# AIR QUALITY INDEX PREDICTION FOR NATIONAL CAPITAL REGION USING LINEAR REGRESSION

**A PROJECT REPORT**

***Submitted by***

**ARYAN VIKAS JAIN [RA1811003030388]**

*Under the guidance of*

**Mr. Naresh Sharma**

( Assistant Professor, Department of Computer Sciene & Engineering)

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE & ENGINEERING**

of

**FACULTY OF ENGINEERING AND TECHNOLOGY**


Delhi NCR Campus, Modinagar, Ghaziabad (U.P.)

**MAY 2022**

# SRM INSTITUTE OF SCIENCE & TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

## BONAFIDE CERTIFICATE

Certified that this project report titled "**AIR QUALITY INDEX PREDICTION FOR NATIONAL CAPITAL REGION USING LINEAR REGRESSION**" is the bonafide work of " **ARYAN VIKAS JAIN [RA1811003030388]** ",who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**SIGNATURE**

Mr. Naresh Sharma
Assistant Professor
Dept. of Computer Sciene & Engineering

Signature of the Internal Examiner

**SIGNATURE**

Prof. (Dr.) R.P. Mahapatra
**HEAD OF THE DEPARTMENT**
Dept. of Computer Science & Engineering

Signature of the External Examiner

# ABSTRACT

India has the some of the world's most polluted cities among top ten [1] . Over the past decades we have seen a deterioration in the quality of the air we breathe in National Capital Region cities like Delhi, Ghaziabad, Noida, Meerut and Bulandshahr. Death rate in India due to pollution is increasing day by day.1,68,000 people were reported dead due to Air pollution in India in the year of 2019 [2], whereas the death rate in 1990 was 43,200. This disastrous increase in death rate due to air pollution in India motivated us to predict the future air-quality-index for years 2022 to 2026 for society welfare.We have been reviewing the trends for calculating air-quality-index for the years 2017 to 2021 considering six air pollutants like Sulphur Dioxide ($SO_2$), Nitrogen Dioxide ($NO_2$), Particulate Matter 2.5 ($PM_{2.5}$) also called as fine particulate matter have diameter of 2.5 micrometers like Dust, spores and pollen, Particulate Matter 10 ($PM_{10}$) also called as coarse particulate matter have diameter of 10 micrometers or less, including smoke, dust, soot, salts, acids, and metals, Carbon Monoxide (CO) and Ozone ($O_3$) at ground level using Smart Internet Computed Devices. We can see pollution is caused around us due to many factors like pollution from cars, air conditioners releasing Chlorofluorocarbon(CFCs), chimneys' emissions, wildfires, burning cultivation techniques and many more. Our main focus is to predict the air-quality-index for years 2022 to 2026 to control and try to normalize the pollution levels in India to lessen the death rate. Our research concluded in a way where it tells us that for our environment we are successful to control the pollutants namely Ozone ($O_3$) , Sulphur Dioxide ($SO_2$) and Nitrogen Dioxide ($NO_2$). We still need to take measures to control the pollutants Particulate Matter 2.5 ($PM_{2.5}$) , Particulate Matter 10 ($PM_{10}$) and Carbon Monoxide (CO).

# ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my guide, Mr. Naresh Sharma his valuable guidance, consistent encouragement, personal caring, timely help and providing me with an excellent atmosphere for doing research. All through the work, in spite of his busy schedule, he has extended cheerful and cordial support to me for completing this research work.

**Aryan Vikas Jain RA1811003030388**

# Contents

# List of Tables

# List of Figures

# ABBREVIATIONS

| | |
|---|---|
| $PM_{2.5}$ | Particulate Matter 2.5 |
| $PM_{10}$ | Particulate Matter 10 |
| CO | Carbon Monoxide |
| $NO_2$ | Nitrogen Dioxide |
| $SO_2$ | Sulphur Dioxide |
| $O_3$ | Ozone |
| AQI | air-quality-index |
| AQHI | Air Quality Health Index |
| NCR | National Capital Region |
| AI | Artificial Intelligence |
| ML | Machine Learning |
| SVR | Support Vector Regression |
| RFR | Random Forest Regression |
| ANN | Artificial Neural Network |
| LR | Linear Regression |
| LSTM | Long Short-Term Memory |
| CPCB | Central Pollution Control Board |
| ISOA | Improved Seagull Optimization Algorithm |
| GRA | Gray Relational Analysis |
| UAV | Unmanned Aerial Vehicle |
| FL | Federated Learning |
| MODWT | Maximal Overlap Discrete Wavelet Package Transfer |
| TCDA | Temporal Convolutional Denoising Autoencoder |
| TCN | Temporal Convolutional Network |
| DAE | Denoising Autoencoder |
| CNN | Convolutional Neural Network |
| RNN | Recurrent Neural Network |

| | |
|---|---|
| ORELM | Outlier Robust Extreme Learning Machine |
| GSA | Grey System Approximation |
| TVFEMD | Time-varying Filtering Empirical Mode Decomposition |
| MFO | Moth Flame Optimization |

# LIST OF SYMBOLS

| | |
|---|---|
| $Y$ | Dependent variable/response variable |
| $m$ | slope |
| $X$ | Independent variable/predictor variable |
| $b$ | Intercept |
| $e$ | Error |
| $\bar{X}$ | Mean value of X |
| $\bar{Y}$ | Mean value of Y |
| $\Sigma$ | Sum |
| $R^2$ | Variation in Best fit |
| $Y_{\mathbf{p}}$ | Predicted value of pollutants |
| $AQI$ | air-quality-index of pollutant p |
| $Conc_{\mathbf{p}}$ | Pollutants's concentration |
| $MaxConc_{\mathbf{p}}$ | Maximum concentration pf pollutant as per reference range |
| $MinConc_{\mathbf{p}}$ | Minimum concentration pf pollutant as per reference range |
| $MaxAccAQI$ | Maximum accepted AQI value corresponding to MaxConc of pollutant |
| $MinAccAQI$ | Maximum accepted AQI value corresponding to MinConc of pollutant |

# Chapter 1

# INTRODUCTION

## 1.1 Introduction

Air contamination is a significant ecological danger factor in proliferation infections for example, cellular breakdown in the lungs, mental imbalance, asthma, low birth weight, and so on . Guideline of air quality is a significant undertaking of the public authority in emerging nations to guarantee individuals' well being and government assistance. Air contamination contrasts from one put to another and relies upon different poison sources, for example, modern emanations, weighty gridlocks, temperature, pressure, wind, dampness, and consumption of petroleum products and so on .Analysing air quality is now one of the most required exercises for the public authority in practically every one of the modern and metropolitan regions today. We have seen air-quality-index took a slight decrease in the duration of 2019 to 2021 as it was the pandemic duration. Due to the fast widespread of virus of COVID-19 during these years lockdown has been imposed all over India to prevent this widespread. In this paper, we have utilized the algorithm of Linear Regression to predict the amount of air pollutants like Nitrogen Dioxide ($NO_2$), Particulate Matter 2.5 ($PM_{2.5}$), Sulphur Dioxide ($SO_2$), which is also called as fine particulate matter that have diameter of 2.5 micrometers like dust, spores and pollen, Particulate Matter 10 ($PM_{10}$) which is also called as Coarse particulate matter that have diameter of less than or equal to ten micrometers, including soot, smoke, salts, dust, acids, and metals, Carbon Monoxide (CO) and Ozone ($O_3$) at ground level. In this paper we investigated the air quality with different toxin fixations through representations during 2018 to 2021 for successful component extraction and direction. An AI model is invented utilizing Linear Regression algorithm to foresee the air quality list as a result of past air quality information. Our model training results show that the model we proposed can be effectively used to identify the nature of air. We have taken the data from the website of **Central Pollution Control Board** [2]. Then we started structuring our data as we had adjusted all null values by the average of remaining

values for each pollutant and have managed to complete the database. Also removed any of the extra pollutants outside from the scope of our study as using more than 8 pollutants was giving a negative impact on calculating air-quality-index and out of them we chose six major pollutants described above.

For these pollutants, the time frame that is needed to calculate vary between 1 hour to 24 hours as explained ahead:-

- For Sulphur Dioxide ($SO_2$), Ozone ($O_3$) and Nitrogen Dioxide ($NO_2$) we calculate the maximum hourly value for 24hours. 24 per hour values are obtained and then highest one chosen as our daily average.
- For both Particulate Matter be it both $PM_{10}$ and $PM_{2.5}$ and we calculate both the average of an hour and average of 24 hours. 24 values at each hour of the day are obtained and then we calculate the average of all 24 hours and further consider those particular values of $PM_{2.5}$ and $PM_{10}$.
- For Carbon Monoxide (CO) we calculate both the average of an hour and average of 8 hours. Over the course of a 24-hour day, 17 8-hour Carbon Monoxide values are obtained, with the highest one chosen as our daily Carbon Monoxide average.

The particular bounded area of each city that we considered to measure the pollutants concentration to calculate air-quality-index are shown below with their exact coordinates :

1. In Delhi we considered **Delhi Institute of Tool Engineering, Wazirpur, Delhi** with coordinates 28.700692917883536, 77.16545969688022 [3]

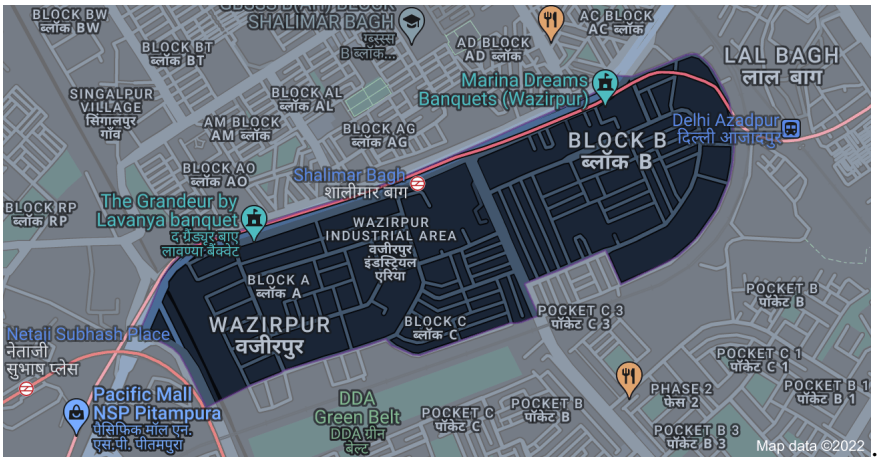


**Figure 1.1:** Wazirpur, Delhi

2. In Ghaziabad we considered **Vasundhara , Ghaziabad** with coordinates 28.661984331671505, 77.36921875054489 [4]

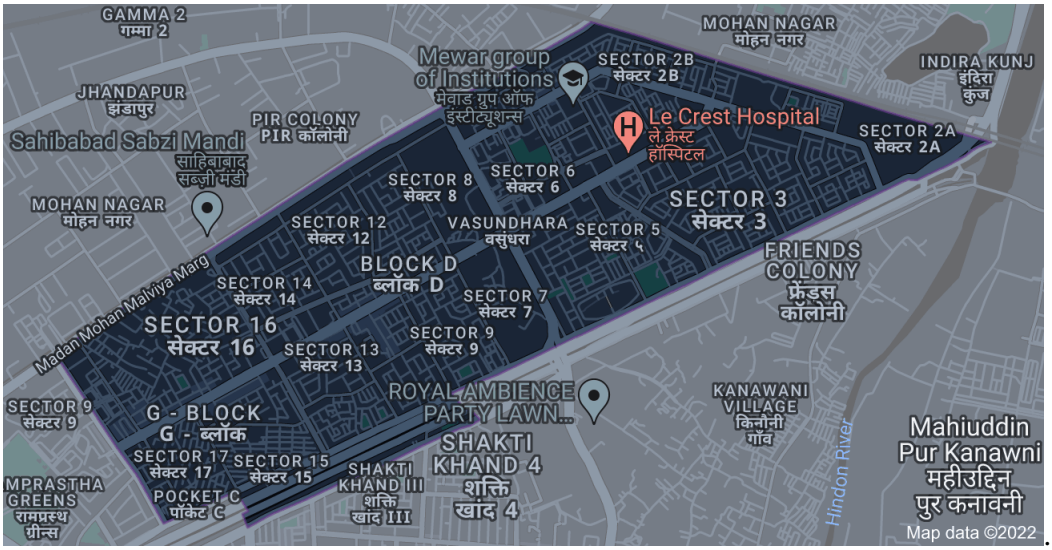


**Figure 1.2:** Vasundhara, Ghaziabad

3. In Meerut we considered **Jai Bhim Nagar , Meerut** with coordinates 28.95781576763979, 77.75672065359652 [5]

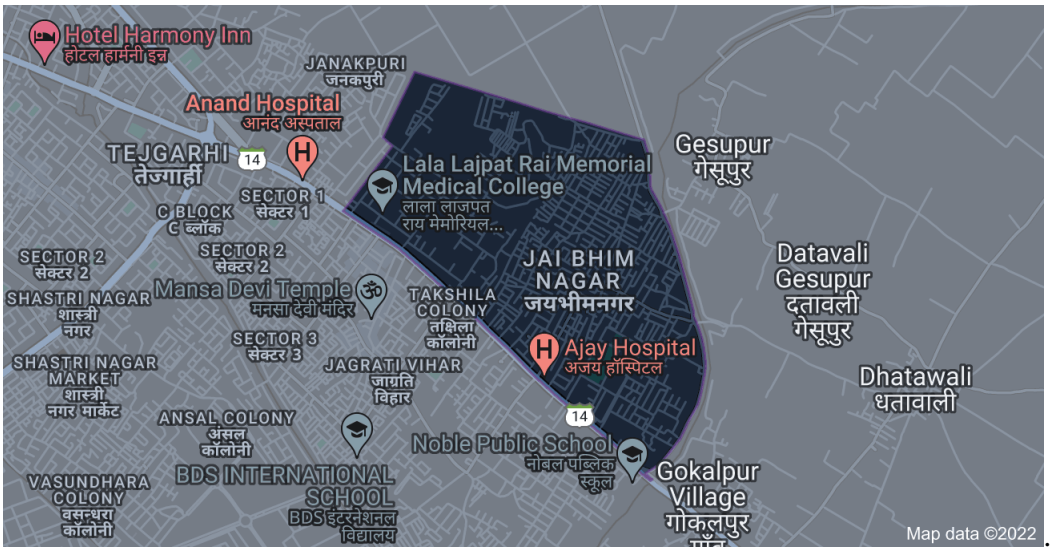


**Figure 1.3:** Jai Bhim Nagar, Meerut

4. In Bulandshahr we considered **Yamunapuram, Bulandshahr** with coordinates 28.41055482650603, 77.83006753517664 [6]

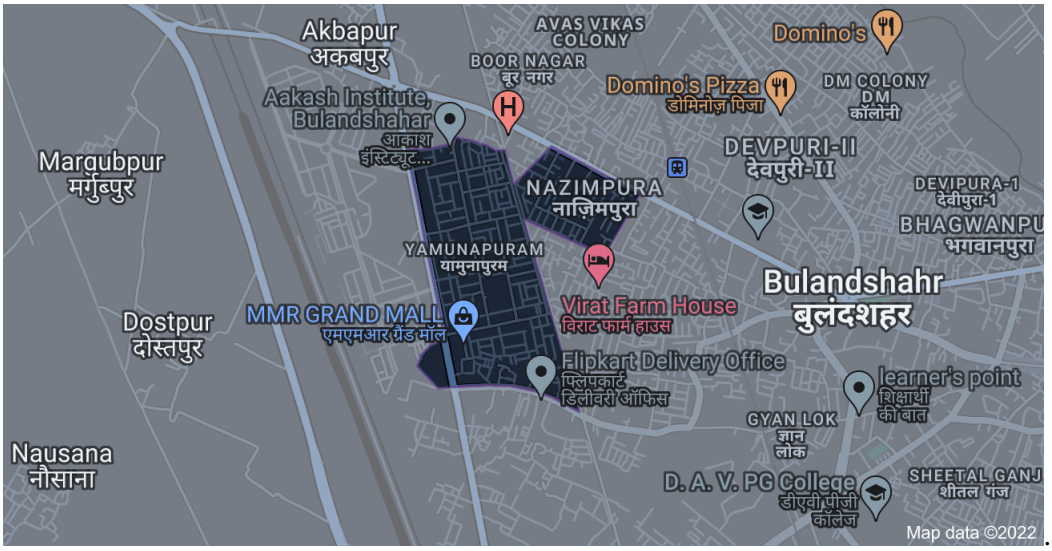


**Figure 1.4:** Yamunapuram, Bulandshahr

5. In Noida we considered **Sector - 116, Noida** with coordinates 28.57004435212374, 77.39686950408193 [7]


**Figure 1.5:** Sector 116, Noida

# Chapter 2

# LITERATURE SURVEY

**Athanasiadis Ioannis N. , Vassilis G. Kaburlasos et al.(2003)** [8]

They used the $\sigma$–FLNMAP classifier and Neural Network backpropagation for Ozone concentration of Gdansk, Poland.

**Honoré Cécile, Laurence Rouil et al.(2008)** [9]

They used the long_term evaluation of working real_time air_quality forecasting and analyzing system using error statistics for over three years consecutively.

**Dragomir Elia Georgiana(2010)** [10]

They predicted the value of the air-quality-index to categorize the level of pollution . They used K-nearest neighbor for air-quality-index of Ploiesti,Romania. The results were relatively good, if we consider that for 19 of the 29 instances the prediction error was zero. The accuracy of the model can be improved by taking into consideration a longer period of time for the model's training set.

**Akhtar Aly, Sarfaraz Masood(2018)** [11]

The value of $PM_{10}$ was predicted using the multilayer_perceptron algorithm, which is an ANN, Naive-Bayes algo. and Support-Vector-Machine algo. Temperature,humidity, wind direction, wind speed and other meteorological conditions were all taken into account when creating the prediction model.

**Nidhi Sharma (et al.2018)** [12]

They diffusely analyzed air pollutants from 2009 to 2017 and also proposed their critical_observation of the 2016 to 2017 air pollutants trends in city Delhi, India. They predicted the future possibility of several pollutants by using data_analytics Time_series_Regression forecasting they predicted the pollutants's future values.

**Ziyue Guan , Richard O.Sinnott (2018)** [13]

They used numerous ML algorithms to predict the concentration of $PM_{2.5}$ . They gathered data from Environment Protection Agency's (EPA) official website for city of Melbourne, which covers $PM_{2.5}$ as parameters, as well as unauthorised data from source of Airbeam, a device with mobility is designed to measure levels of $PM_{2.5}$ . They used the ML Algorithms.

**Mohamed Shakir , N.Rakesh (2018)** [14]

They analyzed the proportion of varied air_pollutants (NO2, NO, PM10, CO and SO2) with regard to the day time and also the weekday and calculated the result of environmental-parameters as temperature, speed of wind and wetness on the air pollutants mentioned on top of with the assistance of rail tool . Used K-means algorithm to show links among various air pollutants.

**Huixiang Liu (et al.2019)** [15]

Used Support Vector Regression(SVR) for air-quality-index of Beijing and Random Forest Regression (RFR) for Nitrogen Oxides (NOx) of Italian city.

**Heidar Maleki (2019)** [16]

They predicted the concentration value per hour for air_pollutants SO2, NO2, PM10, PM2.5, CO and O3 for the stations of Havashenasi, Naderi, Behdasht and MohiteZist in Ahvaz, Islamic Republic of Iran. They calculated both air-quality-index and Air_Quality_Health_Index for stations in Islamic Republic of Iran using ANN.

**Chowdhury Sourangsu, Sagnik Dey et al.(2019)** [17]

The research demonstrated the benefit of pollution analysing using high-resolution data from satellite to development air quality management plan at a large scale. Used high resolution satellite aerosol dataset for $PM_{2.5}$ in Delhi.

**Hui Liu and Chao Chen (2020)** [18]

Spatial_correlation_analysis, mutuality, and binary-grey-wolf-optimization were used to extract spatio-temporal information in three steps (BGWO).To ensure outstanding performance, all of the components of hybrid model proposed have been proven to be essential. The spatio-

temporal data are taken into account in a reasonable manner, which improves air pollution prediction models.

**Xu, Ting and Yan, Huichao and Bai, Yanping (2021)** [19]

During the pandemic, the Chinese government took a number of efforts to restrict the spread of the disease, which resulted in Taiyuan's air quality being substantially better in February 2020 than in prior years. The Gray Relational Analysis (GRA) method was used for the first time in this research to evaluate and analyse the impact of six major contaminants on air quality. Then, to create a hybrid predicted model ISOA-SVR, the improved seagull optimization algorithm (ISOA) was presented and integrated with Support Vector Regression (SVR). Finally, the ISOA-SVR was used to forecast the air-quality-index (AQI). The proposed ISOA-SVR had better generalization ability and robustness than other predicted models, according to the experimental results on two types of data. Furthermore, the proposed ISOA-SVR is appropriate for air-quality-index prediction.

**Chhikara, Prateek and Tekchandani et al.(2021)** [19]

Air-pollution analysing, determining the hazardous-zone, and forecasting future air-quality recently became a major focus for many academics. Because of the negative effects of poor air-quality on an individual's health, it is now required to properly and timely anticipate the air-quality-index . Air quality data can be collected with high geographical and temporal resolutions using an unmanned aerial vehicle (UAV). Using a fleet of unmanned aerial vehicles (UAVs) could be a viable alternative. Using swarm intelligence, a method for locating the area with the greatest air-quality-index value is proposed.

**Erdinc Aladag (2021)** [20]

Particulate matter is a major air contaminant that has a substantial impact on human health. It is feasible to take the required safeguards to avoid potential dangers projecting air-quality with accuracy for future horizons. Monthly PM10 concentration projections for Erzurum, Turkey, were made in this study. Between 2006 and 2018, the first 10 years of monthly data were used to train model, while the latter 2 years were used to test the predictions made by the model.

**Krishna Rani Samal , Korra Sathya Babu , Santos Kumar Das (2021)** [21]

They created a hybrid PM2.5 prediction framework called the Temporal Convolutional Denoising Autoencoder (TCDA) network, which can extract features from complicated datasets quickly, manage missing values, and improve PM2.5 prediction outcomes. To handle nonlinear multivariate large datasets, this research article uses a Temporal Convolutional Network (TCN) and a Denoising Autoencoder (DAE) network. The former employs Convolutional-Neural-Network (CNN) parallel feature processing and Recurrent-Neural-Network (RNN) temporal component modelling capabilities to aid in feature extraction from difficult datasets. The latter is used to recreate the error so that the prediction results may be fine-tuned and missing values can be handled. The suggested model's prediction capacity is assessed by comparing its performance to that of existing baseline models, demonstrating that it outperforms other models.

**Wei Sun and Zhiwei Xu(2021)** [22]

First, Random-Forest (RF) and gray-system-approximation models were used to screen historical data (GSA). Second, time varying filtering based decomposition of empirical mode is used to deconstruct the processed data (TVFEMD). Then, for prediction, the extreme learning machine (ELM) is utilised, which is optimised by the moth flame optimization algorithm (MFO). The following conclusions can be formed based on data from four cities in the Beijing-Tianjin,Hebei region: The proposed model's efficacy and robustness have been validated, and the evaluation indices are the best.

# Chapter 3

# PROBLEM IDENTIFICATION

## 3.1 Predict air-quality-index for National Capital Region cities :

Air contamination is a significant ecological danger factor in proliferation infections for example, cellular breakdown in the lungs, mental imbalance, asthma, low birth weight, and so . Guideline of air quality is a significant undertaking of the public authority in emerging nations to guarantee individuals' wellbeing and government assistance. Air contamination contrasts from one put to another and relies upon different poison sources, for example, modern emanations, weighty gridlocks, temperature, pressure, wind, dampness, and consumption of petroleum products and so on .Analyzing air-quality is one of the most required exercises for the public authority in practically every one of the modern and metropolitan regions today. In this paper, AI calculations are utilized to examine the centralizations of air toxins like Sulphur Dioxide ($SO_2$), Nitrogen Dioxide ($NO_2$), Particulate Matter 2.5 ($PM_{2.5}$) which is also called as fine particulate matter that have diameter of 2.5 micrometers like dust, spores and pollen, Particulate Matter 10 ($PM_{10}$) which is also called as Coarse particulate matter that have diameter of less than or equal to ten micrometers, Carbon Monoxide (CO) and Ozone ($O_3$) at ground level. This paper investigates the air quality given different toxin fixations through representations during pre-Coronavirus and post-Coronavirus days for successful component extraction and direction for future years that are 2022 to 2026. An AI model is invented utilizing Linear Regression algorithm to foresee the air quality list as a result of past air quality information.

## 3.2 Obtain results with 95% or more accuracy for air-quality-index prediction.

Today, the application of low-priced systems that use sensor-based technology for sample assortment and waste material concentration measurement is on the rise. It's due to their easy operating and low maintenance. But also with the increased use of low-cost systems using sensor-based technology to collect samples and measure pollutant concentration, concerns are raised for its accuracy. The most important is the quality and accuracy of these devices. Several calibration methods can be employed to dissolve human errors and clean the data collected.

## 3.3 Choose an appropriate ML model for the model training

Selecting a model is choosing one among many candidate models for predictive model training. There may be several competitive issues once performing model choice on the far side model performance, like complexity, maintainability, and accessible resources.

## 3.4 Use statistical and data preprocessing techniques on our dataset

Quality of the data and its representation are the primary and foremost points to ensure the successful compiling of prediction models. The data pre-processing step often impacts the machine learning algorithm's generalization ability. Data pre-processing usually includes imputation of missing data, the removal or modification of outliers, and data transformation (usually normalization and standardization), and feature engineering. We have taken the data from the website of **Central Pollution Control Board** [2]. Then we started structuring our data as we had adjusted all null values by the average of remaining values for each pollutant and have managed to complete the database. Also removed any of the extra pollutants outside from the scope of our study as using more than 8 pollutants was giving a negative impact on calculating air-quality-index and out of them we chose six major pollutants described above.

## 3.5 Creating insightful graphs for data visualization

We'll be creating line plots and scatter plots for analyzing features like Sulphur Dioxide ($SO_2$), Nitrogen Dioxide ($NO_2$), Particulate Matter 2.5 ($PM_{2.5}$), Particulate Matter 10 ($PM_{10}$), Carbon Monoxide (CO) and Ozone ($O_3$) as these have the major impact upon air-quality-index. Visualization while training our model will help us to create various insights and help us during the model testing phase and hence predicting the air-quality-index for 2022 to 2026 with utmost accuracy. The techniques for estimation recommended by Central Pollution Control Board for separate boundaries are the blend of actual strategy, wet-substance technique, and consistent web-based technique. The ceaseless web-based surrounding air quality checking frameworks are furnished with analyzers for estimation of Sulphur Dioxide ($SO_2$), Nitrogen Dioxide ($NO_2$), Particulate Matter 2.5 ($PM_{2.5}$), Particulate Matter 10 ($PM_{10}$), Carbon Monoxide (CO) and Ozone ($O_3$).

# Chapter 4

# ARCHITECTURAL DESIGN

## 4.1 Architectural Model

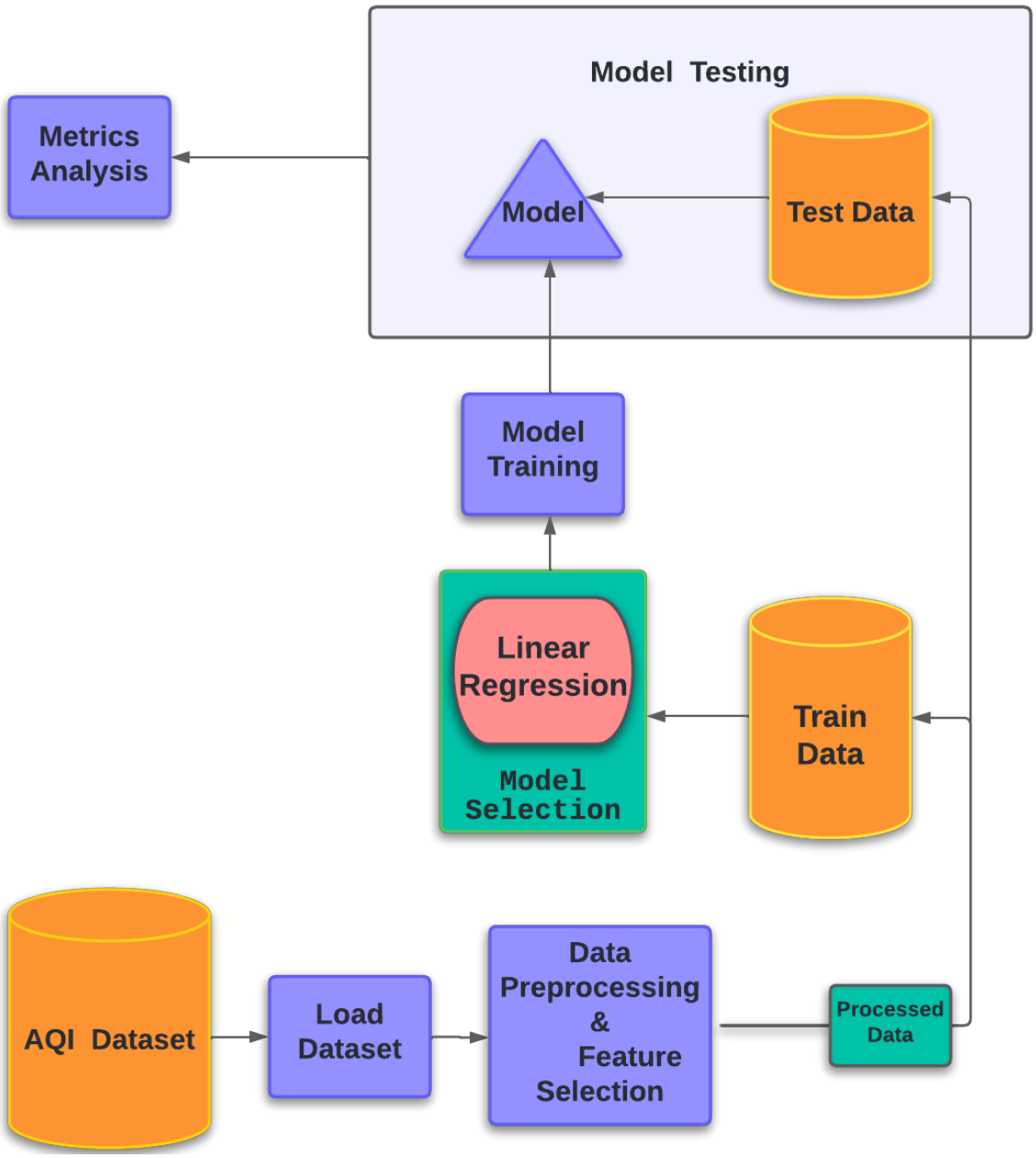


**Figure 4.1:** Architectural Model

In our research, we have introduced our dataset from the website of **Central Pollution Control Board** [2]. The data initially was unstructured so once it is loaded and preprocessed wherein we started structuring our data as we had adjusted all null values by the average of remaining values for each pollutant and have managed to complete the database. Also removed any of the extra pollutants outside from the scope of our study as using more than eight pollutants was giving a negative impact on calculating air-quality-index and out of them we chose six major pollutants which are Sulphur Dioxide ($SO_2$), Nitrogen Dioxide ($NO_2$), Particulate Matter 2.5 ($PM_{2.5}$), Particulate Matter 10 ($PM_{10}$), Carbon Monoxide (CO) and Ozone ($O_3$) as these have the major impact upon air-quality-index. Post the cleaning of data, it will be analyzed creating insightful plots like line plots and scatter plots. After which takes place the process of feature selection and the data will split into "Train dataset" and "Test dataset".

Train dataset will be operated upon the algorithm of Linear Regression and model training line plots and scatter plots will be created as per the yearly average of pollutants like Sulphur Dioxide ($SO_2$), Nitrogen Dioxide ($NO_2$), Particulate Matter 2.5 ($PM_{2.5}$), Particulate Matter 10 ($PM_{10}$), Carbon Monoxide (CO) and Ozone ($O_3$) for each Nation Capital Region city that are Delhi, Ghaziabad, Meerut, Noida and lastly Bulandshahr.

Test dataset will be further used to test the model already trained to derive metric analysis with utmost accuracy among all the four algorithms to predict AQI for future years that are 2022 to 2026.

## 4.2 Database Specifications

**Our dataset contains the following features with Per year average of all cities and each and every pollutant described on page chap:intro:**

1. Year : It represents per year's average value of the recorded pollutant of a particular city.

2. $PM_{2.5}$ : It represents the value of pollutants with diameter less than 2.5 micrometres like Dust, spores and pollen.

3. $PM_{10}$ : It represents the value of pollutants with diameter less than or equal to ten micrometers

4. $O_3$ : Ozone

5. $SO_2$ : Sulphur Dioxide

6. $NO_2$ : Nitrogen Dioxide

7. CO : Carbon Monoxide

# Chapter 5

# METHODOLOGY USED

## 5.1 Reference Range

Reference range for all the pollutants with their units in which they are calculated also including the column of category which describes us about the level of pollution in the air by dividing the range of pollutants and air-quality-index in six categories as shown below : [2]

Table 5.1: Reference Range for air pollutants and air-quality-index

| Category | Air Quality Index | $PM_{2.5}$ (24hr avg) $\mu g/m^3$ | $PM_{10}$ (24hr avg) $\mu g/m^3$ | $O_3$ (8hr avg) $\mu g/m^3$ | $NO_2$ (1hr avg) $\mu g/m^3$ | $SO_2$ (1hr avg) $\mu g/m^3$ | CO (8hr avg) $mg/m^3$ |
|---|---|---|---|---|---|---|---|
| Good | 0 - 50 | 0 - 30 | 0 - 50 | 0 - 50 | 0 - 40 | 0 - 40 | 0 -1.0 |
| Satisfactory | 51 - 100 | 31 - 60 | 51 - 100 | 51 - 100 | 41 - 80 | 41 - 80 | 1.1 - 2.0 |
| Moderate | 101 - 200 | 61 - 90 | 101 - 250 | 101 - 168 | 81 - 180 | 81 - 380 | 2.1 - 10 |
| Poor | 201 - 300 | 91 - 120 | 251 - 350 | 169 - 208 | 181 - 280 | 381 - 800 | 10.1 - 17 |
| Very Poor | 301 - 400 | 121 - 250 | 351 - 430 | 209 - 748 | 281 - 400 | 801 - 1600 | 17.1 - 34 |
| Severe | 401 - 500 | >=251 | >=431 | >=748 | >=401 | >=1601 | >=34.1 |

## 5.2 Linear Regression

Linear Regression tries to establish the link between dependent and the independent variable where in our case dependent variable is the air pollutant concentration as 'Y' and the independent variable is the year as 'X' because we are considering per year average values of pollutants of all the five cities of National Capital Region.

A Linear equation is given by:

$$Y = mX + b + e \tag{5.1}$$

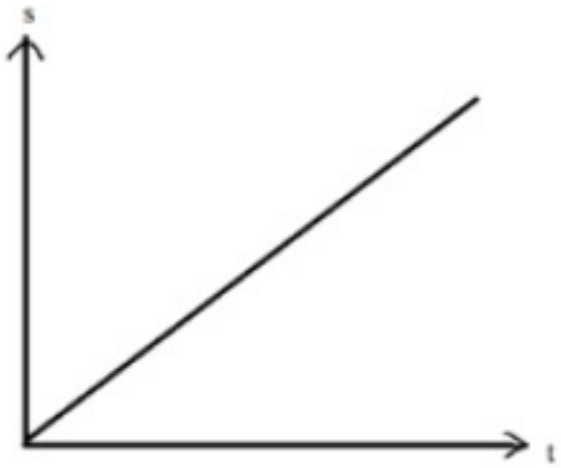

**Figure 5.1:** Best Fit Line

Where b is the intercept, m is the slope of the line,e is the error term.

$$m = \frac{\Sigma[(X - \bar{X})(Y - \bar{Y})]}{\Sigma(x - \bar{x})^2} \tag{5.2}$$

Here, $\bar{X}$ and $\bar{Y}$ are mean value of X and Y respectively.

Accuracy in Linear Regression is calculated by computing the loss, $R^2$.$R^2$ can be defined as:

$$R^2 = \frac{\Sigma(Y_{\mathrm{p}} - \bar{Y})_2}{\Sigma(Y - \bar{Y})_2} \tag{5.3}$$

Where, $Y_{\mathrm{p}}$ is the predicted value of the pollutants on Y-axis.

# Chapter 6

# IMPLEMENTATION

- Initial imports

  We have mainly used these four libraries :

  1. **Pandas**
     We used Pandas for data analysis and machine learning tasks.

  2. **Scipy.stats**
     stats.linregress uses least-squares method to find the parameters that build a linear relationship between two sets of variables.

  3. **Matplotlib**
     We used this library to plot and present our graphical output.

```
In [1]:
1 import pandas as pd
2 import numpy as np
3 from matplotlib import pyplot as plt

In [17]:
1 import matplotlib.pyplot as plt
2 from scipy import stats
```

**Figure 6.1:** Imports done initially

In further pages, we have displayed the training model and then the prediction of future years that is 2022 to 2026 for each of the five cities in the National Capital Region namely Bulandshahr, Ghaziabad, Meerut, Delhi and Noida with each of the 6 pollutants which are :

1. Sulphur Dioxide ($SO_2$)
2. Nitrogen Dioxide ($NO_2$)
3. Particulate Matter 2.5 ($PM_{2.5}$)
4. Particulate Matter 10 ($PM_{10}$)
5. Carbon Monoxide (CO)
6. Ozone ($O_3$)

- Bulandshahr $PM_{2.5}$ Yearly Average Calculation

```
import matplotlib.pyplot as plt
from scipy import stats
```

```
byr=pd.read_csv("bsr_train.csv")

```

```
x = byr['Year']

y = byr['pm2.5']

slope, intercept, r, p, std_err = stats.linregress(x, y)

def myfunc(x):
  return slope * x + intercept


mymodel = list(map(myfunc, x))
plt.xticks(x)
plt.ylim(100,350)
plt.title("Bulandshahr_PM2.5 yearly average")
plt.xlabel("Year")
plt.ylabel("PM2.5")
plt.scatter(x, y)
plt.grid()
plt.plot(x, mymodel)
plt.show()
```

**Figure 6.2:** Bulandshahr $PM_{2.5}$ training code

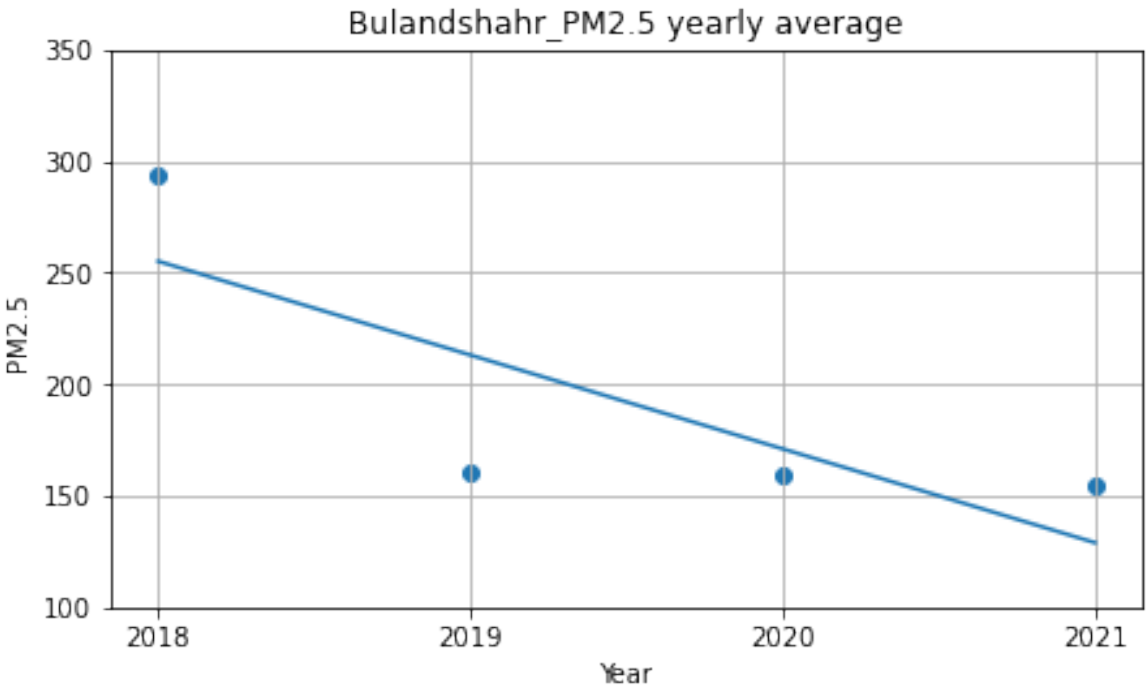


**Figure 6.3:** Bulandshahr $PM_{2.5}$ training output

- Bulandshahr $PM_{2.5}$ Yearly Average Calculation Testing for 2022 - 2026

```
slope, intercept, r, p, std_err = stats.linregress(x, y)

def myfunc(x):
  return slope * x + intercept

mymodel = list(map(myfunc, x))
plt.xticks([2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(100,300)
plt.title("Bulandshahr_PM2.5 yearly average")
plt.xlabel("Year")
plt.ylabel("PM2.5")
plt.scatter(x, y,color="blue")
plt.plot(x, mymodel)

plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [18/2.54, 10/2.54]
plt.grid()

plt.show()
```

**Figure 6.4:** Bulandshahr $PM_{2.5}$ testing code

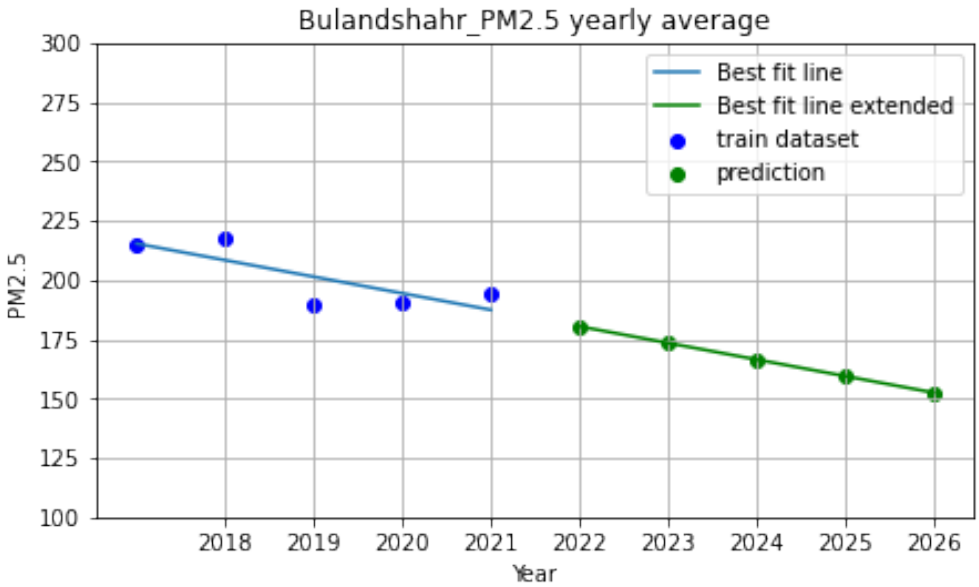


**Figure 6.5:** Bulandshahr $PM_{2.5}$ testing output

- Bulandshahr $NO_2$ Yearly Average Calculation

```
byr=pd.read_csv("bsr_train.csv")
bsr_t=byr.drop(labels=['pm2.5','pm10','so2','o3','co'], axis=1)

x = bsr_t['Year']
y = bsr_t['no2']
slpe, intrcpt, r1, p1, stderr= stats.linregress(x, y)
def myfunc(x):
  return slpe * x + intrcpt

mymodel = list(map(myfunc, x))
plt.xticks(x)
plt.ylim(0,100)
plt.title("Bulandshahr_NO2 yearly average")
plt.xlabel("Year")
plt.ylabel("NO2")
plt.scatter(x, y)
plt.plot(x, mymodel)
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()
plt.show()
```

**Figure 6.6:** Bulandshahr $NO_2$ training code

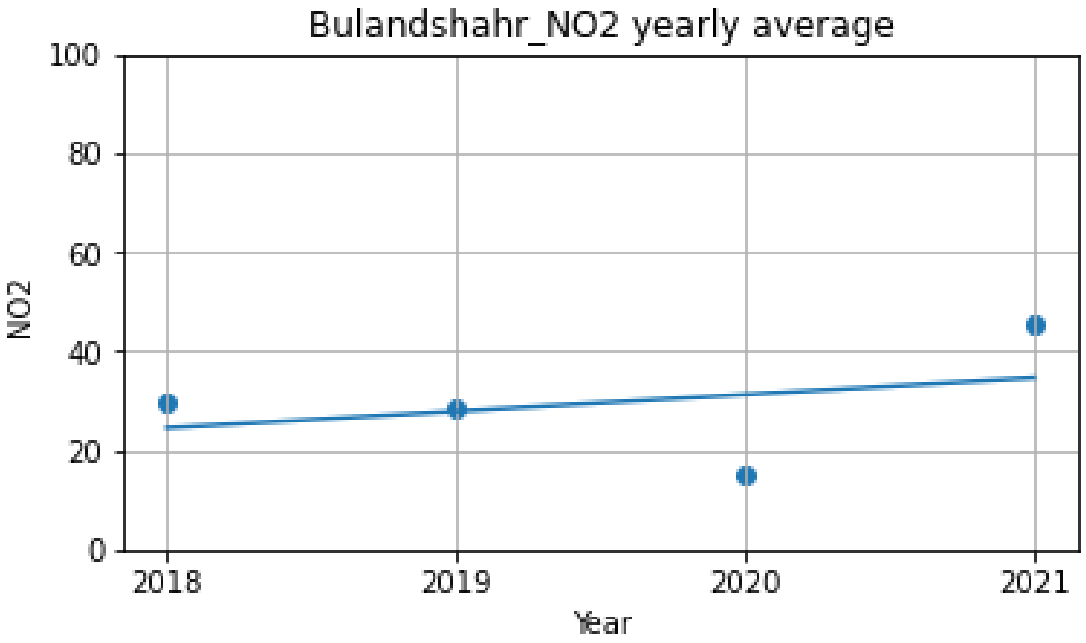


**Figure 6.7:** Bulandshahr $NO_2$ training output

- Bulandshahr $NO_2$ Yearly Average Calculation Testing for 2022 - 2026

```
bsr_test = pd.read_csv("bsr_test.csv")
bsr_test=bsr_test.drop(labels=['pm10','pm2.5','so2','o3','co'], axis=1)
x_t = bsr_test['Year']
y_t = bsr_test['no2']
slpe, intrcpt, r1, p1, stderr= stats.linregress(x, y)
def myfunc(x):
  return slpe * x + intrcpt

mymodel = list(map(myfunc, x))
plt.xticks([2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(0,100)
plt.title("Bulandshahr_NO2 yearly average")
plt.xlabel("Year")
plt.ylabel("NO2")
plt.scatter(x, y,color="blue")
plt.plot(x, mymodel)
plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()


plt.show()
```

**Figure 6.8:** Bulandshahr $NO_2$ testing code

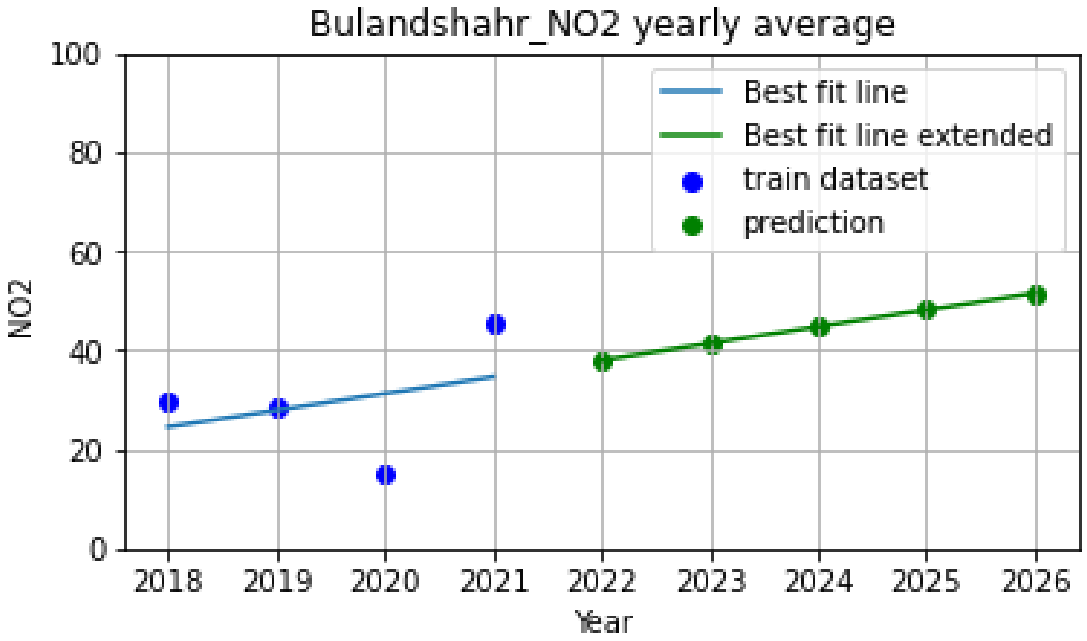


**Figure 6.9:** Bulandshahr $NO_2$ testing output

- Bulandshahr $SO_2$ Yearly Average Calculation

```
byr=pd.read_csv("bsr_train.csv")
bsr_t=byr.drop(labels=['pm2.5','pm10','no2','o3','co'], axis=1)

x = bsr_t['Year']
y = bsr_t['so2']
slpe, intrcpt, r1, p1, stderr= stats.linregress(x, y)
def myfunc(x):
  return slpe * x + intrcpt

mymodel = list(map(myfunc, x))
plt.xticks(x)
plt.ylim(0,100)
plt.title("Bulandshahr_SO2 yearly average")
plt.xlabel("Year")
plt.ylabel("SO2")
plt.scatter(x, y)
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()
plt.plot(x, mymodel)
plt.show()

```

**Figure 6.10:** Bulandshahr $SO_2$ training code

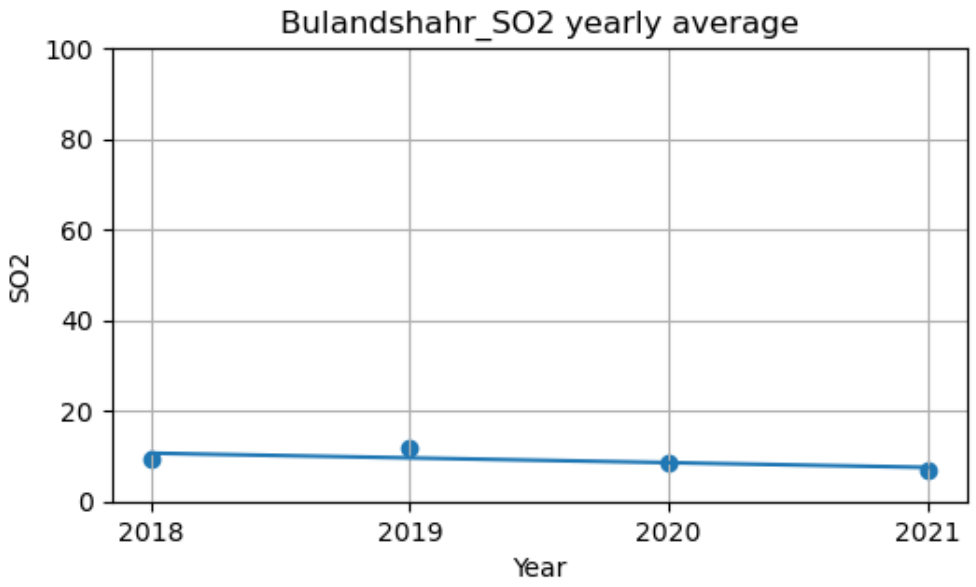


**Figure 6.11:** Bulandshahr $SO_2$ training output

- Bulandshahr $SO_2$ Yearly Average Calculation Testing for 2022 - 2026

```
bsr_test = pd.read_csv("bsr_train.csv")
bsr_test=bsr_test.drop(labels=['pm10','pm2.5','no2','o3','co'], axis=1)
x_t = bsr_test['Year']
y_t = bsr_test['so2']
slpe, intrcpt, r1, p1, stderr= stats.linregress(x_t, y_t)
def myfunc(x_t):
  return slpe * x_t + intrcpt

mymodel = list(map(myfunc, x_t))

plt.xticks([2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(0,30)
plt.title("Bulandshahr_SO2 yearly average")
plt.xlabel("Year")
plt.ylabel("SO2")
plt.scatter(x_t, y_t,color="blue")
plt.plot(x_t, mymodel)
plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()

plt.show()
```

**Figure 6.12:** Bulandshahr $SO_2$ testing code

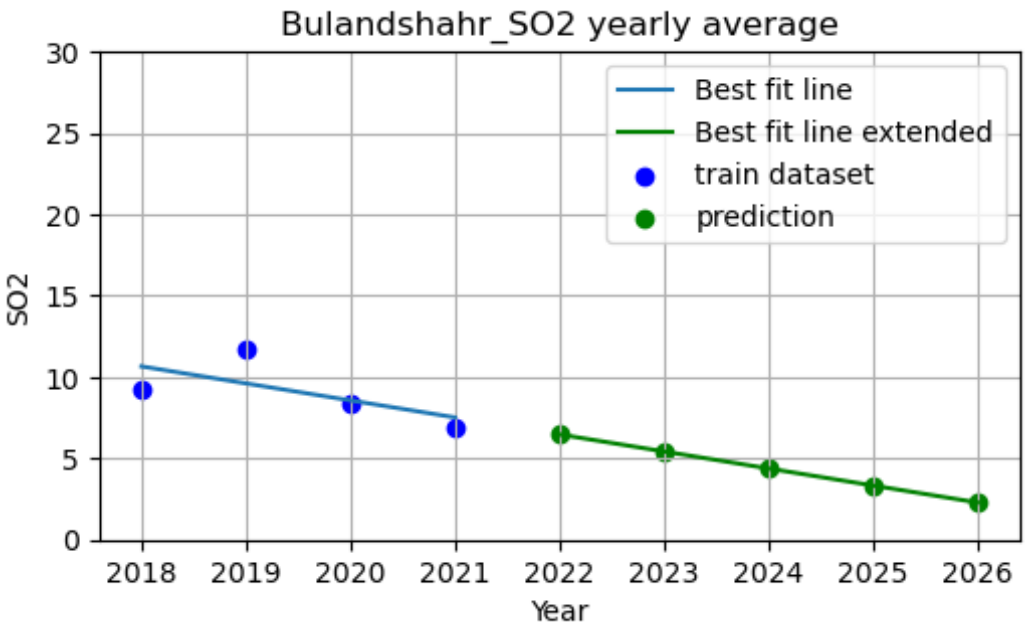


**Figure 6.13:** Bulandshahr $SO_2$ testing output

- Bulandshahr $O_3$ Yearly Average Calculation

```
byr=pd.read_csv("bsr_train.csv")
bsr_t=byr.drop(labels=['pm2.5','pm10','no2','so2','co'], axis=1)

x = bsr_t['Year']
y = bsr_t['o3']
slpe, intrcpt, r1, p1, stderr= stats.linregress(x, y)
def myfunc(x):
  return slpe * x + intrcpt

mymodel = list(map(myfunc, x))
plt.xticks(x)
plt.ylim(30,50)
plt.title("Bulandshahr_O3 yearly average")
plt.xlabel("Year")
plt.ylabel("O3")
plt.scatter(x, y)
plt.plot(x, mymodel)


plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()

plt.show()


```

**Figure 6.14:** Bulandshahr $O_3$ training code

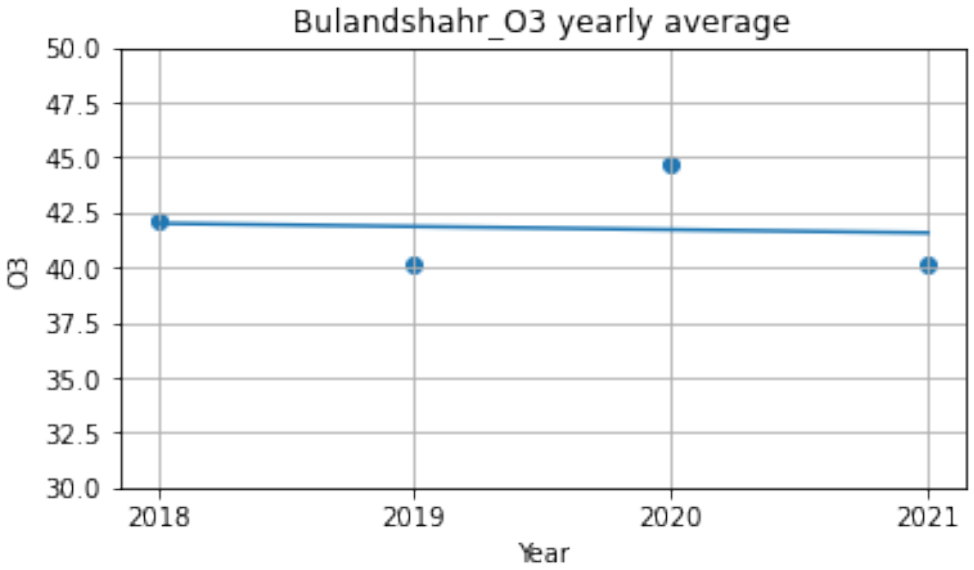


**Figure 6.15:** Bulandshahr $O_3$ training output

- Bulandshahr $O_3$ Yearly Average Calculation Testing for 2022 - 2026

```
bsr_test = pd.read_csv("bsr_test.csv")
bsr_test=bsr_test.drop(labels=['pm10','pm2.5','no2','so2','co'], axis=1)
x_t = bsr_test['Year']
y_t = bsr_test['o3']
slpe, intrcpt, r1, p1, stderr= stats.linregress(x, y)
def myfunc(x):
  return slpe * x + intrcpt

mymodel = list(map(myfunc, x))
plt.xticks([2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(10,80)
plt.title("Bulandshahr_O3 yearly average")
plt.xlabel("Year")
plt.ylabel("O3")
plt.scatter(x, y,color="blue")
plt.plot(x, mymodel)
plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()


plt.show()
```

**Figure 6.16:** Bulandshahr $O_3$ testing code

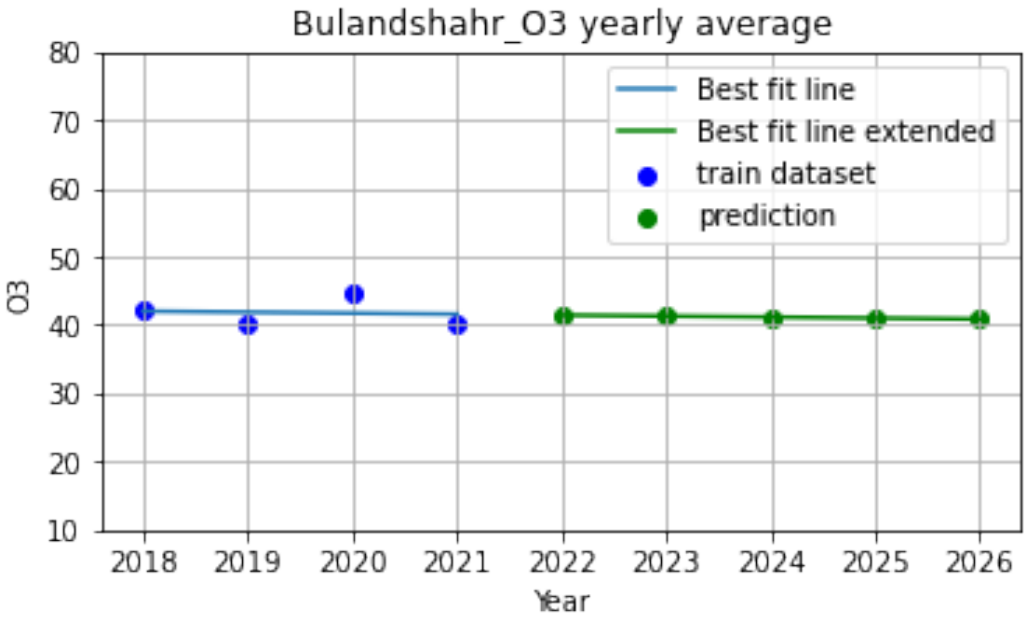


**Figure 6.17:** Bulandshahr $O_3$ testing output

- Bulandshahr CO Yearly Average Calculation

```
byr=pd.read_csv("bsr_train.csv")
bsr_t=byr.drop(labels=['pm2.5','pm10','so2','o3','no2'], axis=1)

x = bsr_t['Year']
y = bsr_t['co']
slpe, intrcpt, r1, p1, stderr= stats.linregress(x, y)
def myfunc(x):
  return slpe * x + intrcpt

mymodel = list(map(myfunc, x))
plt.xticks(x)
plt.ylim(0,100)
plt.title("Bulandshahr_CO yearly average")
plt.xlabel("Year")
plt.ylabel("CO")
plt.scatter(x, y)
plt.plot(x, mymodel)

plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()
plt.show()
```

**Figure 6.18:** Bulandshahr CO training code

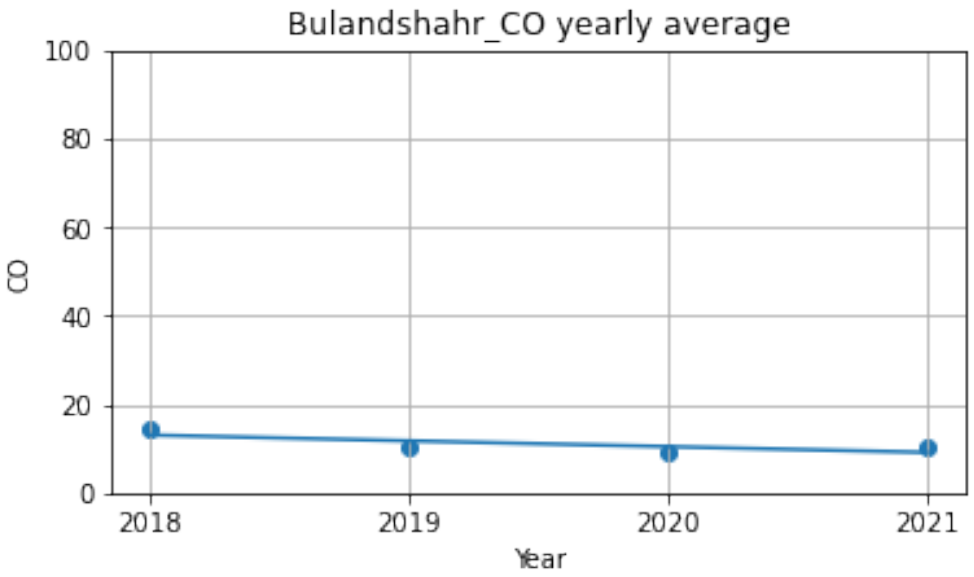


**Figure 6.19:** Bulandshahr CO training output

- Bulandshahr CO Yearly Average Calculation Testing for 2022 - 2026

```
bsr_test = pd.read_csv("bsr_test.csv")
bsr_test=bsr_test.drop(labels=['pm10','pm2.5','so2','o3','no2'], axis=1)
x_t = bsr_test['Year']
y_t = bsr_test['co']
slpe, intrcpt, r1, p1, stderr= stats.linregress(x, y)
def myfunc(x):
  return slpe * x + intrcpt

mymodel = list(map(myfunc, x))
plt.xticks([2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(0,100)
plt.title("Bulandshahr_CO yearly average")
plt.xlabel("Year")
plt.ylabel("CO")
plt.scatter(x, y,color="blue")
plt.plot(x, mymodel)

plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()


plt.show()
```

**Figure 6.20:** Bulandshahr CO testing code

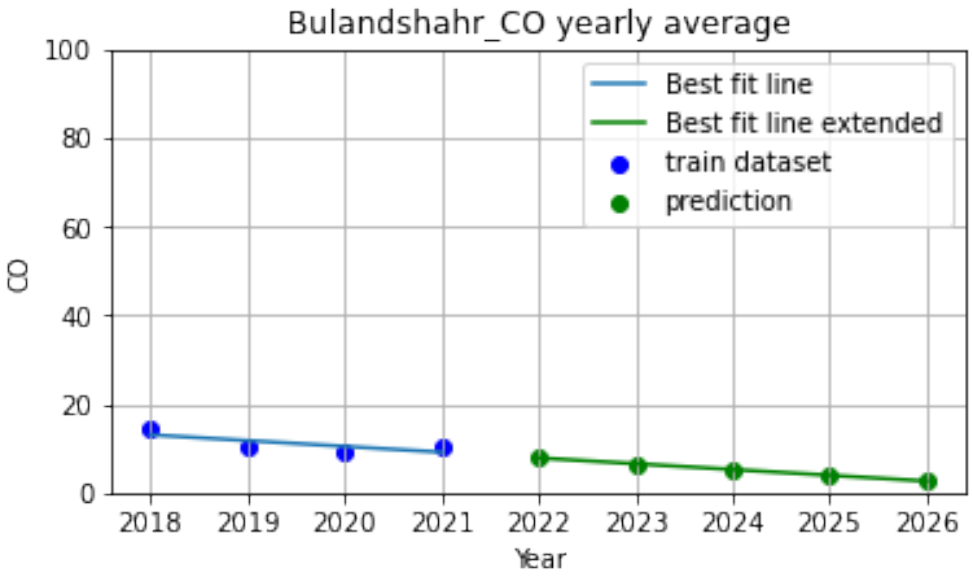


**Figure 6.21:** Bulandshahr CO testing output

- Bulandshahr $PM_{10}$ Yearly Average Calculation

```
byr=pd.read_csv("bsr_train.csv")
bsr_t=byr.drop(labels=['pm2.5','no2','so2','o3','co'], axis=1)

x = bsr_t['Year']
y = bsr_t['pm10']
slpe, intrcpt, r1, p1, stderr= stats.linregress(x, y)
def myfunc(x):
  return slpe * x + intrcpt

mymodel = list(map(myfunc, x))
plt.xticks(x)
plt.ylim(100,200)
plt.title("Bulandshahr_PM10 yearly average")
plt.xlabel("Year")
plt.ylabel("PM10")
plt.scatter(x, y)
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()
plt.plot(x, mymodel)
plt.show()
```

**Figure 6.22:** Bulandshahr $PM_{10}$ training code

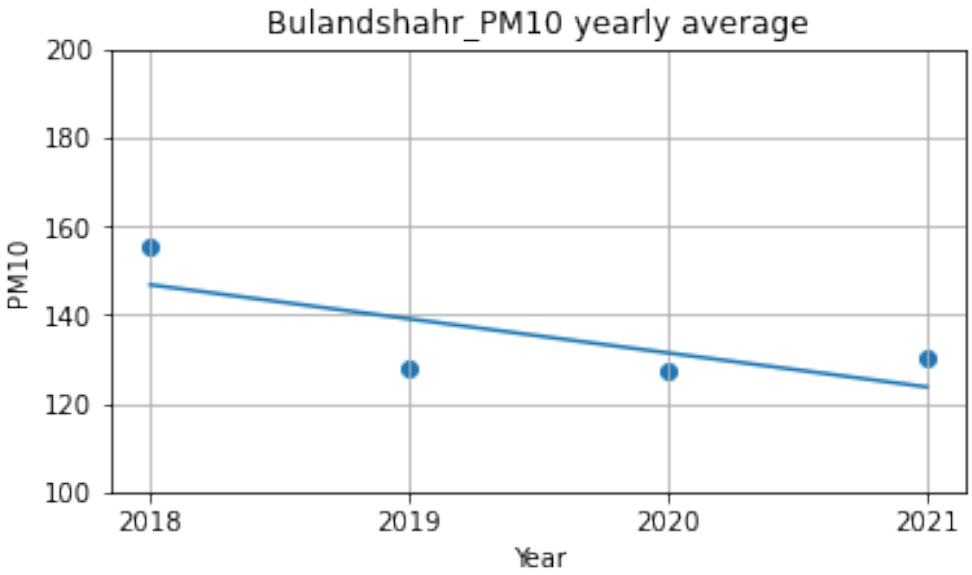


**Figure 6.23:** Bulandshahr $PM_{10}$ training output

- Bulandshahr $PM_{10}$ Yearly Average Calculation Testing for 2022 - 2026

```
bsr_test = pd.read_csv("bsr_test.csv")
bsr_test=bsr_test.drop(labels=['no2','pm2.5','so2','o3','co'], axis=1)
x_t = bsr_test['Year']
y_t = bsr_test['pm10']
slpe, intrcpt, r1, p1, stderr= stats.linregress(x, y)
def myfunc(x):
  return slpe * x + intrcpt

mymodel = list(map(myfunc, x))
plt.xticks([2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(0,200)
plt.title("Bulandshahr PM10 yearly average")
plt.xlabel("Year")
plt.ylabel("PM10")
plt.scatter(x, y,color="blue")
plt.plot(x, mymodel)
plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.grid()
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.show()
```

**Figure 6.24:** Bulandshahr $PM_{10}$ testing code

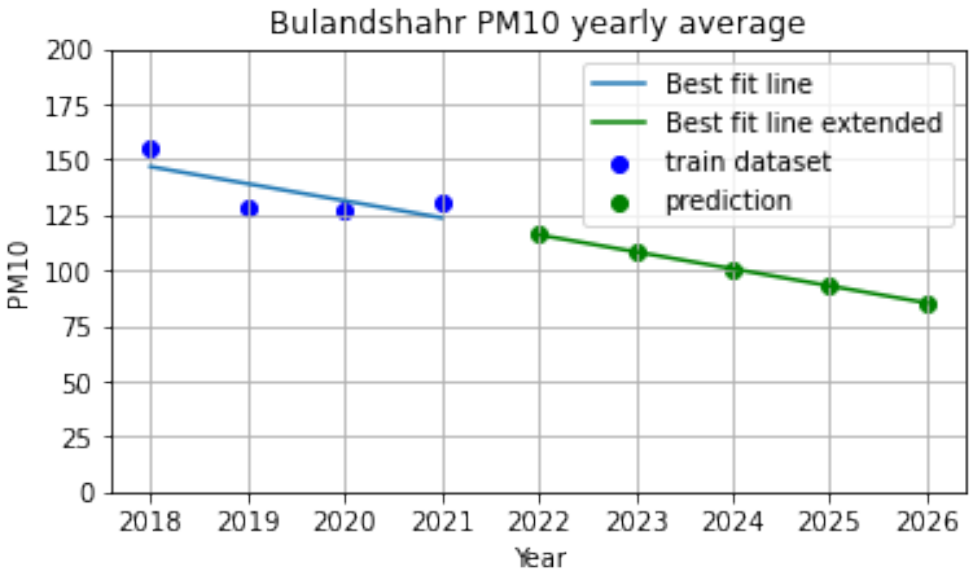


**Figure 6.25:** Bulandshahr $PM_{10}$ testing output

- Ghaziabad $PM_{2.5}$ Yearly Average Calculation

```
gzd=pd.read_csv("gzd train.csv")
```

```
gzd_pm25 = gzd.drop(labels=['pm10','o3','no2','so2','co'], axis=1)
```

```
gx = gzd_pm25['Year']

gy = gzd_pm25['pm25']

slope, intercept, r, p, std_err = stats.linregress(gx, gy)

def myfunc(gx):
  return slope * gx + intercept

# next_year_value=myfunc

mymodel = list(map(myfunc, gx))
plt.xticks(gx)
plt.ylim(120,400)
plt.title("Ghaziabad_PM2.5 yearly average")
plt.xlabel("Year")
plt.ylabel("PM2.5")
plt.scatter(gx, gy)
plt.grid()
plt.plot(gx, mymodel)
plt.show()
```

**Figure 6.26:** Ghaziabad $PM_{2.5}$ training code

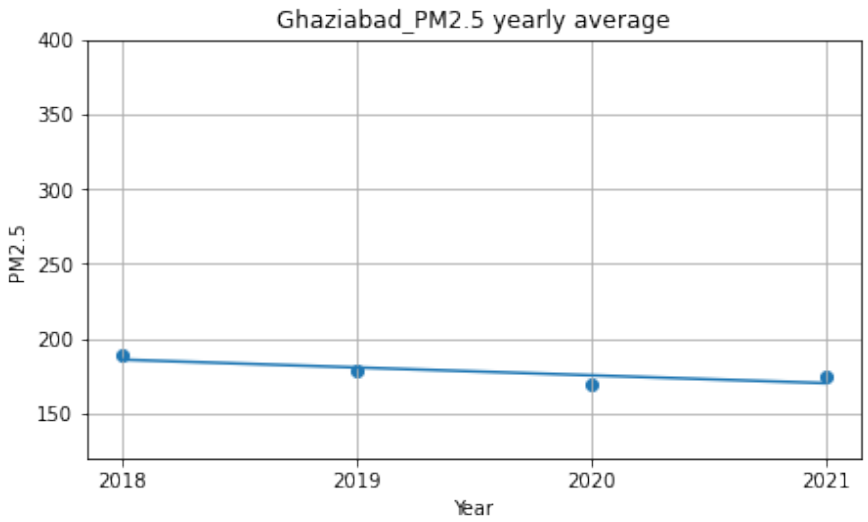


**Figure 6.27:** Ghaziabad $PM_{2.5}$ training output

- Ghaziabad $PM_{2.5}$ Yearly Average Calculation Testing for 2022 - 2026

```
1
2  slope, intercept, r, p, std_err = stats.linregress(gx, gy)
3
4  def myfunc(gx):
5    return slope * gx + intercept
6
7  mymodel = list(map(myfunc, gx))
8  plt.xticks([2018,2019,2020,2021,2022,2023,2024,2025,2026])
9  plt.ylim(0,400)
10 plt.title("Ghaziabad PM 2.5 yearly average")
11 plt.xlabel("Year")
12 plt.ylabel("PM2.5")
13 plt.scatter(gx, gy,color="blue")
14 plt.plot(gx, mymodel)
15 plt.scatter(2022,myfunc(2022),color="green")
16 plt.scatter(2023,myfunc(2023),color="green")
17 plt.scatter(2024,myfunc(2024),color="green")
18 plt.scatter(2025,myfunc(2025),color="green")
19 plt.scatter(2026,myfunc(2026),color="green")
20 X=[2022,2023,2024,2025,2026]
21 Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
22 plt.plot(X,Y,color="green")
23 plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
24 plt.grid()
25
26 plt.show()
```

**Figure 6.28:** Ghaziabad $PM_{2.5}$ testing code

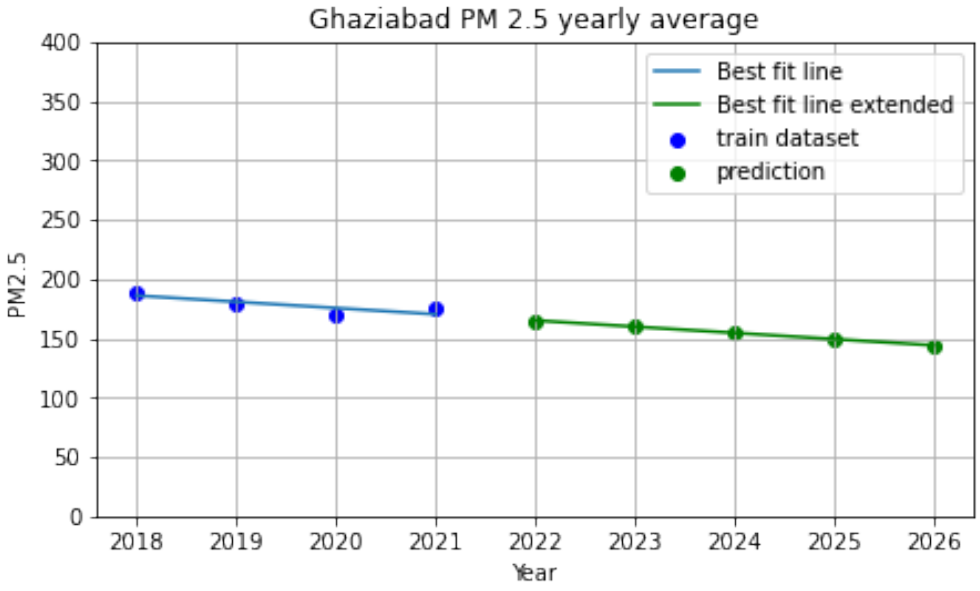


**Figure 6.29:** Ghaziabad $PM_{2.5}$ testing output

- Ghaziabad $NO_2$ Yearly Average Calculation

```
gzd=pd.read_csv("gzd train.csv")
gzd_t=gzd.drop(labels=['pm25','pm10','so2','o3','co'], axis=1)

gx = gzd_t['Year']
gy = gzd_t['no2']
slpe, intrcpt, r1, p1, stderr= stats.linregress(gx, gy)
def myfunc(gx):
  return slpe * gx + intrcpt

mymodel = list(map(myfunc, gx))
plt.xticks(gx)
plt.ylim(0,80)
plt.title("Ghaziabad NO2 yearly average")
plt.xlabel("Year")
plt.ylabel("NO2")
plt.scatter(gx, gy)
plt.plot(gx, mymodel)
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()
plt.show()
```

**Figure 6.30:** Ghaziabad $NO_2$ training code

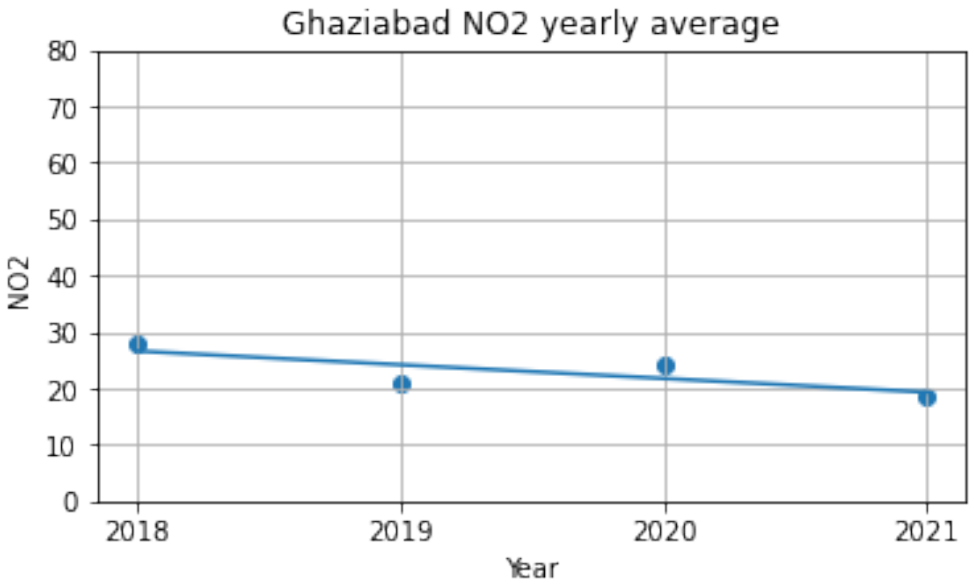


**Figure 6.31:** Ghaziabad $NO_2$ training output

- Ghaziabad $NO_2$ Yearly Average Calculation Testing for 2022 - 2026

```
gzd_test = pd.read_csv("gzd test.csv")
gzd_test=gzd_test.drop(labels=['pm10','pm25','so2','o3','co'], axis=1)
gx_t = gzd_test['Year']
gy_t = gzd_test['no2']
slpe, intrcpt, r1, p1, stderr= stats.linregress(gx, gy)
def myfunc(gx):
  return slpe * gx + intrcpt

mymodel = list(map(myfunc, gx))
plt.xticks([2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(0,80)
plt.title("Ghaziabad NO2 yearly average")
plt.xlabel("Year")
plt.ylabel("NO2")
plt.scatter(gx, gy,color="blue")
plt.plot(gx, mymodel)

plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()


plt.show()
```

**Figure 6.32:** Ghaziabad $NO_2$ testing code

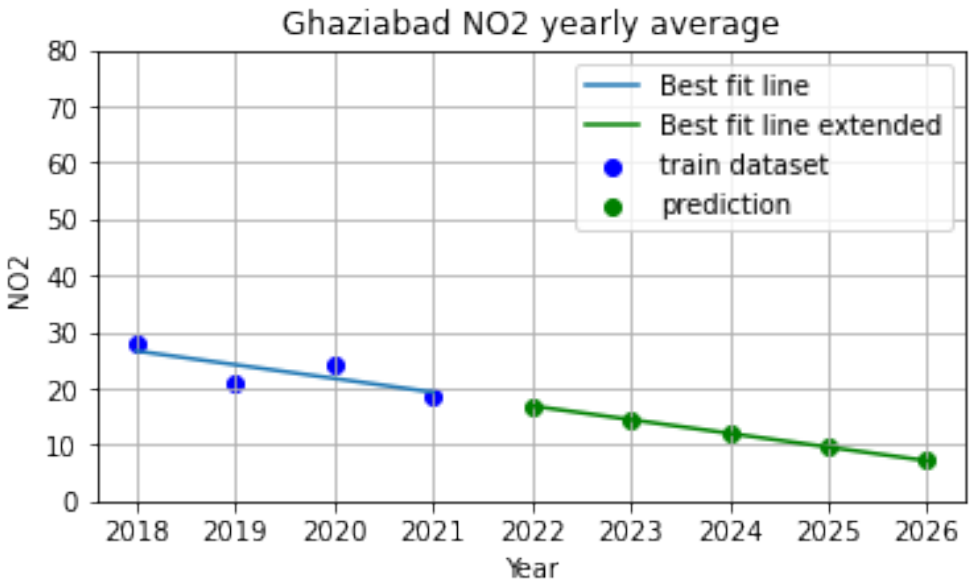


**Figure 6.33:** Ghaziabad $NO_2$ testing output

- Ghaziabad $SO_2$ Yearly Average Calculation

```
gzd=pd.read_csv("gzd train.csv")
gzd_t=gzd.drop(labels=['pm25','pm10','no2','o3','co'], axis=1)

gx = gzd_t['Year']
gy = gzd_t['so2']
slpe, intrcpt, r1, p1, stderr= stats.linregress(gx, gy)
def myfunc(gx):
  return slpe * gx + intrcpt

mymodel = list(map(myfunc, gx))
plt.xticks(gx)
plt.ylim(0,80)
plt.title("Ghaziabad SO2 yearly average")
plt.xlabel("Year")
plt.ylabel("SO2")
plt.scatter(gx, gy)
plt.plot(gx, mymodel)
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()

plt.show()
```

**Figure 6.34:** Ghaziabad $SO_2$ training code

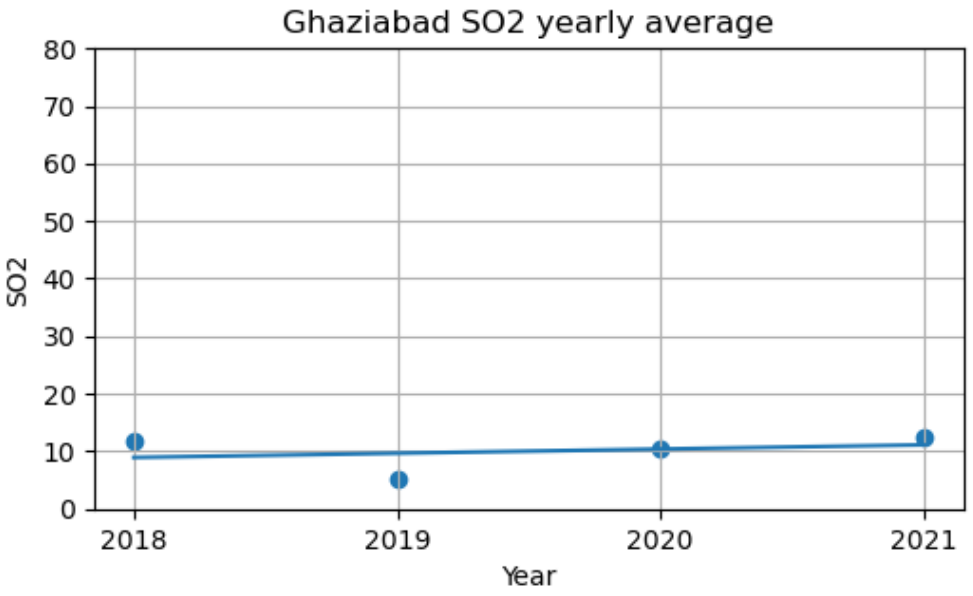


**Figure 6.35:** Ghaziabad $SO_2$ training output

- Ghaziabad $SO_2$ Yearly Average Calculation Testing for 2022 - 2026

```
gzd_test = pd.read_csv("gzd test.csv")
gzd_test=gzd_test.drop(labels=['pm10','pm25','no2','o3','co'], axis=1)
gx_t = gzd_test['Year']
gy_t = gzd_test['so2']
slpe, intrcpt, r1, p1, stderr= stats.linregress(gx, gy)
def myfunc(gx):
  return slpe * gx + intrcpt

mymodel = list(map(myfunc, gx))
plt.xticks([2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(0,80)
plt.title("Ghaziabad SO2 yearly average")
plt.xlabel("Year")
plt.ylabel("SO2")
plt.scatter(gx, gy,color="blue")
plt.plot(gx, mymodel)
plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()



plt.show()
```

**Figure 6.36:** Ghaziabad $SO_2$ testing code

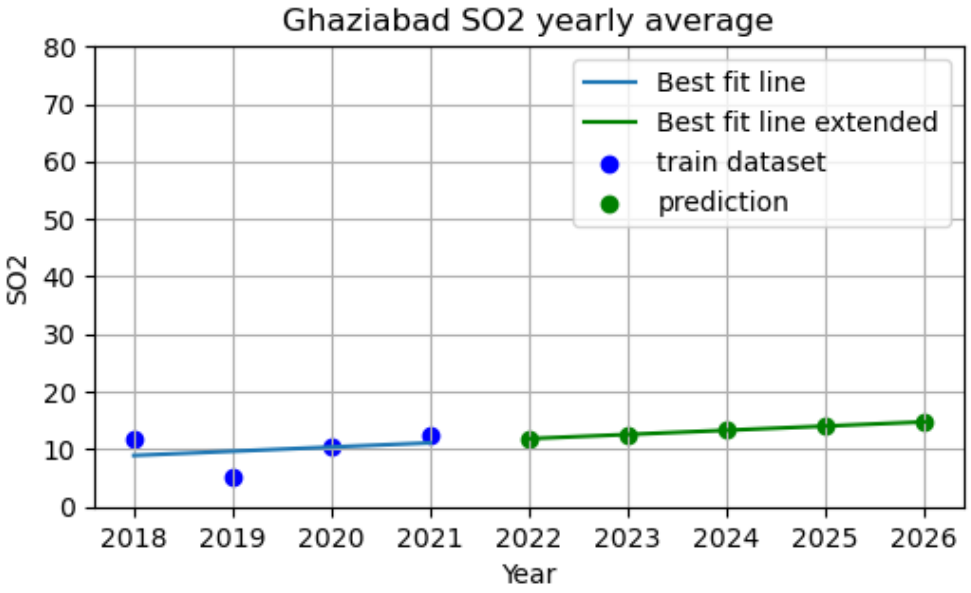


**Figure 6.37:** Ghaziabad $SO_2$ testing output

- Ghaziabad $O_3$ Yearly Average Calculation

```
gzd=pd.read_csv("gzd train.csv")
gzd_t=gzd.drop(labels=['pm25','pm10','no2','so2','co'], axis=1)

gx = gzd_t['Year']
gy = gzd_t['o3']
slpe, intrcpt, r1, p1, stderr= stats.linregress(gx, gy)
def myfunc(gx):
  return slpe * gx + intrcpt

mymodel = list(map(myfunc, gx))
plt.xticks(gx)
plt.ylim(10,80)
plt.title("Ghaziabad O3 yearly average")
plt.xlabel("Year")
plt.ylabel("O3")
plt.scatter(gx, gy)
plt.plot(gx, mymodel)


plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()

plt.show()
```

**Figure 6.38:** Ghaziabad $O_3$ training code

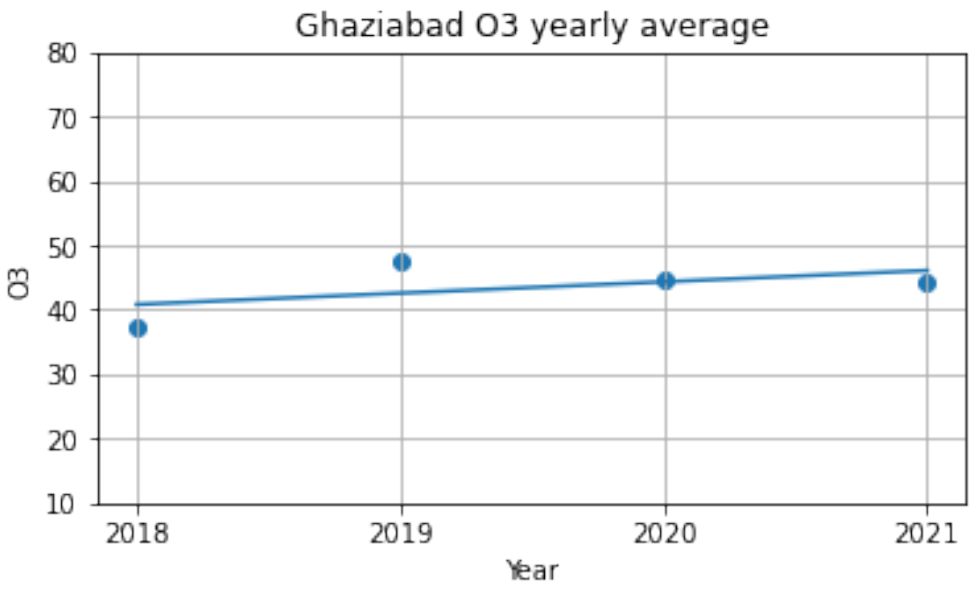


**Figure 6.39:** Ghaziabad $O_3$ training output

- Ghaziabad $O_3$ Yearly Average Calculation Testing for 2022 - 2026

```
gzd_test = pd.read_csv("gzd test.csv")
gzd_test=gzd_test.drop(labels=['pm10','pm25','no2','so2','co'], axis=1)
gx_t = gzd_test['Year']
gy_t = gzd_test['o3']
slpe, intrcpt, r1, p1, stderr= stats.linregress(gx, gy)
def myfunc(gx):
  return slpe * gx + intrcpt

mymodel = list(map(myfunc, gx))
plt.xticks([2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(10,100)
plt.title("Ghaziabad O3 yearly average")
plt.xlabel("Year")
plt.ylabel("O3")
plt.scatter(gx, gy,color="blue")
plt.plot(gx, mymodel)

plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()


plt.show()
```

**Figure 6.40:** Ghaziabad $O_3$ testing code

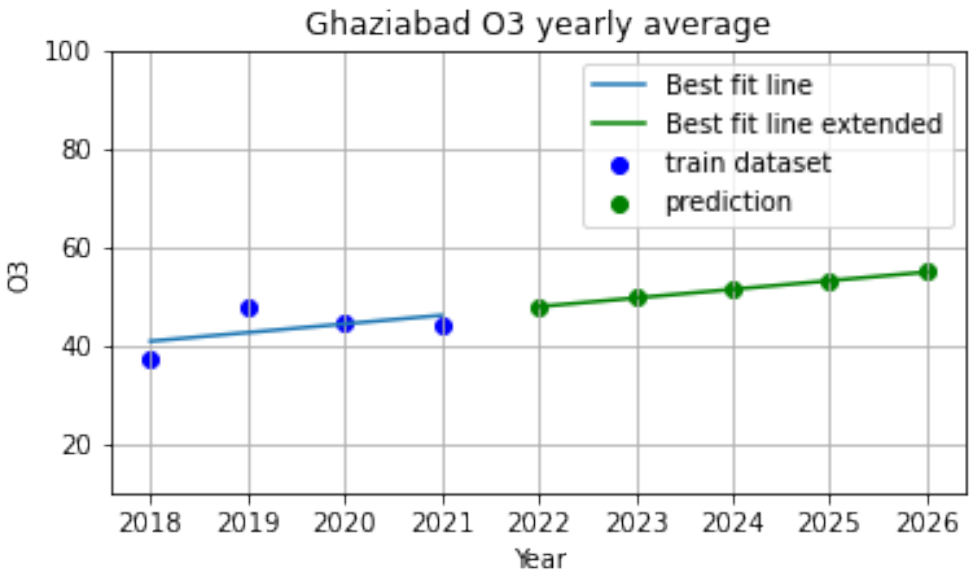


**Figure 6.41:** Ghaziabad $O_3$ testing output

- Ghaziabad CO Yearly Average Calculation

```
gzd=pd.read_csv("gzd train.csv")
gzd_t=gzd.drop(labels=['pm25','pm10','so2','o3','no2'], axis=1)

gx = gzd_t['Year']
gy = gzd_t['co']
slpe, intrcpt, r1, p1, stderr= stats.linregress(gx, gy)
def myfunc(gx):
  return slpe * gx + intrcpt

mymodel = list(map(myfunc, gx))
plt.xticks(gx)
plt.ylim(0,80)
plt.title("Ghaziabad CO yearly average")
plt.xlabel("Year")
plt.ylabel("CO")
plt.scatter(gx, gy)
plt.plot(gx, mymodel)

plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()
plt.show()
```

**Figure 6.42:** Ghaziabad CO training code

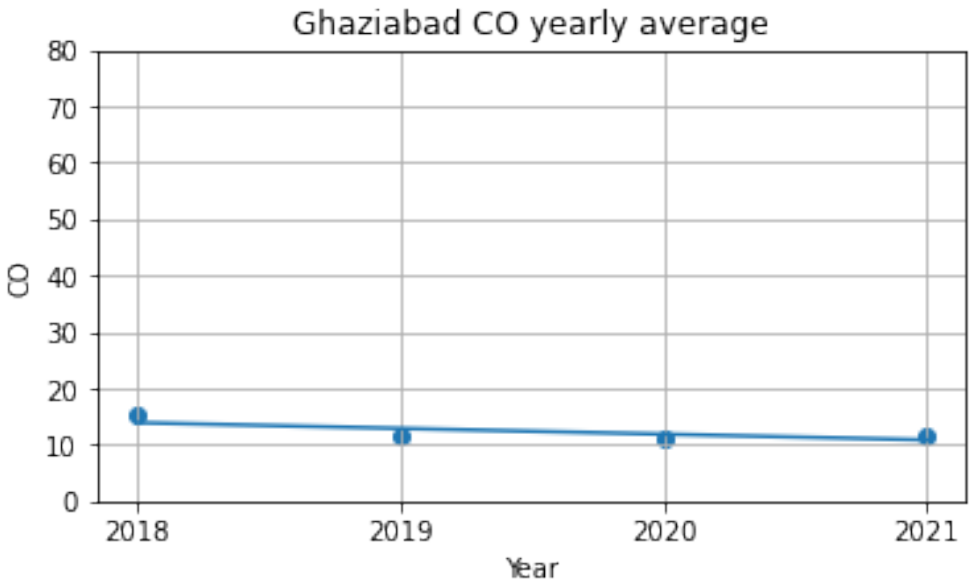


**Figure 6.43:** Ghaziabad CO training output

- Ghaziabad CO Yearly Average Calculation Testing for 2022 - 2026

```
gzd_test = pd.read_csv("gzd test.csv")
gzd_test=gzd_test.drop(labels=['pm10','pm25','so2','o3','no2'], axis=1)
gx_t = gzd_test['Year']
gy_t = gzd_test['co']
slpe, intrcpt, r1, p1, stderr= stats.linregress(gx, gy)
def myfunc(gx):
  return slpe * gx + intrcpt

mymodel = list(map(myfunc, gx))
plt.xticks([2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(0,80)
plt.title("Ghaziabad CO yearly average")
plt.xlabel("Year")
plt.ylabel("CO")
plt.scatter(gx, gy,color="blue")
plt.plot(gx, mymodel)

plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()


plt.show()
```

**Figure 6.44:** Ghaziabad CO testing code

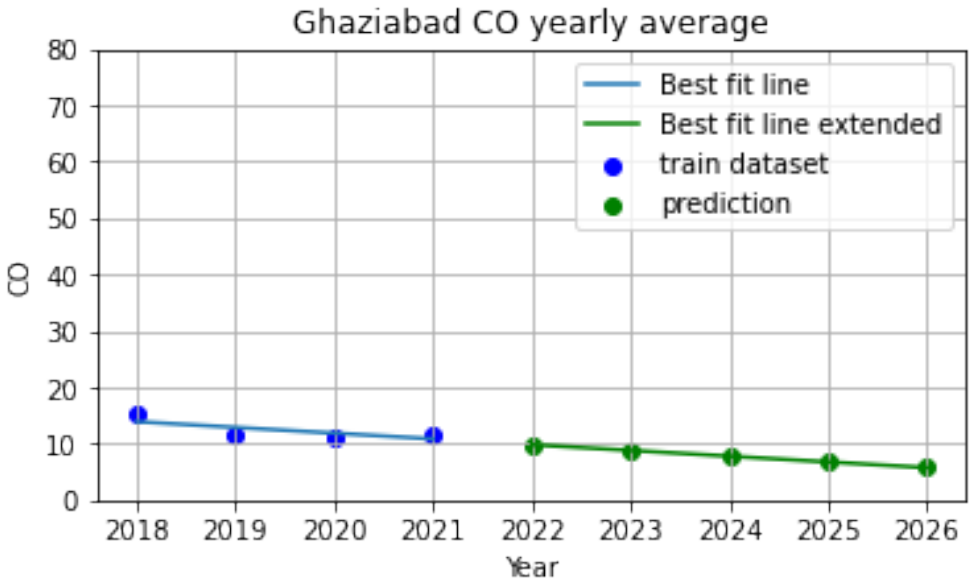


**Figure 6.45:** Ghaziabad CO testing output

- Ghaziabad $PM_{10}$ Yearly Average Calculation

```
gzd=pd.read_csv("gzd train.csv")
gzd_t=gzd.drop(labels=['pm25','no2','so2','o3','co'], axis=1)

gx = gzd_t['Year']
gy = gzd_t['pm10']
slpe, intrcpt, r1, p1, stderr= stats.linregress(gx, gy)
def myfunc(gx):
  return slpe * gx + intrcpt

mymodel = list(map(myfunc, gx))
plt.xticks(gx)
plt.ylim(50,300)
plt.title("Ghaziabad PM10 yearly average")
plt.xlabel("Year")
plt.ylabel("PM10")
plt.scatter(gx, gy)
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()
plt.plot(gx, mymodel)
plt.show()
```

**Figure 6.46:** Ghaziabad $PM_{10}$ training code

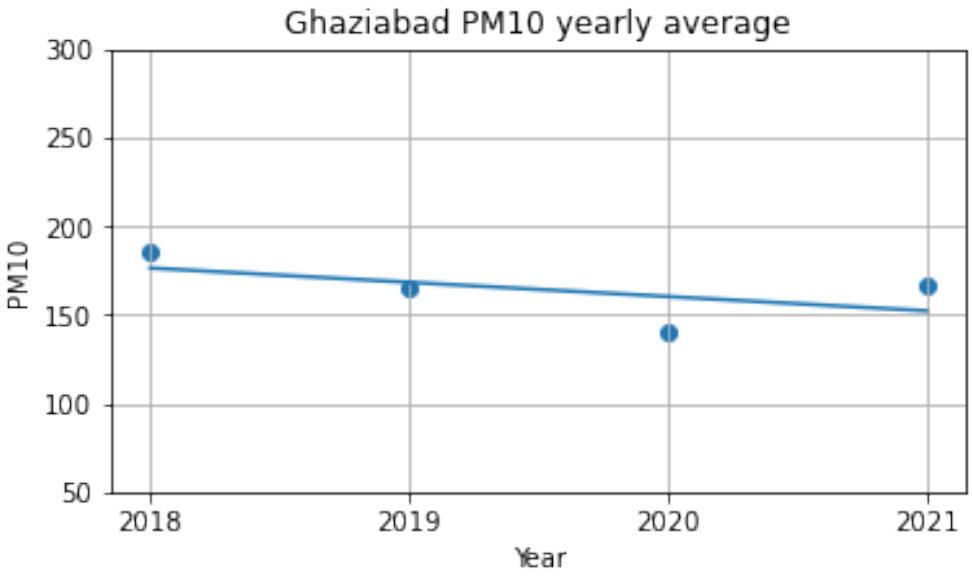


**Figure 6.47:** Ghaziabad $PM_{10}$ training output

- Ghaziabad $PM_{10}$ Yearly Average Calculation Testing for 2022 - 2026

```
gzd_test = pd.read_csv("gzd test.csv")
gzd_test=gzd_test.drop(labels=['no2','pm25','so2','o3','co'], axis=1)
gx_t = gzd_test['Year']
gy_t = gzd_test['pm10']
slpe, intrcpt, r1, p1, stderr= stats.linregress(gx, gy)
def myfunc(gx):
  return slpe * gx + intrcpt

mymodel = list(map(myfunc, gx))
plt.xticks([2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(50,300)
plt.title("Ghaziabad PM10 yearly average")
plt.xlabel("Year")
plt.ylabel("PM10")
plt.scatter(gx, gy,color="blue")
plt.plot(gx, mymodel)
plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()
plt.show()
```

**Figure 6.48:** Ghaziabad $PM_{10}$ testing code

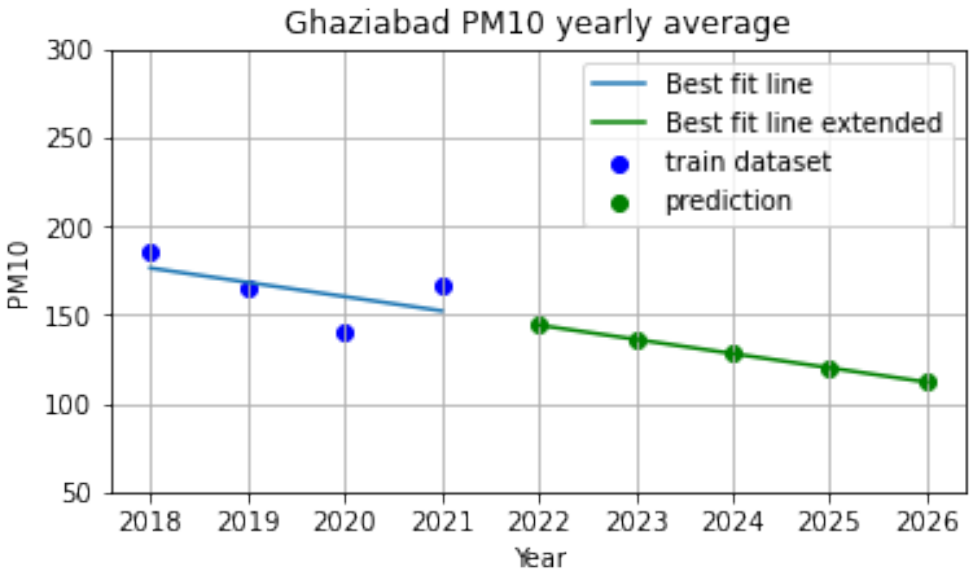


**Figure 6.49:** Ghaziabad $PM_{10}$ testing output

- Meerut $PM_{2.5}$ Yearly Average Calculation

```
mrt=pd.read_csv("mrt train .csv")
```

```
mrt_pm25 = mrt.drop(labels=['pm10','o3','no2','so2','co'], axis=1)
```

```
mx = mrt_pm25['Year']

my = mrt_pm25['pm25']

slope, intercept, r, p, std_err = stats.linregress(mx, my)

def myfunc(mx):
  return slope * mx + intercept


mymodel = list(map(myfunc, mx))
plt.xticks(mx)
plt.ylim(100,250)
plt.title("Meerut_PM2.5 yearly average")
plt.xlabel("Year")
plt.ylabel("PM2.5")
plt.scatter(mx, my)
plt.plot(mx, mymodel)
plt.grid()
plt.show()
```

**Figure 6.50:** Meerut $PM_{2.5}$ training code

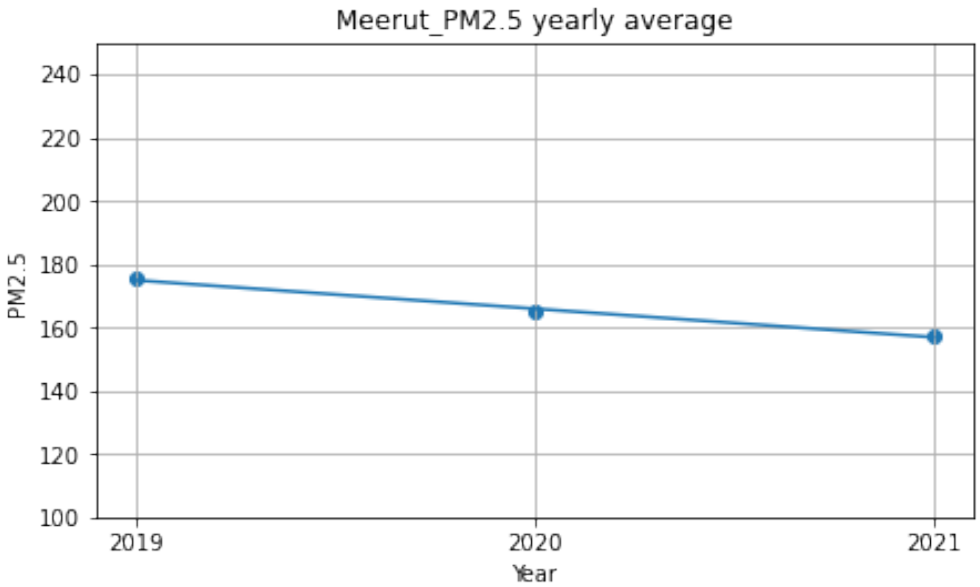


**Figure 6.51:** Meerut $PM_{2.5}$ training output

- Meerut $PM_{2.5}$ Yearly Average Calculation Testing for 2022

```python
mx=mrt_pm25['Year']
my=mrt_pm25['pm25']
slope, intercept, r, p, std_err = stats.linregress(mx, my)

def myfunc(mx):
  return slope * mx + intercept

mymodel = list(map(myfunc, mx))
plt.xticks([2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(100,250)
plt.title("Meerut PM 2.5 yearly average")
plt.xlabel("Year")
plt.ylabel("PM2.5")
plt.scatter(mx, my,color="blue")
plt.plot(mx, mymodel)
plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.grid()

plt.show()
```

**Figure 6.52:** Meerut $PM_{2.5}$ testing code

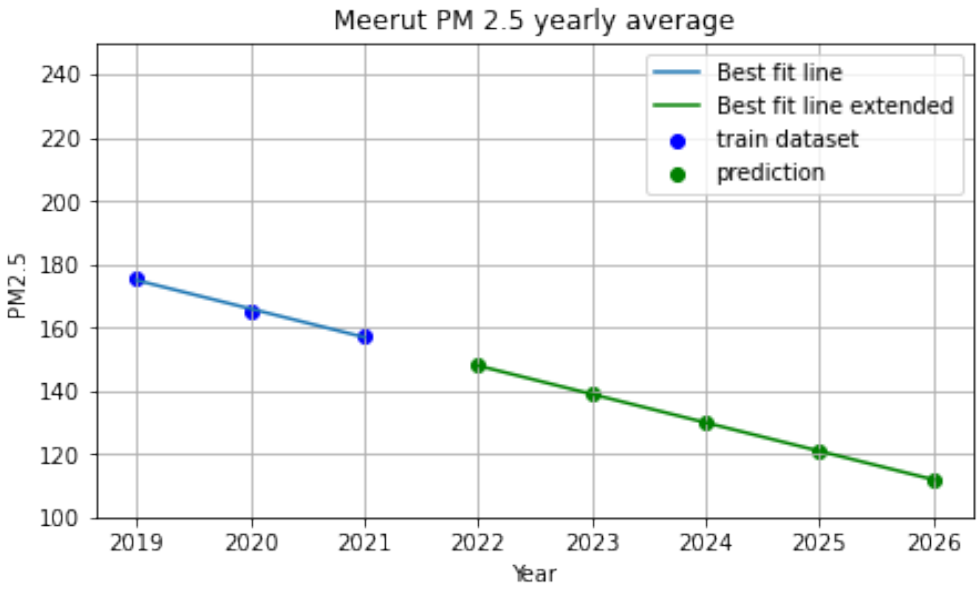


**Figure 6.53:** Meerut $PM_{2.5}$ testing output

- Meerut $NO_2$ Yearly Average Calculation

```
mrt=pd.read_csv("mrt train .csv")
mrt_t=pd.read_csv("mrt test.csv")
mrt_o3 = mrt.drop(labels=['pm10','pm25','so2','o3','co'], axis=1)
mrt_o3_t=mrt_t.drop(labels=['pm10','pm25','so2','o3','co'], axis=1)
```

```
mx = mrt_o3['Year']

my = mrt_o3['no2']

slope, intercept, r, p, std_err = stats.linregress(mx, my)

def myfunc(mx):
  return slope * mx + intercept

mymodel = list(map(myfunc, mx))
plt.xticks(mx)
plt.ylim(0,80)
plt.title("Meerut NO2 yearly average")
plt.xlabel("Year")
plt.ylabel("NO2")
plt.scatter(mx, my)
plt.plot(mx, mymodel)
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()
plt.show()
```

**Figure 6.54:** Meerut $NO_2$ training code

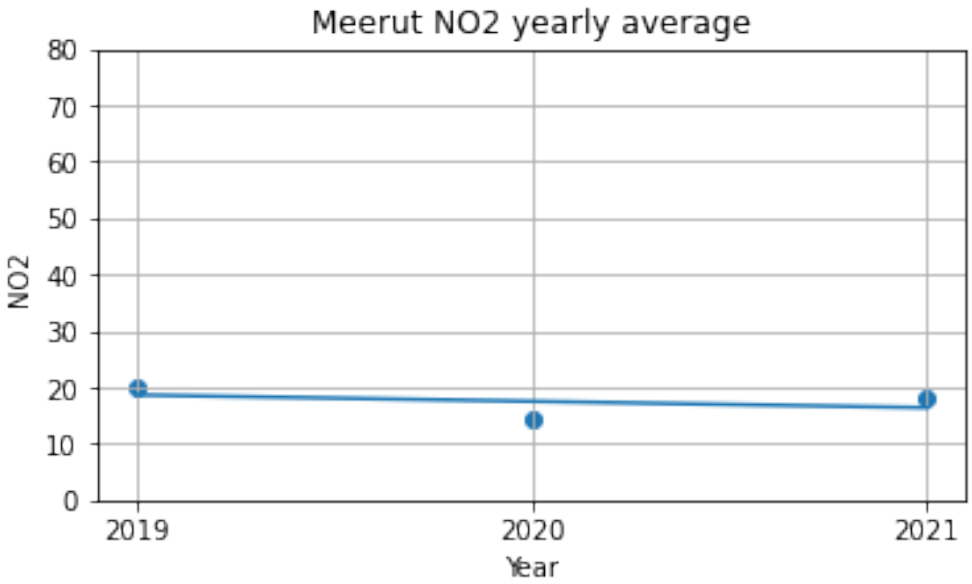


**Figure 6.55:** Meerut $NO_2$ training output

- Meerut $NO_2$ Yearly Average Calculation Testing for 2022

```
mx_t = mrt_o3_t['Year']
mx=mrt_o3['Year']
my_t = mrt_o3_t['no2']
my=mrt_o3['no2']
slope, intercept, r, p, std_err = stats.linregress(mx, my)

def myfunc(mx):
  return slope * mx + intercept

mymodel = list(map(myfunc, mx))
plt.xticks([2017,2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(0,100)
plt.title("Meerut NO2 yearly average")
plt.xlabel("Year")
plt.ylabel("NO2")
plt.scatter(mx, my,color="blue")
plt.plot(mx, mymodel)
plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()


plt.show()
```

**Figure 6.56:** Meerut $NO_2$ testing code

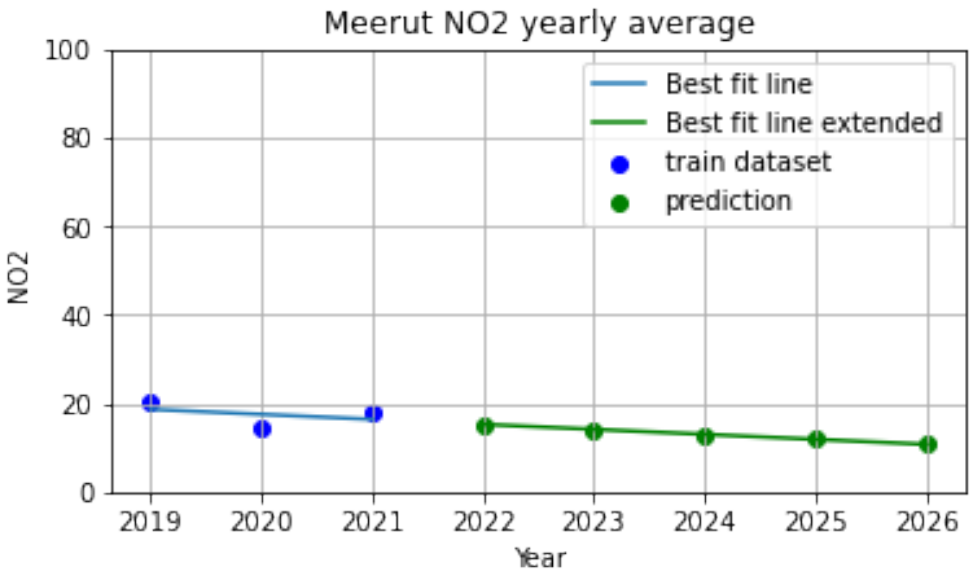


**Figure 6.57:** Meerut $NO_2$ testing output

- Meerut $SO_2$ Yearly Average Calculation

```
mrt=pd.read_csv("mrt train .csv")
mrt_o3 = mrt.drop(labels=['pm10','pm25','no2','o3','co'], axis=1)

```

```
mx = mrt_o3['Year']

my = mrt_o3['so2']

slope, intercept, r, p, std_err = stats.linregress(mx, my)

def myfunc(mx):
  return slope * mx + intercept

mymodel = list(map(myfunc, mx))
plt.xticks(mx)
plt.ylim(0,80)
plt.title("Meerut SO2 yearly average")
plt.xlabel("Year")
plt.ylabel("SO2")
plt.scatter(mx, my)
plt.plot(mx, mymodel)
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()

plt.show()
```

**Figure 6.58:** Meerut $SO_2$ training code

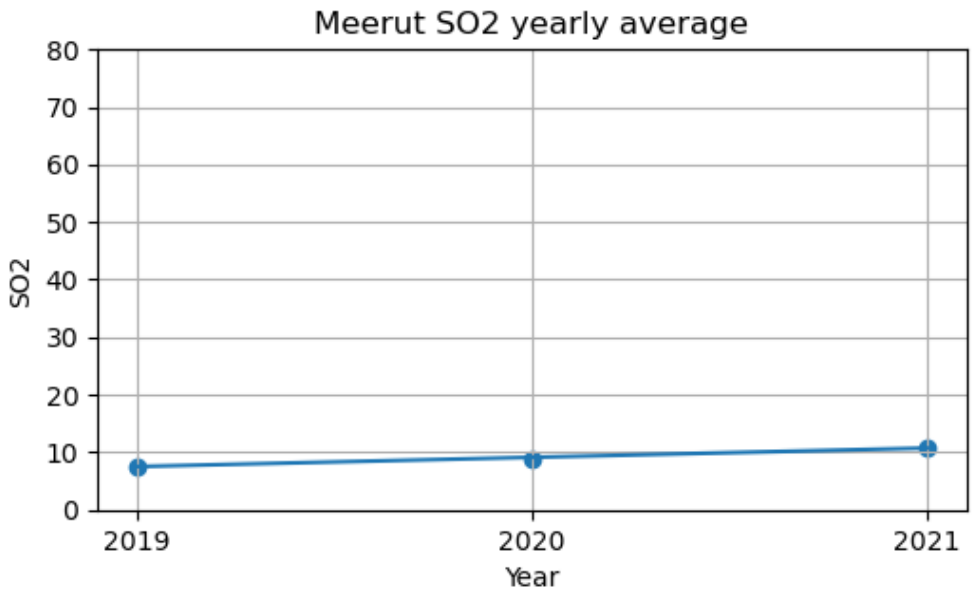


**Figure 6.59:** Meerut $SO_2$ training output

- Meerut $SO_2$ Yearly Average Calculation Testing for 2022

```
mx_t = mrt_o3_t['Year']
mx=mrt_o3['Year']
my_t = mrt_o3_t['so2']
my=mrt_o3['so2']
slope, intercept, r, p, std_err = stats.linregress(mx, my)

def myfunc(mx):
  return slope * mx + intercept

mymodel = list(map(myfunc, mx))
plt.xticks([2017,2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(0,100)
plt.title("Meerut SO2 yearly average")
plt.xlabel("Year")
plt.ylabel("SO2")
plt.scatter(mx, my,color="blue")
plt.plot(mx, mymodel)
plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()



plt.show()
```

**Figure 6.60:** Meerut $SO_2$ testing code

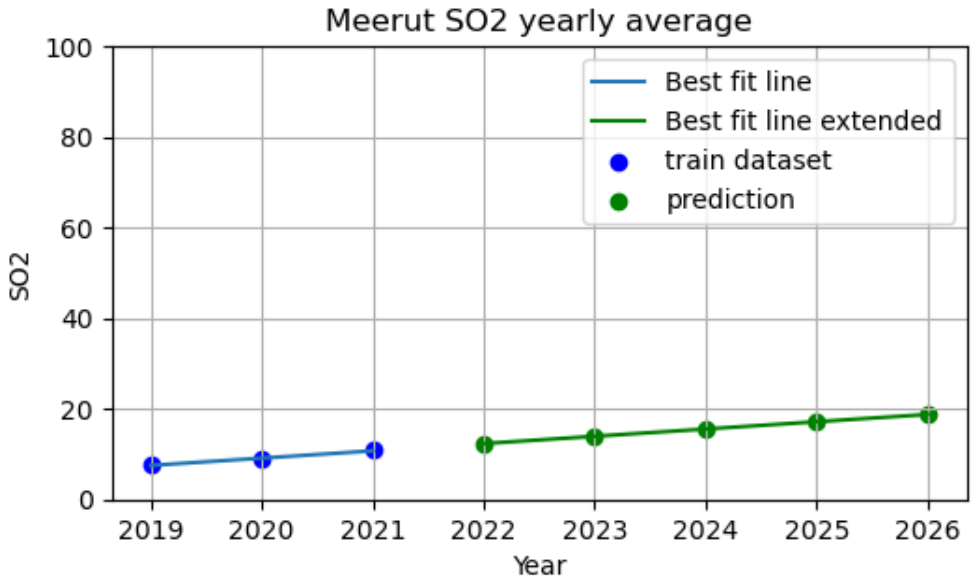


**Figure 6.61:** Meerut $SO_2$ testing output

- Meerut $O_3$ Yearly Average Calculation

```
mrt=pd.read_csv("mrt train .csv")
mrt_t=pd.read_csv("mrt test.csv")
mrt_o3 = mrt.drop(labels=['pm10','pm25','no2','so2','co'], axis=1)
mrt_o3_t=mrt_t.drop(labels=['pm10','pm25','no2','so2','co'], axis=1)
```

```
mx = mrt_o3['Year']

my = mrt_o3['o3']

slope, intercept, r, p, std_err = stats.linregress(mx, my)

def myfunc(mx):
  return slope * mx + intercept

mymodel = list(map(myfunc, mx))
plt.xticks(mx)
plt.ylim(10,80)
plt.title("Meerut O3 yearly average")
plt.xlabel("Year")
plt.ylabel("O3")
plt.scatter(mx, my)
plt.plot(mx, mymodel)


plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()

plt.show()
```

**Figure 6.62:** Meerut $O_3$ training code

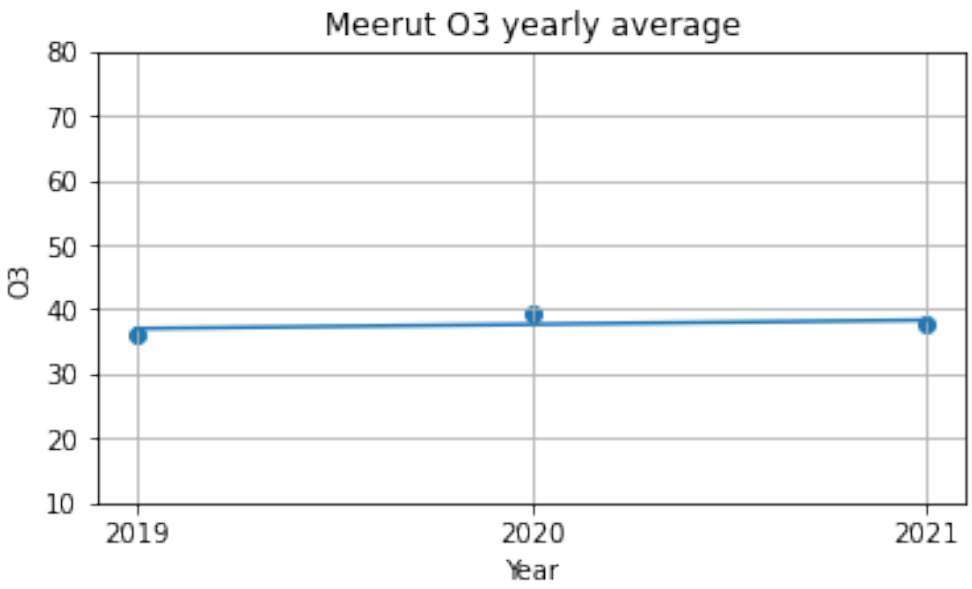


**Figure 6.63:** Meerut $O_3$ training output

- Meerut $O_3$ Yearly Average Calculation Testing for 2022

```
mx_t = mrt_o3_t['Year']
mx=mrt_o3['Year']
my_t = mrt_o3_t['o3']
my=mrt_o3['o3']
slope, intercept, r, p, std_err = stats.linregress(mx, my)

def myfunc(mx):
  return slope * mx + intercept

mymodel = list(map(myfunc, mx))
plt.xticks([2017,2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(10,100)
plt.title("Meerut O3 yearly average")
plt.xlabel("Year")
plt.ylabel("O3")
plt.scatter(mx, my,color="blue")
plt.plot(mx, mymodel)
plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()


plt.show()
```

**Figure 6.64:** Meerut $O_3$ testing code

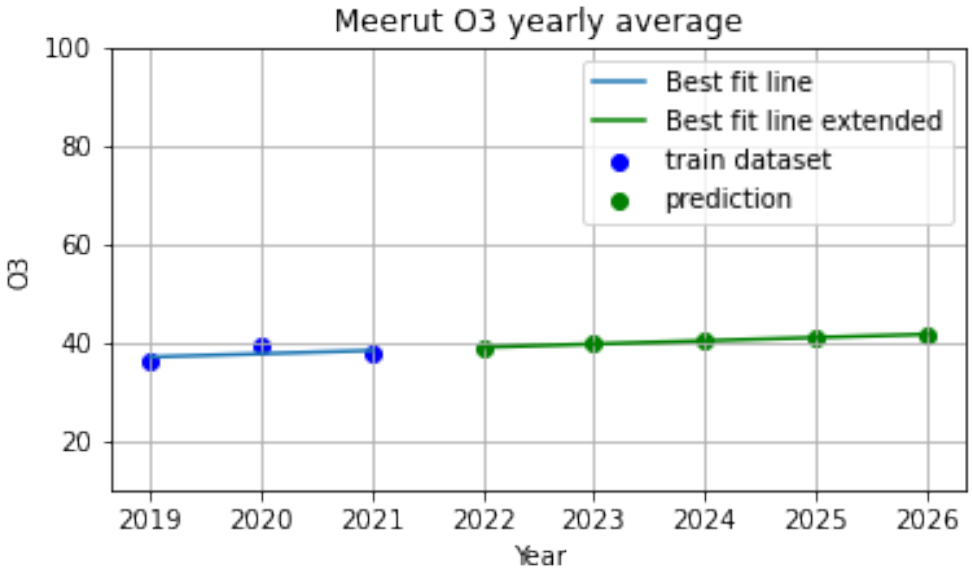


**Figure 6.65:** Meerut $O_3$ testing output

- Meerut CO Yearly Average Calculation

```
mrt=pd.read_csv("mrt train .csv")
mrt_t=pd.read_csv("mrt test.csv")
mrt_o3 = mrt.drop(labels=['pm10','pm25','so2','o3','no2'], axis=1)
mrt_o3_t=mrt_t.drop(labels=['pm10','pm25','so2','o3','no2'], axis=1)
```

```
mx = mrt_o3['Year']

my = mrt_o3['co']

slope, intercept, r, p, std_err = stats.linregress(mx, my)

def myfunc(mx):
  return slope * mx + intercept

mymodel = list(map(myfunc, mx))
plt.xticks(mx)
plt.ylim(0,80)
plt.title("Meerut CO yearly average")
plt.xlabel("Year")
plt.ylabel("CO")
plt.scatter(mx, my)
plt.plot(mx, mymodel)

plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()
plt.show()
```

**Figure 6.66:** Meerut CO training code

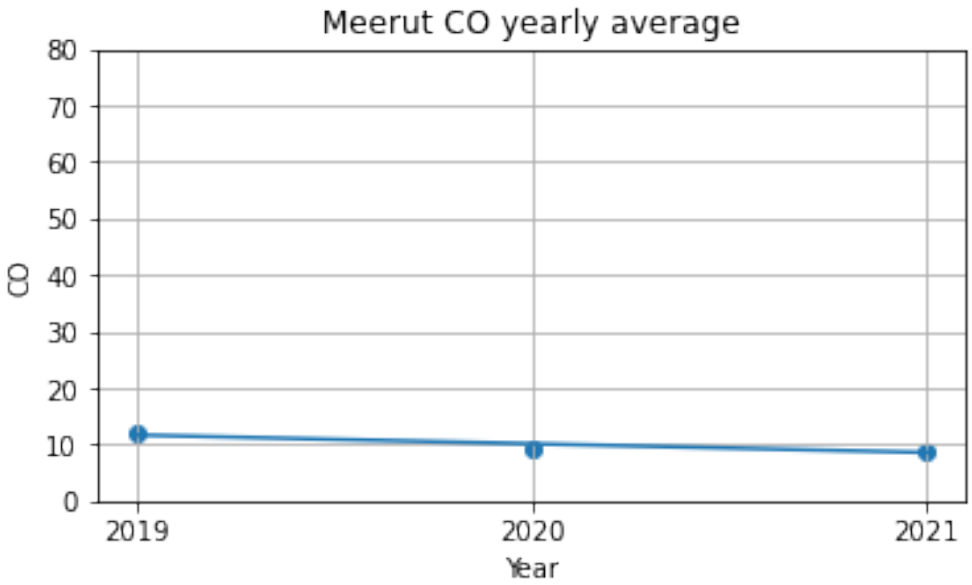


**Figure 6.67:** Meerut CO training output

- Meerut CO Yearly Average Calculation Testing for 2022

```
mx_t = mrt_o3_t['Year']
mx=mrt_o3['Year']
my_t = mrt_o3_t['co']
my=mrt_o3['co']
slope, intercept, r, p, std_err = stats.linregress(mx, my)

def myfunc(mx):
  return slope * mx + intercept

mymodel = list(map(myfunc, mx))
plt.xticks([2017,2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(0,100)
plt.title("Meerut CO yearly average")
plt.xlabel("Year")
plt.ylabel("CO")
plt.scatter(mx, my,color="blue")
plt.plot(mx, mymodel)
plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()


plt.show()
```

**Figure 6.68:** Meerut CO testing code

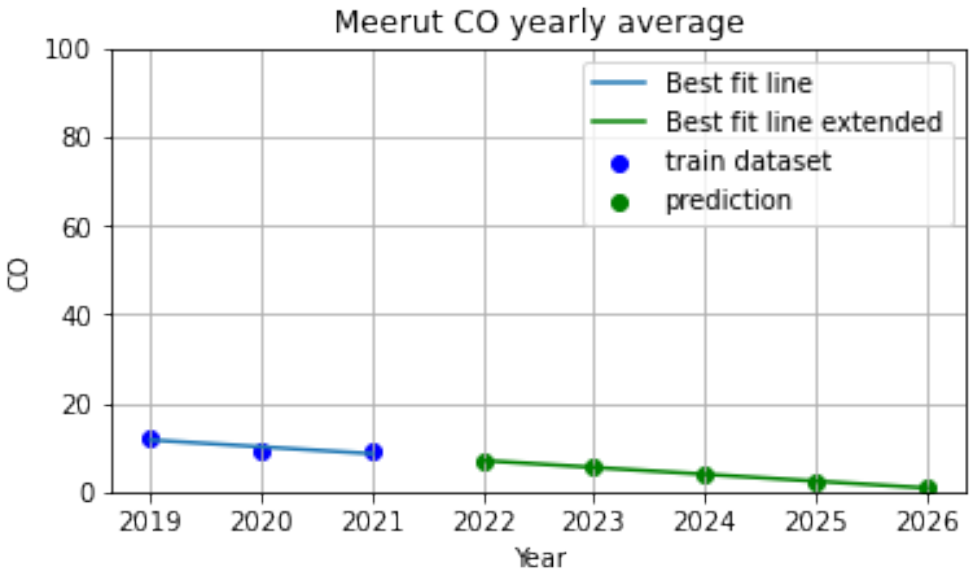


**Figure 6.69:** Meerut CO testing output

- Meerut $PM_{10}$ Yearly Average Calculation

```
mrt=pd.read_csv("mrt train .csv")
mrt_o3 = mrt.drop(labels=['no2','pm25','so2','o3','co'], axis=1)

```

```
mx = mrt_o3['Year']

my = mrt_o3['pm10']

slope, intercept, r, p, std_err = stats.linregress(mx, my)

def myfunc(mx):
  return slope * mx + intercept

mymodel = list(map(myfunc, mx))
plt.xticks(mx)
plt.ylim(100,200)
plt.title("Meerut PM10 yearly average")
plt.xlabel("Year")
plt.ylabel("PM10")
plt.scatter(mx, my)
plt.plot(mx, mymodel)
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()
plt.show()
```

**Figure 6.70:** Meerut $PM_{10}$ training code

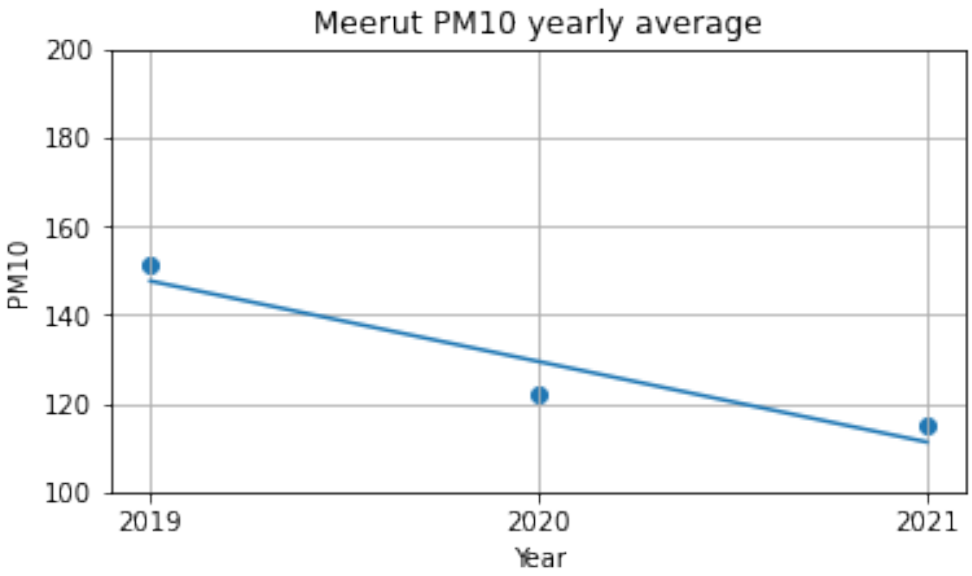


**Figure 6.71:** Meerut $PM_{10}$ training output

- Meerut $PM_{10}$ Yearly Average Calculation Testing for 2022

```
mx=mrt_o3['Year']
my=mrt_o3['pm10']
slope, intercept, r, p, std_err = stats.linregress(mx, my)

def myfunc(mx):
  return slope * mx + intercept

mymodel = list(map(myfunc, mx))
plt.xticks([2017,2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(0,200)
plt.title("Meerut PM10 yearly average")
plt.xlabel("Year")
plt.ylabel("PM10")
plt.scatter(mx, my,color="blue")
plt.plot(mx, mymodel)
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()
plt.show()
```

**Figure 6.72:** Meerut $PM_{10}$ testing code

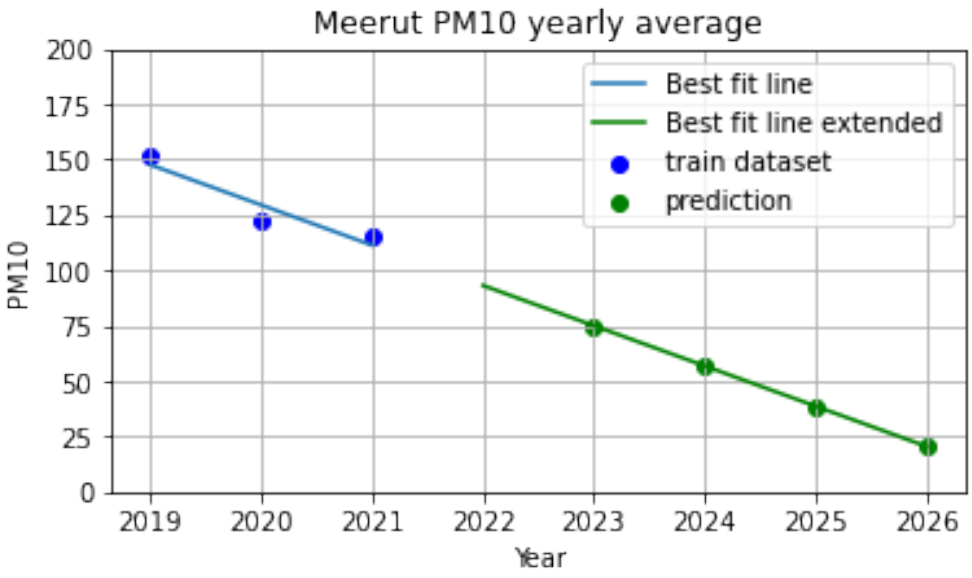


**Figure 6.73:** Meerut $PM_{10}$ testing output

- Delhi $PM_{2.5}$ Yearly Average Calculation

```
delhi=pd.read_csv("delhi_train.csv")
```

```
delhi_pm25 = delhi.drop(labels=['pm10','o3','no2','so2','co'], axis=1)
```

```
x = delhi_pm25['Year']

y = delhi_pm25['pm25']

slope, intercept, r, p, std_err = stats.linregress(x, y)

def myfunc(x):
  return slope * x + intercept

mymodel = list(map(myfunc, x))
plt.xticks(x)
plt.ylim(180,250)
plt.title("Delhi_PM2.5 yearly average")
plt.xlabel("Year")
plt.ylabel("PM2.5")
plt.scatter(x, y)
plt.plot(x, mymodel)
plt.grid()
plt.show()
```

**Figure 6.74:** Delhi $PM_{2.5}$ training code

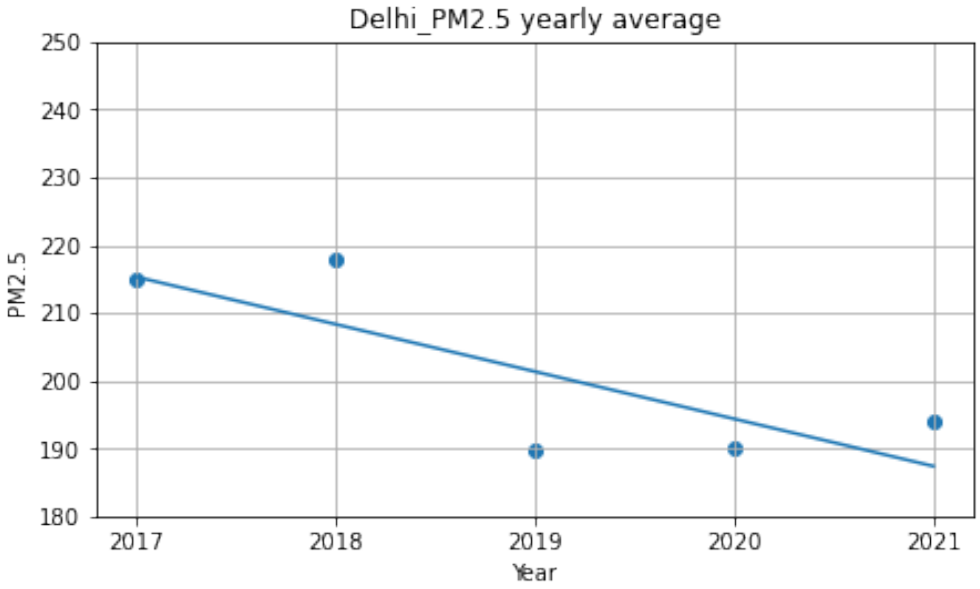


**Figure 6.75:** Delhi $PM_{2.5}$ training output

- Delhi $PM_{2.5}$ Yearly Average Calculation Testing for 2022 - 2026

```
x=delhi_pm25['Year']
y=delhi_pm25['pm25']
slope, intercept, r, p, std_err = stats.linregress(x, y)

def myfunc(x):
  return slope * x + intercept

mymodel = list(map(myfunc, x))
plt.xticks([2017,2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(100,250)
plt.title("Delhi PM 2.5 yearly average")
plt.xlabel("Year")
plt.ylabel("PM2.5")
plt.scatter(x, y,color="blue")
plt.plot(x, mymodel)

plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [20/2.54, 10/2.54]
plt.grid()

plt.show()
```

**Figure 6.76:** Delhi $PM_{2.5}$ testing code

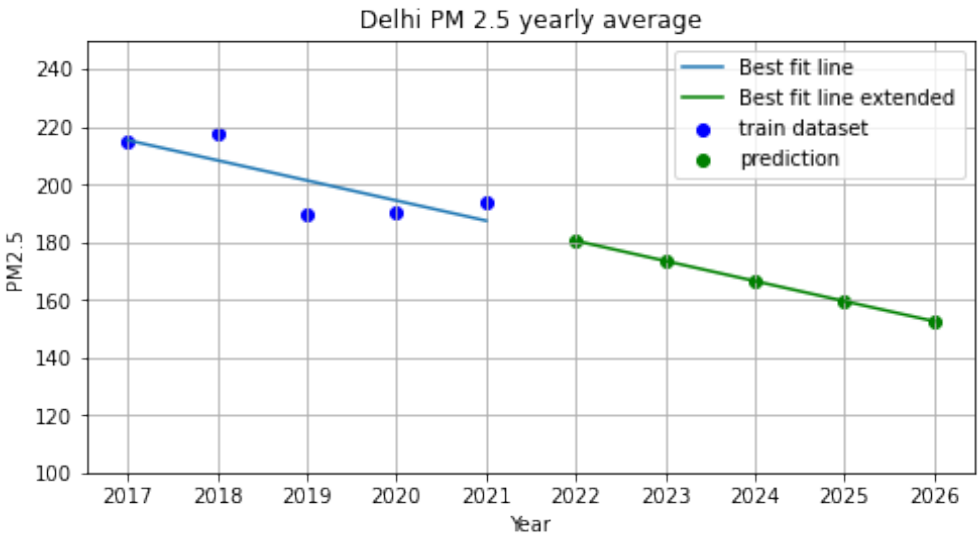


**Figure 6.77:** Delhi $PM_{2.5}$ testing output

- Delhi $NO_2$ Yearly Average Calculation

```
delhi=pd.read_csv("delhi_train.csv")
delhi_t=delhi.drop(labels=['pm25','pm10','so2','o3','co'], axis=1)

dx = delhi_t['Year']
dy = delhi_t['no2']
slpe, intrcpt, r1, p1, stderr= stats.linregress(dx, dy)
def myfunc(dx):
  return slpe * dx + intrcpt

mymodel = list(map(myfunc, dx))
plt.xticks(dx)
plt.ylim(0,50)
plt.title("Delhi NO2 yearly average")
plt.xlabel("Year")
plt.ylabel("NO2")
plt.scatter(dx, dy)
plt.plot(dx, mymodel)
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()
plt.show()
```

**Figure 6.78:** Delhi $NO_2$ training code

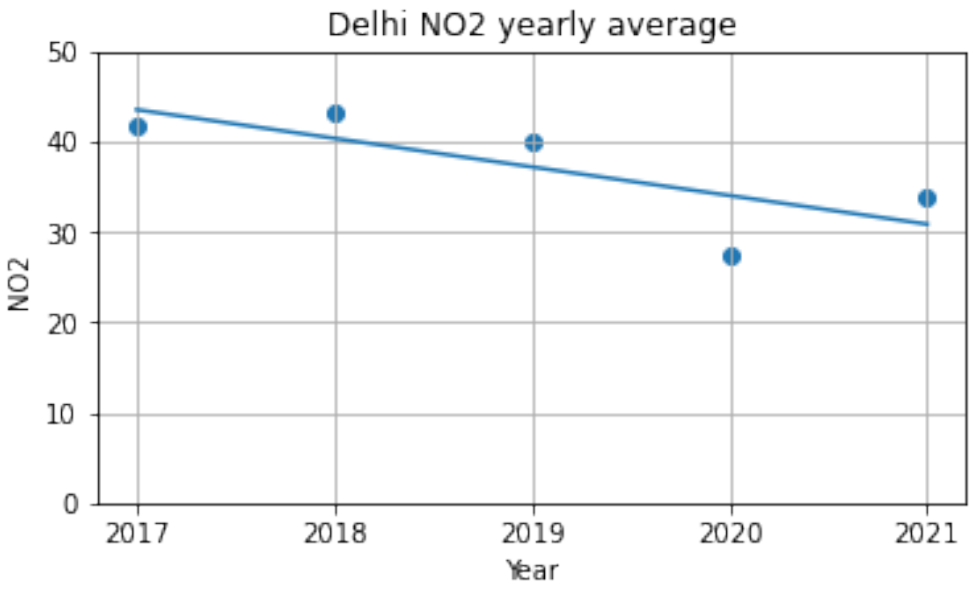


**Figure 6.79:** Delhi $NO_2$ training output

- Delhi $NO_2$ Yearly Average Calculation Testing for 2022 - 2026

```
Delhi_test = pd.read_csv("delhi_test.csv")
Delhi_test=Delhi_test.drop(labels=['pm10','pm25','so2','o3','co'], axis=1)
dx_t = Delhi_test['Year']
dy_t = Delhi_test['no2']
slpe, intrcpt, r1, p1, stderr= stats.linregress(dx, dy)
def myfunc(dx):
  return slpe * dx + intrcpt

mymodel = list(map(myfunc, dx))
plt.xticks([2017,2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(0,80)
plt.title("Delhi NO2 yearly average")
plt.xlabel("Year")
plt.ylabel("NO2")
plt.scatter(dx, dy,color="blue")
plt.plot(dx, mymodel)

plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()

plt.show()
```

**Figure 6.80:** Delhi $NO_2$ testing code

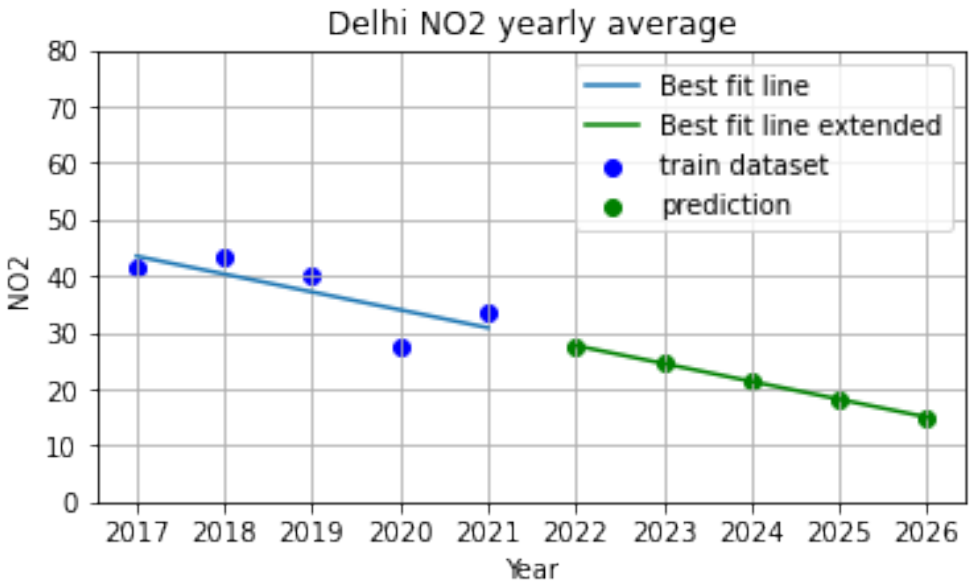


**Figure 6.81:** Delhi $NO_2$ testing output

- Delhi $SO_2$ Yearly Average Calculation

```
delhi=pd.read_csv("delhi_train.csv")
delhi_t=delhi.drop(labels=['pm25','pm10','no2','o3','co'], axis=1)

dx = delhi_t['Year']
dy = delhi_t['so2']
slpe, intrcpt, r1, p1, stderr= stats.linregress(dx, dy)
def myfunc(dx):
  return slpe * dx + intrcpt

mymodel = list(map(myfunc, dx))
plt.xticks(dx)
plt.ylim(0,50)
plt.title("Delhi SO2 yearly average")
plt.xlabel("Year")
plt.ylabel("SO2")
plt.scatter(dx, dy)
plt.plot(dx, mymodel)

plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()

plt.show()
```

**Figure 6.82:** Delhi $SO_2$ training code

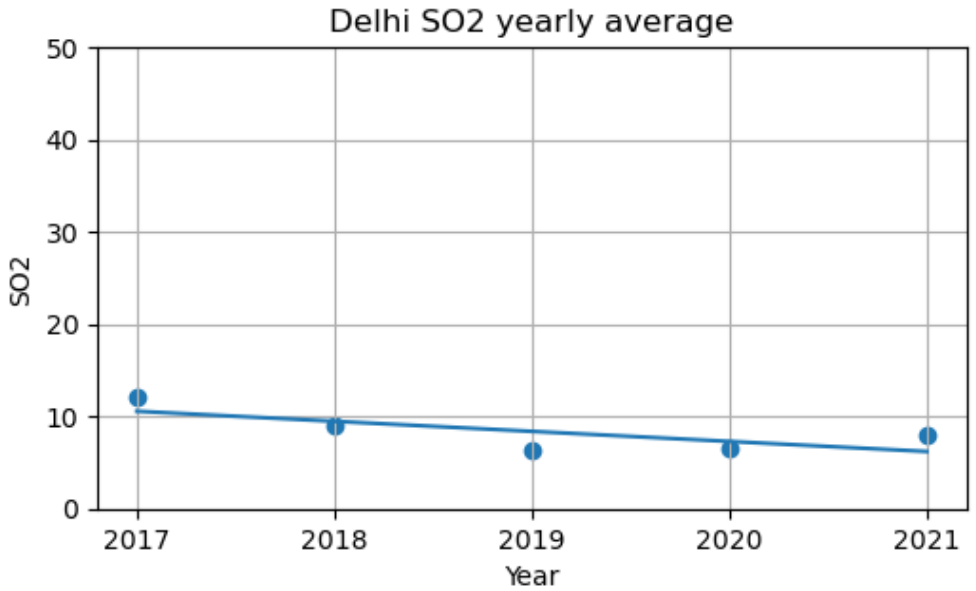


**Figure 6.83:** Delhi $SO_2$ training output

- Delhi $SO_2$ Yearly Average Calculation Testing for 2022 - 2026

```
Delhi_test = pd.read_csv("delhi_test.csv")
Delhi_test=Delhi_test.drop(labels=['pm10','pm25','no2','o3','co'], axis=1)
dx_t = Delhi_test['Year']
dy_t = Delhi_test['so2']
slpe, intrcpt, r1, p1, stderr= stats.linregress(dx, dy)
def myfunc(dx):
  return slpe * dx + intrcpt

mymodel = list(map(myfunc, dx))
plt.xticks([2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(0,50)
plt.title("Delhi SO2 yearly average")
plt.xlabel("Year")
plt.ylabel("SO2")
plt.scatter(dx, dy,color="blue")
plt.plot(dx, mymodel)
plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()
plt.show()
```

**Figure 6.84:** Delhi $SO_2$ testing code

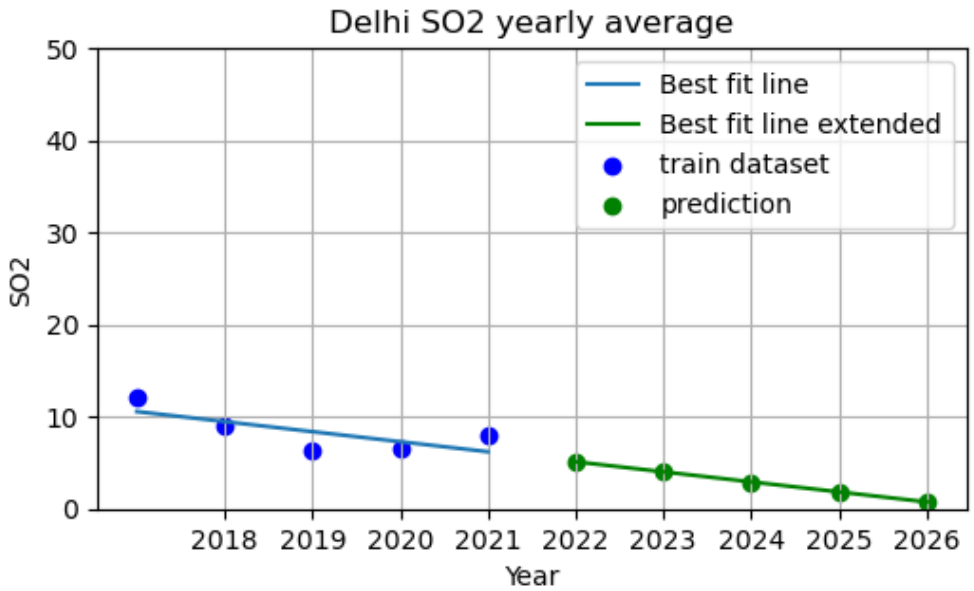


**Figure 6.85:** Delhi $SO_2$ testing output

- Delhi $O_3$ Yearly Average Calculation

```
delhi=pd.read_csv("delhi_train.csv")
delhi_t=delhi.drop(labels=['pm25','pm10','no2','so2','co'], axis=1)

dx = delhi_t['Year']
dy = delhi_t['o3']
slpe, intrcpt, r1, p1, stderr= stats.linregress(dx, dy)
def myfunc(dx):
  return slpe * dx + intrcpt

mymodel = list(map(myfunc, dx))
plt.xticks(dx)
plt.ylim(10,50)
plt.title("Delhi O3 yearly average")
plt.xlabel("Year")
plt.ylabel("O3")
plt.scatter(dx, dy)
plt.plot(dx, mymodel)


plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()

plt.show()

```

**Figure 6.86:** Delhi $O_3$ training code

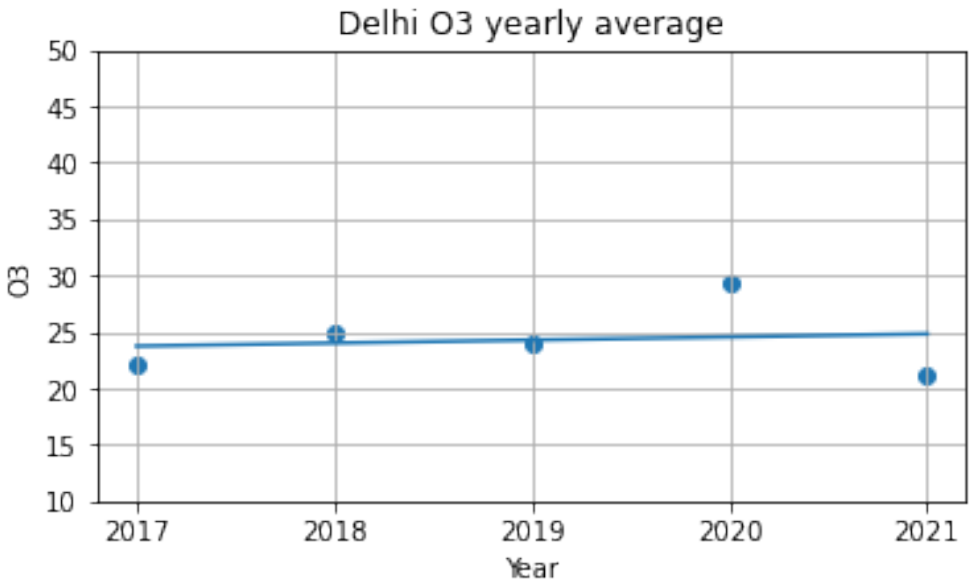


**Figure 6.87:** Delhi $O_3$ training output

- Delhi $O_3$ Yearly Average Calculation Testing for 2022 - 2026

```
Delhi_test = pd.read_csv("delhi_test.csv")
Delhi_test=Delhi_test.drop(labels=['pm10','pm25','no2','so2','co'], axis=1)
dx_t = Delhi_test['Year']
dy_t = Delhi_test['o3']
slpe, intrcpt, r1, p1, stderr= stats.linregress(dx, dy)
def myfunc(dx):
  return slpe * dx + intrcpt

mymodel = list(map(myfunc, dx))
plt.xticks([2017,2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(10,80)
plt.title("Delhi O3 yearly average")
plt.xlabel("Year")
plt.ylabel("O3")
plt.scatter(dx, dy,color="blue")
plt.plot(dx, mymodel)

plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()


plt.show()
```

**Figure 6.88:** Delhi $O_3$ testing code

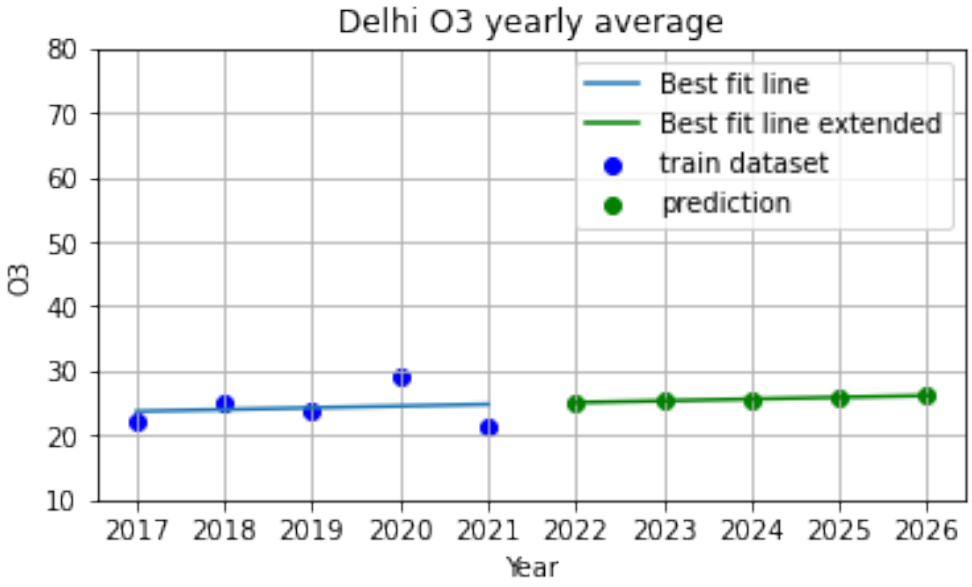


**Figure 6.89:** Delhi $O_3$ testing output

- Delhi CO Yearly Average Calculation

```
delhi=pd.read_csv("delhi_train.csv")
delhi_t=delhi.drop(labels=['pm25','pm10','so2','o3','no2'], axis=1)

dx = delhi_t['Year']
dy = delhi_t['co']
slpe, intrcpt, r1, p1, stderr= stats.linregress(dx, dy)
def myfunc(dx):
  return slpe * dx + intrcpt

mymodel = list(map(myfunc, dx))
plt.xticks(dx)
plt.ylim(0,50)
plt.title("Delhi CO yearly average")
plt.xlabel("Year")
plt.ylabel("CO")
plt.scatter(dx, dy)
plt.plot(dx, mymodel)

plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()
plt.show()
```

**Figure 6.90:** Delhi CO training code

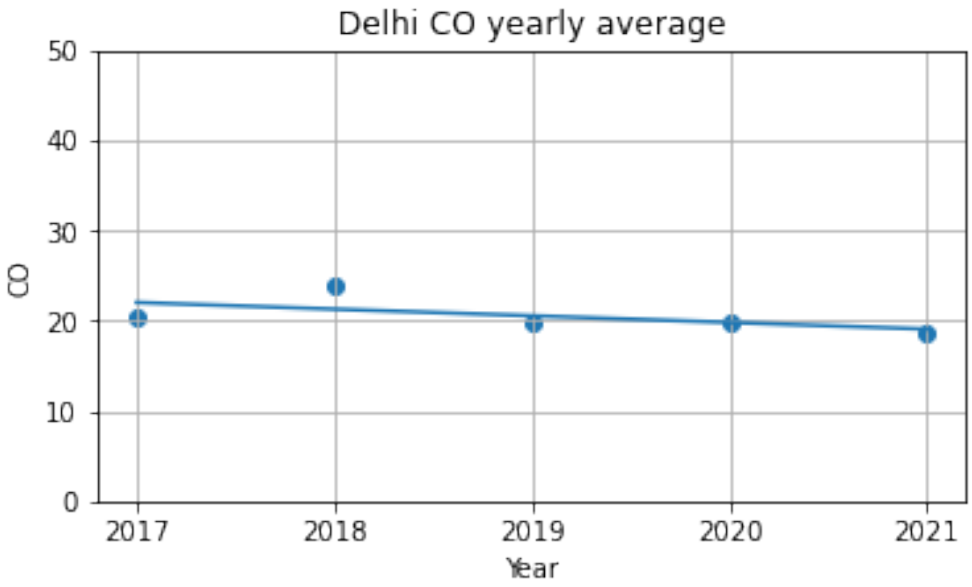


**Figure 6.91:** Delhi CO training output

- Delhi CO Yearly Average Calculation Testing for 2022 - 2026

```
Delhi_test = pd.read_csv("delhi_test.csv")
Delhi_test=Delhi_test.drop(labels=['pm10','pm25','so2','o3','no2'], axis=1)
dx_t = Delhi_test['Year']
dy_t = Delhi_test['co']
slpe, intrcpt, r1, p1, stderr= stats.linregress(dx, dy)
def myfunc(dx):
  return slpe * dx + intrcpt

mymodel = list(map(myfunc, dx))
plt.xticks([2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(0,80)
plt.title("Delhi CO yearly average")
plt.xlabel("Year")
plt.ylabel("CO")
plt.scatter(dx, dy,color="blue")
plt.plot(dx, mymodel)

plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()

plt.show()
```

**Figure 6.92:** Delhi CO testing code

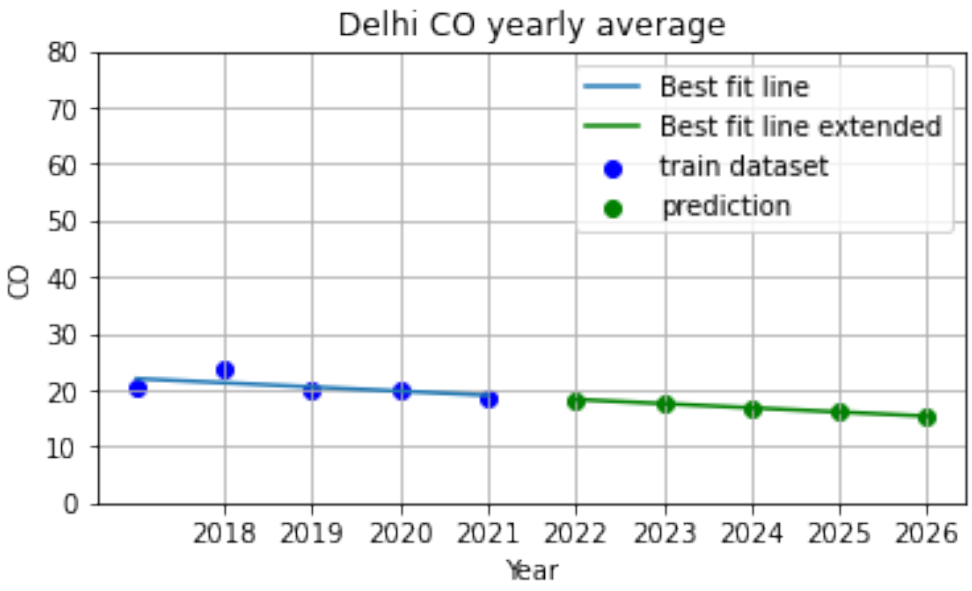


**Figure 6.93:** Delhi CO testing output

- Delhi $PM_{10}$ Yearly Average Calculation

```
delhi=pd.read_csv("delhi_train.csv")
delhi_t=delhi.drop(labels=['pm25','no2','so2','o3','co'], axis=1)

dx = delhi_t['Year']
dy = delhi_t['pm10']
slpe, intrcpt, r1, p1, stderr= stats.linregress(dx, dy)
def myfunc(dx):
  return slpe * dx + intrcpt

mymodel = list(map(myfunc, dx))
plt.xticks(dx)
plt.ylim(100,400)
plt.title("Delhi PM10 yearly average")
plt.xlabel("Year")
plt.ylabel("PM10")
plt.grid()
plt.scatter(dx, dy)
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.plot(dx, mymodel)
plt.show()
```

**Figure 6.94:** Delhi $PM_{10}$ training code

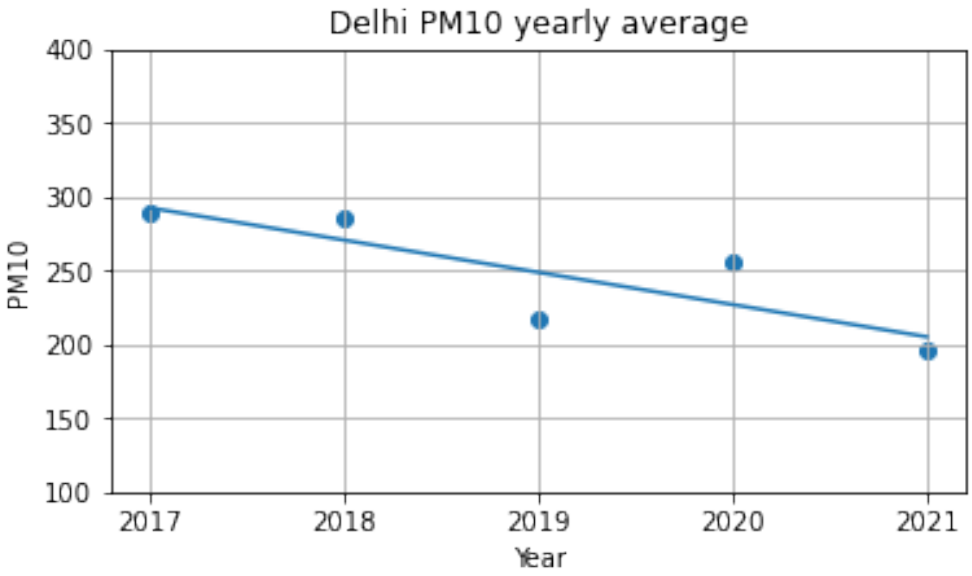


**Figure 6.95:** Delhi $PM_{10}$ training output

- Delhi $PM_{10}$ Yearly Average Calculation Testing for 2022 - 2026

```
Delhi_test = pd.read_csv("delhi_test.csv")
Delhi_test=Delhi_test.drop(labels=['no2','pm25','so2','o3','co'], axis=1)
dx_t = Delhi_test['Year']
dy_t = Delhi_test['pm10']
slpe, intrcpt, r1, p1, stderr= stats.linregress(dx, dy)
def myfunc(dx):
  return slpe * dx + intrcpt

mymodel = list(map(myfunc, dx))
plt.xticks([2017,2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(50,300)
plt.title("Delhi PM10 yearly average")
plt.xlabel("Year")
plt.ylabel("PM10")
plt.scatter(dx, dy,color="blue")
plt.plot(dx, mymodel)
plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()
plt.show()
```

**Figure 6.96:** Delhi $PM_{10}$ testing code

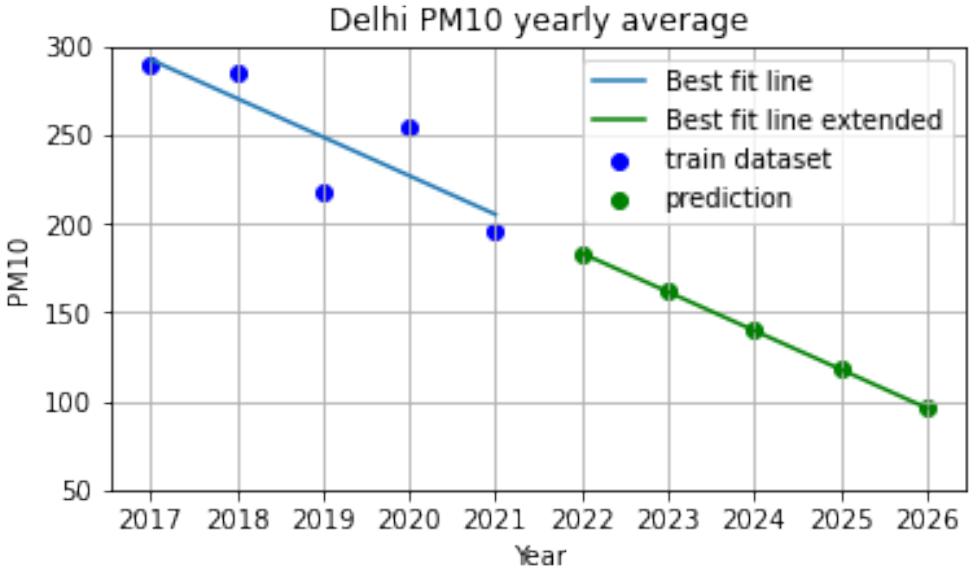


**Figure 6.97:** Delhi $PM_{10}$ testing output

- Noida $PM_{2.5}$ Yearly Average Calculation

```
nd=pd.read_csv("noida train.csv")
```

```
nd_pm25 = nd.drop(labels=['pm10','o3','no2','so2','co'], axis=1)
```

```
nx = nd_pm25['Year']

ny = nd_pm25['pm25']

slope, intercept, r, p, std_err = stats.linregress(nx, ny)

def myfunc(nx):
  return slope * nx + intercept

# next_year_value=myfunc

mymodel = list(map(myfunc, nx))
plt.xticks(nx)
plt.ylim(100,300)
plt.title("Noida_PM2.5 yearly average")
plt.xlabel("Year")
plt.ylabel("PM2.5")
plt.grid()
plt.scatter(nx, ny)
plt.plot(nx, mymodel)
plt.show()
```

**Figure 6.98:** Noida $PM_{2.5}$ training code

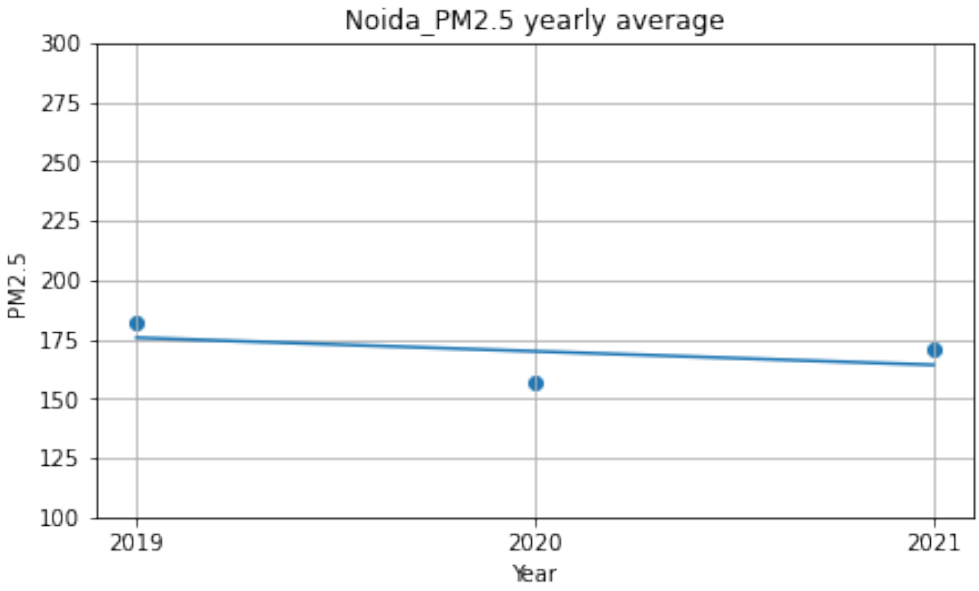


**Figure 6.99:** Noida $PM_{2.5}$ training output

- Noida $PM_{2.5}$ Yearly Average Calculation Testing for 2022 - 2026

```
nx = nd_pm25['Year']
ny = nd_pm25['pm25']
slope, intercept, r, p, std_err = stats.linregress(nx, ny)

def myfunc(nx):
  return slope * nx + intercept

mymodel = list(map(myfunc, nx))
plt.xticks([2017,2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(90,300)
plt.title("Noida PM 2.5 yearly average")
plt.xlabel("Year")
plt.ylabel("PM2.5")
plt.scatter(nx, ny,color="blue")
plt.plot(nx, mymodel)
plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.grid()

plt.show()
```

**Figure 6.100:** Noida $PM_{2.5}$ testing code

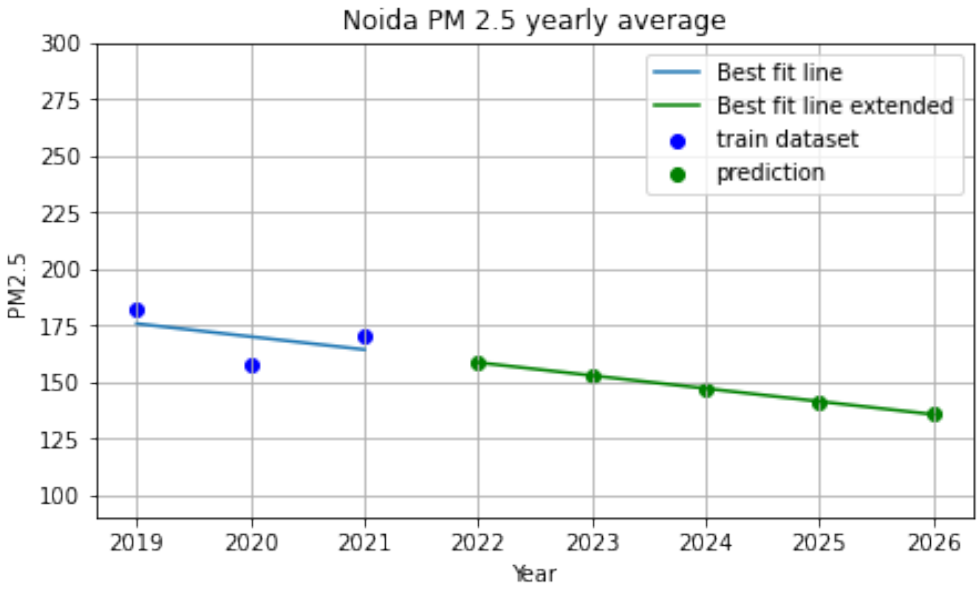


**Figure 6.101:** Noida $PM_{2.5}$ testing output

- Noida $NO_2$ Yearly Average Calculation

```
nd=pd.read_csv("noida train.csv")
nd_t=pd.read_csv("noida test.csv")
nd_o3 = nd.drop(labels=['pm10','pm25','so2','o3','co'], axis=1)
nd_o3_t=nd_t.drop(labels=['pm10','pm25','so2','o3','co'], axis=1)
```

```
nx = nd_o3['Year']

ny = nd_o3['no2']

slope, intercept, r, p, std_err = stats.linregress(nx, ny)

def myfunc(nx):
  return slope * nx + intercept

mymodel = list(map(myfunc, nx))
plt.xticks(nx)
plt.ylim(0,80)
plt.title("Noida NO2 yearly average")
plt.xlabel("Year")
plt.ylabel("NO2")
plt.scatter(nx, ny)
plt.plot(nx, mymodel)
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()
plt.show()
```

**Figure 6.102:** Noida $NO_2$ training code

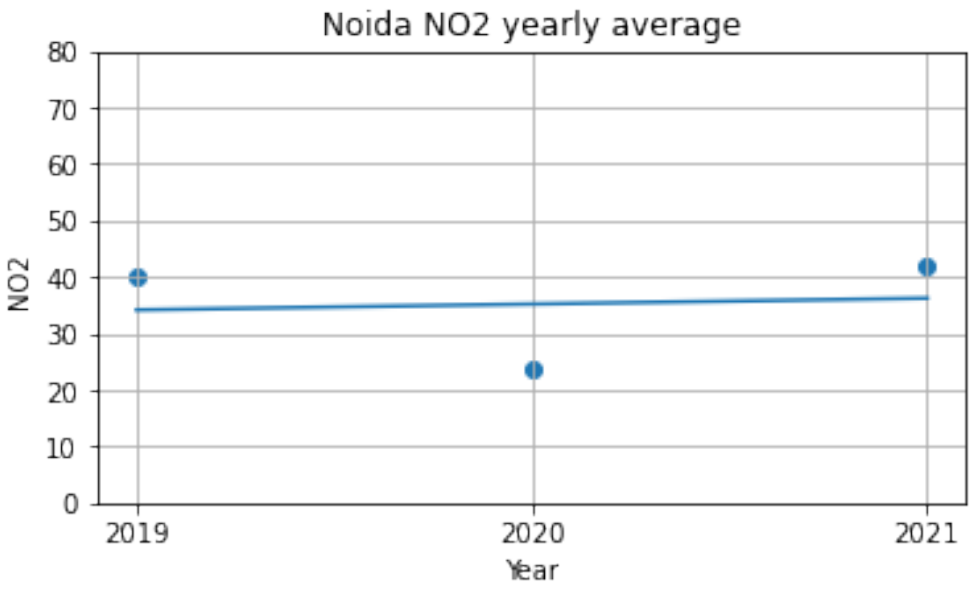


**Figure 6.103:** Noida $NO_2$ training output

- Noida $NO_2$ Yearly Average Calculation Testing for 2022 - 2026

```
nx_t = nd_o3_t['Year']
nx = nd_o3['Year']
ny_t = nd_o3_t['no2']
ny = nd_o3['no2']
slope, intercept, r, p, std_err = stats.linregress(nx, ny)

def myfunc(nx):
  return slope * nx + intercept

mymodel = list(map(myfunc, nx))
plt.xticks([2017,2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(0,80)
plt.title("Noida NO2 yearly average")
plt.xlabel("Year")
plt.ylabel("NO2")
plt.scatter(nx, ny,color="blue")
plt.plot(nx, mymodel)
plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()


plt.show()
```

**Figure 6.104:** Noida $NO_2$ testing code

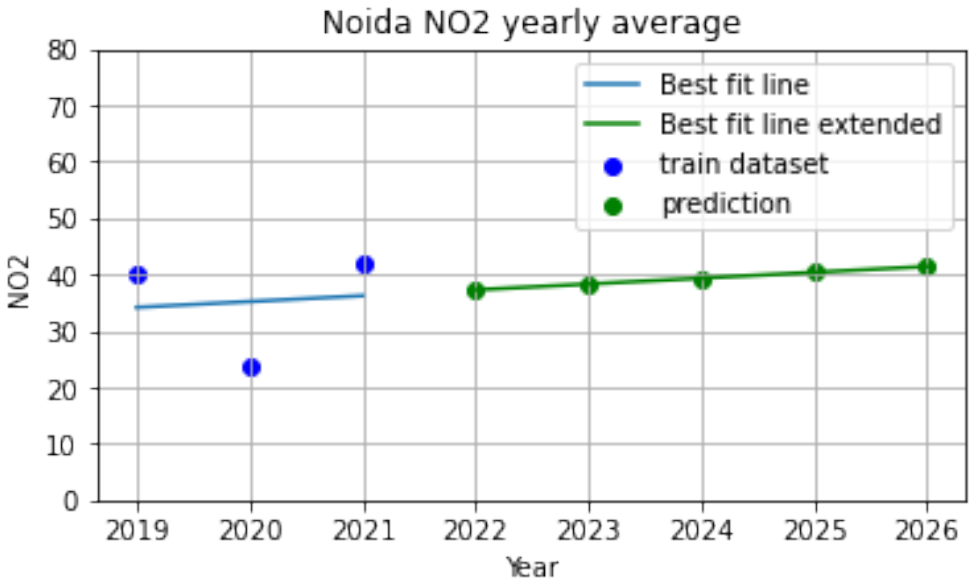


**Figure 6.105:** Noida $NO_2$ testing output

- Noida $SO_2$ Yearly Average Calculation

```
nd=pd.read_csv("noida train.csv")
nd_t=pd.read_csv("noida test.csv")
nd_o3 = nd.drop(labels=['pm10','pm25','no2','o3','co'], axis=1)
nd_o3_t=nd_t.drop(labels=['pm10','pm25','no2','o3','co'], axis=1)
```

```
nx = nd_o3['Year']

ny = nd_o3['so2']

slope, intercept, r, p, std_err = stats.linregress(nx, ny)

def myfunc(nx):
  return slope * nx + intercept

mymodel = list(map(myfunc, nx))
plt.xticks(nx)
plt.ylim(0,80)
plt.title("Noida SO2 yearly average")
plt.xlabel("Year")
plt.ylabel("SO2")
plt.scatter(nx, ny)
plt.plot(nx, mymodel)
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()

plt.show()
```

**Figure 6.106:** Noida $SO_2$ training code

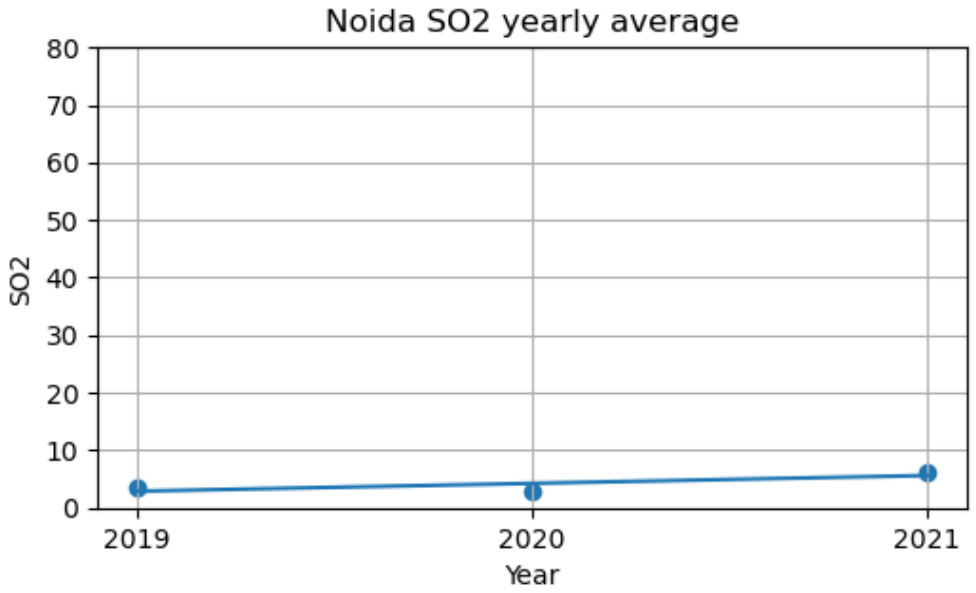


**Figure 6.107:** Noida $SO_2$ training output

- Noida $SO_2$ Yearly Average Calculation Testing for 2022 - 2026

```
nx_t = nd_o3_t['Year']
nx = nd_o3['Year']
ny_t = nd_o3_t['so2']
ny = nd_o3['so2']
slope, intercept, r, p, std_err = stats.linregress(nx, ny)

def myfunc(nx):
  return slope * nx + intercept

mymodel = list(map(myfunc, nx))
plt.xticks([2017,2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(0,80)
plt.title("Noida SO2 yearly average")
plt.xlabel("Year")
plt.ylabel("SO2")
plt.scatter(nx, ny,color="blue")
plt.plot(nx, mymodel)
plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()



plt.show()
```

**Figure 6.108:** Noida $SO_2$ testing code

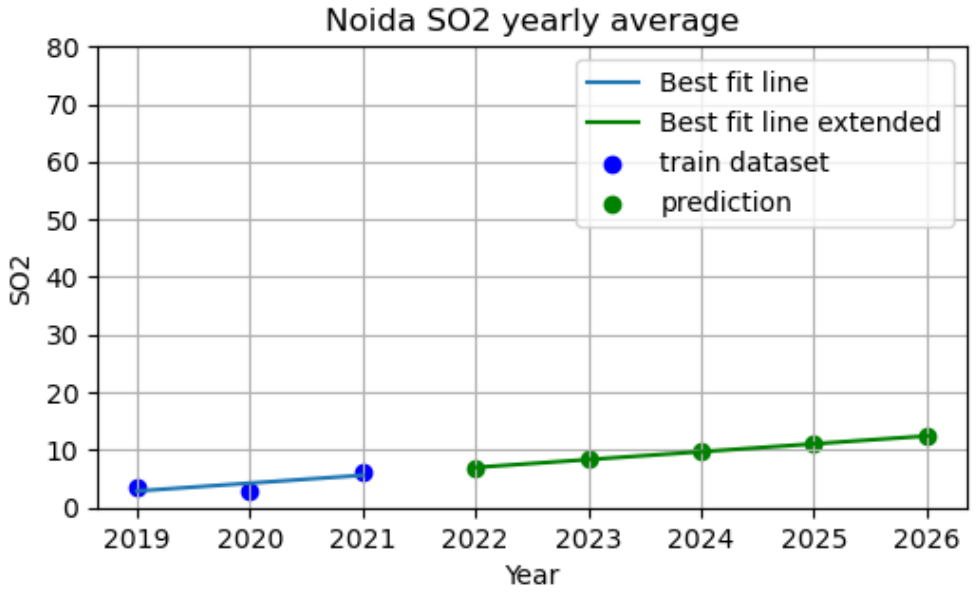


**Figure 6.109:** Noida $SO_2$ testing output

- Noida $O_3$ Yearly Average Calculation

```
nd=pd.read_csv("noida train.csv")
nd_t=pd.read_csv("noida test.csv")
nd_o3 = nd.drop(labels=['pm10','pm25','no2','so2','co'], axis=1)
nd_o3_t=nd_t.drop(labels=['pm10','pm25','no2','so2','co'], axis=1)
```

```
nx = nd_o3['Year']

ny = nd_o3['o3']

slope, intercept, r, p, std_err = stats.linregress(nx, ny)

def myfunc(nx):
  return slope * nx + intercept

mymodel = list(map(myfunc, nx))
plt.xticks(nx)
plt.ylim(10,80)
plt.title("Noida O3 yearly average")
plt.xlabel("Year")
plt.ylabel("O3")
plt.scatter(nx, ny)
plt.plot(nx, mymodel)


plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()

plt.show()
```

**Figure 6.110:** Noida $O_3$ training code

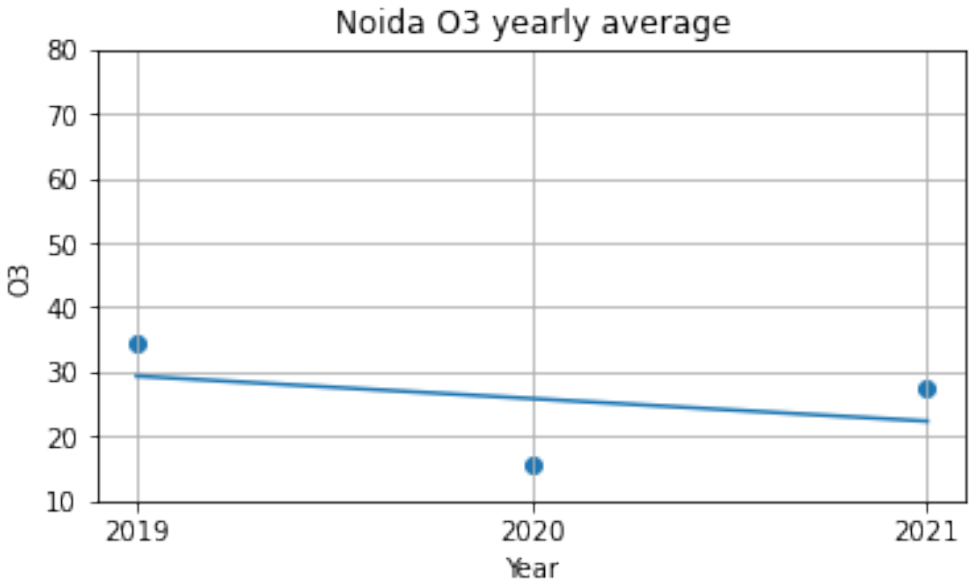


**Figure 6.111:** Noida $O_3$ training output

- Noida $O_3$ Yearly Average Calculation Testing for 2022 - 2026

```
nx_t = nd_o3_t['Year']
nx = nd_o3['Year']
ny_t = nd_o3_t['o3']
ny = nd_o3['o3']
slope, intercept, r, p, std_err = stats.linregress(nx, ny)

def myfunc(nx):
  return slope * nx + intercept

mymodel = list(map(myfunc, nx))
plt.xticks([2017,2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(0,80)
plt.title("Noida O3 yearly average")
plt.xlabel("Year")
plt.ylabel("O3")
plt.scatter(nx, ny,color="blue")
plt.plot(nx, mymodel)
plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()


plt.show()
```

**Figure 6.112:** Noida $O_3$ testing code

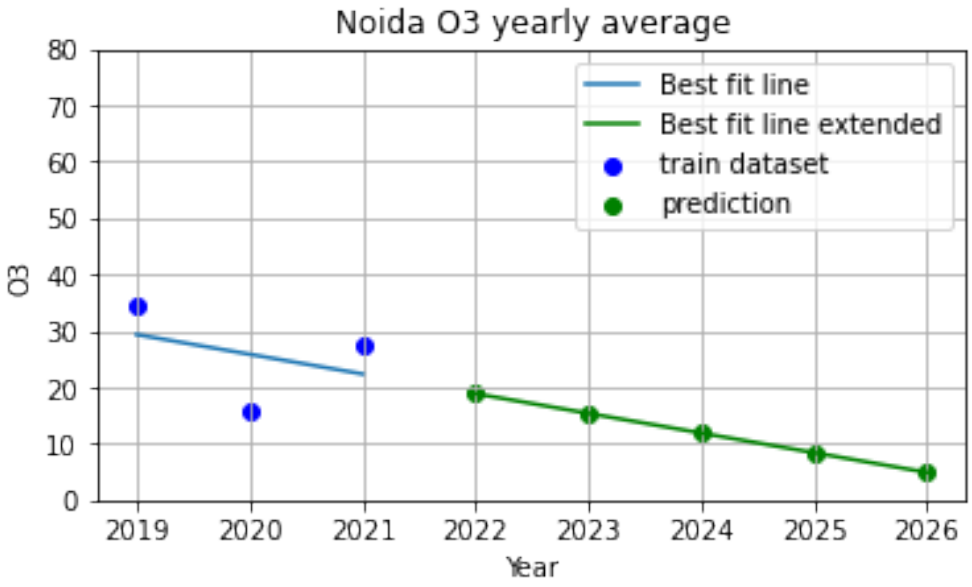


**Figure 6.113:** Noida $O_3$ testing output

- Noida CO Yearly Average Calculation

```
nd=pd.read_csv("noida train.csv")
nd_t=pd.read_csv("noida test.csv")
nd_o3 = nd.drop(labels=['pm10','pm25','so2','o3','no2'], axis=1)
nd_o3_t=nd_t.drop(labels=['pm10','pm25','so2','o3','no2'], axis=1)
```

```
nx = nd_o3['Year']

ny = nd_o3['co']

slope, intercept, r, p, std_err = stats.linregress(nx, ny)

def myfunc(nx):
  return slope * nx + intercept

mymodel = list(map(myfunc, nx))
plt.xticks(nx)
plt.ylim(0,80)
plt.title("Noida CO yearly average")
plt.xlabel("Year")
plt.ylabel("CO")
plt.scatter(nx, ny)
plt.plot(nx, mymodel)

plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()
plt.show()
```

**Figure 6.114:** Noida CO training code

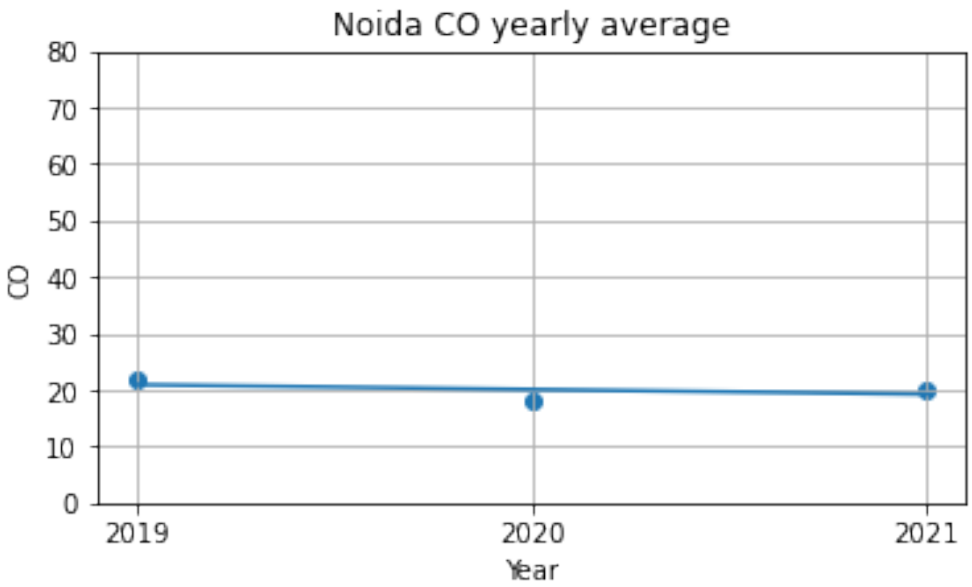


**Figure 6.115:** Noida CO training output

- Noida CO Yearly Average Calculation Testing for 2022 - 2026

```
nx_t = nd_o3_t['Year']
nx = nd_o3['Year']
ny_t = nd_o3_t['co']
ny = nd_o3['co']
slope, intercept, r, p, std_err = stats.linregress(nx, ny)

def myfunc(nx):
  return slope * nx + intercept

mymodel = list(map(myfunc, nx))
plt.xticks([2017,2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(0,80)
plt.title("Noida CO yearly average")
plt.xlabel("Year")
plt.ylabel("CO")
plt.scatter(nx, ny,color="blue")
plt.plot(nx, mymodel)
plt.scatter(2022,myfunc(2022),color="green")
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()

plt.show()
```

**Figure 6.116:** Noida CO testing code

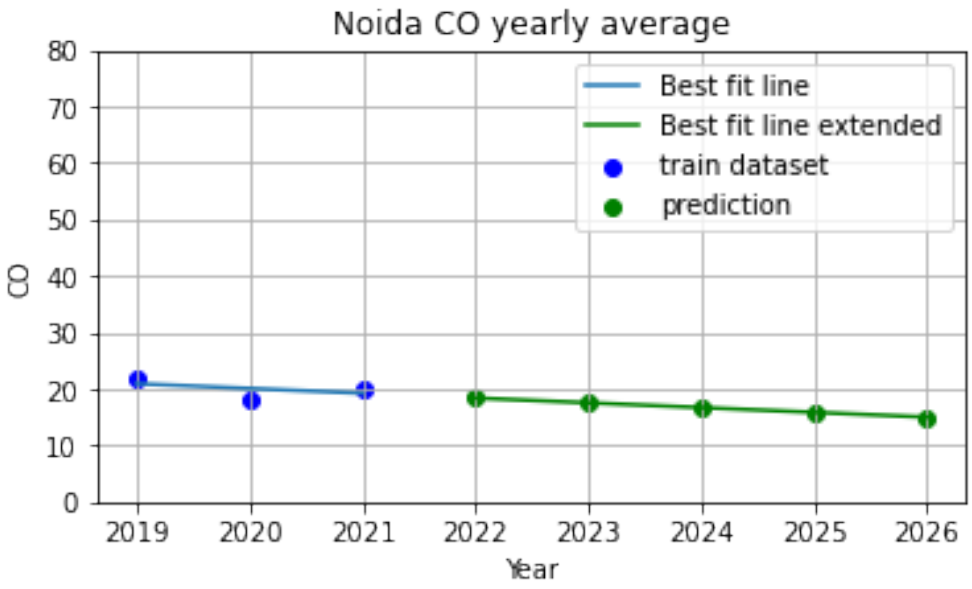


**Figure 6.117:** Noida CO testing output

- Noida $PM_{10}$ Yearly Average Calculation

```
nd=pd.read_csv("noida train.csv")
nd_o3 = nd.drop(labels=['no2','pm25','so2','o3','co'], axis=1)
```

```
nx = nd_o3['Year']

ny = nd_o3['pm10']

slope, intercept, r, p, std_err = stats.linregress(nx, ny)

def myfunc(nx):
  return slope * nx + intercept

mymodel = list(map(myfunc, nx))
plt.xticks(nx)
plt.ylim(100,400)
plt.title("Noida PM10 yearly average")
plt.xlabel("Year")
plt.ylabel("PM10")
plt.scatter(nx, ny)

plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()

plt.plot(nx, mymodel)
plt.show()
```

**Figure 6.118:** Noida $PM_{10}$ training code

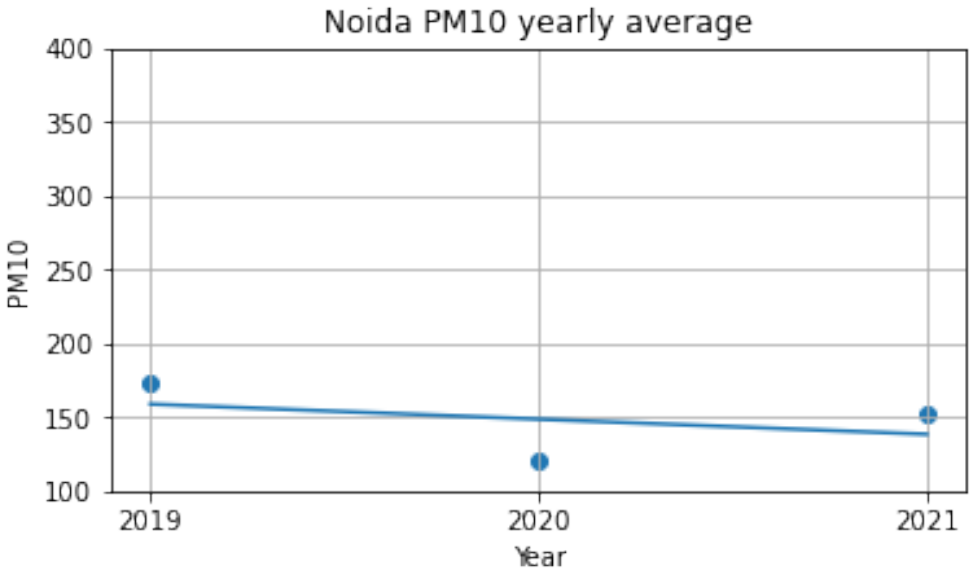


**Figure 6.119:** Noida $PM_{10}$ training output

- Noida $PM_{10}$ Yearly Average Calculation Testing for 2022 - 2026

```
nx = nd_o3['Year']
ny = nd_o3['pm10']
slope, intercept, r, p, std_err = stats.linregress(nx, ny)

def myfunc(nx):
  return slope * nx + intercept

mymodel = list(map(myfunc, nx))
plt.xticks([2017,2018,2019,2020,2021,2022,2023,2024,2025,2026])
plt.ylim(0,200)
plt.title("Noida PM10 yearly average")
plt.xlabel("Year")
plt.ylabel("PM10")
plt.scatter(nx, ny,color="blue")
plt.plot(nx, mymodel)
plt.scatter(2023,myfunc(2023),color="green")
plt.scatter(2024,myfunc(2024),color="green")
plt.scatter(2025,myfunc(2025),color="green")
plt.scatter(2026,myfunc(2026),color="green")
X=[2022,2023,2024,2025,2026]
Y=[myfunc(2022),myfunc(2023),myfunc(2024),myfunc(2025),myfunc(2026)]
plt.plot(X,Y,color="green")
plt.legend(["Best fit line","Best fit line extended","train dataset","prediction"])
plt.rcParams['figure.figsize'] = [15/2.54, 8/2.54]
plt.grid()
plt.show()
```

**Figure 6.120:** Noida $PM_{10}$ testing code

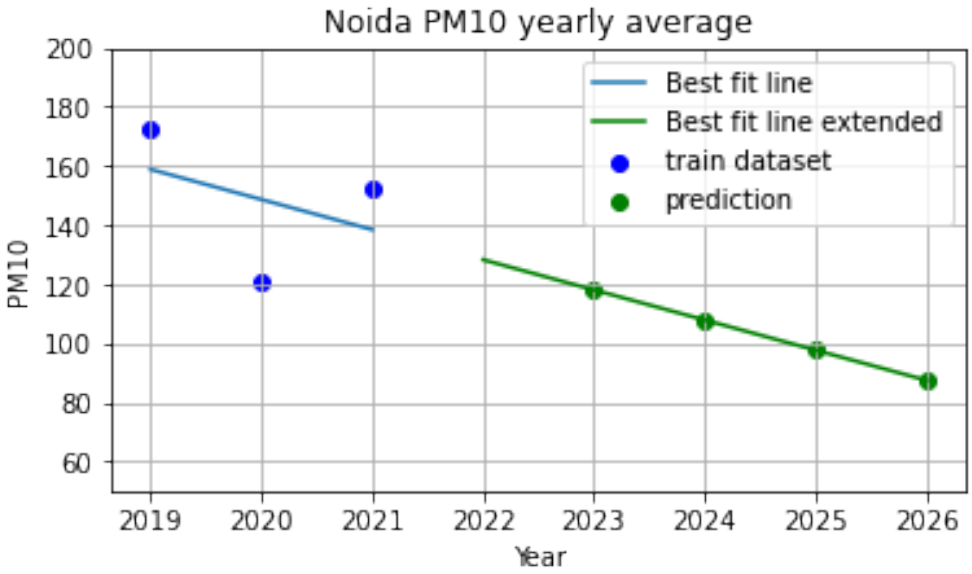


**Figure 6.121:** Noida $PM_{10}$ testing output

# Chapter 7

# AIR-QUALITY-INDEX CALCULATION

## 7.1 Air-Quality-Index Formula

The air-quality-index is calculated separately for each parameter using the equation. Calculating air-quality-index using the following equation :[23]

$$AQI = [\frac{MaxAccAQI - MinAccAQI}{MaxConc_{\text{p}} - MinConc_{\text{p}}}] \times (Conc_{\text{p}} - MinConc_{\text{p}}) + MinAccAQI \qquad (7.1)$$

Where, AQI represents the air-quality-index of pollutant p, $\text{Conc}_{\text{p}}$ represents the Pollutants' concentration, $\text{MaxConc}_{\text{p}}$ represents the Maximum concentration pf pollutant as per reference range [tab:Reference Range for air pollutants and AQI], $\text{MinConc}_{\text{p}}$ represents the Minimum concentration pf pollutant as per reference range [tab:Reference Range for air pollutants and AQI], MaxAccAQI represents the Maximum accepted AQI value corresponding to MaxConc of pollutant, MinAccAQI represents the Maximum accepted AQI value corresponding to MinConc of pollutant.

## 7.2 Air-quality-index Computation Table

Table 7.1: Air-Quality-Index for Bulandshahr

| Year | $PM_{2.5}$ | $PM_{10}$ | $O_3$ | $NO_2$ | $SO_2$ | CO |
|---|---|---|---|---|---|---|
| 2018 | 344 | 101 | 39 | 27 | 13 | 176 |
| 2019 | 210 | 87 | 37 | 26 | 16 | 109 |
| 2020 | 209 | 87 | 41 | 13 | 11 | 96 |
| 2021 | 204 | 88 | 37 | 42 | 9 | 109 |

Table 7.2: air-quality-index for Delhi

| Year | $PM_{2.5}$ | $PM_{10}$ | $O_3$ | $NO_2$ | $SO_2$ | CO |
|---|---|---|---|---|---|---|
| 2017 | 264 | 168 | 20 | 39 | 17 | 231 |
| 2018 | 267 | 166 | 22 | 41 | 11 | 251 |
| 2019 | 239 | 132 | 21 | 37 | 9 | 224 |
| 2020 | 240 | 150 | 27 | 25 | 9 | 224 |
| 2021 | 243 | 121 | 19 | 31 | 10 | 218 |

Table 7.3: air-quality-index for Ghaziabad

| Year | $PM_{2.5}$ | $PM_{10}$ | $O_3$ | $NO_2$ | $SO_2$ | CO |
|---|---|---|---|---|---|---|
| 2018 | 239 | 115 | 34 | 25 | 16 | 193 |
| 2019 | 228 | 106 | 44 | 19 | 7 | 126 |
| 2020 | 219 | 93 | 41 | 23 | 14 | 126 |
| 2021 | 224 | 106 | 41 | 17 | 17 | 126 |

Table 7.4: air-quality-index for Meerut

| Year | $PM_{2.5}$ | $PM_{10}$ | $O_3$ | $NO_2$ | $SO_2$ | CO |
|---|---|---|---|---|---|---|
| 2019 | 225 | 99 | 33 | 19 | 10 | 143 |
| 2020 | 215 | 84 | 36 | 13 | 11 | 96 |
| 2021 | 207 | 81 | 34 | 16 | 14 | 96 |

Table 7.5: air-quality-index for Noida

| Year | $PM_{2.5}$ | $PM_{10}$ | $O_3$ | $NO_2$ | $SO_2$ | CO |
|---|---|---|---|---|---|---|
| 2019 | 232 | 109 | 31 | 37 | 4 | 238 |
| 2020 | 277 | 83 | 14 | 22 | 3 | 218 |
| 2021 | 220 | 160 | 25 | 40 | 9 | 231 |

*- Due to unavailability of data for this duration

# Chapter 8

# CONCLUSION

Our analysis shows exposure to particulate matter, ozone (O3), carbon monoxide (CO), and nitrogen. Carbon dioxide (NO2) and sulfur dioxide (SO2) are all major health hazards. Ozone is a major danger. Contributes to asthma morbidity and mortality while nitrogen dioxide and sulfur dioxide may do the same It is involved in asthma, bronchial symptoms, pneumonia, and decreased lung function. excess Ozone in the air has a great impact on human health. May induce breathing Problems, asthma, decreased lung function, lung disease. Symptoms of bronchitis in children with asthma It also increases as a result of long-term NO2 exposure. SO2 can damage the respiratory system Inflammation of the eyes as well as system and lung function. Inflammation of the airways Exacerbates cough, sputum, asthma, chronic bronchitis and makes more people Prone to respiratory infections. Hospitalization on days with high SO2 levels Due to heart disease and increased mortality. Sulfuric acid is produced when SO2 reacts with water It is the main component of acid rain and contributes to deforestation. Dizziness, confusion, Loss of consciousness and death are one of the symptoms of CO.

# Appendix A

## RESEARCH PAPER

# Appendix B

## PLAGIARISM REPORT

AQI USING LR

ORIGINALITY REPORT

| 3% | 3% | 2% | 2% |
|---|---|---|---|
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | www.researchgate.net<br>Internet Source | 2% |
| 2 | Submitted to World Maritime University<br>Student Paper | <1% |
| 3 | Matthew D. Eastin, Matthew Baber, Adrienne Boucher, Sofia Di Bari, Ryan Hubler, Brandy Stimac-Spalding, Thomas Winesett. "Temporal Variability of the Charlotte (Sub)Urban Heat Island", Journal of Applied Meteorology and Climatology, 2018<br>Publication | <1% |
| 4 | ehjournal.biomedcentral.com<br>Internet Source | <1% |
| 5 | cwww.intechopen.com<br>Internet Source | <1% |
| 6 | www.litcharts.com<br>Internet Source | <1% |
| 7 | www.ncbi.nlm.nih.gov<br>Internet Source | <1% |

**Figure B.1:** Plagiarism Report Screenshot

# Bibliography


1. (1). “World’s Most Air Polluted Cities".
2. (2). “CENTRAL POLLUTION CONTROL BOARD".
3. (2). “Death Rate Increasse due to PM2.5 in India ".
4. (23). “AIR QUALITY FORMULATION".
5. (3). “WAZIRPUR DELHI MAP". https://goo.gl/maps/axJbDJ9xxCrZPrfx8.
6. (4). “VASUNDHARA, GHAZIABAD". https://goo.gl/maps/opQT8WTdjxT8UBY4A.
7. (5). “JAI BHIM NAGAR, MEERUT". https://goo.gl/maps/SH3V3mZQm1VTn6pn8.
8. (6). “YAMUNAPURAM, BULANDSHAHR". https://goo.gl/maps/uYxVa9Gmr3ae6C319.
9. (7). “SECTOR - 116, NOIDA". https://goo.gl/maps/sgijU4bhQSTMvJfN6.
10. Akhtar, A., Masood, S., Gupta, C., and Masood, A. (11). “Prediction and analysis of pollution levels in delhi using multilayer perceptron.” *Data engineering and intelligent computing*, Springer, 563–572.
11. Aladağ, E. (20). “Forecasting of particulate matter with a hybrid arima model based on wavelet transformation and seasonal adjustment.” *Urban Climate*, 39, 100930.
12. Athanasiadis, I. N., Kaburlasos, V. G., Mitkas, P. A., and Petridis, V. (8). “Applying machine learning techniques on air quality data for real-time decision support.” *First international NAISO symposium on information technologies in environmental engineering (ITEE’2003), Gdansk, Poland*, Citeseer.
13. Chhikara, P., Tekchandani, R., Kumar, N., Guizani, M., and Hassan, M. M. (19). “Federated learning and autonomous uavs for hazardous zone detection and aqi prediction in iot environment.” *IEEE Internet of Things Journal*, 8(20), 15456–15467.
14. Chowdhury, S., Dey, S., Di Girolamo, L., Smith, K. R., Pillarisetti, A., and Lyapustin, A. (17). “Tracking ambient pm2. 5 build-up in delhi national capital region during the dry season over 15 years using a high-resolution (1 km) satellite aerosol dataset.” Vol. 204, Elsevier, 142–150.
15. Dragomir, E. G. (10). “Air quality index prediction using k-nearest neighbor technique.” *Bulletin of PG University of Ploiesti, Series Mathematics, Informatics, Physics, LXII*, 1(2010), 103–108.
16. Honoré, C., Rouil, L., Vautard, R., Beekmann, M., Bessagnet, B., Dufour, A., Elichegaray, C., Flaud, J.-M., Malherbe, L., Meleux, F., et al. (9). “Predictability of european air quality: Assessment of 3 years of operational forecasts and analyses by the prev’air system.” *Journal of Geophysical Research: Atmospheres*, 113(D4).
17. Liu, H. and Chen, C. (18). “Spatial air quality index prediction model based on decomposition, adaptive boosting, and three-stage feature selection: A case study in china.” *Journal of Cleaner Production*, 265, 121777.

18. Liu, H., Li, Q., Yu, D., and Gu, Y. (15). “Air quality index and air pollutant concentration prediction based on machine learning algorithms.” *Applied Sciences*, 9(19).

19. Maleki, H., Sorooshian, A., Goudarzi, G., Baboli, Z., Birgani, Y. T., and Rahmati, M. (16). “Air pollution prediction by using an artificial neural network model.” *Clean Technol Environ Policy*, 21(6), 1341–1352.

20. Samal, K. K. R., Babu, K. S., and Das, S. K. (21). “Temporal convolutional denoising autoencoder network for air pollution prediction with missing values.” *Urban Climate*, 38, 100872.

21. Shakir, M. and Rakesh., N. (14). “Investigation on air pollutant data sets using data mining tool.” 480–485.

22. Sharma, N., Taneja, S., Sagar, V., and Bhatt, A. (12). “Forecasting air pollution load in delhi using data analysis tools.” *Procedia Computer Science*, 132, 1077–1085. International Conference on Computational Intelligence and Data Science.

23. Sinnott, R. O. and Guan, Z. (13). “Prediction of air pollution through machine learning approaches on the cloud.” 51–60.

24. Sun, W. and Xu, Z. (22). “A novel hourly pm2.5 concentration prediction model based on feature selection, training set screening, and mode decomposition-reorganization.” *Sustainable Cities and Society*, 75, 103348.

25. Xu, T., Yan, H., and Bai, Y. (19). “Air pollutant analysis and aqi prediction based on gra and improved soa-svr by considering covid-19.” *Atmosphere*, 12(3).