Optimal Machine Learning Techniques to Predict Air Quality Index

Report submitted to SASTRA Deemed to be University
As per the requirement for the course

CSE300: MINI PROJECT

Submitted by

HEMANTH BABU CHAVA
(Reg No.: 125003105, B. Tech Computer Science and Engineering)
MANJUNATH P
(Reg. No.: 125003174, B. Tech Computer Science and Engineering)
PRAVANTH DEVAKI
(Reg. No.: 125003232, B. Tech Computer Science and Engineering)

MAY 2024

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act, 1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

SCHOOL OF COMPUTING

THANJAVUR – 613 401

<u>Bonafide Certificate</u>

This is to certify that the report titled **“Optimal Machine Learning Techniques to Predict Air Quality Index”** submitted as a requirement for the course, **CSE300 : MINI PROJECT** for B.Tech. is a bonafide record of the work done by **Mr. HEMANTH BABU CHAVA** (Reg. No.: 125003105, B. Tech Computer Science and Engineering), **Mr. MANJUNATH P** (Reg. No.: 125003174, B. Tech Computer Science and Engineering) and **Mr. PRAVANTH DEVAKI** (Reg. No.: 125003232, B. Tech Computer Science and Engineering) during the academic year 2023-24, in the School of Computing, under my supervision.

Signature of Project Supervisor	: [illegible]
Name with Affiliation	: Dr.M.SUMATHI, AP-III, SOC
Date	: 17.04.2024

Mini Project *Viva voice* held on ________________

Examiner 1 **Examiner 2**

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. Algeswaran**, Associate Dean, Students Welfare

Our guide **Dr. M SUMATHI** , Assistant Professor - III, School of Computing was the driving force behind this whole idea from the start. Her 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 us an opportunity to showcase our skills through this project.

List of Figures

List of Tables

Abbreviations

ML	Machine Learning
DL	Deep Learning
XGBoost	Extreme Gradient Boosting
LightGBM	Light Gradient Boosting Machine
CPCB	Central Pollution Control Board
AQI	Air Quality Index
EDA	Exploratory Data Analysis
MAE	Mean Absolute Error
RMSE	Root Mean Square Error
MSE	Mean Square Error
R^2	Coefficient of determination
SVM	Support Vector Machine
LSTM	Long Short Term Memory
SARIMA	Seasonal Auto Regressive Integrated Moving Average
ARIMA	Auto Regressive Integrated Moving Average

ABSTRACT

The rapid growth in various sectors has resulted in growing worry regarding air quality degradation in the modern world. Monitoring pollutant levels is crucial for regulating concentrations and taking prompt actions when pollution rises, allowing authorities to maintain control over environmental quality. To maintain ambient air quality, regular monitoring and forecasting of air pollution is necessary and this can be achieved by predicting the Air Quality Index (AQI). The AQI is a numerical scale that communicates the quality of air in a specific location based on the concentrations of various air pollutants. Manual monitoring stations to gather pertinent data entails significant expenses for both installation and maintenance. Since regular monitoring and prediction of the AQI are critical for combating air pollution. To accomplish this objective, machine learning (ML) has surfaced as a hopeful approach for predicting the AQI when compared to traditional methods. In this work, to predict the AQI different ML algorithms are used such as ensemble methods based Bagging and Boosting. The open-source data from the Central Pollution Control Board (CPCB), covering a certain period of time in a region is going to be analysed. The contaminants like Particulate Matter, gaseous pollutants and key meteorological parameters that contribute to the AQI are considered for analysis. The available dataset is pre-processed and subjected to Exploratory Data Analysis with required transformations, before splitting it into Training and Testing sets. The main emphasis of this work is to figure out the most effective ML framework that is scalable and reusable for any city of choice with the data from CPCB. To arrive at this, the various performance metrics (like correlation coefficient, MAE, MSE, RMSE) of the different models used in predicting the AQI values and other factors like generalization ability, robustness to outliers, flexible model customization and reusability are going to be analysed.

KEY WORDS: Machine Learning, AQI prediction, Bagging, Boosting, Ensemble methods

Table of Contents

CHAPTER 1

SUMMARY OF BASE PAPER

Title	:	Impact of air pollutants on climate change and prediction of air quality index using machine learning models
Publisher	:	Elsevier
Year	:	2023
Journal	:	Environmental Research Volume 239, Part 1, Article 117354
Indexing	:	SCI / Scopus
Base paper URL	:	https://www.sciencedirect.com/science/article/pii/S0013935123021588

1.1 INTRODUCTION

Air contamination stemming from industry, transportation, and agriculture threatens public health in rapidly developing countries like India. Factors like gaseous pollutants, meteorological parameters and particulate matter affect air quality in any region. Consistent monitoring and prediction of air pollution levels are imperative to uphold ambient air quality standards. AQI is a metric utilized to assess air pollution levels in a given place based on pollutants like particulate matters ($PM_{2.5}$ and PM_{10}), Carbon dioxide (CO_2), Ammonia (NH_3), benzene, Sulphur dioxide (SO_x), Nitrous Oxide (NO_x), Nitrogen dioxide (NO_2), volatile organic compounds (VOCs), Carbon monoxide (CO) and Ozone. Elevated levels of AQI resulting from these substances detrimentally affect the environment through various avenues, such as contributing to global warming, the formation of acid rain, the emergence of smog and aerosols, reduced visibility, and alterations in climate patterns. Hence, monitoring and forecasting a region's AQI values becomes essential for better strategies to tackle pollution levels and sustainable development for modernization. In this context, IQAir publishes world rankings of 134 countries yearly based on the $PM_{2.5}$ concentrations ($\mu g/m^3$). In 2023, India ranked as the third worst country with a $PM_{2.5}$ concentration of 54.4 ($\mu g/m^3$), which exceeds the WHO guideline by over ten times. A high AQI value indicates a highly hazardous environment for both people and other life forms, posing a significant risk to their well-being and safety. Advancements in sensor technology have simplified the identification of various levels of air pollution, with the AQI now being automatically calculated. With readily accessible datasets, forecasting the AQI has become a straightforward process. In this study, the authors employ Machine Learning (ML) techniques like ensemble methods to find an optimal algorithm that accurately predicts AQI in a given region.

1.2 RELATED WORK

Numerous researchers have developed AQI prediction models, employing both ML and DL techniques, drawing from statistical analysis and dynamic, forward-looking approaches to accurately forecast AQI levels. ML methods extend beyond statistical analysis, establishing models by leveraging the correlations among independent variables utilized in AQI predictions. Researchers have utilized a variety of machine learning algorithms for AQI predictions, including LR, MLR, Decision Tree Regression (DTR), Random Forest Regression (RFR), Support Vector Regression (SVR), Support Vector Machine (SVM), K-Nearest Neighbors Algorithm (KNN), Gaussian Naive Bayes (GNB), Artificial Neural Network (ANN), Long Short-Term Memory (LSTM), Improved Long Short-Term Memory (ILSTM), Convolution Neural Networks (CNN), Adaptive Boosting (Adaboost), Xgboost, Catboost Regression (CR), and SARIMA. In this project, we employ the four ML models namely Random Forest Regressor, Extreme Gradient Boosting algorithm (XGBoost), Bagging Regressor and Light Gradient Boosting Machine (LGBM).

Various other publications were explored that utilize ML algorithms to predict AQI values. Below are some of them with their respective merits and demerits:

- **"Air Quality Index prediction using Machine Learning for Ahmedabad City"** by Nilesh N. Maltare, Safvan Vahora. In this paper, AQI prediction is implemented using ML algorithms like SARIMA, SVM, and LSTM. Many data-preprocessing methods are presented to remove the outliers and normalize the datasets, which are extracted from different sources (CPCB boards). They can also be expanded to forecast other pollution indices at different levels. A major limitation was that lots of missing values are present in the dataset of Ahmedabad City which were not adequately addressed.

- **"Air Quality Index prediction using Machine Learning Algorithms ~International Journal of computer Applications Technology and Research"** by Pooja Bhalgat et al. The authors have implemented ARIMA, Auto Regression and Linear Regression achieving successful prediction and comprehensive time series analysis was also used for the recognition of future data points and air pollution prediction. But not able to show expected output as the data is not in sequence. The error is high which they are working to overcome in near future.

- **"Indian Air Quality Prediction And Analysis using Machine Learning " by** Mrs. A. Gnana Soundari MTech, Mrs. J. Gnana. The models used for the work are Naïve Forest, Linear Regression and Gradient Boosting algorithms. As a merit, parameter-reducing formulations for better performance than standard regression models. But the downside was low accuracy.

- **"Air pollution prediction using machine learning techniques – An approach to replace existing monitoring stations with virtual monitoring stations"** by A.

Samad, S. Garuda, U. Vogt, B. Yang. The paper uses Ridge regression, SVR, Random Forest, Xtreme Gradient Boosting methods with hyperparameter tuning so that the developed technique can be transferred to any location where prediction is required. This study's limitation lies in the inability to forecast pollutant concentration due to the necessity of data from additional monitoring stations for prediction.

- **"Detection and Prediction of Air Pollution using Machine Learning Models"** by C R, Aditya & Deshmukh, Chandana & K, Nayana & Gandhi, Praveen & astu, Vidyav. Mean accuracy and standard deviation accuracy was 0.998859 and 0.000612 respectively and the paper used Logistic and Auto Regression but very less data was taken.

- **"Air quality prediction by machine learning models: A predictive study on the indian coastal city of Visakhapatnam"** by Gokulan Ravindiran, Gasim Hayder, Karthick Kanagarathinam, Avinash Alagumalai, Christian Sonne. CatBoost model yielded high prediction accuracy (0.9998) and low RMSE (0.76). CatBoost incorporates parameters that help mitigate overfitting in datasets. Performance needs to be validated under diverse air quality conditions. The covid lockdown has affected the AQI levels

1.3 PROBLEM STATEMENT

As we aim to enhance accuracy in AQI forecasting and facilitate informed decision-making for environmental regulation, our goal is to provide timely warnings and precautions to the public regarding air quality levels. Through enabling proactive measures for mitigating the influence of air pollution on public health, we can effectively safeguard communities from its adverse effects.

1.4 OBJECTIVE

We aim to implement advanced machine learning techniques on CPCB data to enhance the accuracy and reliability of AQI prediction. By leveraging sophisticated algorithms, we seek to develop models that outperform existing methods, thus validating their effectiveness in air quality forecasting. Through rigorous evaluation of the models' performance against established benchmarks, we aim to showcase the potential of our approach to optimize environmental management strategies. Ultimately, our goal is to safeguard public health by providing more accurate and timely information to mitigate the hazards of air pollution effectively.

1.5 PROPOSED SOLUTION AND SYSTEM ARCHITECTURE

1.5.1 Study area

The current study explored the air quality in Tirupati City, located in the Andhra Pradesh state of India. Tirupati hosts an air quality monitoring station operated by India's CPCB. Situated in the southern region of Andhra Pradesh, Tirupati lies between approximately 130° 38′ and 130° 54′ North latitude and 790° 12′ and 790° 28′ East longitude. Covering an area of approximately 27.44 km², it stands as one of the largest cities in Andhra Pradesh, characterized primarily as a spiritual city. The city is home to the important Hindu shrine of Tirumala Venkateshwara Temple and other historic temples. Tirupati experiences a tropical climate characterized by hot and humid conditions throughout the year. Regarding air quality, Tirupati, like many urban areas in India, faces challenges due to pollution originating from diverse sources, including vehicular emissions, industrial activities, and urban development. The air quality can vary depending on traffic density, industrial operations, and meteorological conditions. Despite these efforts, maintaining consistently good air quality remains a significant challenge for the city. The geographical location of Tirupati is shown in Fig 1.1

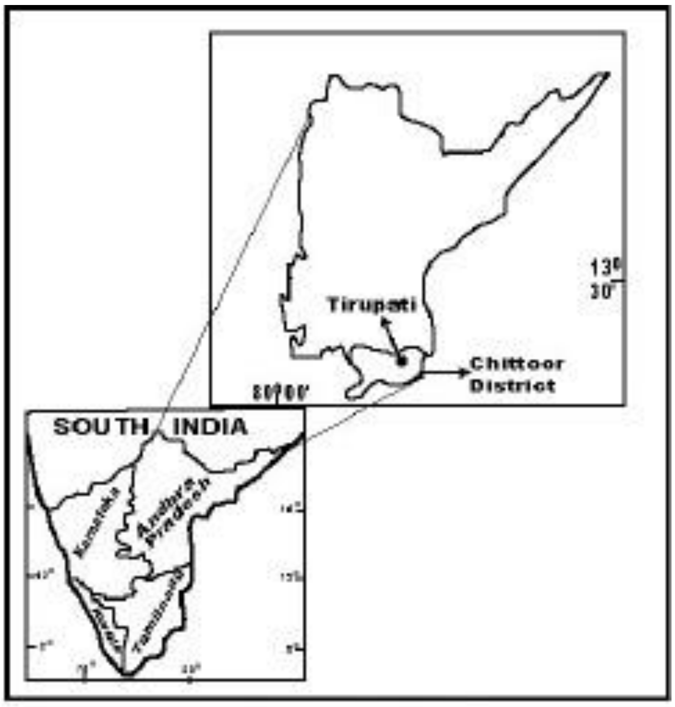


Fig 1.1.Location of Tirupati City

1.5.2 Air quality and meteorological datasets

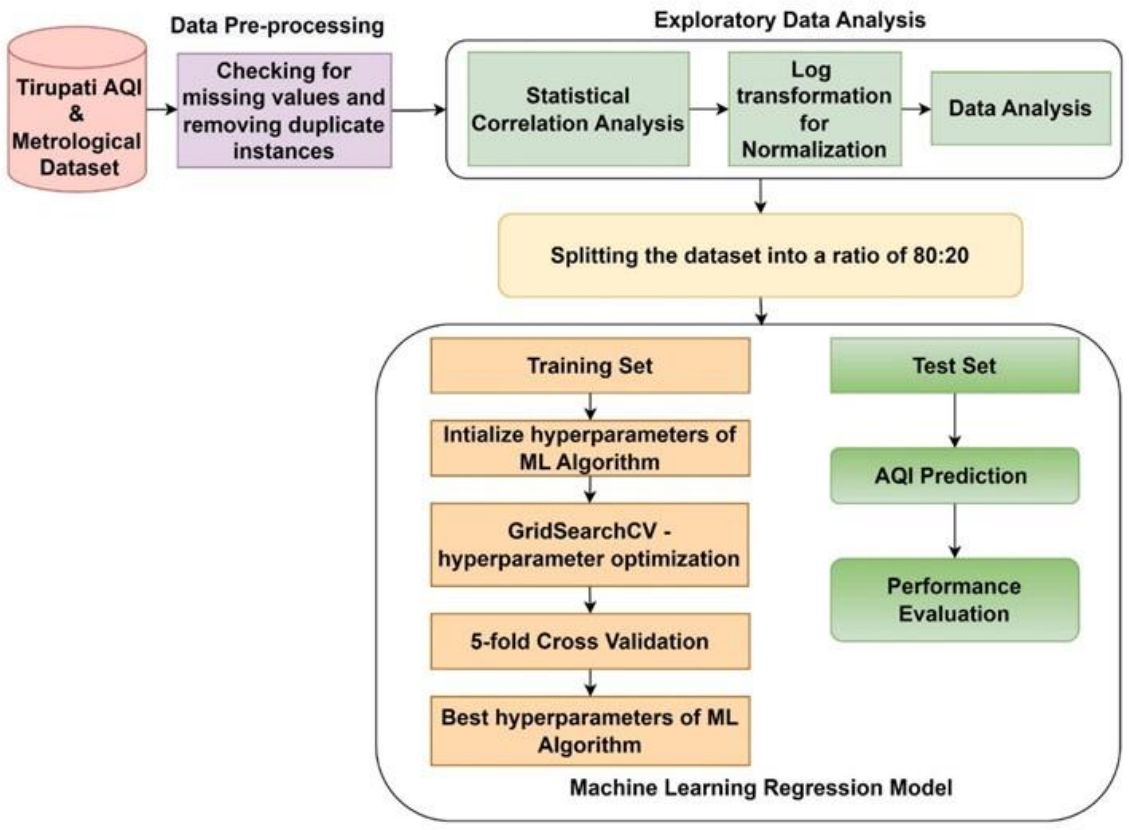
The dataset for the years 2017 to 2022 is taken from an open-source CPCB website (link source: https://airquality.cpcb.gov.in/ccr/#/caaqm-dashboard-all/caaqm-landing/aqi-repository) maintained by the Government of India and constantly updated with real-time pollutant concentrations and AQI values every day. Each of the pollutants and other parameters are described in the Table 1.1Table 1.1 AQI dataset description

Table 1.1 AQI dataset description

S.No	Features	Description
1	City	Tirupathi,Andhra Pradesh
2	Date	1/1/2017-31/31/2022
3	$PM_{2.5}$	particulate matter with diameters of 2.5 micrometers
4	PM_{10}	particulate matter with diameters of 10 micrometers
5	NO	Nitrogen Monoxide , released by Industrial combustion process
6	NO_2	Nitrogen Dioxide released through oxidation of NO
7	NO_x	Group of NO and NO2
8	NH_3	Ammonia, Released from Agriculture activities, Animal Husbandry
9	CO	Carbon Monoxide,released from Fires
10	SO_2	Sulphur Dioxide released from Automobiles
11	Benzene	Coal ,Oil burning ,Tobacco smoking are causes of air pollutants
12	Toluene	Pollutant released by Motor Vehicles
13	Ozone	Ozone, Released form industries
14	RH	Relative humidity denotes the proportion of water vapor at a given temperature
15	Xylene	Coal,Wood Burning are the causes of xylene
16	AT	Ambient Temperature
17	RF	RainFall
18	WS	Windspeed
19	WD	Wind Direction
20	Temp	Temparature
21	BP	Air Pressure (Barometric Pressure)
22	SR	Solar Radiation
23	AQI	Air Quality Index

1.5.3 Work Flow Diagram

The dataset taken from the CPCB website were CSV files containing data pre-processed to check for missing values and handle null instances. We explore the efficiency of different imputation and filling techniques to arrive at the best method for higher accuracy. Then Exploratory Data Analysis (EDA) is performed to get insights and uncover patterns in the historical data. Further, an 80:20 split is done on the dataset, and different ML models are fitted to compare against their performance. The work flow diagram is depicted in Fig 1.2



Tirupati AQI & Metrological Dataset
Data Pre-processing
Checking for missing values and removing duplicate instances
Exploratory Data Analysis
Statistical Correlation Analysis
Log transformation for Normalization
Data Analysis
Splitting the dataset into a ratio of 80:20
Training Set
Intialize hyperparameters of ML Algorithm
GridSearchCV - hyperparameter optimization
5-fold Cross Validation
Best hyperparameters of ML Algorithm
Test Set
AQI Prediction
Performance Evaluation
Machine Learning Regression Model

Fig 1.2.Workflow Diagram

1.6 METHODOLOGY AND IMPLEMENTATION

The methodology is segregated into three modules for proper implementation. The pictorial representation of this module segregation is given in Fig 1.3

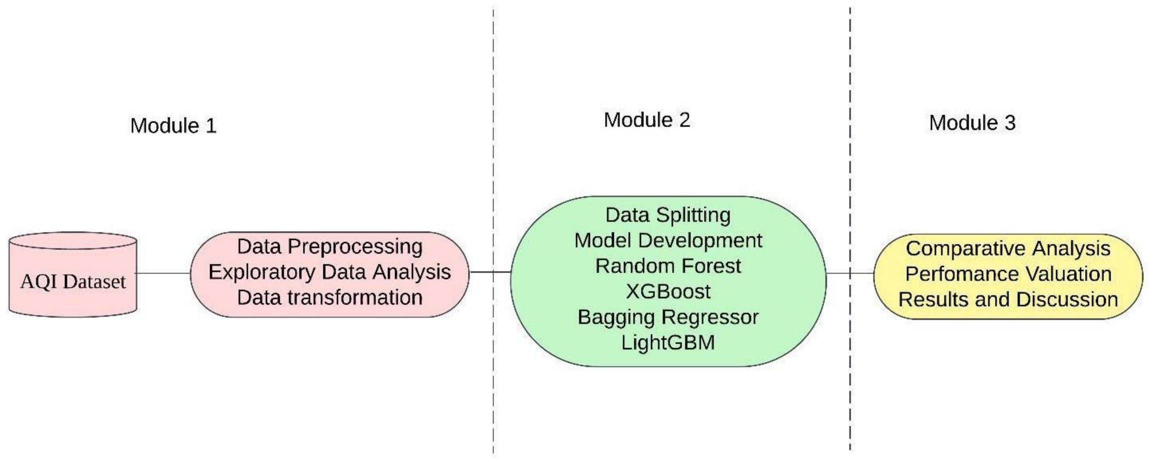


Fig 1.3.Implementation modules

1.6.1 MODULE 1: DATA PREPROCESSING AND EDA

Module 1 covers the DataPreprocessing, EDA, Data transformation. Initially, the collected data underwent preprocessing to eliminate noise and handle any missing values effectively. Following this, an in-depth exploration of the data was conducted to identify underlying patterns and insights, utilizing appropriate tools such as correlation matrices. Subsequently, the data was subjected to transformation and standardization processes before being inputted into the chosen models for training and testing.

The datasets obtained from CPCB website were raw and had many inconsistencies. The data was analysed using python libraries and the missing values were handled properly. Used forward fill (previous day values) for null and 'None' values. For novelty, we also tried to include rows with no values for target variable by filling them with monthly average and choosing the closest row based on Euclidean distance.

In the EDA part, we get the heatmap illustrating correlation analysis among the independent variables and also try to understand the underlying patterns and trends in the dataset. Skewness serves as a metric to evaluate the symmetry or asymmetry of datasets. When datasets are evenly distributed around the centre (from left to right), they are considered symmetrical. Kurtosis, on the other hand, is employed to assess whether the data follows a normal distribution, indicating whether the tails of the distribution are strongly or weakly tapered.

1.6.2 MODULE 2: DATA SPLIT AND MODEL DEVELOPMENT

1.6.2.1 Data splitting and standardization

Module 2 involves Data split and fitting the ML models. The methodologies employed in our study encompassed various models and techniques, including Random Forest, XGBoost, Bagging Regressor, and LGBM Regressor, implemented using Python libraries like numpy, pandas, seaborn, and scikit-learn's model selection module, along with GridSearchCV for hyperparameter tuning. The ensemble methods were applied to three different versions of datasets arrived at after filling in missing target variable rows, i.e., the removed version, filling with the monthly average and choosing the value of the closest row based on Euclidean distance.

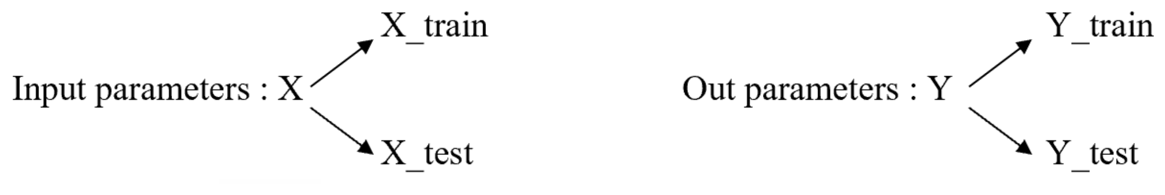


Fig 1.4.Data split

We utilize the sklearn.model_selection module to split the data into two distinct sets: training and testing, using the train_test_split() function. The dataset is divided into 80% for training and 20% for testing. An inadequately selected split can lead to the model being overfit or underfit, which may lead to inadequate predictions on unseen data. Hence, it is crucial to select the split carefully and evaluate the model's performance on the test data to guarantee that it can generalize well to new data. To normalize the features of the training and testing sets, we use the StandardScaler() function from the sklearn module.

1.6.2.2 Random Forest Regressor

The RF algorithm uses the bagging technique for building an ensemble of decision trees. Bagging is known to reduce the variance of the algorithm. Implemented via scikit-learn, we use classes like RandomForestRegressor for model creation and k-fold for cross-validation. The RandomForestRegressor object is initialized with preset hyperparameters: max_depth of 15, max_features set to 'auto', min_samples_leaf at 1, min_samples_split of 2, and n_estimators of 100, alongside a designated random_state.

After bagging multiple decision trees, an ensemble technique is employed to combine these trees into a unified prediction. Out of the total set of N training examples (D), n random instances are selected to form a bootstrap sample (D_b). The process of creating bootstrap samples allows for the replacement of examples. Each distinct regression tree is constructed using the input vector x for the bootstrap samples. In regression tasks, the prediction generated by a random forest model is obtained by calculating the average of the predictions of K regression trees h_k (x).

$$Random\ forest\ prediction = \frac{1}{K}\sum_{k=1}^{K} h_k\,(x) \qquad (1)$$

1.6.2.3 XGBoost Algorithm

XGBoost, rooted in gradient boosting, employs an ensemble learning approach where weak learners (usually decision trees) are sequentially trained, with each1 learning from the errors of its predecessors. After training, the best hyperparameters were determined to be 'learning_rate': 0.1, 'max_depth': 5, and 'n_estimators': 100. These optimized settings allowed the XGBoost model to achieve its highest performance when predicting the AQI.

Effective data organization is paramount when utilizing XGBoost. All categorical data must be transformed into numeric representations since XGBoost exclusively accepts numeric vectors as input. This conversion can be achieved through one-hot encoding. Following this, the process proceeds to feature engineering and data refinement stages. Finally, the estimated model could be obtained using the universal function, as specified using the formula provided.

$$y_i^t = \sum_{k=1}^{t} f_k\,(x_i) = y_i^{(t-1)} + f_t(x_i) \qquad (2)$$

where,

y_i^t = forecasts at the stage t

$f_t(x_i)$ = a learner at stage t

x_i = the input variable

$y_i^{(t-1)}$ = forecasts at the stage t-1

1.6.2.4 Bagging Regressor

A Bagging Regressor works by training multiple base regressors on different subsets of training data and averaging their predictions to improve accuracy and robustness. It reduces variance, improves generalization, and enhances robustness by combining the predictions of individual models. Import the necessary classes from the scikit-learn library, including the Bagging Regressor class, and Kfold classes and GridSearchCV for cross-validation and hyperparameter tuning respectively. The best hyperparameters are found to be 'n_estimators': 300 and 'max_samples': 0.9 .Finally, the bagging regression model is utilized to predict the target variable on the test set, and its efficacy is assessed.

1.6.2.5 LightGBM Algorithm

LGBM Regressor stands for LightGBM Regressor, which is a gradient boosting framework employing tree-based learning algorithms. It's designed to be efficient and scalable, particularly for large datasets. In supervised learning scenarios, this approach allows for extraction of information about a target variable Y exclusively based on input X. Employing the LightGBM technique involves utilizing a supervised training set (X) and a loss function L (y,f(x)), with the objective of minimizing the predicted value to achieve $f^{\wedge}\,(x)$

$$f^{\wedge}(x) = \arg min_f\; E_{y,X}\, L\,(y, f(x)) \qquad (3)$$

1.6.3 MODULE 3: MODEL PERFORMANCES AND COMPARATIVE ANALYSIS

Module 3 consists of Comparative Analysis, Performance valuation, Results and Discussion. To ensure the robustness and reliability of our models, we employed a 5-fold cross-validation technique. For each fold, we computed key evaluation metrics, including MAE, MSE, RMSE, and R^2. These metrics were then aggregated across all folds, providing a comprehensive assessment of model performance. Generally, models exhibiting higher R^2 scores along with lower MAE and MSE scores are considered to perform better. Frontend integration is done to the project with a Graphical User Interface (GUI) to showcase the AQI predictions.

CHAPTER 2

MERITS AND DEMERITS OF BASE PAPER

2.1 MERITS

The proposed work in base paper has some really good implementations and credibility few of the merits are:

- Opted for traditional ML algorithms instead of DL since the datasets were too small to prevent overfitting.
- The domain was thoroughly explored and the relevant features and factors were considered before fitting the models.
- Choosing the right data exploration methods was also an added advantage
- The performance comparisions are clearly documented and excellent analysis was made in the conclusion.
- The future work suggesting forecasting the AQI for years was also suggested
- Hyperparameter tuning using GridSearchCV was done to find the best hyperparameters for fitting a model.
- Cross-Validation was also conducted in conjunction with the hyperparameter tuning.
- 5-fold validation resulted in a proper analysis and inference

2.2 DEMERITS

Thought the base paper was extensive, there were some areas where it could have been improved:

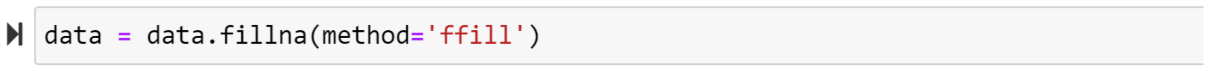
- Trying other Boosting algorithms like CatBoost and AdaBoost which might have performed better.
- Ignored the missing valued rows altogether instead of trying any imputation methods.

CHAPTER 3

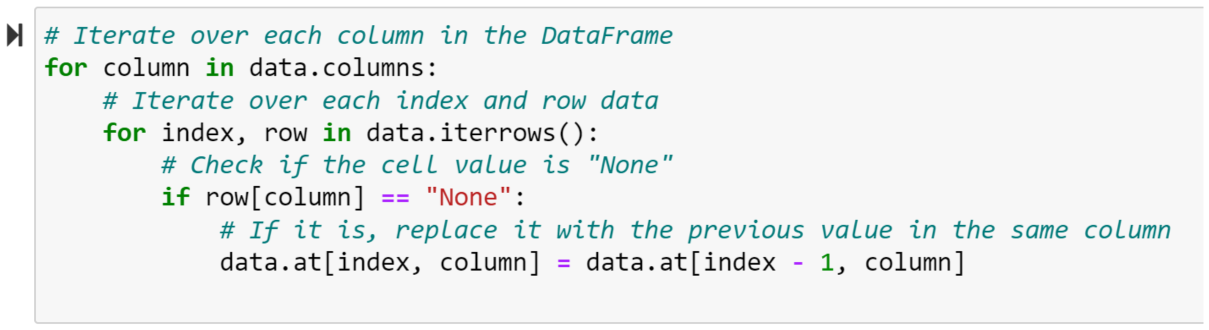
SOURCE CODE

3.1 MODULE 1: DATA PREPROCESSING AND EDA

3.1.1 Data Preprocessing

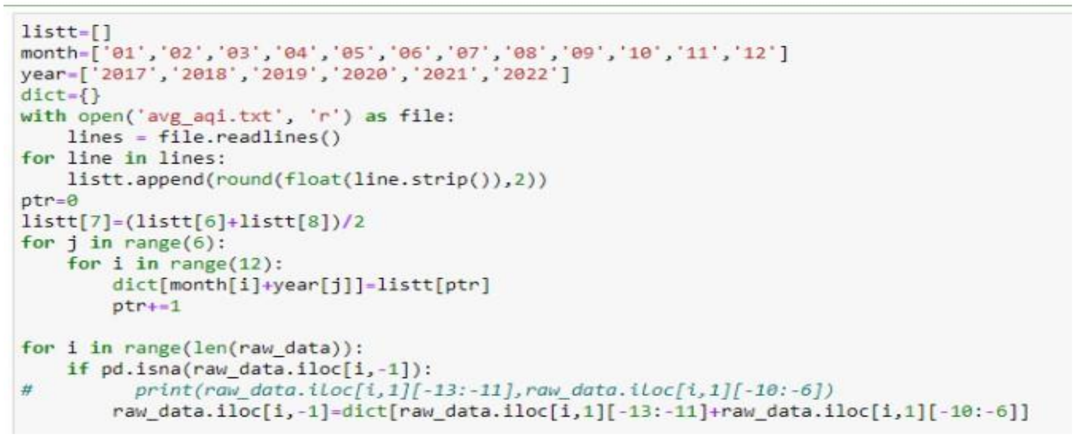


```python
data = data.fillna(method='ffill')
```



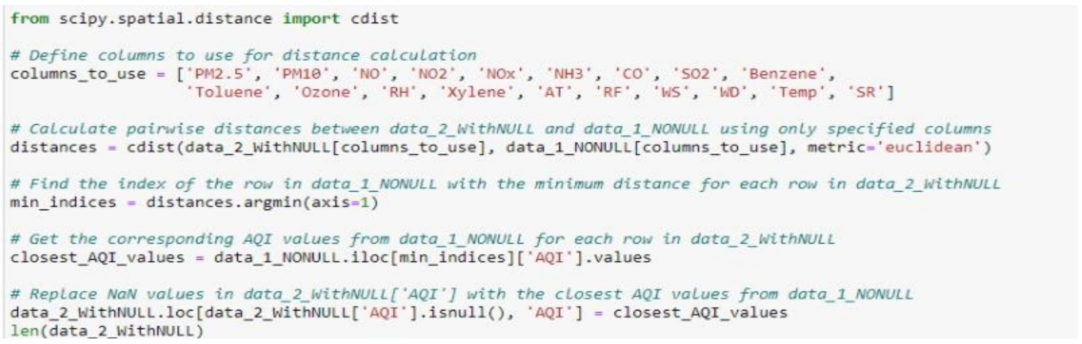
```python
# Iterate over each column in the DataFrame
for column in data.columns:
    # Iterate over each index and row data
    for index, row in data.iterrows():
        # Check if the cell value is "None"
        if row[column] == "None":
            # If it is, replace it with the previous value in the same column
            data.at[index, column] = data.at[index - 1, column]
```

Code to fill with monthly average values



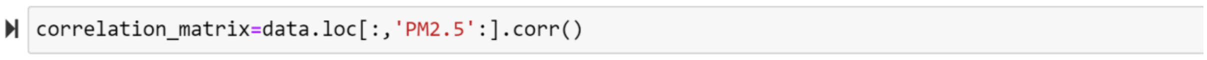
```python
listt=[]
month=['01','02','03','04','05','06','07','08','09','10','11','12']
year=['2017','2018','2019','2020','2021','2022']
dict={}
with open('avg_aqi.txt', 'r') as file:
    lines = file.readlines()
for line in lines:
    listt.append(round(float(line.strip()),2))
ptr=0
listt[7]=(listt[6]+listt[8])/2
for j in range(6):
    for i in range(12):
        dict[month[i]+year[j]]=listt[ptr]
        ptr+=1

for i in range(len(raw_data)):
    if pd.isna(raw_data.iloc[i,-1]):
#           print(raw_data.iloc[i,1][-13:-11],raw_data.iloc[i,1][-10:-6])
        raw_data.iloc[i,-1]=dict[raw_data.iloc[i,1][-13:-11]+raw_data.iloc[i,1][-10:-6]]
```

Code to fill with closest row value based on Euclidean distance

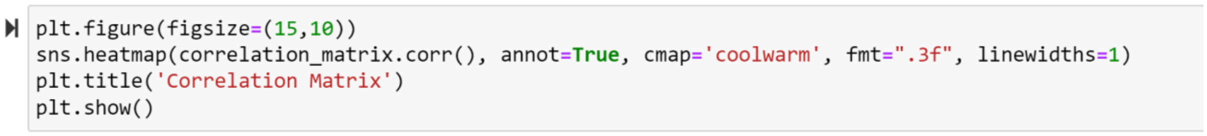


```
from scipy.spatial.distance import cdist

# Define columns to use for distance calculation
columns_to_use = ['PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO', 'SO2', 'Benzene',
                  'Toluene', 'Ozone', 'RH', 'Xylene', 'AT', 'RF', 'WS', 'WD', 'Temp', 'SR']

# Calculate pairwise distances between data_2_WithNULL and data_1_NONULL using only specified columns
distances = cdist(data_2_WithNULL[columns_to_use], data_1_NONULL[columns_to_use], metric='euclidean')

# Find the index of the row in data_1_NONULL with the minimum distance for each row in data_2_WithNULL
min_indices = distances.argmin(axis=1)

# Get the corresponding AQI values from data_1_NONULL for each row in data_2_WithNULL
closest_AQI_values = data_1_NONULL.iloc[min_indices]['AQI'].values

# Replace NaN values in data_2_WithNULL['AQI'] with the closest AQI values from data_1_NONULL
data_2_WithNULL.loc[data_2_WithNULL['AQI'].isnull(), 'AQI'] = closest_AQI_values
len(data_2_WithNULL)
```

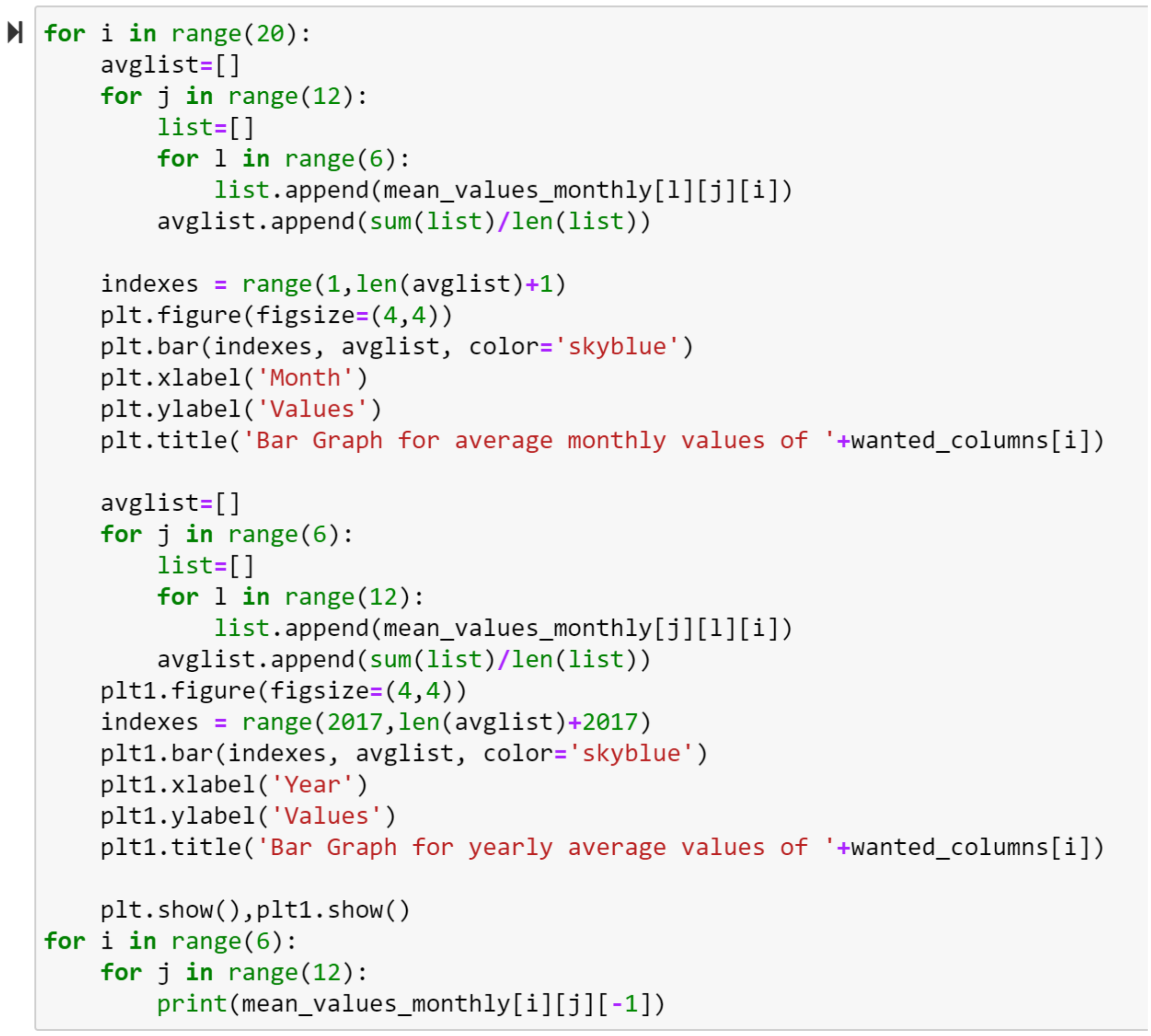
3.1.2 Exploratory Data Analysis



```
correlation_matrix=data.loc[:,'PM2.5':].corr()
```



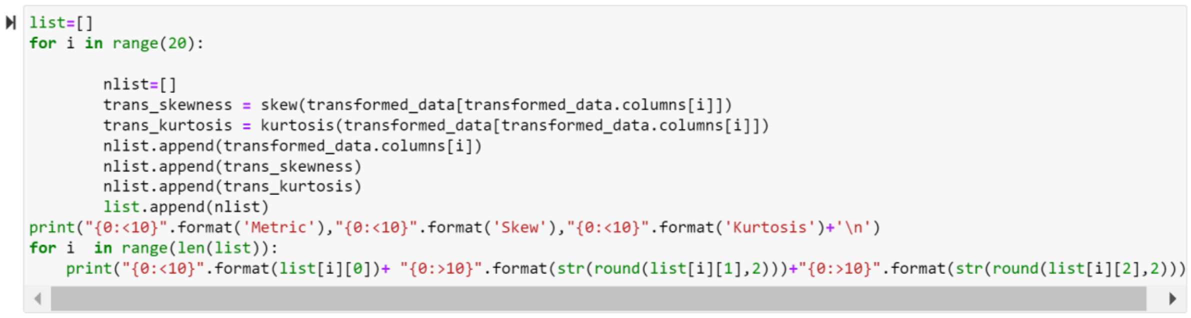
```
plt.figure(figsize=(15,10))
sns.heatmap(correlation_matrix.corr(), annot=True, cmap='coolwarm', fmt=".3f", linewidths=1)
plt.title('Correlation Matrix')
plt.show()
```



```
for 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])
```

3.1.3 Data Transformation

Skewness After Log Transformation

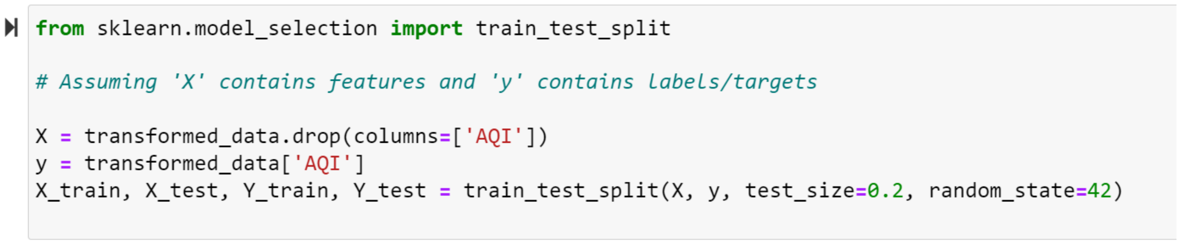


```
list=[]
for i in range(20):

        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)))
```

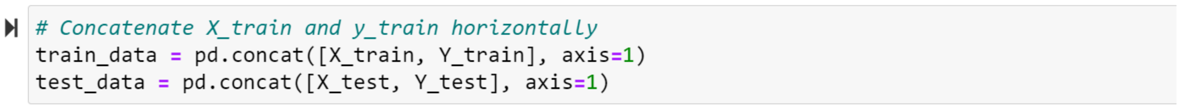
3.2 MODULE 2: DATA SPLIT AND MODEL DEVELOPMENT

3.2.1 Data Split and Standardization

Data Splitting

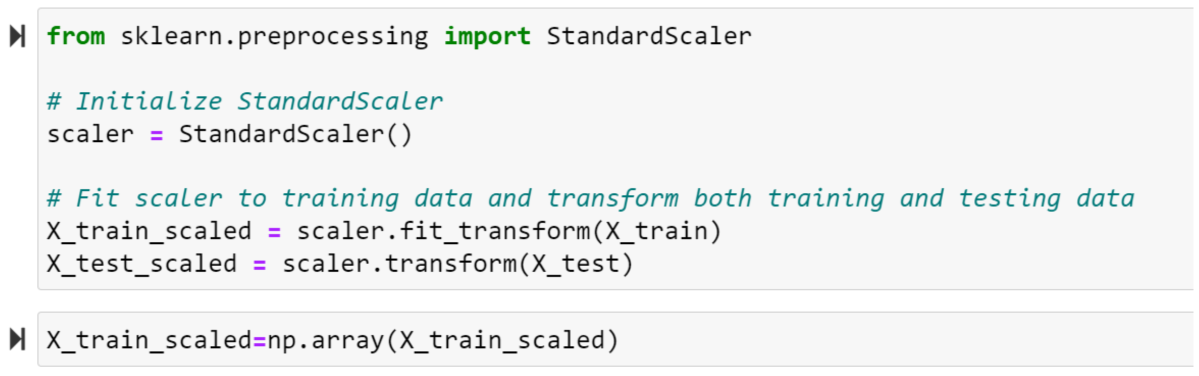


```
from sklearn.model_selection import train_test_split

# Assuming 'X' contains features and 'y' contains labels/targets

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)
```



```
# Concatenate X_train and y_train horizontally
train_data = pd.concat([X_train, Y_train], axis=1)
test_data = pd.concat([X_test, Y_test], axis=1)
```

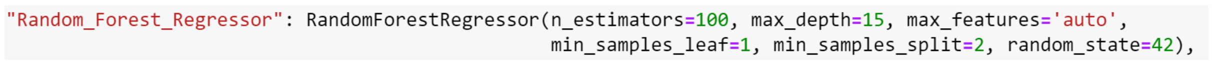
Data Standardisation



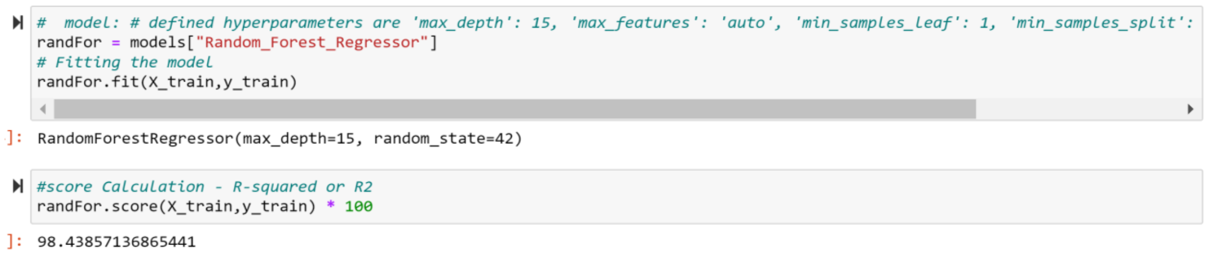
```
from sklearn.preprocessing import StandardScaler

# Initialize StandardScaler
scaler = StandardScaler()

# Fit scaler to training data and transform both training and testing data
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
X_train_scaled=np.array(X_train_scaled)
```

3.2.2 Random Forest Regressor



```
"Random_Forest_Regressor": RandomForestRegressor(n_estimators=100, max_depth=15, max_features='auto',
                                                 min_samples_leaf=1, min_samples_split=2, random_state=42),
```



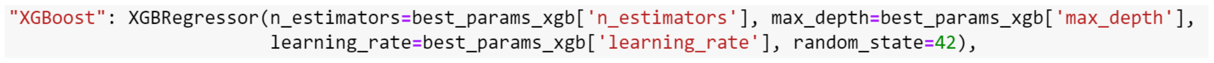
```
#  model: # defined hyperparameters are 'max_depth': 15, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split':
randFor = models["Random_Forest_Regressor"]
# Fitting the model
randFor.fit(X_train,y_train)
```

```
]:  RandomForestRegressor(max_depth=15, random_state=42)
```

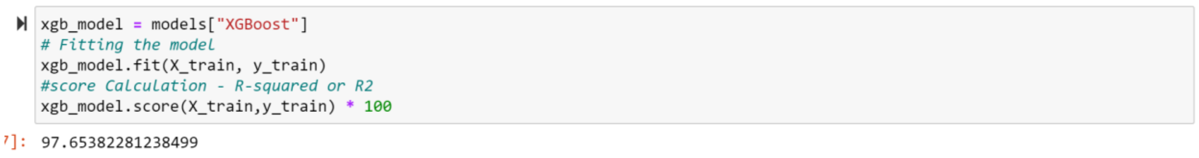
```
#score Calculation - R-squared or R2
randFor.score(X_train,y_train) * 100
```

```
]:  98.43857136865441
```

3.2.3 XGBoost Algorithm



```
"XGBoost": XGBRegressor(n_estimators=best_params_xgb['n_estimators'], max_depth=best_params_xgb['max_depth'],
                        learning_rate=best_params_xgb['learning_rate'], random_state=42),
```



```
xgb_model = models["XGBoost"]
# Fitting the model
xgb_model.fit(X_train, y_train)
#score Calculation - R-squared or R2
xgb_model.score(X_train,y_train) * 100
```

```
7]:  97.65382281238499
```

3.2.4 Bagging Regressor

```
"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"]
#fitting model
bagging_model.fit(X_train, y_train)
#score Calculation - R-squared or R2
r2_train = bagging_model.score(X_train, y_train) * 100
```

```
print(r2_train)
```

```
98.3124086850775
```

3.2.5 LightGBM Algorithm

```
"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)
```

```
lgbm=models["LightGBM"]
# Fitting the model
lgbm.fit(X_train, y_train)
# Calculating R-squared score
# r2_score = lgbm.score(X_train, y_train) * 100
```

```
#train
y_pred_train_lgbm=lgbm.predict(X_train)
# Predict on test set
y_pred_lgbm = lgbm.predict(X_test)
# Calculate mean squared error
mse_lgbm = mean_squared_error(y_test, y_pred_lgbm)
```

3.3 MODULE 3: MODEL PERFORMANCES AND COMPARATIVE ANALYSIS

3.3.1 Comparative analysis

```
# Plotting the normal distribution of true values vs predicted values
plt.figure(figsize=(10, 6))
plt.hist(y_test, bins=20, density=True, alpha=0.5, color='blue', label='True Values')
plt.hist(y_pred, bins=20, density=True, alpha=0.5, color='red', label='Predicted Values')
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, np.mean(y_pred), np.std(y_pred))
plt.plot(x, p, 'k', linewidth=2)
plt.xlabel('Value')
plt.ylabel('Density')
plt.title('True Values vs Predicted Values Distribution')
plt.legend()
plt.show()
```

Comparision of Truevalues vs Predicted Values

```
# Scatter plot of true values vs predicted values
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', alpha=0.5)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='--')
plt.xlabel('True Values')
plt.ylabel('Predicted Values')
plt.title('Random Regressor Model')
plt.grid(True)
plt.show()
```

Comparision of Residuals vs Predicted Values

```
plt.figure(figsize=(10, 6))
plt.scatter(y_pred,y_test-y_pred , color='blue', alpha=0.5)
plt.plot([min(y_pred), max(y_pred)], [min(y_test-y_pred), max(y_test-y_pred)], color='red', linestyle='--')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.title('Random Regressor Model')
plt.grid(True)
plt.show()
```

Frequency Of Residuals

```

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

plt.hist(y_test-y_pred, bins=20, alpha=0.5, label='Random forest regressor residulas Frequency')

plt.title('Residual Histogram of Randomforest Regressor Model')
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.legend()
plt.grid(True)
plt.show()
```

3.3.2 Performance metrics

Score Calculation for training data

```
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))
```

Score Calculation for testing data

```python
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))
```

3.3.3 Graphical User Interface using FLASK

```python
app = Flask(_name_)

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

@app.route('/calculate', methods=['POST'])
def calculate():
    input_data0 = request.form['input0']
    input_data1=request.form['input1']
    input_data2 = request.form['input2']
    result = calculate_result(str(input_data0+','+input_data1+','+input_data2))
    return render_template('result.html', result=result)
if _name_ == '_main_':
    app.run(debug=True)
```

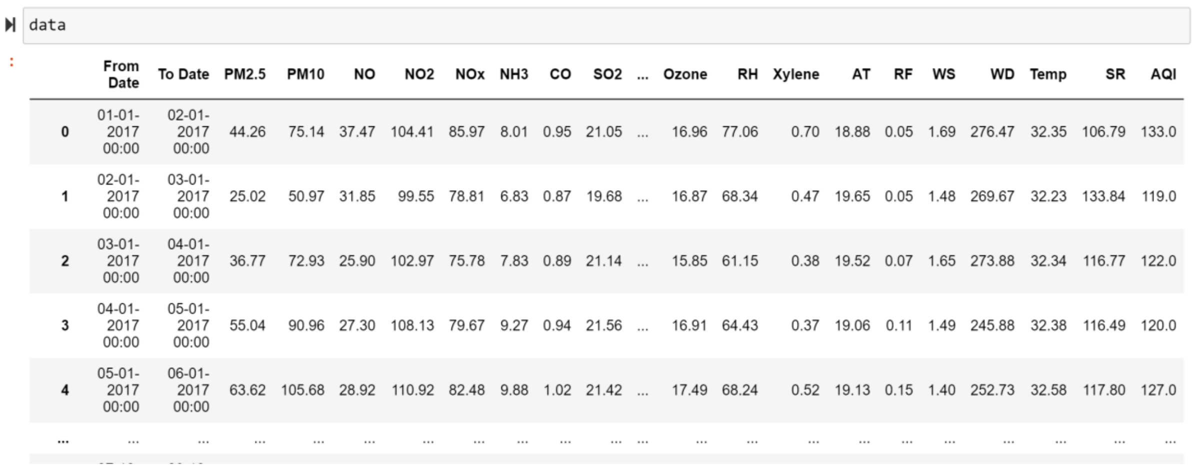
```python
def calculate_result(input_data):
    with open('model.pkl', 'rb') as f:
        model = pickle.load(f)
    input_array = input_data.split(",")
    input_array_1= [np.log(float(x)) for x in input_array]
    input_array_1= np.array(input_array_1)
    input_array_reshaped = input_array_1.reshape(1, -1)
    predictions = model.predict(input_array_reshaped)
    denormalized_predictions = np.exp(predictions[0])
    return denormalized_predictions
```

CHAPTER 4

SNAPSHOTS

4.1 DATA PREPROCESSING AND EDA

4.1.1 Data Preprocessing

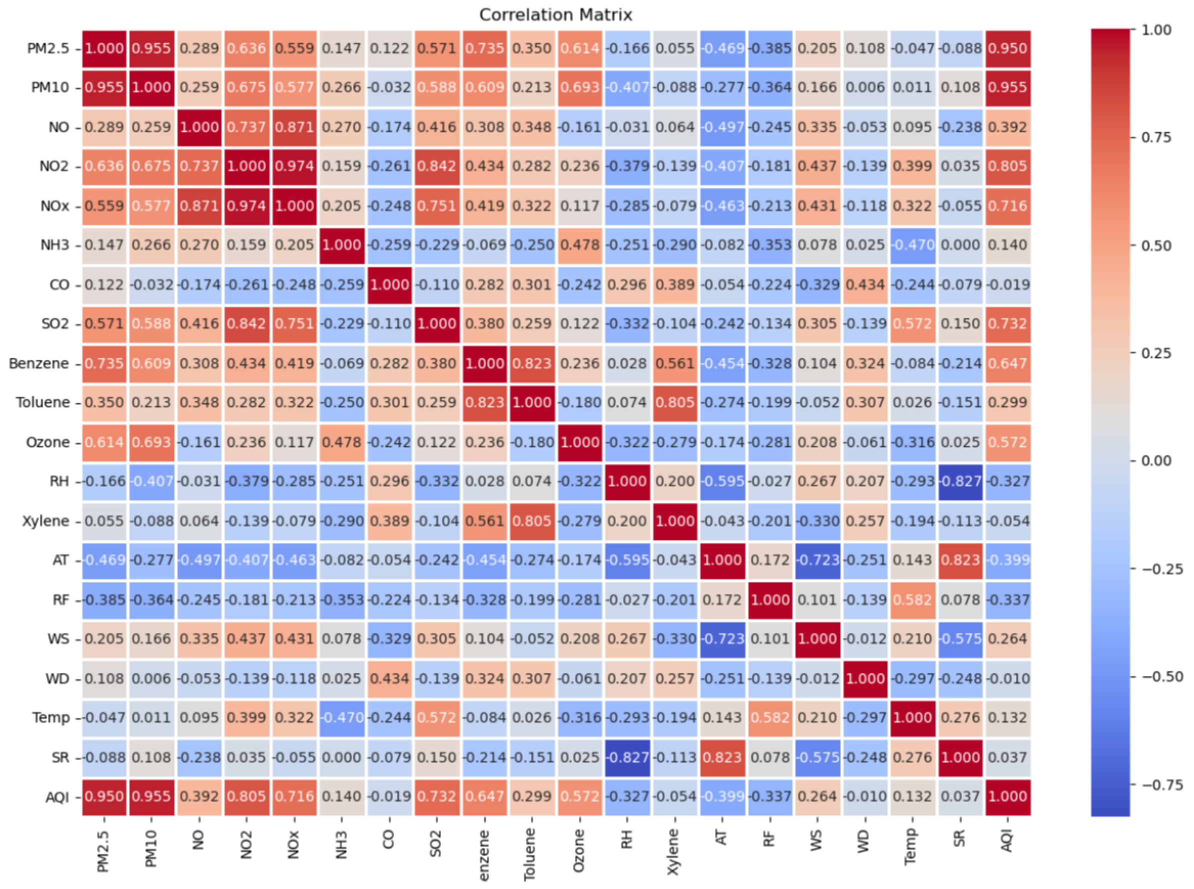


```
data
```

	From Date	To Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	...	Ozone	RH	Xylene	AT	RF	WS	WD	Temp	SR	AQI
0	01-01-2017 00:00	02-01-2017 00:00	44.26	75.14	37.47	104.41	85.97	8.01	0.95	21.05	...	16.96	77.06	0.70	18.88	0.05	1.69	276.47	32.35	106.79	133.0
1	02-01-2017 00:00	03-01-2017 00:00	25.02	50.97	31.85	99.55	78.81	6.83	0.87	19.68	...	16.87	68.34	0.47	19.65	0.05	1.48	269.67	32.23	133.84	119.0
2	03-01-2017 00:00	04-01-2017 00:00	36.77	72.93	25.90	102.97	75.78	7.83	0.89	21.14	...	15.85	61.15	0.38	19.52	0.07	1.65	273.88	32.34	116.77	122.0
3	04-01-2017 00:00	05-01-2017 00:00	55.04	90.96	27.30	108.13	79.67	9.27	0.94	21.56	...	16.91	64.43	0.37	19.06	0.11	1.49	245.88	32.38	116.49	120.0
4	05-01-2017 00:00	06-01-2017 00:00	63.62	105.68	28.92	110.92	82.48	9.88	1.02	21.42	...	17.49	68.24	0.52	19.13	0.15	1.40	252.73	32.58	117.80	127.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

The raw dataset contained a total of 2191 observations spanning from 01-01-2017 to 31-12-2022, which included 20 variables (13 Air pollutant attributes, 06 Meteorological factors, and 1 AQI). Removed rows of instances with null values for the target variable (AQI) and BP (Barometric Pressure) parameter were excluded due to their only 47 instances of recorded data and were not helpful.

4.1.2 Exploratory Data Analysis



	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	enzene	Toluene	Ozone	RH	Xylene	AT	RF	WS	WD	Temp	SR	AQI
PM2.5	1.000	0.955	0.289	0.636	0.559	0.147	0.122	0.571	0.735	0.350	0.614	-0.166	0.055	-0.469	-0.385	0.205	0.108	-0.047	-0.088	0.950
PM10	0.955	1.000	0.259	0.675	0.577	0.266	-0.032	0.588	0.609	0.213	0.693	-0.407	-0.088	-0.277	-0.364	0.166	0.006	0.011	0.108	0.955
NO	0.289	0.259	1.000	0.737	0.871	0.270	-0.174	0.416	0.308	0.348	-0.161	-0.031	0.064	-0.497	-0.245	0.335	-0.053	0.095	-0.238	0.392
NO2	0.636	0.675	0.737	1.000	0.974	0.159	-0.261	0.842	0.434	0.282	0.236	-0.379	-0.139	-0.407	-0.181	0.437	-0.139	0.399	0.035	0.805
NOx	0.559	0.577	0.871	0.974	1.000	0.205	-0.248	0.751	0.419	0.322	0.117	-0.285	-0.079	-0.463	-0.213	0.431	-0.118	0.322	-0.055	0.716
NH3	0.147	0.266	0.270	0.159	0.205	1.000	-0.259	-0.229	-0.069	-0.250	0.478	-0.251	-0.290	-0.082	-0.353	0.078	0.025	-0.470	0.000	0.140
CO	0.122	-0.032	-0.174	-0.261	-0.248	-0.259	1.000	-0.110	0.282	0.301	-0.242	0.296	0.389	-0.054	-0.224	-0.329	0.434	-0.244	-0.079	-0.019
SO2	0.571	0.588	0.416	0.842	0.751	-0.229	-0.110	1.000	0.380	0.259	0.122	-0.332	-0.104	-0.242	-0.134	0.305	-0.139	0.572	0.150	0.732
Benzene	0.735	0.609	0.308	0.434	0.419	-0.069	0.282	0.380	1.000	0.823	0.236	0.028	0.561	-0.454	-0.328	0.104	0.324	-0.084	-0.214	0.647
Toluene	0.350	0.213	0.348	0.282	0.322	-0.250	0.301	0.259	0.823	1.000	-0.180	0.074	0.805	-0.274	-0.199	-0.052	0.307	0.026	-0.151	0.299
Ozone	0.614	0.693	-0.161	0.236	0.117	0.478	-0.242	0.122	0.236	-0.180	1.000	-0.322	-0.279	-0.174	-0.281	0.208	-0.061	-0.316	0.025	0.572
RH	-0.166	-0.407	-0.031	-0.379	-0.285	-0.251	0.296	-0.332	0.028	0.074	-0.322	1.000	0.200	-0.595	-0.027	0.267	0.207	-0.293	-0.827	-0.327
Xylene	0.055	-0.088	0.064	-0.139	-0.079	-0.290	0.389	-0.104	0.561	0.805	-0.279	0.200	1.000	-0.043	-0.201	-0.330	0.257	-0.194	-0.113	-0.054
AT	-0.469	-0.277	-0.497	-0.407	-0.463	-0.082	-0.054	-0.242	-0.454	-0.274	-0.174	-0.595	-0.043	1.000	0.172	-0.723	-0.251	0.143	0.823	-0.399
RF	-0.385	-0.364	-0.245	-0.181	-0.213	-0.353	-0.224	-0.134	-0.328	-0.199	-0.281	-0.027	-0.201	0.172	1.000	0.101	-0.139	0.582	0.078	-0.337
WS	0.205	0.166	0.335	0.437	0.431	0.078	-0.329	0.305	0.104	-0.052	0.208	0.267	-0.330	-0.723	0.101	1.000	-0.012	0.210	-0.575	0.264
WD	0.108	0.006	-0.053	-0.139	-0.118	0.025	0.434	-0.139	0.324	0.307	-0.061	0.207	0.257	-0.251	-0.139	-0.012	1.000	-0.297	-0.248	-0.010
Temp	-0.047	0.011	0.095	0.399	0.322	-0.470	-0.244	0.572	-0.084	0.026	-0.316	-0.293	-0.194	0.143	0.582	0.210	-0.297	1.000	0.276	0.132
SR	-0.088	0.108	-0.238	0.035	-0.055	0.000	-0.079	0.150	-0.214	-0.151	0.025	-0.827	-0.113	0.823	0.078	-0.575	-0.248	0.276	1.000	0.037
AQI	0.950	0.955	0.392	0.805	0.716	0.140	-0.019	0.732	0.647	0.299	0.572	-0.327	-0.054	-0.399	-0.337	0.264	-0.010	0.132	0.037	1.000

From the correlation matrix, we can infer that $PM_{2.5}$ and PM_{10} impact the AQI values significantly, BP feature is removed. When we tried removing the features with values less than the threshold 0.15, there seemed to be no improvement.

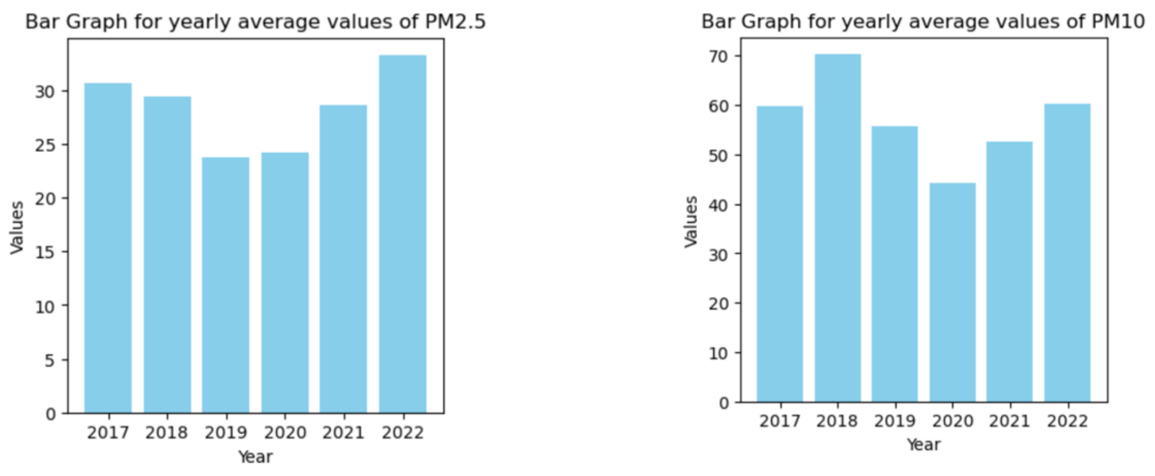
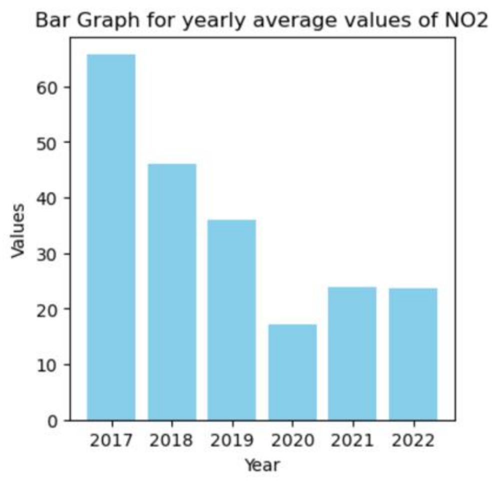


Fig 4.1.Histogram for yearly average of $PM_{2.5}$ and PM_{10}



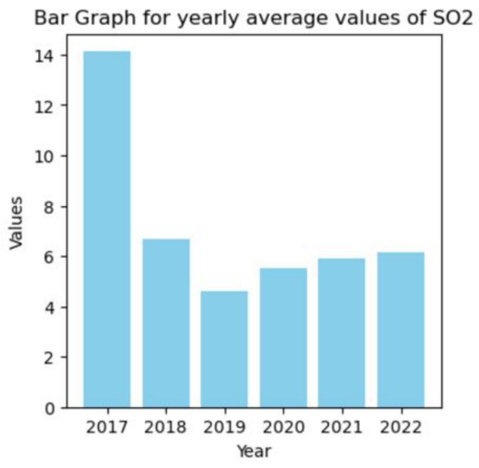
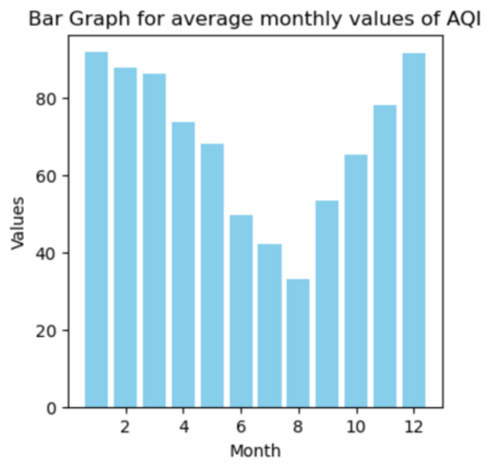


Fig 4.2.Histogram for yearly average of NO_2 and SO_2



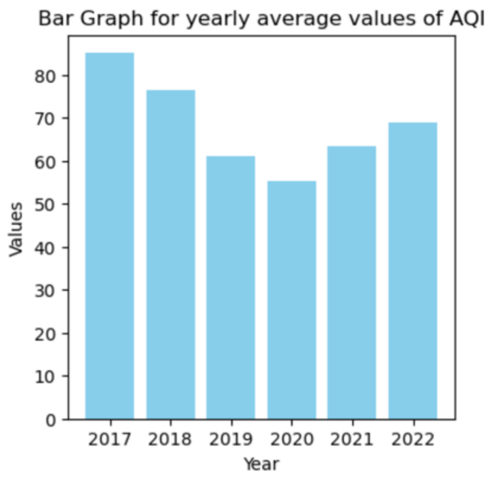


Fig 4.3.Histogram for monthly and yearly average of AQI

Air pollution levels were lower observed during the years of 2020 and 2021 due to covid lockdown. On analysing monthly average we can say that August had the lowest pollution levels whereas in December and January the AQI values were highest.

4.1.3 Data Transformation

Table 4.1 Skew and Kurtosis

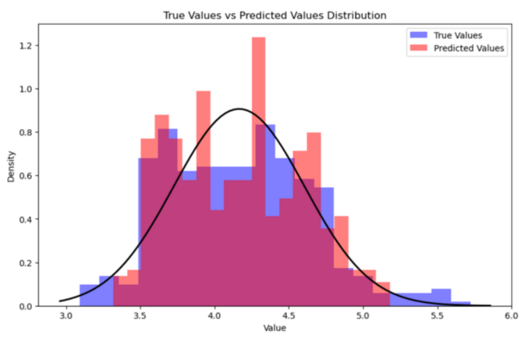
Before Log Transformation				After Log Transformation		
Features	Skew	Kurtosis		Features	Skew	Kurtosis
PM2.5	1.08	0.96		PM2.5	-0.27	0.07
PM10	0.8	1.43		PM10	-0.36	-0.38

NO	1.49	2.74		NO	-0.44	1.05
NO2	2.06	9.4		NO2	-0.21	-0.13
NOx	1.59	5.04		NOx	-5.35	104.62
NH3	1.11	0.62		NH3	-0.99	6.17
CO	5.88	58.01		CO	-4.66	26.7
SO2	1.81	2.9		SO2	0.26	0.35
Benzene	2.87	19.29		Benzene	-4.01	22.12
Toluene	3.36	18.35		Toluene	-3.78	26.74
Ozone	1.12	1.17		Ozone	-0.32	0.75
RH	-0.66	-0.01		RH	-1.08	1.08
Xylene	6.09	61.63		Xylene	-2.24	6.13
AT	-0.01	-0.74		AT	-0.21	-0.75
RF	10.13	103.69		RF	1.13	-0.48
WS	1.03	1.34		WS	-0.31	-0.29
WD	-0.06	-0.54		WD	-0.98	1.68
Temp	4.31	27.93		Temp	3.21	16.23
SR	1.04	2.66		SR	-1.4	2
AQI	1.53	3.96		AQI	0.29	-0.59

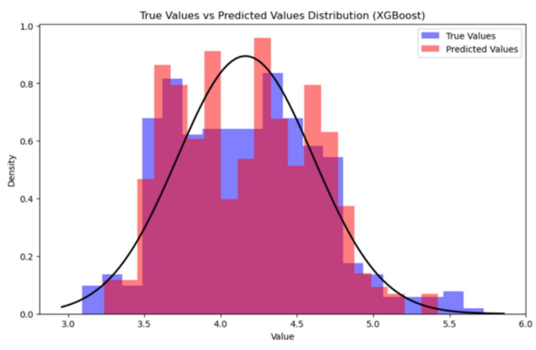
Log transformation is applied to normalize the current dataset, and the skewness, and kurtosis of the same before and after transformation.

4.2 MODEL PERFORMANCES AND COMPARATIVE ANALYSIS

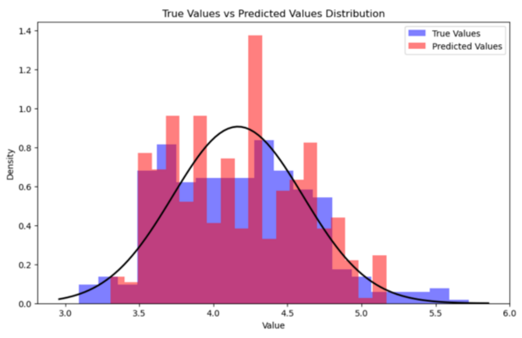
4.2.1 Comparative analysis



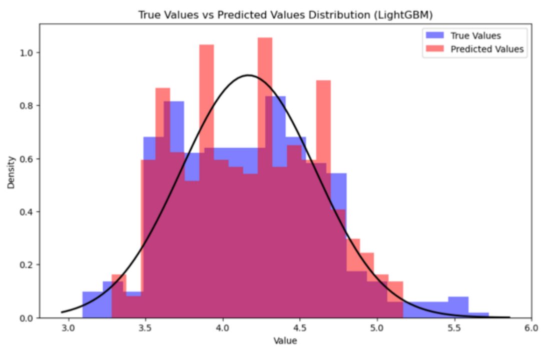
Random Forest



XGBoost

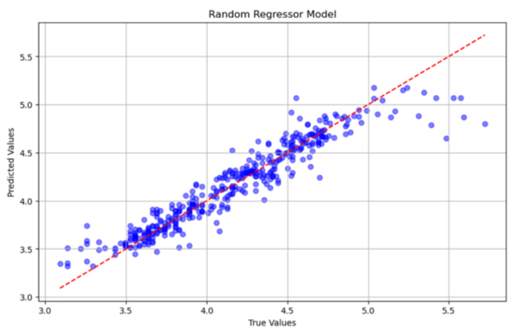


Bagging Regressor

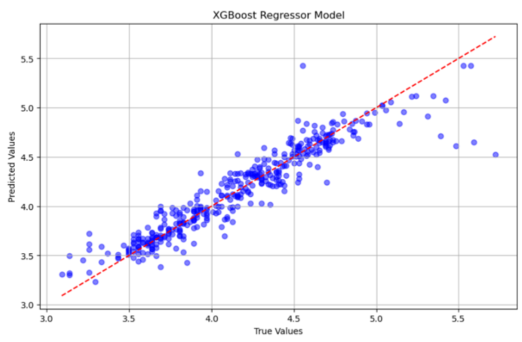


LGBM Regressor

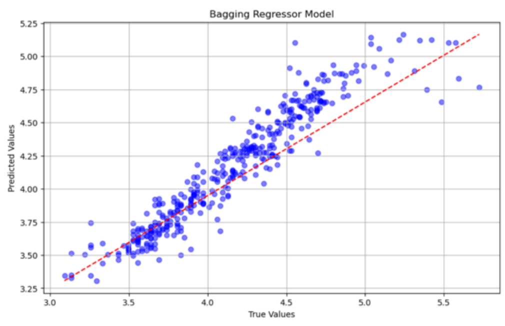
Fig 4.4.Density vs Residuals



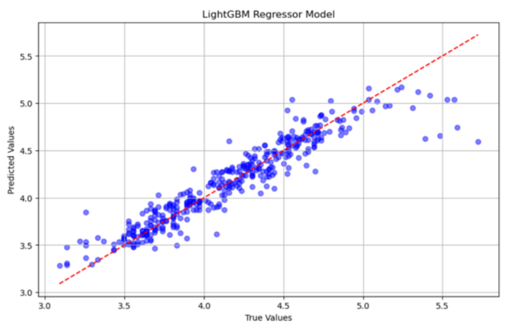
Random Forest



XGBoost

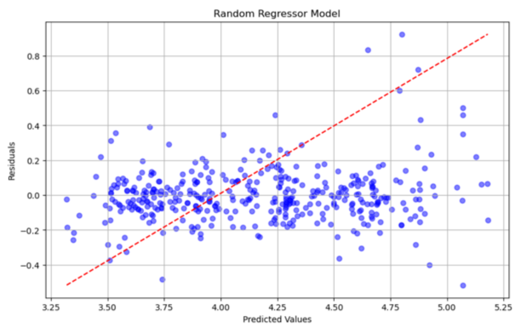


Bagging Regressor

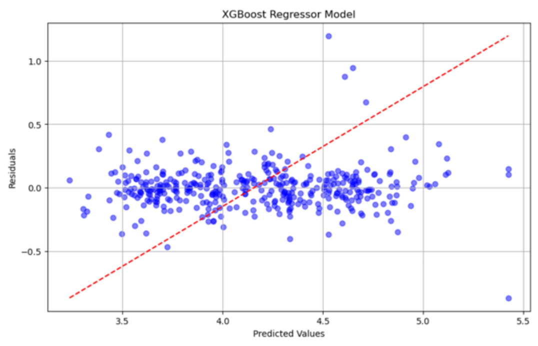


LGBM Regressor

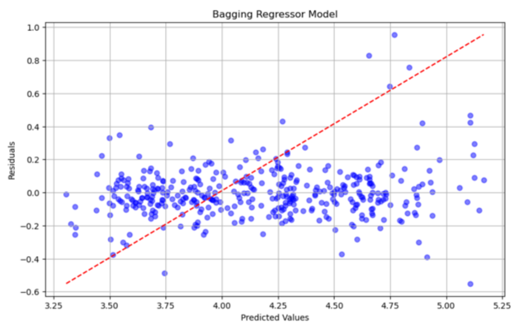
Fig 4.5.True vs Predicted values



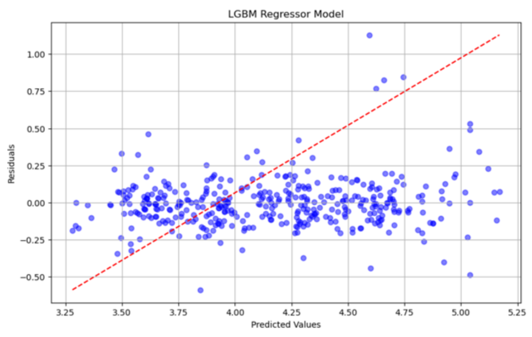
Random Forest



XGBoost

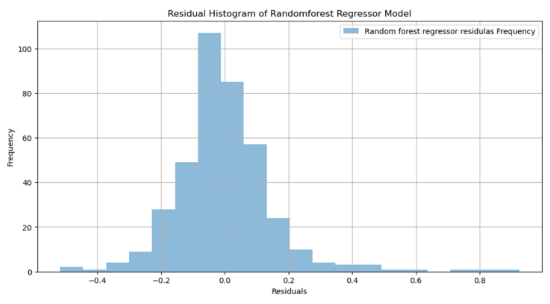


Bagging Regressor

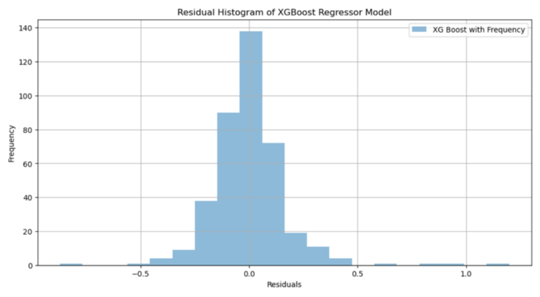


LGBM Regressor

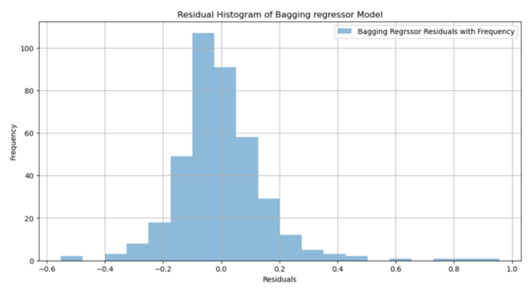
Fig 4.6.Predicted vs Residuals



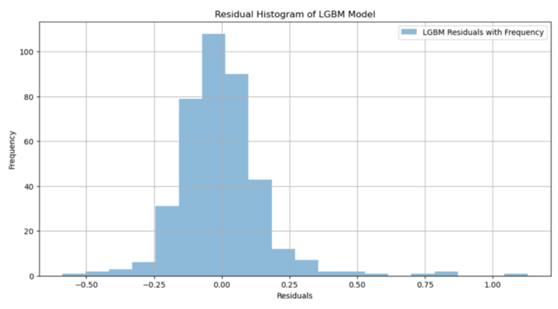
Random Forest



XGBoost



Bagging Regressor



LGBM Regressor

Fig 4.7.Residual Histograms

Histograms with residuals, true and predicted values , and probability distribution function graphs were plotted for the trained model to assess their performance visually. From a visual inspection we can observe that Random Forest and XGBoost give more accurate predictions.

4.2.2 Performance metrics

Table 4.2 Performance metrics on monthly average filling

Monthly Average Data Preprocessing								
Models	Training Data				Testing Data			
	MSE	RMSE	R2	MAE	MSE	RMSE	R2	MAE
Random Forest	0.0046	0.0678	0.9768	0.0444	0.0233	0.1526	0.8822	0.1083
XGBoost	0.0064	0.08	0.9672	0.0567	0.0257	0.1603	0.87	0.1118
Bagging Regressor	0.0048	0.0693	0.9755	0.0448	0.023	0.1517	0.8835	0.1083
LGBM Regressor	0.0093	0.0964	0.9525	0.0669	0.0253	0.1591	0.8719	0.1127

Table 4.3 Performance metrics on closest row filling

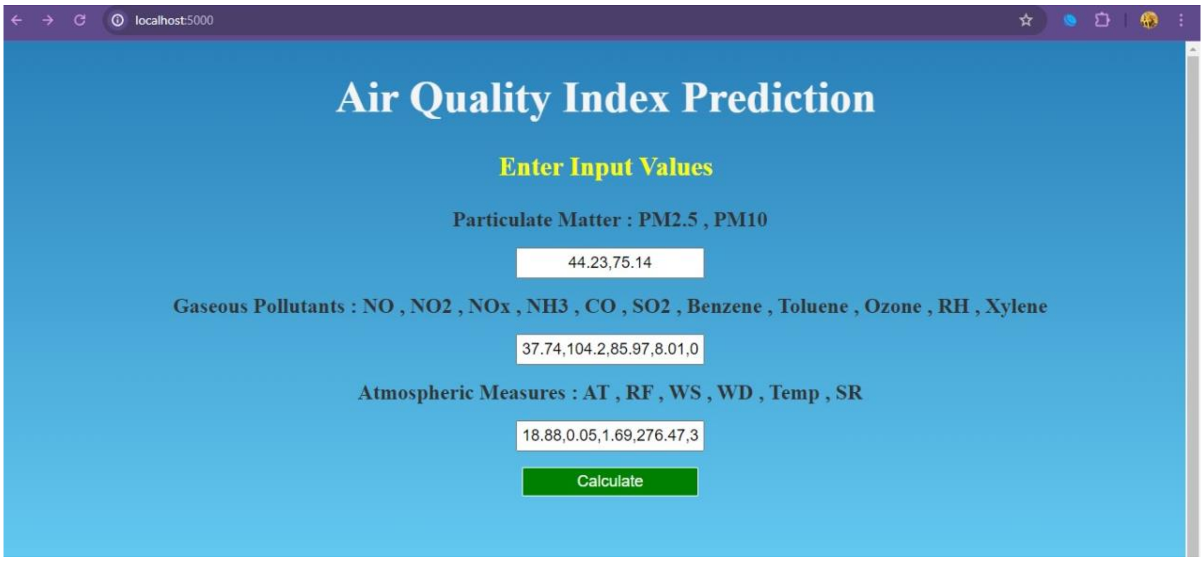
Closest Row Value Data Preprocessing								
Models	Training Data				Testing Data			
	MSE	RMSE	R2	MAE	MSE	RMSE	R2	MAE
Random Forest	0.0042	0.0648	0.9789	0.0412	0.023	0.1517	0.886	0.1022
XGBoost	0.0059	0.0768	0.9704	0.0542	0.0253	0.1591	0.8748	0.106
Bagging Regressor	0.0045	0.0671	0.9774	0.042	0.0229	0.1513	0.8864	0.1029
LGBM Regressor	0.0094	0.097	0.9534	0.0662	0.0248	0.1575	0.8772	0.1083

Table 4.4 Performance metrics on removal of null rows

Removal of Rows With No Target Values								
Models	Training Data				Testing Data			
	MSE	RMSE	R2	MAE	MSE	RMSE	R2	MAE
Random Forest	0.0031	0.0557	0.9844	0.0388	0.0244	0.1562	0.8965	0.1069
XGBoost	0.0047	0.0686	0.9765	0.051	0.0256	0.16	0.8912	0.1068
Bagging Regressor	0.0034	0.0583	0.9831	0.0397	0.0242	0.1556	0.8973	0.1057
LGBM Regressor	0.0076	0.0872	0.9619	0.0608	0.0272	0.1649	0.8846	0.1113

On comparing the above three sets of values, we conclude that the ML models scored high when the rows without target variable values are removed. The highest among them being for Random Forest (0.9844) , for training data. After fitting the ML models (Random Forest, XGBoost, Bagging Regressor, LightGBM), the approach of removing the missing rows turned out to produce high scores of accuracy. The ML models Random Forest and Bagging Regressor showed maximum correlation of 0.9844 and 0.9831 respectively for the training data. The Bagging Regressor score is 0.8973 for testing/validation.

4.3 GRAPHICAL USER INTERFACE USING FLASK


localhost:5000

Air Quality Index Prediction

Enter Input Values

Particulate Matter : PM2.5 , PM10

44.23,75.14

Gaseous Pollutants : NO , NO2 , NOx , NH3 , CO , SO2 , Benzene , Toluene , Ozone , RH , Xylene

37.74,104.2,85.97,8.01,0

Atmospheric Measures : AT , RF , WS , WD , Temp , SR

18.88,0.05,1.69,276.47,3

Fig 4.8.Input Webpage

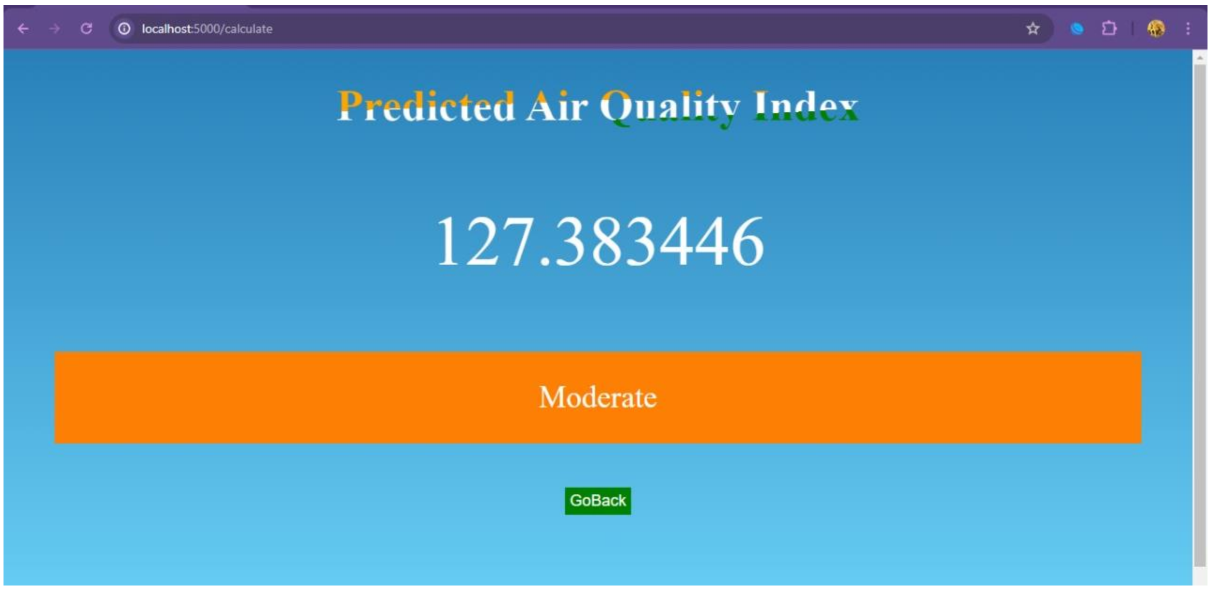


Fig 4.9.Predicted AQI webpage

CHAPTER 5

5.1 CONCLUSION

Nowadays air pollution has turned into a major concern and its high time organisations and government entities around the world focus on improving strategies to tackle the problem. Predicting and forecasting pollution metrics like AQI values help us to arrive at better strategies to combat the Air pollution in the modern world. Many researchers have already investigated various methods to predict AQI using Artificial Intelligence. In this project, the authors implement the ML models like Random Forest Regressor, XGBoost, Bagging Regressor and LightGBM to predict the AQI values accurately.

The current project explored the historical AQI dataset of Tirupati City from 2017 to 2022. The parameters like $PM_{2.5}$ and PM_{10} played significant role in AQI prediction, which huge correlation. Three different approaches were possible when rows with missing target variable values were either removed or replaced with imputation/closest row value. After fitting the ML models (Random Forest, XGBoost, Bagging Regressor, LightGBM), the approach of removing the missing rows turned out to produce high scores of accuracy

The ML models Random Forest and Bagging Regressor showed maximum correlation of 0.9844 and 0.9831 respectively for the training data. The Bagging Regressor score is 0.8973 for testing/validation. Utilizing ML algorithms facilitates the precise prediction of AQI, thereby enabling its utilization as virtual air monitoring stations. This endeavor demonstrated the efficacy of ML models in forecasting AQI for Tirupati City.

5.2 FUTURE PLANS

This project has significant real-world applications like forecasting AQI values and other pollutant concentrations for better strategies. The optimum models can also be implemented with spatial interpolation to predict the AQI values in regions with no manual air monitoring stations. Using this technique, accurate Air quality data can be known by setting up virtual air quality monitoring stations. It sets the foundation for applying any other cutting-edge ML algorithm in the future. Building robust frontend with efficient framework would help users to view the predicted AQI values accurately from anywhere around the globe. Through AQI prediction targeted policy making and improved public awareness is possible about the real time air quality in their city.

CHAPTER 6

REFERENCES

[1] **Gokulan Ravindiran, Sivarethinamohan Rajamanickam, Karthick Kanagarathinam, Gasim Hayder, Gorti Janardhan, Priya Arunkumar, Sivakumar Arunachalam, Abeer A. AlObaid, Ismail Warad, Senthil Kumar Muniasamy,** " Impact of air pollutants on climate change and prediction of air quality index using machine learning models" , *Environmental Research, Volume 239, Part 1, 2023, 117354*

[2] **Gokulan Ravindiran, Gasim Hayder, Karthick Kanagarathinam, Avinash Alagumalai, Christian Sonne,** "Air quality prediction by machine learning models: A predictive study on the indian coastal city of Visakhapatnam", *Chemosphere, Volume 338, 2023, 139518.*

[3] **C R, Aditya & Deshmukh, Chandana & K, Nayana & Gandhi, Praveen & astu, Vidyav.** "Detection and Prediction of Air Pollution using Machine Learning Models". *International Journal of Engineering Trends and Technology. 59. 204-207, (2018)*

[4] **Samad, S. Garuda, U. Vogt, B. Yang,** "Air pollution prediction using machine learning techniques – An approach to replace existing monitoring stations with virtual monitoring stations", *Atmospheric Environment, Volume 310, 2023, 119987*

[5] **Nilesh N. Maltare, Safvan Vahora,** "Air Quality Index prediction using machine learning for Ahmedabad city", *Digital Chemical Engineering, Volume 7, 2023, 100093*

CHAPTER 7

APPENDIX – BASE PAPER

Title	:	Impact of air pollutants on climate change and prediction of air quality index using machine learning models
Author	:	Gokulan Ravindiran, Sivarethinamohan Rajamanickam, Karthick Kanagarathinam, Gasim Hayder, Gorti Janardhan, Priya Arunkumar, Sivakumar Arunachalam, Abeer A. AlObaid, Ismail Warad, Senthil Kumar Muniasamy
Publisher	:	Elsevier
Year	:	2023
Journal	:	Environmental Research Volume 239, Part 1, Article 117354
Indexing	:	SCI / Scopus
Base paper URL	:	https://www.sciencedirect.com/science/article/pii/S0013935123021588