## Analysis of Air Quality Index Prediction using Machine learning and Deep Learning Techniques

*Report submitted to the SASTRA Deemed to be University as the requirement for the course*

**CSE499 – PROJECT PHASE-II**

*Submitted by*

**Sivaganesh S**

**(Reg. No.: 125018063, B.Tech CSBS)**

**Pothireddy Chandrahas Reddy (Reg. No.: 125018015, B.Tech CSBS)**

**Puluru Venkata Gopinadha Reddy (Reg. No.: 125018022, B.Tech CSBS)**

May 2025

****

# SCHOOL OF COMPUTING THANJAVUR, TAMIL NADU, INDIA – 613 401.



**SCHOOL OF COMPUTING THANJAVUR – 613 401**

## Bonafide Certificate

This is to certify that the project report titled “**Analysis of Air Quality Index Prediction using Machine learning and Deep Learning Techniques**” submitted in partial fulfillment of the requirements for the award of the degree of **B. Tech. Computer Science and Business Systems** to the SASTRA Deemed to be University, is a bonafide record of the work done by **Mr.Sivaganesh S(Reg. No: 125018063), Mr.Pothireddy Chandrahas Reddy(Reg.No: 125018015), Mr.Puluru Venkata Gopinadha Reddy(Reg.No: 125018022)** during the final semester of the academic year 2024- 25, in the **School of Computing**, under my supervision. This report has not formed the basis for the award of any degree, diploma, associateship, fellowship or other similar title to any candidate of any University.

#### Signature of Project Supervisor : Name with Affiliation : Dr.



V S Shankar Sriram,

Dean -SoC

**Date :** 05-05-2025

Major Project Viva voce held on

**Examiner 1 Examiner 2**



# SCHOOL OF COMPUTING THANJAVUR – 613 401

**Declaration**

We declare that the project report titled “**Analysis of Air Quality Index Prediction using Machine learning and Deep Learning Techniques**” submitted by us is an original work done by us under the guidance of **Dr. V. S. Shankar Sriram, Dean, School of Computing, SASTRA Deemed to be University** during the final semester of the academic year 2024-25, in the **School of Computing**. The work is original and wherever we have used materials from other sources, we have given due credit and cited them in the text of the report. This report has not formed the basis for the award of any degree, diploma, associate-ship, fellowship or other similar title to any candidate of any University.

#### C:\Users\ambra\AppData\Local\Packages\5319275A.WhatsAppDesktop_cv1g1gvanyjgm\TempState\0BC62AFCE4C0004839D37C08C5F0528A\WhatsApp Image 2025-05-06 at 09.52.58_67ba89f8.jpgC:\Users\ambra\AppData\Local\Packages\5319275A.WhatsAppDesktop_cv1g1gvanyjgm\TempState\1C04ABEB85A834A19E1ACE9220311FE2\WhatsApp Image 2025-05-06 at 09.52.47_7a9f9f94.jpgC:\Users\ambra\AppData\Local\Packages\5319275A.WhatsAppDesktop_cv1g1gvanyjgm\TempState\0B845236DA4CE3D6E20524161C1966C0\WhatsApp Image 2025-05-06 at 09.52.47_b0f3d788.jpgSignature of the candidate(s):

**Name of the candidate(s):** Mr.S Sivaganesh, Mr.Pothireddy Chandrahas Reddy, Mr.Puluru Venkata Gopinadha Reddy.

**Date :** 05-05-2025

**Acknowledgements**

We would like to thank our Honorable Chancellor, **Prof. R. Sethuraman** for providing us with an opportunity and the necessary infrastructure for carrying out this project as a part of our curriculum.

We would like to thank our Honorable Vice-Chancellor **Dr. S.Vaidhyasubramaniam** and **Dr.**

**S. Swaminathan,** Dean**,** Planning & Development, for the encouragement and strategic support at every step of our college life.

We extend our sincere thanks to **Dr. R. Chandramouli,** Registrar, SASTRA Deemed to be University for providing the opportunity to pursue this project.

We extend our heartfelt thanks to **Dr. V. S. Shankar Sriram**, Dean, School of Computing, **Dr. R. Muthaiah**, Associate Dean, Research, **Dr. K.Ramkumar**, Associate Dean, Academics, **Dr. D. Manivannan**, Associate Dean, Infrastructure, **Dr. R. Alageswaran**, Associate Dean, Students Welfare.

Our guide **Dr. V. S. Shankar Sriram**, Dean, School of Computing was the driving force behind this whole idea from the start. His deep insight in the field and invaluable suggestions helped us in making progress throughout our project work. We also thank the project review panel members for their valuable comments and insights which made this project better.

We would like to extend our gratitude to all the teaching and non-teaching faculties of the School of Computing who have either directly or indirectly helped us in the completion of the project.

We gratefully acknowledge all the contributions and encouragement from my family and friends resulting in the successful completion of this project. We thank you all for providing me an opportunity to showcase my skills through project.

## List of Figures

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title** | **Page No.** |
| 3.1 | Methodology Flow Chart | 7 |
| 4.1 | Data Before Transformation | 25 |
| 4.2 | Data After Transformation | 25 |
| 4.3 | Correlation Matrix | 26 |
| 4.4 | Line plot | 27 |
| 4.5 | Scatter plot | 28 |
| 4.6 | Bar Graph | 29 |
| 4.7 | Data before Skewness and Kurtosis | 30 |
| 4.8 | Data after Skewness and Kurtosis | 30 |
| 4.9 | AdaBoost Scatter Plot | 31 |
| 4.10 | CatBoost Scatter Plot | 32 |
| 4.11 | Random Forest Scatter Plot | 32 |
| 4.12 | XGBoost Scatter Plot | 33 |
| 4.13 | Bagging Regressor Scatter Plot | 33 |
| 4.14 | LGBM Scatter Plot | 34 |
| 4.15 | Neural Network Scatter Plot | 34 |
| 4.16 | AdaBoost Density Plot | 35 |
| 4.17 | CatBoost Density Plot | 35 |
| 4.18 | Random Forest Density Plot | 36 |
| 4.19 | XGBoost Density Plot | 36 |
| 4.20 | Bagging Regressor Density Plot | 37 |

|  |  |  |
| --- | --- | --- |
| 4.21 | LGBM Density Plot | 37 |
| 4.22 | Neural Network Density Plot | 38 |
| 4.23 | Loss VS Epoch | 38 |
| 4.24 | MAE VS Epoch | 39 |
| 4.25 | Training Results | 39 |
| 4.26 | Testing Results | 40 |
| 4.27 | MSE Comparison | 41 |
| 4.28 | RMSE Comparison | 41 |
| 4.29 | 𝑅2 Comparison | 42 |
| 4.30 | MAE Comparison | 42 |

**Abbreviations**

|  |  |
| --- | --- |
| AQI | Air Quality Index |
| ML | Machine Learning |
| XG Boost | Extreme Gradient Boosting |
| LGBM | Light Gradient Boosting Machine |
| RFR | Random Forest Regressor |
| BR | Bagging Regressor |
| AP | Air Pollution |
| PM | Particulate Matter |

**ABSTRACT**

Air pollution poses a significant challenge to public health and urban planning in Chennai, a major metropolitan city in southern India. Existing methods for Air Quality Index (AQI) prediction often struggle with accuracy and reliability due to their inability to effectively capture the complex, nonlinear relationships between pollutants and AQI levels. To address these limitations, this study leverages advanced machine learning models, including XGBoost, Random Forest, Bagging Regressor, LightGBM Regressor and CatBoost, AdaBoost and Neural Network Regressor to predict AQI more accurately. The proposed approach integrates historical air quality data from 2018 to 2023, considering key pollutants such as PM2.5, PM10, NO₂, and SO₂, with PM2.5 identified as the most critical contributor to AQI (correlation: 0.91). Among the models, XGBoost demonstrated superior performance, achieving an R² of 0.921, a mean absolute error (MAE) of 0.0921, a mean square error (MSE) of 0.0148, and a root mean square error (RMSE) of 0.1217, significantly outperforming traditional models. Results show that XGBoost not only improves prediction accuracy but also offers robustness in capturing seasonal variations, such as the highest AQI in January (104.6 µg/m³) and the lowest in August (63.87 µg/m³). The study concludes that the proposed machine learning framework provides a reliable tool for AQI forecasting, highlighting its potential to support data-driven air quality management and policy making in urban environments.

***Keywords:****Air Quality Index, Machine Learning, Deep Learning, Neural Networks, Environmental Data Analysis.*

## Table of Contents

|  |  |
| --- | --- |
| **Title** | **Page NO** |
| Bonafide Certificate | **ii** |
| Declaration | **iii** |
| Acknowledgements | **iv** |
| List of Figures | **v** |
| Abbreviations | **vii** |
| Abstract | **viii** |
| 1. Summary of the Base Paper | **1** |
| 2. Merits and Demerits of the Base Paper | **5** |
| 3. Proposed Methodology | **7** |
| 4. Source Code | **9** |
| 5. Output Snapshots | **25** |
| 6. Conclusion and Future Scope | **43** |
| 7. References | **44** |
| 8. Appendix -Base Paper | **45** |

**CHAPTER 1**

### SUMMARY OF THE BASE PAPER

**Title :** Analysis of Air Quality Index Prediction using Machine learning and Deep Learning Techniques.

**Publisher :** Elsevier.

#### Year : 2023

**Journal Name :** Elsevier/Science Direct.

**DOI :** [**https://doi.org/10.1016/j.envres.2023.117354**](https://doi.org/10.1016/j.envres.2023.117354)

**Base paper URL :** [**https://www.sciencedirect.com/science/article/abs/pii/S0013935123021588**](https://www.sciencedirect.com/science/article/abs/pii/S0013935123021588)

### INTRODUCTION

The main contents of this Base paper :

#### Air Quality Prediction Models:

The study focuses on predicting the Air Quality Index (AQI) for Chennai using various machine learning models, including XGBoost, Random Forest, Bagging Regressor, LightGBM Regressor, CatBoost, AdaBoost and Neural Network Regressor. These models are trained using historical data from 2017 to 2022, including pollutant levels and meteorological parameters.

#### Key Pollutants and Features:

The most significant factor influencing AQI was found to be PM2.5, with a correlation of

0.91. Other features used for prediction include particulate matter, gaseous pollutants, and meteorological variables. The data normalization process involved log transformation to address skewness and improve model performance.

#### Model Performance:

The XGBoost model exhibited the best performance among the models tested, achieving an R² of 0.921, a mean absolute error (MAE) of 0.0921, and a root mean square error (RMSE) of 0.1217. This indicates high accuracy in AQI prediction. In contrast, Neural Network Regressor had slightly lower performance with an R² of 0.87.

#### Seasonal AQI Trends:

The study identified seasonal variations in AQI, with January having the highest average AQI of 104.6 µg/m³ and August the lowest at 63.87 µg/m³. These trends provide insight into how air quality fluctuates throughout the year in Chennai.

#### Future Air Quality Predictions:

The study concludes that air pollution in Chennai has been rising over the last two years. If current conditions persist, the city's air quality is expected to deteriorate further, emphasizing the need for ongoing monitoring and the application of machine learning models for future AQI prediction.

#### Enhancing Air quality prediction through effective dataset preparation.

For this research, we utilized an air quality dataset for Chennai, collected from 2017 to 2022. The dataset includes comprehensive records of air pollutants such as PM2.5, PM10, NO₂, CO, and meteorological parameters such as temperature, humidity, and wind speed. These features were used to predict the Air Quality Index (AQI) in the Chennai metropolitan region.

#### Dataset Characteristics

* + - **Source**: The dataset was collected from official government and environmental monitoring agencies, ensuring accurate and reliable data for analysis.
    - **Pollutants and Parameters**: The dataset includes a wide range of air pollutants that influence the AQI, including PM2.5 (Particulate Matter 2.5 microns), PM10, CO (Carbon Monoxide), and NO₂ (Nitrogen Dioxide), along with meteorological parameters such as temperature and humidity.

#### Impact Analysis of Key Pollutants and Meteorological Features

In this study, feature extraction was essential to identify which factors most significantly impact the Air Quality Index (AQI) in Chennai. Using historical data on pollutants and meteorological conditions, we assessed each feature's influence on AQI to determine the optimal predictors for our models. A heatmap was generated to illustrate correlations, revealing that **PM2.5** had the strongest correlation with AQI (0.91), making it a key feature for predictive accuracy. Other impactful features include PM10, NO₂, temperature, and humidity, each contributing to understanding AQI fluctuations.

#### Data Transformation and Normalization

Given that air quality data can often be skewed, **log transformation** was applied to normalize the distribution of features such as pollutant levels. This step addressed skewness, improved data symmetry, and enhanced the models' ability to learn from the data by making it more statistically uniform.

#### Machine Learning and Deep Learning Models for AQI Prediction

In this study, several machine learning models were implemented to predict the Air Quality Index (AQI) based on historical data. The models used include **Random Forest Regressor**, **XGBoost**, **Bagging Regressor**, **LightGBM Regressor, CatBoost, AdaBoost and Neural Network Regressor**. Each of these models has been carefully selected based on their ability to handle large datasets and predict AQI using both pollutant concentrations and meteorological data.

* + - **Model Training**: The machine learning models were trained on preprocessed datasets using pollutants like PM2.5, PM10, CO, and NO₂, as well as weather variables such as temperature and humidity. These models are designed to capture patterns in the data to predict future AQI levels.

#### Model Performance and Evaluation Metrics

To evaluate the models, standard performance metrics were used:

* **R² (Coefficient of Determination)**: This was the primary measure of model accuracy. **XGBoost** achieved an **R² value of 0.921**, the highest among the models, indicating strong predictive performance.
* **Mean Absolute Error (MAE)**: The lowest MAE of **0.0904** was observed in the Bagging Regressor model, showing minimal deviation between predicted and actual values.
* **Mean Squared Error (MSE)** and **Root Mean Squared Error (RMSE)** were also evaluated, with XGBoost demonstrating an **MSE of 0.0148** and **RMSE of 0.1217**, further affirming its superiority over other models such as LightGBM, which had a lower R² value of **0.9174**.

#### Hyperparameter Tuning and Optimization

The performance of the machine learning models was optimized by tuning their hyperparameters. **Grid Search, Random Search, Optuna and Bayesian Optimization** was applied to find the optimal combination of these parameters, further enhancing the accuracy of AQI predictions.

#### Predicting Future AQI Trends

The findings indicate that air quality in Chennai has been deteriorating over the last two years. If current pollution trends continue, the city's AQI will likely worsen, especially during the winter months. The machine learning models, particularly XGBoost, have shown to be highly effective in forecasting future AQI levels, providing valuable insights for policymakers and environmental agencies to take timely action.

**CHAPTER 2**

### MERITS AND DEMERITS OF THE BASE PAPER

**MERITS:**

**Comprehensive Dataset:** The study uses historical data from 2017 to 2022, which provides a robust basis for analyzing long-term trends in air quality and pollutant effects.

**Multiple Machine Learning and Deep Learning Models:** By employing various ML and DL models like XGBoost, Random Forest, Bagging Regressor, LightGBM, CatBoost, AdaBoost and Neural Network Regressor the study offers comparative insights on model performance, enhancing the reliability of AQI predictions.

**High Prediction Accuracy:** The selected models, especially XGBoost, achieved high R² values and low error rates, indicating strong predictive performance for AQI.

**Feature Analysis:** The study thoroughly analyzes the influence of each pollutant, notably PM2.5, on AQI, providing critical insights into pollutant contributions to air quality degradation.

**Data Normalization and Transformation:** Effective data processing techniques, such as log transformation, were used to handle skewed data and improve model performance.

**Hyperparameter Tuning:** The models underwent systematic tuning using Grid Search, Random Search, Optuna and Bayesian Optimization optimizing model configurations for higher predictive accuracy.

**Seasonal and Meteorological Factors:** The study accounts for seasonal variations and meteorological influences, such as wind speed and temperature, which significantly impact AQI, providing a realistic model context.

**Real-World Applicability**: The findings and models can inform local air quality monitoring systems and policy-making, aiding in actionable insights for environmental protection.

**Public Health Relevance**: The research links AQI and pollutant concentration to potential health impacts, emphasizing the importance of managing air quality for public health.

**Future Research Pathways:** The study identifies limitations, such as data gaps and the need for further research on missing data handling, which guides future studies for more refined AQI predictions.

### DEMERITS:

**Limited Generalizability:** The study focuses on data from Chennai, which may limit the generalizability of the model to other regions with different environmental conditions.

**Data Gaps:** The removal of missing values reduces dataset size, potentially excluding valuable information that could improve model accuracy.

**Regulatory Changes Not Accounted For:** The model doesn’t consider shifts in emission regulations, which can significantly impact air quality trends and prediction accuracy.

**Complexity in Model Interpretation**: Some models, like XGBoost and LightGBM, while accurate, lack interpretability, making it difficult to understand specific predictor impacts on AQI.

**Potential Overfitting Risks**: The high performance in the training data may indicate overfitting, which could reduce accuracy on entirely new or unseen datasets.

## Proposed Methodology

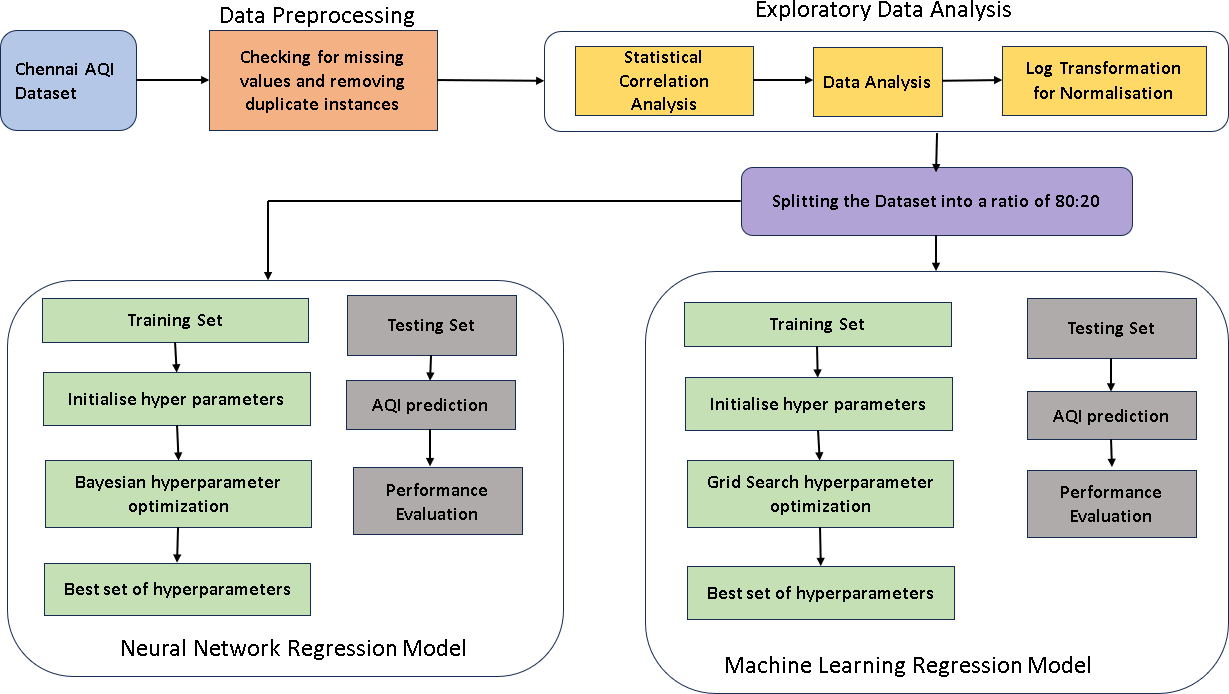
****

Fig.3.1 Proposed Methodology

#### Dataset Characteristics

* **Source**: The dataset was collected from official government and environmental monitoring agencies, ensuring accurate and reliable data for analysis.
* **Pollutants and Parameters**: The dataset includes a wide range of air pollutants that influence the AQI, including PM2.5 (Particulate Matter 2.5 microns), PM10, CO (Carbon Monoxide), and NO₂ (Nitrogen Dioxide), along with meteorological parameters such as temperature and humidity.

#### Data Preprocessing

In the data preprocessing step the data is checked for any missing values and removed any duplicated entries.

#### Data Transformation and Normalization

Given that air quality data can often be skewed, **log transformation** was applied to normalize the distribution of features such as pollutant levels. This step addressed skewness, improved data symmetry, and enhanced the models' ability to learn from the data by making it more statistically uniform.

#### Machine Learning and Deep Learning Models for AQI Prediction

The models used include **CatBoost, AdaBoost and Neural Network Regressor**. Each of these models has been carefully selected based on their ability to handle large datasets and predict AQI using both pollutant concentrations and meteorological data.

* **Model Training**: The machine learning models were trained on preprocessed datasets using pollutants like PM2.5, PM10, CO, and NO₂, as well as weather variables such as temperature and humidity.

#### Model Performance and Evaluation Metrics

To evaluate the models, standard performance metrics were used:

* **R² (Coefficient of Determination)**: This was the primary measure of model accuracy. **CatBoost** achieved an **R² value of 0.9146**, the highest among the models, indicating strong predictive performance.
* **Mean Absolute Error (MAE)**: The lowest MAE of **0.0937** was observed in the CatBoost Regressor model, showing minimal deviation between predicted and actual values.
* **Mean Squared Error (MSE)** and **Root Mean Squared Error (RMSE)** were also evaluated, with CatBoost demonstrating an **MSE of 0.016** and **RMSE of 0.1265**, further affirming its superiority over other models such as Neural Network Regressor, which had a lower R² value of **0.87**.

#### Hyperparameter Tuning and Optimization

The performance of the machine learning models was optimized by tuning their hyperparameters. **Grid Search, Random Search, Optuna and Bayesian Optimization** was applied to find the optimal combination of these parameters, further enhancing the accuracy of AQI predictions.

**CHAPTER 3**

### SOURCE CODE

* 1. **PREPROCESSING**

#### Importing Libraries:

import os

import numpy as np import pandas as pd

import matplotlib.pyplot as plt import matplotlib.pyplot as plt1 import seaborn as sns

import pandas as pd

import matplotlib.pyplot as plt

!pip install xgboost

!pip install lightgbm

!pip install "numpy<2.0" "catboost>=1.2.4" scikit-learn optuna import optuna

import joblib

from sklearn.model\_selection import train\_test\_split from sklearn.ensemble import AdaBoostRegressor from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score from catboost import CatBoostRegressor, Pool

from sklearn.model\_selection import GridSearchCV from sklearn.model\_selection import train\_test\_split

!pip install keras-tuner import numpy as np

from tensorflow import keras

from tensorflow.keras import layers from keras\_tuner import RandomSearch

from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_squared\_error, r2\_score from keras.models import Sequential

from keras.layers import Dense, Dropout from keras.optimizers import Adam

from keras\_tuner import Hyperband

from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score import tensorflow as tf

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense, Dropout, BatchNormalization, GaussianNoise from tensorflow.keras.regularizers import l2

from tensorflow.keras.optimizers import Adam, Nadam, RMSprop from keras\_tuner import BayesianOptimization

from keras\_tuner.engine.hyperparameters import HyperParameters from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout from tensorflow.keras.callbacks import EarlyStopping from sklearn.preprocessing import StandardScaler from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

from tensorflow.keras.layers import Dense, Dropout, BatchNormalization, GaussianNoise from tensorflow.keras.regularizers import l2

from tensorflow.keras.optimizers import Adam, Nadam, RMSprop from keras\_tuner import GridSearch

from sklearn.ensemble import RandomForestRegressor, BaggingRegressor

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score from scipy.stats import norm

from xgboost import XGBRegressor from lightgbm import LGBMRegressor

from sklearn.model\_selection import KFold import warnings

import pickle

pip install tensorflow keras-tuner scikit-learn pandas matplotlib from sklearn.preprocessing import StandardScaler

from keras.regularizers import l1, l2, l1\_l2

from keras.optimizers import Adam, SGD, RMSprop from keras.models import load\_model

from keras\_tuner import RandomSearch from tabulate import tabulate

#### Loading Dataset:

data\_path=r"/content/mp-dataset.xlsx"

raw\_data = pd.read\_excel(data\_path, skiprows=1, header=3)

raw\_data

#### Data Cleaning and Preprocessing:

raw\_data.isna().sum()

raw\_data.drop('BP', axis=1, inplace=True) raw\_data.isna().sum()

from scipy.stats import skew, kurtosis list=[]

for i in range(2,22): nlist=[]

nlist.append(data.columns[i])

raw\_skewness = skew(data[data.columns[i]]) raw\_kurtosis = kurtosis(data[data.columns[i]]) nlist.append(raw\_skewness) nlist.append(raw\_kurtosis)

list.append(nlist) print("{0:<10}".format('Metric'),"{0:<10}".format('Skew'),"{0:<10}".format('Kurtosis')+'\n') for i in range(len(list)):

print("{0:<10}".format(list[i][0])+"{0:>10}".format(str(round(list[i][1],2)))+"{0:>10}".format(str(rou nd(list[ i][2],2))))

columns\_to\_transform=['PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3',

'CO', 'SO2', 'Benzene', 'Toluene', 'Ozone', 'RH', 'Xylene', 'AT',

'RF', 'WS', 'WD', 'Temp', 'SR', 'AQI']

for i in range(len(columns\_to\_transform)):

print(columns\_to\_transform[i],' ',(data[columns\_to\_transform[i]] <= 0).sum()) data['CO']=data['CO'].replace(to\_replace=0,value=0.001) data['Benzene']=data['Benzene'].replace(to\_replace=0,value=0.0001) data['Toluene']=data['Toluene'].replace(to\_replace=0,value=0.0001) data['Xylene']=data['Xylene'].replace(to\_replace=0,value=0.0001) data['RF']=data['RF'].replace(to\_replace=0,value=0.0000000001) columns\_to\_skip = ['From Date','To Date']

columns\_to\_transform=['PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3',

'CO', 'SO2', 'Benzene', 'Toluene', 'Ozone', 'RH', 'Xylene', 'AT',

'RF', 'WS', 'WD', 'Temp', 'SR', 'AQI']

transformed\_data = np.log(data[columns\_to\_transform] + 1e-6) transformed\_data

list=[]

for i in range(20):

nlist=[]

trans\_skewness = skew(transformed\_data[transformed\_data.columns[i]]) trans\_kurtosis = kurtosis(transformed\_data[transformed\_data.columns[i]]) nlist.append(transformed\_data.columns[i])

nlist.append(trans\_skewness) nlist.append(trans\_kurtosis) list.append(nlist)

print("{0:<10}".format('Metric'),"{0:<10}".format('Skew'),"{0:<10}".format('Kurtosis')+'\n') for i in range(len(list)):

print("{0:<10}".format(list[i][0])+ "{0:>10}".format(str(round(list[i][1],2)))+"{0:>10}".format(str(round(list[i][2],2))))

#### Data Visualization:

correlation\_matrix=data.loc[:,'PM2.5':].corr() plt.figure(figsize=(15,10))

sns.heatmap(correlation\_matrix.corr(), annot=True, cmap='coolwarm', fmt=".3f", linewidths=1)

plt.title('Correlation Matrix') plt.show()

for col in df\_excluded.columns: plt.figure(figsize=(10, 6)) plt.plot(df\_excluded[col]) plt.title(col) plt.xlabel('Index') plt.ylabel('Value') plt.tight\_layout()

plt.show()

column\_pairs = [ ('PM10', 'AQI'),

('PM2.5','AQI'),

('PM2.5', 'PM10'),

('NO', 'NO2'),

('WS','WD'),

('CO', 'SO2'),

('NOx', 'NH3'),

('Temp', 'RH')

]

for pair in column\_pairs: plt.figure(figsize=(10, 6)) plt.scatter(df[pair[0]], df[pair[1]])

plt.title(f'Scatter Plot of {pair[0]} vs {pair[1]}') plt.xlabel(pair[0])

plt.ylabel(pair[1]) plt.show()

columns\_to\_transform=['PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3',

'CO', 'SO2', 'Benzene', 'Toluene', 'Ozone', 'RH', 'Xylene', 'AT',

'RF', 'WS', 'WD', 'Temp', 'SR', 'AQI']

data\_yearly=[]

for i in range(2017,2023):

data\_yearly.append(data[data['From Date'].str[-10:-6]== str(i)]) mean\_values\_monthly=[] wanted\_columns=columns\_to\_transform month=['01','02','03','04','05','06','07','08','09','10','11','12']

for i in range(6): nlist=[]

for j in range(12): mlist=[]

for k in range(20):

var=data\_yearly[i][data\_yearly[i]['From Date'].str[-13:-11]==month[j]][wanted\_columns[k]].mean() if(var>=0):

mlist.append(var) else:

mlist.append(0) nlist.append(mlist)

mean\_values\_monthly.append(nlist)

for i in range(20): avglist=[]

for j in range(12): list=[]

for l in range(6): list.append(mean\_values\_monthly[l][j][i])

avglist.append(sum(list)/len(list))

indexes = range(1,len(avglist)+1) plt.figure(figsize=(4,4))

plt.bar(indexes, avglist, color='skyblue') plt.xlabel('Month')

plt.ylabel('Values')

plt.title('Bar Graph for average monthly values of '+wanted\_columns[i])

avglist=[]

for j in range(6): list=[]

for l in range(12): list.append(mean\_values\_monthly[j][l][i])

avglist.append(sum(list)/len(list)) plt1.figure(figsize=(4,4))

indexes = range(2017,len(avglist)+2017) plt1.bar(indexes, avglist, color='skyblue') plt1.xlabel('Year')

plt1.ylabel('Values')

plt1.title('Bar Graph for yearly average values of '+wanted\_columns[i]) plt.show(),plt1.show()

for i in range(6):

for j in range(12): print(mean\_values\_monthly[i][j][-1])

#### Train Test Split:

from sklearn.model\_selection import train\_test\_split X = transformed\_data.drop(columns=['AQI'])

y = transformed\_data['AQI']

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) X\_train

X\_test Y\_train Y\_test

train\_data = pd.concat([X\_train, Y\_train], axis=1) test\_data = pd.concat([X\_test, Y\_test], axis=1)

### MODEL TRAINING AND TESTING

#### AdaBoost:

def tune\_adaboost(): ada\_params = {

'n\_estimators': [50, 100, 200],

'learning\_rate': [0.01, 0.1, 0.5],

'estimator max\_depth': [3, 5, 7]

}

base\_tree = DecisionTreeRegressor(random\_state=42) ada = AdaBoostRegressor(

estimator=base\_tree, random\_state=42

)

grid\_search = GridSearchCV( ada,

ada\_params, cv=5,

scoring='neg\_mean\_squared\_error'

)

grid\_search.fit(X\_train, y\_train)

print("Best AdaBoost Parameters:", grid\_search.best\_params\_) return grid\_search.best\_estimator\_

best\_ada = tune\_adaboost()

def evaluate\_model(model, X\_train, y\_train): preds = model.predict(X\_train)

return {

'MSE': mean\_squared\_error(y\_train, preds), 'MAE': mean\_absolute\_error(y\_train, preds),

'RMSE': np.sqrt(mean\_squared\_error(y\_train, preds)), 'R²': r2\_score(y\_train, preds)

}

print("\nAdaBoost Performance(TRAINING):") ada\_metrics = evaluate\_model(best\_ada, X\_train, y\_train) for metric, value in ada\_metrics.items():

print(f"{metric}: {value:.4f}")

def evaluate\_model(model, X\_test, y\_test): preds = model.predict(X\_test)

return {

'MSE': mean\_squared\_error(y\_test, preds), 'MAE': mean\_absolute\_error(y\_test, preds),

'RMSE': np.sqrt(mean\_squared\_error(y\_test, preds)), 'R²': r2\_score(y\_test, preds)

}

print("\nAdaBoost Performance(TESTING):") ada\_metrics = evaluate\_model(best\_ada, X\_test, y\_test) for metric, value in ada\_metrics.items():

print(f"{metric}: {value:.4f}")

#### CatBoost:

def catboost\_objective(trial): params = {

'iterations': trial.suggest\_int('iterations', 500, 2000),

'depth': trial.suggest\_int('depth', 4, 12),

'learning\_rate': trial.suggest\_float('learning\_rate', 0.001, 0.1, log=True),

'l2\_leaf\_reg': trial.suggest\_float('l2\_leaf\_reg', 1, 10),

'random\_strength': trial.suggest\_float('random\_strength', 0.1, 10)

}

model = CatBoostRegressor(\*\*params, verbose=0)

model.fit(X\_train, y\_train, eval\_set=(X\_test, y\_test), early\_stopping\_rounds=50) return np.sqrt(mean\_squared\_error(y\_test, model.predict(X\_test)))

study = optuna.create\_study(direction='minimize') study.optimize(catboost\_objective, n\_trials=10)

# Add after the Optuna study optimization print("\n=== Best CatBoost Hyperparameters ===") for param, value in study.best\_params.items():

print(f"{param:.<20} {value}")

best\_catboost = CatBoostRegressor(\*\*study.best\_params, verbose=0).fit(X\_train, y\_train)

def evaluate\_model(model, X\_train, y\_train): preds = model.predict(X\_train)

return {

'MSE': mean\_squared\_error(y\_train, preds), 'MAE': mean\_absolute\_error(y\_train, preds),

'RMSE': np.sqrt(mean\_squared\_error(y\_train, preds)), 'R²': r2\_score(y\_train, preds)

}

print("\nCatBoost Performance(TRAINING):")

catboost\_metrics = evaluate\_model(best\_catboost, X\_train, y\_train) for metric, value in catboost\_metrics.items():

print(f"{metric}: {value:.4f}")

def evaluate\_model(model, X\_test, y\_test): preds = model.predict(X\_test)

return {

'MSE': mean\_squared\_error(y\_test, preds), 'MAE': mean\_absolute\_error(y\_test, preds),

'RMSE': np.sqrt(mean\_squared\_error(y\_test, preds)), 'R²': r2\_score(y\_test, preds)

}

print("\nCatBoost Performance(TESTING):")

catboost\_metrics = evaluate\_model(best\_catboost, X\_test, y\_test) for metric, value in catboost\_metrics.items():

print(f"{metric}: {value:.4f}")

#### Bagging Regressor:

bagging\_param\_grid = { 'n\_estimators': [100, 200, 300],

'max\_samples': [0.5, 0.7, 0.9]

}

bagging\_grid\_search = GridSearchCV(estimator=modelss["Bagging\_Regressor"], param\_grid=bagging\_param\_grid, cv=5, n\_jobs=-1)

bagging\_grid\_search.fit(X, y)

best\_params\_bagging = bagging\_grid\_search.best\_params\_ best\_estimator\_bagging = bagging\_grid\_search.best\_estimator\_

[best\_params\_bagging]

models = {"Bagging\_Regressor": BaggingRegressor(n\_estimators=best\_params\_bagging['n\_estimators'], max\_samples=best\_params\_bagging['max\_samples'], random\_state=42)}

bagging\_model=models["Bagging\_Regressor"] bagging\_model.fit(X\_train, y\_train)

r2\_train = bagging\_model.score(X\_train, y\_train) \* 100 print(r2\_train)

y\_pred\_train\_bgr=bagging\_model.predict(X\_train) y\_pred\_bgr = bagging\_model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred) rmse = np.sqrt(mse)

print("Root Mean Squared Error (RMSE):" , rmse)

mse = mean\_squared\_error(y\_test, y\_pred) print("Mean Squared Error:" , mse)

mae = mean\_absolute\_error(y\_test, y\_pred) print("Mean Absolute Error (MAE):" , mae)

#### XGBoost Regressor:

xgb\_param\_grid = { 'n\_estimators': [100, 200],

'max\_depth': [5, 10],

'learning\_rate': [0.1, 0.01]

}

xgb\_grid\_search = GridSearchCV(estimator=modelss["XGBoost"], param\_grid=xgb\_param\_grid, cv=5, n\_jobs=-1)

xgb\_grid\_search.fit(X, y)

best\_params\_xgb = xgb\_grid\_search.best\_params\_ best\_estimator\_xgb = xgb\_grid\_search.best\_estimator\_

[best\_params\_xgb]

models = {max\_depth=best\_params\_xgb['max\_depth'], learning\_rate=best\_params\_xgb['learning\_rate'], random\_state=42),}

xgb\_model = models["XGBoost"] xgb\_model.fit(X\_train, y\_train)

xgb\_model.score(X\_train,y\_train) \* 100

mse = mean\_squared\_error(y\_test, y\_pred) rmse = np.sqrt(mse)

print("Root Mean Squared Error (RMSE):",rmse)

mse = mean\_squared\_error(y\_test, y\_pred) print("Mean Squared Error:",mse)

mae = mean\_absolute\_error(y\_test, y\_pred) print("Mean Absolute Error (MAE):" , mae)

#### LGBM Regressor:

lgbm\_param\_grid = { 'n\_estimators': [50, 100, 200],

'max\_depth': [3, 5, 7],

'learning\_rate': [0.01, 0.1, 1],

'num\_leaves': [31, 61, 121]

}

lgbm\_grid\_search = GridSearchCV(estimator=modelss["LightGBM"], param\_grid=lgbm\_param\_grid, cv=5, n\_jobs=-1)

lgbm\_grid\_search.fit(X, y)

best\_params\_lgbm = lgbm\_grid\_search.best\_params\_ best\_estimator\_lgbm = lgbm\_grid\_search.best\_estimator\_

[best\_params\_lgbm]

models = {"LightGBM": LGBMRegressor(n\_estimators=best\_params\_lgbm['n\_estimators'], max\_depth=best\_params\_lgbm['max\_depth'], learning\_rate=best\_params\_lgbm['learning\_rate'], num\_leaves=61)}

warnings.filterwarnings("ignore", category=UserWarning, module="lightgbm") lgbm=models["LightGBM"]

lgbm.fit(X\_train, y\_train)

r2\_scoree = lgbm.score(X\_train, y\_train) \* 100 print("r2\_score:",r2\_scoree)

y\_pred\_train\_lgbm=lgbm.predict(X\_train) y\_pred\_lgbm = lgbm.predict(X\_test)

mse\_lgbm = mean\_squared\_error(y\_test, y\_pred\_lgbm)

mse = mean\_squared\_error(y\_test, y\_pred) print("Mean Squared Error: " , mse)

mse = mean\_squared\_error(y\_test, y\_pred) rmse = np.sqrt(mse)

print("Root Mean Squared Error (RMSE):" , rmse)

mae = mean\_absolute\_error(y\_test, y\_pred) print("Mean Absolute Error (MAE):", mae)

#### Random Forest Regressor:

models = {"Random\_Forest\_Regressor": RandomForestRegressor(n\_estimators=100, max\_depth=15, max\_features='sqrt', min\_samples\_leaf=1, min\_samples\_split=2, random\_state=42)}

randFor = models["Random\_Forest\_Regressor"] randFor.fit(X\_train,y\_train)

randFor.score(X\_train,y\_train) \* 100

y\_pred\_train=randFor.predict(X\_train) y\_pred = randFor.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred) print("Mean Squared Error:", mse)

mae = mean\_absolute\_error(y\_test, y\_pred) print("Mean Absolute Error (MAE):", mae) mse = mean\_squared\_error(y\_test, y\_pred) rmse = np.sqrt(mse)

print("Root Mean Squared Error (RMSE):", rmse)

#### Neural Network Regressor:

def model\_builder(hp): model = Sequential()

for i in range(hp.Int('num\_layers', 1, 3)): model.add(Dense(

units=hp.Int(f'units\_{i}', min\_value=32, max\_value=256, step=32), activation=hp.Choice('activation', ['relu', 'tanh']), kernel\_initializer=hp.Choice('initializer', ['he\_normal', 'glorot\_uniform'])

))

if hp.Boolean(f'dropout\_{i}'): model.add(Dropout(rate=hp.Float(f'dropout\_rate\_{i}', 0.2, 0.5, step=0.1)))

model.add(Dense(1, activation='linear')) optimizer = hp.Choice('optimizer', ['adam', 'sgd'])

lr = hp.Float('lr', min\_value=1e-4, max\_value=1e-2, sampling='log')

if optimizer == 'adam':

opt = Adam(learning\_rate=lr) else:

opt = SGD(learning\_rate=lr) model.compile(optimizer=opt, loss='mse', metrics=['mae']) return model

tuner = RandomSearch( model\_builder, objective='val\_loss',

max\_trials=20, # Number of trials for random search executions\_per\_trial=1, # Number of executions per trial directory='random\_search\_tuning', project\_name='aqi\_random\_search',

overwrite=True

)

early\_stop = keras.callbacks.EarlyStopping( monitor='val\_loss',

patience=10, restore\_best\_weights=True

)

tuner.search( X\_train, Y\_train, epochs=100,

validation\_split=0.2, callbacks=[early\_stop],

verbose=1

)

best\_hps = tuner.get\_best\_hyperparameters(num\_trials=1)[0] model = tuner.hypermodel.build(best\_hps)

history = model.fit( X\_train, Y\_train, epochs=200, batch\_size=32, validation\_split=0.2, callbacks=[early\_stop], verbose=1

)

model.save('Neural\_network\_regressor.keras')

def plot\_history(history): plt.figure(figsize=(12, 6))

plt.plot(history.history['loss'], label='Train Loss') plt.plot(history.history['val\_loss'], label='Validation Loss') plt.title('Model Performance')

plt.xlabel('Epochs') plt.ylabel('Loss') plt.legend() plt.grid(True)

plt.savefig('random\_search\_training\_performance.png') plt.show()

plot\_history(history)

nn\_pred\_train = model.predict(X\_train)

mae\_train = mean\_absolute\_error(Y\_train, nn\_pred\_train) mse\_train = mean\_squared\_error(Y\_train, nn\_pred\_train) rmse\_train = np.sqrt(mse\_train)

r2\_train = r2\_score(Y\_train, nn\_pred\_train)

print("Training Metrics:") print(f"MAE: {mae\_train:.2f}")

print(f"MSE: {mse\_train:.2f}") print(f"RMSE: {rmse\_train:.2f}") print(f"R²: {r2\_train:.2f}")

nn\_pred\_test = model.predict(X\_test)

mae\_test = mean\_absolute\_error(Y\_test, nn\_pred\_test) mse\_test = mean\_squared\_error(Y\_test, nn\_pred\_test) rmse\_test = np.sqrt(mse\_test)

r2\_test = r2\_score(Y\_test, nn\_pred\_test)

print("\nTesting Metrics:") print(f"MAE: {mae\_test:.2f}")

print(f"MSE: {mse\_test:.2f}") print(f"RMSE: {rmse\_test:.2f}") print(f"R²: {r2\_test:.2f}")

plt.figure(figsize=(10, 5)) plt.plot(history.history['mae'], label='Train MAE') plt.plot(history.history['val\_mae'], label='Val MAE') plt.title('Model Performance')

plt.xlabel('Epochs') plt.ylabel('Mean Absolute Error') plt.legend()

plt.grid(True) plt.show()

#### Training Resutls:

listt=[y\_pred\_train,y\_pred\_train\_xgb,y\_pred\_train\_bgr,y\_pred\_train\_lgbm] models\_list=['Random Forest','XGBoost','BaggingRegressor','LGBMRegressor']

print("{:<20} {:<10} {:<10} {:<10} {:<10}".format("Model", "MSE", "RMSE", "R2", "MAE"))

for i in range(len(listt)):

mse = round(mean\_squared\_error(y\_train, listt[i]),4) rmse=round(np.sqrt(mse),4) r2=round(r2\_score(y\_train,listt[i]),4) mae=round(mean\_absolute\_error(y\_train,listt[i]),4)

print("{:<20} {:<10} {:<10} {:<10} {:<10}".format(models\_list[i], mse, rmse, r2, mae))

#### Testing Results:

listt=[y\_pred,y\_pred\_xgb,y\_pred\_bgr,y\_pred\_lgbm]

models\_list=['Random Forest','XGBoost','BaggingRegressor','LGBMRegressor']

print("{:<20} {:<10} {:<10} {:<10} {:<10}".format("Model", "MSE", "RMSE", "R2", "MAE"))

for i in range(len(listt)):

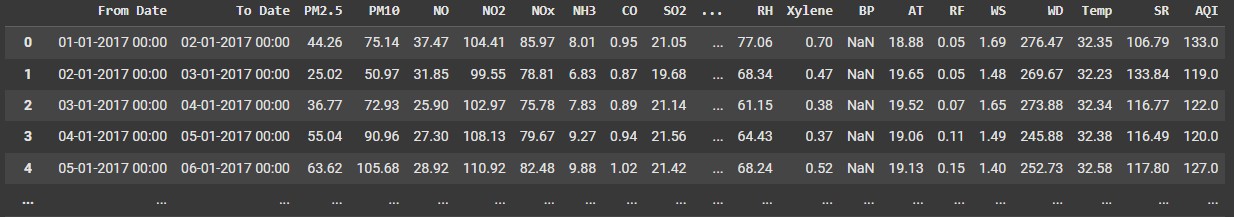
mse = round(mean\_squared\_error(y\_test, listt[i]),4) rmse=round(np.sqrt(mse),4) r2=round(r2\_score(y\_test,listt[i]),4) mae=round(mean\_absolute\_error(y\_test,listt[i]),4)

print("{:<20} {:<10} {:<10} {:<10} {:<10}".format(models\_list[i], mse, rmse, r2, mae))

# CHAPTER - 4

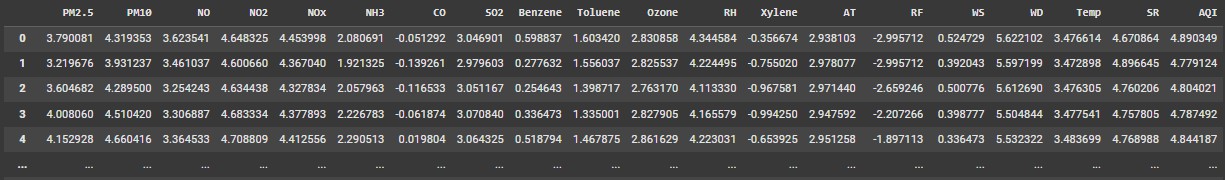
**OUTPUT SNAPSHOTS**

**Data Before Transformation:**

****

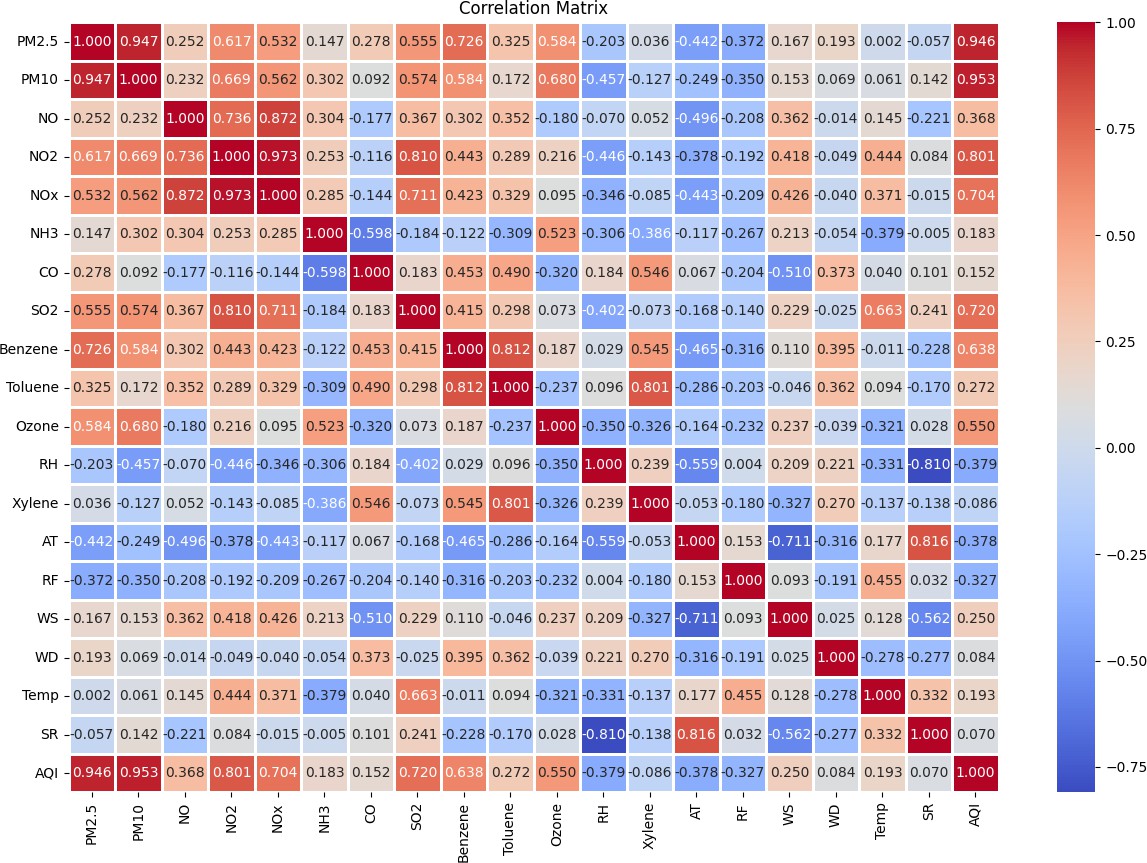
**Fig. 4.1 Before Transformation**

**Data After Transformation:**

****

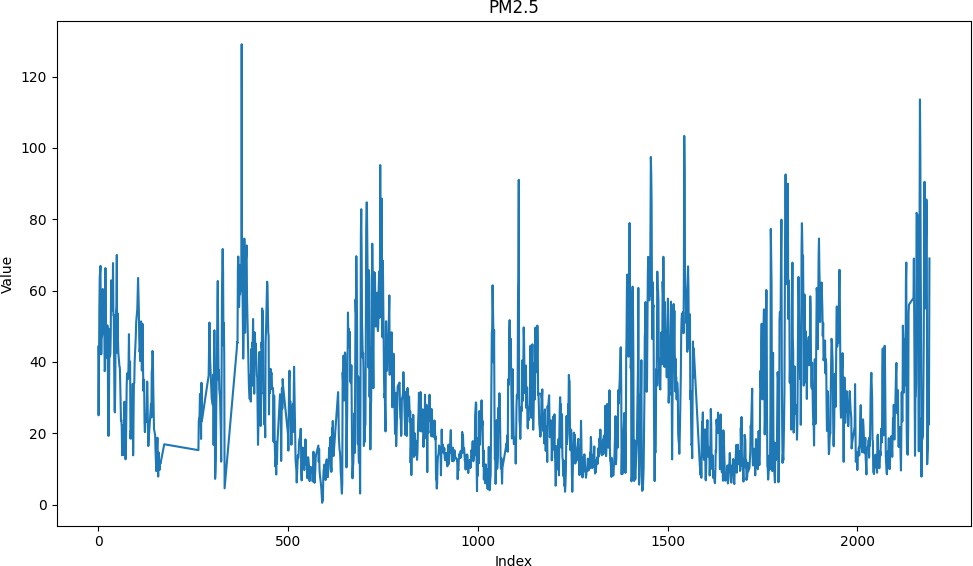
**Fig. 4.2 After Transformation**

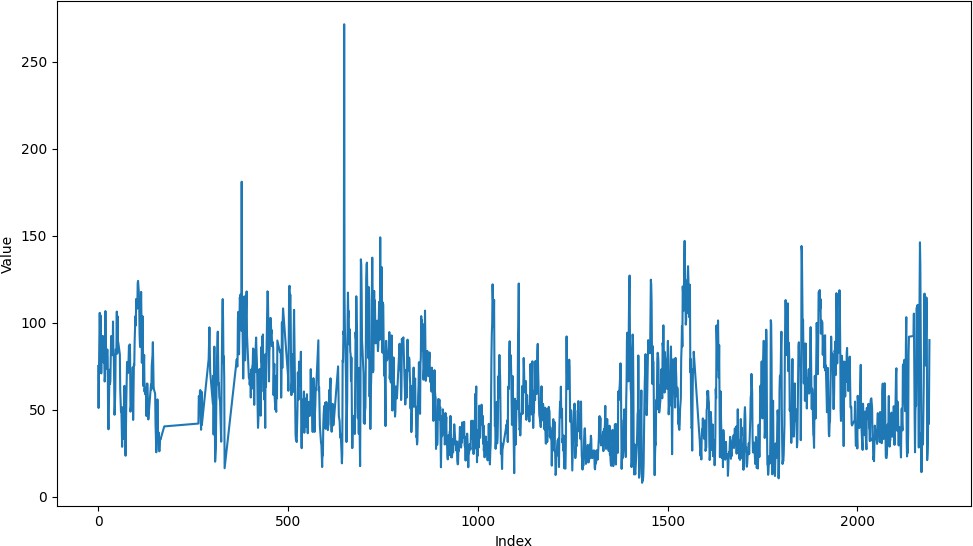
**Correlation Matrix:**

****

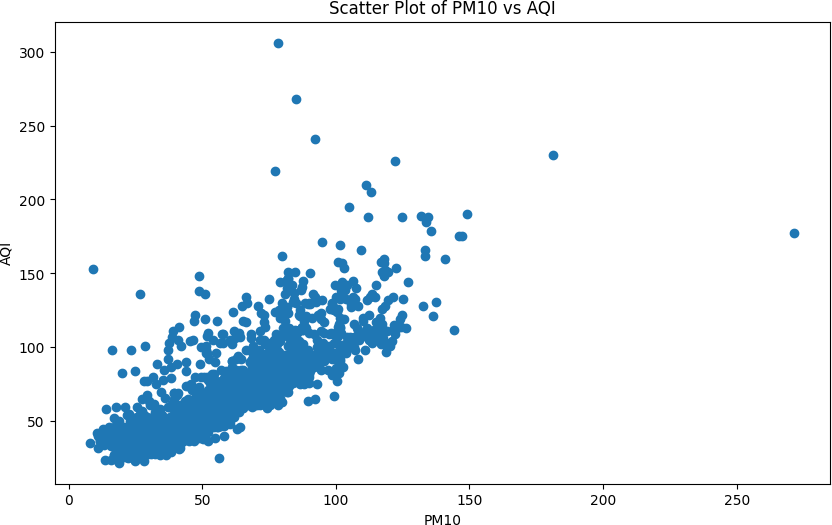
**Fig. 4.3 Correlation Matrix**

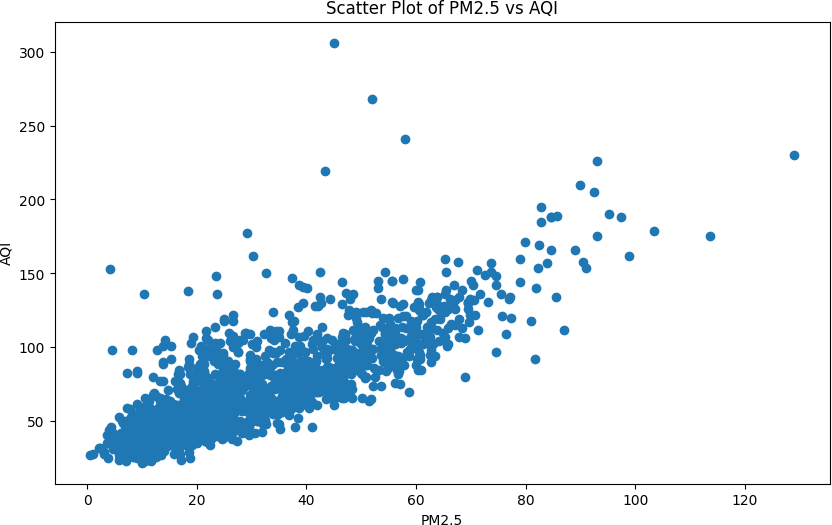
**Data Visualization:**

****

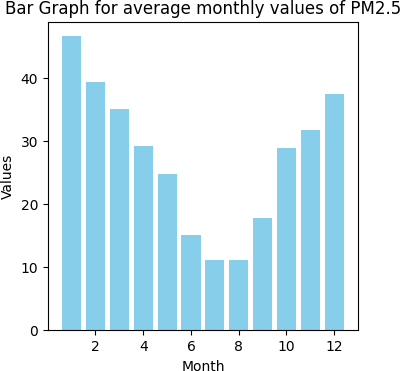
****

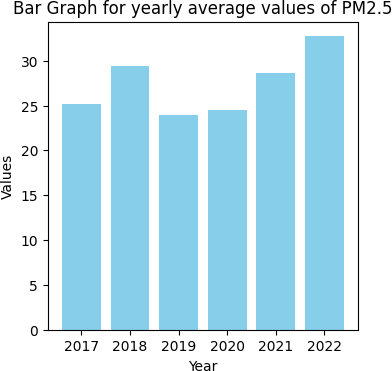
**Fig.4.4 Line Plot**



**\**

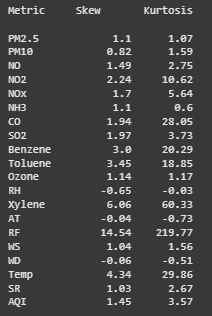
**Fig.4.5 Scatter Plot**



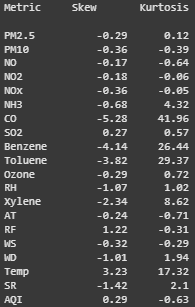
****

**Fig.4.6 Bar graph**

**Data Before Skewness and Kurtosis**

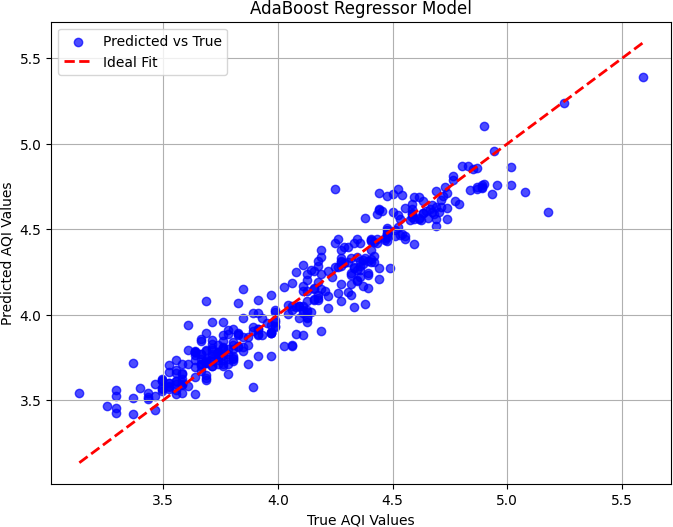
****

**Fig.4.7 Data Before Skewness and Kurtosis Data After Skewness and Kurtosis**

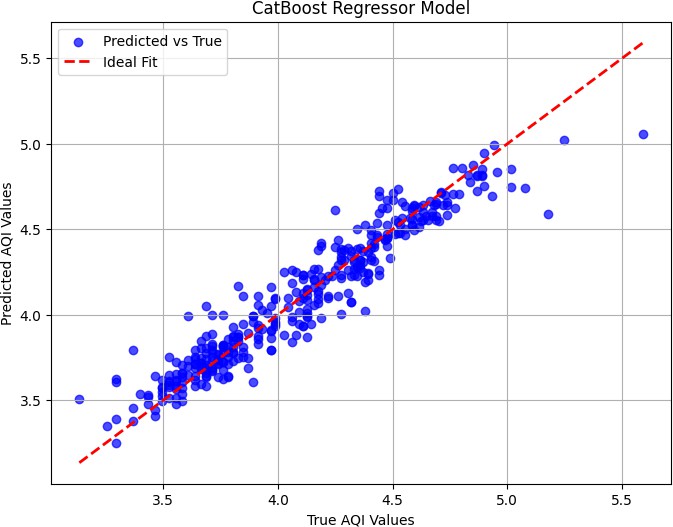


**Fig.4.8 Data After Skewness and Kurtosis**

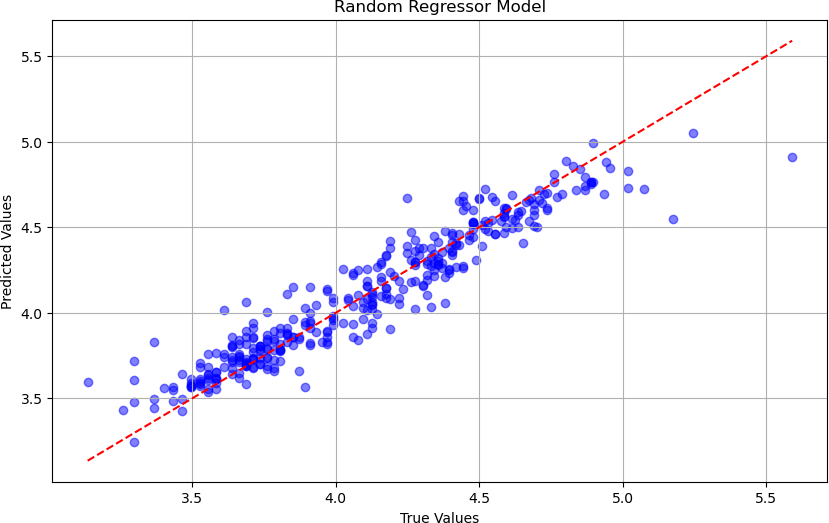
**Scatter plots of all models**

****

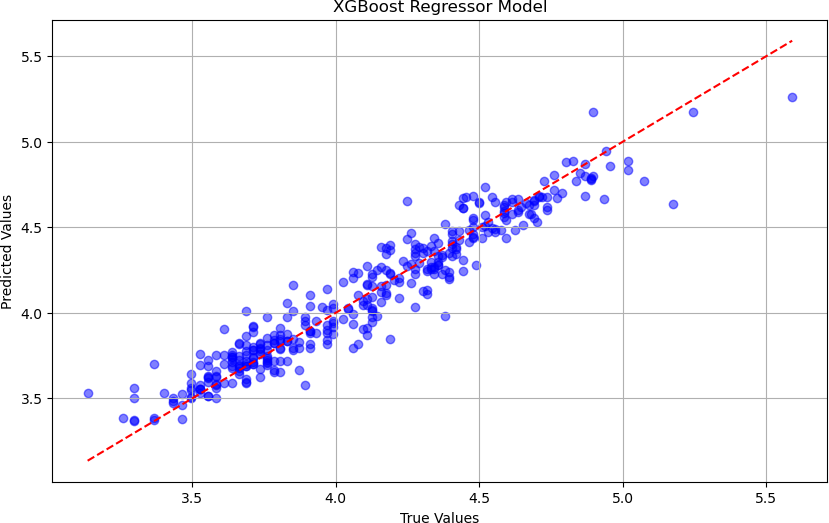
**Fig.4.9 AdaBoost Scatter Plot**



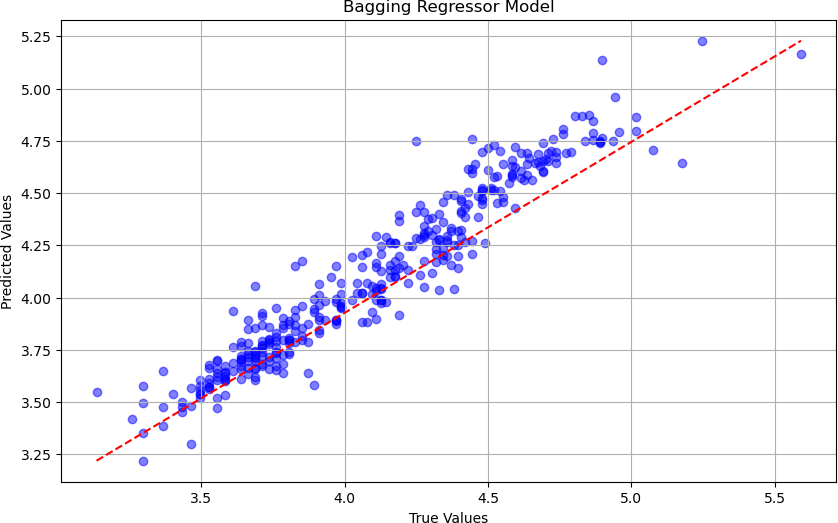
**Fig.4.10 CatBoost Scatter Plot**

****

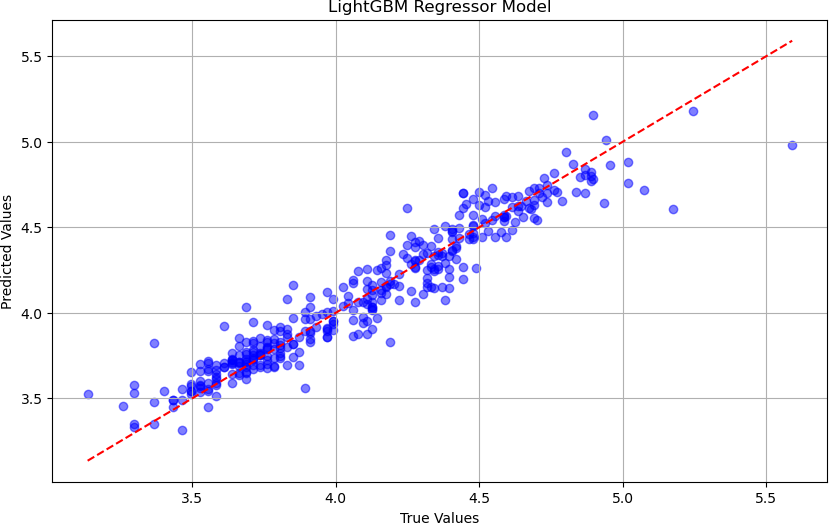
**Fig.4.11 Random Forest Scatter Plot**



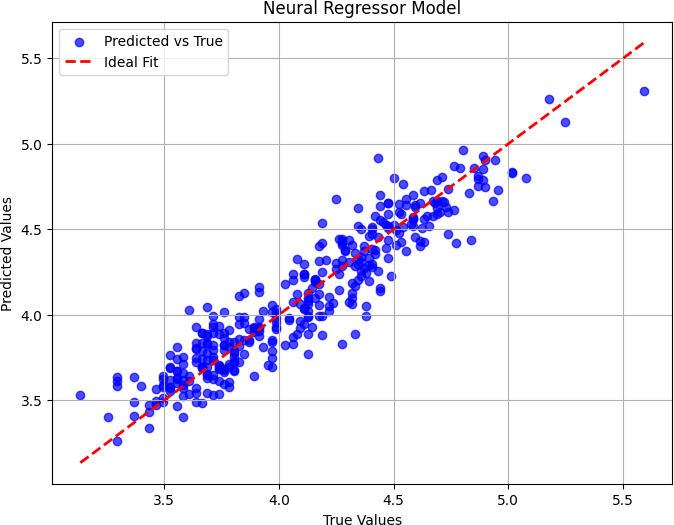
**Fig.4.12 XGBoost Scatter Plot**

****

**Fig.4.13 Bagging Regressor Scatter Plot**



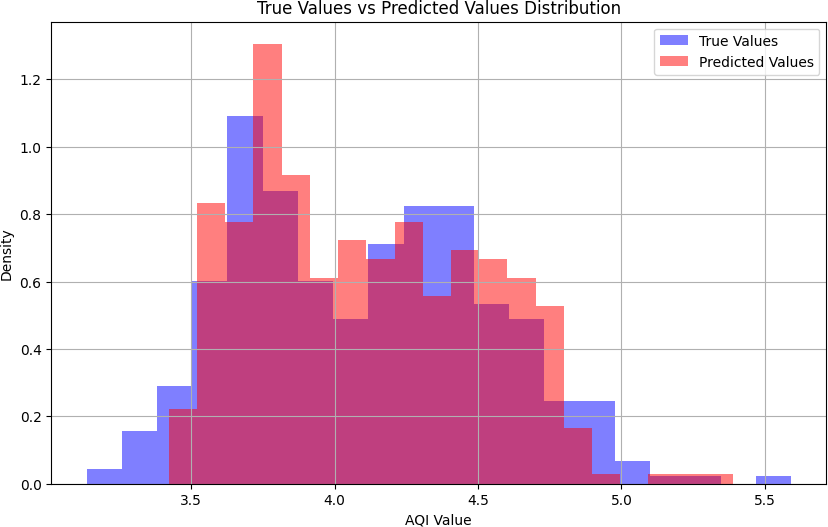
**Fig.4.14 LGBM Scatter Plot**

****

**Fig.4.15 Neural Network Scatter Plot**

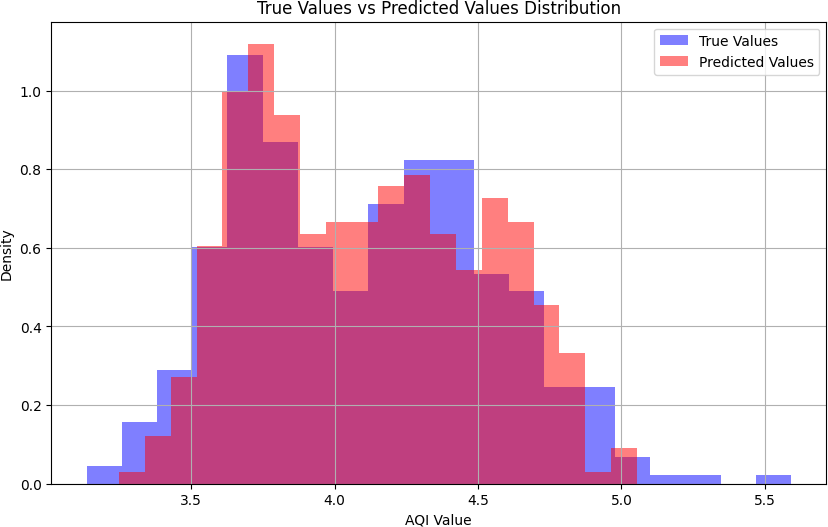
**Density plots of all models**

**AdaBoost:**

****

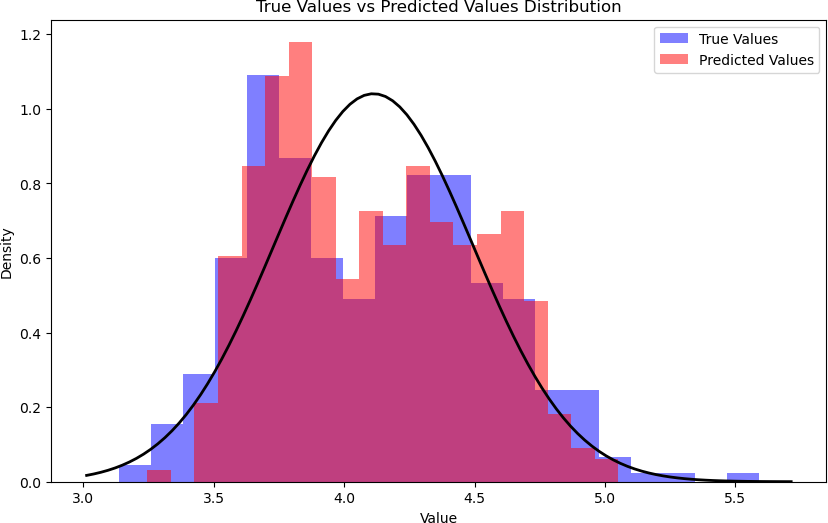
**Fig.4.16 AdaBoost Density Plot**

**CatBoost:**

****

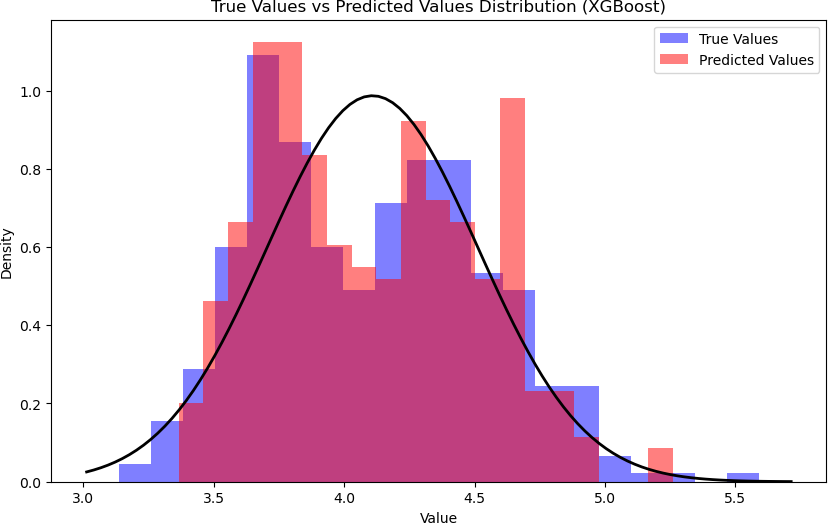
**Fig.4.17 CatBoost Density Plot**

**Random Forest Regressor:**

****

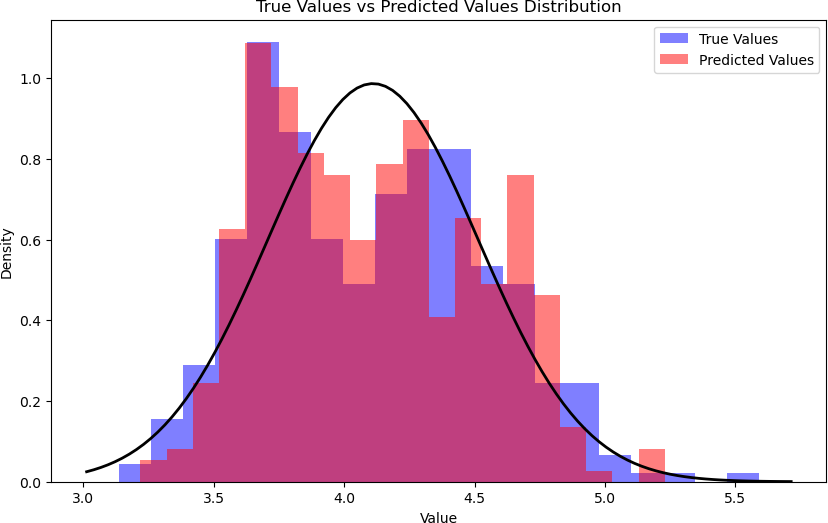
**Fig.4.18 Random Forest Density Plot**

**XGBoost Regressor:**

****

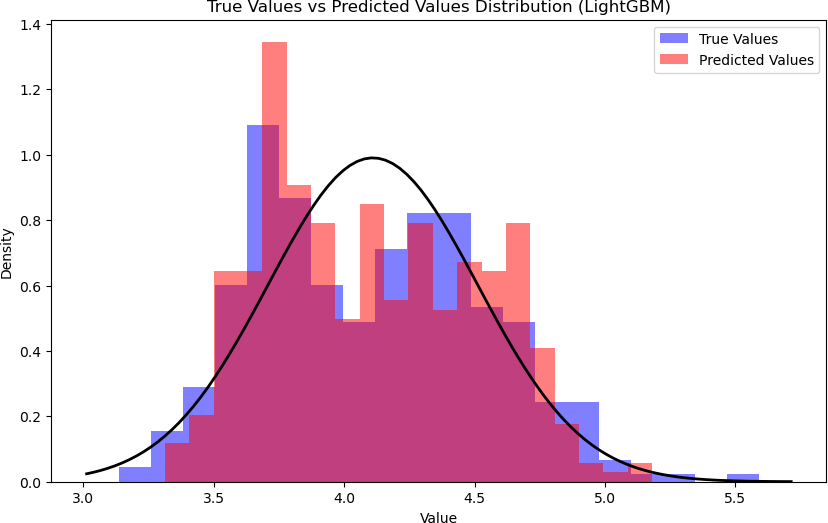
**Fig.4.19 XGBoost Density Plot**

**Bagging Regressor:**

****

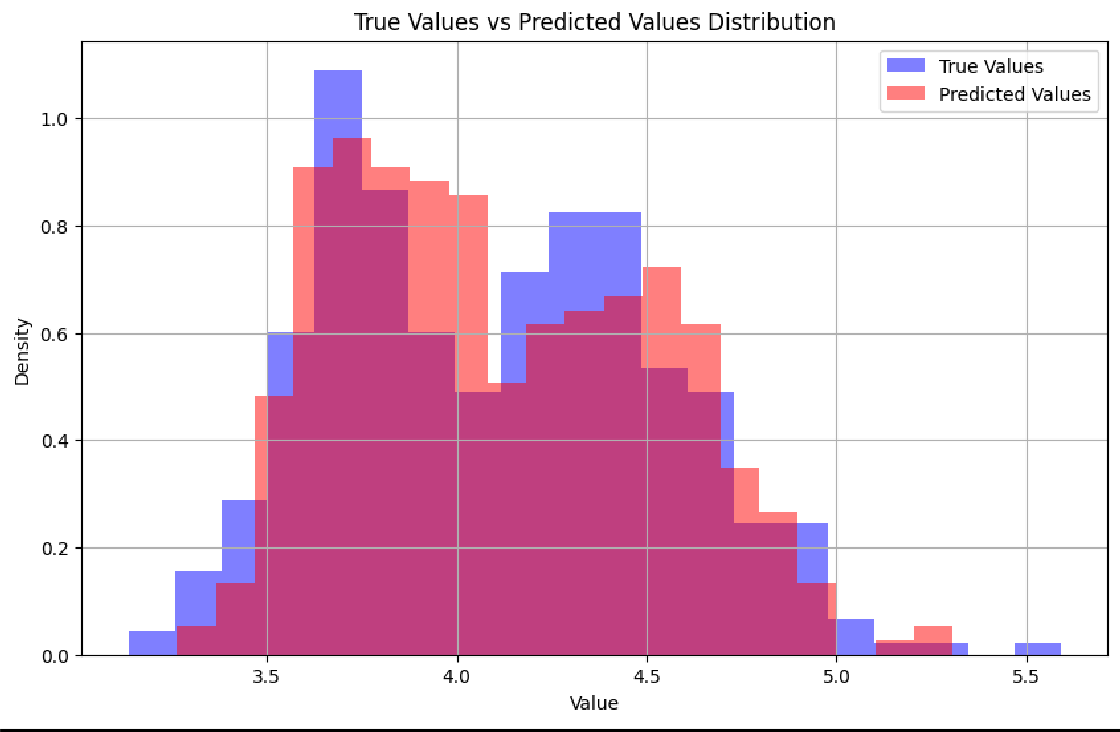
**Fig.4.20 Bagging Regressor Density Plot**

**LGBM Regressor:**

****

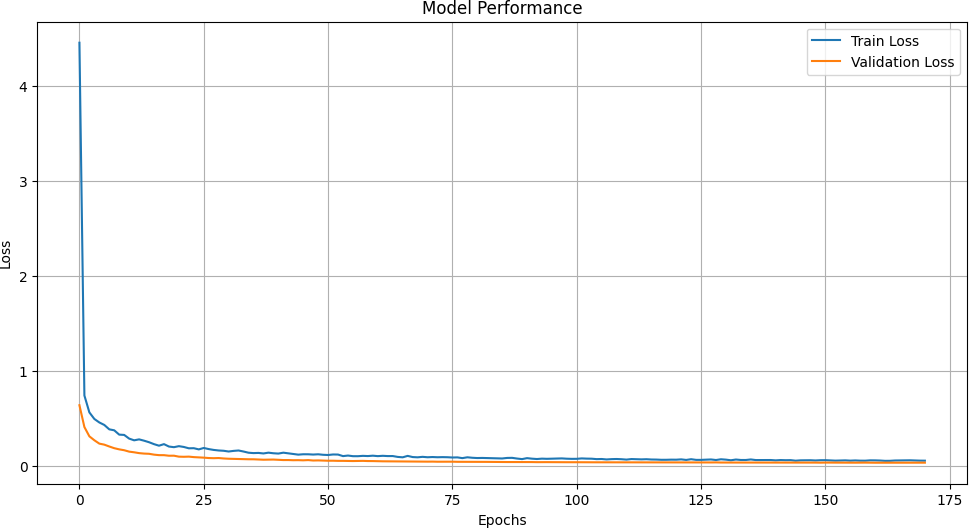
**Fig.4.21 LGBM Density Plot**

**Neural Network Regressor:**

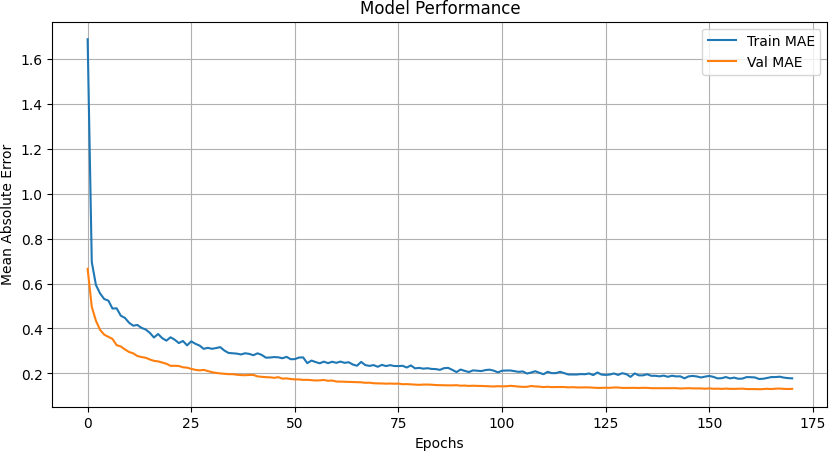


**Fig.4.22 Neural Network Density Plot**

* 1. **Neural Network Regressor Model Performance Visualisation:**

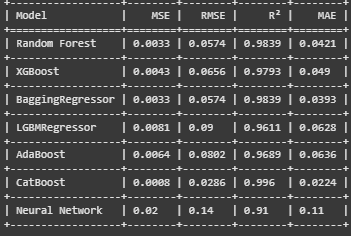
****

**Fig.4.23 Loss vs Epoch**



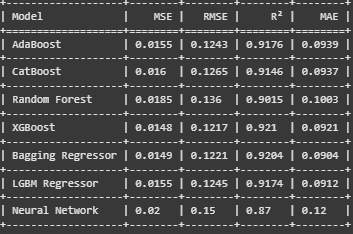
**Fig.4.24 MAE vs Epoch**

* 1. **Comparison of training Results:**

****

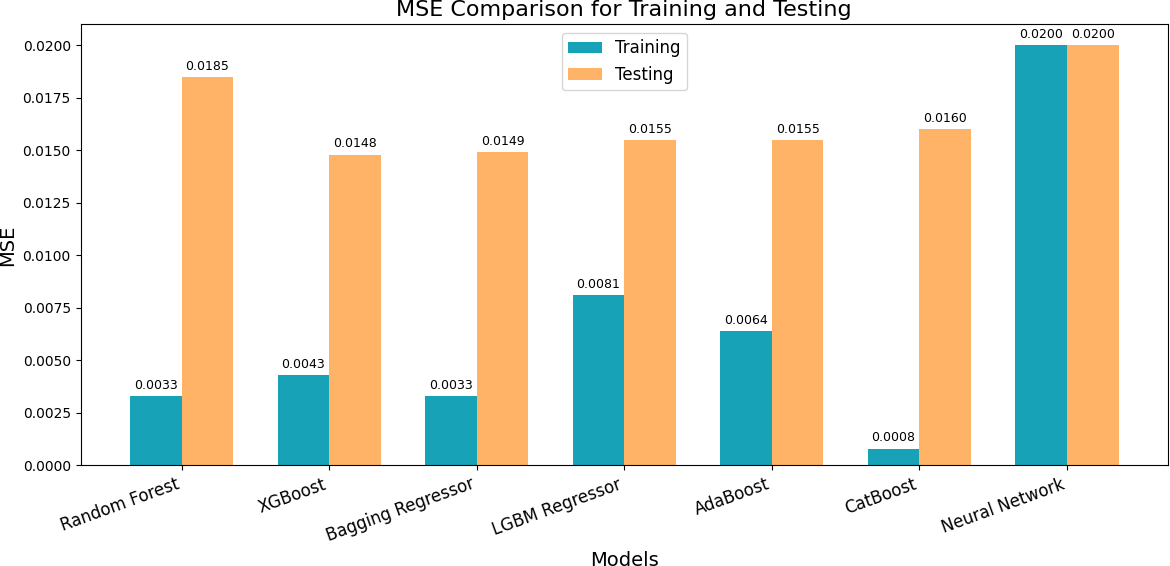
**Fig.4.25 Training results**

* 1. **Comparison of testing Results:**

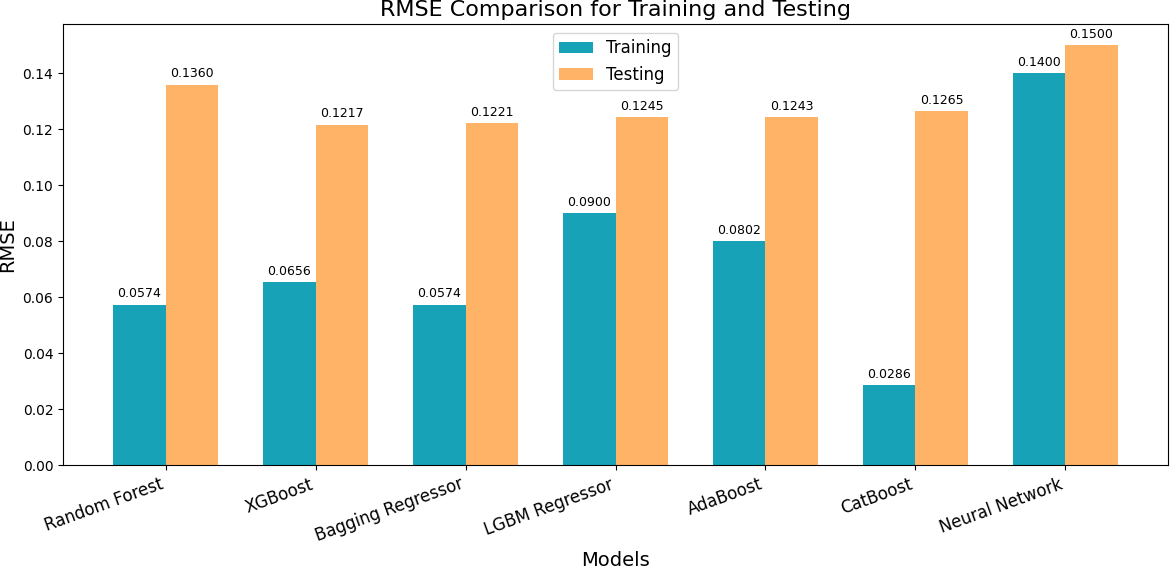
****

**Fig.4.26 Testing Results**

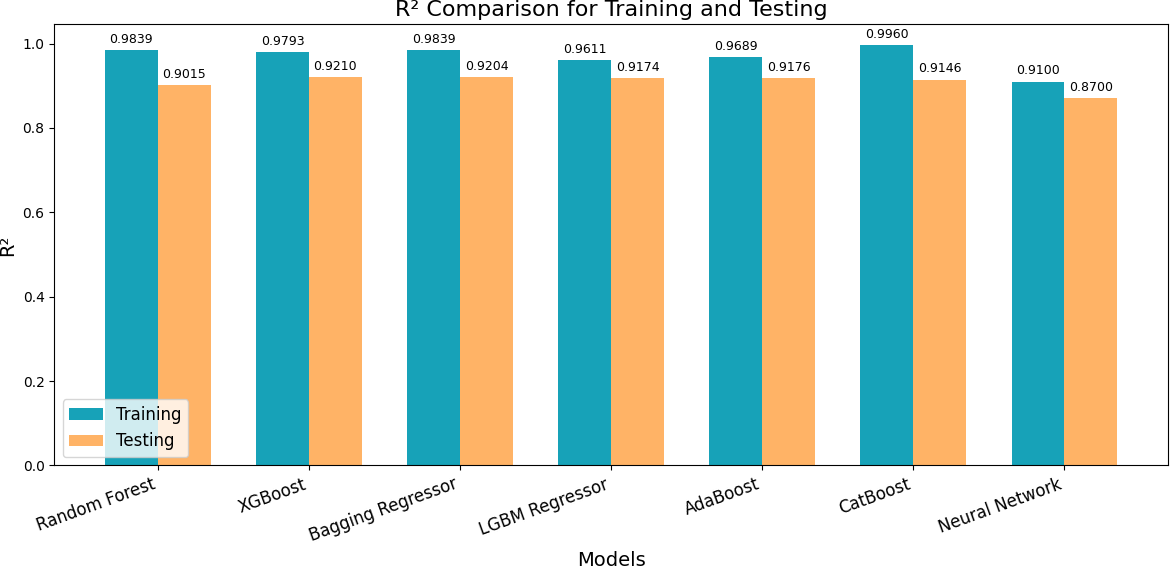
* 1. **Comparison of Results:**

****

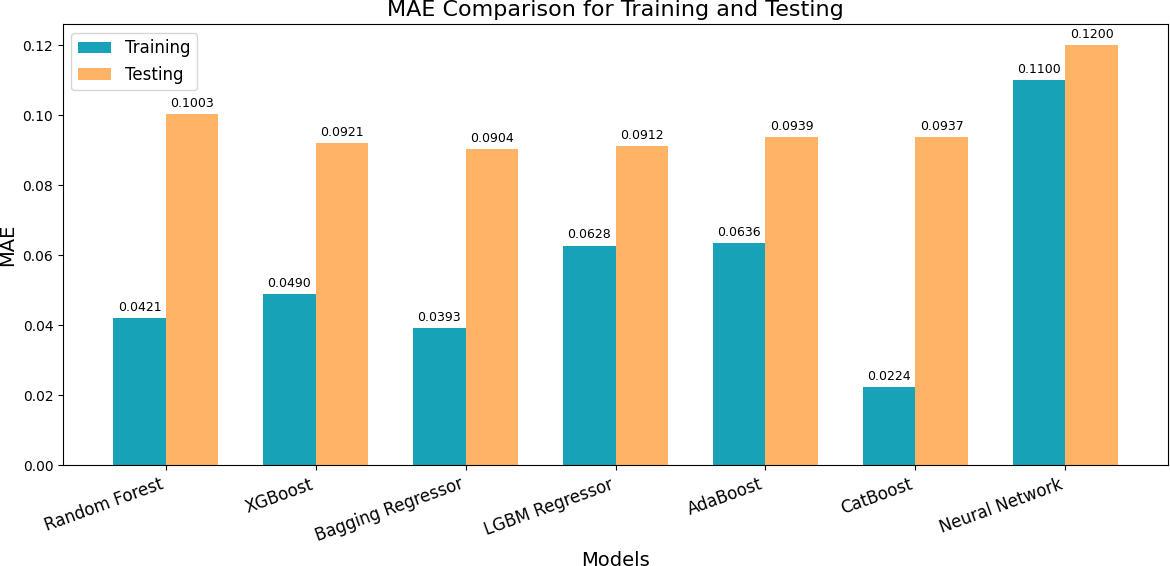
**Fig.4.27 MSE Comparison**

****

**Fig.4.28 RMSE Comparison**



**Fig.4.29 R2 Comparison**

****

**Fig.4.30 MAE Comparison**

# CHAPTER - 5

### CONCLUSION AND FUTURE SCOPE

This study emphasizes the importance of an accurate AQI prediction model in the management of air quality for urban centers such as Chennai. Historical data from 2017 to 2022 was analyzed, and it was found that PM2.5 is the most significant pollutant affecting AQI, with high correlation values. Machine learning models, such as XGBoost and CatBoost Regressor achieved high predictive accuracy and robust AQI forecasts. These findings reveal the necessity of proactively managing air quality to reduce health effects due to pollution. Machine learning methods' results, which are applied for dealing with high-dimensional, multi-parameter AQI datasets indicate the relevance of machines for the operation of real-time AQI monitoring systems, thus supporting environmental policies based on data. Further studies can take this AQI prediction model to other regions by making it adaptable to varied environmental conditions and pollutant profiles. Advanced data handling techniques like imputation of missing data will also increase the accuracy of the data and enhance the reliability of the models. Also, policy frameworks can be used in order to integrate these models to respond dynamically to levels of pollution in cities. Further investigation in deep learning techniques could therefore provide further enhancements in predictability, especially for urban areas with multiple sources of pollution. Lastly, through the connection of AQI predictions to health impact assessments, it can be useful in further developing more integrated environmental and public health policies.

# CHAPTER 6 - REFERENCES

* Wang, J., Li, X., Jin, L., Li, J., Sun, Q., and Wang, H. "An Air Quality Index Prediction Model Based on CNN-ILSTM." \*Scientific Reports\*, 2022. <https://doi.org/10.1038/S41598-022-12355-6>.
* Wang, S., Aggarwal, C., and Liu, H. "Random-Forest-Inspired Neural Networks." \*ACM Transactions on Intelligent Systems and Technology\*, 2018. <https://doi.org/10.1145/3232230>.
* Singh, R. P., and A. Chauhan. "Impact of Lockdown on Air Quality in India During COVID-19 Pandemic." \*Air Quality, Atmosphere & Health\* 13, no. 8 (2020): 921–928. [Analysis\_of\_Air\_Quality\_Index\_Prediction\_using\_Machine\_learning\_and\_Deep\_Learning\_Techniq](https://pubmed.ncbi.nlm.nih.gov/32837613/) [ues\_To\_be\_reframed[1].docx](https://pubmed.ncbi.nlm.nih.gov/32837613/).
* Lu, X., Zhang, L., Shen, L., 2021. Tropospheric ozone interacts with weather and climate[. Air](https://www.researchgate.net/publication/367883441_Tropospheric_ozone_interacts_with_weather_and_climate) [Pollution, Clim. Heal. An Integr. Perspect. Their Interact](https://www.researchgate.net/publication/367883441_Tropospheric_ozone_interacts_with_weather_and_climate).
* Khillare, P.S., Sarkar, S., 2012. Airborne inhalable metals in residential areas of Delhi, India: distribution, source apportionment and health risks. Atmos. Pollut. Res. 3, 46–54. [https://doi.org/10.5094/APR.2012.004.](https://doi.org/10.5094/APR.2012.004)
* Air-pollution prediction in smart city, deep learning approach. J. Big Data 8, 1–21. [https://doi.org/10.1186/S40537-021- 00548-1/FIGURES/17.](https://doi.org/10.1186/S40537-021-)

# CHAPTER 7 - APPENDIX

### BASE PAPER

Gokulan Ravindran a b, Sivarethinamohan Rajamanickam c, Karthick Kanagarathinam d, Gasim Hayder a e, Gorti Janardhan f, Priya Arunkumar g, Sivakumar Arunachalam h, Abeer A. AlObaid i, Ismail Warad j k, Senthil Kumar Muniasamy.

Institute of Energy Infrastructure, Universiti Tenaga Nasional (UNITEN), 43000, Kajang, Selangor Darul Ehsan, Malaysia.

Department of Electrical and Electronics Engineering, Panimalar Engineering College, Chennai, India.

Symbiosis Centre for Management Studies (Constituent of Symbiosis International Deemed University), Bengaluru, 560 100, Karnataka, India.

Department of Chemical Engineering, KPR Institute of Engineering and Technology, Coimbatore, 641 407, India

Department of Biotechnology, Karpaga Vinayaga College of Engineering and Technology, Chengalpattu, 603308, Tamil Nadu, India.

**DOI :** [**https://doi.org/10.1016/j.envres.2023.117354**](https://doi.org/10.1016/j.envres.2023.117354)

**KEYWORDS:** Air pollution ,Air quality index ,Machine learning ,Deep learning.

**URL :** [**https://www.sciencedirect.com/science/article/abs/pii/S0013935123021588**](https://www.sciencedirect.com/science/article/abs/pii/S0013935123021588)