyuMrTpIETqZHzXP5XErNuL7o0Xc0KF+IaJJMAalRVqwbBPoOpOgUGMrtgtNzF41aO
vguXY2KJh9paTnVGVWuzRAZTmB/USPquc7QMq0wKpfqIyg6eEK9ieO06b/wCVrfyHnsuP4pxA1n
EybmYm3ouzDCVq+jXlnGKpdmXVqFzpJmVucA7z8sSXCB0usvD4Uv0XbcF4YaJDy3NOXzW7PN
```

RjRrw45SlZcZQq0GtDyMmYAba/SU7CBzKrTNgdAZ1vvqtbGUBWZliAIIG/gqvDuA8sh9R5cZm1o6
Lz901z2d2tM6ai4kTOvwUkprBGmicsLNT7BDhKWUEpYGNCVBKQuUsoITZQliiuHpQ9Vw9LnWn
Y6NSyHpc6rcxLzE2MdCxmRmVfmIzq7DQs5kocq3MS502JqWsyXOqoejOmxNC3mXmHa7s3V59Sr
SYSx7swaLuktBcQBtmlejCoud7ccTdQwhdTeWPc9rQ4axcuAO1gujx8jU0l88GnLBatv4PIoIN1vYGv/
ougiWw4T7isF75Kka8gWK9lxtHnRlqzquDcVYagD294kAkb7A+a1u0fFn03NpscWw3vDUyfHyj3rjO
FUXOdnB/TefJXuKtqVHB8OJgCYuYETHkFzyxx3TOqOSXrZDxHiVSoAxz3FrZhs2E6rOpXcJ0lWa
mH/wBMme8095p1g6QqTXwuhVXBzSu7Z33Z5uGbBcLiS152Mbq3jOKEuBkkCXAX20Jhc9wIPqB2
RlgGknbXVbXFP9Oi0tbJIhxIjLEfOSuDIlvT5PQx367NLheOfWeH1HmzoaAIZp0/lrqtHi5JpPDWnM8
QTGo0geixOF47D0KHMc4vfY8sA2dlka20Gqw+LdtqrpFMBjSbCxI9QtSwylP7VwjKWWMY8sw8dR
IqFr3GyqA3tom18c6o7M7UoAtJXppNLk85tN8HQdnqZe8AiQJjwPVegNYRTES4wYnZcJ2Wx+R7Q
0XJAk318F6YxoOq8vym1I9Lx60s57hZqOq5nhwjQTaNyV0c5hASOb4eakbZcrlbNvwTMNgnSoc6TO
psYak+ZBeoM6TMmw0Ji5JmUOdGdTYuhNmSKHMhTYupnCol5ioiqjmrGjv9Zf5iOYqPMS81KHrL
3MRzFS5qXmKUT1l3mo5ip81BqpRPWXuYjmLPOJA1MeaVmIaRIII6gyrqyesv8xYPbem1+DdIu1z
HA9CLfIkeq0ecFV4jSbWpPpumCNjluNLrbies4t/DNeTC3Br9Hj7tVIx2yZWbDj5lNYbr6A+dOr7Nub
TzcxmZjgJdBtG4PqtXH9qqbC4YenBMd8xNug2XM4LiL2tNOZY4OaWnxESOhVWrSJ0Wj1JzbkdSz
NQSiNxOKc9xcTcmSoGCUgbeFNTGUybjcLd0c/b5Os7GYoND2lpMgSeo6LoeN8RFKg9z2AtMQ0a
knQHouM4ZxN7YZTa0S5p0vbYnouh4/wAbpFnJcHPLgMzW2g6xmI6rz8uOTyp12eniyRWFq+ji63E
XOBy90fxB+fVUKzyVYxuXOQwQNgq9R8wOi9BL8HmSbb5H4cCbrVaw1CMjTAEuJ3AVXhlMO
MEHbyXXYDhrjBDYm/SRI9y05sih2dODC59Gt2RwAZTzuaAZdFhIvI26LpuYFm4doptDW6BTcxeJ
km5ybPWji1SRc5qOaqedQYvEljZBE7D5LBW3Rl6zTNZN5qweH1aznZqhte3+Fqc1WS1dBY/0WuYj
mKrzEc1YWXQs8xHMVbmpOag0LPMQqvNQg0OdGMThi1hjEpfal3ek6FliboxaUYorDGKSjFJ6WZ
eyJuDFeKX2nxWF7Uh+MDRJKnoZHlglbZs4nGODZZHqm4birX2JAd0+y5qrxVx/Toqz8W11xZy3x8
W1TOGfnRUrjyjpOJ4uA6DMj3LPw3EsjcjDlJ2i09ZWdh2OP6zAJFydfALRp0GAyRJWfrjGNPk1qeT
NLaPH/TewTnZAXmT9FLiSCxw2Ivcj4rKGKWX2hxOakG54l0xfvQNLeYWiOJymvg7cs448TfdIwOI
0MlRzfy91TbEpzp3T6FPM4DqvWXCPmW7fBucEoZ3tGze9MaR80nH8cHVHBrQ2LEgC5FibdVPhn
NoNOV8yDa0G0BYNZp16rUo7Sv8Ah0TeuNRXbdsawSrdLTx3VAKegHGQFtZzou4EhtQHoZ6Kbid
XvFwtImPGyoYZ8PHmpeLY/muAYO60BoIHvusGvuRtTXrav5MstJkpZT2tIHnbxUgpQ+OkStlmmjZ
4FTaYmMxByg736rXxPEMRhzLrtdo7UNvoRssLHUwzKWEjQQNBHirPFOMZ2tZEgtacx38QuScH
KSdWmd+OcYQabprqvk7Lh3EObTD+uvmLKy7EQJlcnwnigFMA2i07H0VfjvEs7RTYTBuY+C434rc
6XCPS+pxxwqTduuv2WXcXrMqF5eSCZyAy2OgV2nxkEcxwu4wI1HvXNYCjuTHmtBrGWv5rpnih1
X8OLDmyd3/WdXhcV8dCrHtC5U8TbTFpNwIEbmEHjJ/j8VyS8aTdpHorysKX3Pk6n2hJ7SuVPHBIa
WmTJF+kTePEKti+IPc4OYS0jYOMeo0KkfEk3zwYy8zCla5OxOLA3S+1BcmOIyAHDMRv19FMeK
RqPij8SRlHy8L7Z0vtKFzH/VvD4oU+kn+C/V4PyYftQQ3FXN+iyPaPFHtAnU7L2PWjwPqJ/k2faR1S
+0jqsrOYmR5EgHSRY3iN0ja/ip60PqZ/k0cRjCLDVVxVe6xJPqq+cC5I8t0x+IBEA28FkopdGEskpctl+
m6Ac0CNpEpzMTTbvfyJ+QhY7HgzMalPcR1TVMxWRx6NPF4sPyZXfpe0zpF/krwxXiuecIaTbS1/j8
Cp6GKhoBOgG2wHSFi4I2R8iabd9mx7aC3XY++FRx+KDjBuRoRBsdlUfVDLlwNyfPNuqr67ZkT7v
eNVlGCTsk885KmyZzxufgp8M9kgTcxeDY3t+dVnvxA6fkBNGIIMwsmaE6OmEaeFtIWZlJJhV6fFn
AC2nloVDWxuaLaA6+JJ+qkVRnKSZZy/MLUwzg2mT5262WG3GCACNyZFz75U7cW2NdwRboN/
WEkrEXRcpPgkx+dE2o4CAB/jcquyuDAEg/qk2FpkpC4aZ97X23PnYe7ylRNi1hnAGY0c6D6W+SUh
sl0/uaI/pGpVUkGe9vqevQ+6fQoBE2PS0ztCUVSNPiFQZf8AcFC9rJYTs35ALOr13SZcYnTy8FG/FO
OUFxgTA6WU1oyc7fRssxQaCGxF4EJlI53ZnHRZoxQH+FYdjm5Y3IgiDbx0/JRxroKd9l7CvGQXi7v
+RUuYdVitxvdDQYidvFSU8aGkwTfwv7pUcTJZF0aGJggXOo0spszf5LDdinHVxR7S7+RTUnsX4Nd
5HMZf9tT/ANUNd3yZsBCyGYp2dpmbO19FZdij+R9k1Y3RffVhwDbmPqnNo/ucTv5e9ZT8RMajVI5
4O8Dx801G6/BtU6gj37eKFiNqx+FCmhfajMa+E4P8FEllbDnHh6dn8FFKA5ASOcUNcVHmSB5QEr
HG/mUuYqNlUjQnWdd08VnTOYz1zGfehQc4/NGaw8k0vMHy6pzq7iBLnGNJcTHlOiEEJSEpC8nf4
oJQCyhIHJcyAEJzapAjaZhNzhACEc07GybmQDy4pQUwVEgegJs5iPfrfzSipG1/kq+ZLm3QFitXzOk

NgW7oJO1zJJ1URqaW36+GmiYXph1QpfoVKUjO18TfK5sxB0lusx+XV6hw5lVhfSqS5rcz6b25SGjV
zTPfA3i/gsZr7RAUtHElkFu3nuIOhGyjT+GZRkl2iSrTLdRrBHiDoUyVf4hx99ZrWup0m5QACxhBgA
jLcm19BZZoxH9I8xKRuuRKr4JEspntA6D4/f8AITW1vD5/dUxHON2+vyT5UTqwzNMaT62UntDf4/
E/dAKHpwemGu3+PxKA8HQD1dHzQpJnCE00x1H97PukQclKUKc0htmPoEnIdsD7kMSFIpH0y3UE
eYhRFSwOSSkQlgcEQhqUhUCFKAlMdUmYIUIRCJRKAISQnfmyJ/LIBIRlWpQwlGAXVLkXEafF
OrYeg0WeCdpmPUhxhY7K65/hs9Tq7X9MmEZVbNBn/kYP7z8mFQVWhujmu8W5h/yaFka6I8qIUg
LY3H+0H4ymPI2J9wHyKlotPsSEQkBSFWzEVBSNSlSwKnW/CPsk9fn9kF35AVstCgjofePsnd3o73g
/RRg9fkg+fwSyErcm+b0y/ZDsm2b1DSoZU9Ll/vzf7Y+qjdGSVkBPQ/RNlW6vJju8yfHLHwVSFE2w
1QuY9VKyDq4DzB+gUbQN59IKnbyxqHH1aPhdLoJWJlH82+53/wAoSzS/i/8Aub9kK7foUMbiXAQ
HOA6BxSHEu/k7+4qJClC2Oc8nUk+d0xCFSCpEIQCyhIhACWUiEAqEiEAqJQkQCoQhAEolCEAIl
CRAKhCEAJEqEAiEqEAIQhAIlQhAIlQhACRKhAIhKhAIhCFACEIQAhCEAIQhUAhCFACEIVAI
QhQAhCFQCEIUAIQhUAhCEAIQhQAhCFQCEIUAIQhACEIQAhCFQCEIUB//2Q==">

<h1>What is AQI ?</h1>
<P>The air quality index (AQI) is an index for reporting air quality on a daily basis. It is a measure of how air pollution affects one's health within a short time period. The purpose of the AQI is to help people know how the local air quality impacts their health
The higher the AQI value, the greater the level of air pollution and the greater the health concerns. The concept of AQI has been widely used in many developed countries for over the last three decades. AQI quickly disseminates air quality information in real time
</P>

<script type="text/javascript" src="https://d1tgh8fmlzexmh.cloudfront.net/ccbp-static-website/js/ccbp-ui-kit.js"></script>
</body>

</html>

## PREDICT.HTML

<!DOCTYPE html>
<html lang="en">
<head>
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z" crossorigin="anonymous" />
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js" integrity="sha384-9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"

```
crossorigin="anonymous"></script>
   <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"
integrity="sha384-B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV"
crossorigin="anonymous"></script>
   <script src="https://kit.fontawesome.com/961c028791.js" crossorigin="anonymous"></script>
   <link rel="stylesheet" href="/static/style.css">
</head>
<body>
   <div class="second_main">
      <div class="main_1">
         <div>
            <h4 class="m-2 head1">Air Quality Analysis</h4>
         </div>
         <div class="m-2">
            <button class="style_but2 mr-3"><a href="{{url_for('home')}}">HOME</a></button>
            <button class="style_but2"><a href="{{url_for('predict')}}">PREDICT</a></button>
         </div>
      </div>
      <div class="next_main">
         <p class="para">Enter the values values between 0 to 2500</p>
         <form method="POST" action="{{ url_for('predict') }}">
            <label>Enter the AQI value:</label>
            <input type="number" name="aqiInput" required>
            <br>
            <button class="style_but mt-3" type="submit">Predict</button>
         </form>

         <!-- Display predictions here -->
<div class="predictions">
   <h1>Predictions</h1>

   <p>AQI value: {{ model1_info['Prediction'] }}</p>
   <p>Air Quality Class: {{ model1_info['Air Quality Class'] }}</p>
   <p>Symptoms: {{ model1_info['Symptoms'] }}</p>
   <p>Diseases: {{ model1_info['Diseases'] }}</p>
   <p>Precautions: {{ model1_info['Precautions'] }}</p>
</div>

      </div>
   </div>
</body>
</html>
```

## APP.PY

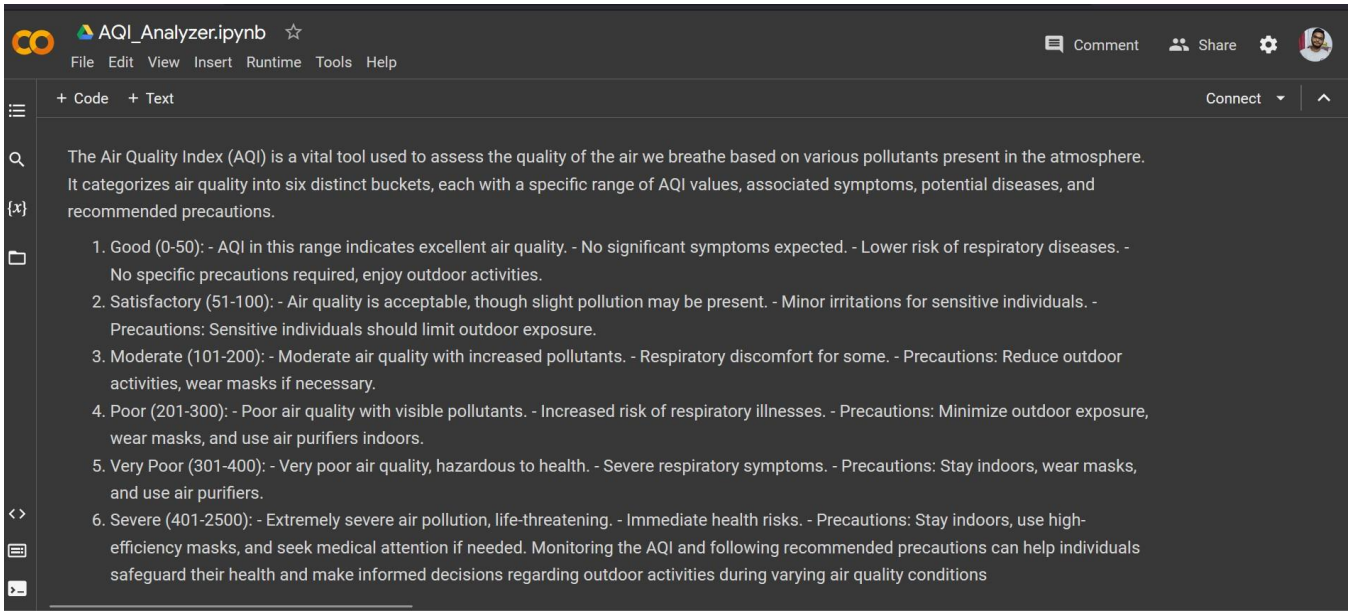
```
import numpy as np
```

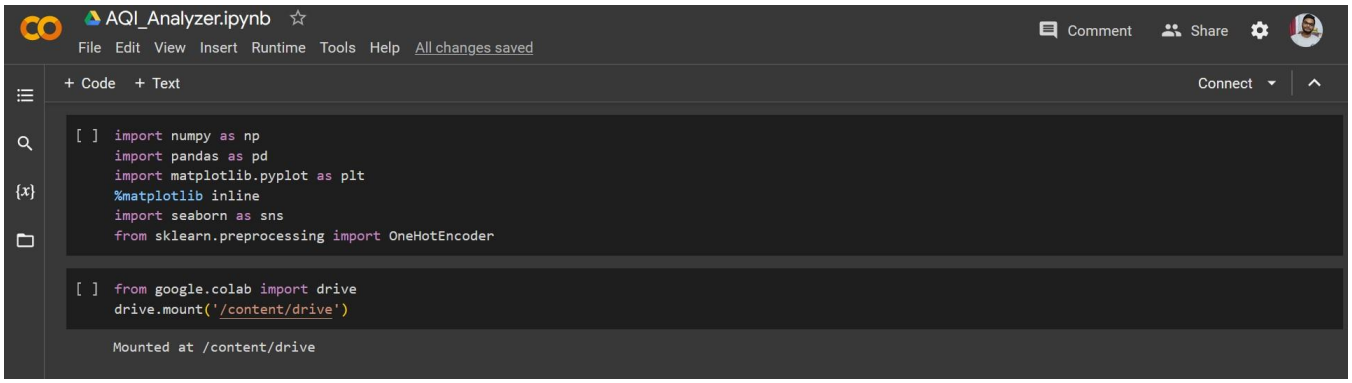
```
import pandas as pd
from  joblib import *
import requests
from flask import Flask, render_template,request
from flask_ngrok import run_with_ngrok

app = Flask(__name__)

def load_models():
   model1 = load("RFS.joblib")   #random forest model
   model2=load("RFS1.joblib")    #random forest model
   return model1, model2

model1, model2 = load_models()

# Define air quality intervals and labels
intervals = [0, 50, 100, 200, 300, 400, 2500]
labels = ['Good', 'Satisfactory', 'Moderate', 'Poor', 'Very Poor', 'Severe']

# Define symptoms, diseases, and precautions
symptoms = ['Air quality is good in this range, and most people will not experience any symptoms',
        'Experience mild symptoms like coughing or throat irritation',
        'Coughing, Shortness of breath, or Chest discomfort',
        'Respiratory conditions, including shortness of breath, coughing, and chest tightness',
        'Respiratory symptoms for most individuals, including coughing, throat irritation, and difficulty breathing',
        'Severe respiratory distress for everyone, even healthy individuals']

diseases = ['None',
        'Mild symptoms of allergies and sinusitis. Asthma symptoms can be aggravated',
        'Respiratory Infections',
        'Exacerbation of asthma, Chronic Obstructive Pulmonary, and increased cardiovascular diseases',
        'Exacerbation of respiratory diseases, cardiovascular issues, and general health risks',
        'Severe risk of heart attacks and respiratory diseases']

precautions = ['Enjoy outdoor activities and open-air exercise',
         'Sensitive individuals should reduce outdoor activities during periods of elevated AQI',
         'People with asthma and heart conditions should limit outdoor activities. Consider using air purifiers
indoors',
         'Minimize outdoor activities, especially for children, the elderly, and individuals with pre-existing
conditions. Use N95 or equivalent masks if outdoor exposure is unavoidable. Create a clean indoor environment
with air purifiers and keep windows closed',
         'Stay indoors as much as possible, and keep windows and doors sealed. Use air purifiers with HEPA
filters. Vulnerable populations, like children and the elderly, should take extra precautions',
         'Wear N95 or higher-rated masks if you must go outside,although its best to avoid outdoor exposure.
Seek immediate medical attention for severe symptoms.']

@app.route("/")
def home():
   return render_template('index.html')

@app.route("/predict", methods=['GET', 'POST'])
```

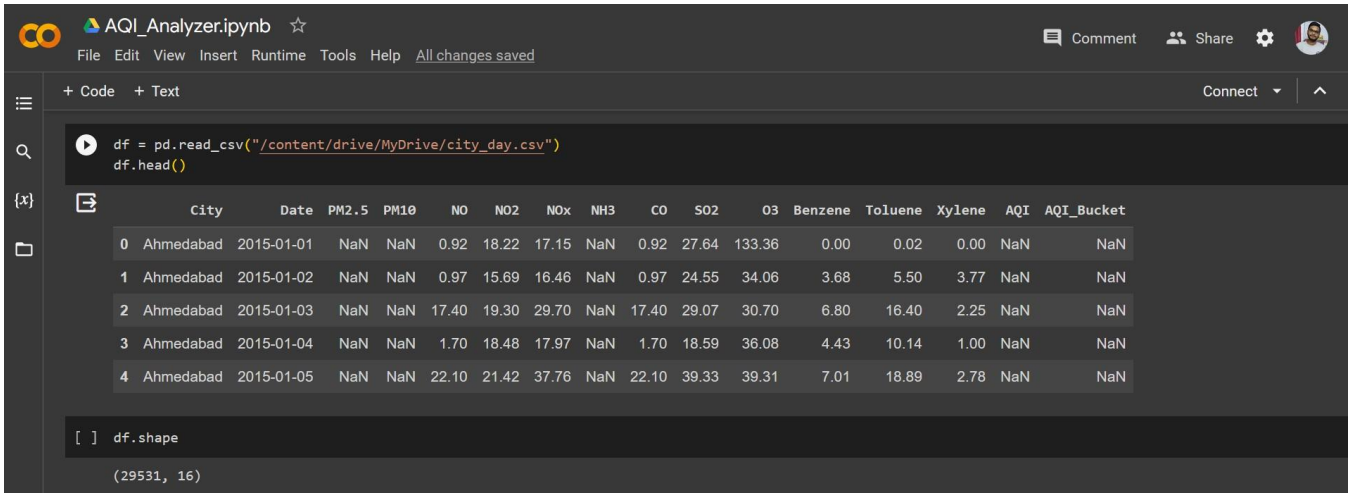
```
def predict():
    model1_info = None
    model2_info = None

    if request.method == 'POST':
        try:
            input_data = float(request.form['aqiInput'])

            # Check if the input value is within the expected range
            if 0 <= input_data <= 2500:
                # Make predictions using both models
                prediction1 = [input_data]  # Wrap the scalar value in a list

              def get_info(prediction, model_name):
                    air_quality = pd.cut(prediction, bins=intervals, labels=labels)
                    symptoms_info = pd.cut(prediction, bins=intervals, labels=symptoms)
                    disease_info = pd.cut(prediction, bins=intervals, labels=diseases)
                    precaution_info = pd.cut(prediction, bins=intervals, labels=precautions)
                    return {
                        'Model Name': model_name,
                        'Prediction': prediction[0],
                        'Air Quality Class': air_quality[0],
                        'Symptoms': symptoms_info[0],
                        'Diseases': disease_info[0],
                        'Precautions': precaution_info[0]
                    }

                model1_info = get_info(prediction1, 'Model 1')

                print("\nModel 1 Prediction:")
                print(model1_info)

            else:
                print("AQI value is out of range (0-2500).")
        except Exception as e:
            return "An error occurred: " + str(e)

    # Handle the GET request to display the form
    return render_template('predict.html', model1_info=model1_info)


if __name__ == "__main__":
    app.debug = True  # Enable debug mode
    app.run()
```

# CODE SNIPPETS

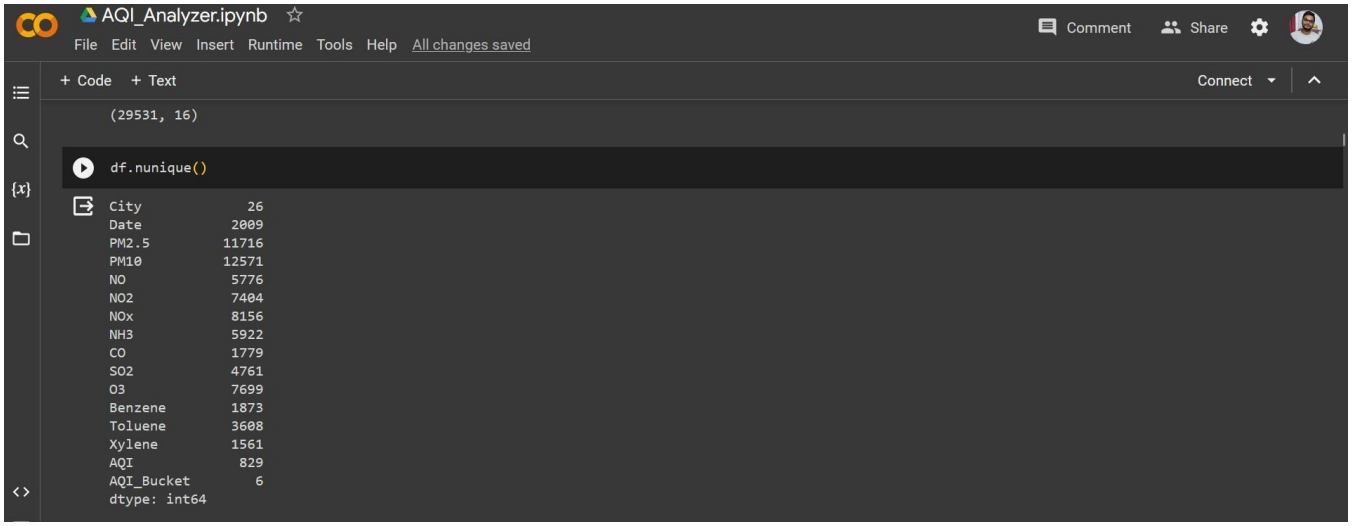
## MODEL BUILDING

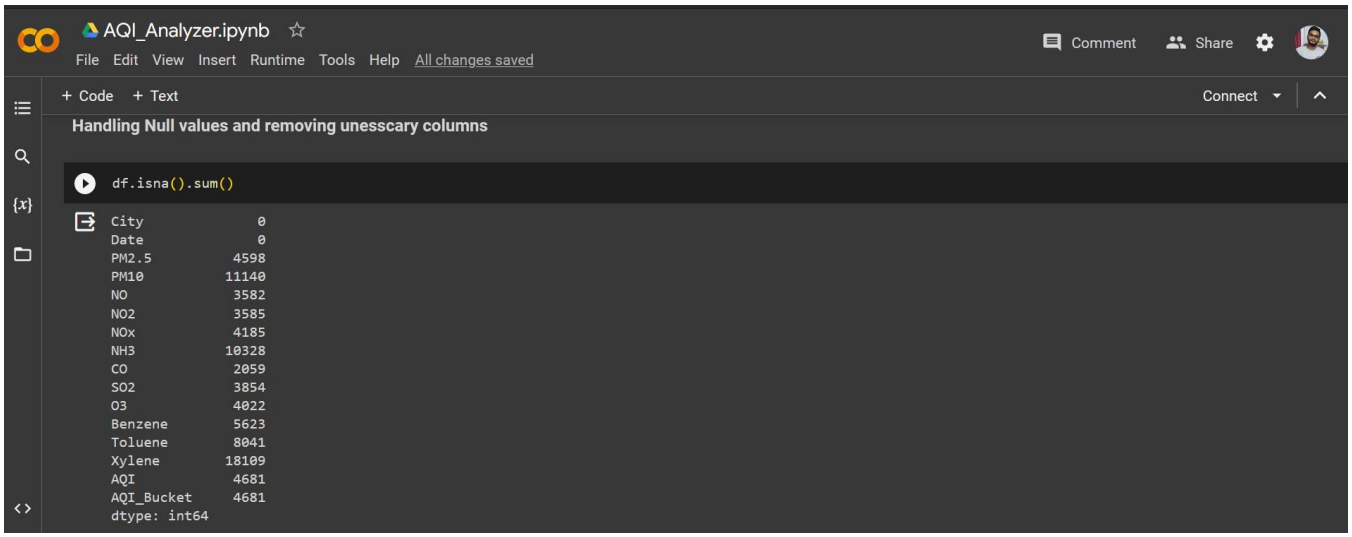


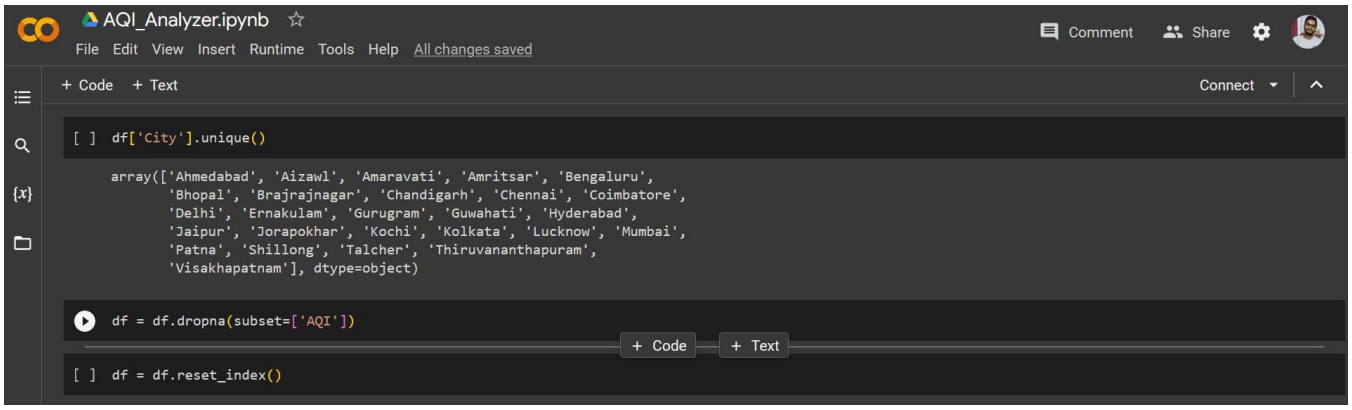



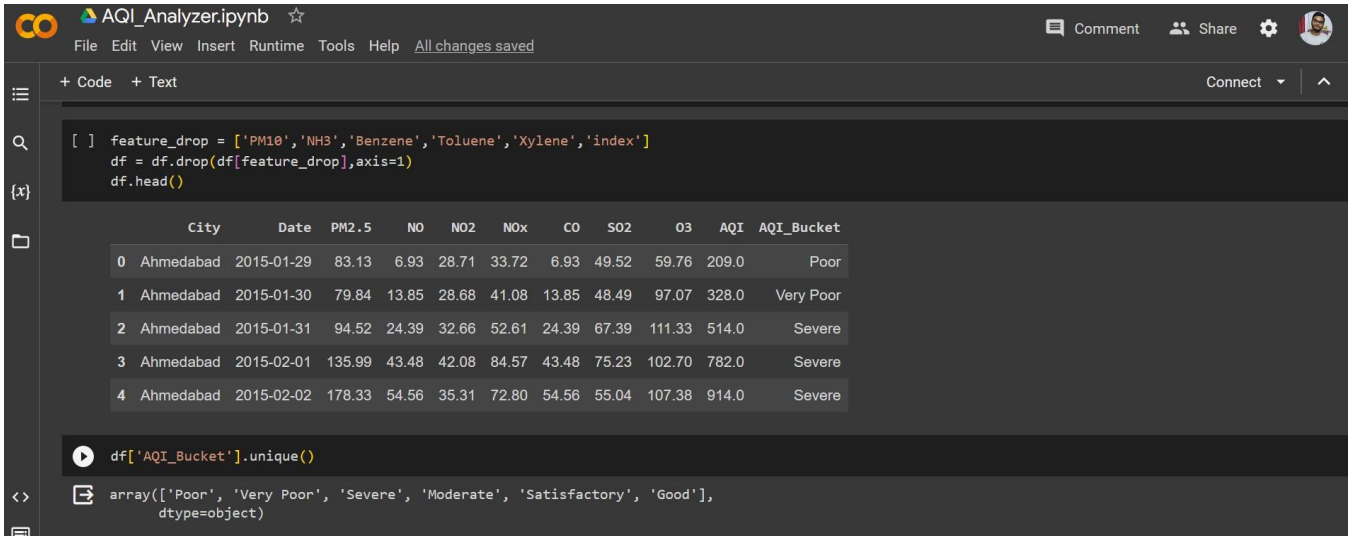




| | City | Date | PM2.5 | PM10 | NO | NO2 | NOx | NH3 | CO | SO2 | O3 | Benzene | Toluene | Xylene | AQI | AQI_Bucket |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Ahmedabad | 2015-01-01 | NaN | NaN | 0.92 | 18.22 | 17.15 | NaN | 0.92 | 27.64 | 133.36 | 0.00 | 0.02 | 0.00 | NaN | NaN |
| 1 | Ahmedabad | 2015-01-02 | NaN | NaN | 0.97 | 15.69 | 16.46 | NaN | 0.97 | 24.55 | 34.06 | 3.68 | 5.50 | 3.77 | NaN | NaN |
| 2 | Ahmedabad | 2015-01-03 | NaN | NaN | 17.40 | 19.30 | 29.70 | NaN | 17.40 | 29.07 | 30.70 | 6.80 | 16.40 | 2.25 | NaN | NaN |
| 3 | Ahmedabad | 2015-01-04 | NaN | NaN | 1.70 | 18.48 | 17.97 | NaN | 1.70 | 18.59 | 36.08 | 4.43 | 10.14 | 1.00 | NaN | NaN |
| 4 | Ahmedabad | 2015-01-05 | NaN | NaN | 22.10 | 21.42 | 37.76 | NaN | 22.10 | 39.33 | 39.31 | 7.01 | 18.89 | 2.78 | NaN | NaN |

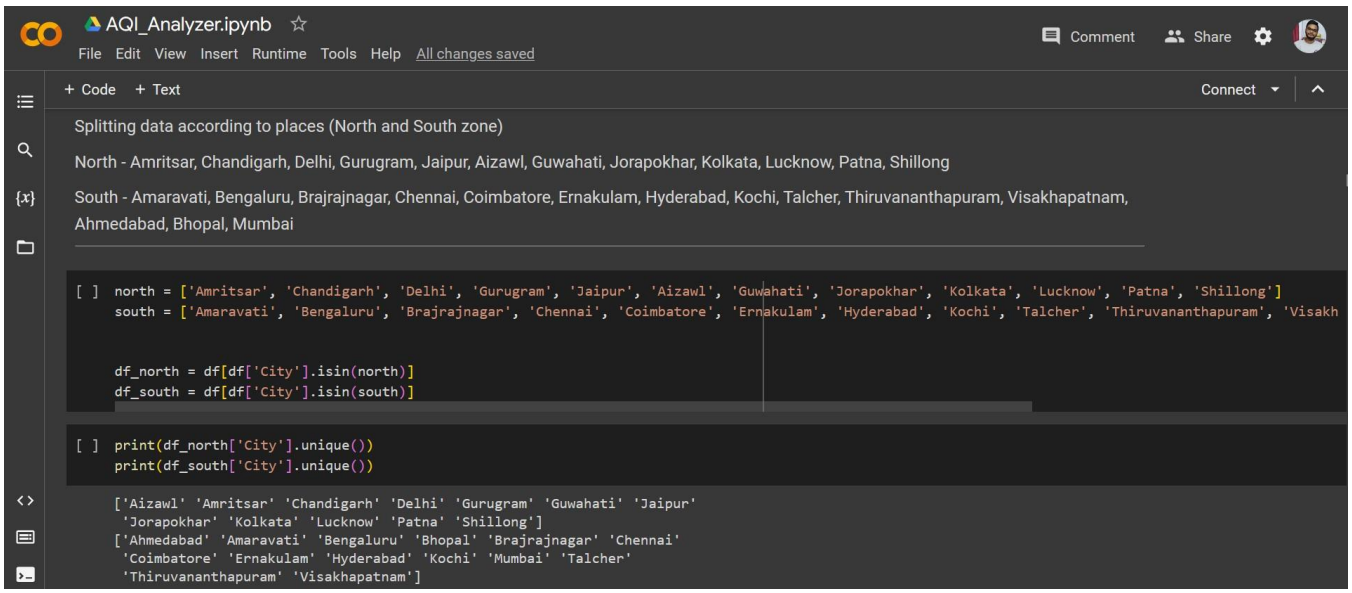

AQI_Analyzer.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Connect
+ Code + Text
(29531, 16)
df.nunique()
City 26
Date 2009
PM2.5 11716
PM10 12571
NO 5776
NO2 7404
NOx 8156
NH3 5922
CO 1779
SO2 4761
O3 7699
Benzene 1873
Toluene 3608
Xylene 1561
AQI 829
AQI_Bucket 6
dtype: int64

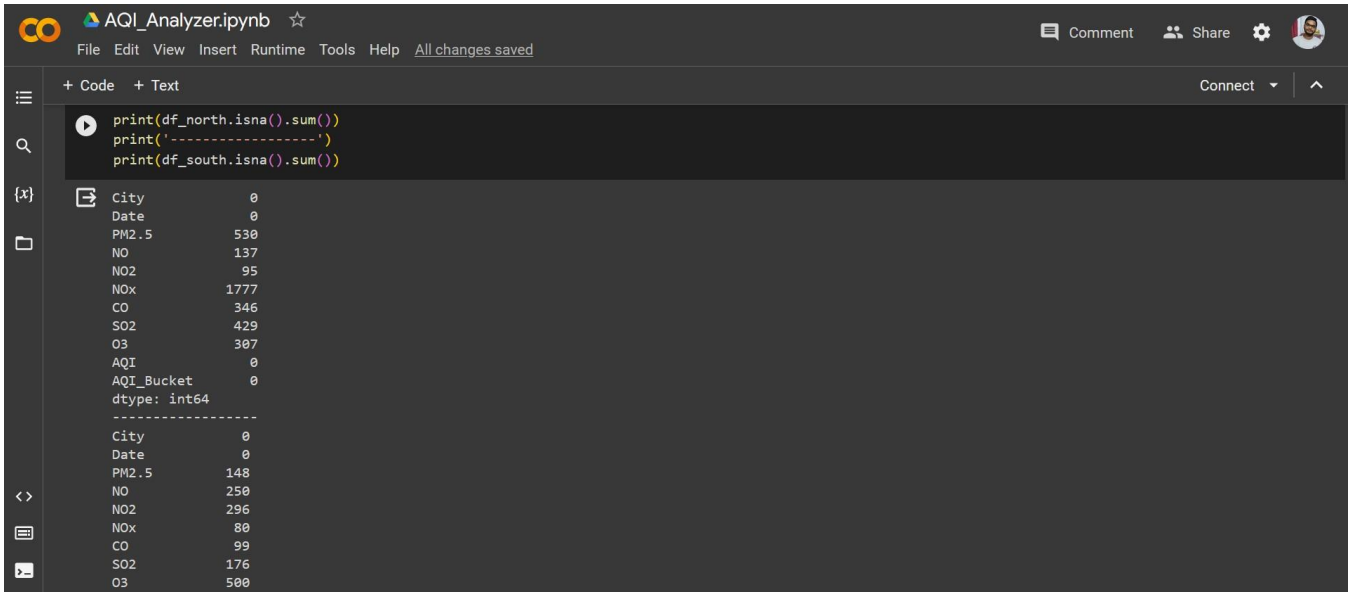

AQI_Analyzer.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Connect
+ Code + Text
Handling Null values and removing unesscary columns
df.isna().sum()
City 0
Date 0
PM2.5 4598
PM10 11140
NO 3582
NO2 3585
NOx 4185
NH3 10328
CO 2059
SO2 3854
O3 4022
Benzene 5623
Toluene 8041
Xylene 18109
AQI 4681
AQI_Bucket 4681
dtype: int64


AQI_Analyzer.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Connect
+ Code + Text
[ ] df['City'].unique()
array(['Ahmedabad', 'Aizawl', 'Amaravati', 'Amritsar', 'Bengaluru',
'Bhopal', 'Brajrajnagar', 'Chandigarh', 'Chennai', 'Coimbatore',
'Delhi', 'Ernakulam', 'Gurugram', 'Guwahati', 'Hyderabad',
'Jaipur', 'Jorapokhar', 'Kochi', 'Kolkata', 'Lucknow', 'Mumbai',
'Patna', 'Shillong', 'Talcher', 'Thiruvananthapuram',
'Visakhapatnam'], dtype=object)
df = df.dropna(subset=['AQI'])
+ Code + Text
[ ] df = df.reset_index()




| | City | Date | PM2.5 | NO | NO2 | NOx | CO | SO2 | O3 | AQI | AQI_Bucket |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Ahmedabad | 2015-01-29 | 83.13 | 6.93 | 28.71 | 33.72 | 6.93 | 49.52 | 59.76 | 209.0 | Poor |
| 1 | Ahmedabad | 2015-01-30 | 79.84 | 13.85 | 28.68 | 41.08 | 13.85 | 48.49 | 97.07 | 328.0 | Very Poor |
| 2 | Ahmedabad | 2015-01-31 | 94.52 | 24.39 | 32.66 | 52.61 | 24.39 | 67.39 | 111.33 | 514.0 | Severe |
| 3 | Ahmedabad | 2015-02-01 | 135.99 | 43.48 | 42.08 | 84.57 | 43.48 | 75.23 | 102.70 | 782.0 | Severe |
| 4 | Ahmedabad | 2015-02-02 | 178.33 | 54.56 | 35.31 | 72.80 | 54.56 | 55.04 | 107.38 | 914.0 | Severe |






```
df_south.head()
```

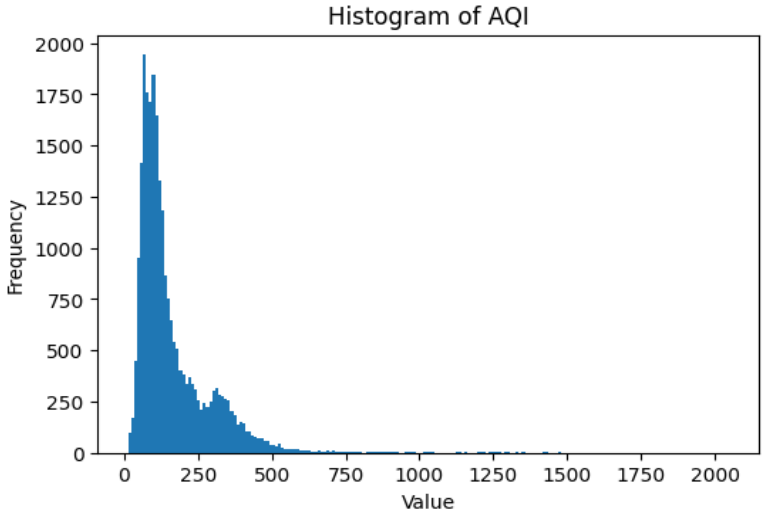
| | index | City | Date | PM2.5 | NO | NO2 | NOx | CO | SO2 | O3 | AQI | AQI_Bucket |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | Ahmedabad | 2015-01-29 | 83.13 | 6.93 | 28.71 | 33.72 | 6.93 | 49.52 | 59.76 | 209.0 | Poor |
| 1 | 1 | Ahmedabad | 2015-01-30 | 79.84 | 13.85 | 28.68 | 41.08 | 13.85 | 48.49 | 97.07 | 328.0 | Very Poor |
| 2 | 2 | Ahmedabad | 2015-01-31 | 94.52 | 24.39 | 32.66 | 52.61 | 24.39 | 67.39 | 111.33 | 514.0 | Severe |
| 3 | 3 | Ahmedabad | 2015-02-01 | 135.99 | 43.48 | 42.08 | 84.57 | 43.48 | 75.23 | 102.70 | 782.0 | Severe |
| 4 | 4 | Ahmedabad | 2015-02-02 | 178.33 | 54.56 | 35.31 | 72.80 | 54.56 | 55.04 | 107.38 | 914.0 | Severe |

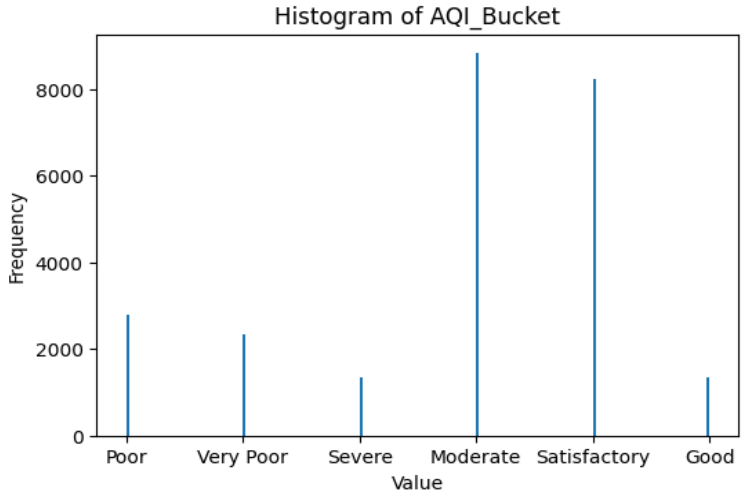
```
[ ] df_north.head()
```

| | index | City | Date | PM2.5 | NO | NO2 | NOx | CO | SO2 | O3 | AQI | AQI_Bucket |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1334 | Aizawl | 2020-03-12 | 31.21 | 7.20 | 1.27 | 10.65 | 0.56 | 4.22 | 2.81 | 52.0 | Satisfactory |
| 1 | 1335 | Aizawl | 2020-03-13 | 38.39 | 7.19 | 0.91 | 10.37 | 0.57 | 4.46 | 0.18 | 60.0 | Satisfactory |
| 2 | 1336 | Aizawl | 2020-03-14 | 43.23 | 7.14 | 1.07 | 10.48 | 0.57 | 4.53 | 0.41 | 62.0 | Satisfactory |
| 3 | 1337 | Aizawl | 2020-03-15 | 33.82 | 7.09 | 0.36 | 9.73 | 0.48 | 4.63 | 0.30 | 70.0 | Satisfactory |
| 4 | 1338 | Aizawl | 2020-03-16 | 27.14 | 5.63 | 2.32 | 8.09 | 0.50 | 4.71 | 13.02 | 54.0 | Satisfactory |

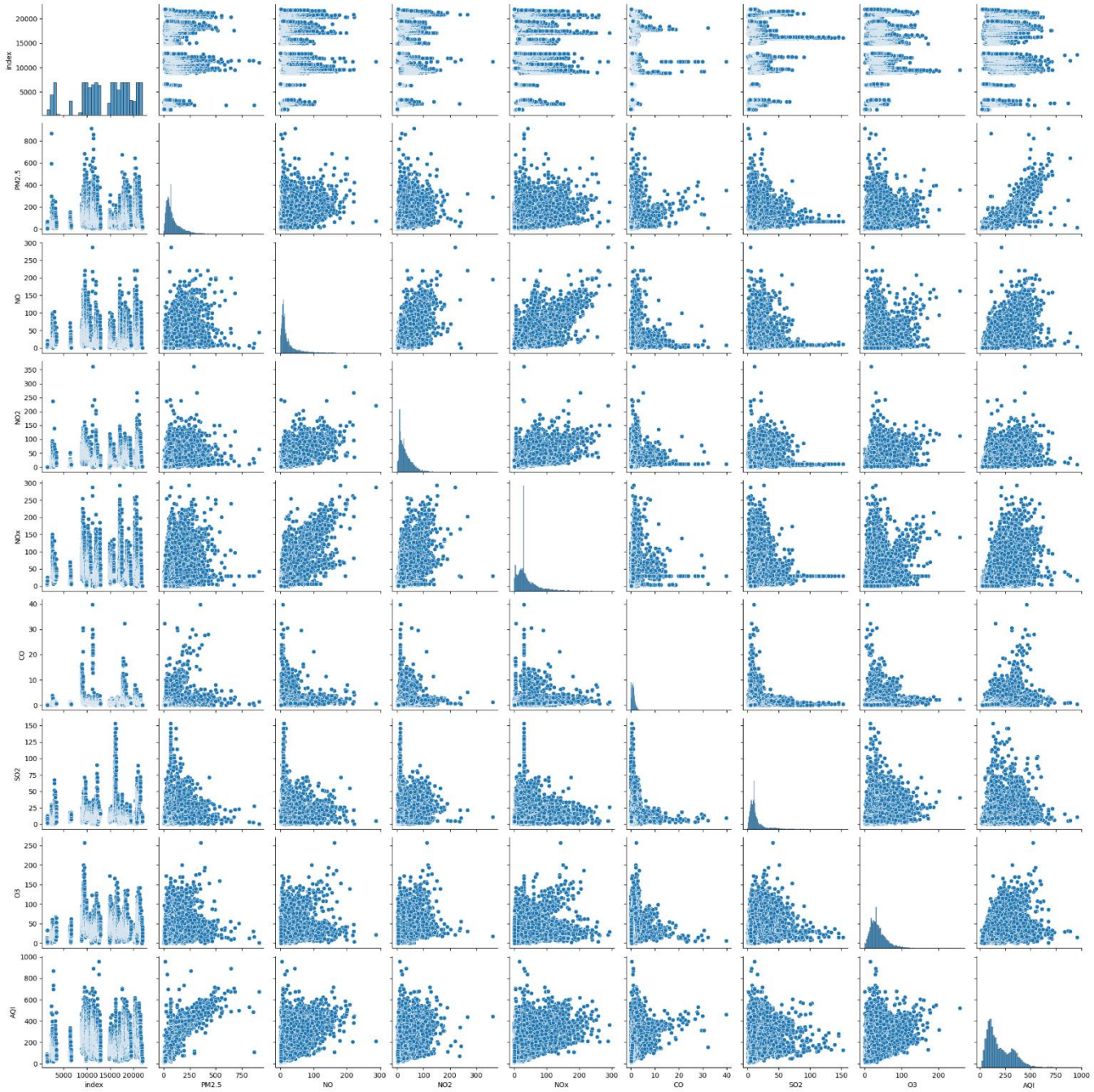
*Univariate Analysis,Bivariate Ananlysis *

```
for column in df_south.iloc[:,2:]:
    plt.figure(figsize=(6, 4))  # Adjust the figure size as needed
    plt.hist(df[column], bins=200)  # Adjust the number of bins as needed
    plt.title(f'Histogram of {column}')
    plt.xlabel('Value')
    plt.ylabel('Frequency')
    plt.show()
```

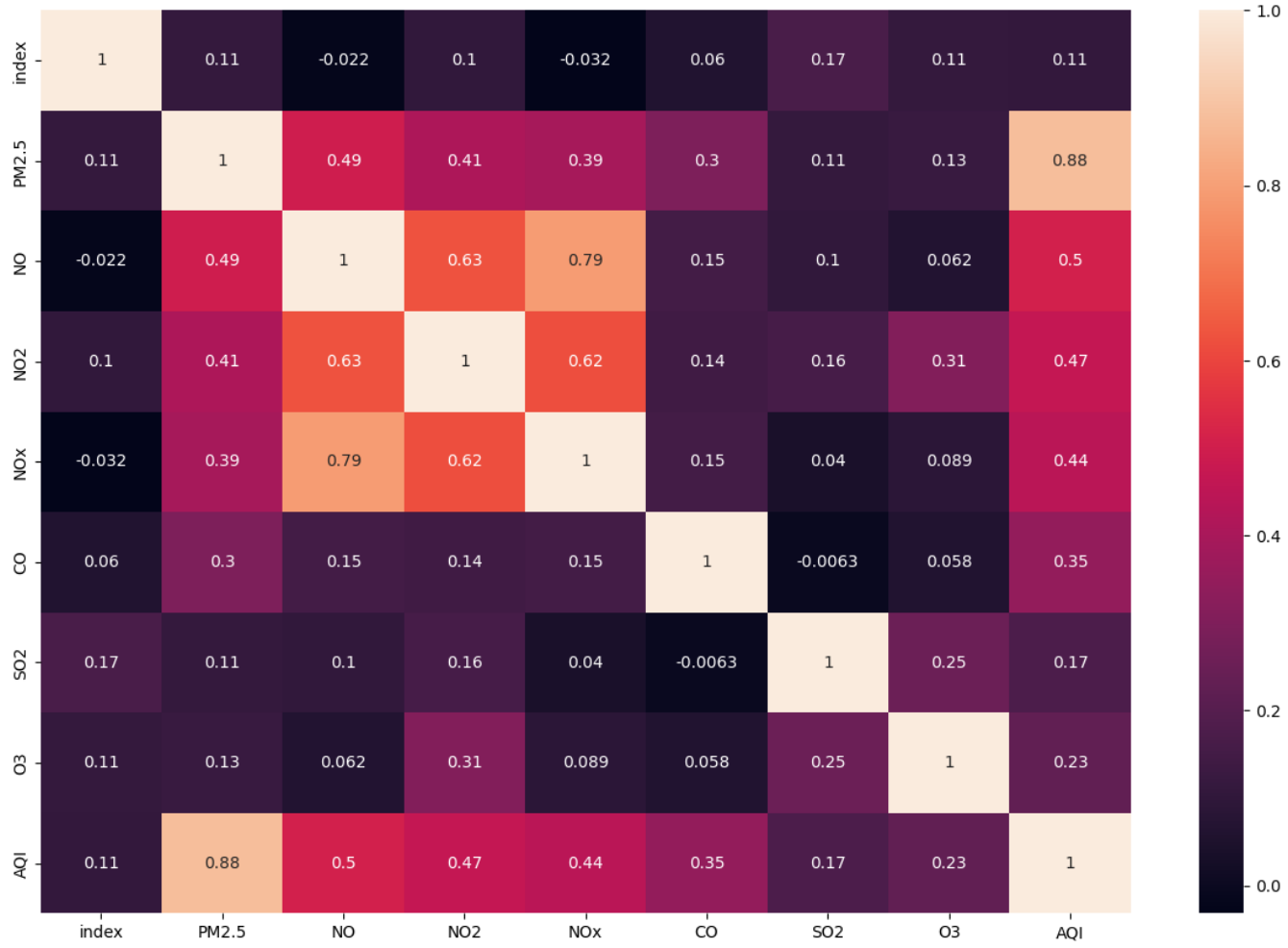


# PAIRPLOT



# HEAT MAP




Ready for model run

```
[ ]  df_north.tail()
```

| | index | City | Date | PM2.5 | NO | NO2 | NOx | CO | SO2 | O3 | AQI | AQI_Bucket |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **11654** | 21924 | Shillong | 2020-06-27 | 9.41 | 0.99 | 2.89 | 1.16 | 0.16 | 4.76 | 24.97 | 35.0 | Good |
| **11655** | 21925 | Shillong | 2020-06-28 | 13.55 | 1.07 | 3.10 | 1.25 | 0.17 | 5.00 | 34.91 | 53.0 | Satisfactory |
| **11656** | 21926 | Shillong | 2020-06-29 | 9.53 | 1.07 | 3.05 | 1.22 | 0.18 | 5.33 | 10.85 | 26.0 | Good |
| **11657** | 21927 | Shillong | 2020-06-30 | 14.74 | 1.01 | 2.85 | 1.12 | 0.20 | 5.05 | 6.20 | 21.0 | Good |
| **11658** | 21928 | Shillong | 2020-07-01 | 16.70 | 0.97 | 2.70 | 1.04 | 0.10 | 4.29 | 17.71 | 24.0 | Good |

```
[ ] minmax_north = df_north.groupby('AQI_Bucket')['AQI'].agg([min, max])

    minmax_north = minmax_north.rename(columns={'min': 'MinValue', 'max': 'MaxValue'})

    print(minmax_north)
```

```
                MinValue  MaxValue
AQI_Bucket
Good             14.0      50.0
Moderate        101.0     200.0
Poor            201.0     300.0
Satisfactory     51.0     100.0
Severe          401.0     956.0
Very Poor       301.0     400.0
```

```
minmax_south = df_south.groupby('AQI_Bucket')['AQI'].agg([min, max])

minmax_south = minmax_south.rename(columns={'min': 'MinValue', 'max': 'MaxValue'})

print(minmax_south)
```

```
                MinValue  MaxValue
AQI_Bucket
Good             13.0      50.0
Moderate        101.0     200.0
Poor            201.0     300.0
Satisfactory     51.0     100.0
Severe          401.0    2049.0
Very Poor       301.0     400.0
```

Classification :

Good : 0-50

Satisfactory : 51-100

Moderate : 101-200

Poor : 201-300

Very Poor : 301-400

Severe : 401-2500

**Splitting data into train, validation and test sets**

```
[ ] from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import StandardScaler
```

```
[ ] X_north = df_north.iloc[:,3:10]    # input features from PM2.5 to O3
    y_north = df_north['AQI']          # AQI numerical values
    yc_north = df_north['AQI_Bucket']  # AQI categorical values (6)
```

```
[ ] X_south = df_south.iloc[:,3:10]
    y_south = df_south['AQI']
    yc_south = df_south['AQI_Bucket']
```

```
scaler = StandardScaler()
Xn_scaled = scaler.fit_transform(X_north)
Xs_scaled = scaler.fit_transform(X_south)
```

```
Xn_train, Xn_test, yn_train, yn_test = train_test_split(Xn_scaled, y_north, test_size=0.3, random_state=42)
_,_, yn_val_train, yn_val_test = train_test_split(Xn_scaled, yc_north, test_size=0.3, random_state=42)

print("X_train shape:", Xn_train.shape)
print("X_test shape:", Xn_test.shape)
print("y_train shape:", yn_train.shape)
print("y_test shape:", yn_test.shape)
print("y_val_train shape:", yn_val_train.shape)
print("y_val_test shape:", yn_val_test.shape)
```

```
X_train shape: (8161, 7)
X_test shape: (3498, 7)
y_train shape: (8161,)
y_test shape: (3498,)
y_val_train shape: (8161,)
y_val_test shape: (3498,)
```

```
[ ] Xs_train, Xs_test, ys_train, ys_test = train_test_split(Xs_scaled, y_south, test_size=0.3, random_state=42)
    _, _, ys_val_train, ys_val_test = train_test_split(Xs_scaled, yc_south, test_size=0.3, random_state=42)

    print("X_train shape:", Xs_train.shape)
    print("X_test shape:", Xs_test.shape)
    print("y_train shape:", ys_train.shape)
    print("y_test shape:", ys_test.shape)
    print("y_val_train shape:", ys_val_train.shape)
    print("y_val_test shape:", ys_val_test.shape)
```

```
[ ] X_train shape: (9233, 7)
    X_test shape: (3958, 7)
    y_train shape: (9233,)
    y_test shape: (3958,)
    y_val_train shape: (9233,)
    y_val_test shape: (3958,)
```

```
[ ] from sklearn.linear_model import LinearRegression
    from sklearn.metrics import mean_squared_error, r2_score
    from sklearn.ensemble import RandomForestRegressor
    from sklearn.svm import SVR
    from sklearn.ensemble import GradientBoostingRegressor
```

```
intervals = [0, 50, 100, 200, 300, 400, 2500]
labels = ['Good', 'Satisfactory', 'Moderate', 'Poor', 'Very Poor', 'Severe']
symptoms = [' Air quality is good in this range, and most people will not experience any symptoms','experience mild symptoms like coughing or throat ir

diseases = ['None','mild symptoms of allergies and sinusitis\nAsthma symptoms can be aggravated','Respiratory Infections','Exacerbation of asthma,Chron

precautions = ['Enjoy outdoor activities and open-air exercise','Sensitive individuals should reduce outdoor activities during periods of elevated AQI'
               'People with asthma and heart conditions should limit outdoor activities\nConsider using air purifiers indoors',
                'Minimize outdoor activities, especially for children, the elderly, and individuals with pre-existing conditions\nUse N95 or equivalent
                'Stay indoors as much as possible, and keep windows and doors sealed\nUse air purifiers with HEPA filters\nVulnerable populations, like
                'Wear N95 or higher-rated masks if you must go outside, although its best to avoid outdoor exposure\nSeek immediate medical attention fo
```

```
[ ] def accuracy_score(y_true, y_pred):
        if y_true.shape != y_pred.shape:
            raise ValueError("Input arrays must have the same shape")

        # Calculate the accuracy
        correct_predictions = np.sum(y_true == y_pred)
        total_predictions = len(y_true)
        accuracy = correct_predictions / total_predictions

        return accuracy
```

## Linear Regression

```
## yn1tn - y-data, n-north, 1-model number, tn-train data
## ys3tt - y-data, s-south, 3-model number, tt-test data
```

+ Code + Text

```
[ ] modeln1 = LinearRegression()

    modeln1.fit(Xn_train, yn_train)

    yn1tn_pred = modeln1.predict(Xn_train)
    yn1tt_pred = modeln1.predict(Xn_test)

    mse_n1 = mean_squared_error(yn_test, yn1tt_pred)
```

```
[ ] r2_n1 = r2_score(yn_test, yn1tt_pred)

    print("Coefficients:", modeln1.coef_)
    print("Intercept:", modeln1.intercept_)
    print("Mean Squared Error:", mse_n1)
    print("R-squared:", r2_n1)
```

```
Coefficients: [93.25763879  1.41136581  4.46157436  9.74706356 11.76129544  6.20145633
 10.31584989]
Intercept: 191.49767365012002
Mean Squared Error: 2726.7661873380316
R-squared: 0.8102139591612361
```

```
yn1tn_cls = np.array(pd.cut(yn1tn_pred, bins=intervals, labels=labels))
yn1tt_cls = np.array(pd.cut(yn1tt_pred, bins=intervals, labels=labels))
sym_cls = np.array(pd.cut(yn1tt_pred, bins=intervals, labels=symptoms))
dis_cls = np.array(pd.cut(yn1tn_pred, bins=intervals, labels=diseases))
pre_cls = np.array(pd.cut(yn1tn_pred, bins=intervals, labels=precautions))
```

```
[ ] print("Enter the values between 0-2500")
    AQI_example = int(input("Enter a AQI Value:"))
    cls = pd.cut([AQI_example], bins=intervals, labels=labels)
    sym = pd.cut([AQI_example], bins=intervals, labels=symptoms)
    dis = pd.cut([AQI_example], bins=intervals, labels=diseases)
    pre = pd.cut([AQI_example], bins=intervals, labels=precautions)
```

```
print("\nAQI_Bucket:")
print(cls[0])

print("\nSymptoms:")
print(sym[0])

print("\nDiseases:")
print(dis[0])

print("\nPrecautions:")
print(pre[0])
```

```
Enter the values between 0-2500
Enter a AQI Value:200

AQI_Bucket:
Moderate

Symptoms:
Coughing, Shortness of breath, or Chest discomfort

Diseases:
Respiratory Infections

Precautions:
People with asthma and heart conditions should limit outdoor activities
Consider using air purifiers indoors
```

```
[ ] acc_n1tn = accuracy_score(yn_val_train, yn1tn_cls)
    print("Accuracy Score:", acc_n1tn)

    acc_n1tt = accuracy_score(yn_val_test, yn1tt_cls)
    print("Accuracy Score:", acc_n1tt)
```

```
Accuracy Score: 0.6089939958338438
Accuracy Score: 0.6203544882790166
```

```
models1 = LinearRegression()

models1.fit(Xs_train, ys_train)

ys1tn_pred = models1.predict(Xs_train)
ys1tt_pred = models1.predict(Xs_test)

mse_s1 = mean_squared_error(ys_test, ys1tt_pred)
r2_s1 = r2_score(ys_test, ys1tt_pred)

print("Coefficients:", models1.coef_)
print("Intercept:", models1.intercept_)
print("Mean Squared Error:", mse_s1)
print("R-squared:", r2_s1)
```

```
Coefficients: [ 43.9404843    4.85987822   8.50970544  -3.62364971 107.93669651
  17.60871042   3.12154787]
Intercept: 144.7356236524405
```

```
[ ]
```

```
Coefficients: [ 43.9404843    4.85987822   8.50970544  -3.62364971 107.93669651
  17.60871042   3.12154787]
Intercept: 144.7356236524405
Mean Squared Error: 4268.043990311957
R-squared: 0.8297421345361442
```

```
[ ] ys1tn_cls = np.array(pd.cut(ys1tn_pred, bins=intervals, labels=labels))
    ys1tt_cls = np.array(pd.cut(ys1tt_pred, bins=intervals, labels=labels))
```

```
acc_s1tn = accuracy_score(ys_val_train, ys1tn_cls)
print("Accuracy Score:", acc_s1tn)

acc_s1tt = accuracy_score(ys_val_test, ys1tt_cls)
print("Accuracy Score:", acc_s1tt)
```

```
Accuracy Score: 0.7173183147406044
Accuracy Score: 0.7230924709449217
```

Random Forest Regressor

```
modeln2 = RandomForestRegressor(n_estimators=100, random_state=42)

modeln2.fit(Xn_train, yn_train)

yn2tn_pred = modeln2.predict(Xn_train)
yn2tt_pred = modeln2.predict(Xn_test)

mse_n2 = mean_squared_error(yn_test, yn2tt_pred)
r2_n2 = r2_score(yn_test, yn2tt_pred)

print("Mean Squared Error:", mse_n2)
print("R-squared:", r2_n2)
```

```
Mean Squared Error: 1436.996272052074
R-squared: 0.8999834182926159
```

```
[ ] yn2tn_cls = np.array(pd.cut(yn2tn_pred, bins=intervals, labels=labels))
    yn2tt_cls = np.array(pd.cut(yn2tt_pred, bins=intervals, labels=labels))
```

```
[ ] acc_n2tn = accuracy_score(yn_val_train, yn2tn_cls)
    print("Accuracy Score:", acc_n2tn)

    acc_n2tt = accuracy_score(yn_val_test, yn2tt_cls)
    print("Accuracy Score:", acc_n2tt)
```

```
Accuracy Score: 0.895355961279255
Accuracy Score: 0.7489994282447112
```

```
models2 = RandomForestRegressor(n_estimators=100, random_state=42)

models2.fit(Xs_train, ys_train)

ys2tn_pred = models2.predict(Xs_train)
ys2tt_pred = models2.predict(Xs_test)

mse_s2 = mean_squared_error(ys_test, ys2tt_pred)
r2_s2 = r2_score(ys_test, ys2tt_pred)

print("Mean Squared Error:", mse_s2)
print("R-squared:", r2_s2)
```

```
Mean Squared Error: 3020.29352735225
R-squared: 0.879516534925942
```

```
[ ] ys2tn_cls = np.array(pd.cut(ys2tn_pred, bins=intervals, labels=labels))
    ys2tt_cls = np.array(pd.cut(ys2tt_pred, bins=intervals, labels=labels))
```

```
acc_s2tn = accuracy_score(ys_val_train, ys2tn_cls)
print("Accuracy Score:", acc_s2tn)

acc_s2tt = accuracy_score(ys_val_test, ys2tt_cls)
print("Accuracy Score:", acc_s2tt)
```

```
Accuracy Score: 0.9120545868081881
Accuracy Score: 0.781455280444669
```

SVM

```
[ ] modeln3 = SVR(kernel='linear', C=1.0)

    modeln3.fit(Xn_train, yn_train)

    yn3tn_pred = modeln3.predict(Xn_train)
    yn3tt_pred = modeln3.predict(Xn_test)

    mse_n3 = mean_squared_error(yn_test, yn3tt_pred)
    r2_n3 = r2_score(yn_test, yn3tt_pred)
```

```
[ ] print("Mean Squared Error:", mse_n3)
    print("R-squared:", r2_n3)
```

```
Mean Squared Error: 3070.865799021244
R-squared: 0.7862642331969184
```

```
[ ] yn3tn_cls = np.array(pd.cut(yn3tn_pred, bins=intervals, labels=labels))
    yn3tt_cls = np.array(pd.cut(yn3tt_pred, bins=intervals, labels=labels))
```

```
acc_n3tn = accuracy_score(yn_val_train, yn3tn_cls)
print("Accuracy Score:", acc_n3tn)

acc_n3tt = accuracy_score(yn_val_test, yn3tt_cls)
print("Accuracy Score:", acc_n3tt)
```

```
Accuracy Score: 0.6739370175223625
Accuracy Score: 0.6841052029731275
```

```
[ ] models3 = SVR(kernel='linear', C=1.0)

    models3.fit(Xs_train, ys_train)

    ys3tn_pred = models3.predict(Xs_train)
    ys3tt_pred = models3.predict(Xs_test)
```

```
[ ]
mse_s3 = mean_squared_error(ys_test, ys3tt_pred)
r2_s3 = r2_score(ys_test, ys3tt_pred)

print("Mean Squared Error:", mse_s3)
print("R-squared:", r2_s3)
```

```
Mean Squared Error: 4523.934669807952
R-squared: 0.8195343201410776
```

```
ys3tn_cls = np.array(pd.cut(ys3tn_pred, bins=intervals, labels=labels))
ys3tt_cls = np.array(pd.cut(ys3tt_pred, bins=intervals, labels=labels))
```

+ Code + Text

```
[ ] acc_s3tn = accuracy_score(ys_val_train, ys3tn_cls)
print("Accuracy Score:", acc_s3tn)

acc_s3tt = accuracy_score(ys_val_test, ys3tt_cls)
print("Accuracy Score:", acc_s3tt)
```

```
Accuracy Score: 0.7278241091736164
Accuracy Score: 0.7397675593734209
```

## Gradient Boosting

```
modeln4 = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)

modeln4.fit(Xn_train, yn_train)

yn4tn_pred = modeln4.predict(Xn_train)
yn4tt_pred = modeln4.predict(Xn_test)

mse_n4 = mean_squared_error(yn_test, yn4tt_pred)
r2_n4 = r2_score(yn_test, yn4tt_pred)

print("Mean Squared Error:", mse_n4)
print("R-squared:", r2_n4)
```

```
Mean Squared Error: 1492.872156979357
R-squared: 0.8960943928866408
```

```
[ ] yn4tn_cls = np.array(pd.cut(yn4tn_pred, bins=intervals, labels=labels))
yn4tt_cls = np.array(pd.cut(yn4tt_pred, bins=intervals, labels=labels))
```

```
acc_n4tn = accuracy_score(yn_val_train, yn4tn_cls)
print("Accuracy Score:", acc_n4tn)

acc_n4tt = accuracy_score(yn_val_test, yn4tt_cls)
print("Accuracy Score:", acc_n4tt)
```

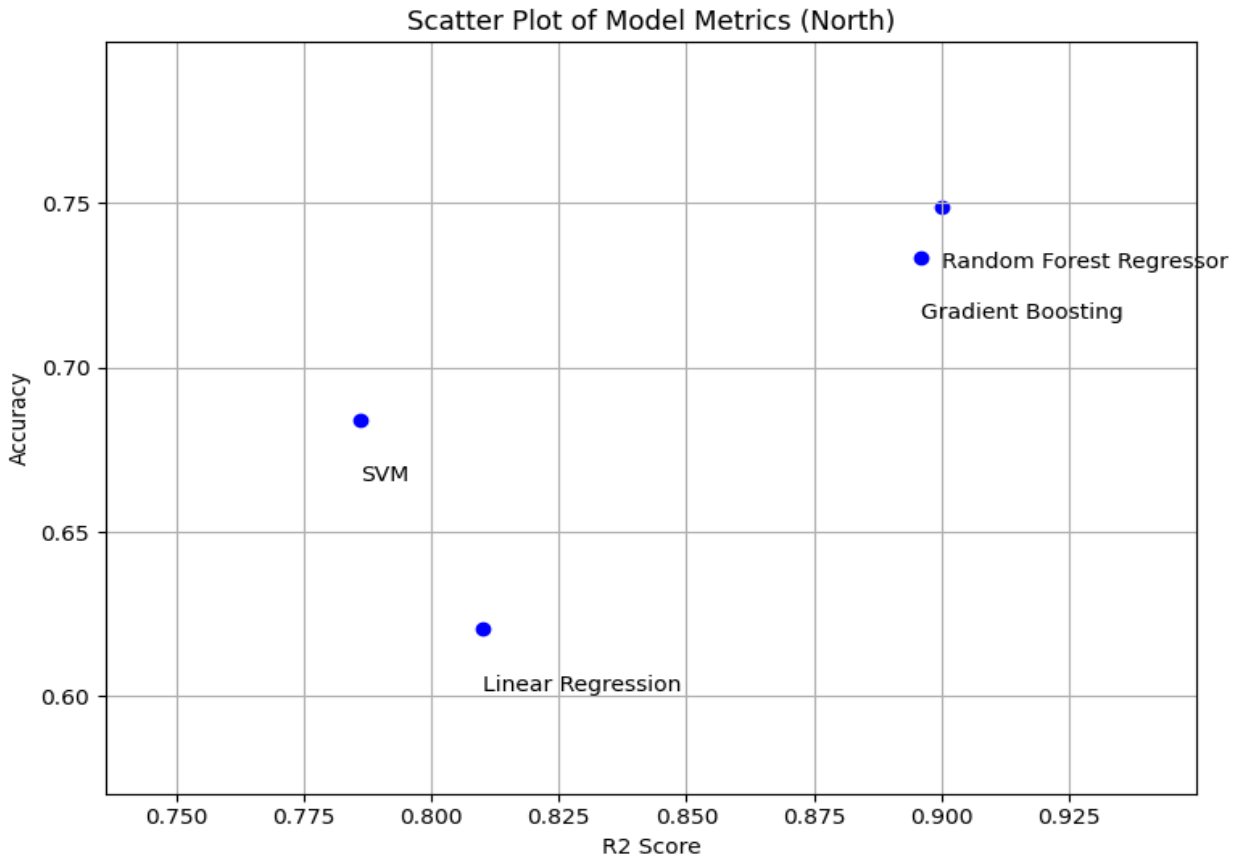
```
Accuracy Score: 0.7361842911407915
Accuracy Score: 0.7332761578044596
```

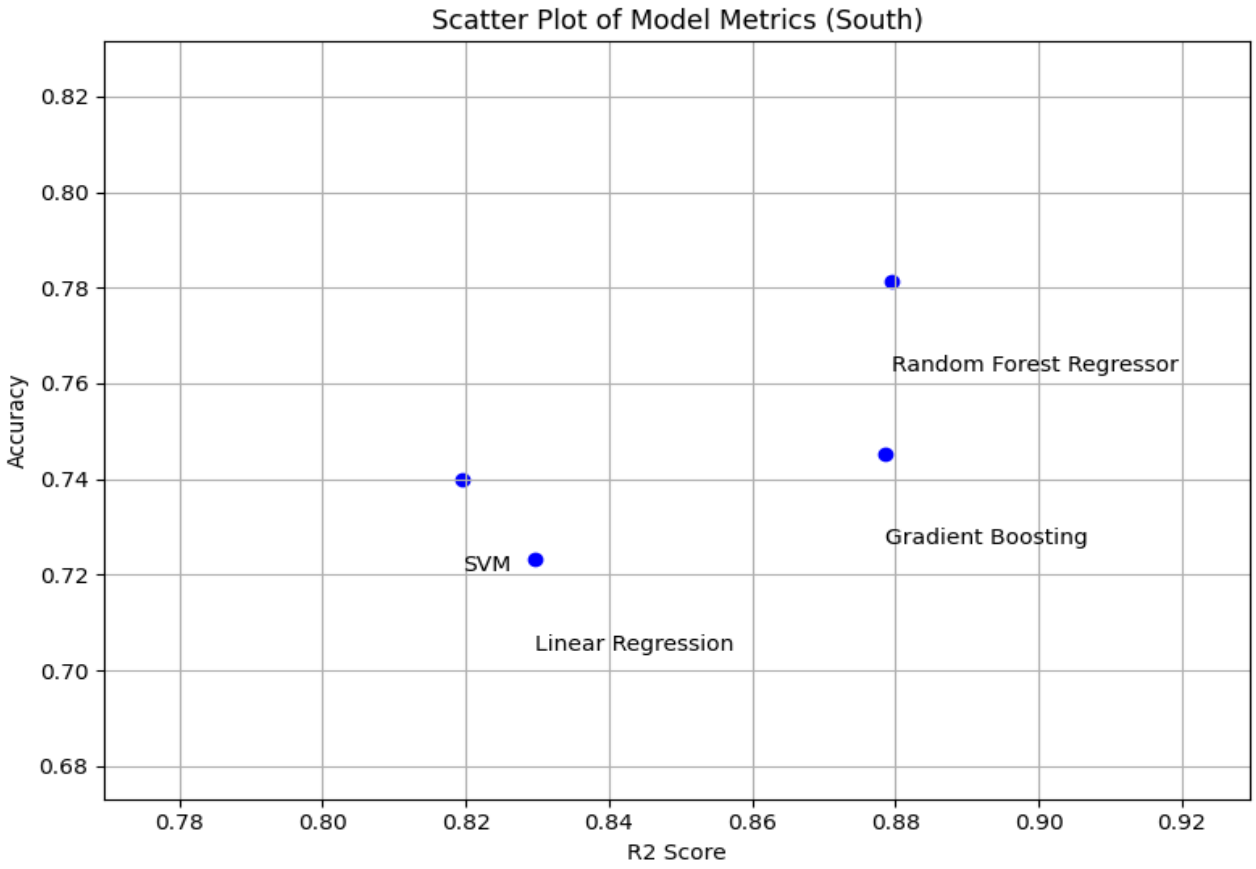
```
models4 = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)

models4.fit(Xs_train, ys_train)

ys4tn_pred = models4.predict(Xs_train)
ys4tt_pred = models4.predict(Xs_test)

mse_s4 = mean_squared_error(ys_test, ys4tt_pred)
r2_s4 = r2_score(ys_test, ys4tt_pred)

print("Mean Squared Error:", mse_s4)
print("R-squared:", r2_s4)
```

```
Mean Squared Error: 3043.4927217343657
R-squared: 0.8785910886735262
```

```
[ ] ys4tn_cls = np.array(pd.cut(ys4tn_pred, bins=intervals, labels=labels))
    ys4tt_cls = np.array(pd.cut(ys4tt_pred, bins=intervals, labels=labels))

[ ] acc_s4tn = accuracy_score(ys_val_train, ys4tn_cls)
    print("Accuracy Score:", acc_s4tn)

    acc_s4tt = accuracy_score(ys_val_test, ys4tt_cls)
    print("Accuracy Score:", acc_s4tt)

    Accuracy Score: 0.7516516841763241
    Accuracy Score: 0.7453259221829207

[ ] # North data metrics
    model_names = ['Linear Regression','Random Forest Regressor', 'SVM', 'Gradient Boosting']
    R2_values = [r2_n1,r2_n2,r2_n3,r2_n4]
    Accuracy_scores = [acc_n1tt,acc_n2tt,acc_n3tt,acc_n4tt]

    metric_north = {'Model': model_names, 'R2 Score': R2_values, 'Accuracy': Accuracy_scores}
    north_metric = pd.DataFrame(metric_north)
```

# DESCRIPTIVE ANALYSIS




Scatter Plot of Model Metrics (South)
Accuracy
R2 Score
0.82
0.80
0.78
0.76
0.74
0.72
0.70
0.68
0.78
0.80
0.82
0.84
0.86
0.88
0.90
0.92
Random Forest Regressor
Gradient Boosting
SVM
Linear Regression