ONLINE SUMMER TRAINING
MACHINE LEARNING MADE EASY
FROM BASIC TO AI APPLICATION

SUBMITTED BY

SHREYA S - 12320371

ADITHYAN S – 12315255

ALAN KRISHNAN P - 12321224

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

BTech CSE – ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

LOVELY PROFESSIONAL UNIVERSITY, PUNJAB

Undertaking from the student

We the students of Bachelor of Technology under CSE discipline at Lovely Professional University, Punjab, hereby declare that all the information furnished in this project report is based on our own work and is genuine.

Name of the student: Shreya S,

Adithyan S, Alan Krishnan P

Date: 13/ 07/ 2025

Acknowledgement

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals. I would like to extend my sincere thanks to them and especially Mahipal Singh Sir . I am highly indebted to 'Lovely Professional University' for their guidance and constant supervision as well as for providing necessary information regarding the project and also their support in completing the project. I would like to express my gratitude towards our parents and members of our families for their kind cooperation and encouragement which helped us in completing this project. My thanks and appreciation also go to the people who are directly or indirectly helped us out in developing the project.

TOPIC OF THE PROJECT

BREATHEWISE – THE SMART AQI PREDICTOR

INDEX

1.ABSTRACT

Air pollution is a growing environmental and public health concern, making real-time air quality assessment essential. This project introduces Breathe Wise – A Smart Real-Time AQI Prediction Application, a web-based platform powered by machine learning models for predicting Air Quality Index (AQI) values and categories. Integrated with the Open Weather API, the application fetches live weather parameters along with pollutant concentrations for PM2.5, PM10, NO, NO2, NOx, NH3, CO, SO2, and O3.

The data is processed through a trained ML-based regression and classification pipeline that predicts the current AQI value and categorizes it as per CPCB standards. Built using Streamlit, the app delivers real-time AQI reports, predicted and actual categories, pollutant trends, and health recommendations — providing an accessible, AI-driven solution for public awareness and air quality management.

2.INTRODUCTION

Air pollution has become one of the most pressing environmental challenges of the 21st century, directly impacting human health, climate, and ecosystems. With increasing urbanization, industrialization, and vehicular emissions, air quality monitoring and management have gained significant importance. The Air Quality Index (AQI) serves as a standardized tool for reporting daily air quality levels, helping people understand the associated health risks. Traditionally, air quality data is obtained from fixed monitoring stations, which may have limited coverage and accessibility. To overcome these limitations and provide real-time, location-based air quality insights, this project introduces BreatheWise – A Smart Real-Time AQI Prediction Application. The system integrates the OpenWeather API to fetch live weather parameters and pollutant concentrations for PM2.5, PM10, NO, NO2, NOx, NH3, CO, SO2, and O3.

Leveraging machine learning models trained on historical air quality datasets, the application predicts real-time AQI values and classifies them into standard categories defined by the Central Pollution Control Board (CPCB). The platform is built using Streamlit, offering an interactive, user-friendly interface where users can select a city and instantly view the predicted AQI value, its corresponding category, live pollutant data, actual reported category from OpenWeather, and health-based recommendations.

This AI-driven solution aims to bridge the gap between static monitoring systems and dynamic, accessible air quality awareness tools, empowering individuals and communities to make informed decisions for better health and safety.

3.DATA MINING AND TASK IDENTIFICATION

In this project, data mining involves extracting, organizing, and preparing air quality data to support real-time AQI prediction. The original dataset comprises over two million records of hourly and daily air quality readings collected from multiple monitoring stations across various Indian cities. To manage this data effectively and retain its relevance for the intended prediction task, the station_day.csv file was selected, containing 108,036 records and 16 columns. This dataset provides daily pollutant concentration values such as PM2.5, PM10, NO, NO2, NOx, NH3, CO, SO2, along with station identifiers, date information, and additional environmental attributes.

The primary task identified during the data mining process is to perform data preprocessing by removing all rows containing missing or null values to ensure a clean and reliable dataset for model development. Following preprocessing, the next major task involves developing machine learning models through both regression and classification techniques. Regression models are employed to predict the exact numerical AQI value based on the pollutant concentrations, while classification models are designed to categorize the predicted AQI value into its respective air quality category, such as Good, Satisfactory, Moderate, Poor or Very Poor, as per the standards set by the Central Pollution Control Board (CPCB). Additionally, integrating real-time pollutant and weather data through the OpenWeather API constitutes a secondary task, enabling the deployment of a live AQI prediction application capable of displaying both predicted and actual reported categories for comparative analysis and public awareness.

4.METHODS APPLIED AND THEIR BRIEF DESCRIPTION

This project implements a systematic approach combining data preprocessing, feature engineering, machine learning classification algorithms, ensemble techniques, hyperparameter optimization, and evaluation metrics to build a reliable real-time AQI prediction system. The following methods and techniques were applied:

1.Data Preprocessing

2.Feature Engineering

3.Label Encoding

4.Feature Scaling

5.Classification Models

5.1 XGBoost Classifier (XGBClassifier)

5.2 LightGBM Classifier (LGBMClassifier)

5.3 CatBoost Classifier (CatBoostClassifier)

6. Ensemble Learning (Stacking Classifier)

7. Cross-Validation

8. Hyperparameter Optimization

9. Model Evaluation

9.1 Accuracy Score

4.1 DATA PREPROCESSING

Before training any machine learning model, the dataset needs to be cleaned and made consistent. In your case:

- Rows containing missing values in any important column (like pollutant concentrations) were dropped to avoid misleading the models.

- Irrelevant columns (StationId, Date..) were removed since they don't contribute to the prediction of AQI_Bucket.
  This step ensures the data passed to the models is clean, reliable, and optimized for learning.

4.2 FEATURE ENGINEERING

To improve model performance, new features were created from existing ones:

- PM2.5/PM10 — ratio between fine and coarse particulate matter.

- NO/NO2 — ratio between nitric oxide and nitrogen dioxide.
  These derived features capture hidden relationships between pollutants, helping the model better understand pollutant dynamics in determining air quality.

4.3 LABEL ENCODING

The target variable AQI_Bucket contains categorical labels (Good, Moderate, Poor, etc.). Most ML models require numeric input, so these categories were converted into numerical labels using LabelEncoder, ensuring compatibility with your classification algorithms.

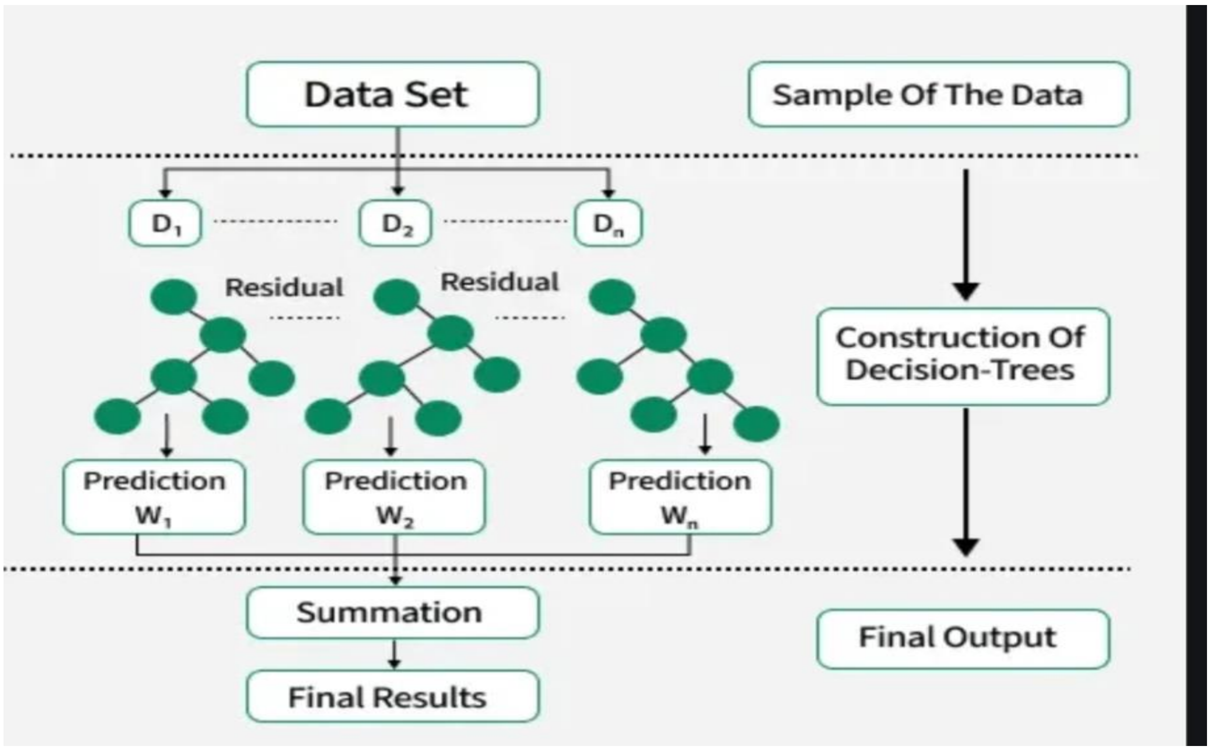
4.4 FEATURE SCALING

Since different pollutant values have different units and scales, it's important to normalize them so no single feature dominates. Using StandardScaler, each feature was rescaled to have a mean of 0 and standard deviation of 1 — ensuring fair weightage during training.

4.5 CLASSIFICATION MODELS

4.5.1 XGBoost Classifier: is a highly efficient and scalable implementation of gradient boosting algorithms, widely recognized for its superior performance on structured and tabular datasets. It works by building an ensemble of decision trees sequentially, where each subsequent tree focuses on minimizing the errors of its predecessors. XGBoost offers advanced regularization techniques (L1 and L2) to prevent overfitting, handles missing values internally, and provides parallel processing capabilities, making it one of the most preferred algorithms in competitive machine learning tasks. In this project, XGBoost was selected for its speed, accuracy, and capability to handle numerical pollutant concentration data effectively.

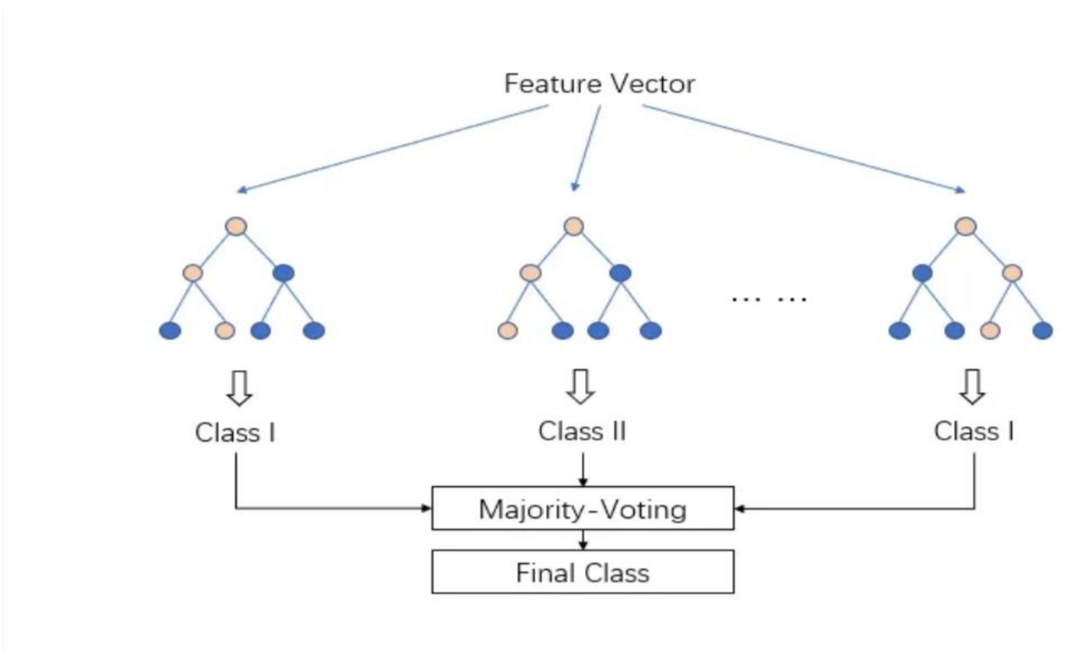
XGBoost



LOGO AND ARCHITECTURE DIAGRAM

4.5.2 LightGBM Classifier: is a fast, distributed, and highly efficient gradient boosting framework based on decision tree algorithms. Unlike traditional boosting methods, LightGBM uses a histogram-based algorithm, which speeds up the training process and reduces memory consumption. It builds trees leaf-wise (with depth constraints) rather than level-wise, enabling it to reduce more loss and improve accuracy. LightGBM handles large datasets with ease and provides support for parallel learning and categorical feature handling. Its ability to deliver high accuracy with faster training times made it a valuable component in the AQI prediction model ensemble.

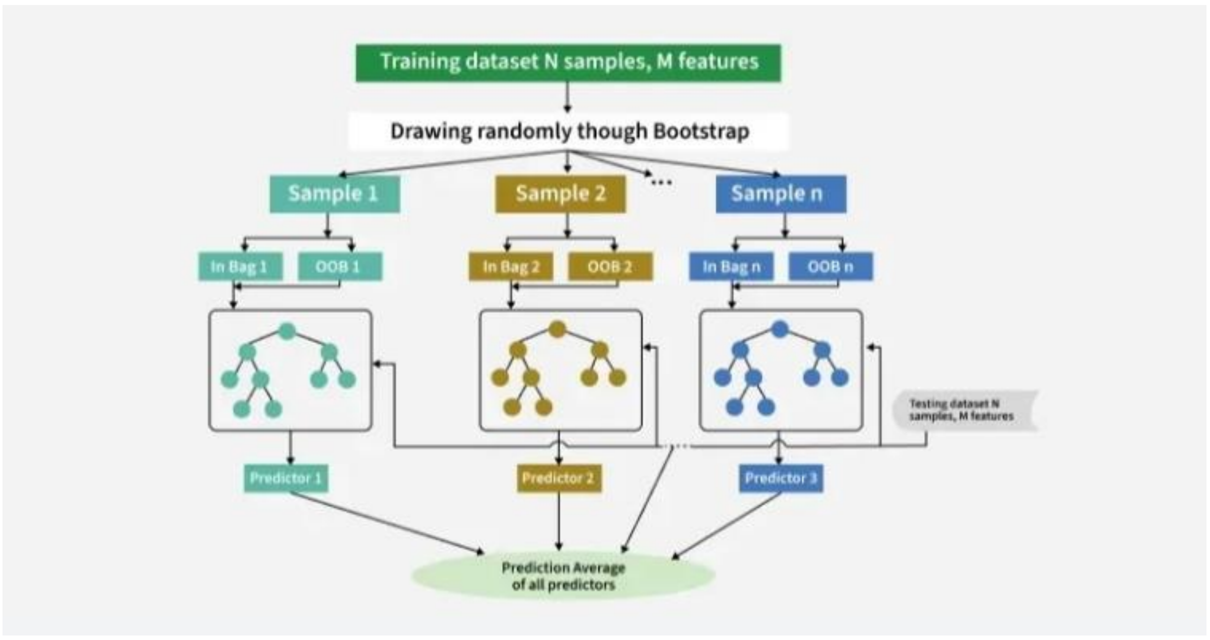
LightGBM



LOGO AND ARCHITECTURE DIAGRAM

4.5.3 CatBoost Classifier: is a gradient boosting library developed by Yandex, optimized for both numerical and categorical data. While this project’s dataset primarily involves numerical features, CatBoost remains valuable due to its ability to automatically handle missing values, prevent overfitting through techniques like ordered boosting, and require minimal hyperparameter tuning. CatBoost offers stable and reliable performance, especially when used as part of an ensemble, and is known for its simplicity in implementation without sacrificing model quality.

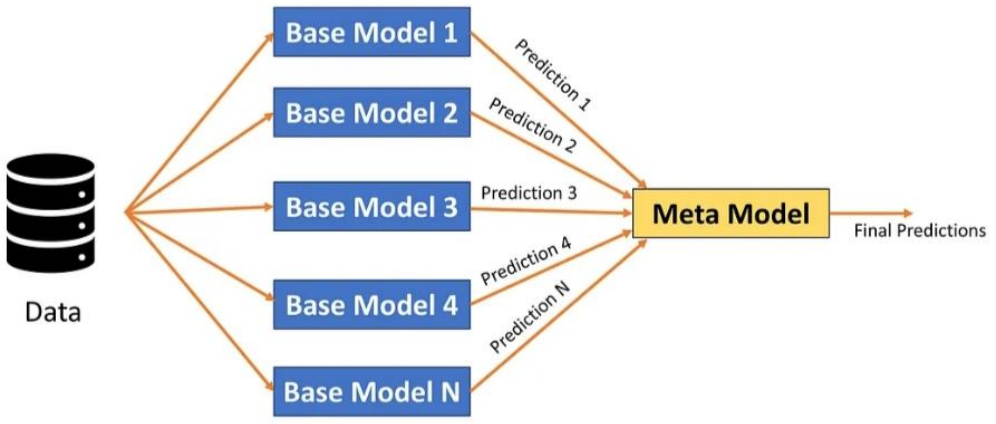
CatBoost


Training dataset N samples, M features
Drawing randomly though Bootstrap
Sample 1
Sample 2
Sample n
In Bag 1
OOB 1
In Bag 2
OOB 2
In Bag n
OOB n
Testing dataset N samples, M features
Predictor 1
Predictor 2
Predictor 3
Prediction Average of all predictors

LOGO AND ARCHITECTURE DIAGRAM

4.6 ENSEMBLE LEARNING(STACKING CLASSIFIER):

4.6 ENSEMBLE LEARNING(STACKING CLASSIFIER): is an ensemble machine learning technique that integrates the predictions of multiple base learners to create a more robust and accurate final model. In this project, XGBoost, LightGBM, and CatBoost classifiers were employed as base learners, each trained independently on the same data. Their predictions were then combined and fed into a meta-classifier, another CatBoost model, which learned to make final predictions based on the strengths and weaknesses of the base models. Stacking typically outperforms individual models by leveraging their complementary decision-making capabilities, effectively reducing errors and improving overall prediction reliability. By using this ensemble approach, the system achieves a balanced and high-performing AQI classification model suitable for real-time prediction applications.

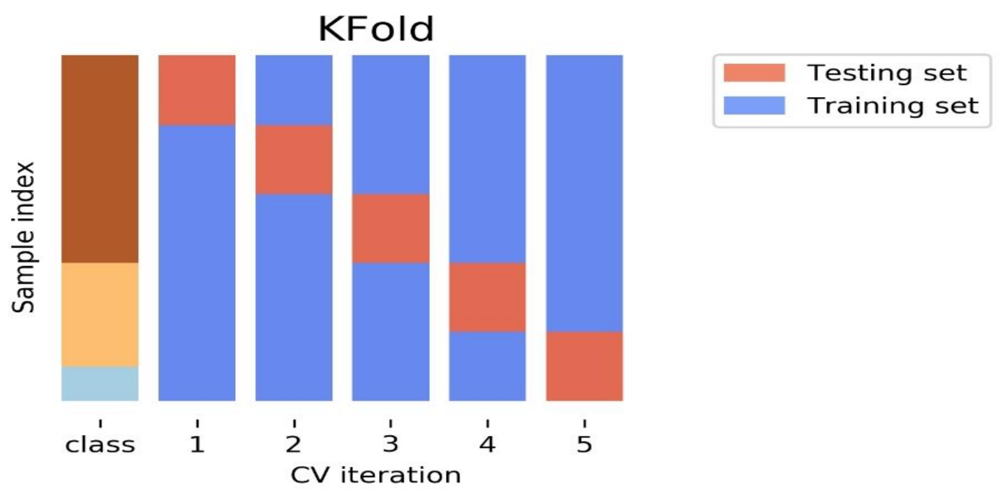


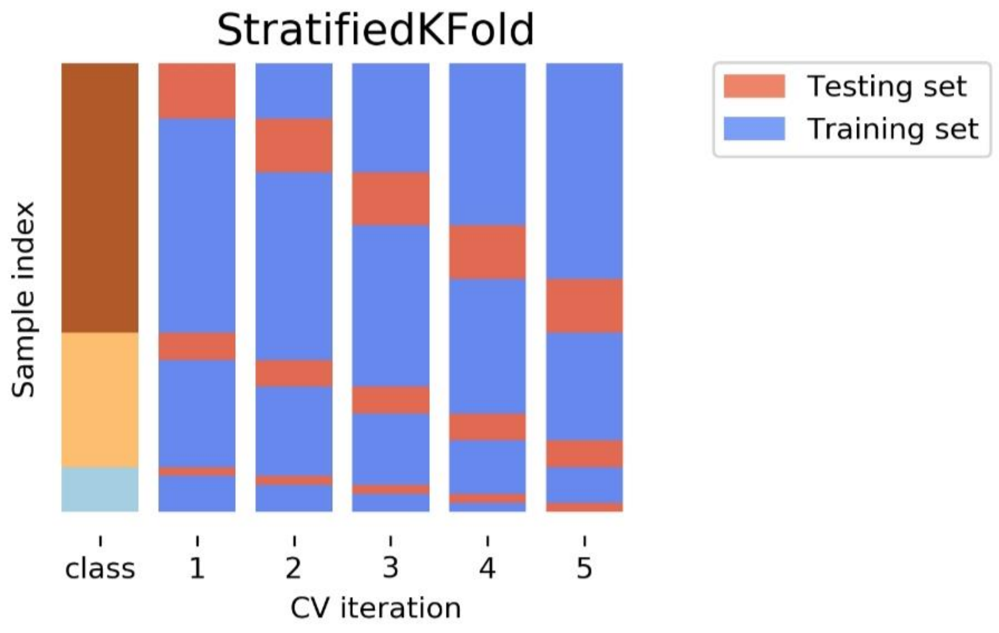
A *stacking ensemble diagram* showing base models feeding into a final meta-learner.

4.7 CROSS VALIDATION

Stratified K-Fold Cross-Validation (StratifiedKFold):
A robust technique used to split data into multiple folds while preserving the percentage of samples for each class. Ensures that each fold is a good representative of the overall dataset's class distribution essential in classification tasks, especially with imbalanced data.

Nested Cross-Validation within StackingClassifier:
The stacking model itself uses internal cross-validation (via cv=StratifiedKFold(n_splits=3)) to prevent overfitting and ensure generalization of the ensemble model.





4.8 HYPERPARAMETER OPTIMIZATION:

Optuna is a state-of-the-art hyperparameter optimization framework designed for automatic and efficient tuning of machine learning models. It employs advanced optimization strategies such as Bayesian optimization to intelligently sample hyperparameter combinations within specified ranges. In this project, functions like trial.suggest_int and trial.suggest_float were used to define ranges for parameters like the number of estimators, maximum tree depth, learning rate, and regularization factors for each model. Optuna evaluates these combinations through cross-validation, selecting the set of parameters that yields the highest validation accuracy. This automated tuning ensures that each model operates under optimal conditions, significantly improving the overall performance of the final ensemble.

4.9 MODEL EVALUATION:

4.9.1 Accuracy Score:

It measures the percentage of total correct predictions made by the model out of all predictions. It is a straightforward metric, useful when class distributions are balanced. However, it can be misleading for imbalanced datasets.

4.9.2 Precision Score (Weighted):

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The weighted version calculates the score for each class and averages them, considering the number of samples in each class. It reflects how precise the model’s predictions are across all categories.

4.9.3 Recall Score (Weighted):

Recall is the ratio of correctly predicted positive observations to all actual positives. The weighted average considers class-wise recall scores based on their support (number of true instances per class). It indicates how well the model captures true instances from each category.

4.9.4 F1-Score (Weighted):

The F1-score is the harmonic mean of precision and recall, balancing both metrics. The weighted version computes this for each class and averages it, factoring in class proportions. It’s a good overall measure for multiclass classification performance, especially when dealing with uneven class distributions.

4.9.5 Confusion Matrix:

A confusion matrix is a table showing the actual versus predicted classifications for each category. It highlights how many predictions were correct and where the model misclassified. It provides detailed insights into the performance for each individual class.

4.9.6 ROC-AUC Score (Macro Average):

The ROC-AUC score measures the model's ability to distinguish between classes by plotting the true positive rate against the false positive rate. In multiclass problems, classes are one-hot encoded, and a macro-average computes the average ROC-AUC score across all classes equally. A higher score indicates better model discrimination ability.

4.10 Regression Models

4.10.1 XGBoost Regressor (XGBRegressor)

XGBoost Regressor is an optimized gradient boosting library designed to be highly efficient, flexible, and portable. It works by building a series of weak learners (decision trees) in sequence, where each tree attempts to correct the errors of the previous one. Known for its regularization capabilities, handling missing values, and parallel computation, it’s widely used for regression tasks with structured/tabular data.

4.10.2 LightGBM Regressor (LGBMRegressor)

LightGBM is a fast, distributed, high-performance gradient boosting framework based on decision tree algorithms. It uses histogram-based algorithms, which bucket continuous feature values into discrete bins, reducing memory usage and speeding up training time. It is particularly effective for large datasets with numerous numerical features and delivers high accuracy with lower computational resources.

4.10.3 CatBoost Regressor (CatBoostRegressor)

CatBoost is a gradient boosting algorithm developed by Yandex, optimized for categorical and numerical data. It handles categorical variables internally without the need for explicit encoding and combats overfitting with ordered boosting and random permutations. It is reliable, user-friendly, and delivers strong predictive performance with minimal tuning in regression tasks.

4.11. Ensemble Learning (Regressor)

To improve model robustness and generalization, a Stacking Regressor ensemble was constructed. This ensemble combined the predictions from XGBoost, LightGBM, and CatBoost regressors as base models. Their predictions were then passed to a Gradient Boosting Regressor meta-model, which learned to optimally weight and blend these outputs to generate final AQI predictions. This approach leverages the strengths of multiple models, reducing prediction variance and improving overall accuracy.

4.12. Model Evaluation

4.12.1 R^2 Score (Coefficient of Determination)

This metric indicates the proportion of variance in the target variable (AQI) that can be explained by the input features. An R^2 score closer to 1 signifies a better fit, indicating the model's high predictive capability.

4.12.2 Mean Absolute Error (MAE)

MAE represents the average absolute difference between the actual and predicted AQI values. It is an intuitive and direct measure of prediction accuracy, where lower values indicate better model performance.

4.12.3 Root Mean Squared Error (RMSE)

RMSE is the square root of the average of squared differences between predicted and actual values. It penalizes larger errors more than MAE, providing a more sensitive measure of prediction performance, especially in cases where occasional large errors matter.

5. DATASET DESCRIPTION

The unbalanced dataset contains 47387 observations and 10 features and the balanced dataset contains 17527 observations and 10 features. Each observation represent a station and all the features together contributes to the pollutant and air quality information about a particular customer.

5.1 FEATURE DESCRIPTION

The original dataset comprised several important features representing daily pollutant concentrations, including PM2.5 (fine particulate matter), PM10 (coarse particulate matter), NO (Nitric Oxide), NO2 (Nitrogen Dioxide), NOx (Oxides of Nitrogen), NH3 (Ammonia), CO (Carbon Monoxide), and SO2 (Sulphur Dioxide). Along with these, metadata columns like StationId, Date, AQI (Air Quality Index value), and AQI_Bucket (air quality category) were also present. Since these pollutants directly impact air quality, they were used as predictive features for AQI category classification.
To improve model performance and capture hidden relationships between pollutant levels, feature engineering was applied by creating two additional ratio-based features:

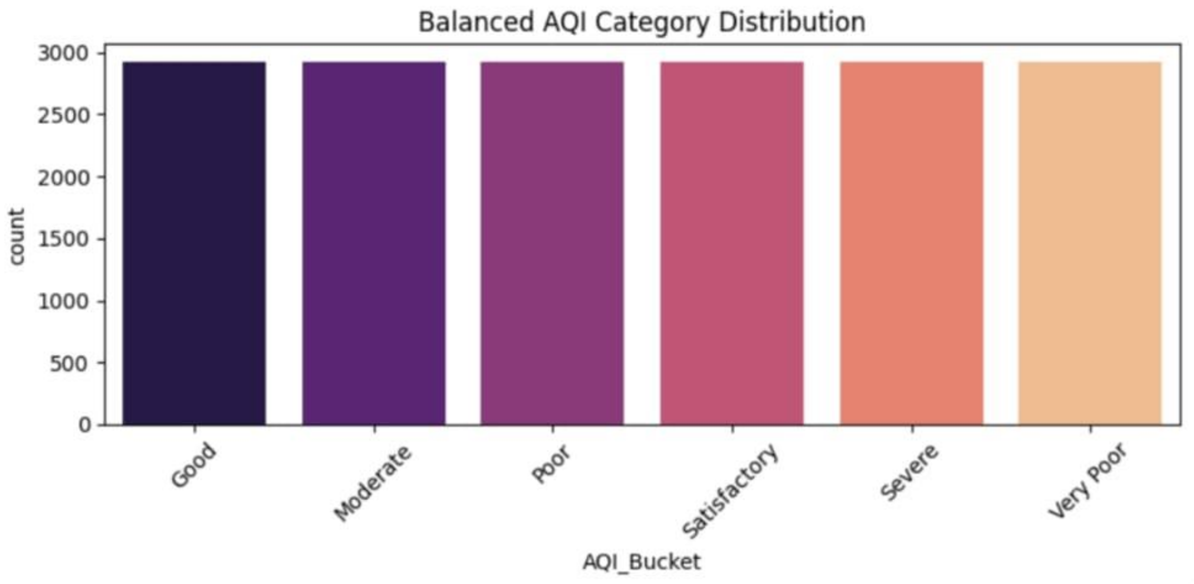
- PM2.5/PM10: Represents the proportion of fine particulate matter to total particulate matter, useful in assessing pollution type and severity.
- NO/NO2: Highlights the chemical balance between nitric oxide and nitrogen dioxide, important for understanding vehicular emission impact and photochemical reactions in air.

These engineered features introduced meaningful interactions into the dataset, enriching the input variables and enabling the machine learning models to make more accurate AQI category predictions.

5.2 EDA – EXPLORATORY DATA ANALYSIS

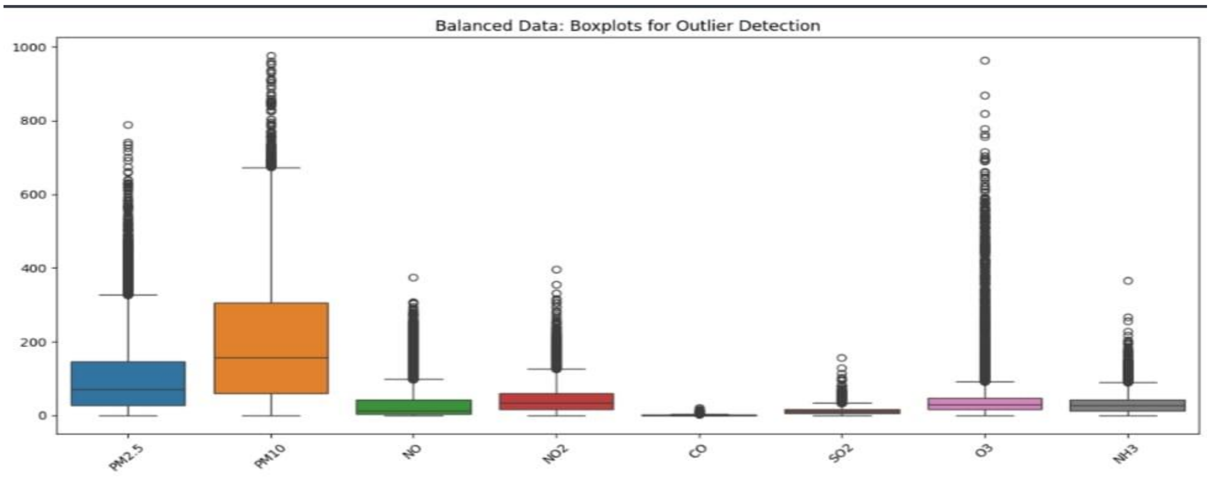
Exploratory Data Analysis (EDA) was performed to understand the distribution, patterns, relationships, and anomalies within the dataset, helping identify trends in pollutant concentrations and their influence on air quality categories before model development.

5.2.1 BALANCED AQI CATEGORY DISTRIBUTION DATA



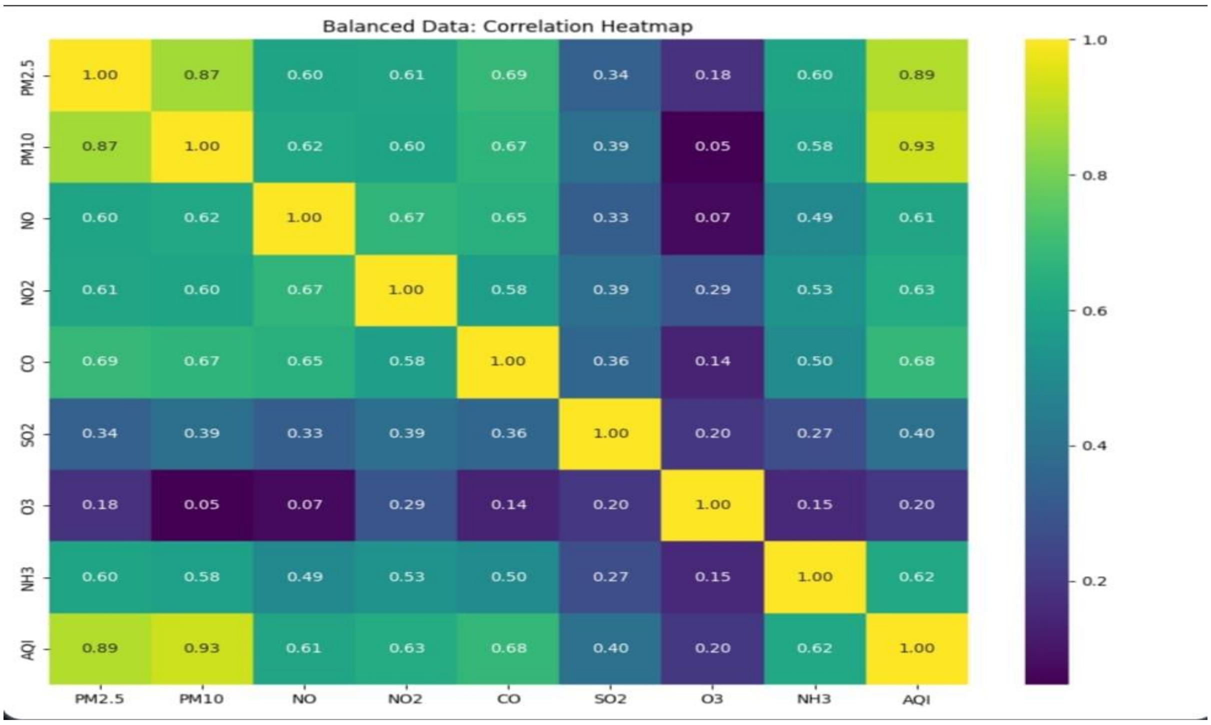
5.2.2 BOX PLOT OF BALANCED DATASET

Boxplots were used to visualize the distribution, spread, and presence of outliers in individual pollutant concentration features, making it easier to detect extreme values and data variability.

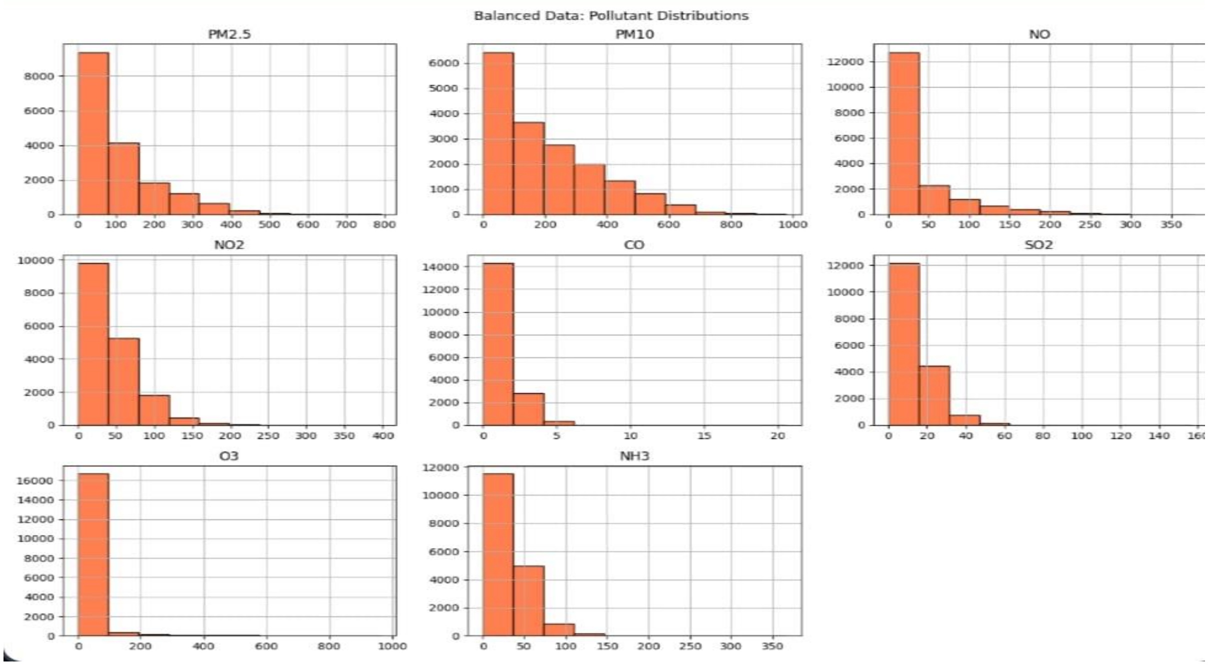


5.2.3 CORRELATION HEATMAP OF BALANCED DATASET

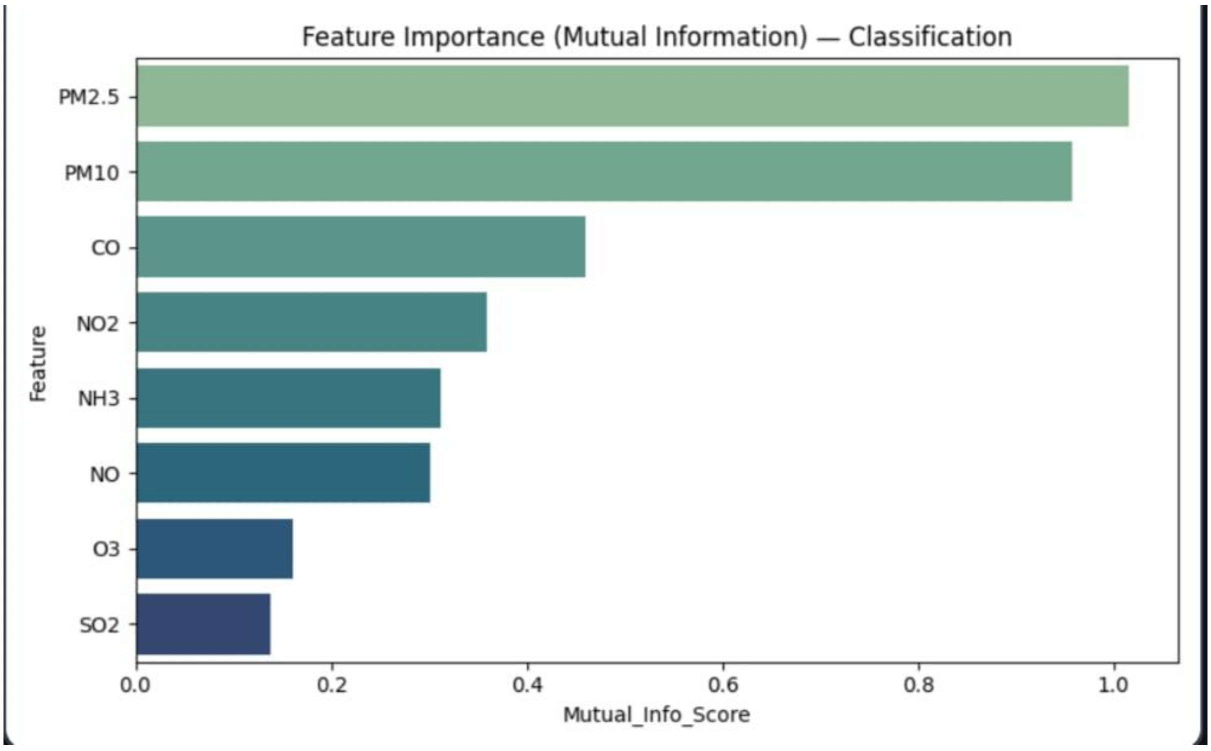
Correlation heatmaps provided a graphical representation of the relationships between pollutants, highlighting how certain features like PM2.5 and PM10 or NO and NO2 are strongly correlated, aiding in feature selection and engineering decisions.



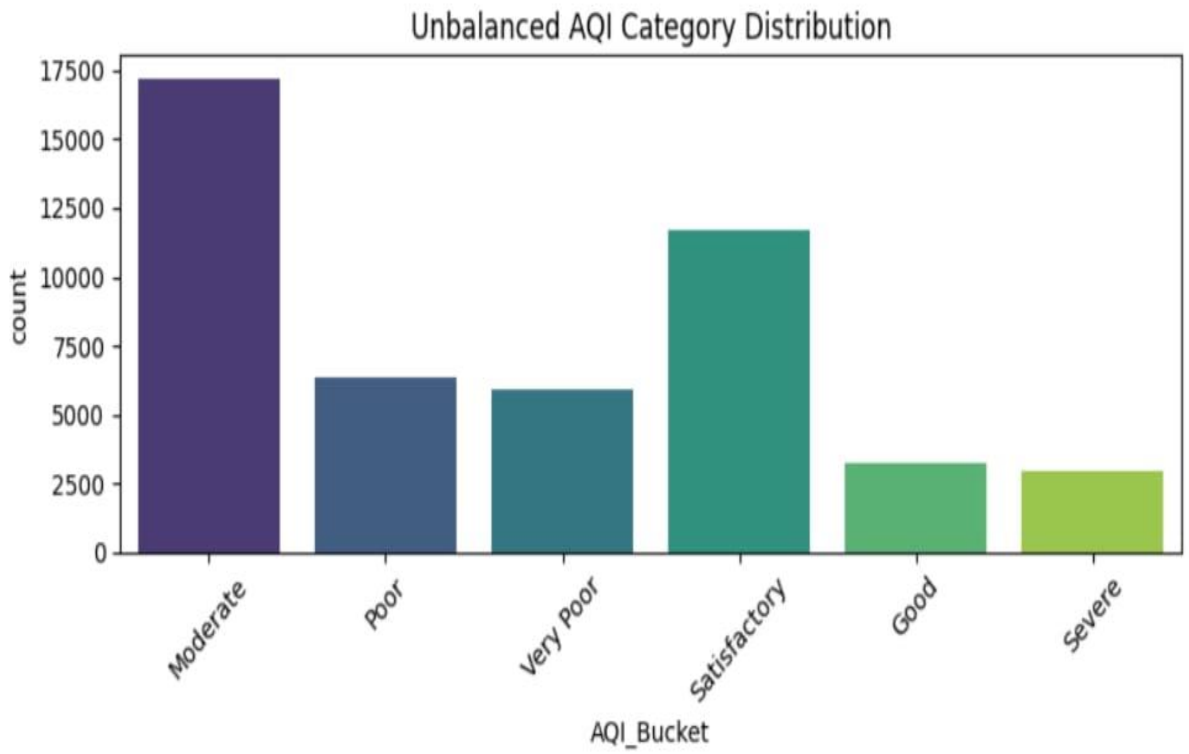
5.2.4 POLLUTANT DISTRIBUTION OF BALANCED DATASET



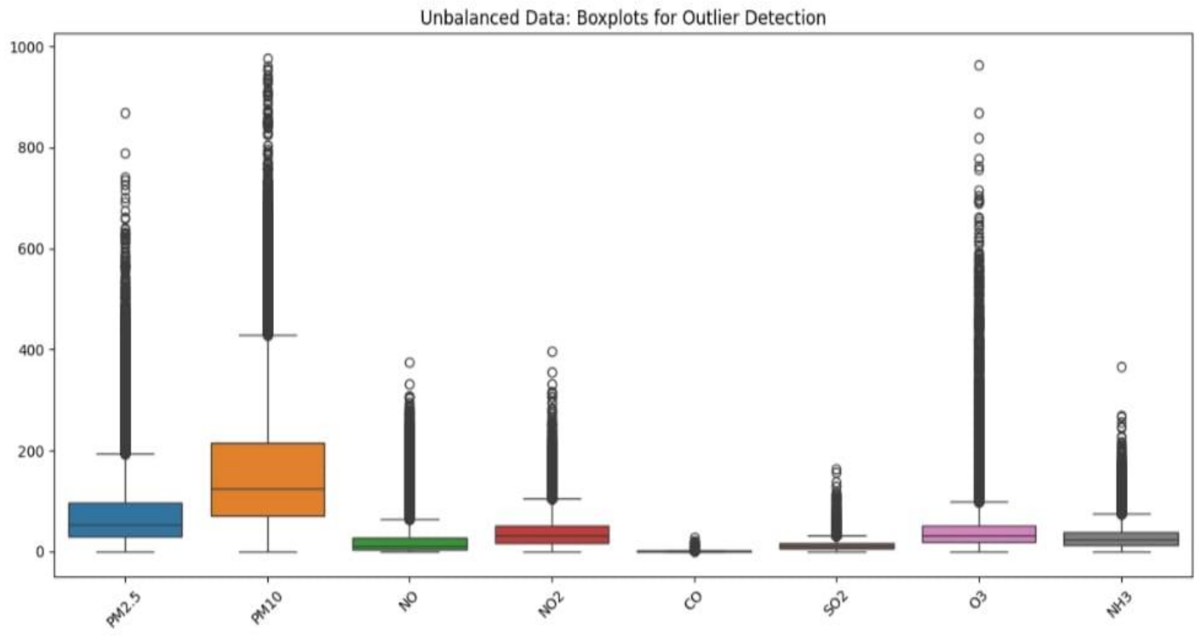
5.2.5 FEATURE IMPORTANCE



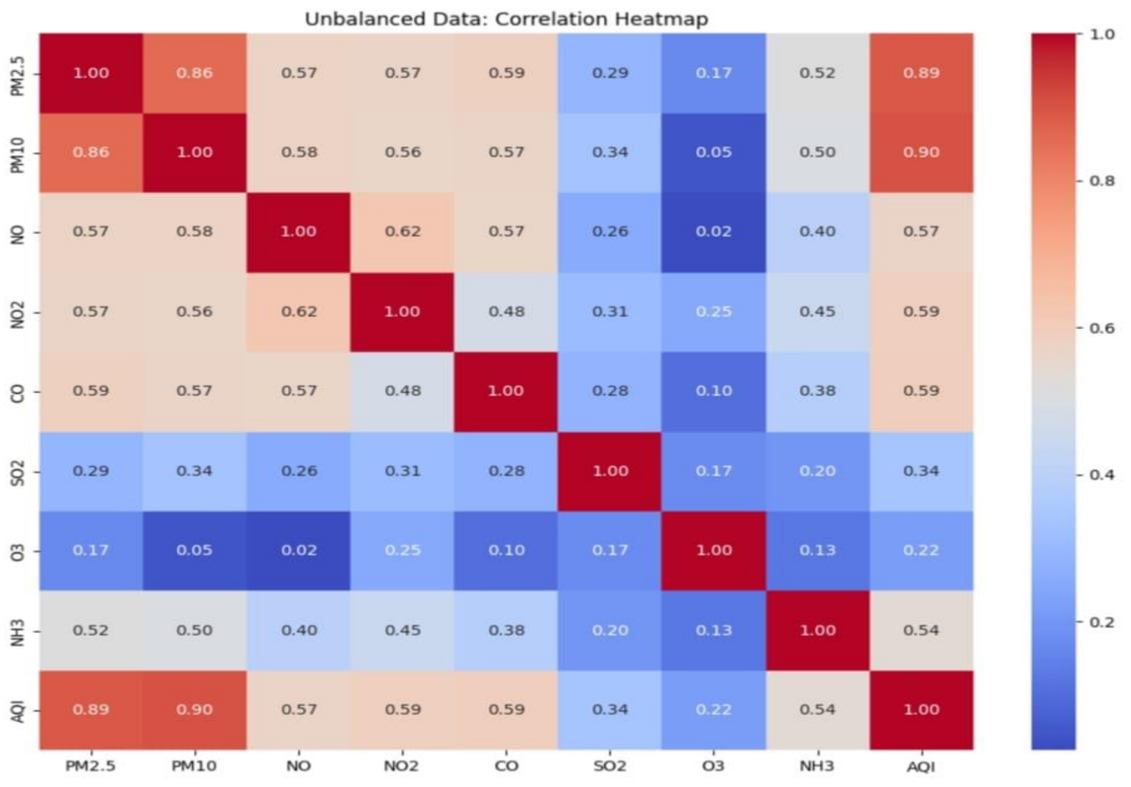
5.2.6 AQI CATEGORY DISTRIBUTION ON UNBALANCED DATASET



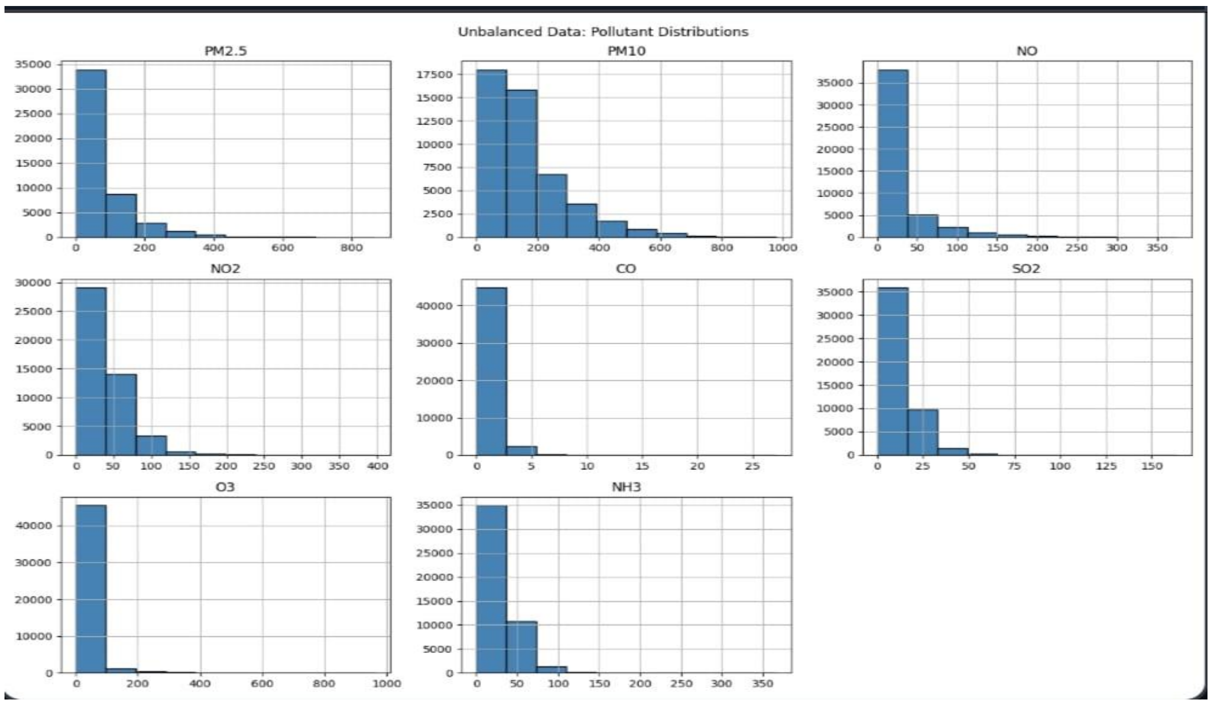
5.2.7 BOXPLOT OF UNBALANCED DATASET



5.2.8 CORRELATION HEATMAP OF UNBALANCED DATASET



5.2.9 POLLUTANT DISTRIBUTION OF UNBALANCED DATASET



6. DATA PREPROCESSING

The first crucial step in this project was data preprocessing, aimed at ensuring the dataset was clean, consistent, and reliable for machine learning. Initially, the ‘Date’ column, which was in string format, was converted into a proper datetime object to standardize date values and enable potential time-based analysis if needed. Certain pollutant columns such as ‘Benzene’, ‘Toluene’, ‘Xylene’, and ‘NOx’ were dropped since either their values were missing for many records or they were not retrievable via the OpenWeather API in real-time, making them impractical for a live AQI prediction system.
To maintain data integrity and avoid issues during model training, all rows containing missing or null values in any of the relevant columns were completely removed. This was done using the dropna() function in pandas, ensuring that every record in the dataset was complete and consistent. The result was a clean, reliable dataset containing only the essential pollutant concentration features and the corresponding AQI categories, suitable for further analysis and model development.

6.1 DATA MANIPULATION

Once the dataset was cleaned, the next critical step involved addressing class imbalance in the target variable ‘AQI_Bucket’. The initial dataset exhibited an unequal distribution of air quality categories, with some classes like ‘Moderate’ and ‘Satisfactory’ having a higher number of records than others like ‘Severe’ or ‘Very Poor’. This imbalance can bias a classifier toward the majority classes, reducing accuracy and recall for minority classes.
To resolve this, a downsampling strategy was applied where the number of records in each AQI category was reduced to match the count of the category with the fewest samples. This was done by randomly selecting an equal number of records from each category using the pandas groupby() and sample() functions. As a result, a balanced dataset was created with equal representation from all AQI categories, ensuring that the models learned to classify each air

quality level fairly without favoritism toward the majority classes.

6.2 FEATURE SCALING

Since the pollutant concentration values in the dataset were measured in different units and had varying ranges (for example, PM2.5 values could range from 10 to 400 while SO2 might range from 1 to 50), it was necessary to normalize these values before training the models. Unequal ranges can lead to certain features disproportionately influencing the learning process, especially in algorithms sensitive to the magnitude of feature values.
In this project, StandardScaler from scikit-learn was applied to scale the features. This technique transforms the data so that each feature has a mean of 0 and a standard deviation of 1. This ensures that no feature dominates the learning process, and the models converge faster and more reliably during training, particularly important when using ensemble models with internal boosting mechanisms.

6.2.1 StandardScaler

It standardizes the data by subtracting the mean and scaling to unit variance, so each feature has a mean of 0 and a standard deviation of 1. It works well when data follows a normal distribution and is commonly used in models sensitive to feature magnitudes, like logistic regression or ensemble models.
In this project, StandardScaler was chosen since it effectively standardizes pollutant values without being overly sensitive to moderate outliers, ensuring balanced input for ensemble learning models.

6.3 LABEL ENCODING

In this project, Label Encoding was applied to the target variable

‘AQI_Bucket’, which contained categorical values representing different air quality levels such as ‘Good’, ‘Satisfactory’, ‘Moderate’, ‘Poor’, ‘Very Poor’, and ‘Severe’. Since most machine learning algorithms cannot directly process categorical string values, Label Encoding converted these categories into unique numeric labels like 0, 1, 2, 3, 4, and 5. This technique is particularly efficient when dealing with a single categorical feature, especially when the categories are mutually exclusive without any explicit ordinal relationship.

Although label encoding can sometimes introduce unintended ordinal relationships between categories, it was appropriate here because the air quality categories were treated as distinct classes in a multiclass classification problem. Furthermore, it ensured compatibility with ensemble models like XGBoost, LightGBM, and CatBoost, which require numerical target variables during training. This simple yet effective encoding step made it possible to train and evaluate the classification models smoothly.

6.4 FEATURE SELECTION

Feature Selection is a critical process in any machine learning workflow, aimed at identifying and retaining the most significant and impactful variables while discarding those that add noise, redundancy, or unnecessary complexity. In this AQI prediction project, feature selection was conducted meticulously in multiple phases — combining domain expertise, data understanding from EDA, and correlation analysis insights to determine which features directly influenced the air quality category predictions.
The initial dataset contained several columns, some of which like ‘StationId’ and ‘Date’ served only as identifiers or temporal markers, without contributing any predictive power to the model. These were immediately dropped to reduce dimensionality and streamline the dataset. The ‘AQI’ column, though highly correlated to the target variable ‘AQI_Bucket’, was also removed to prevent data leakage, which could result in artificially inflated model accuracy without genuinely predictive learning.

As part of feature selection, EDA techniques such as correlation heatmaps and distribution plots helped reveal strong linear relationships between pollutant pairs like PM2.5 and PM10 or NO and NO2, indicating possible redundancy or valuable interaction effects. Rather than removing correlated features outright, the project opted for feature engineering by creating new ratio-based variables like 'PM2.5/PM10' and 'NO/NO2'. These new features helped capture the relative proportion of pollutant concentrations, providing additional predictive insight that individual pollutant values alone might miss.

This combination of feature selection and feature engineering ensured that the final dataset retained only those features that had meaningful relationships with the AQI categories. It reduced computational complexity, prevented overfitting, improved model generalization, and made the overall prediction system more efficient. Careful feature selection like this is especially valuable in real-time applications where model response time and interpretability matter, such as a live AQI prediction app integrated with OpenWeather API.

7. MODEL EVALUATION

After developing both classification and regression models for AQI prediction, it is essential to assess their performance using appropriate evaluation metrics. These metrics offer valuable insights into how well the model is performing and help identify areas of improvement. In this project, different evaluation techniques were applied based on the type of model — classification or regression — to quantify prediction accuracy and reliability.

7.1 MODEL EVALUATION PROCEDURES

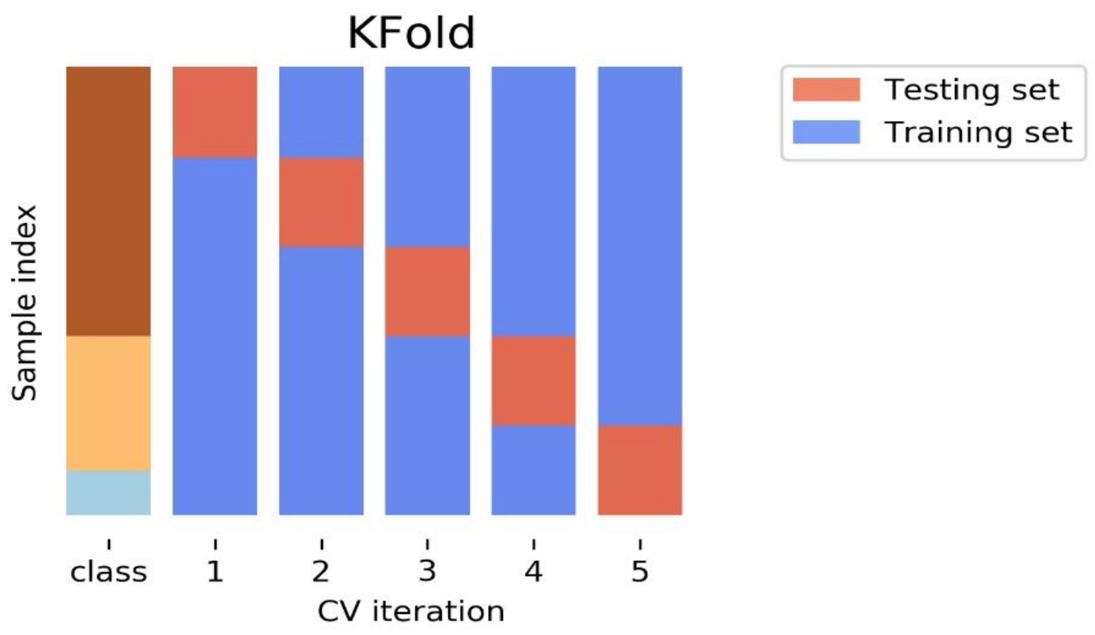
7.1.1 Train-Test Split (Holdout Method)

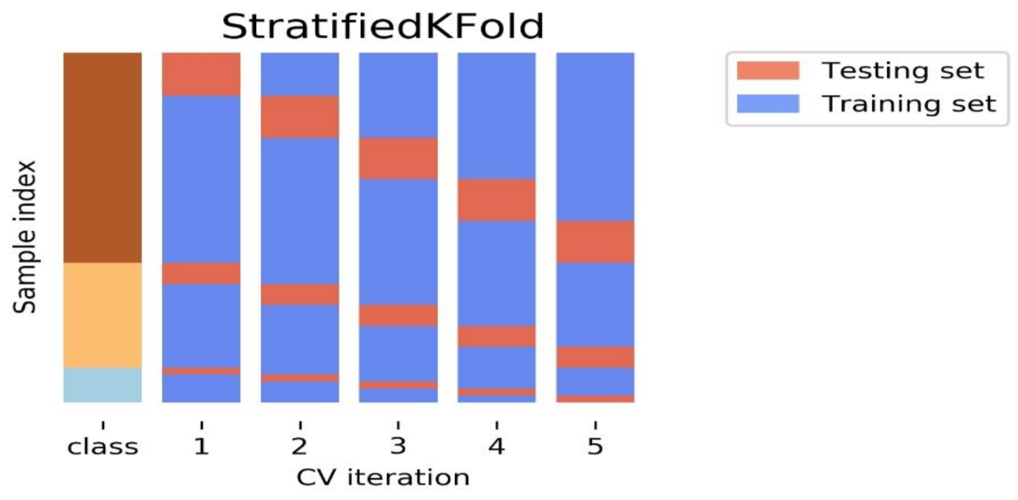
This is a widely-used, straightforward evaluation method where the dataset is randomly divided into two separate parts — typically 70% for training and 30% for testing. The model is trained on the training data and then tested on the unseen testing data to estimate how well it generalizes to new, real-world samples. Although quick and easy to implement, this method can sometimes produce unstable results depending on how the data was split, especially with limited data size.

7.1.2 K-Fold Cross-Validation

To obtain a more reliable and unbiased estimate of model performance, K-Fold Cross-Validation was implemented — with 5 folds in this project. The entire dataset is divided into 5 equal subsets (folds). The model is trained on 4 of them and validated on the remaining one. This process is repeated 5 times, each time using a different fold as the validation set. The final performance score is averaged across all folds.
For classification models, Stratified K-Fold Cross-Validation was applied to ensure that each fold maintained the same class distribution

as the original dataset, thus preserving the balance of AQI categories.





7.2 CLASSIFICATION MODEL EVALUATION METRICS

To assess the performance of the AQI category prediction model, several standard classification evaluation metrics were used:

- **Accuracy Score:**
  Represents the percentage of total predictions the model got correct. While useful as an overall performance indicator, it can

be misleading in imbalanced datasets where some classes are underrepresented.

- **Precision Score (Weighted):**
  Measures how many of the positive predictions made by the model were actually correct, averaged across classes with weighting based on the number of instances per class. It reflects the model's ability to avoid false positives.

- **Recall Score (Weighted):**
  Indicates the proportion of actual positive cases correctly identified by the model, averaged with class weights. It reflects the model's ability to capture all relevant instances and avoid false negatives.

- **F1-Score (Weighted):**
  A balanced metric combining Precision and Recall, calculated as their harmonic mean. It is especially valuable in cases of class imbalance, as it accounts for both false positives and false negatives.

- **Confusion Matrix:**
  A tabular representation showing actual versus predicted classifications for each class. It breaks down correct and incorrect predictions, making it easier to understand specific areas where the model struggles.

- **ROC-AUC Score (Macro Average):**
  Measures the model's ability to distinguish between classes by plotting the True Positive Rate against the False Positive Rate at various thresholds. The Area Under the Curve (AUC) represents this ability numerically — higher values indicate better discrimination. In multi-class problems, the Macro Average computes the average AUC score across all classes.

7.3 REGRESSION MODEL EVALUATION METRICS

For the AQI numeric value prediction model, the following regression metrics were used:

- **R^2 Score (Coefficient of Determination):**

  Indicates how well the independent features explain the variability of the target variable. It ranges from 0 to 1 — where values closer to 1 suggest that the model explains most of the variance, and values near 0 indicate poor performance.
- **Mean Absolute Error (MAE):**

  Measures the average absolute difference between the actual and predicted values. It treats all errors equally without penalizing larger errors more, making it intuitive and easy to interpret.

- **Root Mean Squared Error (RMSE):**
  Represents the square root of the average squared differences between predicted and actual values. Unlike MAE, it penalizes larger errors more severely, making it sensitive to outliers. Lower RMSE values indicate a better fitting model.

8. RESULTS AND DISCUSSION

8.1 BEFORE AND AFTER OPTIMIZATION

8.1.1 FOR CLASSIFICATION MODEL

METRIC	BEFORE TUNING AND STACKING	AFTER TUNING AND STACKING
ACCURACY	83.65%	87.43%
Precision(Weighted)	83.70%	87.42%
Recall(Weighted)	83.60%	87.41%
F1- Score(Weighted)	83.65%	87.41%
ROC-AUC (Macro Avg)	90.21%	94.03%

Classification Model:

- Initially with base models and default parameters (XGBoost, LGBM, CatBoost individually) you were getting around 83.6% accuracy.
- ROC-AUC was already decent around 90.21% thanks to tree-based models.
- After Optuna tuning, adding feature interactions (PM2.5/PM10, NO/NO2) and integrating Stacking with CatBoost as meta-classifier — your performance jumped to 87.43% accuracy and 94.03% ROC-AUC.

8.1.2 FOR REGRESSION MODEL

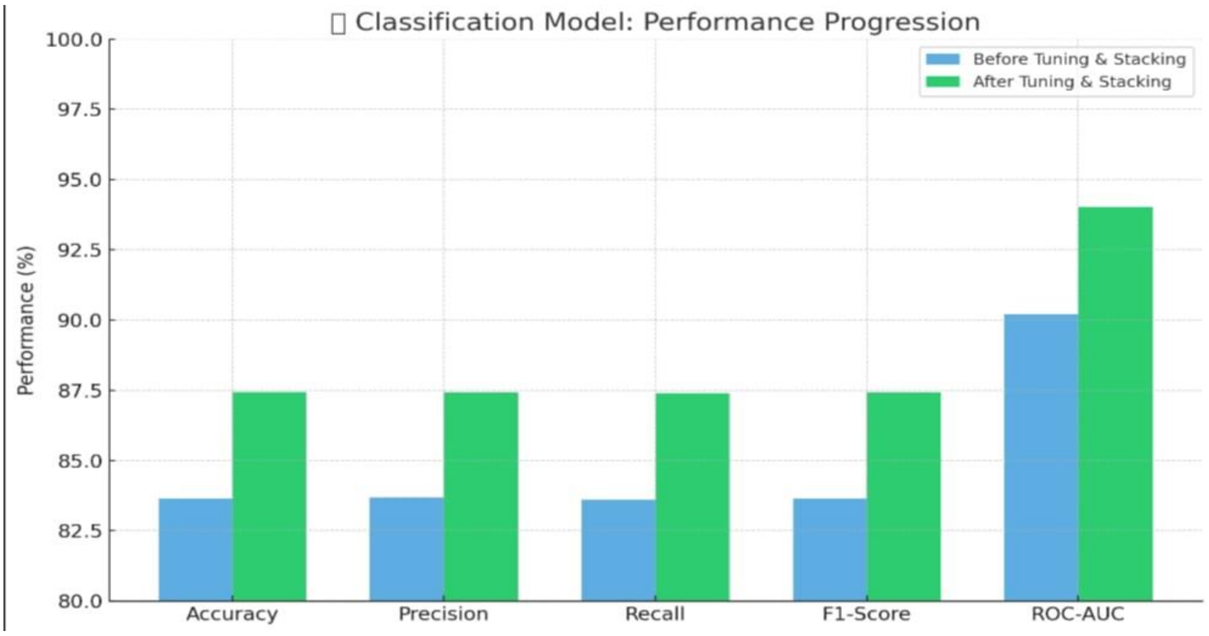
METRIC	VALUE

R^2 SCORE	0.9221
MAE	10.36
RMSE	13.42

REGRESSION MODEL:
The Stacking Regressor delivered a high R² of 0.9221, with a MAE of 10.36 and RMSE of 13.42, indicating precise AQI value predictions with minimal error.

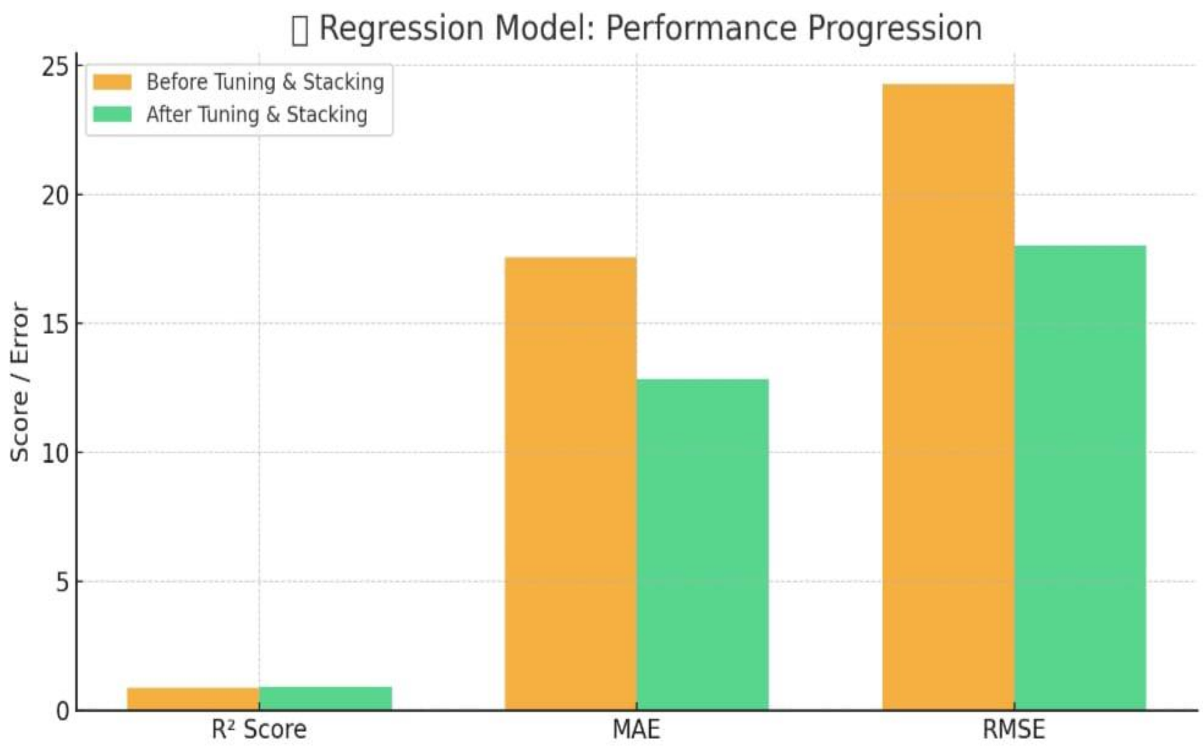
8.2 GRAPHS PLOTTING THE RESULTS

8.2.1 CLASSIFICATION MODEL



Classification Chart: shows how each metric (Accuracy, Precision, Recall, F1-Score, and ROC-AUC) improved after hyperparameter tuning and stacking.

8.2.2 REGRESSION MODEL



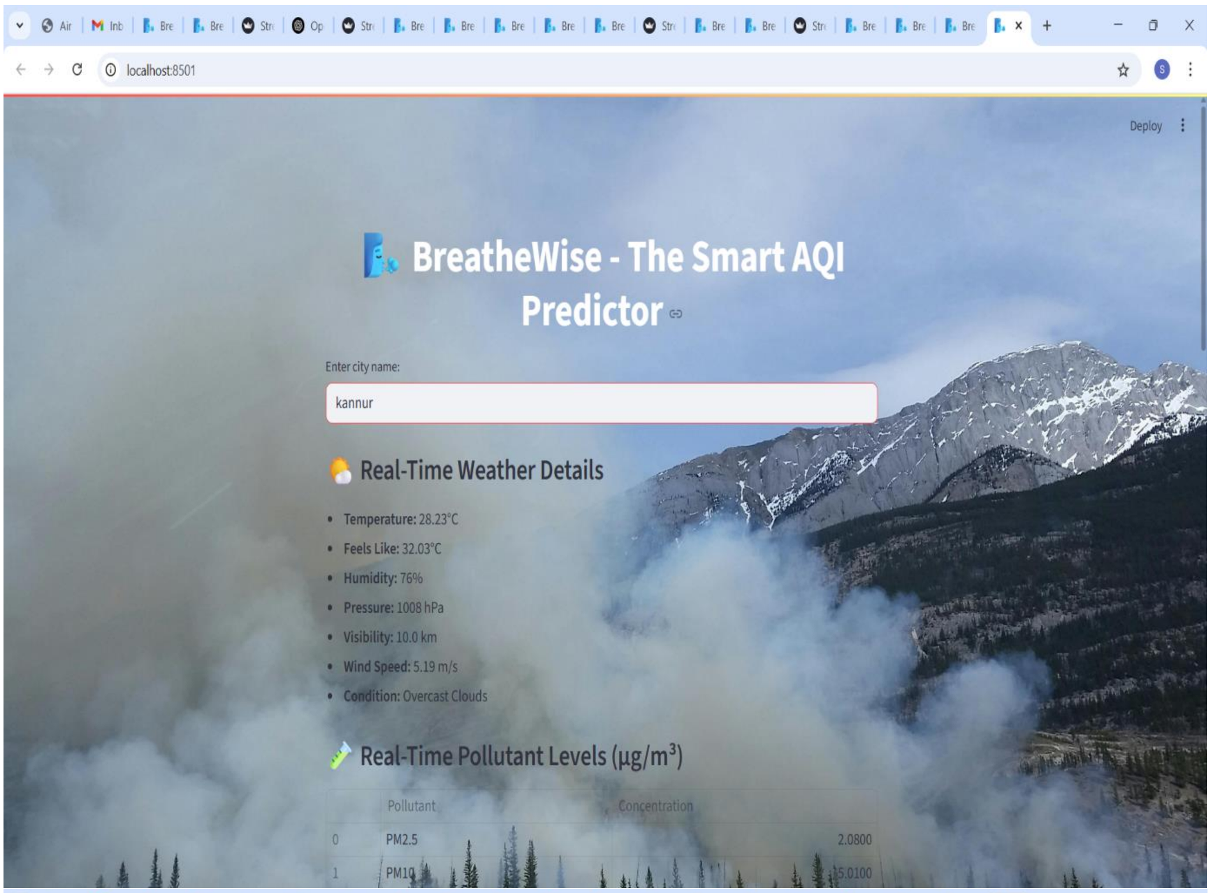
Regression Chart: compares R², MAE, and RMSE before and after tuning + stacking — with higher R² and lower MAE/RMSE reflecting better performance.

9. CONCLUSION

The final outcome of this project is a fully functional, interactive real-time AQI prediction application titled BreatheWise - The Smart AQI Predictor. The system seamlessly integrates real-time environmental data from the OpenWeather API and processes it through optimized, carefully tuned machine learning models, including a stacked regression ensemble and a stacked classification model. The application predicts the real-time AQI index value for any given city, identifies its corresponding AQI category based on trained models, and fetches the official OpenWeather AQI category for direct, side-by-side comparison. Users can also view live weather details and pollutant concentration levels through a clean, organized, and visually appealing interface enhanced with interactive ECharts visualizations.

The machine learning pipeline underwent rigorous feature engineering, hyperparameter optimization using Optuna, and model evaluation with metrics such as accuracy, precision, recall, F1-score, and ROC-AUC for classification, alongside R^2, MAE, and RMSE for regression performance. The final stacking ensembles demonstrated strong predictive capability on both balanced and unbalanced datasets. The models are integrated into a Streamlit application, styled with custom imagery and deployed for easy public access, making real-time environmental intelligence available at a glance.

In essence, BreatheWise not only serves as a technical demonstration of real-time air quality prediction using ensemble learning but also functions as a public awareness and decision-support tool for individuals, communities, and policymakers. It bridges the gap between complex environmental data and accessible, actionable insights. The project lays the groundwork for future enhancements like location-based health recommendations, pollutant-specific forecasting, historical AQI trend visualization, and integration with additional air quality data sources for a more comprehensive urban health management platform.



BreatheWise - The Smart AQI Predictor

Enter city name:

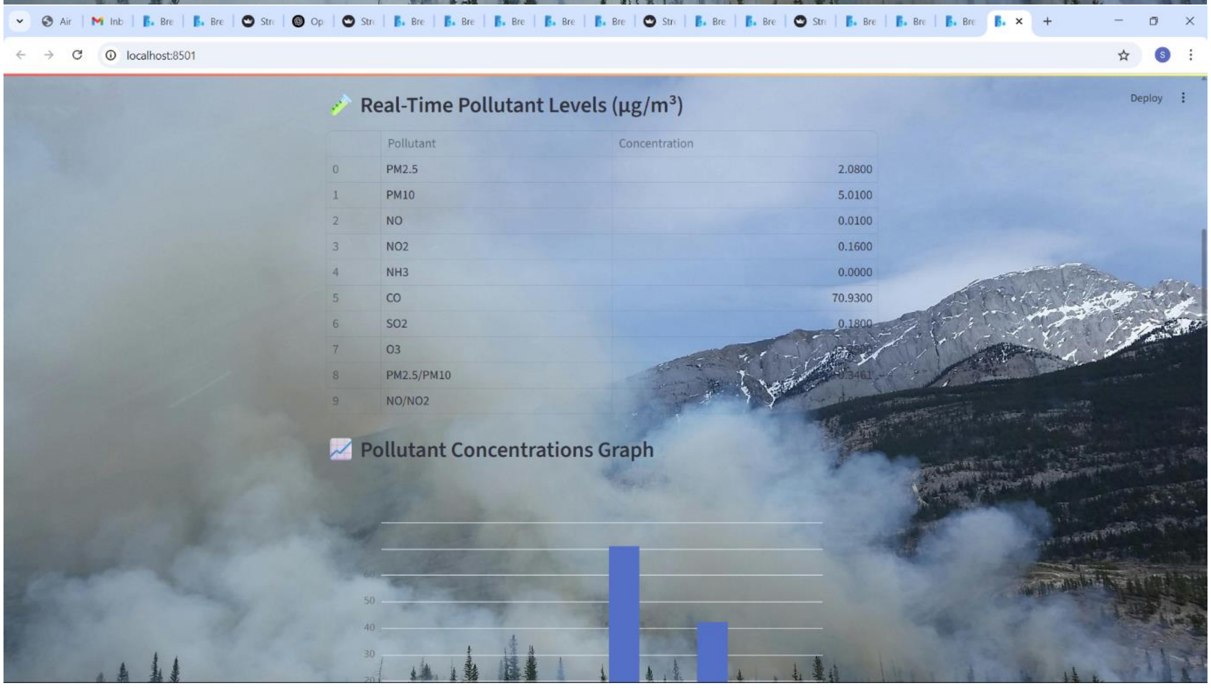
kannur

Real-Time Weather Details

- **Temperature:** 28.23°C
- **Feels Like:** 32.03°C
- **Humidity:** 76%
- **Pressure:** 1008 hPa
- **Visibility:** 10.0 km
- **Wind Speed:** 5.19 m/s
- **Condition:** Overcast Clouds

Real-Time Pollutant Levels (µg/m³)

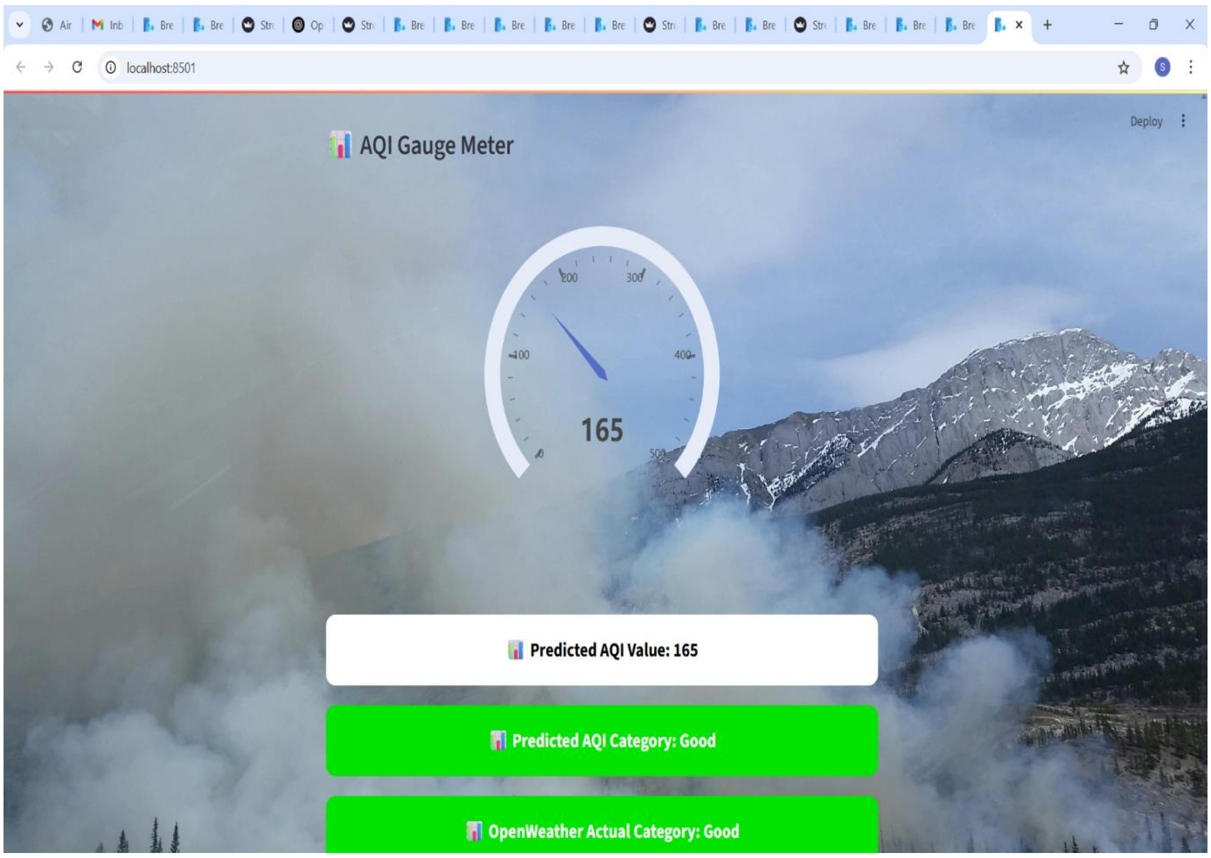
	Pollutant	Concentration
0	PM2.5	2.0800
1	PM10	5.0100



Real-Time Pollutant Levels (µg/m³)

	Pollutant	Concentration
0	PM2.5	2.0800
1	PM10	5.0100
2	NO	0.0100
3	NO2	0.1600
4	NH3	0.0000
5	CO	70.9300
6	SO2	0.1800
7	O3	[illegible]
8	PM2.5/PM10	0.3461
9	NO/NO2	[illegible]

Pollutant Concentrations Graph



AQI Gauge Meter

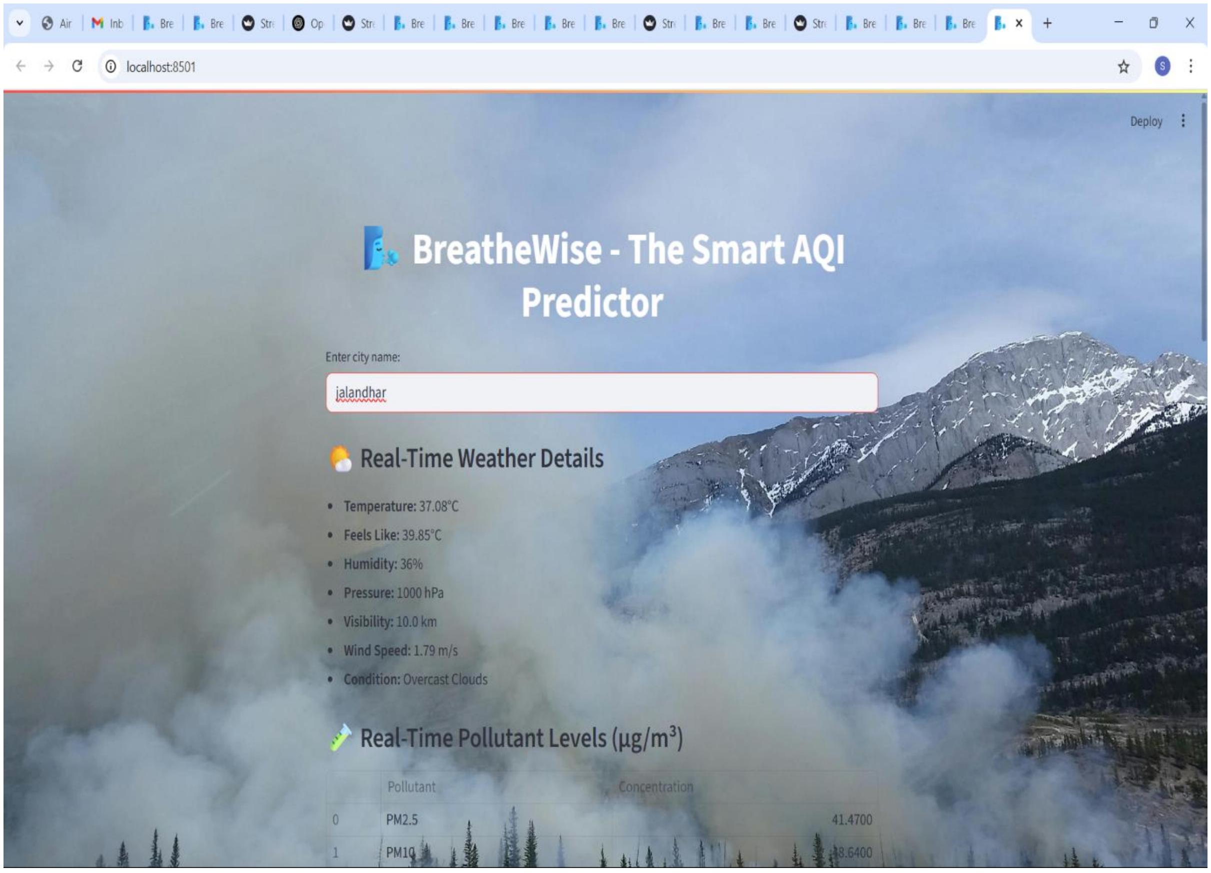
165

Predicted AQI Value: 165

Predicted AQI Category: Good

OpenWeather Actual Category: Good

FINAL VERSION OF THE PROJECT



BreatheWise - The Smart AQI Predictor

Enter city name:

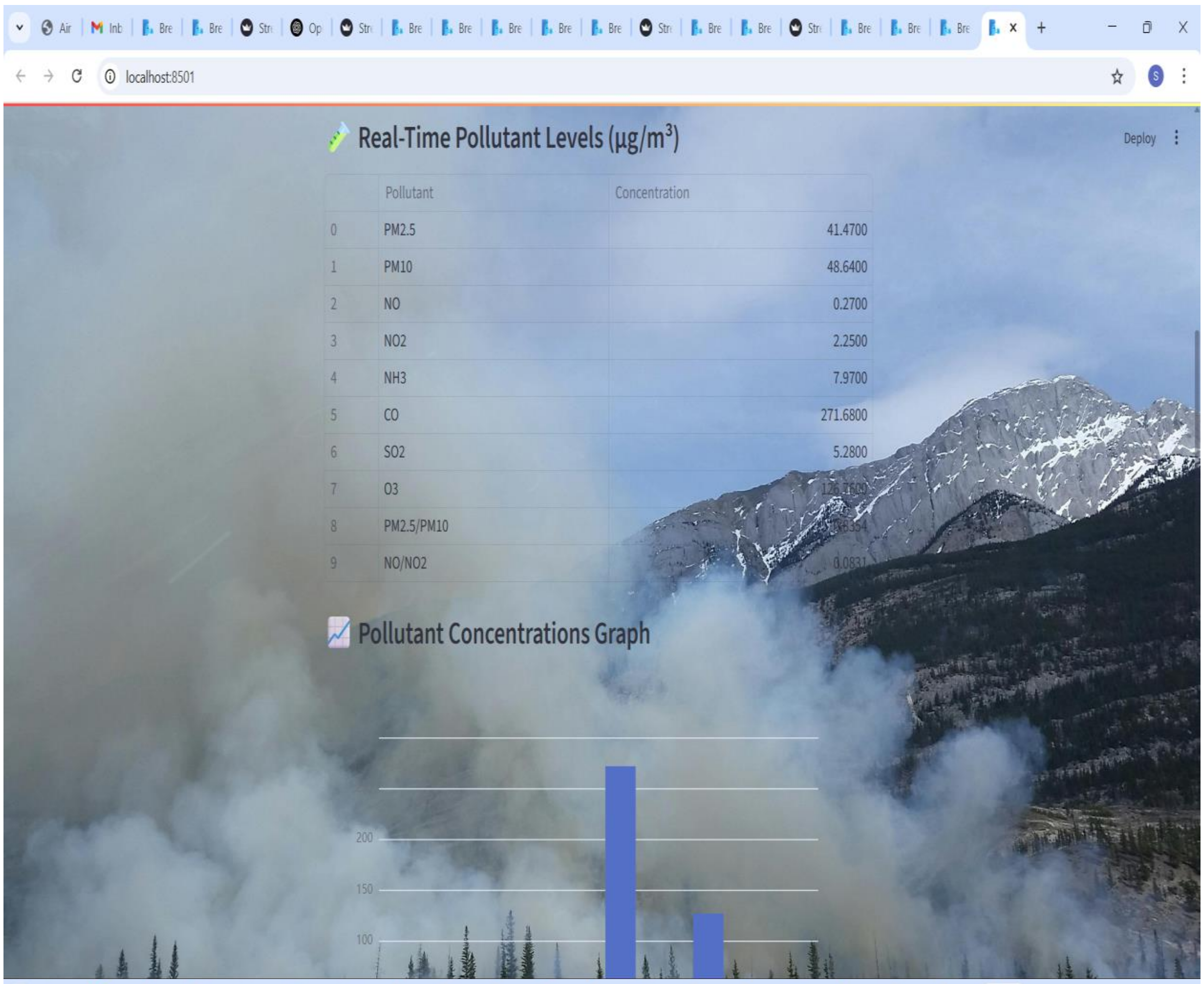
jalandhar

Real-Time Weather Details

- **Temperature:** 37.08°C
- **Feels Like:** 39.85°C
- **Humidity:** 36%
- **Pressure:** 1000 hPa
- **Visibility:** 10.0 km
- **Wind Speed:** 1.79 m/s
- **Condition:** Overcast Clouds

Real-Time Pollutant Levels (µg/m^3)

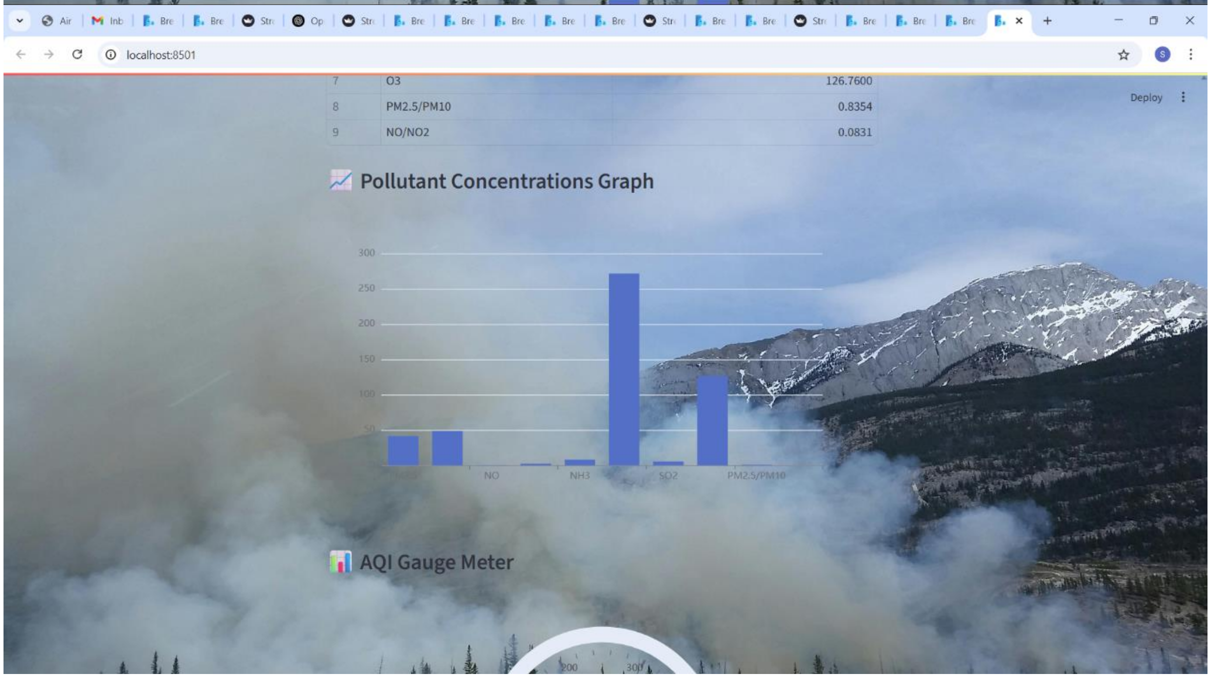
	Pollutant	Concentration
0	PM2.5	41.4700
1	PM10	48.6400



Real-Time Pollutant Levels (µg/m³)

	Pollutant	Concentration
0	PM2.5	41.4700
1	PM10	48.6400
2	NO	0.2700
3	NO2	2.2500
4	NH3	7.9700
5	CO	271.6800
6	SO2	5.2800
7	O3	126.7600
8	PM2.5/PM10	0.8354
9	NO/NO2	0.0831

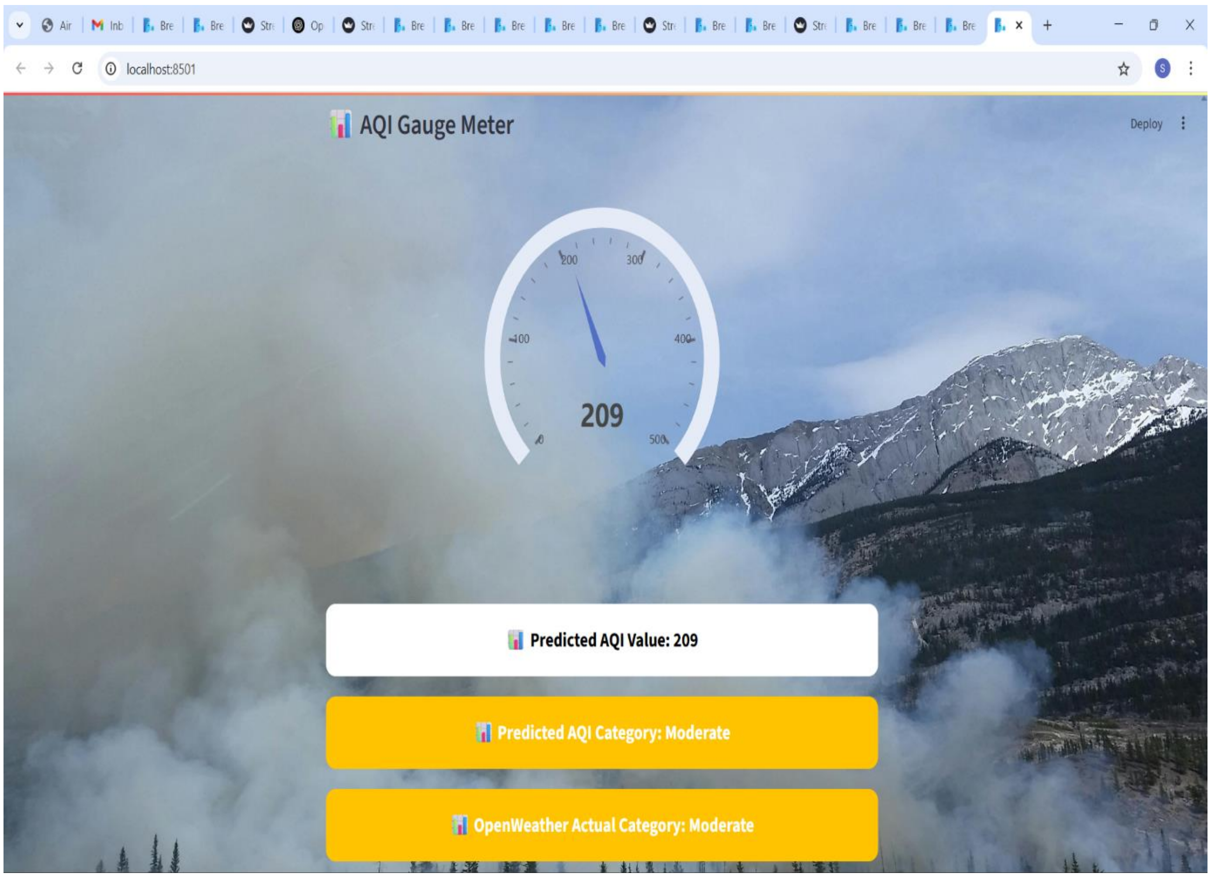
Pollutant Concentrations Graph



	Pollutant	Concentration
7	O3	126.7600
8	PM2.5/PM10	0.8354
9	NO/NO2	0.0831

Pollutant Concentrations Graph

AQI Gauge Meter



Predicted AQI Value: 209

Predicted AQI Category: Moderate

OpenWeather Actual Category: Moderate

10. REFERNCES

1. Central Pollution Control Board (CPCB), Government of India.
   National Air Quality Index (NAQI) standards and guidelines.
   Available at: *https://cpcb.nic.in/*
   Accessed: July 2025.

2. OpenWeather API Documentation.
   API for real-time weather and air pollution data integration.
   Available at: *https://openweathermap.org/api*
   Accessed: July 2025.

3. Bergstra, J., & Bengio, Y. (2012).
   Random search for hyper-parameter optimization.
   Journal of Machine Learning Research, 13(Feb), 281-305.

4. Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019).
   Optuna: A next-generation hyperparameter optimization framework.
   In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 2623-2631). ACM.
   DOI: https://doi.org/10.1145/3292500.3330701

5. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011).
   Scikit-learn: Machine learning in Python.
   Journal of Machine Learning Research, 12, 2825-2830.

6. *Open-source app framework for deploying machine learning and data science applications.*
   Available at: https://docs.streamlit.io/
   Accessed: July 2025.

7. Kaggle: Air Quality Data in India (2015–2020)

Dataset containing real-time and historical AQI readings from multiple Indian cities between 2015 and 2020, sourced from CPCB and official monitoring stations.
Available at: https://www.kaggle.com/datasets/rohanrao/air-quality-data-in-india
Accessed: July 2025.