Air Quality Prediction Using AI and ML: A Case of Indian Cities

Abhinav Mallick

Abstract

Classical machine learning algorithms have shown success in predicting the Air Quality Index (AQI) for Indian cities, but many previous approaches overlook the temporal dynamics of air quality data, leading to overly optimistic results that fail in real-world applications. This paper presents an alternative approach that incorporates lagged pollutant values and employs nested time series cross-validation to provide a more accurate and robust AQI prediction across different cities using various regression models. Boosting algorithms, in particular, demonstrated superior performance, with Gradient Boosting achieving an R^2 score of 94.24% for Delhi. The study concludes that boosted decision trees effectively capture temporal dependencies, showing strong potential for real-life AQI applications as more data becomes available. The code is publicly available for perusal: https://github.com/curemeongithub

1. Introduction

India has long struggled with air quality management, especially in urban centres where rapid industrialization and population growth have led to significant environmental challenges. Air quality measurement in India is overseen by the Central Pollution Control Board (CPCB), which monitors air pollution across the country through various stations equipped to capture real-time data on pollutants like particulate matter (PM2.5 and PM10), nitrogen oxides (NOx), sulphur dioxide (SO2), and others. The Air Quality Index (AQI) is a key metric used to quantify and communicate the state of air quality to the public. It is calculated by considering concentrations of the primary pollutants, including PM2.5, PM10, nitrogen dioxide (NO2), ammonia (NH3), carbon monoxide (CO), sulphur dioxide (SO2), and ozone (O3). The AQI scale ranges from "Good" (0–50) to "Severe" (above 400), with higher values indicating more dangerous levels of air pollution. The CPCB uses a color-coded system to help residents understand the potential health impacts of various AQI levels, which influences public awareness and policy interventions. In recent years, there has been a growing focus on improving the accuracy and scope of AQI predictions in India, especially given the country's ongoing air pollution crises. Traditional models, though useful, often fail to capture the temporal and spatial complexities of air quality data. Therefore, incorporating advanced techniques like machine learning, which can consider these dynamics, is becoming crucial in enhancing the predictive capabilities of AQI monitoring systems.

2. Related Works

Numerous studies have employed machine learning techniques to predict air quality and develop models for forecasting the Air Quality Index (AQI). These efforts have significantly advanced our understanding of how various pollutants influence AQI levels, with different algorithms offering varying degrees of predictive accuracy. However, most of these approaches have not fully addressed the temporal dependencies inherent in air quality data, often leading to overly optimistic results in real-world applications Gupta et al. (2023) explored the use of machine learning techniques like Support Vector Regression (SVR), Random Forest Regression (RFR), and CatBoost Regression (CR) to predict AQI across Indian cities. Their study highlighted that RF provided the best performance, particularly when combined with data balancing techniques such as SMOTE. While this work underscored the importance of feature

balancing and optimization, it did not account for the temporal dynamics of air quality data, limiting the generalizability of the models in real-world settings. Similarly, Aram et al. (2024) compared machine learning models like Random Forest, Gradient Boosting, and Lasso for predicting AQI and Air Quality Grades (AQG). The ensemble approach mitigated issues such as overfitting and ensured a more stable prediction across different cities. This work aligns with the findings from Gupta et al., reinforcing the utility of ensemble methods for improving the accuracy of AQI predictions. Lastly, Natarajan et al. (2024) explored optimized machine learning models for AQI prediction in Indian cities. Their study employed Grey Wolf Optimization (GWO) and Decision Trees, achieving high predictive accuracy in cities like Hyderabad and Visakhapatnam. This research underscores the effectiveness of optimization algorithms in improving the performance of traditional machine learning models and highlights the importance of feature selection in AQI prediction. While these studies have advanced AQI prediction methods, a significant gap remains in addressing the temporal dynamics and dependencies of air quality data. Traditional models, though effective, often ignore the sequential nature of AQI data, resulting in models that may not perform well when predicting future AQI levels based on past values. This paper aims to fill this gap by incorporating lagged pollutant variables into the model, capturing temporal dependencies that are critical in air quality forecasting. Furthermore, it employs a nested time series cross-validation approach, ensuring that models are evaluated under realistic conditions where future data points are predicted based on past information. This methodology not only improves predictive accuracy but also enhances the robustness of the models in real-world applications, addressing the limitations found in previous studies. By focusing on temporal dynamics and utilizing advanced cross-validation techniques, this paper contributes a more realistic and applicable approach to AQI prediction in Indian cities.

3. Materials

3.1 Dataset

The dataset used in the proposed model evaluation is a publicly available Air Quality Data in India (2015–2020) from Kaggle repository. The dataset includes air quality data and air quality index (AQI) data for hourly and daily levels of various stations across major cities of India. The selected cities are Ahmedabad, Aizawl, Amaravati, Amritsar, Bengaluru, Bhopal, Braj Rajnagar, Chandigarh, Chennai, Coimbatore, Delhi, Ernakulam, Gurugram, Guwahati, Hyderabad, Jaipur, Jorapokhar, Kochi, Kolkata, Lucknow, Mumbai, Patna, Shillong, Talcher, Thiruvananthapuram, Visakhapatnam. The attributes in the data for each city are data, month, year, PM2.5, PM10, NO, NO2, NOx, NH3, CO, SO2, O3, Benzene, Toluene, AQI, and AQI_Bucket. The AQI bucket is categorized into six categories that are good, satisfactory, moderate, poor, very poor, and severe.

3.2 Data Cleaning

The calculation of Air Quality Index takes into consideration only PM2.5, PM10, NOx, NH3, CO, SO2, O3. As a result, all the other pollutant's data columns were removed from consideration for inputs into our model. After only keeping the aforementioned pollutants data along with Date, City, AQI and AQI_Bucket, there were still missing values in the data in all columns except Date and City. Since we are interested in retaining as much as data possible without imputation, we manually calculate AQI for days where the data was sufficient for AQI to be calculated according to the eligibility defined by the CPCB. To calculate the AQI manually, we borrowed Rohan Rao's code to assign AQI values wherever sufficient pollutant data was available for the given day. All the days, irrespective of city, which didn't have sufficient data for the AQI to be calculated was removed and the AQI bucket was also assigned according to the AQI Value for that day. Lastly, the five cities' data, namely, **Delhi**, **Guwahati**, **Hyderabad**, **Kolkata** and **Visakhapatnam** were extracted to be used for our analysis.

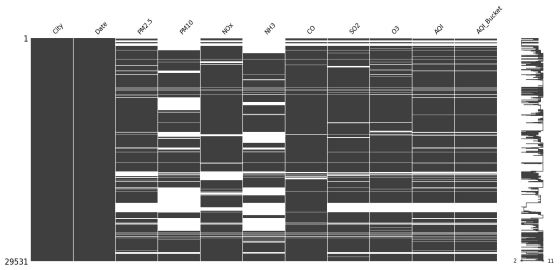


Figure 1. Missing data visualization pre data-preprocessing

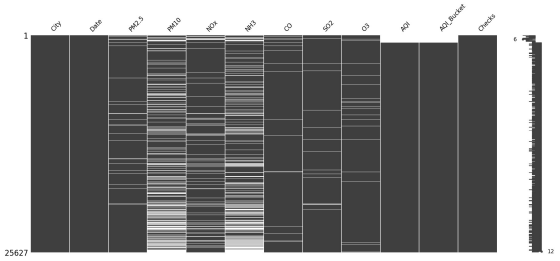


Figure 2. Missing data visualization post data-preprocessing

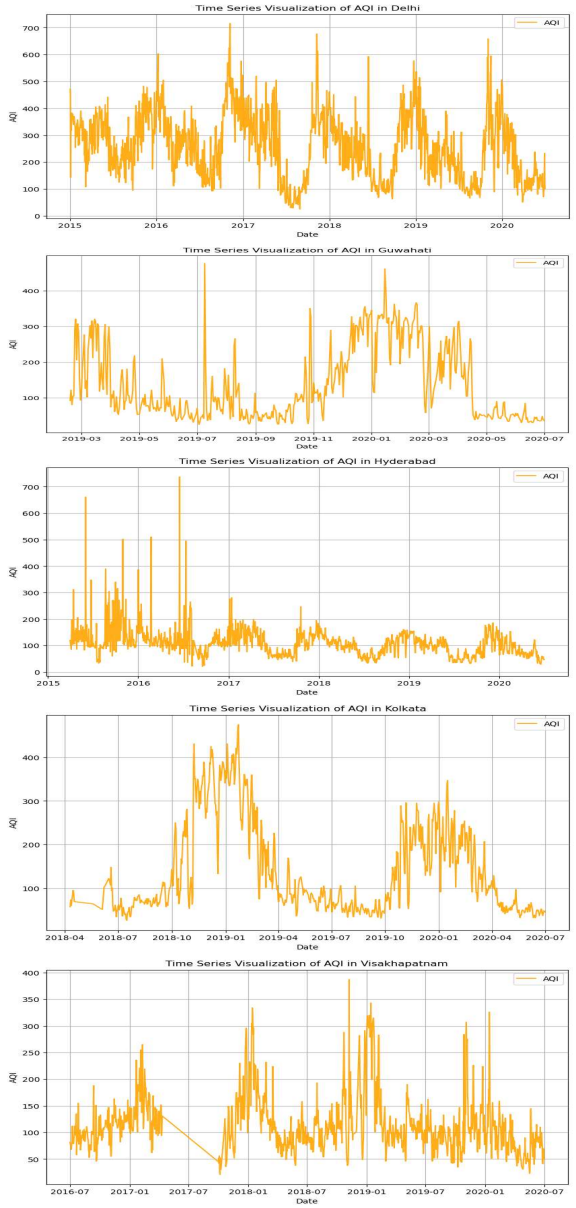


Figure 3. AQI visualization across cities

4. Methods

4.1 Data Imputation

In our approach we plan to employ Regression algorithms that cannot handle null values in their input data because they rely on complete information to identify patterns and make accurate predictions. Therefore, missing values need to be addressed before applying regression models, typically through a process known as data imputation. While employing data imputation it's necessary to make sure that the *statistical integrity of the data is preserved* which generally means that the imputed values should not shift the distribution of the data as it can lead to biased or inaccurate results. Equally important is retaining correlations and relationships between features, if these relationships are violated the model's predictions may be skewed. Therefore, before imputing we first examine how much pollutants' data is missing in each city.

Table 1. Missing Pollutant Data across cities

City Name	Pollutant Name	Percentage Missing (%)
Hyderabad	NH3	13.10
Hyderabad	PM10	12.78
Delhi	SO2	5.38
Delhi	O3	4.08
Delhi	PM10	3.73
Kolkata	SO2	0.92
Visakhapatnam	PM10	0.88
Visakhapatnam	NH3	0.72
Kolkata	O3	0.52
Visakhapatnam	PM2.5	0.40
Visakhapatnam	O3	0.40
Delhi	NH3	0.34
Visakhapatnam	SO2	0.24
Guwahati	PM2.5	0.19
Visakhapatnam	CO	0.16
Kolkata	NOx	0.13
Hyderabad	PM2.5	0.10
Hyderabad	NOx	0.10
Visakhapatnam	NOx	0.08

Table 2. Dataset Size across cities

City Name	Start Date	End Date	Number of Days
Delhi	2015-01-01	2020-07-01	2007
Guwahati	2019-02-17	2020-07-01	497
Hyderabad	2015-03-31	2020-07-01	1893
Kolkata	2018-04-10	2020-07-01	759
Visakhapatnam	2016-07-01	2020-07-01	1237

We employ two commonly used imputation techniques for Time Series data as given below:

4.1.1 Linear Interpolation Imputation

Gaps in pollutant measurements often occur due to sensor malfunctions or maintenance. Linear interpolation estimates missing values by connecting adjacent known data points with straight lines, effectively capturing the temporal continuity and trends inherent in pollutant levels. This method is simple, efficient, and preserves the inherent patterns in the data without introducing complex biases.

4.1.2 Backward Fill

This method fills missing values by propagating the next valid observation backward to replace any gaps. It is especially effective in scenarios where pollutant levels remain relatively stable over short periods or when sudden changes are captured in subsequent readings. We know examine what effect does the imputation have on your datasets.

Table 3. Change in correlation between features pre and post imputation

City	Difference in Correlation Matrix								
Delhi		PM2.5	PM10	NOx	NH3	CO	SO2	O3	AQI
	PM2.5	0.00	-0.41	0.00	-0.01	0.00	1.35	2.68	0.00
	PM10	-0.41	0.00	0.77	0.25	1.76	1.70	0.93	-1.52
	NOx	0.00	0.77	0.00	-0.03	0.00	2.50	2.48	0.00
	NH3	-0.01	0.25	-0.03	0.00	-0.03	2.28	1.56	-0.02
	CO	0.00	1.76	0.00	-0.03	0.00	3.00	3.06	0.00
	SO2	1.35	1.70	2.50	2.28	3.00	0.00	2.37	1.41
	O3	2.68	0.93	2.48	1.56	3.06	2.37	0.00	3.69
	AQI	0.00	-1.52	0.00	-0.02	0.00	1.41	3.69	0.00
Guwahati		PM2.5	PM10	NOx	NH3	CO	SO2	O3	AQI
	PM2.5	0.00	0.14	0.28	0.03	0.12	-0.13	-0.18	-0.05
	PM10	0.14	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	NOx	0.28	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	NH3	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	CO	0.12	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	SO2	-0.13	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	O3	-0.18	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	AQI	-0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Hyderabad		PM2.5	PM10	NOx	NH3	CO	SO2	O3	AQI
	PM2.5	0.00	-15.43	-0.04	-1.84	0.01	-0.03	0.02	0.00
	PM10	-15.43	0.00	-3.19	-2.23	-20.34	0.37	0.67	-17.43
	NOx	-0.04	-3.19	0.00	0.76	-0.01	0.00	-0.02	0.00
	NH3	-1.84	-2.23	0.76	0.00	1.27	-5.28	-8.28	-0.97
	CO	0.01	-20.34	-0.01	1.27	0.00	0.00	0.00	0.00
	SO2	-0.03	0.37	0.00	-5.28	0.00	0.00	0.00	0.00
	O3	0.02	0.67	-0.02	-8.28	0.00	0.00	0.00	0.00
	AQI	0.00	-17.43	0.00	-0.97	0.00	0.00	0.00	0.00

		PM2.5	PM10	NOx	NH3	CO	SO2	O3	AQI
Kolkata	PM2.5	0.00	0.00	0.00	0.00	0.00	-1.64	-0.14	0.00
	PM10	0.00	0.00	0.00	0.00	0.00	-1.43	-0.07	0.00
	NOx	0.00	0.00	0.00	-0.02	-0.03	-2.21	-0.14	-0.01
	NH3	0.00	0.00	-0.02	0.00	0.00	-2.20	-0.39	0.00
	CO	0.00	0.00	-0.03	0.00	0.00	-2.01	0.21	0.00
	SO2	-1.64	-1.43	-2.21	-2.20	-2.01	0.00	-1.27	-1.64
	O3	-0.14	-0.07	-0.14	-0.39	0.21	-1.27	0.00	-0.14
	AQI	0.00	0.00	-0.01	0.00	0.00	-1.64	-0.14	0.00
		PM2.5	PM10	NOx	NH3	CO	SO2	O3	AQI
Visakhapatnam	PM2.5	0.00	-0.13	-0.35	-0.11	-0.19	-0.32	0.01	0.00
	PM10	-0.13	0.00	-0.46	-0.08	0.07	-0.35	0.24	-0.12
	NOx	-0.35	-0.46	0.00	0.55	0.04	-0.12	0.12	-0.02
	NH3	-0.11	-0.08	0.55	0.00	-0.48	-0.01	-0.97	-0.14
	CO	-0.19	0.07	0.04	-0.48	0.00	0.07	-0.09	0.06
	SO2	-0.32	-0.35	-0.12	-0.01	0.07	0.00	0.09	-0.06
	O3	0.01	0.24	0.12	-0.97	-0.09	0.09	0.00	-0.02
	AQI	0.00	-0.12	-0.02	-0.14	0.06	-0.06	-0.02	0.00

It's clearly visible that there is no significant change to the statistical structure of the data except significant difference in Hyderabad's PM2.5 data.

4.2 Feature Engineering

Lagged Variables refer to adding past values of features as new features in your model. Simply meaning that instead of using only today's PM2.5 levels to predict today's AQI, you could include PM2.5 from the previous day or even previous weeks as additional predictors. Air quality tends to be heavily influenced by temporal factors like weather, seasonal patterns, and pollution events. Therefore, it's useful to add Lagged Variables is due to three sequentially complementary reasons. Captures Temporal Dependencies: Since pollutant levels (and therefore AQI) often depend on past values, adding lagged variables captures these relationships, which are crucial in time-series data. Improves Predictive Power: It allows our model to make more informed predictions by understanding trends and patterns over time. Reflects Real-World Scenarios**:** In practice, AQI is influenced by both short-term and long-term pollution trends and including lagged variables makes our model more reflective of real-world processes. Therefore, *we added (t-1) lagged variables for our pollutants* where t-1 lagged variable represents the value of a given feature (e.g., PM2.5 or CO) from one time step prior to the current observation. In this context, t-1 means "the value from the previous day".

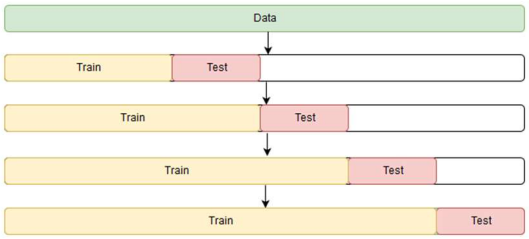
We chose to not include the AQI lagged variable as we expected that the AQI to be predicted for a given day might be highly correlated to the (t-1) AQI value of that day which leads to the issue of multicollinearity in our model something that must be avoided. Apart from this, *we also tested adding (t-1) AQI variable to our data but witnessed an insignificant minor improvement* which further cemented our reasons to not add it our model. Due to similar reasons, we chose to not add any more lagged variables such as (t-3) or (t-7). We shall review in the Results section how much

improvement we had in our accuracy due to addition of lagged inputs.

4.3 Time Series Cross Validation (Outer Loop)

Traditional cross-validation techniques assume that observations are independent of each other, which is not the case with time-series data, where past values influence future ones. To prevent data leakage and ensure that the model is evaluated under real-world conditions, we used TimeSeriesSplit from Scikit-learn library for cross-validation. This method respects the temporal order of the data, ensuring that the model is always trained on past data and validated on future data. We applied a 5-fold* TimeSeriesSplit to train and evaluate the model, progressively increasing the amount of training data in each fold while using later data for validation. By doing so, the model was tested on unseen data from future periods, providing a more realistic assessment of its predictive power. *For cities with smaller datasets (fewer than 1000 rows), we opted to use 10-fold TimeSeriesSplit instead of a 5-fold.* Given the limited data available, 10-fold cross-validation was employed to maximize the use of the dataset for both training and validation while avoiding overfitting. This approach ensures that the model is trained on a larger portion of the data, with each fold providing a new combination of training and validation sets. By doing so, we prevent the model from overfitting to small training sets and gain a more reliable estimate of performance. Incorporating this method ensured that our results were more robust and prevented the model from overfitting to future data points, a common issue when time dependencies are ignored. The use of lagged variables (e.g., PM2.5 lagged by one day) was carefully managed within this cross-validation framework to avoid introducing future information into the training sets, thereby maintaining the integrity of the time-series structure.

Figure 4. Illustration of Time Series Cross Validation (Expanding Window)



4.4 Model Selection

4.4.1 Linear Regression (Baseline Model)

Using Linear Regression as a baseline model for AQI prediction was the obvious choice due to its simplicity, interpretability, and effectiveness in providing a reference for more advanced models. Linear Regression works by establishing a linear relationship between input features (e.g., PM2.5, PM10) and the target variable (AQI), fitting a line that minimizes the difference between the predicted and actual values. As a baseline, it helps assess whether more complex models capture non-linear relationships and interactions effectively. Additionally, it serves as a good benchmark for detecting overfitting and evaluating whether increased model complexity genuinely improves performance over this simple, interpretable model.

4.4.2 Linear Regression with L1 (Lasso) Regularization

Lasso Regression was included in this study because it helps in feature selection by adding a penalty for the absolute size of coefficients, effectively shrinking less important feature weights to zero. In the context of AQI prediction, where there are multiple input features (like pollutants and their lagged values), Lasso can simplify the model by automatically selecting the most important predictors, reducing overfitting. This regularization technique can improve the model's generalizability, especially when dealing with a large number of potentially correlated features.

4.4.3 Linear Regression with L2 (Ridge) Regularization

Ridge Regression addresses multicollinearity by applying a penalty on the size of the coefficients, but unlike Lasso, it doesn't shrink them to zero. This means Ridge can still include all features in the model while reducing the impact of those that are highly correlated. In the context of AQI prediction, where pollutants and lagged variables may be correlated, Ridge helps stabilize the model, making it more robust and reducing overfitting, ultimately improving generalization to unseen data.

4.4.4 Decision Tree Regressor

Decision Tree Regressor (DTR) is capable of handling both non-linear relationships and interaction effects between features. It works by recursively splitting the dataset into smaller subsets based on feature values, creating a tree-like structure where each node represents a decision based on a feature threshold, and the leaves represent the predicted output (in this case, AQI). This allows it to model complex patterns in the data, making it particularly useful for capturing intricate relationships between pollutants and their lagged variables. Unlike linear models, DTs don't assume a linear relationship between the inputs and the target, making them better suited for datasets with non-linear dynamics. Additionally, they don't require feature scaling, can handle multicollinearity effectively, and provide clear interpretability by showing how different features lead to specific predictions.

4.4.5 XGBoost Regressor

XGBoost (XGB) Regressor is a powerful model because it builds on the strengths of decision trees through gradient boosting, where multiple trees are sequentially added to correct the errors of the previous ones. XGBoost works by creating an ensemble of decision trees, with each tree focused on minimizing the errors made by the previous trees, making it highly effective at capturing complex relationships in the data. It handles both non-linear interactions and feature importance while being robust to overfitting through techniques like regularization and early stopping. XGBoost also excels with large datasets and can efficiently handle missing data, making it particularly suited for AQI prediction with various pollutants and lagged variables. Its ability to optimize performance through parallel computation also makes it faster and more scalable compared to other models.

4.4.6 Gradient Boosting Regressor

Gradient Boosting Regressor (GBR) also combines the strengths of multiple decision trees by building them sequentially to correct the errors of the previous trees. GBR is effective at capturing non-linear relationships and interactions between features like pollutant levels and their lagged variables. It works by iteratively improving the model, with each tree trained to reduce the residual errors from the previous ones, making it highly robust for predictive tasks such as AQI estimation. GBR also includes regularization techniques to prevent overfitting, especially important in complex datasets, and provides high accuracy by focusing on minimizing errors in every iteration. Additionally, its flexibility to handle different loss functions makes it adaptable to various predictive challenges.

4.5 Hyperparameter Tuning (Inner Loop)

To optimize the performance of the machine learning models, a comprehensive hyperparameter tuning process was employed. This process was carried out using an inner cross-validation loop facilitated by GridSearchCV, which allowed for the systematic exploration of various hyperparameter configurations. In particular, for each model, a set of hyperparameters was defined based on its architecture and key characteristics. The inner loop utilized 3-fold TimeSeriesSplit, ensuring that each combination of hyperparameters was evaluated across multiple temporal folds. This method ensures that the data was *split in a way that respects the temporal dependencies of the time-series data, thus preventing data leakage* from

future to past data points. The evaluation criterion used during this inner loop was negative mean squared error (neg_MSE), as minimizing prediction error was the primary objective. This tuning process was applied across all models to ensure that *each algorithm was operating under its optimal hyperparameters before proceeding to the outer cross-validation loop for final model evaluation.* By using an inner loop for hyperparameter tuning, this approach minimized the risk of overfitting to the training data, thus ensuring that the models could generalize more effectively to unseen data. This method also allowed for a fair and standardized comparison of the performance of the different algorithms, each tuned to its optimal configuration for the dataset used in this study. The results of the hyperparameter tuning were critical in ensuring that the models were properly configured to capture the underlying patterns in the data, resulting in improved model accuracy and robustness.

4.6 Nested Cross Validation for Model Evaluation and Hyperparameter Tuning

To ensure robust model evaluation and prevent overfitting, nested cross-validation was employed in this study for both model evaluation and hyperparameter tuning. *This method combines an inner cross-validation loop for hyperparameter optimization and an outer cross-validation loop for model evaluation.* The inner loop used GridSearchCV to systematically explore different combinations of hyperparameters and identify the optimal configuration for each model. Each fold of the outer loop then evaluated the performance of the model, trained with its best-tuned hyperparameters, on unseen validation data. *By separating the tuning and evaluation processes, nested cross-validation provides an unbiased estimate of model performance and ensures that the results are generalizable to new, unseen data*.

This approach was applied uniformly across all models, including Gradient Boosting Regressor, XGBoost, and Decision Tree Regressor, ensuring a fair comparison between the models and their configurations. By utilizing this rigorous evaluation framework, the study was able to mitigate the risk of overfitting while maximizing model performance.

4.7 Evaluation Metrics

The metrics used in the proposed work are as follows:

4.7.1 Root Mean Squared Error

RMSE indicates how densely the data are distributed along the line of best fit. RMSE values in the range of 0.2–0.5 demonstrate that the model can reasonably predict the data.

4.7.2 Mean Squared Error

MSE is a parameter that measures how closely a fitted line resembles a set of data points. The lower the value, the closer it is to the line, and hence the better. If the MSE value = 0, the model is perfect.

4.7.3 Mean Absolute Error

MAE evaluates the absolute distance of the observations to the predictions on the regression line.

4.7.4 R-Squared

R-Squared indicates to what extent the regression model is in line with the observed data. A higher R square value denotes a better model fit.

5. Results and Discussion

5.1 Results

5.1.1 Fold Wise Best Model Performances across cities in comparison to Lagged Variables

Table 4. Delhi Fold Wise Model Performance for Linear Regression

	Without Lagged Variables				With Lagged Variables			
Fold	**RMSE**	**MSE**	**MAE**	**R^2 Score**	**RMSE**	**MSE**	**MAE**	**R^2 Score**
1	51.86	2689	40.64	77.53	38.89	1512	29.56	89.95
2	73.54	5409	60.81	76.80	40.40	1632	30.24	88.42
3	38.11	1452	28.10	88.73	35.77	1279	27.23	88.97
4	36.98	1367	30.18	89.06	44.60	1989	30.98	86.29
5	38.71	1498	32.48	89.66	35.31	1247	26.56	91.91
Average	**47.84**	**2483**	**38.44**	**84.36**	**39.00**	**1532**	**28.91**	**89.11**

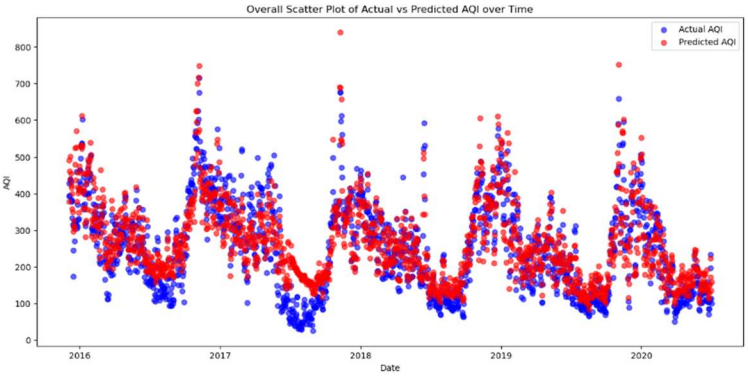


Figure 5: Scatter Plot of Actual vs Predicted AQI by Linear Regression.

Figure 6: Scatter Plot of Actual vs Predicted AQI by GradientBoosting Regressor

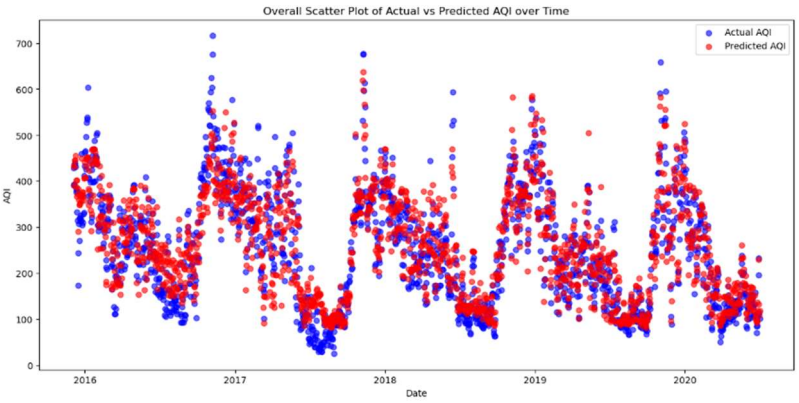


Table 5. Delhi Fold Wise Model Performance for GradientBoosting Regressor

	Without Lagged Variables				With Lagged Variables			
Fold	**RMSE**	**MSE**	**MAE**	**R^2 Score**	**RMSE**	**MSE**	**MAE**	**R^2 Score**
1	50.95	2595.60	39.60	78.32%	29.96	897.73	22.17	94.04%
2	61.84	3824.13	47.77	83.60%	28.61	818.80	20.11	94.20%
3	37.42	1400.20	28.59	89.14%	28.48	811.01	20.01	93.02%
4	32.18	1035.67	23.90	91.72%	29.76	885.63	20.71	93.90%
5	26.89	723.26	20.70	95.01%	24.75	612.44	17.20	96.03%
Average	**41.86**	**1915.77**	**32.11**	**87.56%**	**28.31**	**805.12**	**20.04**	**94.24%**

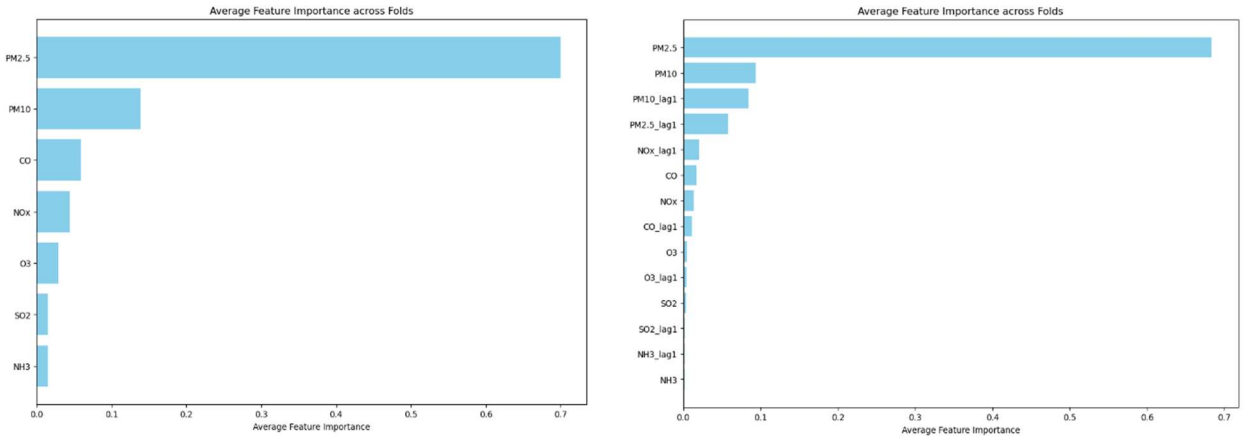


Figure 7: (Left) Feature Importance from GBR without Lagged Variables. (Right) Feature Importance from GBR with Lagged Variables

Table 6. Guwahati Fold Wise Model Performance for GradientBoosting Regressor

	Without Lagged Variables				With Lagged Variables			
Fold	**RMSE**	**MSE**	**MAE**	**R² Score**	**RMSE**	**MSE**	**MAE**	**R² Score**
1	123.10	15154.40	38.64	36.80%	23.87	570.01	18.31	63.33%
2	104.85	10993.90	32.34	22.08%	24.24	587.69	19.89	43.85%
3	108.78	11833.46	55.41	52.29%	48.00	2304.45	28.63	62.24%
4	16.14	260.57	11.43	-0.73%	9.16	83.96	6.67	67.54%
5	48.27	2330.48	27.92	52.62%	36.47	1329.79	19.82	72.97%
6	50.11	2511.03	40.09	55.90%	31.65	1001.98	25.39	82.40%
7	36.86	1358.44	27.59	62.17%	31.20	973.16	21.76	72.90%
8	38.31	1467.90	30.71	73.42%	33.06	1092.66	24.31	80.21%
9	28.14	791.61	18.51	82.87%	17.82	317.73	12.51	93.12%
10	9.70	94.02	6.86	50.71%	8.25	68.07	6.51	64.31%
Average	**56.43**	**4679.58**	**28.95**	**48.81%**	**26.37**	**832.95**	**18.38**	**70.29%**

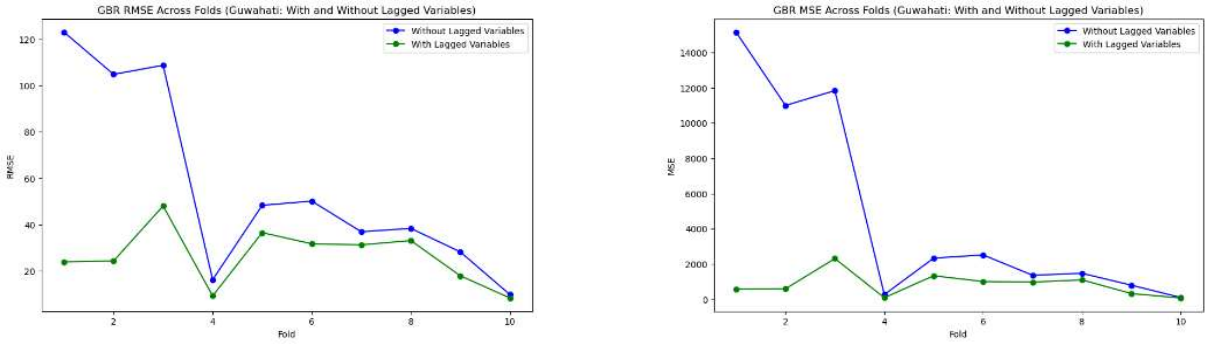


Figure 8: Metric scores of Guwahati visualized across folds

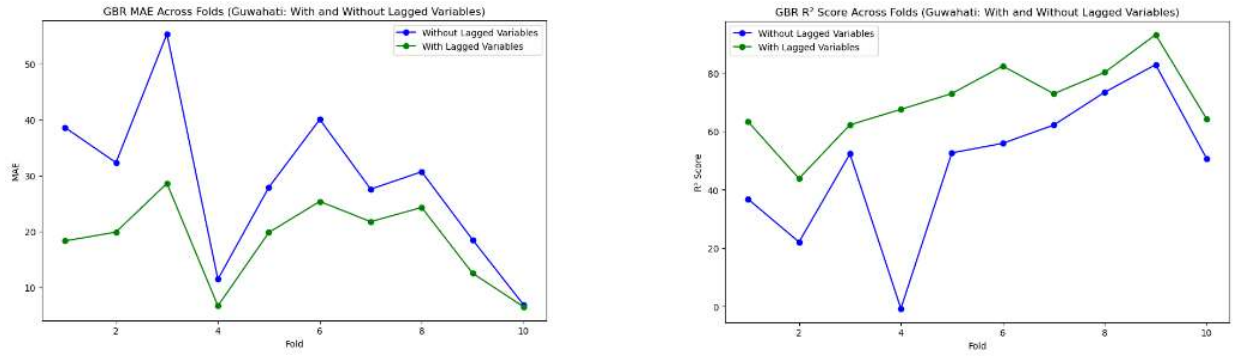


Table 7. Hyderabad Fold Wise Model Performance for GradientBoosting Regressor

	Without Lagged Variables				With Lagged Variables			
Fold	**RMSE**	**MSE**	**MAE**	**R² Score**	**RMSE**	**MSE**	**MAE**	**R² Score**
1	58.01	3365.64	42.12	22.14%	31.83	1013.30	19.92	76.56%
2	26.94	725.99	20.37	56.32%	27.65	764.39	21.19	54.01%
3	17.90	320.43	14.27	72.98%	13.98	195.50	11.16	83.52%
4	12.24	149.83	9.59	87.14%	8.35	69.66	6.67	94.02%
5	14.24	202.78	12.43	84.15%	8.83	78.00	7.00	93.90%
Average	**25.87**	**952.93**	**19.76**	**64.55%**	**18.13**	**424.17**	**13.19**	**80.40%**

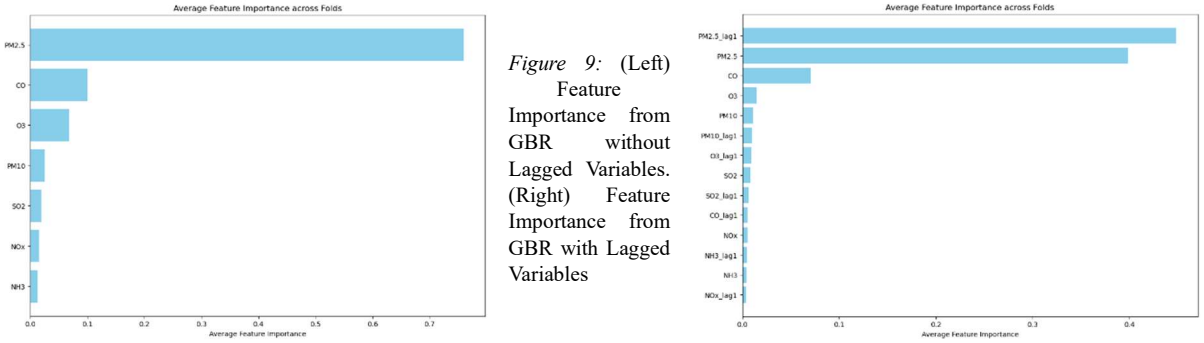


Figure 9: (Left) Feature Importance from GBR without Lagged Variables. (Right) Feature Importance from GBR with Lagged Variables

Table 8. Kolkata Fold Wise Model Performance for Ridge Regression

	Without Lagged Variables				With Lagged Variables			
Fold	**RMSE**	**MSE**	**MAE**	**R² Score**	**RMSE**	**MSE**	**MAE**	**R² Score**
1	13.92	193.68	9.85	89.51%	13.31	177.06	9.60	90.41%
2	88.34	7803.25	75.85	19.18%	81.50	6642.39	68.16	31.21%
3	36.23	1312.88	30.26	83.59%	27.39	750.07	22.53	90.62%
4	26.58	706.72	22.73	65.16%	21.68	470.12	18.23	76.82%
5	14.74	217.17	10.00	24.40%	10.18	103.56	8.01	63.95%
6	8.00	64.01	6.16	57.93%	5.33	28.40	4.28	81.34%
7	26.84	720.23	20.06	86.84%	24.27	589.16	15.72	89.24%
8	25.84	667.72	19.97	77.73%	21.62	467.21	16.99	84.42%
9	15.92	253.30	11.50	91.36%	11.69	136.62	8.23	95.34%
10	5.18	26.80	4.12	77.65%	6.74	45.45	5.18	62.11%
Average	**26.16**	**1196.58**	**21.05**	**67.34%**	**22.37**	**941.00**	**17.69**	**76.54%**

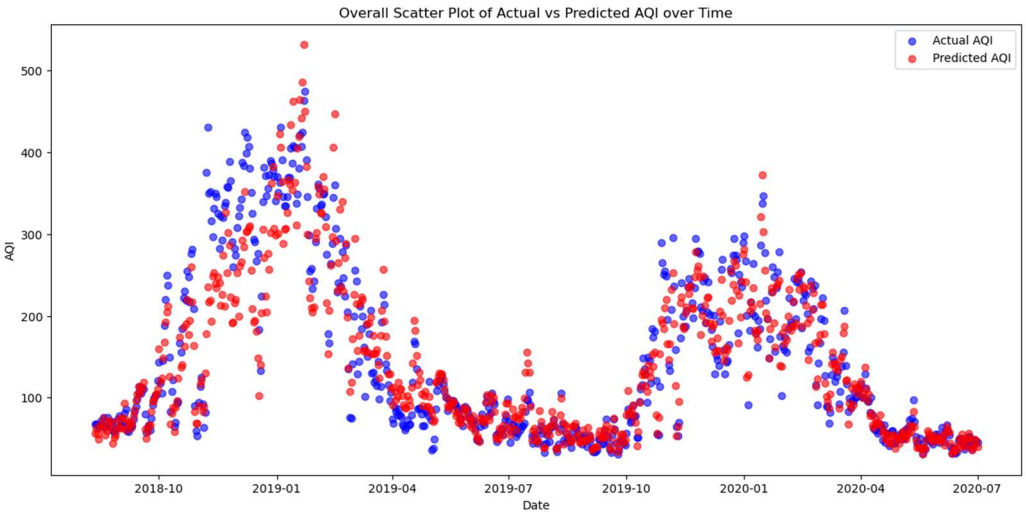


Figure 10: Scatter Plot of Actual vs Predicted AQI by Ridge Regression

Table 9. Visakhapatnam Fold Wise Model Performance for GradientBoosting Regressor

	Without Lagged Variables				With Lagged Variables			
Fold	**RMSE**	**MSE**	**MAE**	**R² Score**	**RMSE**	**MSE**	**MAE**	**R² Score**
1	41.84	1750.45	30.16	50.38%	16.95	287.31	11.35	92.54%
2	17.07	291.45	12.82	57.48%	19.03	362.13	11.88	88.43%
3	25.94	672.99	17.90	88.90%	15.57	242.32	10.47	89.69%
4	24.05	578.18	14.76	67.51%	14.03	196.93	10.37	93.60%
5	16.90	285.45	12.14	81.65%	19.89	395.45	11.85	83.16%
Average	**25.16**	**715.70**	**17.56**	**69.18%**	**17.09**	**296.83**	**11.19**	**89.49%**

5.1.2 Overall model performances across cities

Table 10. R^2 scores across all models and all cities

	Without Lagged Variables					With Lagged Variables				
Model	Delhi	Guwahati	Hyderabad	Visakhapatnam	Kolkata	Delhi	Guwahati	Hyderabad	Visakhapatnam	Kolkata
Linear	84.36%	40.11%	71.52%	69.36%	67.25%	89.11%	39.83%	74.18%	86.02%	76.09%
Ridge	84.36%	44.67%	71.46%	69.86%	67.34%	89.10%	28.36%	74.66%	86.03%	76.54%
Lasso	84.36%	41.23%	71.35%	69.53%	67.12%	89.11%	32.69%	74.34%	86.11%	76.26%
DTR	82.45%	37.80%	59.21%	58.42%	55.05%	88.36%	42.59%	70.26%	81.90%	53.93%
XGB	85.71%	35.50%	69.91%	67.56%	58.85%	94.01%	64.51%	79.10%	89.44%	69.52%
GBR	87.56%	48.81%	64.55%	69.18%	62.08%	94.24%	70.29%	80.40%	89.49%	69.50%
Average	**84.80%**	**41.35%**	**68.00%**	**67.32%**	**62.95%**	**90.66%**	**46.38%**	**75.49%**	**86.50%**	**70.31%**

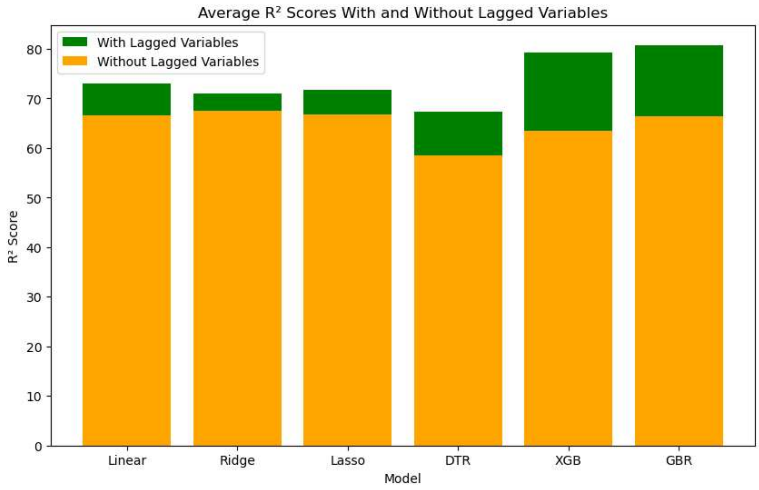


Figure 11. Increase in R^2 score due to Lagged Variables across all models

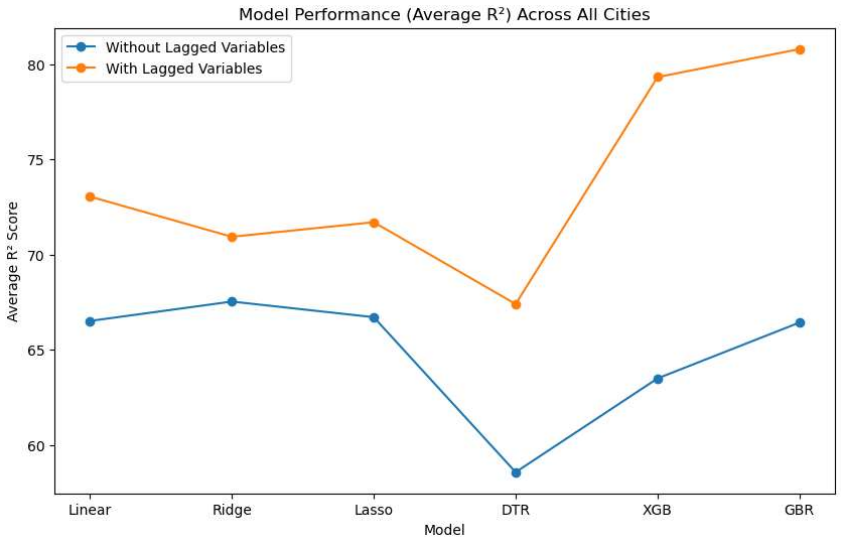


Figure 12. R^2 score visualized across Models with and without Lagged Variables

5.2 Discussions

5.2.1 Delhi

The Linear Regression model improved significantly with the inclusion of lagged variables. The average R^2 score increased from 84.36% (without lagged) to 89.11% (with lagged). The RMSE and MAE metrics also improved, showing better predictive performance with lagged variables. The GradientBoosting Regressor (GBR) achieved an impressive improvement, with the R^2 increasing from 87.56% (without lagged) to 94.24% (with lagged). *This shows that GBR effectively learned from the lagged features and performed robustly in Delhi.*

5.2.2 Guwahati

The overall performance across models was lower without lagged variables, but lagged inputs provided substantial improvements. For GBR, the R^2 increased from 48.81% to 70.29% when lagged variables were included. This highlights the importance of considering temporal features in AQI prediction for Guwahati, which greatly boosted the model's predictive capability.

5.2.3 Hyderabad

Similar improvements were observed in Hyderabad, with GBR showing a marked increase in R^2 from 64.55% (without lagged) to 80.40% (with lagged). Lagged features clearly helped the models better capture air quality variations in this city.

5.2.4 Kolkata

There was an overall improvement in model performance with lagged variables for all models, with GBR showing an R^2 increase from 62.08% to 69.50%. This suggests that including lagged features helps models account for the temporal dependencies in AQI better in Kolkata.

5.2.5 Visakhapatnam

GBR exhibited improved performance here as well, with an R^2 increasing from 69.18% to 89.49% due to lagged variables, making it the best-performing model for this city. The addition of lagged variables in Visakhapatnam resulted in more accurate predictions across all models, especially for GBR and XGB.

5.2.6 R^2 Score Comparison

Across all models and cities, the inclusion of lagged variables consistently improved the R^2 scores. The Gradient Boosting Regressor (GBR) emerged as the best-performing model across most cities. For example, in Delhi, it achieved an R^2 score of 94.24% with lagged variables. XGBoost (XGB) also showed good performance, particularly in Hyderabad and Visakhapatnam, where its R^2 scores increased to 79.10% and 89.44%, respectively, when lagged variables were included.

5.2.7 Model Average Performance

The average R^2 score across all cities for all models improved from 84.80% (without lagged) to 90.66% (with lagged), showing that the inclusion of lagged variables led to a significant overall improvement in AQI prediction. GBR consistently performed better than other models, indicating its effectiveness in capturing temporal dependencies and non-linear relationships in the data.

5.2.8 City-Specific Observations

Delhi and Visakhapatnam had the highest gains in R^2 score with the inclusion of lagged variables, particularly for advanced models like GBR and XGB. Guwahati and Hyderabad saw notable improvements, though their R^2 scores remained lower than Delhi and Visakhapatnam, suggesting the models may have struggled with more complex or noisier data in these cities. However, lagged variables did contribute to improved predictions across all cities.

6. Conclusion and Future Work

This research presents an alternative and robust approach to Air Quality Index (AQI) prediction by incorporating temporal dependencies through the inclusion of lagged pollutant variables. Unlike traditional models that often ignore the time sequence in AQI data, our study

emphasizes the importance of respecting the temporal nature of air quality data. Through the use of a Nested Cross-Validation framework, this paper ensures rigorous and unbiased evaluation of different models, preventing data leakage and overfitting, which are common issues in time-series forecasting.

The introduction of lagged variables (e.g., PM2.5_lag1, PM10_lag1) significantly improved the predictive power of the models across all cities, as demonstrated by the consistent increase in R^2 scores. Notably, models like Gradient Boosting Regressor (GBR) and XGBoost (XGB) capitalized on the temporal information, yielding better performance, with GBR achieving the highest R^2 score of 94.24% for Delhi. The results underscore that incorporating lagged variables better captures the dynamic nature of pollutants and their effect on AQI, making the predictions more reflective of real-world conditions.

Moreover, the study utilized a TimeSeriesSplit method within the cross-validation framework, ensuring that the models trained on past data were evaluated on future unseen data, replicating real-world applications more effectively. The nested hyperparameter tuning, carried out within this time-series structure, further enhanced model performance by optimizing key parameters in a systematic and unbiased manner.

In comparison to conventional approaches that often jumble rows or ignore temporal structures, our model is more rigorous and applicable to real-world AQI forecasting, where time dependencies are critical. The integration of lagged features and time-respecting model evaluations ensures that this method not only yields higher predictive accuracy but also enhances the model's generalizability for real-world applications in urban air quality monitoring and policymaking.

Future work can extend this approach by incorporating additional temporal features, such as weather data or long-term seasonal effects, to further improve model performance and reliability. The field can also explore how exogenous factors—such as economic, social, and quality-of-life indicators—might influence air quality. The aim could be to investigate any correlations that exist which could lead to deeper insights and recommendations for policy interventions. Alongside this exploring Hybrid Models that combine traditional forecasting methods with non-linear techniques such as decision trees and neural networks could also be employed with the goal of capturing any unexplained variance in the residuals that may not have been addressed by the initial forecasting models. The use of advanced ensemble methods or deep learning architectures in a similar framework could also push the boundaries of AQI prediction accuracy. Hence, the author believes there remains a lot of work to be discovered in this sub-field that is at the intersection of Air Quality and Machine Learning.

References

Prediction of Air Quality Index Using Machine Learning Techniques: A Comparative Analysis; https://doi.org/10.1155/2023/4916267

https://www.kaggle.com/code/rohanrao/calculating-aqi-air-quality-index-tutorial

https://www.kaggle.com/datasets/rohanrao/air-quality-data-in-india

Machine learning-based prediction of air quality index and air quality grade: a comparative analysis; https://doi.org/10.1007/s13762-023-05016-2

Optimized machine learning model for air quality index prediction in major cities in India; https://doi.org/10.1038/s41598-024-54807-1

Time series forecasting using a hybrid ARIMA and neural network model by G. Peter Zhang.

https://clip.cpcb.gov.in/index.php/faq/