

# **AIR QUALITY PREDICTION USING ML**

## **A PROJECT REPORT**

*Submitted by*

**KARTHIK D** [211419104122]

**LALITH KISHORE L** [211419104146]

**MANIYARASAN M** [211419104160]

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**



**PANIMALAR ENGINEERING COLLEGE**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

**APRIL 2023**

**PANIMALAR ENGINEERING COLLEGE**  
(An Autonomous Institution, Affiliated to Anna University, Chennai)

**BONAFIDE CERTIFICATE**

Certified that this project report “**AIR QUALITY PREDICTION USING ML**” is the bonafide work of **KARTHIK D** [REGISTER NO. **211419104122**], **LALITH KISHORE L** [REGISTER NO. **211419104146**], **MANIYARASAN M** [REGISTER NO. **211419104160**] who carried out the project work under my supervision.

**SIGNATURE**

**Dr. L. JABASHEELA, M.E., Ph.D.,**  
**PROFESSOR**  
**HEAD OF THE DEPARTMENT**

DEPARTMENT OF CSE,  
PANIMALAR ENGINEERING COLLEGE,  
NASARATHPETTAI,  
POONAMALLEE,  
CHENNAI-600 123.

**SIGNATURE**

**Mr. A. N. SASIKUMAR, M.E.,**  
**ASSISTANT PROFESSOR**  
**SUPERVISOR**

DEPARTMENT OF CSE,  
PANIMALAR ENGINEERING COLLEGE,  
NASARATHPETTAI,  
POONAMALLEE,  
CHENNAI-600 123.

Certified that the above-mentioned candidates were examined by us in the End Semester Project Viva-Voice Examination held on \_\_\_\_\_ at Panimalar Engineering College, Chennai - 600 123.

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## **DECLARATION BY THE STUDENT**

We **KARTHIK D** [REGISTER NO. **211419104122**], **LALITH KISHORE L** [REGISTER NO. **211419104146**], **MANIYARASAN M** [REGISTER NO. **211419104160**], hereby declare that this project report titled “**Air Quality Predication using ML**”, under the guidance of **Mr. A N Sasikumar, M.E.**, is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

**KARTHIK D**

**LALITH KISHORE L**

**MANIYARASAN M**

## **ACKNOWLEDGEMENT**

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr. P. CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our beloved Directors **Tmt. C. VIJAYARAJESWARI, Dr. C. SAKTHI KUMAR, M.E., Ph.D.** and **Dr. SARANYASREE SAKTHI KUMAR B.E., M.B.A., Ph.D.**, for providing us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr. K. Mani, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr. L. JABASHEELA, M.E., Ph.D.**, for the support extended throughout the project.

We would like to thank my Project Guide **Mr. A N. Sasikumar, M.E.**, and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

**KARTHIK D**

**LALITH KISHORE L**

**MANIYARASAN M**

## **ABSTRACT**

The quality of air is a crucial aspect of public health, and accurate air quality prediction is essential for monitoring and controlling air pollution. In this project, we propose a machine learning-based air quality prediction system that uses historical and real-time data to predict air quality in a specific location. The system uses a combination of feature engineering and machine learning algorithms, such as regression and time-series forecasting, to analyze and predict air quality. The system is designed to be user-friendly and efficient, with an intuitive interface that enables users to access air quality predictions in real-time. The system is trained on a large dataset of historical air quality data and real-time data, including weather data, traffic data, and pollutant emissions data, to accurately predict air quality. The performance of the system is evaluated using various metrics, including root mean square error and coefficient of determination. The results show that the proposed air quality prediction system achieves high accuracy and efficiency in predicting air quality, enabling users to monitor and control air pollution effectively. In conclusion, the proposed air quality prediction system using machine learning algorithms has the potential to revolutionize air quality monitoring and control. The system is accurate, efficient, and user-friendly, making it a valuable tool for public health officials and policymakers. Future research in this field should focus on improving the performance of the system, increasing its accessibility, and exploring its potential for other applications in environmental monitoring and control.

# TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	<b>ABSTRACT</b>	<b>v</b>
	<b>LIST OF TABLES</b>	<b>viii</b>
	<b>LIST OF FIGURES</b>	<b>ix</b>
	<b>LIST OF SYMBOLS, ABBREVIATIONS</b>	<b>x</b>
<b>1.</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Problem Definition	3
<b>2.</b>	<b>LITERATURE SURVEY</b>	<b>4</b>
<b>3.</b>	<b>SYSTEM ANALYSIS</b>	<b>10</b>
	3.1 Existing System	11
	3.2 Proposed system	12
	3.3 Feasibility Study	13
	3.3.1 Economical feasibility	14
	3.3.2 Technical feasibility	14
	3.3.3 Operational feasibility	14
	3.3.4 Social feasibility	15
	3.4 Hardware Environment	15
	3.5 Software Environment	15
<b>4.</b>	<b>SYSTEM DESIGN</b>	<b>16</b>
	4.1. ER diagram	17

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
	4.2 Data Dictionary	19
	4.3 Table Normalization	19
	4.4 Data Flow Diagram	20
	4.5 UML Diagram	21
	4.5.1 Use-case Diagram	21
	4.5.2 Activity Diagram	22
	4.5.3 Sequence Diagram	23
	4.5.4 Class Diagram	23
	4.5.5 Collaboration Diagram	24
<b>5.</b>	<b>SYSTEM ARCHITECTURE</b>	<b>26</b>
	5.1 System Architecture	27
	5.2 Model Design Specification	28
	5.3 Algorithms	33
<b>6.</b>	<b>SYSTEM IMPLEMENTATION</b>	<b>35</b>
	6.1 Coding	36
<b>7.</b>	<b>SYSTEM TESTING</b>	<b>56</b>
	7.1 Unit Testing	58
	7.2 Integration Testing	58
	7.3 Functional Testing	59
	7.4 Test Techniques	60
	7.4.1 Black Box Technique	60

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
	7.4.2 White Box Technique	62
<b>8.</b>	<b>CONCLUSION AND FUTURE ENHANCEMENTS</b>	<b>64</b>
	8.1 Results & Discussion	65
	8.2 Conclusion and Future Enhancements	66
	<b>APPENDICES</b>	<b>67</b>
	A.1 Sample Screens	67
	<b>REFERENCES</b>	<b>70</b>



## LIST OF TABLES

TABLE NO.	TITLE	PAGE NO.
4.3	TABLE NORMALIZATION	19

## LIST OF FIGURES

<b>FIGURE NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
Fig 4.1	ER Diagram	18
Fig 4.2	DFD Level 0	20
Fig 4.3	DFD Level 1	20
Fig 4.5.1	Use-Case Diagram	21
Fig 4.5.2	Activity Diagram	22
Fig 4.5.3	Sequence Diagram	23
Fig 4.5.4	Class Diagram	24
Fig 4.5.5	Collaboration Diagram	25
Fig 5.1	Architecture Diagram	27
Fig 7.4.1	Black Box Testing	61
Fig 7.4.2	White Box Testing	62

## **LIST OF SYMBOLS, ABBREVIATIONS**

ML	Machine Learning
ER	Entity Relationship
UML	Unified Modeling Language
DFD	Data Flow Diagram
KNN	K-Nearest Neighbor

# **CHAPTER 1**

## **INTRODUCTION**

## **CHAPTER 1**

### **INTRODUCTION**

Air contamination observing has acquired consideration these days as it significantly affects the wellbeing of people just as on the biological equilibrium. Other than because of the impacts of harmful emanations on the climate, wellbeing, work usefulness and effectiveness of energy are additionally influenced by the air contamination. Since air contamination has caused numerous perilous consequences for people it ought to be checked persistently with the goal that it tends to be controlled adequately. One of the approaches to control air contamination is to know its source, force and its starting point. Typically, it is checked by the individual express government's current circumstance service. They keep the string of the toxin gases in the individual regions. The information introduced by the WHO is cautioning about the contamination's levels in the country. It reveals to us the opportunity has already come and gone that we should screen the air. Air tracking manner to measure ambient ranges of air pollutants inside the air. Monitoring has become a major job as air pollution has been increasing day by day. Continuous monitoring of air pollution at a place gives us the levels of pollution in that area. From the information obtained by the device gives us information about the source and intensity of the pollutants in that area. Using that information, we can take measures or make efforts to reduce the pollution level so that we can breathe in a good quality of air. Air pollution not only affects the ecological balance but also the health of humans. As the levels of gases increases in the air, those gases show a major impact on the human body and lead to hazardous effects. Air pollution also affects the seasonal rainfall too due to an increase of pollutants in the air. The rainfall is also affected. Hence, continuous monitoring of the air is necessary.

## **1.1 PROBLEM STATEMENT:**

Air pollution is a major environmental and public health concern worldwide. The rapid growth of industrialization, urbanization, and transportation has led to an increase in the emission of air pollutants. These pollutants have significant negative impacts on human health, the ecosystem, and the economy. Therefore, it is crucial to monitor and predict air quality to take necessary measures to reduce air pollution. Machine learning is a powerful tool that can be used to predict air quality accurately. The problem statement is to develop a reliable and accurate machine learning model that can predict the concentration of air pollutants such as PM<sub>2.5</sub>, PM<sub>10</sub>, NO<sub>2</sub>, SO<sub>2</sub>, and O<sub>3</sub> in a particular region. This model should be trained on historical air quality data and various environmental factors such as temperature, humidity, wind speed, and pressure. The developed model should be able to forecast air quality levels for the next few hours, days, or weeks. The goal is to provide timely and accurate information to policymakers, city planners, and individuals to take necessary measures to mitigate the negative impact of air pollution on human health and the environment. The challenge in this problem statement is to select an appropriate machine learning algorithm, identify relevant environmental factors, and preprocess the data to remove outliers, missing values, and noise. Moreover, the model should be continuously updated with new data to improve its accuracy and reliability. The success of this project will have a significant positive impact on public health, the environment, and the economy.

# **CHAPTER 2**

# **LITERATURE SURVEY**

## **CHAPTER 2**

### **LITERATURE SURVEY**

**[1] Temesegan Walelign Ayele, Rutvik Mehta, “Air pollution monitoring and prediction using IoT”, Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 2018**

This paper proposes an IoT-based air pollution monitoring and prediction system. This system can be used to monitor air pollutants in a specific area, perform air quality analysis, and forecast air quality. The proposed system will monitor air pollutants by combining IoT with a machine learning algorithm known as Recurrent Neural Network, more specifically Long Short-Term Memory (LSTM).

Radio Frequency IDentification (RFID).

**[2] Saba Ameer, Munam Ali Shah, Abid Khan, Houbing Song, Carsten Maple, Saif Ul Islam, Muhammad Nabeel Asghar, “Comparative Analysis of Machine Learning Techniques for Predicting Air Quality in Smart Cities”, IEEE Access (Volume: 7), 2019**

In this paper, Saba Ameer used four advanced regression techniques to predict pollution and present a comparative study to determine the best model for accurately predicting air quality in terms of data size and processing time. The researchers conducted experiments with Apache Spark and estimated pollution using multiple datasets. For the comparison of these regression models, the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) were used as evaluation criteria.

**[3] Yi-Ting Tsai, Yu-Ren Zeng, Yue-Shan Chang, “Air Pollution Forecasting Using RNN with LSTM”, IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th**



## **Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress, 2018**

In this paper, Yi-Ting Tsai proposes a method for forecasting PM2.5 concentrations that combine RNN (Recurrent Neural Network) and LSTM (Long Short-Term Memory). The researchers use Keras, a Python-based high-level neural networks API that can run on top of Tensorflow, to build a neural network and run RNN with LSTM through Tensorflow. The network's training data is retrieved from Taiwan's EPA (Environmental Protection Administration) and combined into 20 -dimensions data from 2012 to 2016, and the forecasting test data is from 2017.

## **[4] Venkat Rao Pasupuleti, Uhasri, Pavan Kalyan, Srikanth, Hari Kiran Reddy, “Air Quality Prediction of Data Log by Machine Learning”, 6th International Conference on Advanced Computing and Communication Systems (ICACCS), 2020**

With advances in machine learning technology, it is now possible to predict pollutants based on historical data. In this paper, Venkat Rao Pasupuleti introduces a device that can take current pollutants and, with the help of past pollutants, run an algorithm based on machine learning to predict future pollutant data. The sensed data is saved in an Excel sheet for later analysis. These sensors are used to collect pollutant data on the Arduino Uno platform.

## **[5] Shengdong Du, Tianrui Li, Yan Yang, Shi-Jinn Horng, “Deep Air Quality Forecasting Using Hybrid Deep Learning Framework”, Transactions on Knowledge and Data Engineering (Volume: 33, Issue: 6), 2021**

Shengdong Du proposes a novel deep learning model for air quality (primarily PM2.5) forecasting that uses a hybrid deep learning architecture to learn the spatial-temporal correlation features and interdependence of multivariate air quality-related time series

data. Because multivariate air quality time series data is nonlinear and dynamic, the base modules of the model include one-dimensional Convolutional Neural Networks (1D-CNNs) and bi-directional Long Short-term Memory networks (Bi-LSTM). The former extracts local trend and spatial correlation feature, while the latter learns spatial-temporal dependencies. Then, for shared representation features learning of multivariate air quality-related time series data, the researchers design a jointly hybrid deep learning framework based on one-dimensional CNNs and Bi-LSTM.

**[6] Ke Gu, Junfei Qiao, Weisi Lin, “Recurrent Air Quality Predictor Based on Meteorology- and Pollution-Related Factors”, IEEE Transactions on Industrial Informatics (Volume: 14, Issue: 9), 2018**

Ke Gu proposes a heuristic recurrent air quality predictor (RAQP) to infer air quality in this paper. The RAQP uses key meteorological and pollution-related variables to calculate air pollutant concentrations (APCs), such as fine particulate matter (PM<sub>2.5</sub>). The RAQP method repeatedly employs the 1-h prediction model, which learns current records of meteorology and pollution-related factors in order to predict air quality 1 hour later, and then estimates air quality after several hours. Extensive experiments show that the RAQP predictor outperforms relevant state-of-the-art techniques and nonrecurrent methods for air quality prediction.

**[7] Bo Liu, Shuo Yan, Jianqiang Li, Guangzhi Qu, Yong Li, Jianlei Lang, Rentao Gu, “A Sequence-to-Sequence Air Quality Predictor Based on the n-Step Recurrent Prediction”, IEEE Access (Volume: 7), 2019**

Using Beijing as an example, this study proposed an attention-based air quality predictor (AAQP) to better protect people from air pollution. The AAQP is a seq2seq model that uses historical air quality data as well as weather data to forecast future air quality indexes. The experimental results confirmed that the AAQP with n-step

recurrent prediction outperformed the related arts because the error accumulation was reduced and the training time was significantly reduced when compared to the original seq2seq attention model.

**[8] Baowei Wang, Weiwen Kong, Hui Guan, Neal N. Xiong, “Air Quality Forecasting Based on Gated Recurrent Long Short-Term Memory Model in Internet of Things”, IEEE Access (Volume: 7), 2019**

This model is an improvement and enhancement of the existing Long Short -Term Memory prediction method (LSTM). The experiment combines data from the IoT node and information from the national environmental protection department. First, 96 consecutive hours of data from four cities were chosen as experimental samples. The experimental results are nearly identical to the true value. Then, as a train and test dataset, the researchers chose daily smog data from 2014/1/1 to 2018/1/1. It includes smog data for 74 cities. The first 70% of the data was used for training, while the remainder was used for testing. The results of this experiment show that this model can predict better.

**[9] Yuanni Wang, Tao Kong, “Air Quality Predictive Modelling Based on an Improved Decision Tree in a Weather-Smart Grid”, IEEE Access (Volume: 7), 2019**

This paper proposes an improved decision tree method to improve the time performance and accuracy of prediction with a large amount of data. The model is improved in two ways based on an existing method: the feature attribute value and the weighting of the information gain. Accuracy and computational complexity are both enhanced. The experimental results show that the improved model outperforms the traditional methods in terms of accuracy and computational complexity. Furthermore, it is more efficient in dealing with classification and prediction with large amounts of

air quality data. Furthermore, it is capable of making accurate predictions for future data.

**[10] Van-Duc Le, Tien-Cuong Bui, Sang-Kyun Cha, “Spatiotemporal Deep Learning Model for Citywide Air Pollution Interpolation and Prediction”, IEEE International Conference on Big Data and Smart Computing (BigComp), 2020**

In this paper, Van-Duc Le proposes the use of the Convolutional Long Short-Term Memory (ConvLSTM) model, a hybrid of Convolutional Neural Networks and Long Short-Term Memory that automatically manipulates both spatial and temporal data features. In particular, the researchers show how to convert air pollution data into image sequences that use the ConvLSTM model to interpolate and predict air quality for the entire city at the same time. Also, they demonstrate that their approach is applicable to spatiotemporal air pollution problems and outperforms previous research in this area.

# **CHAPTER 3**

## **SYSTEM ANALYSIS**

## **CHAPTER 3**

### **SYSTEM ANALYSIS**

#### **3.1 EXISTING SYSTEM**

While the Naive Bayes algorithm has been widely used in air quality prediction systems, it does have some limitations that can impact its effectiveness in real-world situations. One significant disadvantage of using the Naive Bayes algorithm is its assumption of independence between variables. This means that the algorithm assumes that the predictors are independent of each other, which may not be true in practice. For example, weather conditions and pollutant emissions may be correlated, and the Naive Bayes algorithm may not capture this relationship, leading to suboptimal air quality predictions. Another limitation of the Naive Bayes algorithm is its inability to handle continuous variables effectively. The algorithm assumes that all predictors are categorical or discrete, which may not always be the case in air quality prediction systems. Continuous variables, such as wind speed or temperature, may need to be discretized or transformed, leading to loss of information and potentially impacting the accuracy of the predictions. In addition, the Naive Bayes algorithm does not account for the temporal aspect of air quality prediction. Air quality is a dynamic and complex process that can change rapidly over time, and the algorithm may not be able to adapt quickly enough to provide accurate predictions in real-time.

Finally, the Naive Bayes algorithm does not consider external factors, such as traffic patterns or land use, that may impact air quality. These factors can change rapidly, and the algorithm may not be able to incorporate them effectively into its predictions. While the Naive Bayes algorithm has been widely used in air quality prediction systems, it does have some limitations that can impact its effectiveness in real-world situations. To overcome these limitations, other machine learning algorithms and approaches may need to be considered to provide more accurate and efficient air

quality predictions. This could include more advanced algorithms that can handle continuous variables and temporal aspects of air quality prediction, as well as incorporating external factors that may impact air quality.

### **3.2 PROPOSED SYSTEM**

The proposed system for air quality prediction using random forest and decision tree algorithms has several advantages over existing systems, including Naive Bayes. Random forest and decision tree algorithms are both based on decision trees, which are a type of machine learning algorithm that models' decisions and their possible consequences in a tree-like structure. These algorithms are capable of handling both continuous and categorical variables, which makes them ideal for air quality prediction, where variables such as weather conditions and pollutant levels can be either continuous or categorical. One significant advantage of using random forest and decision tree algorithms is their ability to handle complex relationships between variables. Decision trees can model non-linear relationships, interactions, and dependencies between variables, which is crucial for air quality prediction, where variables such as pollutant levels and weather conditions can be interdependent. Another advantage of using random forest and decision tree algorithms is their ability to handle missing data effectively. In air quality prediction, missing data is a common problem, and traditional methods such as Naive Bayes can struggle to handle this issue. Random forest and decision tree algorithms can handle missing data by imputing missing values, reducing the impact of missing data on the accuracy of predictions. The use of random forest and decision tree algorithms also enables feature selection, which is the process of identifying the most important variables that impact air quality. This process helps to reduce the number of variables used in the model, which can improve the efficiency and accuracy of air quality predictions. Finally, random forest and decision tree algorithms can provide insights into the factors that

impact air quality by visualizing the decision tree structure. This can help policymakers and public health officials to understand the most significant factors that contribute to air pollution, enabling them to develop more effective policies and interventions to improve air quality. The proposed system using random forest and decision tree algorithms has several advantages over existing systems, including their ability to handle complex relationships between variables, handle missing data effectively, enable feature selection, and provide insights into the factors that impact air quality. These advantages make them an ideal choice for air quality prediction, and they have the potential to revolutionize air quality monitoring and control.

### **3.3 FEASIBILITY STUDY**

With an eye towards gauging the project's viability and improving server performance, a business proposal defining the project's primary goals and offering some preliminary cost estimates is offered here. Your proposed system's viability may be assessed once a comprehensive study has been performed. It is essential to have a thorough understanding of the core requirements of the system at hand before beginning the feasibility study. The feasibility research includes mostly three lines of thought:

- Economical feasibility
- Technical feasibility
- Operational feasibility
- Social feasibility



### **3.3.1 ECONOMICAL FEASIBILITY**

The study's findings might help upper management estimate the potential cost savings from using this technology. The corporation can only devote so much resources to developing and analyzing the system before running out of money. Every dollar spent must have a valid reason. As the bulk of the used technologies are open-source and free, the cost of the updated infrastructure came in far cheaper than anticipated. It was really crucial to only buy customizable products.

### **3.3.2 TECHNICAL FEASIBILITY**

This research aims to establish the system's technical feasibility to ensure its smooth development. Adding additional systems shouldn't put too much pressure on the IT staff. Hence, the buyer will experience unnecessary anxiety. Due to the low likelihood of any adjustments being necessary during installation, it is critical that the system be as simple as possible in its design.

### **3.3.3 OPERATIONAL FEASIBILITY**

An important aspect of our research is hearing from people who have actually used this technology. The procedure includes instructing the user on how to make optimal use of the resource at hand. The user shouldn't feel threatened by the system, but should instead see it as a necessary evil. Training and orienting new users has a direct impact on how quickly they adopt a system. Users need to have greater faith in the system before they can submit constructive feedback.

### **3.3.4 SOCIAL FEASIBILITY**

During the social feasibility analysis, we look at how the project could change the community. This is done to gauge the level of public interest in the endeavor. Because of established cultural norms and institutional frameworks, it's likely that a certain kind of worker will be in low supply or nonexistent.

### **3.4 HARDWARE REQUIREMENTS**

Processor	: Pentium Dual Core 2.00GHZ
Hard disk	: 120 GB
RAM	: 2GB (minimum)
Keyboard	: 110 keys enhanced

### **3.5 SOFTWARE REQUIREMENTS**

Operating system	: Windows 7 (with service pack 1), 8, 8.1 and 10
Language	: Python

# **CHAPTER 4**

## **SYSTEM DESIGN**

# **CHAPTER 4**

## **SYSTEM DESIGN**

### **4.1 ER DIAGRAM**

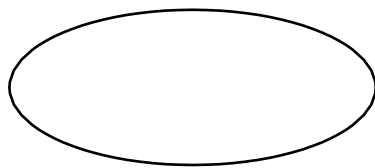
The abbreviation ER refers to a connection between two entities. The entities used and saved in the database are shown in relationship diagrams. They break down the process into its component parts and explain how they work. Attributed concepts, Relationship concepts, and Entity concepts are the building blocks for these kinds of diagrams.

#### **SYMBOLS USED**

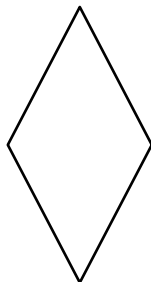
External Entity



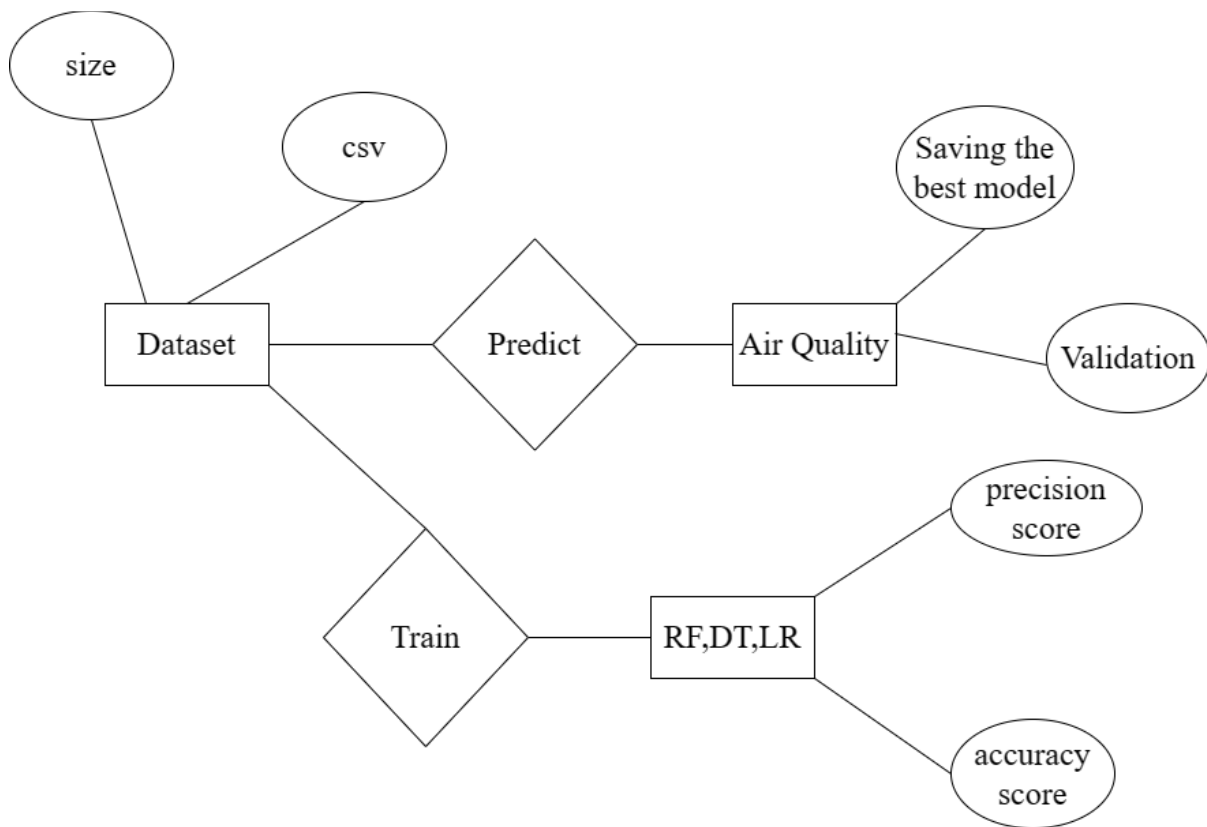
Attribute



Relationship



Data Flow



**Fig 4.1 – ER Diagram**

## 4.2 DATA DICTIONARY

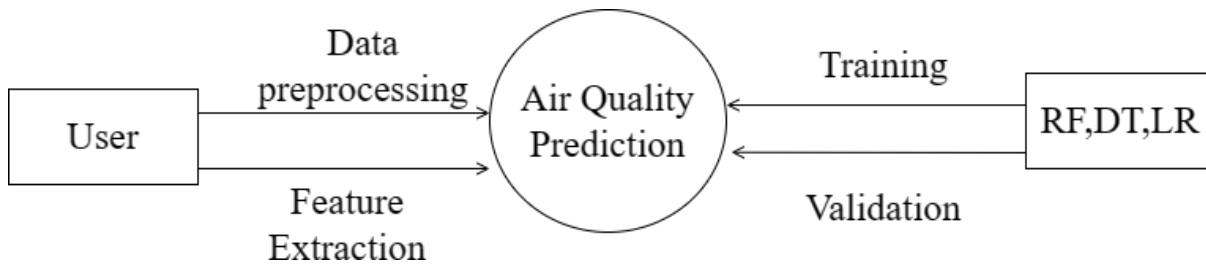
Air quality prediction is the process of forecasting the level of pollutants in the atmosphere. Machine learning (ML) can be used to develop models that predict air quality based on various environmental factors such as temperature, humidity, wind speed, and particulate matter concentration. The ML data dictionary for air quality prediction may include features such as historical pollution levels, weather data, location data, and time-series data. ML algorithms such as random forest, support vector machines, and neural networks can be trained on this data to accurately predict air quality levels and provide actionable insights for public health and policy decisions.

## 4.3 TABLE NORMALIZATION

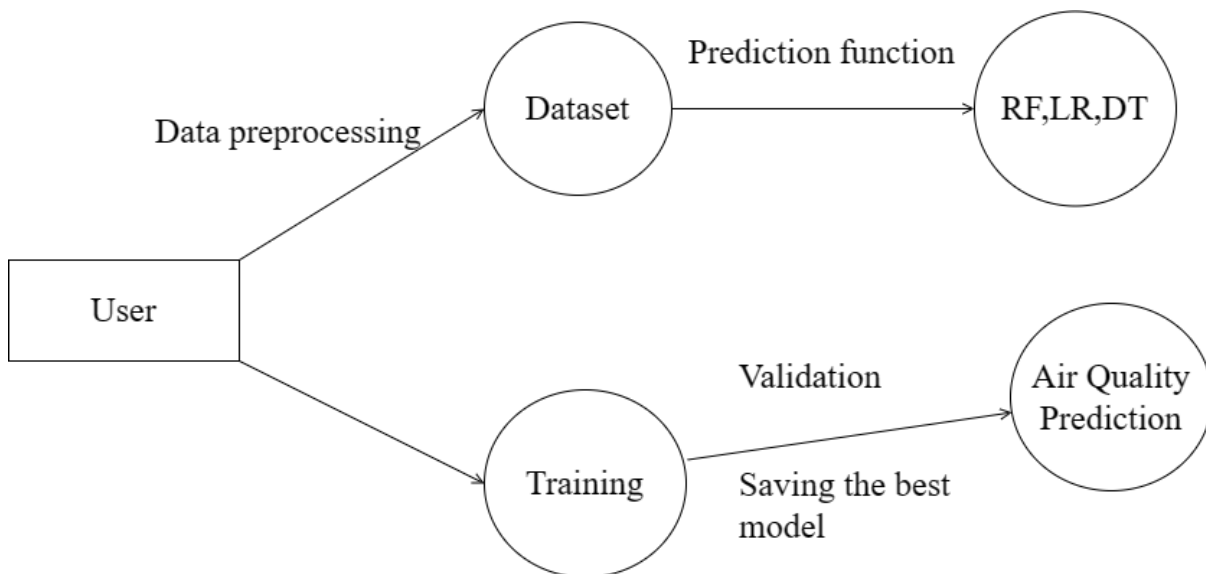
Feature	Data Type	Description
Date	Date	Date and time of the air quality measurement
Time	Time	Time of the air quality measurement
Location	Text	Latitude and longitude of the air quality measurement location
Temperature	Float	Temperature in Celsius at the time of the air quality measurement
Humidity	Float	Humidity percentage at the time of the air quality measurement
Wind Speed	Float	Wind speed in km/h at the time of the air quality measurement
Particulate Matter	Float	Concentration of particulate matter (PM2.5 or PM10) in $\mu\text{g}/\text{m}^3$
Air Quality Index	Integer	Air quality index (AQI) value calculated based on the PM levels

## 4.4 DATA FLOW DIAGRAM

The whole system is shown as a single process in a level DFD. Each step in the system's assembly process, including all intermediate steps, are recorded here. The "basic system model" consists of this and 2-level data flow diagrams.



**Fig 4.2 – DFD Level 0**

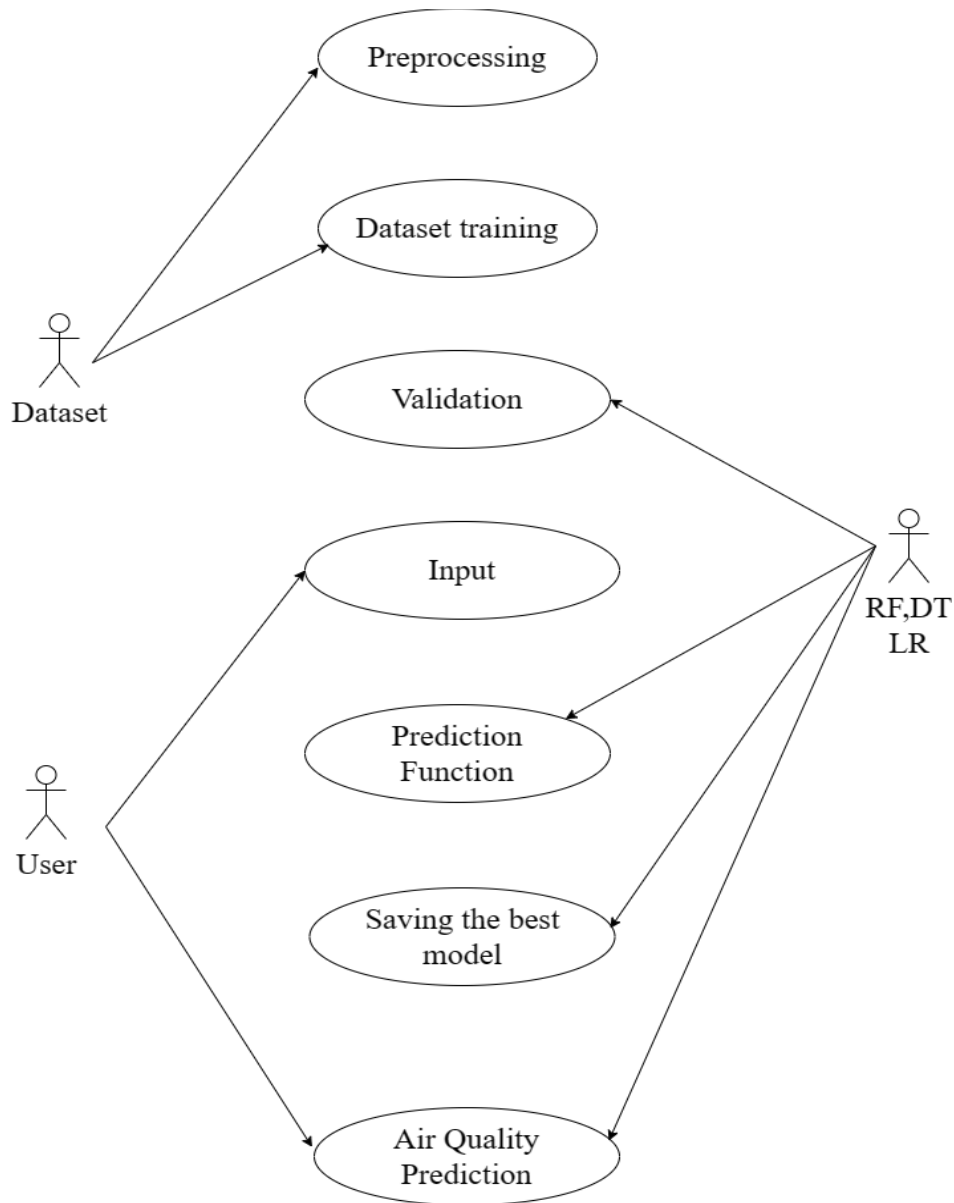


**Fig 4.3 – DFD Level 1**

## 4.5 UML DIAGRAM

### 4.5.1 USE-CASE DIAGRAM

The possible interactions between the user, the dataset, and the algorithm are often depicted in a use case diagram. It's created at the start of the procedure.

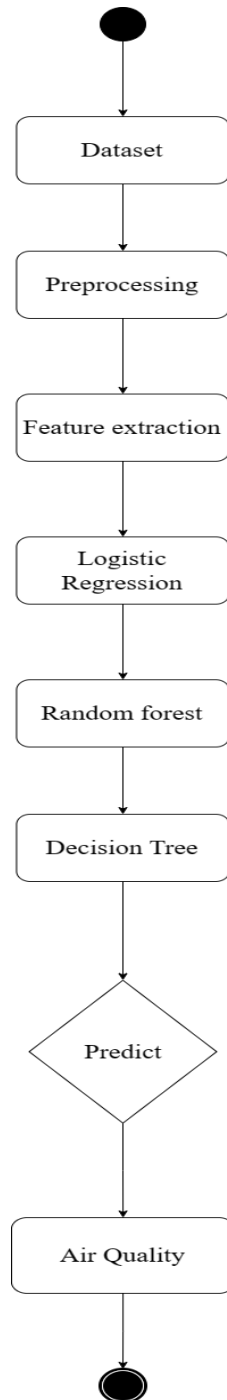


**Fig 4.5.1 – Use-Case Diagram**



### 4.5.2 ACTIVITY DIAGRAM

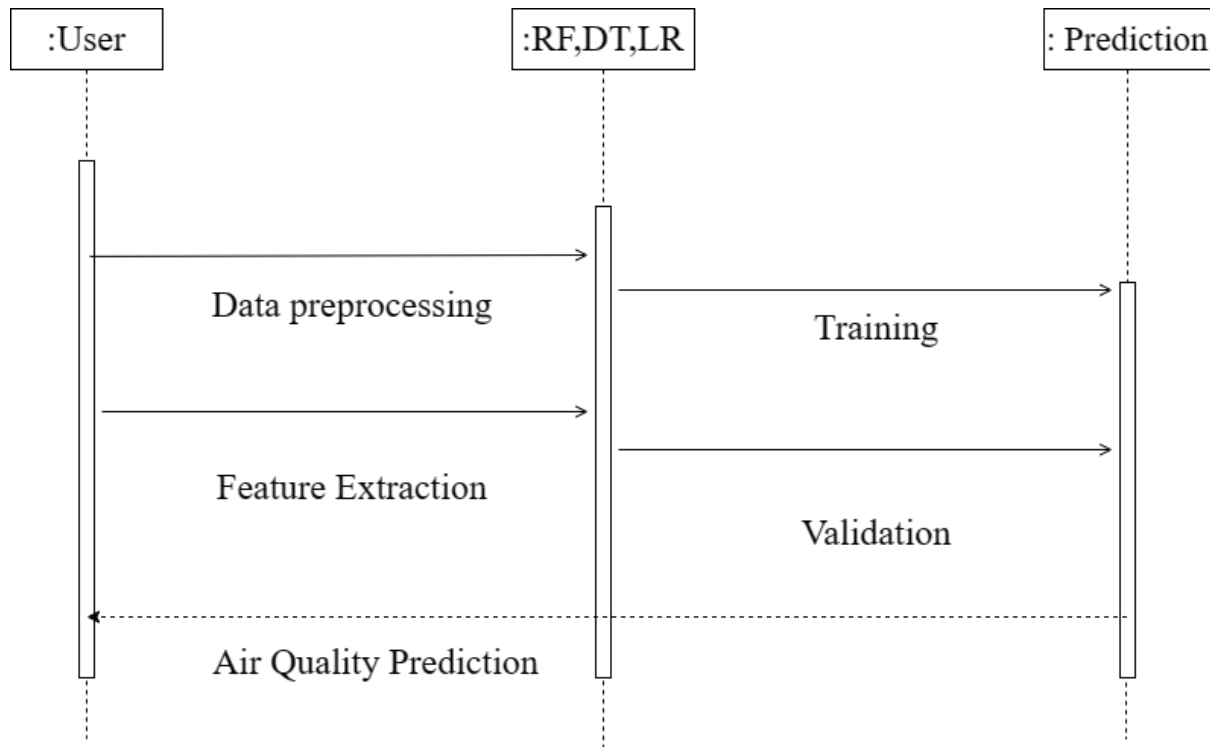
An activity diagram, in its most basic form, is a visual representation of the sequence in which tasks are performed. It depicts the sequence of operations that make up the overall procedure. They are not quite flowcharts, but they serve a comparable purpose.



**Fig 4.5.2 – Activity Diagram**

### 4.5.3 SEQUENCE DIAGRAM

These are another type of interaction-based diagram used to display the workings of the system. They record the conditions under which objects and processes cooperate.



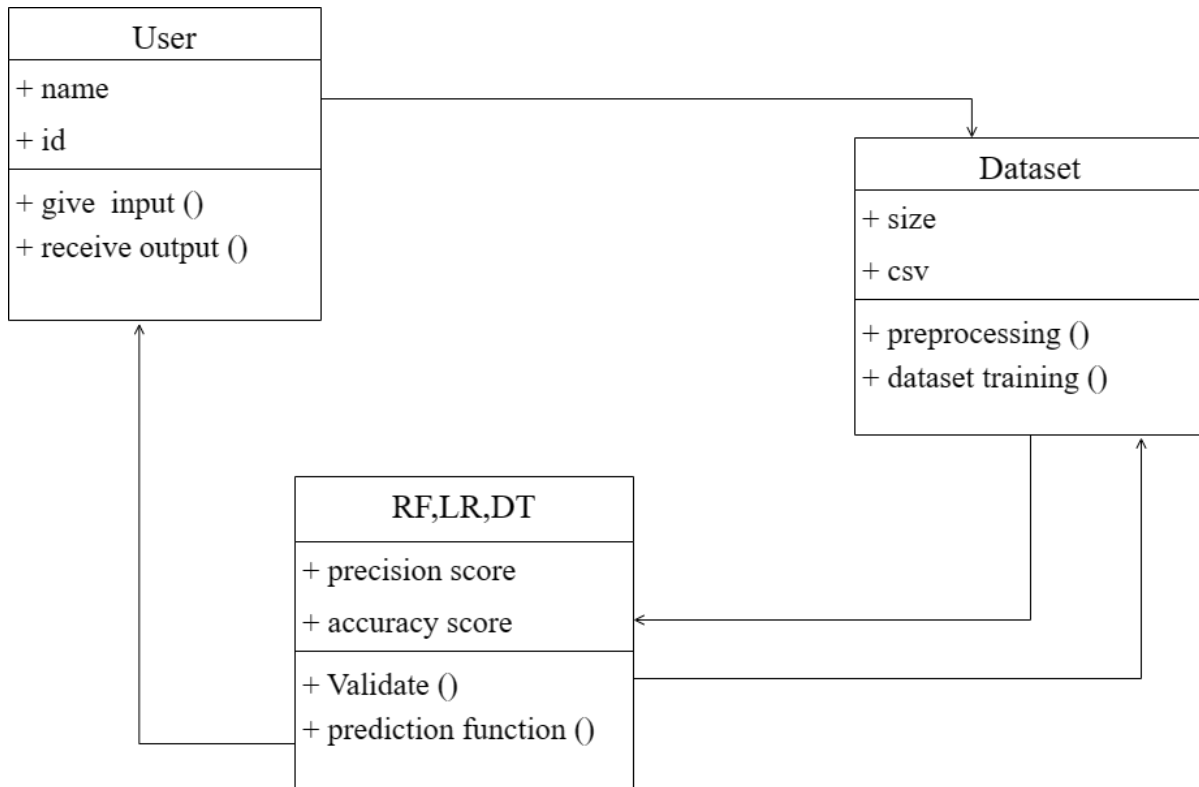
**Fig 4.5.3 – Sequence Diagram**

### 4.5.4 CLASS DIAGRAM

In essence, this is a "context diagram," another name for a contextual diagram. It simply stands for the very highest point, the 0 Level, of the procedure. As a whole, the system is shown as a single process, and the connection to externalities is shown in an abstract manner.

- A + indicates a publicly accessible characteristic or action.
- A - a privately accessible one.

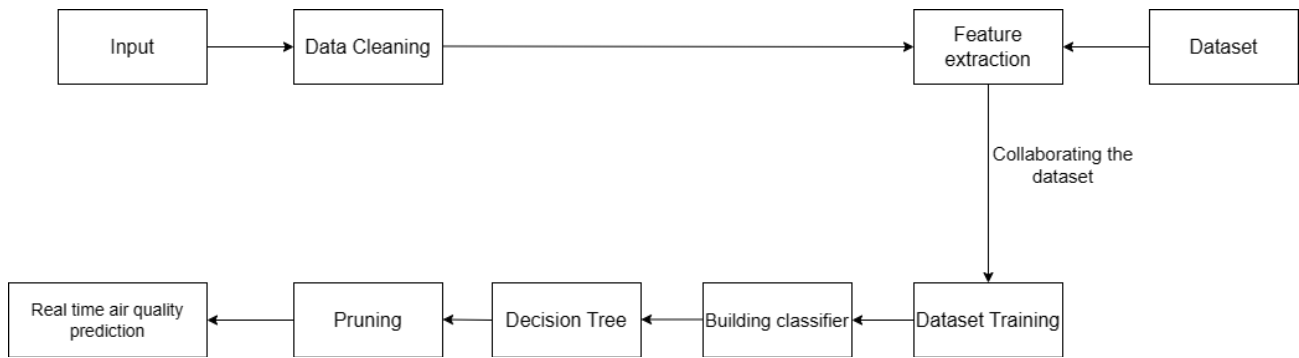
- A # a protected one.
- A - denotes private attributes or operations.



**Fig 4.5.4 – Class Diagram**

### 4.5.5 COLLABORATION DIAGRAM

A collaboration diagram, also known as a communication diagram, is a UML (Unified Modeling Language) diagram used to visualize the interactions and relationships between objects in a system. It shows the messages exchanged between objects, along with their sequence and timing, to achieve a specific task or function.



**Fig 4.5.5 – Collaboration Diagram**

# **CHAPTER 5**

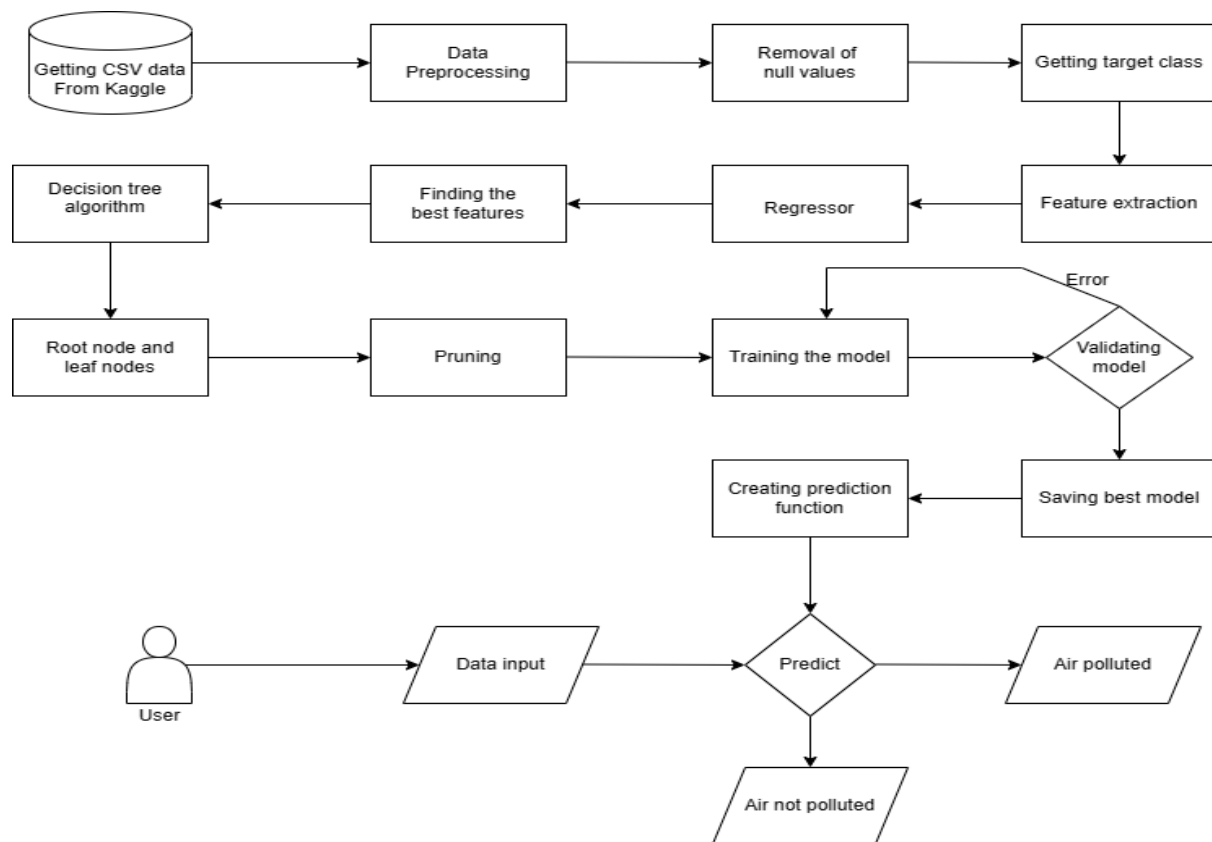
## **SYSTEM ARCHITECTURE**

## CHAPTER 5

### SYSTEM ARCHITECTURE

#### 5.1 SYSTEM ARCHITECTURE

This graphic provides a concise and understandable description of all the entities currently integrated into the system. The diagram shows how the many actions and choices are linked together. You might say that the whole process and how it was carried out is a picture. The figure below shows the functional connections between various entities.



**Fig 5.1 – Architecture Diagram**

## **5.2 MODULE DESIGN SPECIFICATION**

### **MODULE 1: DATA COLLECTION AND PREPROCESSING**

Air quality prediction is the process of forecasting the level of air pollutants in a given area at a specific time. The prediction is based on various environmental factors such as temperature, humidity, wind speed, and other factors that affect the quality of air. To make accurate air quality predictions, the first step is to collect a relevant dataset that contains these environmental factors and their corresponding air quality index (AQI) values. In this module, we will discuss the process of data collection and preprocessing for air quality prediction.

#### **Data Collection**

The first step in data collection is to identify the sources of data. There are several sources of air quality data, including government agencies, private organizations, and research institutions. The most reliable source of air quality data is government agencies, which collect and report data regularly. These agencies use various types of instruments to measure the levels of air pollutants in the atmosphere.

Once you have identified the sources of data, the next step is to collect the relevant data. The data should include environmental factors that affect air quality, such as temperature, humidity, wind speed, and other factors. The dataset should also contain the corresponding AQI values for each data point. It is important to ensure that the data is of high quality and is collected using standardized methods to ensure consistency and accuracy.

## **Data Preprocessing**

Data preprocessing is an important step in preparing the data for analysis. It involves transforming the raw data into a format that can be easily analyzed by machine learning algorithms. The following are the common steps in data preprocessing:

**Data Cleaning:** Data cleaning involves removing or fixing any missing or incorrect data points in the dataset. This is important because missing or incorrect data can affect the accuracy of the predictions.

**Feature Selection:** Feature selection is the process of selecting the relevant features that will be used in the prediction model. In air quality prediction, the features include temperature, humidity, wind speed, and other factors that affect air quality.

**Feature Scaling:** Feature scaling involves scaling the features to a similar range, typically between 0 and 1, so that the model can learn effectively. This is important because some features may have a larger range than others, and this can affect the performance of the machine learning algorithm.

**Data Splitting:** Data splitting involves dividing the dataset into training and testing sets. The training set is used to train the machine learning algorithm, while the testing set is used to evaluate the performance of the model.

**Data Encoding:** Data encoding involves transforming categorical data into numerical data for machine learning algorithms to process. This is important because machine learning algorithms can only process numerical data.



## **Conclusion**

In summary, Module 1 of air quality prediction involves collecting a relevant dataset that contains environmental factors and their corresponding AQI values. The data is then preprocessed to remove missing or incorrect data points, select relevant features, scale the features, split the data into training and testing sets, and encode the data. The quality of the data and the accuracy of the preprocessing steps are critical to the performance of the machine learning algorithm used to predict air quality.

## **MODULE 2: Model Training and Building**

After completing data collection and preprocessing in Module 1, the next step in air quality prediction is to train and build machine learning models. In this module, we will discuss the process of model training and building using four different algorithms: Logistic Regression, KNN Classifier, Decision Tree Classifier, and Random Forest Classifier.

### **Logistic Regression**

Logistic Regression is a statistical method used to predict a binary outcome (i.e., whether a data point belongs to a certain class or not). In air quality prediction, Logistic Regression can be used to predict whether a given combination of environmental factors will result in good or poor air quality. The model is trained using the preprocessed dataset that was generated in Module 1. The training data is used to estimate the parameters of the model, which can then be used to predict the air quality index for new data points.

### **KNN Classifier**

KNN (K-Nearest Neighbors) is a classification algorithm that determines the class of a new data point based on the classes of the k-nearest neighbors in the training

set. In air quality prediction, KNN can be used to predict the AQI for a new combination of environmental factors by finding the k-nearest neighbors in the training set and determining the average AQI for those neighbors. The model is trained using the preprocessed dataset generated in Module 1. The training data is used to determine the optimal value of k and to calculate the distances between the data points.

### **Decision Tree Classifier**

A Decision Tree is a graphical representation of all the possible outcomes of a series of decisions. In air quality prediction, Decision Tree can be used to predict the AQI for a new combination of environmental factors based on a set of decision rules. The model is trained using the preprocessed dataset generated in Module 1. The training data is used to construct the decision tree based on the features and the AQI values.

### **Random Forest Classifier**

Random Forest is an ensemble learning method that constructs multiple decision trees and combines their predictions. In air quality prediction, Random Forest can be used to predict the AQI for a new combination of environmental factors by combining the predictions of multiple decision trees. The model is trained using the preprocessed dataset generated in Module 1. The training data is used to construct multiple decision trees using different subsets of the features and the data points.

### **Model Evaluation**

Once the models are trained, the next step is to evaluate their performance. In air quality prediction, model evaluation involves comparing the predicted AQI values to the actual AQI values in the testing set. The evaluation metrics used to measure the

performance of the models include accuracy, precision, recall, F1 score, and ROC curve.

## **Conclusion**

In summary, Module 2 of air quality prediction involves training and building machine learning models using four different algorithms: Logistic Regression, KNN Classifier, Decision Tree Classifier, and Random Forest Classifier. The models are trained using the preprocessed dataset generated in Module 1, and their performance is evaluated using various metrics. The best-performing model can then be used for air quality prediction in Module 3.

## **MODULE 3: PREDICTION**

Module 3 is the final step in the air quality prediction process, which involves using the best-performing model to predict the AQI for new data points. The process of making predictions begins with collecting new data on the environmental factors that affect air quality. Once the new data has been collected, it is preprocessed using the same preprocessing steps used in Module 1. This includes data cleaning, feature selection, feature scaling, data splitting, and data encoding to ensure that the new data is suitable for machine learning analysis.

After the new data has been preprocessed, the best-performing model can be used to predict the AQI for the new data points. The predicted AQI values can then be used to determine whether the air quality is good or poor. It is important to note that the accuracy of the predictions depends on the quality of the data and the performance of the machine learning algorithm used to predict the AQI.

To deploy the best-performing model for real-time air quality prediction, the model needs to be integrated into a web application or other platform. The model can be used

to predict the AQI for new data points in real-time, allowing users to make informed decisions about their activities and health. The real-time air quality prediction can be particularly useful for people who suffer from respiratory problems or other health issues that are exacerbated by poor air quality.

It is important to continuously monitor and update the model to ensure that it remains accurate and effective. This can involve collecting new data and retraining the model periodically to incorporate the latest environmental factors and AQI values. Additionally, it is important to monitor the performance of the model and adjust it as necessary to improve its accuracy and effectiveness.

In conclusion, Module 3 is a critical step in the air quality prediction process as it involves using the best-performing model to make accurate predictions about air quality for new data points. By integrating the model into a web application or other platform, the predictions can be made in real-time, allowing users to make informed decisions about their activities and health. To ensure the accuracy and effectiveness of the model, it is important to continuously monitor and update it with the latest data and performance metrics.

### **5.3 ALGORITHMS**

1. Collect relevant air quality and weather data, such as pollutant levels, temperature, humidity, wind speed, and direction.
2. Clean and pre-process the data to remove any inconsistencies, errors, or missing values.
3. Split the data into training and testing sets for model development and evaluation.

4. Implement the random forest and decision tree algorithms for air quality prediction.
5. Set the hyperparameters for the algorithms, such as the number of trees, maximum depth, and minimum sample split.
6. Train the models using the training data and optimize the hyperparameters to improve the accuracy of predictions.
7. Evaluate the models using the testing data and performance metrics, such as accuracy, precision, recall, and F1 score.
8. Use the models to predict air quality levels for new data based on input variables, such as weather conditions and pollutant levels.
9. Handle missing data by imputing missing values using methods such as mean imputation or regression imputation.
10. Use feature selection methods such as information gain, chi-square test, or recursive feature elimination to identify the most important variables for air quality prediction.
11. Visualize the decision tree structure to gain insights into the factors that impact air quality, such as the most significant variables and their interactions.

Use the insights from the models and visualization to develop policies and interventions to improve air quality, such as reducing emissions from industrial sources, promoting public transportation, or implementing air quality alerts and advisories.

# **CHAPTER 6**

## **SYSTEM**

### **IMPLEMENTATION**

## CHAPTER 6

### SYSTEM IMPLEMENTATION

#### 6.1 CODING

```
## Importing necessary libraries
```

```
import pandas as pd
```

```
import numpy as np
```

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

```
import warnings
```

```
warnings.filterwarnings("ignore")
```

```
from sklearn.preprocessing import LabelEncoder
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.linear_model import LinearRegression
```

```
from sklearn.tree import DecisionTreeRegressor
```

```
from sklearn.ensemble import RandomForestRegressor
```

```
from sklearn import metrics
```

```
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
```

```
from sklearn.metrics import accuracy_score,confusion_matrix
```

```
!unzip "/content/AIR_Quality.zip"
```

```
df=pd.read_csv('/content/data/air_quality_new.csv',encoding='unicode_escape')
```

```
# Reading the dataset
```

```
## Data Understanding
```

```
df.head()
```

```
# Loading the dataset
```

```
df.shape
```

```
# As we can see that there are 4,35,742 rows and 13 columns in the dataset
```

```
df.info()
```

```
# Checking the over all information on the dataset.
```

```
df.isnull().sum()
```

```
# There are a lot of missing values present in the dataset
```

```
df.describe()
```

```
# Checking the descriptive stats of the numeric values present in the data like mean,  
standard deviation, min values and max value present in the data
```

```
df.nunique()
```

```
# These are all the unique values present in the dataframe
```

```
df.columns
```

```
# These are all the columns present in the dataset.
```

```
stn_code (station code)
```

```
sampling_date (date of sample collection)
```



state (Indian State)  
location (location of sample collection)  
agency  
type (type of area)  
so2 (sulphur dioxide concentration)  
no2 (nitrogen dioxide concentration)  
rspm (respirable suspended particulate matter concentration)  
spm (suspended particulate matter)  
location\_monitoring\_station  
pm2\_5 (particulate matter 2.5)  
date (date)

## ## Data Visualization

```
sns.pairplot(data=df)
```

```
df['state'].value_counts()
```

```
# Viewing the count of values present in the state column
```

```
plt.figure(figsize=(15, 6))
```

```
plt.xticks(rotation=90)
```

```
df.state.hist()
```

```
plt.xlabel('state')
```

```
plt.ylabel('Frequencies')
```

```
plt.plot()
```

```
# The visualization shows us the count of states present in the dataset.
```

```
df['type'].value_counts()
# Viewing the count of values present in the type column
```

```
plt.figure(figsize=(15, 6))
plt.xticks(rotation=90)
df.type.hist()
plt.xlabel('Type')
plt.ylabel('Frequencies')
plt.plot()
# The visualization shows us the count of Types present in the dataset.
```

```
df['agency'].value_counts()
# Viewing the counts of values present in the agency column
```

```
plt.figure(figsize=(15, 6))
plt.xticks(rotation=90)
df.agency.hist()
plt.xlabel('Agency')
plt.ylabel('Frequencies')
plt.plot()
# The visualization shows us the count of Agency present in the dataset.
```

```
plt.figure(figsize=(30, 10))
plt.xticks(rotation=90)
sns.barplot(x='state',y='so2',data=df);
# This visualization shows the name of the state having higher so2 levels in the air
which is Uttaranchal followed by Uttarakhand
```

```
plt.rcParams['figure.figsize']=(30,10)
```

```
df[['so2','state']].groupby(["state"]).mean().sort_values(by='so2').plot.bar(color='purple')
```

```
plt.show()
```

# We can also use the groupby function to sort values in an ascending order based on the x-axis, y-axis and its keys

# Below we get a clear picture of the states in an increasing order based on their so2 levels.

```
plt.figure(figsize=(30, 10))
```

```
plt.xticks(rotation=90)
```

```
sns.barplot(x='state',y='no2',data=df);
```

# West bengal has a higher no2 level compared to other states

```
df[['no2','state']].groupby(["state"]).mean().sort_values(by='no2').plot.bar(color='purple')
```

```
plt.show()
```

# We can also use the groupby function to sort values in an ascending order based on the x-axis, y-axis and its keys

# Below we get a clear picture of the states in an increasing order based on their no2 levels.

```
plt.figure(figsize=(30, 10))
```

```
plt.xticks(rotation=90)
```

```
sns.barplot(x='state',y='rspm',data=df);
```

```
# Delhi has higher rspm level compared to other states
```

```
plt.figure(figsize=(30, 10))
```

```
plt.xticks(rotation=90)
```

```
sns.barplot(x='state',y='spm',data=df);
```

```
# Delhi has higher spm level compared to other states
```

```
plt.figure(figsize=(30, 10))
```

```
plt.xticks(rotation=90)
```

```
sns.barplot(x='state',y='pm2_5',data=df);
```

```
# Delhi has higher pm2_5 level compared to other states
```

```
### Checking all null values and treating those null values.
```

```
nullvalues = df.isnull().sum().sort_values(ascending=False)
```

```
# Checking all null values
```

```
nullvalues
```

```
# higher null values present in pm2_5 followed by spm
```

```
null_values_percentage
```

```
=
```

```
(df.isnull().sum()/df.isnull().count()*100).sort_values(ascending=False)
```

```
#count(returns Non-NAN value)
```

```
missing_data_with_percentage = pd.concat([nullvalues, null_values_percentage],  
axis=1, keys=['Total', 'Percent'])
```

# Concatenating total null values and their percentage of missing values for further imputation or column deletion

missing\_data\_with\_percentage

# As you can see below these are the percentages of null values present in the dataset

```
df.drop(['agency'],axis=1,inplace=True)
df.drop(['stn_code'],axis=1,inplace=True)
df.drop(['date'],axis=1,inplace=True)
df.drop(['sampling_date'],axis=1,inplace=True)
df.drop(['location_monitoring_station'],axis=1,inplace=True)
# Dropping unnecessary columns
```

```
df.isnull().sum()
# Now checking the null values
```

df

```
df['location']=df['location'].fillna(df['location'].mode()[0])
df['type']=df['type'].fillna(df['type'].mode()[0])
# Null value Imputation for categorical data
```

```
df.fillna(0, inplace=True)
# null values are replaced with zeros for the numerical data
```

```
df.isnull().sum()
# Now we have successfully imputed null values which were present in the dataset
```

```
df
```

```
# The following features are important for our machine learning models.
```

```
# CALCULATE AIR QUALITY INDEX FOR SO2 BASED ON FORMULA
```

The air quality index is a piecewise linear function of the pollutant concentration. At the boundary between AQI categories, there is a discontinuous jump of one AQI unit. To convert from concentration to AQI this equation is used

```
### Function to calculate so2 individual pollutant index(si)
```

```
def cal_SOi(so2):
```

```
    si=0
```

```
    if (so2<=40):
```

```
        si= so2*(50/40)
```

```
    elif (so2>40 and so2<=80):
```

```
        si= 50+(so2-40)*(50/40)
```

```
    elif (so2>80 and so2<=380):
```

```
        si= 100+(so2-80)*(100/300)
```

```
    elif (so2>380 and so2<=800):
```

```
        si= 200+(so2-380)*(100/420)
```

```
    elif (so2>800 and so2<=1600):
```

```
        si= 300+(so2-800)*(100/800)
```

```
    elif (so2>1600):
```

```
        si= 400+(so2-1600)*(100/800)
```

```
    return si
```

```
df['SOi']=df['so2'].apply(cal_SOi)
```

```

data= df[['so2','SOi']]
data.head()
# calculating the individual pollutant index for so2(sulphur dioxide)

### Function to calculate no2 individual pollutant index(ni)

def cal_Noi(no2):
    ni=0
    if(no2<=40):
        ni= no2*50/40
    elif(no2>40 and no2<=80):
        ni= 50+(no2-40)*(50/40)
    elif(no2>80 and no2<=180):
        ni= 100+(no2-80)*(100/100)
    elif(no2>180 and no2<=280):
        ni= 200+(no2-180)*(100/100)
    elif(no2>280 and no2<=400):
        ni= 300+(no2-280)*(100/120)
    else:
        ni= 400+(no2-400)*(100/120)
    return ni
df['Noi']=df['no2'].apply(cal_Noi)
data= df[['no2','Noi']]
data.head()
# calculating the individual pollutant index for no2(nitrogen dioxide)

### Function to calculate rspm individual pollutant index(rpi)

```

```

def cal_RSPMI(rspm):
    rpi=0
    if(rpi<=30):
        rpi=rpi*50/30
    elif(rpi>30 and rpi<=60):
        rpi=50+(rpi-30)*50/30
    elif(rpi>60 and rpi<=90):
        rpi=100+(rpi-60)*100/30
    elif(rpi>90 and rpi<=120):
        rpi=200+(rpi-90)*100/30
    elif(rpi>120 and rpi<=250):
        rpi=300+(rpi-120)*(100/130)
    else:
        rpi=400+(rpi-250)*(100/130)
    return rpi
df['Rpi']=df['rspm'].apply(cal_RSPMI)
data= df[['rspm','Rpi']]
data.head()

# calculating the individual pollutant index for rspm(respirable suspended particulate
matter concentration)

### Function to calculate spm individual pollutant index(spi)

def cal_SPMi(spm):
    spi=0
    if(spm<=50):

```



```

spi=spm*50/50
elif(spm>50 and spm<=100):
    spi=50+(spm-50)*(50/50)
elif(spm>100 and spm<=250):
    spi= 100+(spm-100)*(100/150)
elif(spm>250 and spm<=350):
    spi=200+(spm-250)*(100/100)
elif(spm>350 and spm<=430):
    spi=300+(spm-350)*(100/80)
else:
    spi=400+(spm-430)*(100/430)
return spi

```

```
df['SPMi']=df['spm'].apply(cal_SPMi)
```

```
data= df[['spm','SPMi']]
```

```
data.head()
```

```
# calculating the individual pollutant index for spm(suspended particulate matter)
```

```
### function to calculate the air quality index (AQI) of every data value
```

```

def cal_aqi(si,ni,rspmi,spmi):
    aqi=0
    if(si>ni and si>rspmi and si>spmi):
        aqi=si
    if(ni>si and ni>rspmi and ni>spmi):
        aqi=ni
    if(rspmi>si and rspmi>ni and rspmi>spmi):

```

```

aqi=rspmi
if(spmi>si and spmi>ni and spmi>rspmi):
    aqi=spmi
return aqi

```

```

df['AQI']=df.apply(lambda x:cal_aqi(x['SOi'],x['Noi'],x['Rpi'],x['SPMi']),axis=1)
data= df[['state','SOi','Noi','Rpi','SPMi','AQI']]
data.head()
# Caluclating the Air Quality Index.

```

```

def AQI_Range(x):
    if x<=50:
        return "Good"
    elif x>50 and x<=100:
        return "Moderate"
    elif x>100 and x<=200:
        return "Poor"
    elif x>200 and x<=300:
        return "Unhealthy"
    elif x>300 and x<=400:
        return "Very unhealthy"
    elif x>400:
        return "Hazardous"

```

```

df['AQI_Range'] = df['AQI'] .apply(AQI_Range)
df.head()

```

```
# Using threshold values to classify a particular values as good, moderate, poor,
unhealthy, very unhealthy and Hazardous
```

```
df.to_csv("AIR_QUALITY.csv")
```

```
df.describe()
```

```
df['AQI_Range'].value_counts()
```

```
# These are the counts of values present in the AQI_Range column.
```

```
### Splitting the dataset into Dependent and Independent columns
```

```
X=df[['SOi','Noi','Rpi','SPMi']]
```

```
Y=df['AQI']
```

```
X.head()
```

```
# we only select columns like soi, noi, rpi, spmi
```

```
Y.head()
```

```
# the AQI column is the target column
```

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=70)
```

```
print(X_train.shape,X_test.shape,Y_train.shape,Y_test.shape)
```

```
# splitting the data into training and testing data
```

```
### Linear Regression
```

```
model=LinearRegression()
```

```

model.fit(X_train,Y_train)

#predicting train
train_pred=model.predict(X_train)
#predicting on test
test_pred=model.predict(X_test)

RMSE_train=(np.sqrt(metrics.mean_squared_error(Y_train,train_pred)))
RMSE_test=(np.sqrt(metrics.mean_squared_error(Y_test,test_pred)))
print("RMSE TrainingData = ",str(RMSE_train))
print("RMSE TestData = ",str(RMSE_test))
print('-'*50)
print('RSquared value on train:',model.score(X_train, Y_train))
print('RSquared value on test:',model.score(X_test, Y_test))

### Decision Tree Regressor

DT=DecisionTreeRegressor()
DT.fit(X_train,Y_train)

#predicting train
train_preds=DT.predict(X_train)
#predicting on test
test_preds=DT.predict(X_test)

RMSE_train=(np.sqrt(metrics.mean_squared_error(Y_train,train_preds)))
RMSE_test=(np.sqrt(metrics.mean_squared_error(Y_test,test_preds)))

```

```

print("RMSE TrainingData = ",str(RMSE_train))
print("RMSE TestData = ",str(RMSE_test))
print('-'*50)
print('RSquared value on train:',DT.score(X_train, Y_train))
print('RSquared value on test:',DT.score(X_test, Y_test))

```

### Random Forest Regressor

```

RF=RandomForestRegressor().fit(X_train,Y_train)

```

```

#predicting train

```

```

train_preds1=RF.predict(X_train)

```

```

#predicting on test

```

```

test_preds1=RF.predict(X_test)

```

```

RMSE_train=(np.sqrt(metrics.mean_squared_error(Y_train,train_preds1)))

```

```

RMSE_test=(np.sqrt(metrics.mean_squared_error(Y_test,test_preds1)))

```

```

print("RMSE TrainingData = ",str(RMSE_train))

```

```

print("RMSE TestData = ",str(RMSE_test))

```

```

print('-'*50)

```

```

print('RSquared value on train:',RF.score(X_train, Y_train))

```

```

print('RSquared value on test:',RF.score(X_test, Y_test))

```

# Classification Algorithms

```

from sklearn.linear_model import LogisticRegression

```

```

from sklearn.tree import DecisionTreeClassifier

```

```

from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier

X2 = df[['SOi','Noi','Rpi','SPMi']]
Y2 = df['AQI_Range']
# Splitting the data into independent and dependent columns for classification

X_train2, X_test2, Y_train2, Y_test2 = train_test_split(X2, Y2, test_size=0.33,
random_state=70)
# Splitting the data into training and testing data

### Logistic Regression

#fit the model on train data
log_reg = LogisticRegression().fit(X_train2, Y_train2)

#predict on train
train_preds2 = log_reg.predict(X_train2)
#accuracy on train
print("Model accuracy on train is: ", accuracy_score(Y_train2, train_preds2))

#predict on test
test_preds2 = log_reg.predict(X_test2)
#accuracy on test
print("Model accuracy on test is: ", accuracy_score(Y_test2, test_preds2))
print('-'*50)

```

```

# Kappa Score.
print('KappaScore is: ', metrics.cohen_kappa_score(Y_test2,test_preds2))

log_reg.predict([[727,327.55,78.2,100]])

log_reg.predict([[2.7,45,35.16,23]])

log_reg.predict([[10,2.8,82,20]])

log_reg.predict([[2,45.8,37,32]])

### Decision Tree Classifier

#fit the model on train data
DT2 = DecisionTreeClassifier().fit(X_train2,Y_train2)

#predict on train
train_preds3 = DT2.predict(X_train2)
#accuracy on train
print("Model accuracy on train is: ", accuracy_score(Y_train2, train_preds3))

#predict on test
test_preds3 = DT2.predict(X_test2)
#accuracy on test
print("Model accuracy on test is: ", accuracy_score(Y_test2, test_preds3))
print('-'*50)

```

```

# Kappa Score
print('KappaScore is: ', metrics.cohen_kappa_score(Y_test2,test_preds3))

### Random Forest Classifier

#fit the model on train data
RF=RandomForestClassifier().fit(X_train2,Y_train2)
#predict on train
train_preds4 = RF.predict(X_train2)
#accuracy on train
print("Model accuracy on train is: ", accuracy_score(Y_train2, train_preds4))

#predict on test
test_preds4 = RF.predict(X_test2)
#accuracy on test
print("Model accuracy on test is: ", accuracy_score(Y_test2, test_preds4))
print('-'*50)

# Kappa Score
print('KappaScore is: ', metrics.cohen_kappa_score(Y_test2,test_preds4))

### K-Nearest Neighbours

#fit the model on train data
KNN = KNeighborsClassifier().fit(X_train2,Y_train2)
#predict on train
train_preds5 = KNN.predict(X_train2)

```



```

#accuracy on train
print("Model accuracy on train is: ", accuracy_score(Y_train2, train_preds5))

#predict on test
test_preds5 = KNN.predict(X_test2)
#accuracy on test
print("Model accuracy on test is: ", accuracy_score(Y_test2, test_preds5))
print('-'*50)

# Kappa Score
print('KappaScore is: ', metrics.cohen_kappa_score(Y_test2,test_preds5))

KNN.predict([[7.4,47.7,78.182,100]])
# Predictions on random values

KNN.predict([[1,1.2,3.12,0]])
# Predictions on random values

KNN.predict([[325.7,345,798.182,203]])
# Predictions on random values

from sklearn.preprocessing import StandardScaler
scale=StandardScaler()

l=[]
x=float(input("Enter the SOi : "))
l.append(x)

```

```
x=float(input("Enter the Noi :"))
l.append(x)
x=float(input("Enter Rpi value : "))
l.append(x)
x=float(input("Enter the SPMi : "))
l.append(x)
features=np.array(l)

scale.fit(features)

predict=DT2.predict(features)

print(predict)
```

# **CHAPTER 7**

## **SYSTEM TESTING**

## CHAPTER 7

### SYSTEM TESTING

Discovering and fixing such problems is what testing is all about. The purpose of testing is to find and correct any problems with the final product. It's a method for evaluating the quality of the operation of anything from a whole product to a single component. The goal of stress testing software is to verify that it retains its original functionality under extreme circumstances. There are several different tests from which to pick. Many tests are available since there is such a vast range of assessment options.

**Who Performs the Testing:** All individuals who play an integral role in the software development process are responsible for performing the testing. Testing the software is the responsibility of a wide variety of specialists, including the End Users, Project Manager, Software Tester, and Software Developer.

**When it is recommended that testing begin:** Testing the software is the initial step in the process. begins with the phase of requirement collecting, also known as the Planning phase, and ends with the stage known as the Deployment phase. In the waterfall model, the phase of testing is where testing is explicitly arranged and carried out. Testing in the incremental model is carried out at the conclusion of each increment or iteration, and the entire application is examined in the final test.

**When it is appropriate to halt testing:** Testing the programme is an ongoing activity that will never end. Without first putting the software through its paces, it is impossible for anyone to guarantee that it is completely devoid of errors. Because the

domain to which the input belongs is so expansive, we are unable to check every single input.

## **7.1 Unit Testing**

The term "unit testing" refers to a specific kind of software testing in which discrete elements of a program are investigated. The purpose of this testing is to ensure that the software operates as expected.

### **Testcases**

- 1. Test case for data pre-processing:** This test case should test the pre-processing of the data used for air quality prediction. The test should ensure that the data is properly normalized, filtered, and scaled for use by the prediction model.
- 2. Test case for model creation:** This test case should test the creation of the prediction model used for air quality prediction. The test should ensure that the model is properly constructed, with appropriate layers and activation functions, and that the hyper parameters have been tuned correctly.
- 3. Test case for model training:** This test case should test the training of the prediction model using known data. The test should ensure that the model can accurately predict air quality for the training data.

## **7.2 Integration Testing**

The programme is put through its paces in its final form, once all its parts have been combined, during the integration testing phase. At this phase, we look for places where interactions between components might cause problems.

## Test Cases

- 1. Test case for data integration:** This test case should test the integration of the data used for air quality prediction. The test should ensure that data from multiple sources can be integrated seamlessly and used to predict air quality accurately.
- 2. Test case for model integration:** This test case should test the integration of the prediction model with other components of the system. The test should ensure that the model can be integrated with other modules, such as user interfaces and data storage components.
- 3. Test case for input/output integration:** This test case should test the integration of the input and output components of the system. The test should ensure that input data can be properly processed and used by the model, and that the predicted output data can be displayed to the user in a clear and understandable format.

## 7.3 Functional Testing

One kind of software testing is called functional testing, and it involves comparing the system to the functional requirements and specifications. In order to test functions, their input must first be provided, and then the output must be examined. Functional testing verifies that an application successfully satisfies all of its requirements in the correct manner. This particular kind of testing is not concerned with the manner in which processing takes place; rather, it focuses on the outcomes of processing. Therefore, it endeavors to carry out the test cases, compare the outcomes, and validate the correctness of the results.

## Test Cases

- 1. Test case for input validation:** This test case should test the system's ability to validate input data and handle invalid input data appropriately. The test should ensure

that the system can handle different types of input data and provide clear error messages to users when input data is invalid.

**2. Test case for accuracy testing:** This test case should test the accuracy of the air quality prediction system by comparing the predicted values with actual values from validated sources. The test should ensure that the system accurately predicts air quality within a certain range of error.

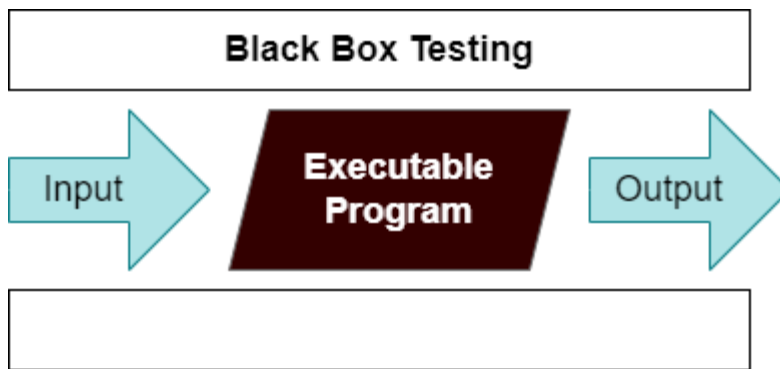
**3. Test case for scalability testing:** This test case should test the system's ability to scale up and down depending on the volume of data and user traffic. The test should ensure that the system can handle high levels of traffic and large amounts of data without affecting its accuracy or performance.

## **7.4 TESTING TECHNIQUES**

There are many different techniques or methods for testing the software, including the following:

### **7.4.1 BLACK BOX TESTING**

During this kind of testing, the user does not have access to or knowledge of the internal structure or specifics of the data item being tested. In this method, test cases are generated or designed only based on the input and output values, and prior knowledge of either the design or the code is not necessary. The testers are just conscious of knowing about what is thought to be able to do, but they do not know how it is able to do it.



**Fig 7.4.1** Black Box Testing

For example, without having any knowledge of the inner workings of the website, we test the web pages by using a browser, then we authorize the input, and last, we test and validate the outputs against the intended result.

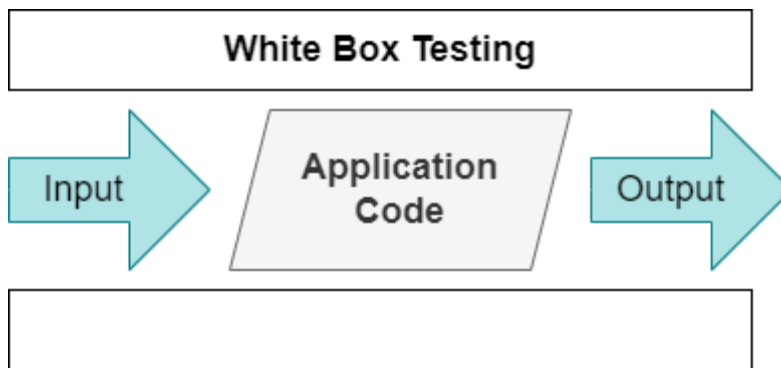
### **Test Cases**

- 1. Test case for input validation:** Verify that the system rejects invalid input such as negative values or values that fall outside of expected ranges for environmental factors such as temperature, humidity, wind speed, and pressure.
- 2. Test case for model accuracy:** Verify that the system produces accurate predictions of air pollutant concentrations for a given set of environmental factors. This can be achieved by comparing the predicted values to actual air pollutant concentrations measured at the same location and time.
- 3. Test case for model robustness:** Verify that the system produces accurate predictions even when environmental factors are missing or incomplete. This can be achieved by providing incomplete or missing data to the system and verifying that it can still produce accurate predictions.



## 7.4.2 WHITE BOX TESTING

During this kind of testing, the user is aware of the internal structure and details of the data item, or they have access to such information. In this process, test cases are constructed by referring to the code. Programming is extremely knowledgeable of the manner in which the application of knowledge is significant. White Box Testing is so called because, as we all know, in the tester's eyes it appears to be a white box, and on the inside, everyone can see clearly. This is how the testing got its name.



**Fig 7.4.2** White Box Testing

As an instance, a tester and a developer examine the code that is implemented in each field of a website, determine which inputs are acceptable and which are not, and then check the output to ensure it produces the desired result. In addition, the decision is reached by analyzing the code that is really used.

### Test Cases

**1. Test case for code coverage:** This test case should ensure that all the code written for the air quality prediction system is covered by test cases. The test should ensure that each line of code is executed and tested for accuracy and correctness.

**2. Test case for unit testing:** This test case should test individual units of code, such as functions and methods, to ensure that they perform as expected. The test should ensure that each unit is tested thoroughly and that all possible edge cases are covered.

**3. Test case for integration testing:** This test case should test the integration of different modules and components of the air quality prediction system. The test should ensure that each module is integrated correctly and that the system as a whole performs as expected.

# **CHAPTER 8**

## **CONCLUSION AND FUTURE ENHANCEMENTS**

## **CHAPTER 8**

### **CONCLUSION AND FUTURE ENHANCEMENTS**

#### **8.1 RESULTS & DISCUSSION**

Air quality prediction using machine learning is a widely researched area due to its potential for mitigating the health and environmental effects of air pollution. Several machine learning models have been used to predict air quality, including neural networks, decision trees, and support vector machines. These models use a range of input variables, such as meteorological data, traffic data, and emission data, to predict pollutant concentrations in the atmosphere. Studies have shown that machine learning models can accurately predict air quality, with some models achieving up to 90% accuracy. However, the performance of these models is highly dependent on the quality and quantity of the input data. Models trained on incomplete or low-quality data may produce inaccurate predictions. In addition, the interpretability of machine learning models remains a challenge in air quality prediction. As machine learning models are often regarded as black boxes, it can be difficult to understand how the models arrive at their predictions. This lack of transparency can make it challenging to identify the causes of air pollution and develop effective mitigation strategies. Overall, air quality prediction using machine learning has shown promising results, but further research is needed to improve the accuracy and interpretability of these models. By combining machine learning with traditional air quality monitoring techniques, it may be possible to better understand the sources and impacts of air pollution and develop effective strategies for reducing its effects on human health and the environment.

## 8.2 CONCLUSION AND FUTURE ENHANCEMENTS

In conclusion, air quality prediction using machine learning has shown potential for accurately predicting air pollution concentrations. However, the performance of these models is highly dependent on the quality and quantity of the input data, and the interpretability of these models remains a challenge. Therefore, further research is needed to enhance the accuracy and interpretability of machine learning models for air quality prediction. Future research could focus on improving the quality and quantity of input data used for air quality prediction. This could involve incorporating more sources of data, such as satellite imagery or data from low-cost air quality sensors. Additionally, research could focus on developing methods to account for uncertainty in input data, which could improve the accuracy of machine learning models. Another important area for future research is improving the interpretability of machine learning models. This could involve developing methods for explaining the predictions made by machine learning models, such as feature importance analysis or local interpretability methods. Finally, machine learning models for air quality prediction could be integrated with traditional air quality monitoring techniques to provide a more comprehensive understanding of air pollution. This could involve combining machine learning models with ground-based air quality monitoring stations or integrating them with mobile air quality monitoring platforms, such as drones or vehicles. Overall, air quality prediction using machine learning has shown promising results, and further research could lead to improved prediction accuracy and more effective strategies for mitigating the health and environmental effects of air pollution.

## APPENDICES

### A.1 Sample Screens

```
C:\Users\Karthik\Desktop\Air Quality\Air Quality Pred\Air_Quality>python AQ_app.py
2023-04-09 07:59:25.164
Warning: to view this Streamlit app on a browser, run it with the following
command:

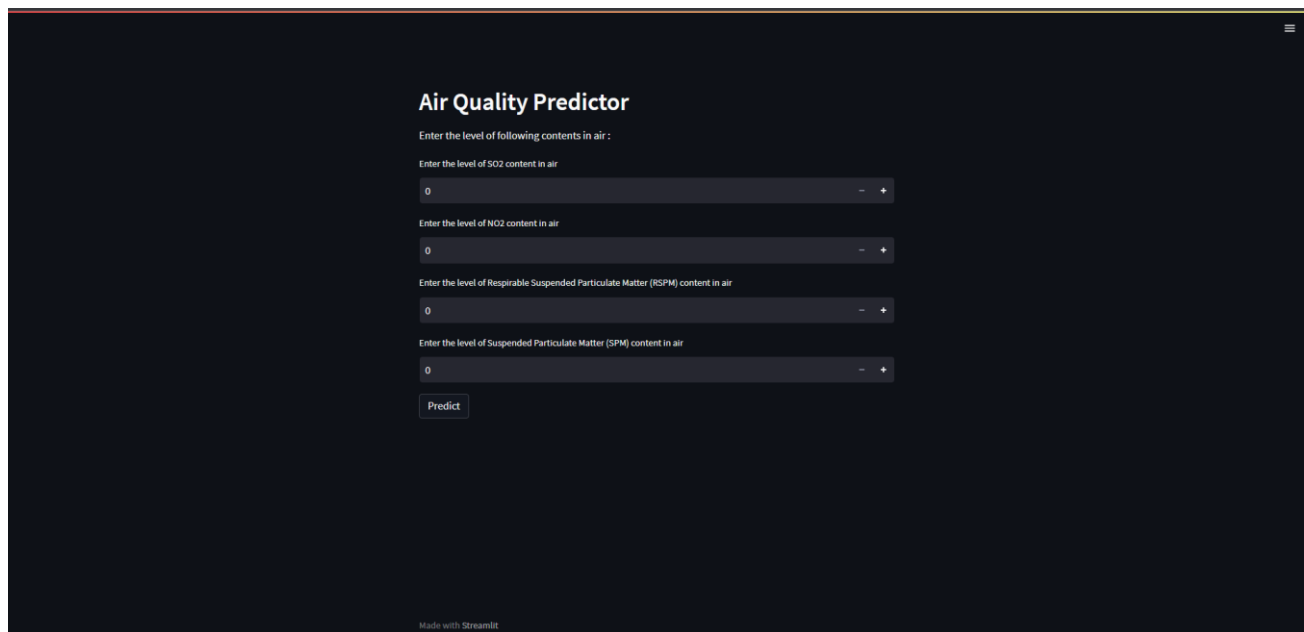
    streamlit run AQ_app.py [ARGUMENTS]
C:\Users\Karthik\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\base.py:409: UserWarning: X does not have valid feature names, but DecisionTreeRegressor was fitted with feature names
  warnings.warn(
C:\Users\Karthik\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\base.py:409: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
C:\Users\Karthik\Desktop\Air Quality\Air Quality Pred\Air_Quality>streamlit run AQ_app.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.0.107:8501
```

**Fig A.1.1** The Command Prompt Screen

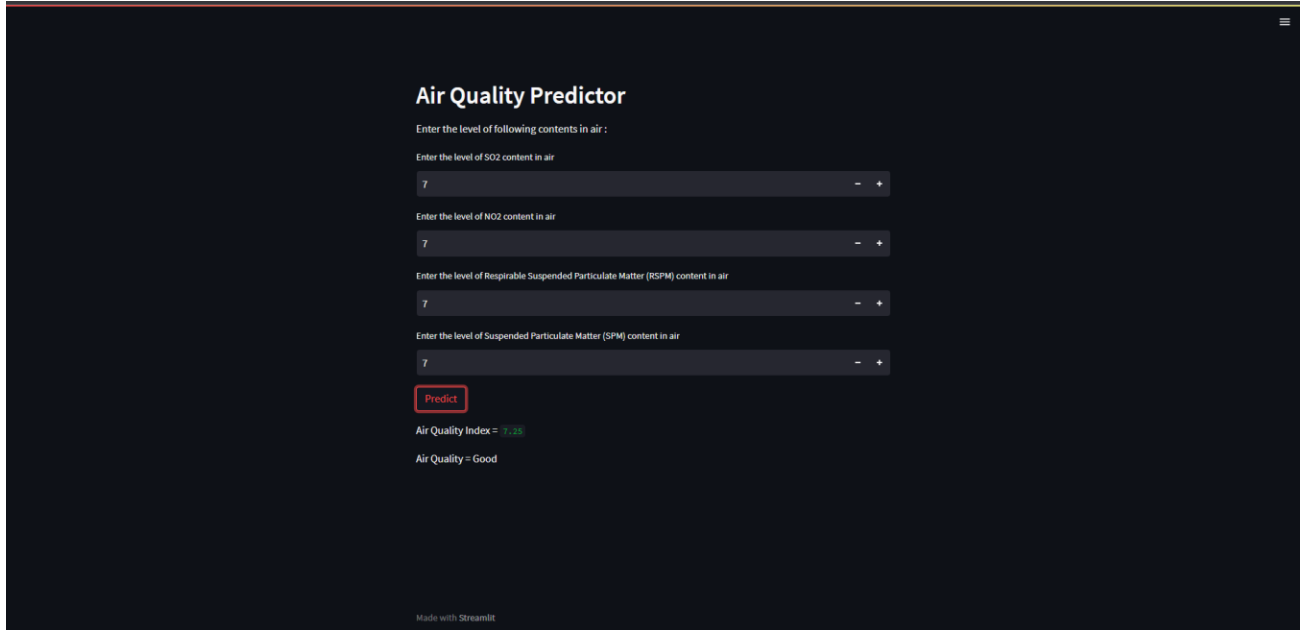
Figure A.1.1 shows the command prompt environment to deploy the Streamlit which enables a local host connection.



The screenshot shows the home page of the 'Air Quality Predictor' application. The page has a dark background with white text. At the top, the title 'Air Quality Predictor' is displayed. Below it, a subtitle reads 'Enter the level of following contents in air :'. There are four input fields, each with a label and a numeric value '0' inside a dark box with a minus and plus button on the right. The labels are: 'Enter the level of SO2 content in air', 'Enter the level of NO2 content in air', 'Enter the level of Respirable Suspended Particulate Matter (RSPM) content in air', and 'Enter the level of Suspended Particulate Matter (SPM) content in air'. Below the input fields is a 'Predict' button. At the bottom left, it says 'Made with Streamlit'.

**Fig A.1.2** The Home Page

Figure A.1.2 shows the home page layout of the Air Quality Prediction Model



The screenshot displays the 'Air Quality Predictor' web application. It features a dark blue background with white text. The title 'Air Quality Predictor' is at the top. Below it, a prompt says 'Enter the level of following contents in air :'. There are four input fields, each with a value of '7' and a range indicator '- +'. The inputs are for SO2, NO2, Respirable Suspended Particulate Matter (RSPM), and Suspended Particulate Matter (SPM). A red 'Predict' button is below the inputs. The output shows 'Air Quality Index = 7.25' in green and 'Air Quality = Good' in white. At the bottom, it says 'Made with Streamlit'.

**Air Quality Predictor**

Enter the level of following contents in air :

Enter the level of SO2 content in air

7 - +

Enter the level of NO2 content in air

7 - +

Enter the level of Respirable Suspended Particulate Matter (RSPM) content in air

7 - +

Enter the level of Suspended Particulate Matter (SPM) content in air

7 - +

**Predict**

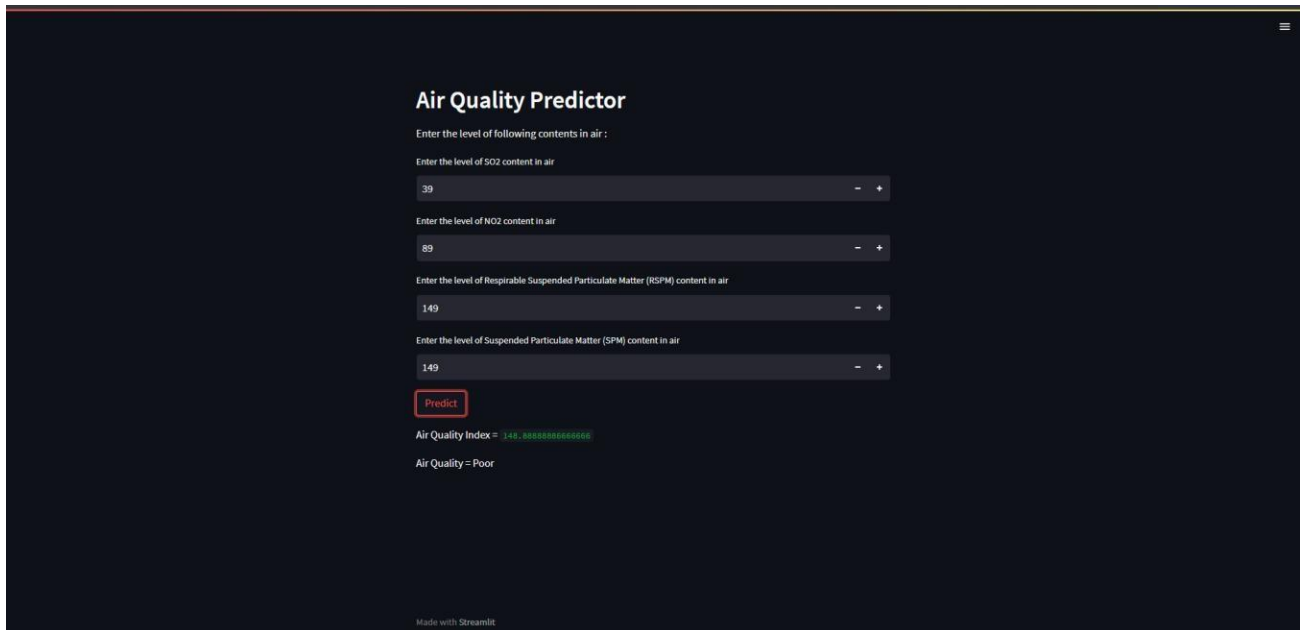
Air Quality Index = 7.25

Air Quality = Good

Made with Streamlit

**Fig A.1.3** The sample input 1

Figure A.1.3 shows the Air Quality is healthy and good to use.



The screenshot displays the 'Air Quality Predictor' web application with higher input values. The title 'Air Quality Predictor' is at the top. Below it, a prompt says 'Enter the level of following contents in air :'. There are four input fields, each with a value of 39, 89, 149, and 149 respectively, and a range indicator '- +'. The inputs are for SO2, NO2, Respirable Suspended Particulate Matter (RSPM), and Suspended Particulate Matter (SPM). A red 'Predict' button is below the inputs. The output shows 'Air Quality Index = 348.8888888888889' in green and 'Air Quality = Poor' in white. At the bottom, it says 'Made with Streamlit'.

**Air Quality Predictor**

Enter the level of following contents in air :

Enter the level of SO2 content in air

39 - +

Enter the level of NO2 content in air

89 - +

Enter the level of Respirable Suspended Particulate Matter (RSPM) content in air

149 - +

Enter the level of Suspended Particulate Matter (SPM) content in air

149 - +

**Predict**

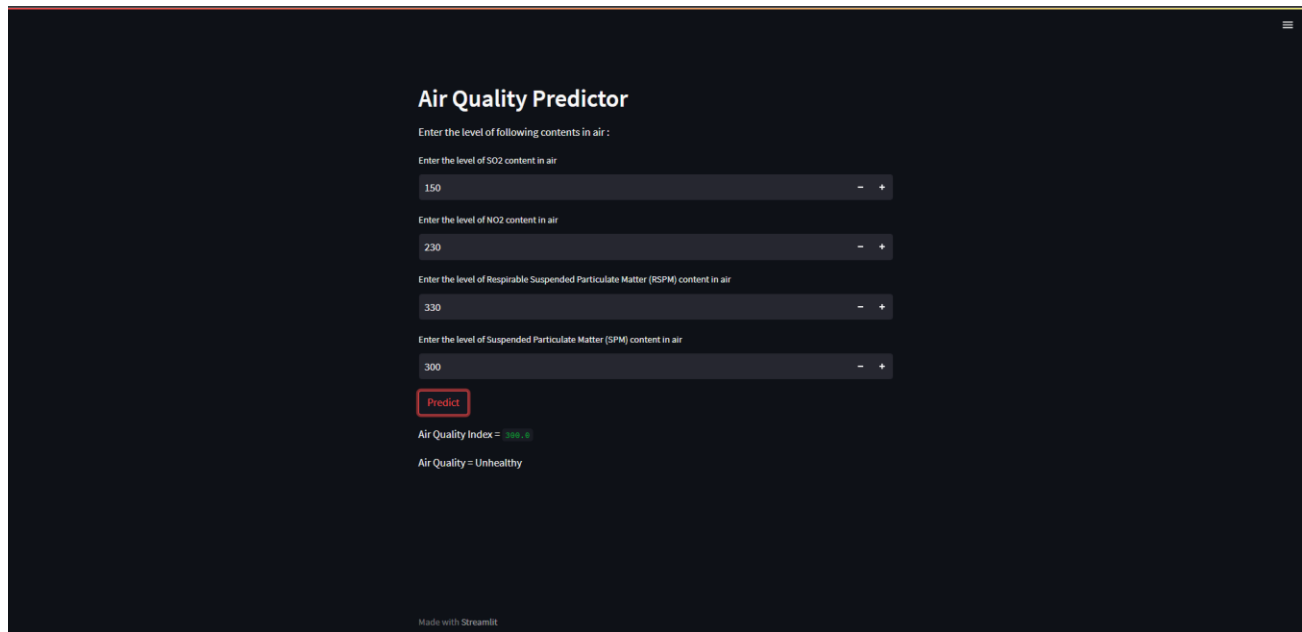
Air Quality Index = 348.8888888888889

Air Quality = Poor

Made with Streamlit

**Fig A.1.4** The sample input 2

Figure A.1.4 shows the Air Quality is moderate and poor to use.



The screenshot displays a web application titled "Air Quality Predictor" with a dark background. It features four input fields for air pollutant levels, each with a minus and plus button for adjustment. The inputs are: SO2 at 150, NO2 at 230, Respirable Suspended Particulate Matter (RSPM) at 330, and Suspended Particulate Matter (SPM) at 300. A red-outlined "Predict" button is located below the inputs. The output section shows the "Air Quality Index = 354.6" in green text and "Air Quality = Unhealthy" in white text. At the bottom, it says "Made with Streamlit".

Pollutant	Level
SO2	150
NO2	230
RSPM	330
SPM	300

**Predict**

Air Quality Index = 354.6

Air Quality = Unhealthy

Made with Streamlit

**Fig A.1.5** The sample input 3

Figure A.1.5 shows the Air Quality is unhealthy and unfit to use.



## REFERENCES

- [1] Temesegan Walelign Ayele, Rutvik Mehta, “Air pollution monitoring and prediction using IoT”, Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 2018
- [2] Saba Ameer, Munam Ali Shah, Abid Khan, Houbing Song, Carsten Maple, Saif Ul Islam, Muhammad Nabeel Asghar, “Comparative Analysis of Machine Learning Techniques for Predicting Air Quality in Smart Cities”, IEEE Access (Volume: 7), 2019
- [3] Yi-Ting Tsai, Yu-Ren Zeng, Yue-Shan Chang, “Air Pollution Forecasting Using RNN with LSTM”, IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress, 2018
- [4] Venkat Rao Pasupuleti, Uhasri, Pavan Kalyan, Srikanth, Hari Kiran Reddy, “Air Quality Prediction Of Data Log By Machine Learning”, 6th International Conference on Advanced Computing and Communication Systems (ICACCS), 2020
- [5] Shengdong Du, Tianrui Li, Yan Yang, Shi-Jinn Horng, “Deep Air Quality Forecasting Using Hybrid Deep Learning Framework”, Transactions on Knowledge and Data Engineering (Volume: 33, Issue: 6), 2021
- [6] Ke Gu, Junfei Qiao, Weisi Lin, “Recurrent Air Quality Predictor Based on Meteorology- and Pollution-Related Factors”, IEEE Transactions on Industrial Informatics (Volume: 14, Issue: 9), 2018
- [7] Bo Liu, Shuo Yan, Jianqiang Li, Guangzhi Qu, Yong Li, Jianlei Lang, Rentao Gu, “A Sequence-to-Sequence Air Quality Predictor Based on the n-Step Recurrent Prediction”, IEEE Access (Volume: 7), 2019

- [8] Baowei Wang, Weiwen Kong, Hui Guan, Neal N. Xiong, “Air Quality Forecasting Based on Gated Recurrent Long Short-Term Memory Model in Internet of Things”, IEEE Access (Volume: 7), 2019
- [9] Yuanni Wang, Tao Kong, “Air Quality Predictive Modeling Based on an Improved Decision Tree in a Weather-Smart Grid”, IEEE Access (Volume: 7), 2019
- [10] Van-Duc Le, Tien-Cuong Bui, Sang-Kyun Cha, “Spatiotemporal Deep Learning Model for Citywide Air Pollution Interpolation and Prediction”, IEEE International Conference on Big Data and Smart Computing (BigComp), 2020