Programme – BTECH CSE

Institute of Engineering and Technology

Course Code – CS1138

Course Name – Machine Learning

Project Topic - Air Quality Prediction and Classification

Project Report

Section – **B**

Group **– 5**

Team Members -

Aryan Lunawat (2022BTECH021)

Yash Puri Goswami (2022BTECH116)

Aryan Rawtani (2022BTECH020)

Niharika Sharma (2022BTECH063)

Faculty Guide –

Dr. Arpan Gupta

Table of Contents:

1. INTRODUCTION:

Source - Jovian

Mitigating Air Pollution through Machine Learning-based Air Quality Prediction and Classification

Air pollution poses a significant threat to human health and environmental well-being. Exposure to pollutants like Ozone (O3), Particulate Matter (PM2.5/PM10), Carbon Monoxide (CO) and Sulphur Dioxide (SO2) can trigger respiratory problems, heart disease, and even premature death. Accurate and accessible air quality information empowers individuals and authorities to take proactive measures in safeguarding public health and fostering sustainable environmental practices. This project tackles this challenge by leveraging machine learning models for predicting future Air Quality Index (AQI) values and classifying air quality into categories like good, satisfactory, moderate, poor, very poor or severe based on pollutant concentrations. This approach is particularly relevant in urban environments with growing industrial and vehicular activity, where AQI analysis and prediction gaps often exist.

Motivation:

This project is driven by the pressing need to combat air pollution's adverse effects. Accurate prediction of AQI values and their categories are vital for:

- Protecting public health by enabling informed decisions to minimize exposure to pollutants during outdoor activities, especially for sensitive individuals. Policy development and enforcement
- Guiding policy development and enforcement to implement targeted interventions, such as traffic restrictions or promoting public transportation, on high-pollution days.
- Enhancing environmental management by facilitating the adoption of cleaner production processes in industries and promoting green initiatives in urban planning to mitigate pollution.

Applications:

The applications of this project extend far beyond mere prediction. Accurate air quality information can be:

- Disseminated through public health advisories and mobile applications, empowering individuals to take personal precautions.
- Integrated into air quality monitoring systems, providing real-time data for policy decisions.
- Utilized by industries to optimize production processes, minimizing their environmental footprint.

Survey findings:

1. India lacks a consistent system for predicting future air quality, despite having AQI measurement stations since 2014.
2. Existing research focuses on specific cities, leading to biased conclusions.
3. There's a need to enhance the performance of current air quality monitoring systems.

India's air pollution stems from vehicles, industries & crop burning. Industries contribute 50%, vehicles 27%, crop burning 17%, and domestic cooking the remainder. Over 2 million Indians suffer health issues annually, with inadequate enforcement of pollution control laws.

2. PROBLEM STATEMENT:

Combatting Air Pollution in India through Machine Learning-based AQI Prediction and Classification

Air pollution poses a critical threat to public health and environmental well-being in India. This project addresses this challenge by developing a comprehensive machine learning model to analyse historical air quality data for Indian cities from 2015 to 2020. Using pollutant gas concentrations, the system will forecast AQI levels and categorize them. This tool aims to aid policymakers, urban planners, and individuals in combating air pollution and protecting public health.

Task:

- **Predict** future AQI values based on user inputs of different gas concentration values.
- **Classify** Air Quality into categories like good, satisfactory, moderate, poor, very poor or severe based on user inputs of different gas concentration values.

Novel Approaches:

- **Feature Importance Analysis:** Utilize Random Forest to identify the most influential pollutant gases for AQI prediction and classification, resulting in a more focused and efficient model.
- **Anomaly Detection with k-means Clustering:** Identify unusual spikes in pollutant concentrations that may represent instrument malfunctions or unexpected pollution events.
- **Seasonal Variation Analysis:** Investigate historical trends in pollutant levels to uncover their seasonal trends during specific times of the year thus identifying potential causes.
- **Pollution Source Analysis:** Explore potential relationships between specific pollutants and their sources (e.g., vehicular vs. industrial emissions) to provide insights for targeted policy interventions.

3. LITERATURE REVIEW:

The sample of a dataset used in all below research papers is –

City	Datetime	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket
Delhi	02-01-2015 15:00	188	49.89	16.71	29.06	43.74	33.5	10.93	5.5	15.8	3.51	10.28	1	402	Severe
Delhi	02-01-2015 16:00	185.2	64.7	13.86	27.68	38.4	32.75	14.06	5	30.36	3.6	15.2	1.16	402	Severe
Delhi	02-01-2015 17:00	157.2	60.09	16.32	26.73	41.17	31.25	18.04	4.83	26.17	3.64	11.53	1.11	400	Very Poor
Delhi	02-01-2015 18:00	137.4	99.63	12.82	26.85	37.01	32.25	15.52	2.83	17.24	3.46	8.64	1.57	398	Very Poor
Delhi	02-01-2015 19:00	50.68	79.18	9.46	27.32	32.08	32	8.38	1.75	17.35	2.74	7.11	1.36	393	Very Poor
Delhi	03-01-2015 04:00	44.78	58.67	30.38	24.08	48.28	30.5	19.74	1.83	16.87	4.13	9.13	1.1	142	Moderate
Delhi	03-01-2015 05:00	51.44	77.92	23.6	23.73	41.25	31.25	12.27	2.5	16.88	3.39	8.63	1.33	103	Moderate
Delhi	03-01-2015 11:00	91.19	72.47	15.25	21.7	41.18	0	11.27	3	15.45	2.44	8.57	1.25	98	Satisfactory
Delhi	03-01-2015 12:00	87.84	72.14	12.58	22.94	39.58	0	9.59	3.25	15.53	2.36	5.54	1.25	90	Satisfactory
Delhi	04-01-2015 01:00	263.5	392.3	71.46	54.18	99.97	146.8	14.54	4.5	36.4	8.73	25.33	5.49	233	Poor
Delhi	04-01-2015 02:00	168.2	325	52.5	41.9	73.18	124.8	12.91	3.12	31.93	8.27	25.18	4.54	256	Poor
Delhi	29-06-2017 21:00	31.5	105	15.8	42.77	5.44	35.88	0	0	0	0.21	0	0	44	Good
Delhi	29-06-2017 22:00	35.5	90	17.1	33.15	4.72	34.53	0	0	0	0.14	0	0	43	Good

Table 1: Sample data in air quality dataset for Delhi City

We have used the same dataset for our project too.

(For more details about dataset please move to section 5)

Air pollution is a growing global concern, posing significant risks to human health and the environment. Machine learning (ML) offers a promising approach for analysing air quality data and developing models to predict and classify Air Quality Index (AQI) values. This literature review explores existing research in this field, focusing on methodologies, applications, and advancements relevant to our project.

Project Focus: Our project builds upon existing research by employing a combination of prediction and classification models, incorporating feature importance analysis, anomaly detection, and seasonal variation analysis. This comprehensive approach aims to develop a robust model for predicting and classifying AQI in Indian cities, utilizing data from 2015 to 2020. This focus on a specific region allows for a more targeted analysis relevant to the air quality challenges faced by India.

By leveraging existing research and incorporating advanced techniques, our project aims to contribute to the development of more accurate and informative air quality prediction and classification models, ultimately aiding in combating air pollution and protecting public health.

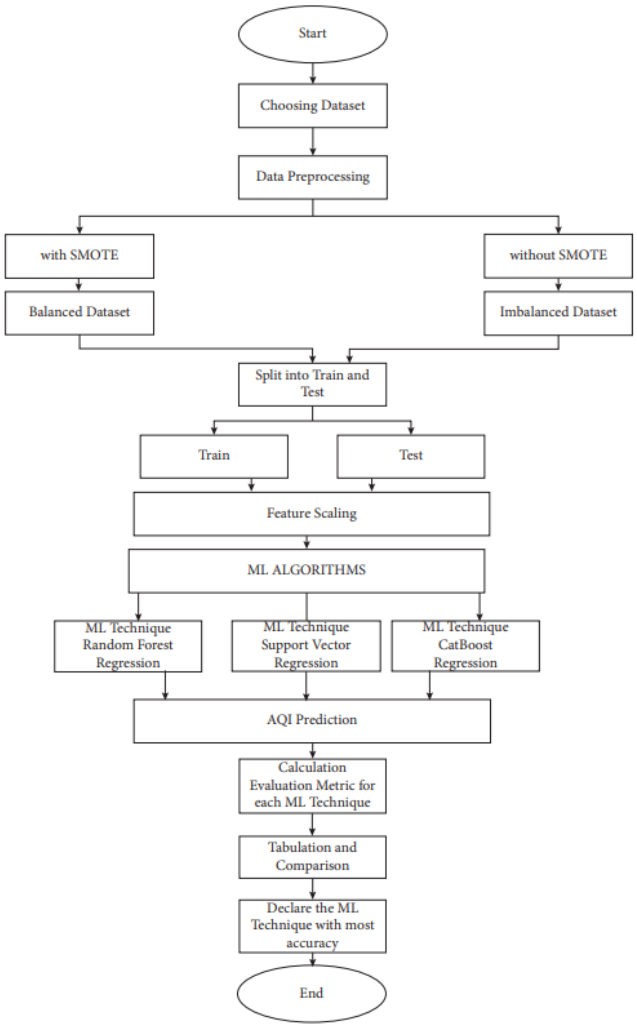
Research Paper 1: (Hindawi) [Taken by Aryan Lunawat]

Prediction of Air Quality Index Using Machine Learning Techniques: A Comparative Analysis

CITATION:

N. Srinivasa Gupta, Yashvi Mohta, Khyati Heda, Raahil Armaan, B. Valarmathi, G. Arulkumaran, "Prediction of Air Quality Index Using Machine Learning Techniques: A Comparative Analysis", Journal of Environmental and Public Health, vol. 2023, Article ID 4916267, 26 pages, 2023.
https://doi.org/10.1155/2023/4916267

Flowchart for proposed methodology in this research paper:



In this paper, the proposed methods use three different algorithms to draw a comparative analysis of the AQI values which will then compare the three algorithms and find the most accurate and efficient algorithm.

The algorithms being used for prediction are –

- Support Vector Regression
- Random Forest Regression
- CatBoost Regression

AQI Prediction is done which is in turn evaluated & compared for each ML technique by R-SQUARE, MSE, RMSE, MAE, and the accuracy (1-MAE)

Results:

- Cleaning dataset improves accuracy.
- Random Forest regression generally produced the lowest Root Mean Squared Error (RMSE) across most cities.
- CatBoost regression and Support Vector Regression achieved the highest accuracy in some cases but varied depending on the city and dataset balance.

Name of the algorithm	Accuracy in percentage (%)
Naïve Bayes (NB)	86.663
Support vector machine (SVM)	92.40
Artificial neural network (ANN)	84–93
Gradient boost (GB)	96
Decision tree (DT)	91.9978
Enhanced k-means	71.28
Support vector regression (SVR)	99.4
Random forest regression (RFR)	99.985
CatBoost regression (CR)	99.88

Table 2: Different algorithms accuracy in %

Conclusion:

Random forest regression consistently yielded promising results and achieved the highest accuracy in different cities Random Forest consistently outperformed Support Vector Regression. We will be using Random Forest Regression due to its highest accuracy as demonstrated in previous different works studied through literature survey for AQI Prediction.

Research Paper 2: (Diva) [Taken by Aryan Rawtani]

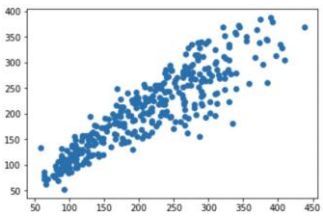
Prediction of Air Quality Index Using Supervised Machine Learning

CITATION:

Gogineni, A. C., & Murukonda, V. S. N. M. (2022). Prediction of Air Quality Index Using Supervised Machine Learning [Bachelor's thesis, Blekinge Institute of Technology]. diva-portal.org. https://www.divaportal.org/smash/get/diva2:1681590/FULLTEXT02.pdf

- This paper explores using machine learning to predict Air Quality Index (AQI) due to growing worries about air pollution's impact. We aim to find the best machine learning method for accurate AQI prediction.
- The methodology involves using methods Linear Regression, Ridge Regression, LASSO Regression, and Support Vector Regression (SVR) and the criteria used to select these as the machine learning algorithms for this study were based on the outcomes of a literature review.
- These algorithms were identified as having higher accuracy in predicting the Air Quality Index (AQI) from previous research

Results:



Graph 1: Scatter plot for linear regression model between observed (X-axis) & predicted (Y-axis) AQI values

Model	MAE	RMSE	R-square
Linear regression	29.227	38.485	0.7441
Lasso regression	27.908	36.791	0.8089
Ridge regression	27.907	36.791	0.8089
Support vector regression	29.828	41.330	0.6814

Table 3: Comparison of performance metrics for all models

Conclusion:

Ridge regression excelled in AQI prediction (Table 3): lowest MAE, RMSE, and highest R-squared error indicating being better for forecasting the AQI.

Research Paper 3: (International Journal of Environmental Science & Technology) [Taken by – Niharika Sharma]

Springer

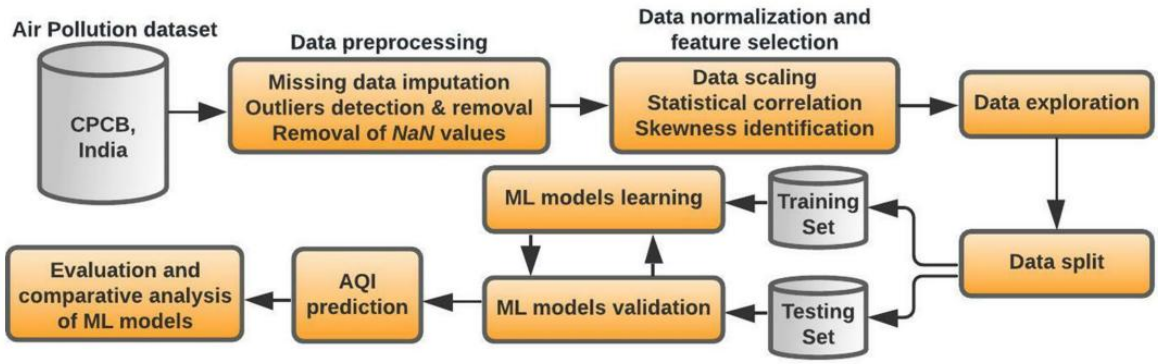
Air pollution prediction with machine learning: a case study of Indian cities

Citation:

Kumar K, Pande BP. Air pollution prediction with machine learning: a case study of Indian cities. Int J Environ Sci Technol (Tehran). 2023;20(5):5333-5348. doi: 10.1007/s13762-022-04241-5. Epub 2022 May 15. PMID: 35603096; PMCID: PMC9107909.

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9107909/pdf/13762_2022_Article_4241.pdf

Flowchart of the proposed model:



- Used air pollution data from 23 Indian cities from Jan 2015 to July 2020 (29,531 instances). Pre-processed data by filling missing values, normalizing, and log transformations.
- Applied SMOTE to handle imbalanced classes in AQI target variable. Employed 5 machine learning models: KNN, Gaussian Naive Bayes, SVM, Random Forest, XGBoost.
- Evaluated models using accuracy, precision, recall, F1-score, MAE, RMSE, RMSLE, R-squared.

Results:

Table 4 Comparison of model results in the training set

Model	Accuracy	Precision	Recall	F1-score	Training time (in seconds)
KNN	89	94	90	96	0.104
GNB	85	91	94	88	0.110
SVM	**81**	90	93	88	0.258
RF	88	93	88	92	0.102
XGBoost	**91**	95	95	91	0.532

Table 5: Comparison of model results in the testing set

Model	Accuracy	Precision	Recall	F1-Score	Prediction time (in seconds)
KNN	85	92	85	94	0.018
GNB	83	88	89	92	0.016
SVM	**78**	91	90	83	0.027
RF	86	92	91	90	0.023
XGBoost	90	96	95	91	0.041

Conclusion:

As KNN is having

- Accuracy > 80. It has overall proportion of correct predictions
- Precision > 80. It has Proportion of correct positive predictions.
- Recall > 80. It has proportion of actual positives correctly identified.
- F1-Score > 80. It has balanced view of precision and recall.

Therefore, we are choosing to go with KNN model.

Research Paper 4: (Nature Scientific Reports) [Taken by – Yash Puri Goswami] nature portfolio

Optimized machine learning model for air quality index prediction in major cities in India

Citation:

https://nature.com/scientificreports

by Suresh Kumar Natarajan, Prakash Shanmurthy, Daniel Arockiam, Balamurugan Balusamy & Shitharth Selvarajan

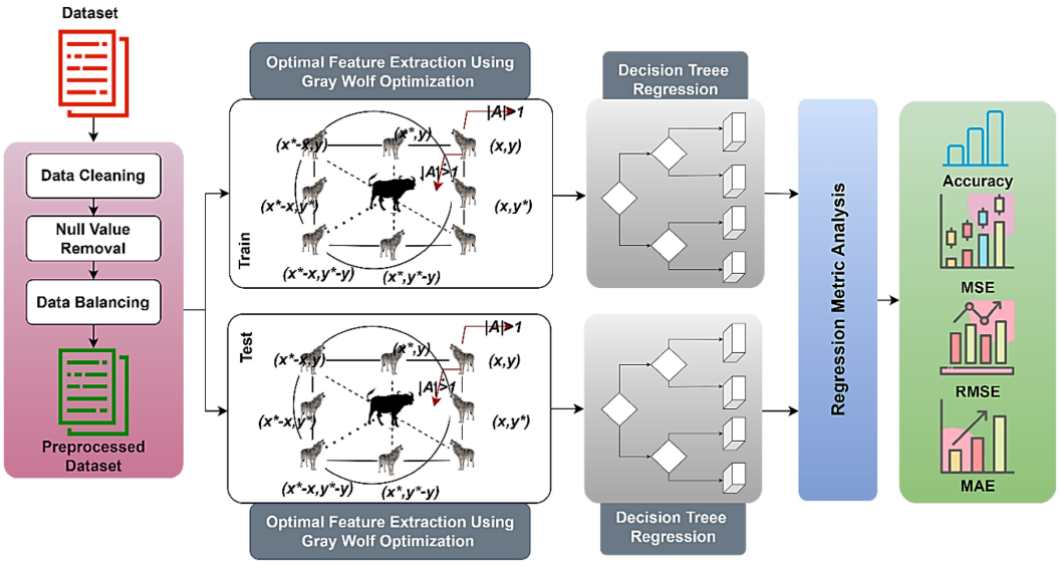
Thoroughly studying the paper gives an intuition about the need for a precise and accurate model for analysing the air quality in developing nations such as India.

The model used in this research paper utilizes machine learning techniques such as Decision-Tree Regression and Grey Wolf Optimization.

For more preciseness of the model, the authors and scientists also utilized metrics such as Accuracy, Mean Squared Error and Mean Absolute Error along with Root Mean Squared Error.

The model utilizes data of air pollutants of the major cities in India namely New Delhi, Bangalore, Mumbai, Chennai, Vishakhapatnam and Hyderabad.

Proposed Prediction Model:



Results:

S. no.	City	Accuracy			
		Support vector regression	K-nearest neighbor	Random forest regression	Proposed GWO-DT
1	New Delhi	84.83	83.68	84.73	88.98
2	Bangalore	87.18	89.47	90.31	91.49
3	Kolkata	91.56	90.65	93.74	94.48
4	Hyderabad	93.57	93.68	97.61	97.66
5	Chennai	92.65	93.48	94.48	95.22
6	Visakhapatnam	92.24	92.11	95.65	97.68

Table 6: Performance comparison with conventional algorithms.

Conclusion:

The proposed methodology used in the research paper was accurate than most of the common and basic machine learning algorithms with that its accuracy was also on a higher side.

The model used two techniques together to make the model more optimized in analysing the AQI of the cities. While the model was accurate the data used was not optimum.

The dataset used by us has large number of rows and columns which involves the scope of overfitting. Also, GWO has proven inefficient when large number of data is involved due to scalability and curse of dimensionality.

Using Decision Tree has a greater scope in making the model understandable as its easier to interpret. Its ability to handle both numerical and categorical data will enable us in handling the large amount of dataset having variety of data.

4. METHODOLOGY:

This section gives details of the methods that we applied, the details of novel approaches in our work and analysis of our systematic approach used to carry out our project to meet project goals. It offers an organized summary of the project's execution. Included in this section are:

1. **Data Preprocessing and Feature Engineering:** Addressing missing values, outliers, and performing feature importance analysis with Random Forest enhances data quality and focuses the model on the most impactful pollutant gases.

2. **Exploratory Data Analysis:** Visualizing and analysing the data allows you to understand trends, corelations and relationships between different variables like pollutant concentrations & AQI to identify patterns in data.

3. **Seasonal and Source Variation Analysis**: Investigating seasonal variations and pollutant sources provides valuable insights into air quality dynamics and potential contributing factors. This can aid in interpreting model predictions and future policy interventions.

4. **Anomaly Detection with K-means Clustering:** Identifying unusual data points with K-means helps ensure data quality by potentially uncovering sensor malfunctions or unexpected pollution events, leading to more reliable models.

5. **Data Splitting:** Split the dataset into training and test sets to train and evaluate your machine learning models effectively. This ensures that the models generalize well to unseen data.

6. **Model Selection:** Various machine learning algorithms, including linear regression, decision tree regressor, random forest regressor, KNN, logistic regression, and decision tree classifier, were applied to predict AQI and classify air quality based on pollutant concentrations.

Model Selection:

This section deals with the explanation behind the selection of machine learning algorithms used for the project.

1. Linear Regression:
   It is a statistical method used for formulating a relationship between a dependent and one or more independent variables (features). It assumes a linear relationship between the input variables and the output variables, represented by a straight line in a multi-dimensional space. The main goal of linear regression is to find the best-fitting straight line that describes the relationship between the independent variables and the dependent variable.

   ⇨ Equation of Linear Regression Hypothesis function:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

   ⇨ Cost Function:
   The cost function, also known as the loss function is a key concept in machine learning algorithms, including Linear Regression. It quantifies the difference between the predicted values and the actual values of the target variable. The goal of the learning algorithm is to minimize this cost function, which leads to better model predictions.

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

2. Logistic Regression:
   Logistic Regression is a statistical model used for binary classification tasks, where the target variable has two possible outcomes, often labelled as 0 and 1. It is named "Logistic" because it uses the logistic function (also known as the sigmoid function) to model the probability of the target variable belonging to a particular class.
   ⇨ Logistic Regression Hypothesis Function:

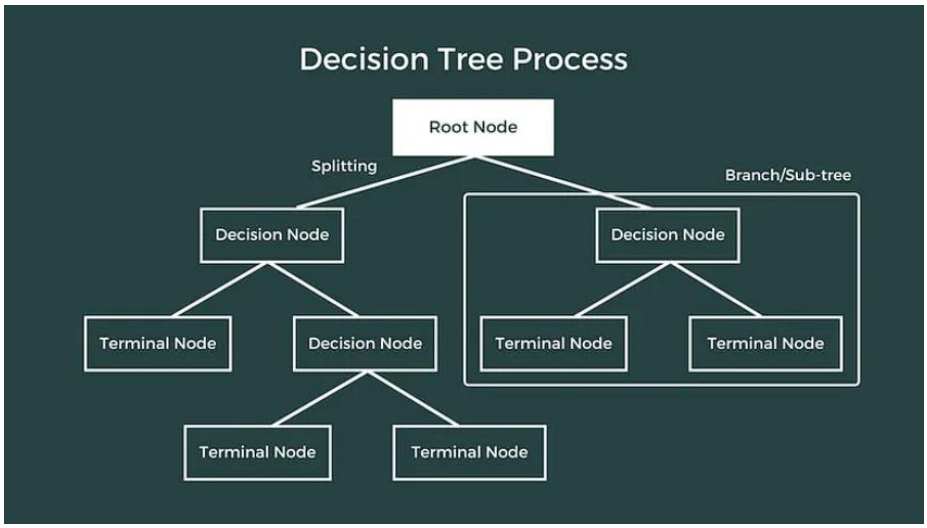
$$h_\Theta(x) = \frac{1}{1 + e^{-\Theta^T x}}$$

   ⇨ Cost function of Logistic Regression:

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

3. Decision Tree:
   A Decision Tree is a supervised learning algorithm used for classification and regression tasks. It's a flowchart-like tree structure where each internal node represents a feature or attribute, each branch represents a decision rule, and each leaf node represents the outcome or the class label.
   a. Decision Tree Regressor:
      A Decision Tree Regressor is a supervised learning algorithm used for solving regression problems. It's a tree-like structure where each internal node represents a decision based on a feature, each branch represents the outcome of that decision and each leaf node represents the value of the target variable.

   b. Decision Tree Classifier:
      A Decision Tree Classifier is a supervised learning algorithm used for classification tasks, where the goal is to predict the categorical class label of new instances based on the features available in the dataset. It operates by recursively partitioning the feature space into regions, each associated with a specific label.



4. Random Forest:
   Random Forest is a versatile and powerful ensemble learning method used for both classification and regression tasks. It is an extension of decision tree algorithms and combines the concepts of concepts of bootstrap

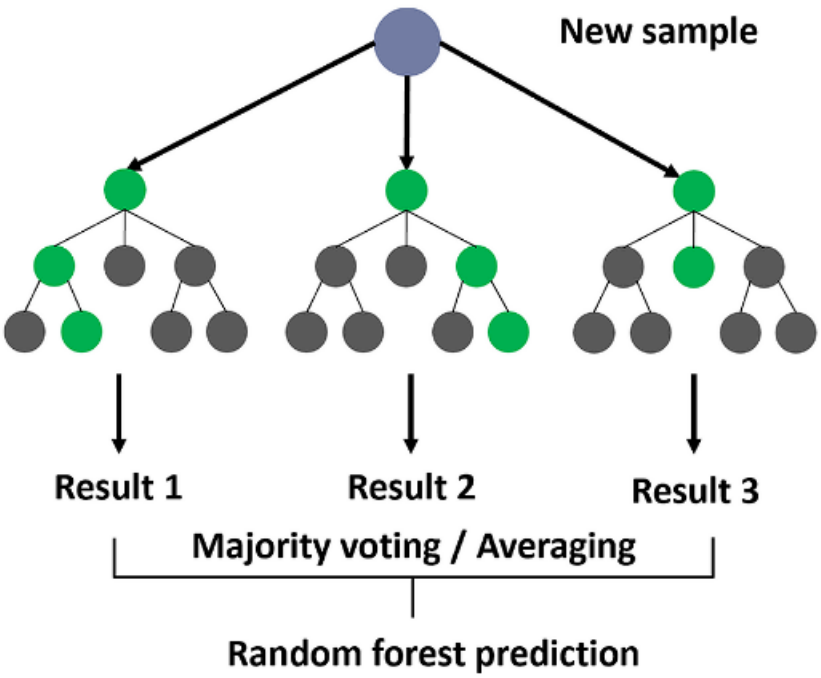
aggregation (bagging) and random feature selection to build a robust and accurate predictive model.

A) Random Forest Regressor:

The Random Forest Regressor is a supervised learning algorithm used for regression tasks. It is an ensemble learning method based on the Random Forest Algorithm, which combines the predictions of multiple decision trees to improve accuracy and robustness in predicting continuous numerical outcomes.

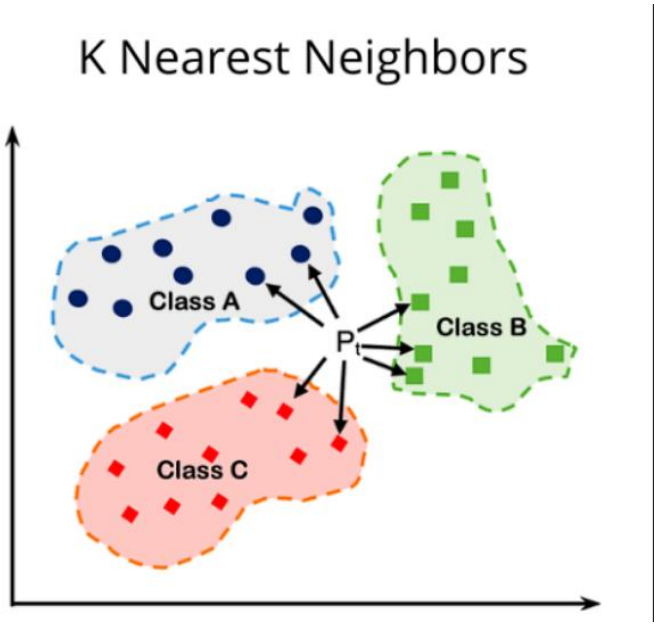
B) Random Forest Classifier:

The Random Forest Classifier is a supervised learning algorithm used for classification tasks. It is an ensemble learning method based on the Random Forest algorithm, which combines the predictions of multiple decision trees to improve accuracy and robustness in classifying data into multiple classes or categories.



5. K-Nearest Neighbours:
   The K-Nearest Neighbours (KNN) is a supervised machine-learning algorithm used for both classification and regression tasks. It is a non-parametric and instance-based learning algorithm, meaning it doesn't make

strong assumptions about the underlying data distribution and learns directly from the training instances.



Model Evaluation Metrics:

1. Mean Squared Error:
   Mean Squared Error (MSE) is a common metric used to evaluate the performance of regression models. It quantifies the average squared difference between the actual values (observed) and the predicted values produced by the model. MSE is a measure of the average squared deviation of predicted values from the actual values, providing a measure of the model's accuracy in predicting continuous numeric outcomes.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

2. Root Mean Squared Error:
   Root Mean Squared Error (RMSE) is a widely used metric for evaluating the performance of regression models, particularly when dealing with continuous numerical data. It measures the average magnitude of the residuals or errors between predicted values and actual values, providing a measure of the model's accuracy in predicting quantitative outcomes.

$$RMSE = \sqrt{\sum_{i=1}^{n}\frac{(\hat{y}_i - y_i)^2}{n}}$$

3. Mean Absolute Error:
   Mean Absolute Error is another metric for evaluating regression models. It calculates the average absolute difference between predicted and actual values, providing a measure of the model's predictive accuracy without the influence of squared errors.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|x_i - x|$$

4. R-Squared Error:
   R2 square is a metric that explains the proportion of variance in the dependent variable that is predictable from the independent variables. It is valuable for understanding how well the model fits the data relative to a simple baseline model. A higher R2 score indicates a better fit.

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

5. Accuracy:
   Accuracy is a metric used to measure the performance of a classification model. It represents the proportion of correct predictions (both true positives and true negatives) among all the predictions made by the model. In other words, accuracy quantifies how often the model correctly predicts the outcome or class label of the data points.

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

6. Kappa Score:
   Cohen's Kappa coefficient, often referred to simply as the Kappa Score or Kappa statistic, is a statistical measure used to assess the level of agreement between two raters or between an algorithm and human raters in classification tasks. It takes into account the possibility of agreement occurring by chance.

$$\kappa = \frac{p_0 - p_e}{1 - p_e}$$

7. F1 Score, Precision & Recall:
   Precision measures the proportion of true positive predictions (correctly identified positive instances) among all instances predicted as positive (including both true positives and false positives). It focuses on the accuracy of positive predictions.

Recall measures the proportion of true positive predictions (correctly identified positive instances) among all actual positive instances. It focuses on how many actual positive instances were correctly identified by the model.

F1 Score is the harmonic mean of Precision and Recall. It provides a balance between Precision and Recall, considering both false positives and false negatives. The F1 Score reaches its best value at 1 (perfect precision and recall) and worst at 0.

$$Precision = \frac{T_p}{T_p + F_p}$$

$$Recall = \frac{T_p}{T_p + T_n}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

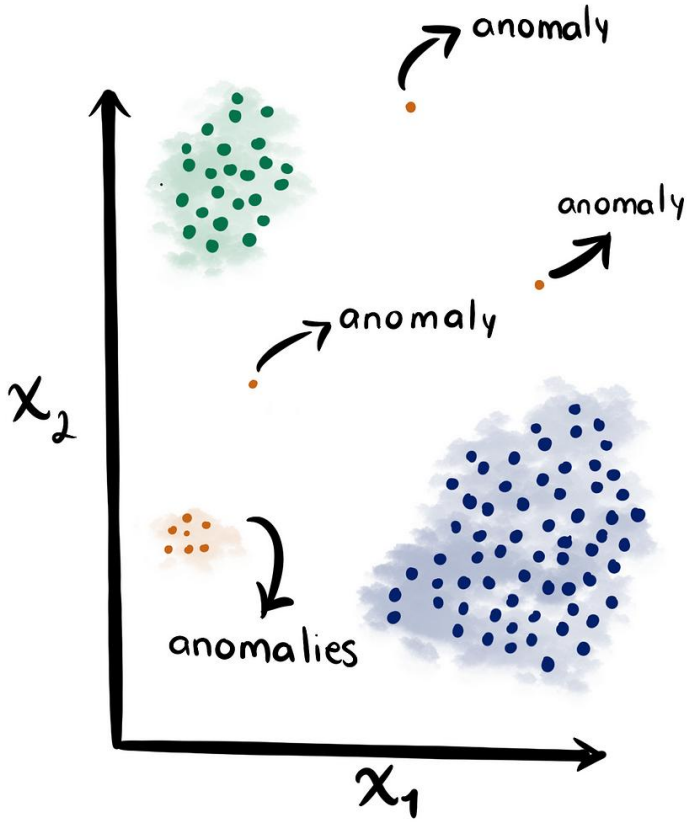
NOVEL APPROACHES:

1. Anomaly Detection:

Used to detect unusual spikes in pollutant concentrations. To Identify unusual patterns or outliers in the dataset that may indicate pollution events or instrument malfunction.

Approach: To anomaly detection using K-means clustering and calculating anomaly scores based on distances from cluster centers.

1. Model Training:
   Choose the number of clusters: Determine the appropriate number of clusters (K) for the K-means algorithm. This can be done using techniques like the elbow method or silhouette analysis.
2. Train the K-means model:
   Fit the K-means clustering algorithm to the standardized feature data. K-means partitions the data into K clusters based on feature similarity.
3. Anomaly Score Calculation:
   Predict cluster labels: Assign each data point to its nearest cluster centroid based on Euclidean distance.
4. Calculate distances:
   Compute the distance of each data point to its nearest cluster centroid. This distance serves as an anomaly score, indicating how far a data point is from the cluster center.
5. Normalize distances: Normalize the distances to a range between 0 and 1, typically by dividing each distance by the maximum distance.
6. Anomaly Identification:
   Define an anomaly threshold: Choose a threshold value (e.g., 0.95) above which data points are considered anomalies. This threshold can be adjusted based on domain knowledge or experimentation.
7. Identify anomalies:
   Flag data points as anomalies if their normalized distance (anomaly score) exceeds the defined threshold. These anomalies represent unusual spikes or outliers in pollutant concentrations that may indicate pollution events or instrument malfunction.



This picture depicts anomaly detection using K – Means Clustering

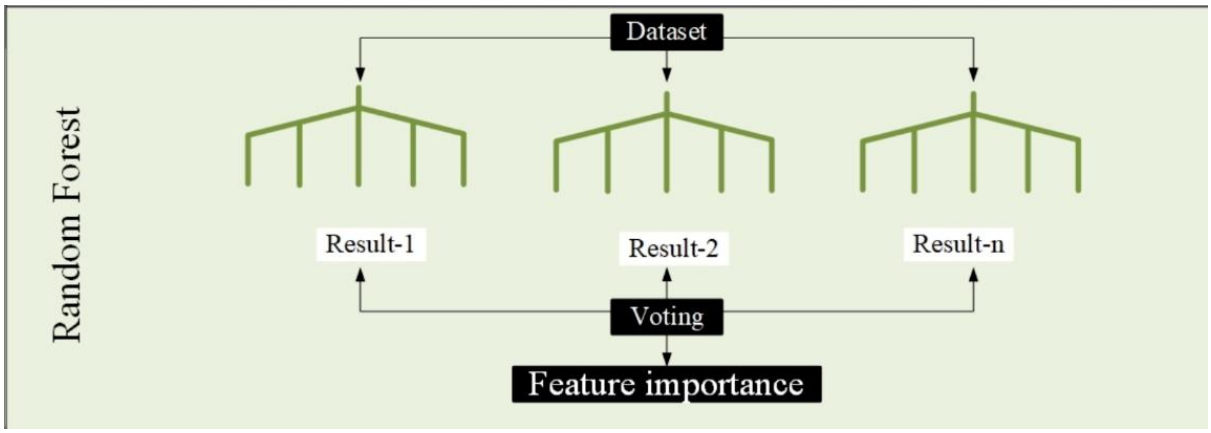
2. Seasonal Trend Analysis:

Seasonal trends significantly influence air quality dynamics, with distinct variations observed across different seasons. Summers and winters often exhibit higher pollutant concentrations due to temperature inversions and stagnant weather conditions, trapping pollutants close to the ground. Conversely, monsoon months experience lower pollutant levels attributed to increased rainfall and wind, aiding in dispersing pollutants from the atmosphere. Historical data analysis enables the identification of these patterns, highlighting the impact of seasonal factors on the Air Quality Index (AQI). Understanding these trends is crucial for implementing targeted strategies to mitigate pollution and improve overall environmental health. Analysing pollutant levels across seasons provides valuable insights for informed decision-making in air quality management and pollution control measures.

3. Feature Importance Analysis:

Analysing feature importance helps in understanding which factors significantly contribute to predicting the AQI. This information aids in feature selection, model optimization, and interpreting the model's behavior, ultimately contributing to better decision-making in air quality management and pollution control strategies.

1. Load Data and Select Features:
   Load the dataset and select relevant features that can predict AQI.
2. Build Random Forest Regressor Model:
   Initialize Random Forest Regressor with parameters such as n_estimators=100 for the number of trees and random_state for reproducibility.
3. Fit and Extract Feature Importances:
   Train the Random Forest Regressor model using selected features and AQI as the target variable.
4. Extract feature importances using the feature_importances_ attribute of the trained model.
5. Visualize Feature Importance:
   Create a data frame to store feature importances.
6. Plot a horizontal bar chart to visualize feature importance scores.



4. Pollution Source Analysis:

In preprocessing, a new perspective is added by aggregating pollutants associated with vehicular and industrial sources. Vehicular pollution encompasses PM2.5, PM10, NOx, and CO, while industrial pollution is represented by O3 levels. These aggregations offer a more focused analysis, enabling a comparative assessment of their impact on air quality. Visualized through scatter plots, each point represents a city's pollution level, categorized by its Air Quality Index (AQI) bucket, providing insights into how industrial and vehicular emissions correlate with overall air quality. Such analyses are crucial for understanding pollution sources and devising targeted mitigation strategies, guiding policymakers toward sustainable urban development, and ensuring public health and environmental well-being.

5. DATASET USED:

City	Datetime	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket
Delhi	02-01-2015 15:00	188	49.89	16.71	29.06	43.74	33.5	10.93	5.5	15.8	3.51	10.28	1	402	Severe
Delhi	02-01-2015 16:00	185.2	64.7	13.86	27.68	38.4	32.75	14.06	5	30.36	3.6	15.2	1.16	402	Severe
Delhi	02-01-2015 17:00	157.2	60.09	16.32	26.73	41.17	31.25	18.04	4.83	26.17	3.64	11.53	1.11	400	Very Poor
Delhi	02-01-2015 18:00	137.4	99.63	12.82	26.85	37.01	32.25	15.52	2.83	17.24	3.46	8.64	1.57	398	Very Poor
Delhi	02-01-2015 19:00	50.68	79.18	9.46	27.32	32.08	32	8.38	1.75	17.35	2.74	7.11	1.36	393	Very Poor
Delhi	03-01-2015 04:00	44.78	58.67	30.38	24.08	48.28	30.5	19.74	1.83	16.87	4.13	9.13	1.1	142	Moderate
Delhi	03-01-2015 05:00	51.44	77.92	23.6	23.73	41.25	31.25	12.27	2.5	16.88	3.39	8.63	1.33	103	Moderate
Delhi	03-01-2015 11:00	91.19	72.47	15.25	21.7	41.18	0	11.27	3	15.45	2.44	8.57	1.25	98	Satisfactory
Delhi	03-01-2015 12:00	87.84	72.14	12.58	22.94	39.58	0	9.59	3.25	15.53	2.36	5.54	1.25	90	Satisfactory
Delhi	04-01-2015 01:00	263.5	392.3	71.46	54.18	99.97	146.8	14.54	4.5	36.4	8.73	25.33	5.49	233	Poor
Delhi	04-01-2015 02:00	168.2	325	52.5	41.9	73.18	124.8	12.91	3.12	31.93	8.27	25.18	4.54	256	Poor
Delhi	29-06-2017 21:00	31.5	105	15.8	42.77	5.44	35.88	0	0	0	0.21	0	0	44	Good
Delhi	29-06-2017 22:00	35.5	90	17.1	33.15	4.72	34.53	0	0	0	0.14	0	0	43	Good

This is a sample dataset of Delhi city.

We've collected this dataset named: Air Quality Data in India (2015 - 2020) from Kaggle's Repository to use for our model.

About the dataset:

- The dataset contains air quality data and AQI (Air Quality Index) on a daily basis of various stations across multiple cities in India.
- The data set contains **29531 rows** and **16 columns**
- The columns include parameters such as Date, City, PM2.5, PM10, NO2, SO2, CO, O3, NO, NOx, NH3, Benzene, Toluene, Xylene, AQI (Air Quality Index) and AQI Bucket.
- The "City" column has these cities - Ahmedabad, Aizawl, Amaravati, Amritsar, Bengaluru, Bhopal, Brajrajnagar, Chandigarh, Chennai, Coimbatore, Delhi, Ernakulam, Gurugram, Guwahati, Hyderabad, Jaipur, Jorapokhar, Kochi, Kolkata, Lucknow, Mumbai, Patna, Shillong, Talcher, Thiruvananthapuram, Visakhapatnam
- The "AQI_Bucket" has these unique entries – good, satisfactory, moderate, poor, very poor and severe.

The data has been made publicly available by the Central Pollution Control Board: https://cpcb.nic.in/ which is the official portal of the Government of India.

The link to this dataset:

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

6. EXPERIMENTS:

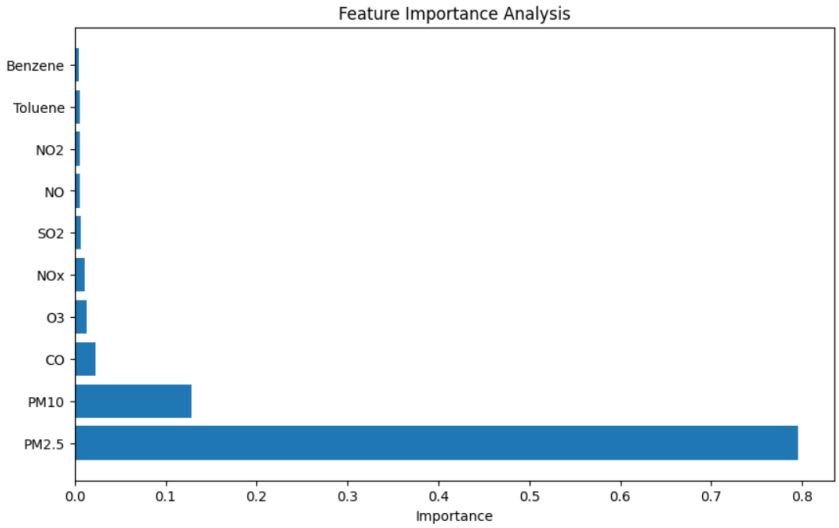
1) Data Preprocessing and Feature Engineering:

Data Preprocessing:

1. **Setting Date Column as Index:**
   We did this as it would be easier to visualize data taking time as index.
2. **Data Cleaning:**
   - **Handling Null Values:** Firstly, we found out about percentage of null values in each column of our dataset and dropped columns that has null values more than 40% (“Xylene”).
   - **Outlier Detection and Filtering:** Then we found that the many concentration values of the gases and AQI in our dataset were very high which was not possible. We used box plot to detect outliers and keeps only the rows in the data frame where the concentrations of all pollutants and the AQI itself fall below specific thresholds. This left us with cleaned data with no null values now having size (10584,14)

Feature Importance Analysis:

We did this using Random Forest as we were having a lot of gases and wanted to know which gases have the most impact in Air Quality prediction and classification. and drop the others which don’t affect it. This allows focussing models on the most relevant features. Here we found out that PM2.5, PM10, CO, O3 and NOx had the most impact so we dropped all the other gas columns.



Now this is our final cleaned dataset (shape = 10584,8) where we will perform exploratory analysis, novel ideas, model training, evaluations etc.

Columns present are – City, PM2.5, PM10, NOx, CO, O3, AQI and AQI_Bucket.

Cities are - Aizawl, Amaravati, Amritsar, Bengaluru, Chandigarh, Chennai, Coimbatore, Delhi, Gurugram, Hyderabad, Jaipur, Kolkata, Patna, Shillong, Talcher, Visakhapatnam = 16 Cities

AQI_Bucket contains these categories – good, satisfactory, moderate, poor, very poor, severe

2) Performed Exploratory Data Analysis: Look at results to view visualizations.

3) Seasonal and Source Variation Analysis:

- **Seasonal Variation Analysis:** In this, we calculated mean of pollutant concentration at each month over a period of 5 years to analyse patterns and trends in the concentration values.
- **Pollutant Variation Analysis:** In this, we wanted to find which major source of pollution between these – Industrial and Vehicular are contributing to more Air Pollution. We grouped different pollutants that each source releases and compared which had higher value.

4) Anomaly Detection with K-means Clustering:

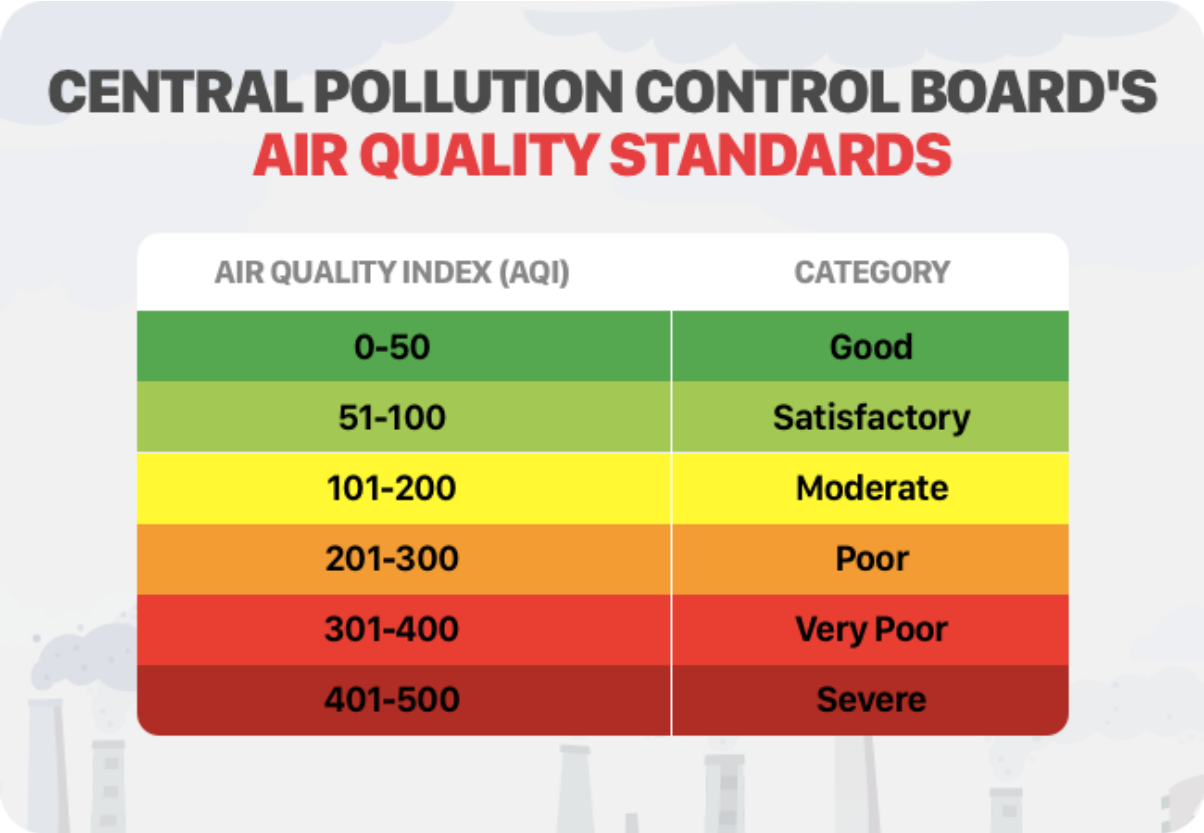
- Used to detect unusual spikes in pollutant concentrations. To Identify unusual patterns or outliers in the dataset that may indicate pollution events or instrument malfunction.
  Approach: To anomaly detection using K-means clustering and calculating anomaly scores based on distances from cluster centers.

5) Splitting Data into Training and Test Sets:

- Training: The model learns from the training set (80% of data), identifying relationships between pollutant concentrations and AQI.
- Testing: The testing set (20% of data) is unseen by the model during training. By evaluating on this separate data, you assess how well the model generalizes its predictions to new information.

6) Model Training:

i. Firstly, we imported all necessary libraries, modules, classes.
ii. Then we divided our models into two parts – prediction & classification.
iii. Then we train each model using their method specific functions.
    - In Prediction models we have –
        a. Linear Regression
        b. Random Forest Regressor
        c. Decision Tree Regressor
    - In Classification models we have –
        a. Logistic Regression
        b. Decision Tree Classifier
        c. Random Forest Classifier
        d. K – Nearest Neighbours
iv. In Prediction models we give input gas concentration values of PM2.5, PM10, CO, O3, NOx and we get output a predicted AQI value
v. In Classification models we give input gas concentration values of PM2.5, PM10, CO, O3, NOx and we get output a category into which it's Air Quality falls into.



7. MODEL EVALUATIONS:

i. Then we evaluate each model using evaluation metrics
   - In Prediction we have –
     a. RMSE
     b. R Squared Error
     c. MAE
     d. MSE
   - In Classification we have –
     a. Accuracy
     b. Kappa Score
     c. Precision
     d. Recall
     e. F1 – Score

ii. Then we compare evaluation results with different models

iii. We look at the results of output to finalize our model.

8. RESULTS:

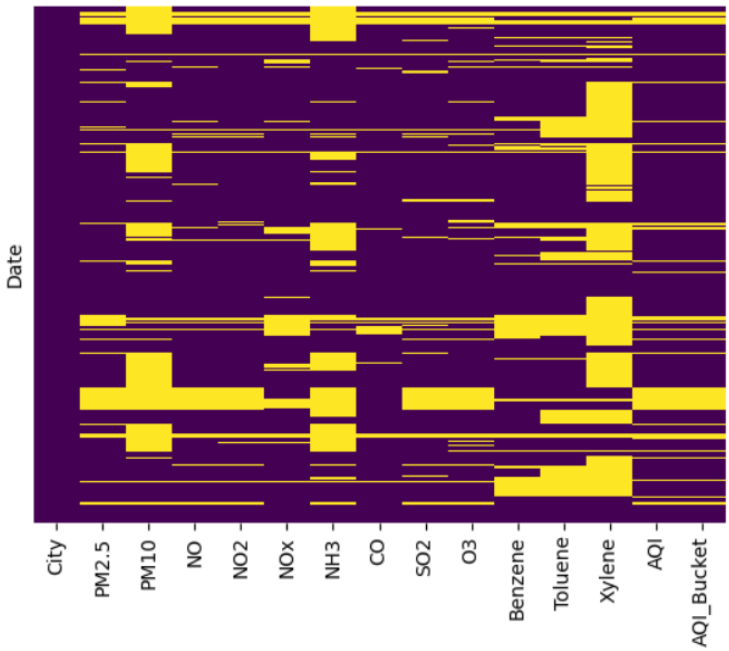
This includes results of our experiments, novel ideas, models trained and their accuracy. This section also includes all resultant visualizations, graphs, tables that we obtained in our project.

1. This was our dataset in the start

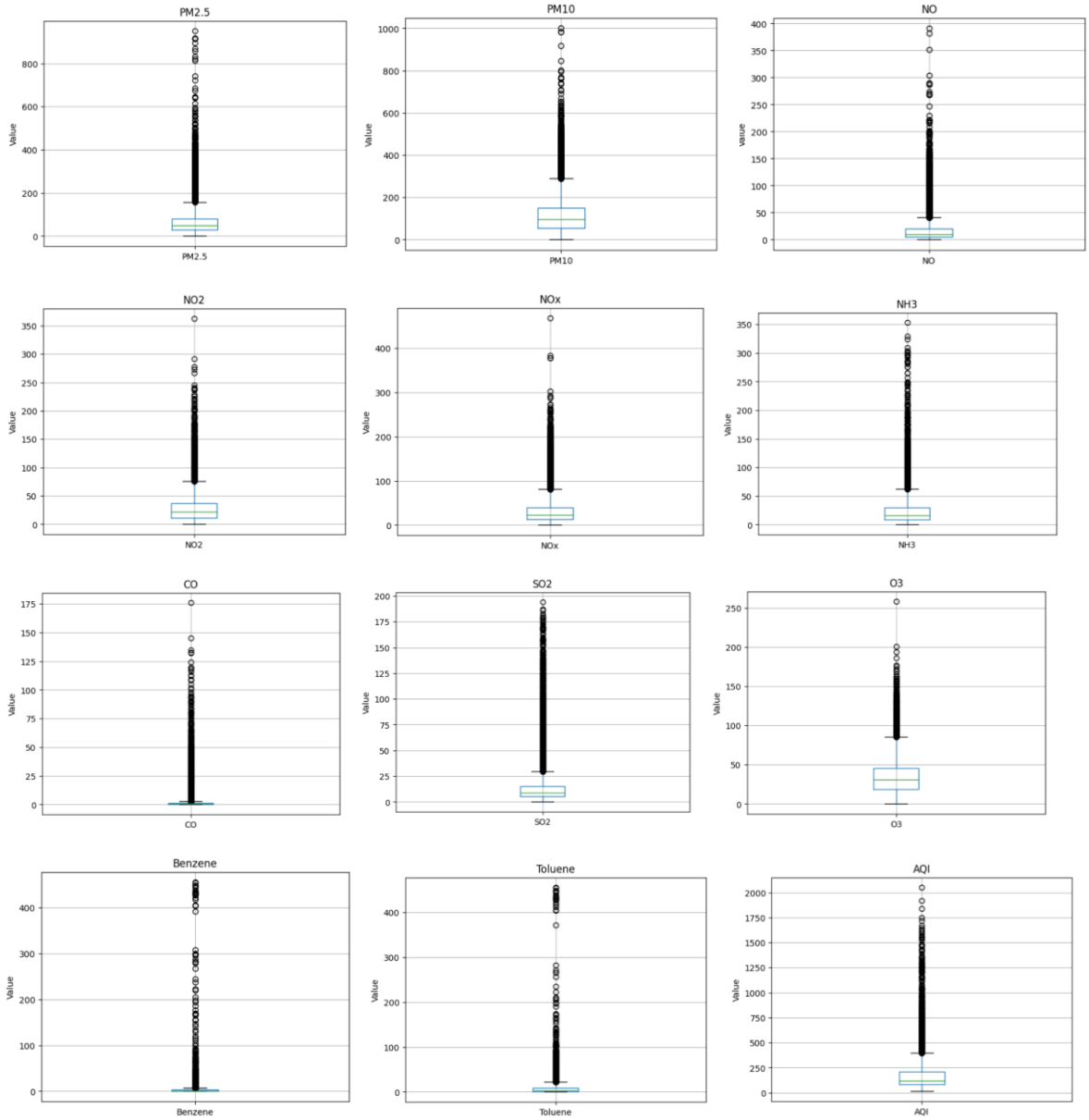
	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket
0	Ahmedabad	2015-01-01	NaN	NaN	0.92	18.22	17.15	NaN	0.92	27.64	133.36	0.00	0.02	0.00	NaN	NaN
1	Ahmedabad	2015-01-02	NaN	NaN	0.97	15.69	16.46	NaN	0.97	24.55	34.06	3.68	5.50	3.77	NaN	NaN
2	Ahmedabad	2015-01-03	NaN	NaN	17.40	19.30	29.70	NaN	17.40	29.07	30.70	6.80	16.40	2.25	NaN	NaN
3	Ahmedabad	2015-01-04	NaN	NaN	1.70	18.48	17.97	NaN	1.70	18.59	36.08	4.43	10.14	1.00	NaN	NaN
4	Ahmedabad	2015-01-05	NaN	NaN	22.10	21.42	37.76	NaN	22.10	39.33	39.31	7.01	18.89	2.78	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
29526	Visakhapatnam	2020-06-27	15.02	50.94	7.68	25.06	19.54	12.47	0.47	8.55	23.30	2.24	12.07	0.73	41.0	Good
29527	Visakhapatnam	2020-06-28	24.38	74.09	3.42	26.06	16.53	11.99	0.52	12.72	30.14	0.74	2.21	0.38	70.0	Satisfactory
29528	Visakhapatnam	2020-06-29	22.91	65.73	3.45	29.53	18.33	10.71	0.48	8.42	30.96	0.01	0.01	0.00	68.0	Satisfactory
29529	Visakhapatnam	2020-06-30	16.64	49.97	4.05	29.26	18.80	10.03	0.52	9.84	28.30	0.00	0.00	0.00	54.0	Satisfactory
29530	Visakhapatnam	2020-07-01	15.00	66.00	0.40	26.85	14.05	5.20	0.59	2.10	17.05	NaN	NaN	NaN	50.0	Good

29531 rows × 16 columns

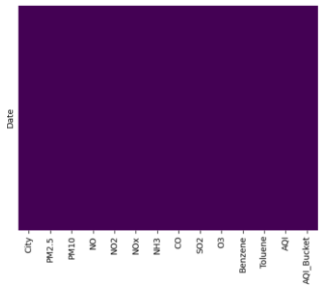
2. This is a heatmap of null values before data cleaning and preprocessing. At this point we have set out date column as index of rows.



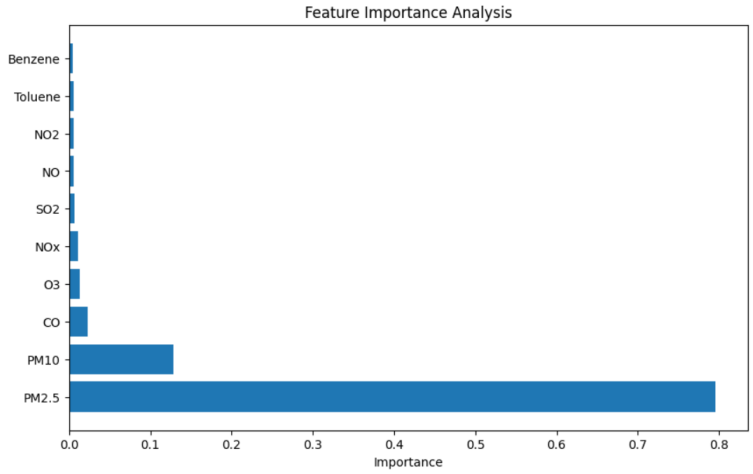
3. Outlier Detection. (At this point we have dropped Xylene as it has 61% null values) We are detecting outliers for all gas concentrations and AQI.



Now we filtered all outlier values and keeps only the rows in the data frame where the concentrations of all pollutants and the AQI itself fall below specific thresholds. This left us with cleaned data with no null values now having size (10584,14). Now our dataset it cleaned. Below is heatmap of null values



4. Now we conducted Feature Importance Analysis

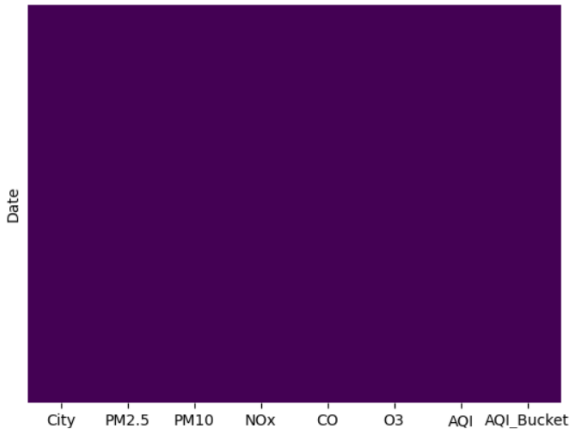


Here we found out that only PM2.5, PM10, O3, CO, NOx are having impact on Air Quality therefore we dropped rest all gas columns.

5. Resulting Dataset and heatmap representing null values

Date	City	PM2.5	PM10	NOx	CO	O3	AQI	AQI_Bucket
2020-03-12	Aizawl	31.21	38.66	10.65	0.56	2.81	52.0	Satisfactory
2020-03-13	Aizawl	38.39	46.68	10.37	0.57	0.18	60.0	Satisfactory
2020-03-14	Aizawl	43.23	50.83	10.48	0.57	0.41	62.0	Satisfactory
2020-03-15	Aizawl	33.82	41.03	9.73	0.48	0.30	70.0	Satisfactory
2020-03-16	Aizawl	27.14	35.04	8.09	0.50	13.02	54.0	Satisfactory
...	...	...	...	...	...	...	...	...
2020-06-26	Visakhapatnam	7.63	32.27	17.19	0.46	19.90	47.0	Good
2020-06-27	Visakhapatnam	15.02	50.94	19.54	0.47	23.30	41.0	Good
2020-06-28	Visakhapatnam	24.38	74.09	16.53	0.52	30.14	70.0	Satisfactory
2020-06-29	Visakhapatnam	22.91	65.73	18.33	0.48	30.96	68.0	Satisfactory
2020-06-30	Visakhapatnam	16.64	49.97	18.80	0.52	28.30	54.0	Satisfactory

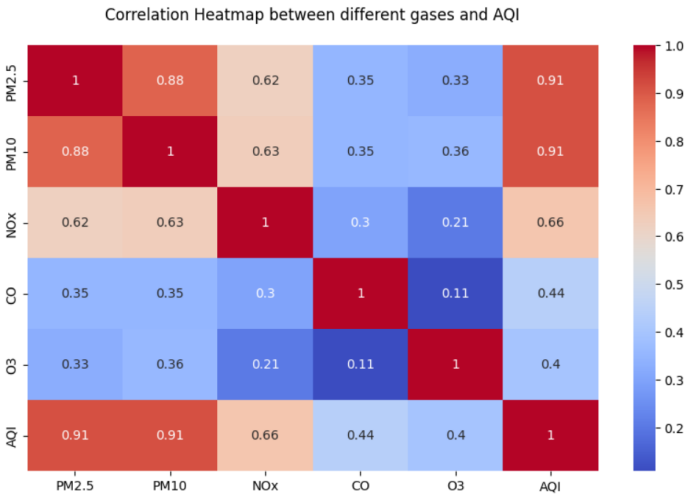
10584 rows × 8 columns



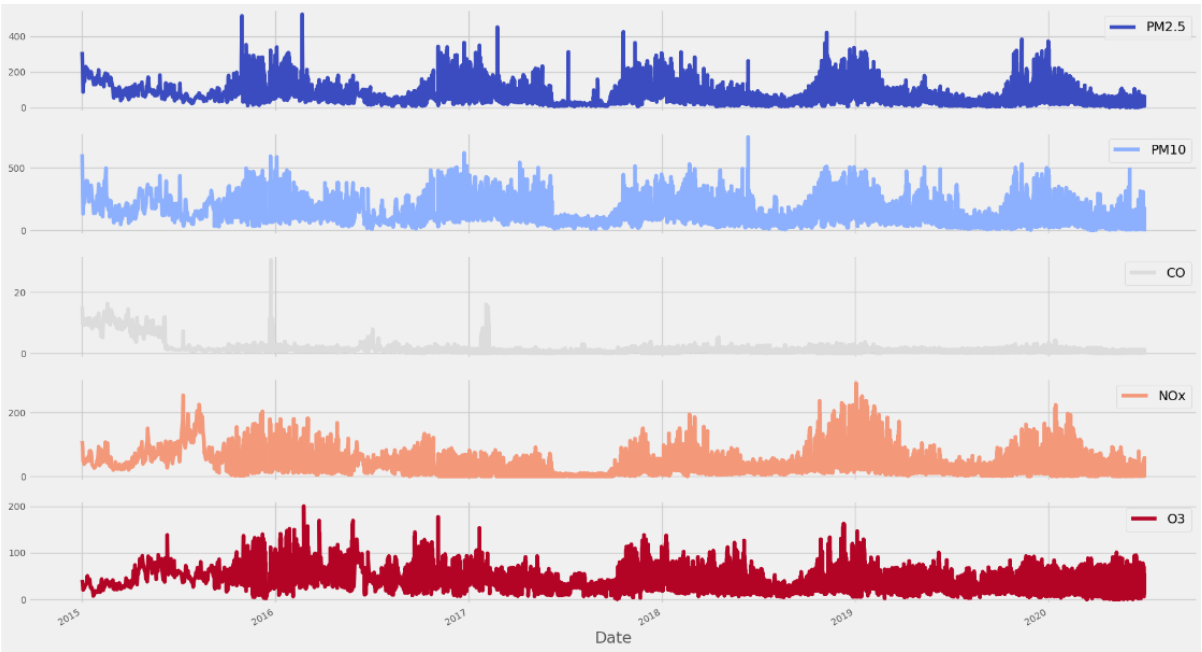
THIS IS OUR FINAL DATASET WE WILL DO ALL OPERATIONS ON THIS.

6. Exploratory Data Analysis:

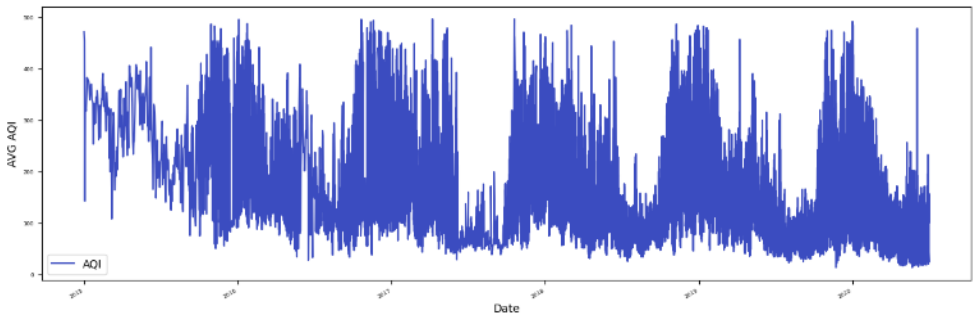
a) Correlation Heatmap between different gases and AQI



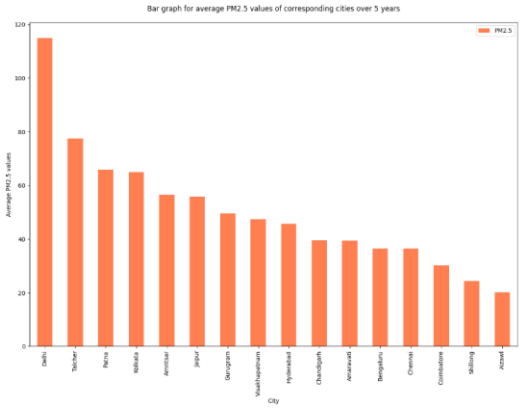
b) Distribution of different pollutants in 5 years

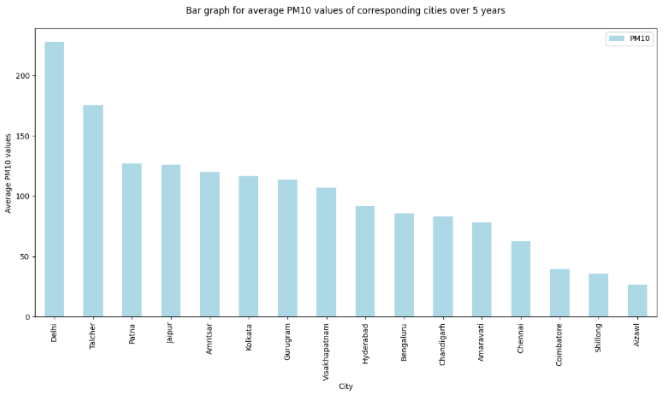


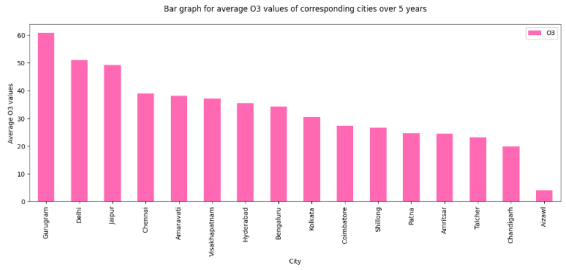
c) Distribution of average AQI over 5 years

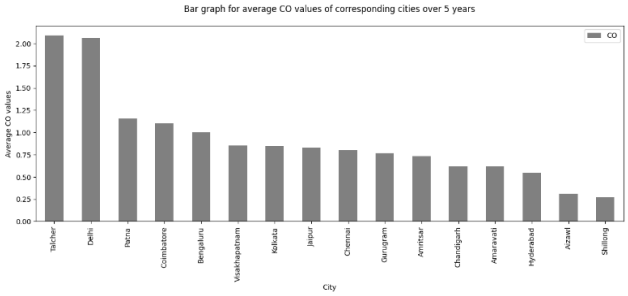


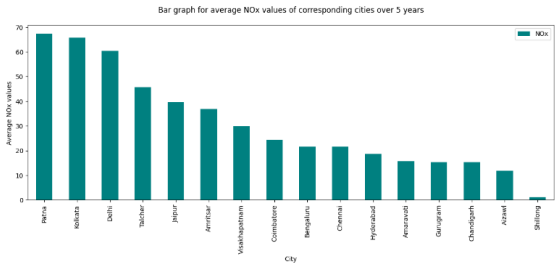
d) Bar graphs for average gas values across different cities over 5 years



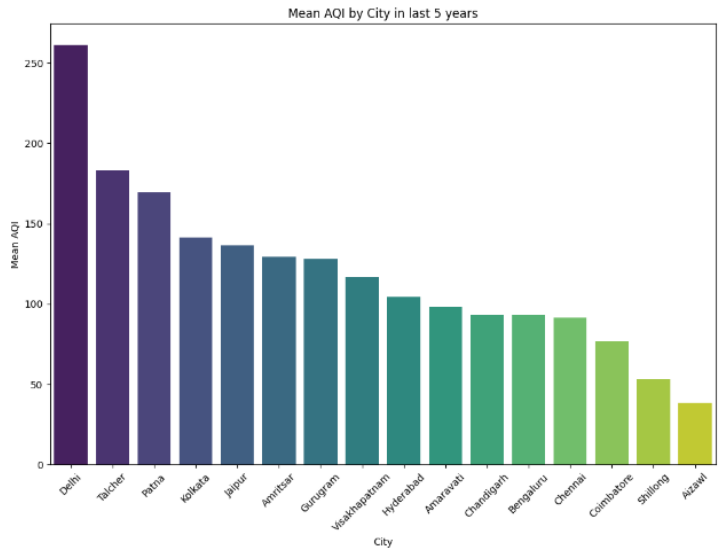






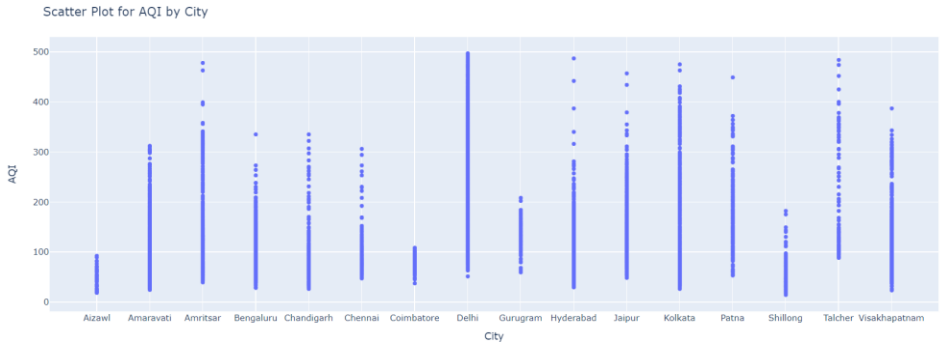


e) Bar graph of mean AQI of different cities over 5 years.

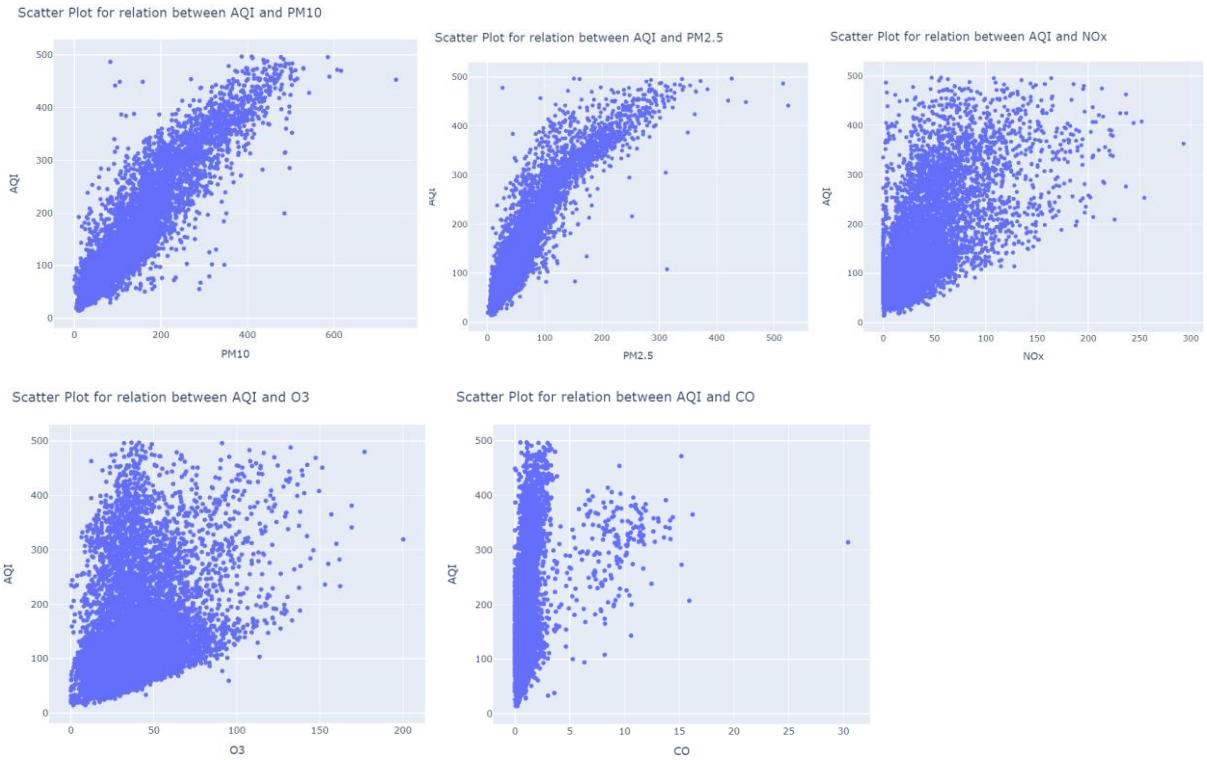


MOST POLLUTED CITY = DELHI, LEAST POLLUTED CITY = AIZAWL

f) Scatter plot for AQI by different Cities

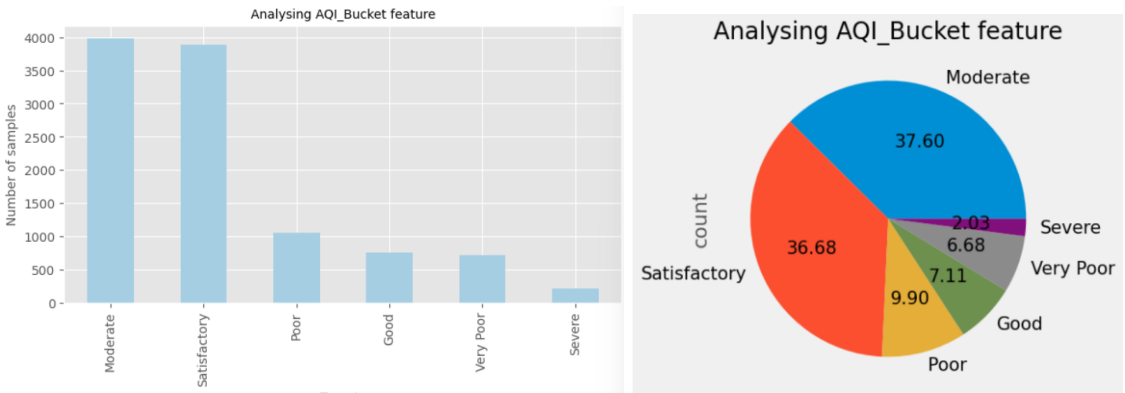


g) Below are correlation scatter plots between different gases and AQI



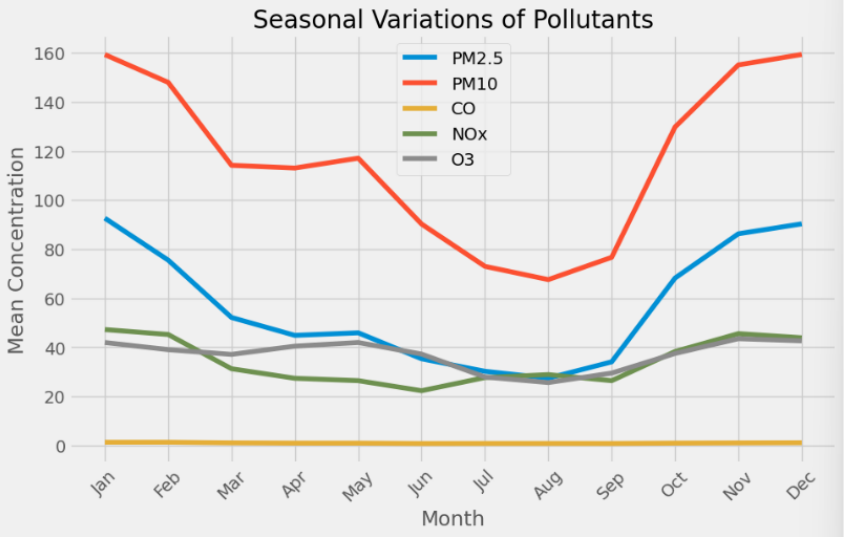
ALL of these correlations successfully prove that AQI is directly proportional to these different gases. And hence proving our feature analysis.

h) Analysing AQI_Bucket



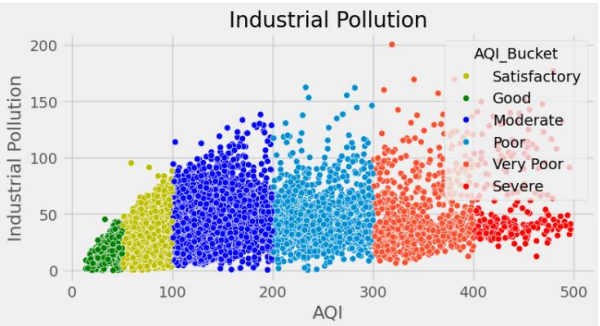
7. NOVEL IDEAS RESULTS:

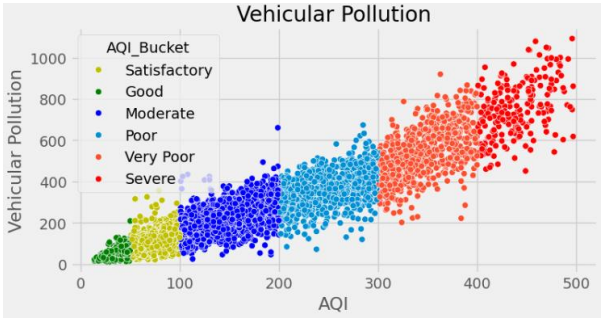
a) Seasonal Variation of Pollutants



This shows that concentration values of these gases tend to dip in rainy season (i.e. May – August) which is logical and true as rain has moisture/water which reduces pollution. This also corelated to the fact that people have less respiratory diseases due to pollution in rainy season.

b) Pollution Source Analysis





This indicates that vehicular source of pollution is musch greater than industrial pollution.

Hence we got to a conclusion that to improve air quality or to reduce air pollution we need to decrease vehicular gases emmisions.

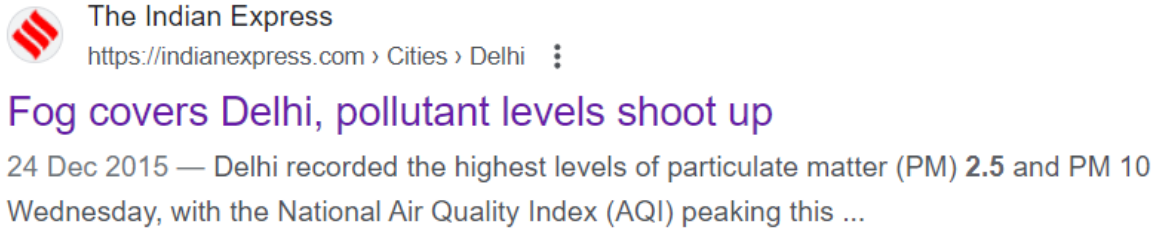
People can reduce air pollution by using Electric Vehicles for travel instead of using traditional diesel/petrol ones.

c) Anomaly Detection

Our model detected anomaly at

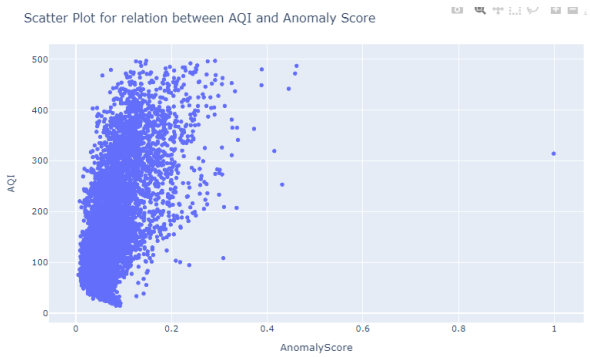
```
Anomalies:
             City    PM2.5    PM10   NOx     CO     O3    AQI AQI_Bucket Month  \
Date
2015-12-24  Delhi  129.08  486.66  53.4  30.44  32.42  314.0  Very Poor   Dec

            AnomalyScore
Date
2015-12-24           1.0
```

This is cross checked by an article of "The Indian Express" –



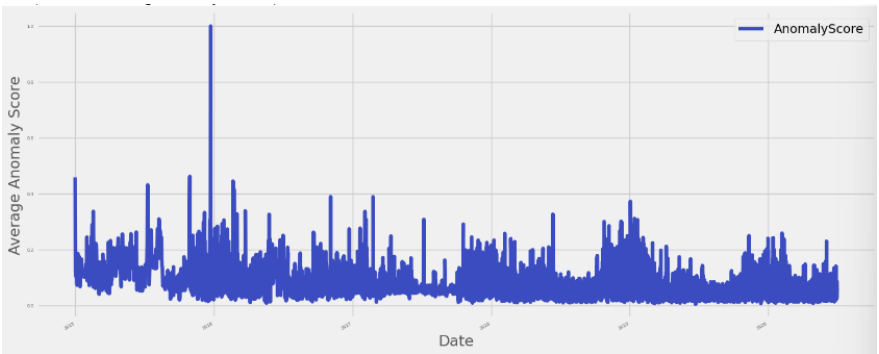
Thus proving our correct working of the model.

i) Below is scatter plot belween Anomaly Score and AQI.



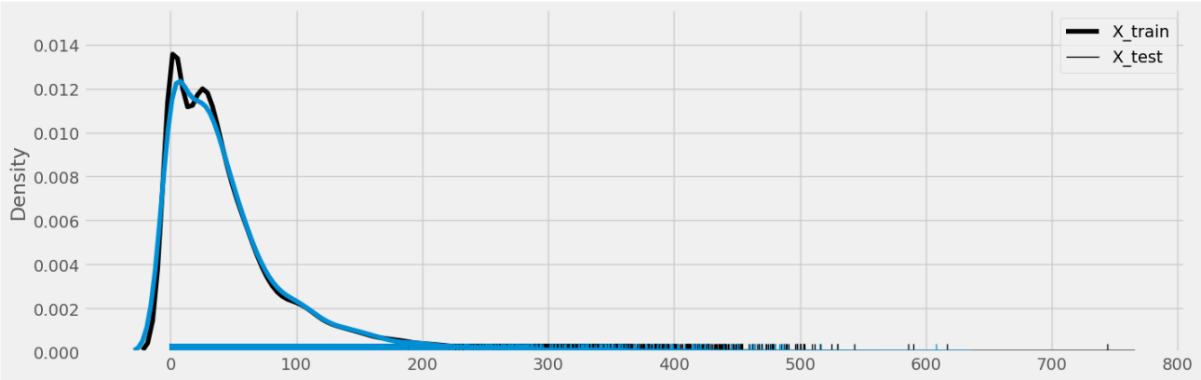
This successfully shows that with increase in AQI, Anomaly also increases.

ii) Below is a line graph showing Average Anomaly Score over a period of 5 years.



Thus successfully showing spike at 24 Dec 2015

8. Splitting the dataset



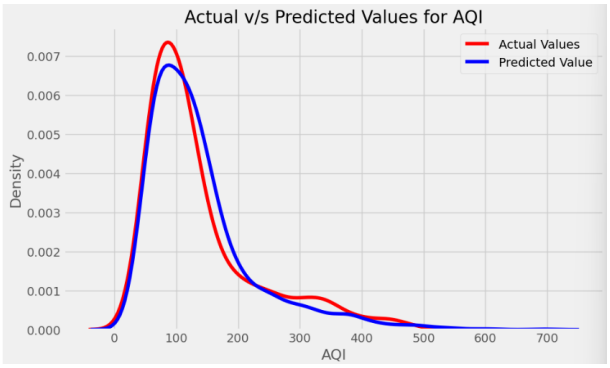
This is the graph showing line chart of X_train and X_test wrt densities.

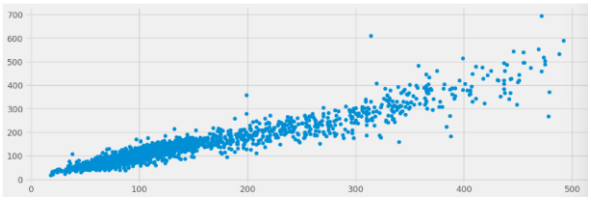
9. Model Training Results

A) PREDICTION

i. Linear Regression:

- RMSE Training data = 27.980
- RMSE Test Data = 28.632
- RSquaredError value on train: 0.907
- RSquaredError value on test: 0.903
- MAE = -18.721
- MSE = -819.844
- Plots





These plots indicate that our model has been able to predict Y values accurately as Y predict is close to Y test.

This is a good fit as it is not undrfitting or overfitting as seem in RSquared Error Values. They are able to generalize on new samples. Thus making a good model.

This model is able to correctly predict AQI value corresponding gas inputs.

```
44s
def predict_aqi_linear_regression():
    pm25 = float(input("Enter PM2.5 concentration: "))
    pm10 = float(input("Enter PM10 concentration: "))
    co = float(input("Enter CO concentration: "))
    nox = float(input("Enter NOx concentration: "))
    o3 = float(input("Enter O3 concentration: "))
    predicted_aqi = model.predict([[pm25, pm10, co, nox, o3]])
    return predicted_aqi[0]

predicted_aqi_linear = predict_aqi_linear_regression()
print("Predicted AQI value using Linear Regression:", predicted_aqi_linear)
```

```
Enter PM2.5 concentration: 1
Enter PM10 concentration: 1
Enter CO concentration: 1
Enter NOx concentration: 1
Enter O3 concentration: 1
Predicted AQI value using Linear Regression: 19.317566439168914
```

```
1m
[78] predicted_aqi_linear = predict_aqi_linear_regression()
print("Predicted AQI value using Linear Regression:", predicted_aqi_linear)
```

```
Enter PM2.5 concentration: 10
Enter PM10 concentration: 10
Enter CO concentration: 10
Enter NOx concentration: 10
Enter O3 concentration: 10
Predicted AQI value using Linear Regression: 112.99712684817305
```

```
30s
[79] predicted_aqi_linear = predict_aqi_linear_regression()
print("Predicted AQI value using Linear Regression:", predicted_aqi_linear)
```

```
Enter PM2.5 concentration: 30
Enter PM10 concentration: 30
Enter CO concentration: 40
Enter NOx concentration: 50
Enter O3 concentration: 50
Predicted AQI value using Linear Regression: 418.71051636226395
```

<u>This is therefore the best model for Prediction that we tested.</u>

ii) Decision Tree Regressor:

- RMSE Training data = 0.0
- RMSE Test Data = 32.270
- RSquaredError value on train: 1.0
- RSquaredError value on test: 0.877
- MAE = -20.874
- MSE = -1041.383

NO THIS IS NOT A BETTER RESULT AS
RMSE on training is 0 and RSquaredError on train = 1
This signifies that it is overfitting on Training data and FAILS to generalize for new examples. Therefore a bad model.

This is evidently shown by below example:

```
def predict_aqi():
    pm25 = float(input("Enter PM2.5 concentration: "))
    pm10 = float(input("Enter PM10 concentration: "))
    co = float(input("Enter CO concentration: "))
    nox = float(input("Enter NOx concentration: "))
    o3 = float(input("Enter O3 concentration: "))
    predicted_aqi = DT.predict([[pm25, pm10, co, nox, o3]])
    return predicted_aqi[0]

predicted_aqi = predict_aqi()
print("Predicted AQI value:", predicted_aqi)
```

```
Enter PM2.5 concentration: 1
Enter PM10 concentration: 1
Enter CO concentration: 1
Enter NOx concentration: 1
Enter O3 concentration: 1
Predicted AQI value: 75.0
```

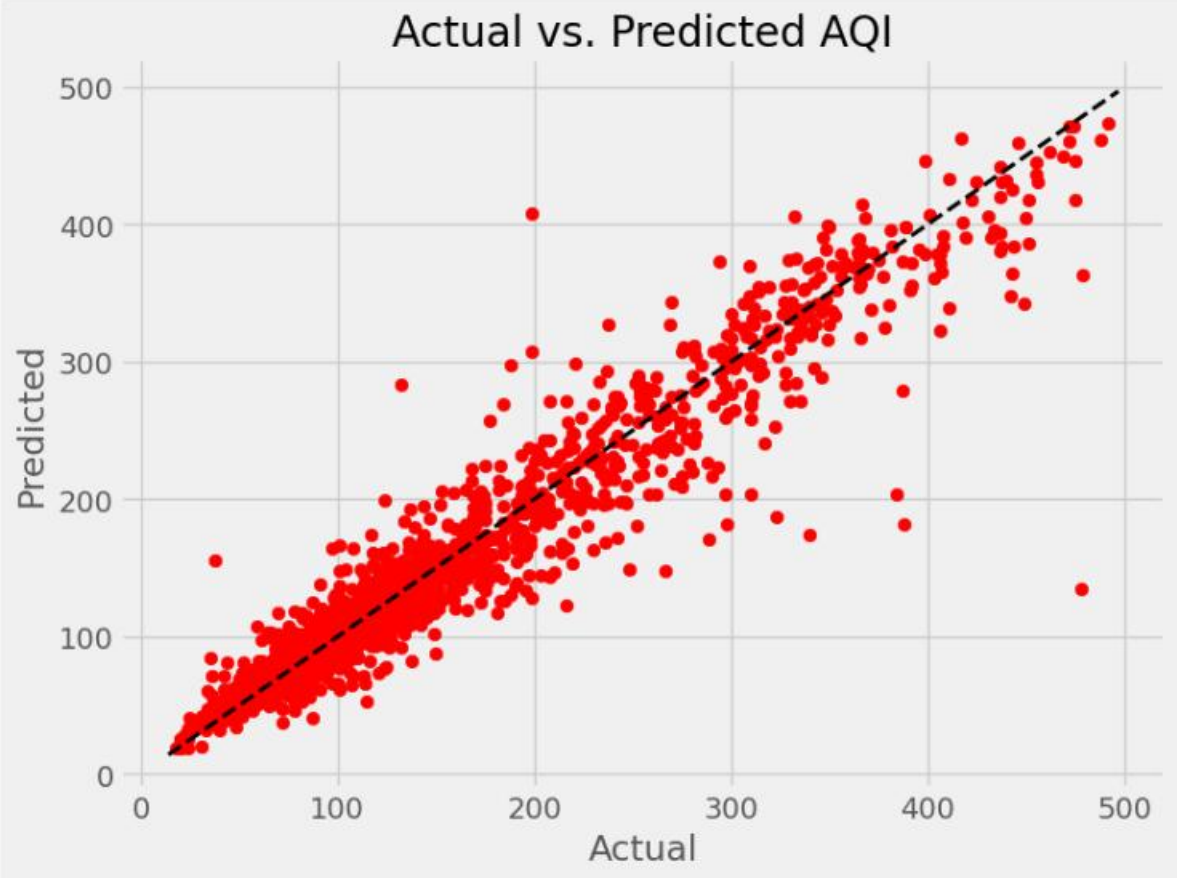
iii) Random Forest Regressor:

RMSE Training data = 8.485
RMSE Test Data = 24.581
RSquaredError value on train: 0.991
RSquaredError value on test: 0.929

MAE = -14.867
MSE = -604.250

NO This is NOT a better model as its RMSE on training data is very low and RSquaredError value on training set is 0.99 which is very close to 1. All this indicated that our model is overfitting and fails to generalize on new samples. Therefore it's a bad model.

This is evident through regression graph of actual vs predicted AQI below –



THIS SHOWS OVERFITTING

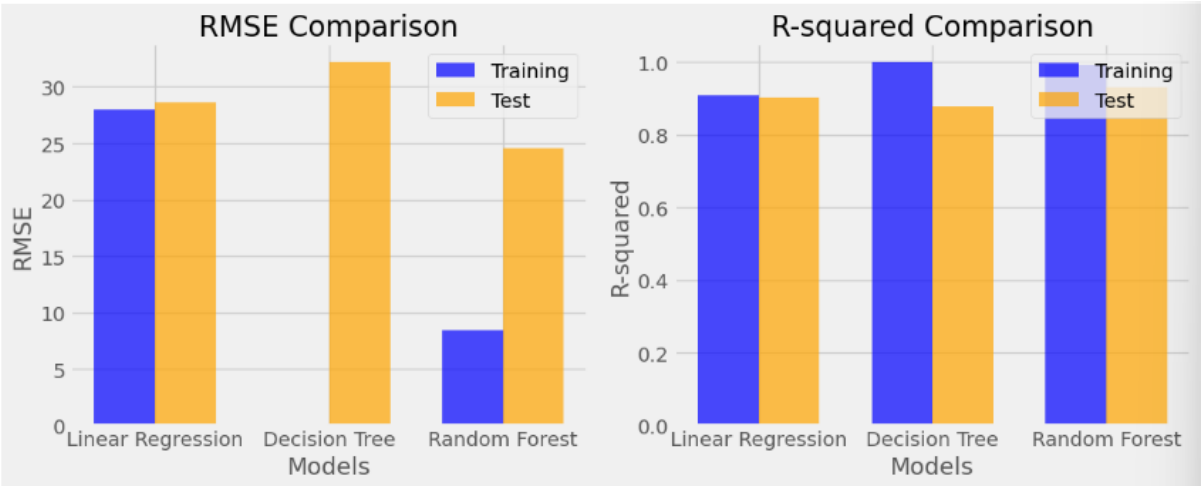
```
[93] def predict_aqi_random_forest():
         pm25 = float(input("Enter PM2.5 concentration: "))
         pm10 = float(input("Enter PM10 concentration: "))
         co = float(input("Enter CO concentration: "))
         nox = float(input("Enter NOx concentration: "))
         o3 = float(input("Enter O3 concentration: "))
         predicted_aqi = RF.predict([[pm25, pm10, co, nox, o3]])
         return predicted_aqi[0]

     predicted_aqi_rf = predict_aqi_random_forest()
     print("Predicted AQI value using Random Forest Regressor:", predicted_aqi_rf)
```

```
Enter PM2.5 concentration: 1
Enter PM10 concentration: 1
Enter CO concentration: 1
Enter NOx concentration: 1
Enter O3 concentration: 1
Predicted AQI value using Random Forest Regressor: 62.17
```

THEREFORE OUR PREDICTED AQI VALUE IS ALSO WRONG

Regression Models Evaluation

Below is bar chart that shows comparision between different evaluation metrics of different models of training and test set.



This shows that for Prediction, Linear Regression is the best model as it is not overfitting and is able to generalize on new samples thus giving correct outputs as its RMSE on training is higher than other two and its R-squared Error on training set is lower than other two.

B) Classification

i) Logistic Regression

Model accuracy on train is: 0.559
Model accuracy on test is: 0.556
KappaScore is: 0.342

Classification Report

	precision	recall	f1-score
Good	0.11	0.01	0.01
Moderate	0.57	0.68	0.62
Poor	0.30	0.15	0.20
Satisfactory	0.64	0.68	0.66
Severe	0.18	0.18	0.18
Very Poor	0.39	0.48	0.43
accuracy			0.56
macro avg	0.36	0.36	0.35
weighted avg	0.52	0.56	0.53

Its accuracy and Kappa Score is very low thus underfitting and making a bad model for classification.
This is evident by the classification output from gas value input.

```
def predict_aqi_category():
    pm25 = float(input("Enter PM2.5 concentration: "))
    pm10 = float(input("Enter PM10 concentration: "))
    co = float(input("Enter CO concentration: "))
    nox = float(input("Enter NOx concentration: "))
    o3 = float(input("Enter O3 concentration: "))
    predicted_aqi_category = log_reg.predict([[pm25, pm10, co, nox, o3]])
    return predicted_aqi_category[0]

predicted_aqi_category = predict_aqi_category()
print("Predicted AQI category using Logistic Regression:", predicted_aqi_category)
```

```
Enter PM2.5 concentration: 1
Enter PM10 concentration: 1
Enter CO concentration: 1
Enter NOx concentration: 1
Enter O3 concentration: 1
Predicted AQI category using Logistic Regression: Satisfactory
```

ii) Decision Tree Classifier

Model accuracy on train is: 1.0
Model accuracy on test is: 0.767
KappaScore is: 0.669

Classification Report

	precision	recall	f1-score
Good	0.68	0.73	0.70
Moderate	0.79	0.82	0.80
Poor	0.68	0.67	0.68
Satisfactory	0.82	0.79	0.80
Severe	0.71	0.55	0.62
Very Poor	0.71	0.74	0.72
accuracy			0.77
macro avg	0.73	0.72	0.72
weighted avg	0.78	0.77	0.77

This model's accuracy on Training set is 1. Therefore overfitting and being unable to generalize on new samples. Therefore, it's a pad model to classify.

This is evident and seen through example below

```
def predict_aqi_category_decision_tree():
    pm25 = float(input("Enter PM2.5 concentration: "))
    pm10 = float(input("Enter PM10 concentration: "))
    co = float(input("Enter CO concentration: "))
    nox = float(input("Enter NOx concentration: "))
    o3 = float(input("Enter O3 concentration: "))
    predicted_aqi_category = DT2.predict([[pm25, pm10, co, nox, o3]])
    return predicted_aqi_category[0]

predicted_aqi_category_dt = predict_aqi_category_decision_tree()
print("Predicted AQI category using Decision Tree Classifier:", predicted_aqi_category_dt)
```

```
Enter PM2.5 concentration: 1
Enter PM10 concentration: 1
Enter CO concentration: 1
Enter NOx concentration: 1
Enter O3 concentration: 1
Predicted AQI category using Decision Tree Classifier: Satisfactory
```

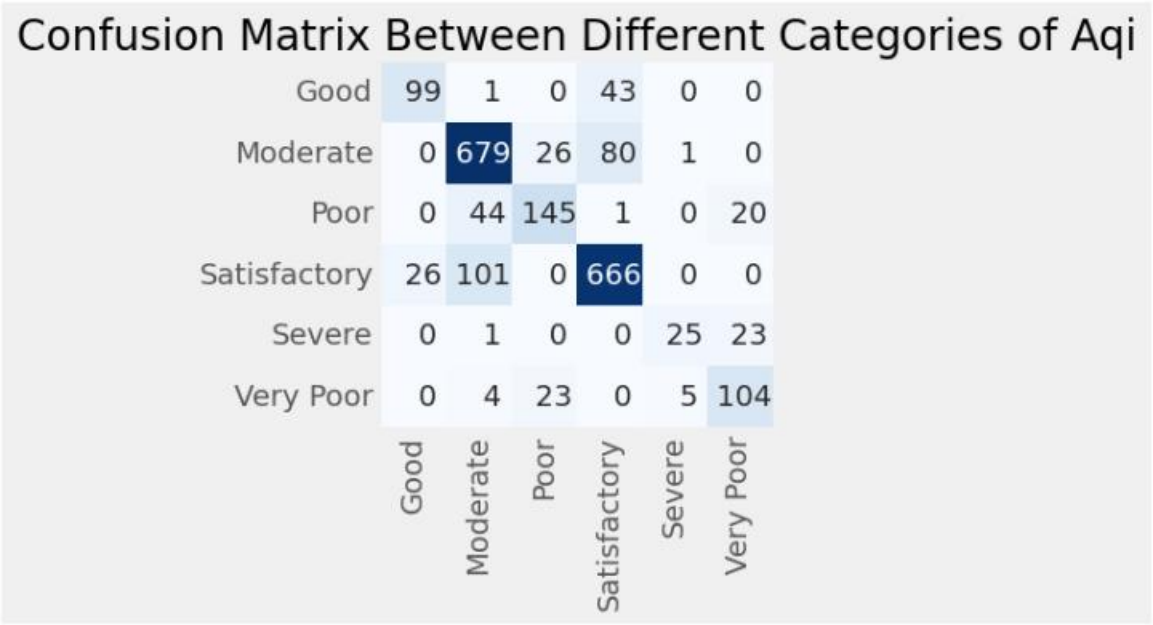
iii) Random Forest Classifier

Model accuracy on train is: 1.0
Model accuracy on test is: 0.809
KappaScore is: 0.726

Classification Report

	precision	recall	f1-score
Good	0.79	0.69	0.74
Moderate	0.82	0.86	0.84
Poor	0.75	0.69	0.72
Satisfactory	0.84	0.84	0.84
Severe	0.81	0.51	0.62
Very Poor	0.71	0.76	0.73
accuracy			0.81
macro avg	0.79	0.73	0.75
weighted avg	0.81	0.81	0.81

This models accuracy on Training set is 1. Therefore overfitting and unable to generalize on new samples. Therfore a bad model for classification



This is seen through below example as it is unable to classify correctly.

```
def predict_aqi_category_random_forest():
    pm25 = float(input("Enter PM2.5 concentration: "))
    pm10 = float(input("Enter PM10 concentration: "))
    co = float(input("Enter CO concentration: "))
    nox = float(input("Enter NOx concentration: "))
    o3 = float(input("Enter O3 concentration: "))
    predicted_aqi_category = RF.predict([[pm25, pm10, co, nox, o3]])
    return predicted_aqi_category[0]

predicted_aqi_category_rf = predict_aqi_category_random_forest()
print("Predicted AQI category using Random Forest Classifier:", predicted_aqi_category_rf)
```

```
Enter PM2.5 concentration: 1
Enter PM10 concentration: 1
Enter CO concentration: 1
Enter NOx concentration: 1
Enter O3 concentration: 1
Predicted AQI category using Random Forest Classifier: Satisfactory
```

iv) KNN Classifier

Model accuracy on train is: 0.847
Model accuracy on test is: 0.788
KappaScore is: 0.696

Classification Report

	precision	recall	f1-score
Good	0.64	0.62	0.63
Moderate	0.81	0.84	0.83
Poor	0.73	0.66	0.69
Satisfactory	0.81	0.82	0.81
Severe	0.84	0.55	0.67
Very Poor	0.73	0.76	0.75
accuracy			0.79
macro avg	0.76	0.71	0.73
weighted avg	0.79	0.79	0.79

This is a very good model as it is having good accuracy on training and test along with having a good Kappa Score without overfitting. Therefore it will be able to generalize on new samples and classify correctly.

<u>This is the best classification sample we tested due to above reason.</u>

This can be successfully seen throgh below example

```
def predict_aqi_category_knn():
    pm25 = float(input("Enter PM2.5 concentration: "))
    pm10 = float(input("Enter PM10 concentration: "))
    co = float(input("Enter CO concentration: "))
    nox = float(input("Enter NOx concentration: "))
    o3 = float(input("Enter O3 concentration: "))
    predicted_aqi_category = KNN.predict([[pm25, pm10, co, nox, o3]])
    return predicted_aqi_category[0]

predicted_aqi_category_knn = predict_aqi_category_knn()
print("Predicted AQI category using K Nearest Neighbors (KNN) Classifier:", predicted_aqi_category_knn)
```

Loading...

```
Enter PM2.5 concentration: 1
Enter PM10 concentration: 1
Enter CO concentration: 1
Enter NOx concentration: 1
Enter O3 concentration: 1
Predicted AQI category using K Nearest Neighbors (KNN) Classifier: Good
```

```
[113] predicted_aqi_category_knn = predict_aqi_category_knn()
      print("Predicted AQI category using K Nearest Neighbors (KNN) Classifier:", predicted_aqi_category_knn)

      Enter PM2.5 concentration: 50
      Enter PM10 concentration: 50
      Enter CO concentration: 50
      Enter NOx concentration: 50
      Enter O3 concentration: 50
      Predicted AQI category using K Nearest Neighbors (KNN) Classifier: Moderate
```

```
[114] predicted_aqi_category_knn = predict_aqi_category_knn()
      print("Predicted AQI category using K Nearest Neighbors (KNN) Classifier:", predicted_aqi_category_knn)

      Enter PM2.5 concentration: 200
      Enter PM10 concentration: 150
      Enter CO concentration: 100
      Enter NOx concentration: 100
      Enter O3 concentration: 100
      Predicted AQI category using K Nearest Neighbors (KNN) Classifier: Very Poor
```

Classification Models Evaluation:

Below is bar chart that shows comparision between different evaluation metrics of different models of training and test set.

These show that KNN is the best model for classifying as –

It has a really good Kappa Score (One of the highest) and It has a good amount of training and test accuracy unlike other who are either overfitting or underfitting as shown by their accuracy .

Therefore as KNN is a balance, it is not overfitting or underfitting and is able to generalize on new samples and thus making it good for classifying.

Final Results as compared to previous research papers:

Random Forest -

Name of the algorithm	Accuracy in percentage (%)
Naïve Bayes (NB)	86.663
Support vector machine (SVM)	92.40
Artificial neural network (ANN)	84–93
Gradient boost (GB)	96
Decision tree (DT)	91.9978
Enhanced k-means	71.28
Support vector regression (SVR)	99.4
Random forest regression (RFR)	99.985
CatBoost regression (CR)	99.88

This is from HINDAWI research paper and I chose Random Forest Regressor because it had the highest accuracy from all other methods

For Predicting AQI

When I ran it in my model, this performs really well in training set giving 99% R squared error and gave 92% R squared error on test set. This indicates , this is very good model for samples within our dataset as it it tending to overfit, but for new/general samples this becomes a bad model.

For Classifying AQI

When I ran it in my model, this performs really well in training set giving 100% accuracy on my training set and 81% accuracy in test set. This is good for samples within our dataset as it tends to overfit but for new samples it becomes a bad model.

Overall my model fails to correctly predict/classify samples as it tend to overfit. And thus unable to perform well on new samples

For Classifying AQI — Decision Tree Classifier | Regressor

S. no.	City	Accuracy			
		Support vector regression	K-nearest neighbor	Random forest regression	Proposed GWO-DT
1	New Delhi	84.83	83.68	84.73	88.98
2	Bangalore	87.18	89.47	90.31	91.49
3	Kolkata	91.56	90.65	93.74	94.48
4	Hyderabad	93.57	93.68	97.61	97.66
5	Chennai	92.65	93.48	94.68	95.22
6	Visakhapatnam	92.24	92.11	95.65	97.68

Table 8. Performance comparison with conventional algorithms.

2.The RMSE value on test set of Regression of our model came out to be:-
32.27047507620295
which is an accepted value considering the dataset.
R_Squared value on test set on Regression is coming:-
0.8777340003635178
But on the training set the value of R_Squared is coming :-
1.0
which certainly says that the model is overfitting.

1.Our model was certainly different from the model that was used in the research paper.
The research paper analyzed all the metrics such as RMSE, R Squared error etc. for the particular cities, whereas our model tends to classify a huge number of data of cities of India.

3. For Classifier the accuracy on train is:-
1.0
which is overfitting
For classifier the accuracy on test is:-
0.7675956542276807
this is better value, but we cannot accept it becuase the case of overfitting is applied here.

For Classifying AQI — K Nearest Neighbours

Table 4 Comparison of model results in the training set

Model	Accuracy	Precision	Recall	F1-score	Training time (in seconds)
KNN	89	94	90	96	0.104
GNB	85	91	94	88	0.110
SVM	**81**	90	93	88	0.258
RF	88	93	88	92	0.102
XGBoost	**91**	95	95	91	0.532

Achieving an 84% accuracy in a training set and 78% in test set implies that the model correctly predicts the class of the samples around without overfitting or underfitting

Table 6 Comparison of model results in the testing set

Model	Accuracy	Precision	Recall	F1-Score	Prediction time (in seconds)
KNN	85	92	85	94	0.018
GNB	83	88	89	92	0.016
SVM	**78**	91	90	83	0.027
RF	86	92	91	90	0.023
XGBoost	90	96	95	91	0.041

KNN model is able to perform well for new samples (i.e. able to generalize well) as it is not overfitting and provides a best fit line.

9. CONCLUSION:

From all the models we tested (Linear Regression, Logistic Regression, Decision Tree Regressor, Random Forest Regressor, Logistic Regression, Decision Tree Regressor, Random Forest Regressor, K Nearest Neighbours) **Linear Regression** and **KNN** are giving best output for prediction and classification respectively.

For Prediction:

- **LINEAR REGRESSION –** We are choosing this as this is neither overfitting or underfitting like the others and is providing a best fit and therefore being able to generalize on new samples and giving a good balance of RMSE and RMSE being –
  RMSE (train) = 27
  RMSE (test) = 28
  RSE (train) = 0.90
  RSE(test) = 0.90
  RMSE is low but not too low & RSE is high but not too high. This indicates it's a good balance and not overfitting and is further proved with testing samples.

For Classification:

- **KNN –** We are choosing this as this is neither overfitting or underfitting like the others and is providing a best fit and therefore being able to generalize on new samples and giving a good balance of
  Accuracy and Kappa Score being –
  Accuracy (train) = 0.84
  Accuracy (test) = 0.78
  Kappa Score = 0.69
  Accuracy is good not too high on both training and test set indicating it is not overfitting or underfitting and is being able to generalize on new samples.

Linear Regression can successfully be used to predict AQI values based on input of pollutant gases.

KNN can successfully be used to classify AQI values in categories like good, satisfactory, moderate, poor, very poor, severe based on input of pollutant gases.

10. Future Scope:

Some exciting possibilities for expanding our air quality prediction and classification project:

1. Integration of Addition Data Sources:
   Incorporating additional data sources such as meteorological data, satellite imagery, and geographical features could enhance the accuracy and robustness of the air quality prediction and classification models.
2. Real-time Monitoring and Alerts:
   Implementing a real-time monitoring system using IoT devices and sensors could enable continuous monitoring of air quality parameters. Integration with mobile applications or online platforms could provide users with real-time updates and alerts about air quality conditions in their surroundings.
3. Geospatial Analysis and Visualization:
   Leveraging geospatial analysis techniques and visualization tools can provide spatial insights into air quality patterns and trends across different regions. This could facilitate targeted interventions and policy decisions at local and regional levels to mitigate air pollution.
4. Predictive Analytics for Policy Planning:
   Utilizing predictive analytics to forecast future air quality trends and assess the potential impact of policy interventions could aid policymakers in designing effective strategies for air quality management and pollution control. This could include scenario analysis and predictive modeling to evaluate the effectiveness of different policy measures.
5. Health Impact Assessment:
   Integrating our AQI predictions with data on health outcomes like respiratory illnesses. This could help estimate the potential health risks associated with different air quality levels.

11. References:

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

2. Kumar, K. S., & Pande, B. P. (2022). Air pollution prediction with machine learning: a case study of Indian cities. International Journal of Environmental Science and Technology, 20(5), 5333–5348. https://doi.org/10.1007/s13762-022-04241-5

3. Gupta, N. S., Mohta, Y., Heda, K., Armaan, R., Valarmathi, B., & Arulkumaran, G. (2023). Prediction of air quality Index using Machine Learning techniques: A Comparative analysis. Journal of Environmental and Public Health, 2023, 1–26. https://doi.org/10.1155/2023/4916267

4. Natarajan, S. K., Shanmurthy, P., Arockiam, D., Balusamy, B., & Selvarajan, S. (2024). Optimized machine learning model for air quality index prediction in major cities in India. Scientific Reports, 14(1). https://doi.org/10.1038/s41598-024-54807-1

5. Murukonda, V. S. N. M., & Gogineni, A. C. (2022). Prediction of air quality index using supervised machine learning. DIVA. https://urn.kb.se/resolve?urn=urn:nbn:se:bth-23439

6. Ahuja, A. (2019, December 18). Air pollution: What is air quality index, how is it measured and its health impact. NDTV-Dettol Banega Swasth Swachh India. https://swachhindia.ndtv.com/air-pollution-what-is-air-quality-index-how-is-it-measured-and-its-health-impact-40387/

7. https://www.divaportal.org/smash/get/diva2:1681590/FULLTEXT02.pdf

8. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9107909/pdf/13762_2022_Article_4241.pdf

9. https://nature.com/scientificreports

10. https://towardsdatascience.com/unsupervised-anomaly-detection-on-spotify-data-k-means-vs-local-outlier-factor-f96ae783d7a7

11. https://jovian.com/najo1on1/exploring-indias-air-quality-2015-to-2020

12. https://swachhindia.ndtv.com/air-pollution-what-is-air-quality-index-how-is-it-measured-and-its-health-impact-40387/

13. https://nature.com

14. https://images.app.goo.gl/sQXeHLKRW9DpGzcH8

15. https://www.w3schools.com/PYTHON/

16. https://www.javatpoint.com/python-tutorial

17. https://www.youtube.com/@machinelearningandai3274

18. https://www.youtube.com/@SebastianRaschka

19. https://www.aqi.in/in/dashboard/india

20. https://www.iqair.com/in-en/india

12. Appendix:

```
import numpy as np
import pandas as pd

from google.colab import files
uploaded = files.upload()

df=pd.read_csv('city_day.csv',parse_dates = ["Date"])
df
```

SETTING DATE COLUMN AS INDEX

```
df['Date']=df['Date'].apply(pd.to_datetime)
df.set_index('Date',inplace=True)

df.shape

df.describe()

df.info()

df.nunique()
```

DATA CLEANING

```
import seaborn as sns
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')\

df.isnull().sum().sort_values(ascending=False)

(df.isnull().sum() / df.shape[0] * 100).sort_values(ascending=False)

df.drop(['Xylene'], axis=1, inplace=True)

df
```

OUTLIER DETECTION AND REMOVAL

```
import matplotlib.pyplot as plt

for feature in df.select_dtypes(include='number').columns:
  data = df.copy()
  if feature != "AQI_Bucket":
      data.boxplot(column=feature)
      plt.ylabel('Value')
      plt.title(feature)
```

```
    plt.show()
```

```
df.shape()
```

```
df= df[df['PM2.5']<801]
df= df[df['PM10']<811]
df= df[df['NO']<311]
df= df[df['NO2']<301]
df= df[df['NOx']<311]
df= df[df['NH3']<326]
df= df[df['CO']<126]
df= df[df['SO2']<201]
df= df[df['O3']<211]
df= df[df['Benzene']<321]
df= df[df['Toluene']<301]
df= df[df['AQI']<501]
```

```
df.shape()
```

```
df2 = df.copy()
```

```
df2
```

```
sns.heatmap(df2.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

FEATURE IMPORTANCE ANALYSIS

```
import pandas as pd

from sklearn.ensemble import RandomForestRegressor

import matplotlib.pyplot as plt


features = df2[['PM10','PM2.5', 'NO', 'NO2', 'NOx', 'CO', 'SO2', 'O3', 'Benzene',
'Toluene']]

target = df2['AQI']


rf_model = RandomForestRegressor(n_estimators=100, random_state=42)


rf_model.fit(features, target)


feature_importances = rf_model.feature_importances_


importance_df2 = pd.DataFrame({'Feature': features.columns, 'Importance':
feature_importances})


importance_df2 = importance_df2.sort_values(by='Importance',
ascending=False)


plt.figure(figsize=(10, 6))
```

```
plt.barh(importance_df2['Feature'], importance_df2['Importance'])

plt.xlabel('Importance')

plt.title('Feature Importance Analysis')

plt.show()


# Therefore now after feature analysis we will only select top five features i.e. PM2.5, PM10, C0, 03 and NOx for our model

# now we got to know which ones have most influence now well drop rest features directly in our original dataframe

df


df.drop(['NO'], axis=1, inplace=True)

df.drop(['NO2'], axis=1, inplace=True)

df.drop(['SO2'], axis=1, inplace=True)

df.drop(['NH3'], axis=1, inplace=True)

df.drop(['Benzene'], axis=1, inplace=True)

df.drop(['Toluene'], axis=1, inplace=True)


UPDATED DATA


df


sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

```
df.columns

unique_values = df['City'].unique()
print(unique_values)

print(len(unique_values))

df['City'].value_counts()
```

DATA VISUALIZATION AND GRAPHS

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

numeric_df = df.select_dtypes(include='number')
plt.figure(figsize=(10, 6))
sns.heatmap(numeric_df.corr(), cmap='coolwarm', annot=True)
plt.title('Correlation Heatmap between different gases and AQI', pad=20)
plt.show()

df_city_day = df.copy()

pollutants = ['PM2.5', 'PM10', 'CO', 'NOx','O3',]
```

```
df_city_day = df_city_day[pollutants]


print('Distribution of different pollutants in last 5 years')

df_city_day.plot(kind='line',figsize=(18,18),cmap='coolwarm',subplots=True,fo
ntsize=10)


df3_city_day = df.copy()


aqi_level = ['AQI']

df3_city_day = df3_city_day[aqi_level]

df3_city_day.plot(kind='line',figsize=(15,5),cmap='coolwarm',subplots=True,fo
ntsize=5)

plt.ylabel('AVG AQI')


import matplotlib.pyplot as plt


plt.rcParams['figure.figsize'] = (15, 10)

df[['PM2.5', 'City']].groupby(["City"]).mean().sort_values(by='PM2.5',
ascending = False).plot.bar(color='coral')

plt.ylabel('Average PM2.5 values')

plt.title('Bar graph for average PM2.5 values of corresponding cities over 5
years', pad=20)

plt.show()


import matplotlib.pyplot as plt
```

```
plt.rcParams['figure.figsize'] = (15, 7)

df[['PM10', 'City']].groupby(["City"]).mean().sort_values(by='PM10', ascending = False).plot.bar(color='lightblue')

plt.ylabel('Average PM10 values')

plt.title('Bar graph for average PM10 values of corresponding cities over 5 years', pad=20)

plt.show()


import matplotlib.pyplot as plt


plt.rcParams['figure.figsize'] = (15, 5)

df[['O3', 'City']].groupby(["City"]).mean().sort_values(by='O3', ascending = False).plot.bar(color='hotpink')

plt.ylabel('Average O3 values')

plt.title('Bar graph for average O3 values of corresponding cities over 5 years', pad=20)

plt.show()


import matplotlib.pyplot as plt


plt.rcParams['figure.figsize'] = (15, 5)

df[['CO', 'City']].groupby(["City"]).mean().sort_values(by='CO', ascending = False).plot.bar(color='grey')

plt.ylabel('Average CO values')
```

```
plt.title('Bar graph for average CO values of corresponding cities over 5 years', pad=20)

plt.show()



import matplotlib.pyplot as plt



plt.rcParams['figure.figsize'] = (15, 5)

df[['NOx', 'City']].groupby(["City"]).mean().sort_values(by='NOx', ascending = False).plot.bar(color='teal')

plt.ylabel('Average NOx values')

plt.title('Bar graph for average NOx values of corresponding cities over 5 years', pad=20)

plt.show()



import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns



city_grouped = df.groupby('City')



city_stats = city_grouped.agg({

   'AQI': 'mean',

   'PM2.5': 'mean',

   'PM10': 'mean',

   'CO': 'mean',
```

```
    'NOx': 'mean',
    'O3': 'mean'
}).reset_index()

city_stats_sorted = city_stats.sort_values(by='AQI', ascending=False)

plt.figure(figsize=(12, 8))
sns.barplot(data=city_stats_sorted, x='City', y='AQI', palette='viridis')
plt.title('Mean AQI by City in last 5 years')
plt.xticks(rotation=45)
plt.xlabel('City')
plt.ylabel('Mean AQI')
plt.show()

import plotly.express as px

fig = px.scatter(df, x="City", y="AQI")
fig.update_layout(title='Scatter Plot for AQI by City')
fig.show()

import plotly.express as px

fig2= px.scatter(df, x='PM10', y='AQI')
```

```
fig2.update_layout(title='Scatter Plot for relation between AQI and PM10')
fig2.show()

import plotly.express as px

fig3= px.scatter(df, x='PM2.5', y='AQI')
fig3.update_layout(title='Scatter Plot for relation between AQI and PM2.5')
fig3.show()

import plotly.express as px

fig4= px.scatter(df, x='NOx', y='AQI')
fig4.update_layout(title='Scatter Plot for relation between AQI and NOx')
fig4.show()

import plotly.express as px

fig5= px.scatter(df, x='O3', y='AQI')
fig5.update_layout(title='Scatter Plot for relation between AQI and O3')
fig5.show()

import plotly.express as px
```

```
fig6= px.scatter(df, x='CO', y='AQI')
fig6.update_layout(title='Scatter Plot for relation between AQI and CO')
fig6.show()

from google.colab import files
from IPython.display import Image

uploaded = files.upload()

file_names = list(uploaded.keys())
file_name = file_names[0]

width = 400
height = 300

Image(file_name, width=width, height=height)

class_labels=df["AQI_Bucket"].unique().tolist()
class_labels.sort()
print(class_labels)

df['AQI_Bucket'].value_counts()
```

```
import matplotlib.pyplot as plt


plt.style.use('ggplot')

df["AQI_Bucket"].value_counts().plot.bar(fontsize=10, figsize=(8, 4),
colormap="Paired")


plt.title('Analysing AQI_Bucket feature', fontsize=10)

plt.xlabel('Targets', fontsize=10)

plt.ylabel('Number of samples', fontsize=10)

plt.show()


import matplotlib.pyplot as plt


plt.style.use('fivethirtyeight')

df["AQI_Bucket"].value_counts().plot.pie(fontsize=15, figsize=(5, 5),
autopct="%.2f")


plt.title('Analysing AQI_Bucket feature', fontsize=20)

plt.show()
```

NOVEL IDEAS

SEASONAL VARIATION OF POLLUTANTS

```
import pandas as pd
import matplotlib.pyplot as plt

month_names = {
   1: 'Jan', 2: 'Feb', 3: 'Mar', 4: 'Apr', 5: 'May', 6: 'Jun',
   7: 'Jul', 8: 'Aug', 9: 'Sep', 10: 'Oct', 11: 'Nov', 12: 'Dec'
}

df.index = pd.to_datetime(df.index)

df['Month'] = df.index.month.map(month_names)

numeric_columns = ['PM2.5', 'PM10', 'CO', 'NOx', 'O3']
df_numeric = df[['Month'] + numeric_columns]

monthly_mean = df_numeric.groupby('Month').mean()

months_order = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct',
'Nov', 'Dec']

monthly_mean = monthly_mean.reindex(months_order)

plt.figure(figsize=(10, 6))
for pollutant in numeric_columns:
```

```
    plt.plot(monthly_mean.index, monthly_mean[pollutant], label=pollutant)

plt.xlabel('Month')
plt.ylabel('Mean Concentration')
plt.title('Seasonal Variations of Pollutants')
plt.xticks(rotation=45)
plt.legend()
plt.grid(True)
plt.show()
```

ANOMALY DETECTION

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

X = df[['PM2.5','PM10','CO','NOx','O3']].values

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

kmeans = KMeans(n_clusters=5)
```

```
kmeans.fit(X_scaled)

cluster_labels = kmeans.predict(X_scaled)

distances = kmeans.transform(X_scaled).min(axis=1)

anomaly_scores = distances / distances.max()

df['AnomalyScore'] = anomaly_scores

anomaly_threshold = 0.95

anomalies = df[df['AnomalyScore'] > anomaly_threshold]

print("Anomalies:")

print(anomalies)

import plotly.express as px

fig7= px.scatter(df, x='AnomalyScore', y='AQI')

fig7.update_layout(title='Scatter Plot for relation between AQI and Anomaly Score')

fig7.show()
```

```
import matplotlib.pyplot as plt


average_anomaly_sorted = df[['City',
'AnomalyScore']].groupby('City').mean().sort_values('AnomalyScore',
ascending=False)


plt.figure(figsize=(3, 3))

average_anomaly_sorted.plot(kind='bar', cmap='Blues_r')

plt.title('Average Anomaly in last 5 years')

plt.xlabel('City')

plt.ylabel('Average Anomaly Score')

plt.show()


df_city_day2 = df.copy()


df


anomaly_score = ['AnomalyScore']

df_city_day2 = df_city_day2[anomaly_score]

df_city_day2.plot(kind='line',figsize=(15,7),cmap='coolwarm',subplots=True,fo
ntsize=5)

plt.ylabel('Average Anomaly Score')
```

POLLUTION SOURCE ANALYSIS

```
pollution_df = df.copy()

pollution_df['Vehicular Pollution'] = pollution_df['PM2.5'] +
pollution_df['PM10'] + pollution_df['NOx'] + pollution_df['CO']

pollution_df['Industrial Pollution'] = pollution_df['O3']

pollution_df.drop(['PM2.5', 'PM10', 'NOx', 'CO', 'O3'], axis=1, inplace=True)


def pollution_based_scatter(y):

  plt.figure(figsize=(8, 4))

  plt.title(y)

  plt.legend(loc='best')

  palette = {'Good': "g", 'Poor': "C0", 'Very Poor': "C1", 'Severe': "Red",
"Moderate": 'b', "Satisfactory": 'y'}

  sns.scatterplot(x='AQI', y=y, data=pollution_df, hue='AQI_Bucket',
palette=palette)


pollution_based_scatter('Industrial Pollution')

pollution_based_scatter('Vehicular Pollution') # Corrected column name here
```

SPLITTING DATASET INTO TRAINING AND TEST SETS

```
X = df[['PM2.5','PM10','CO','NOx','O3']]

Y = df['AQI']

X.head()


Y.head()
```

```
from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test=train_test_split(X, Y, test_size = 0.2,
random_state = 70)

print(X_train.shape,X_test.shape,Y_train.shape,Y_test.shape)


import seaborn as sns

sns.distplot(X_train, color = "black", hist = False, rug = True, bins = 100)

sns.distplot(X_test, hist = False, rug = True, bins = 100)

plt.legend(labels=["X_train", "X_test"])
```

APPLYING MODELS

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import warnings

warnings.filterwarnings ("ignore")

from sklearn.preprocessing import LabelEncoder

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor
```

```
from sklearn import metrics

from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score

from sklearn.metrics import accuracy_score, confusion_matrix, f1_score,
classification_report
```

PREDICTION MODELS

LINEAR REGRESSION

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()

model.fit(X_train, Y_train)


train_pred = model.predict(X_train)

test_pred = model.predict(X_test)


RMSE_train = (np.sqrt(metrics.mean_squared_error(Y_train,train_pred)))

RMSE_test = (np.sqrt(metrics.mean_squared_error(Y_test,test_pred)))

print("RMSE Training data = ", str(RMSE_train))

print("RMSE Test Data = ",str(RMSE_test))

print("-"*50)


print("RSquaredError value on train:", model.score(X_train, Y_train))
```

```
print("RSquaredError value on test:", model.score(X_test, Y_test))


plt.figure(figsize = (10,6))

ax = sns.distplot(Y_test, hist = False, color = "r", label = "Actual Values")

sns.distplot(test_pred, hist = False, color = "b", label = "Predicted Value",
ax=ax)

plt.title ('Actual v/s Predicted Values for AQI')


plt.legend()

plt.show()

plt.close()


model.intercept_


model_coef_


coef_df=pd.DataFrame(model.coef_,X.columns,columns=["Coefficient"])

coef_df


plt.scatter(Y_test,test_pred)


sns.distplot((Y_test-test_pred),bins=50)


print(f"MAE:-{mean_absolute_error(Y_test,test_pred)}")
```

```
print(f"MSE:-{mean_squared_error(Y_test,test_pred)}")


def predict_aqi_linear_regression():
    pm25 = float(input("Enter PM2.5 concentration: "))
    pm10 = float(input("Enter PM10 concentration: "))
    co = float(input("Enter CO concentration: "))
    nox = float(input("Enter NOx concentration: "))
    o3 = float(input("Enter O3 concentration: "))
    predicted_aqi = model.predict([[pm25, pm10, co, nox, o3]])
    return predicted_aqi[0]

predicted_aqi_linear = predict_aqi_linear_regression()
print("Predicted AQI value using Linear Regression:", predicted_aqi_linear)

predicted_aqi_linear = predict_aqi_linear_regression()
print("Predicted AQI value using Linear Regression:", predicted_aqi_linear)
```

DECISION TREE REGRESSOR

```
DT = DecisionTreeRegressor()
DT.fit(X_train, Y_train)
```

```
train_preds = DT.predict(X_train)
test_preds = DT.predict(X_test)

RMSE_train = (np.sqrt(metrics.mean_squared_error(Y_train,train_preds)))
RMSE_test = (np.sqrt(metrics.mean_squared_error(Y_test,test_preds)))

print("RMSE Training data = ", str(RMSE_train))
print("RMSE Test Data = ",str(RMSE_test))
print("-"*50)

print("RSquaredError value on train:", DT.score(X_train, Y_train))
print("RSquaredError value on test:", DT.score(X_test, Y_test))

print(f"MAE:-{mean_absolute_error(Y_test,test_preds)}")

print(f"MSE:-{mean_squared_error(Y_test,test_preds)}")

def predict_aqi():
   pm25 = float(input("Enter PM2.5 concentration: "))
   pm10 = float(input("Enter PM10 concentration: "))
   co = float(input("Enter CO concentration: "))
   nox = float(input("Enter NOx concentration: "))
   o3 = float(input("Enter O3 concentration: "))
```

```
    predicted_aqi = DT.predict([[pm25, pm10, co, nox, o3]])
    return predicted_aqi[0]

predicted_aqi = predict_aqi()
print("Predicted AQI value:", predicted_aqi)
```

RANDOM FOREST REGRESSOR

```
RF= RandomForestRegressor().fit(X_train, Y_train)

train_preds1 = RF.predict(X_train)
test_preds1 = RF.predict(X_test)

print(test_preds1)

RMSE_train = (np.sqrt(metrics.mean_squared_error(Y_train,train_preds1)))
RMSE_test = (np.sqrt(metrics.mean_squared_error(Y_test,test_preds1)))

print("RMSE Training data = ", str(RMSE_train))
print("RMSE Test Data = ",str(RMSE_test))
print("-"*50)
```

```
print("RSquaredError value on train:", RF.score(X_train, Y_train))
print("RSquaredError value on test:", RF.score(X_test, Y_test))

fig, ax = plt.subplots(figsize=(8, 6))
ax.scatter(Y_test, test_preds1, cmap="cividis", color="red")
ax.plot([Y.min(), Y.max()], [Y.min(), Y.max()], 'k--', lw=2)
ax.set_xlabel('Actual')
ax.set_ylabel('Predicted')
ax.set_title('Actual vs. Predicted AQI')
plt.show()

print(f"MAE:-{mean_absolute_error(Y_test,test_preds1)}")

print(f"MSE:-{mean_squared_error(Y_test,test_preds1)}")

def predict_aqi_random_forest():
    pm25 = float(input("Enter PM2.5 concentration: "))
    pm10 = float(input("Enter PM10 concentration: "))
    co = float(input("Enter CO concentration: "))
    nox = float(input("Enter NOx concentration: "))
    o3 = float(input("Enter O3 concentration: "))
    predicted_aqi = RF.predict([[pm25, pm10, co, nox, o3]])
    return predicted_aqi[0]
```

predicted_aqi_rf = predict_aqi_random_forest()

print("Predicted AQI value using Random Forest Regressor:", predicted_aqi_rf)

REGRESSION MODELS EVALUATION/ANALYSIS

import numpy as np

import matplotlib.pyplot as plt

RF_RMSE_train = 8.432566793162845

RF_RMSE_test = 24.529180439961944

RF_R2_train = 0.9915813035787213

RF_R2_test = 0.9293583132017508

DT_RMSE_train = 0.0

DT_RMSE_test = 32.18729141853344

DT_R2_train = 1.0

DT_R2_test = 0.8783635185056798

LR_RMSE_train = 27.980371825373254

LR_RMSE_test = 28.63292286966045

LR_R2_train = 0.9073101387005679

LR_R2_test = 0.9037443232621972

```
labels = ['Linear Regression', 'Decision Tree', 'Random Forest']

num_models = len(labels)

bar_width = 0.35

index = np.arange(num_models)

plt.figure(figsize=(12, 5))


plt.subplot(1, 2, 1)

plt.bar(index - bar_width/2, [LR_RMSE_train, DT_RMSE_train,
RF_RMSE_train], bar_width, color='blue', alpha=0.7, label='Training')

plt.bar(index + bar_width/2, [LR_RMSE_test, DT_RMSE_test,
RF_RMSE_test], bar_width, color='orange', alpha=0.7, label='Test')

plt.xlabel('Models')

plt.ylabel('RMSE')

plt.title('RMSE Comparison')

plt.xticks(index, labels)

plt.legend()


plt.subplot(1, 2, 2)

plt.bar(index - bar_width/2, [LR_R2_train, DT_R2_train, RF_R2_train],
bar_width, color='blue', alpha=0.7, label='Training')

plt.bar(index + bar_width/2, [LR_R2_test, DT_R2_test, RF_R2_test],
bar_width, color='orange', alpha=0.7, label='Test')

plt.xlabel('Models')

plt.ylabel('R-squared')
```

```
plt.title('R-squared Comparison')

plt.xticks(index, labels)

plt.legend()


plt.tight_layout()

plt.show()
```

CLASSIFICATION MODELS

SPLITTING DATA INTO TRAINING AND TEST SETS

```
from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier


X2 = df[['PM2.5','PM10','CO','NOx','O3']]

Y2 = df['AQI_Bucket']


X_train2, X_test2, Y_train2, Y_test2 = train_test_split(X2, Y2, test_size=0.2,
random_state=70)
```

LOGISTIC REGRESSION

```
log_reg = LogisticRegression().fit(X_train2, Y_train2)

train_preds2 = log_reg.predict(X_train2)

print("Model accuracy on train is:", accuracy_score (Y_train2, train_preds2))

test_preds2 = log_reg.predict(X_test2)

print("Model accuracy on test is:", accuracy_score(Y_test2, test_preds2))
print('-'*50)

print('KappaScore is:', metrics.cohen_kappa_score (Y_test2, test_preds2))

from sklearn.metrics import classification_report
print("               Classification Report")
print("")
cr = classification_report(Y_test2, test_preds2)
print(cr)

def predict_aqi_category():
   pm25 = float(input("Enter PM2.5 concentration: "))
   pm10 = float(input("Enter PM10 concentration: "))
   co = float(input("Enter CO concentration: "))
```

```
    nox = float(input("Enter NOx concentration: "))
    o3 = float(input("Enter O3 concentration: "))
    predicted_aqi_category = log_reg.predict([[pm25, pm10, co, nox, o3]])
    return predicted_aqi_category[0]

predicted_aqi_category = predict_aqi_category()
print("Predicted AQI category using Logistic Regression:",
predicted_aqi_category)
```

DECISION TREE CLASSIFIER

```
DT2 = DecisionTreeClassifier().fit(X_train2, Y_train2)

train_preds3 = DT2.predict(X_train2)

print("Model accuracy on train is:", accuracy_score(Y_train2, train_preds3))

test_preds3 = DT2.predict(X_test2)

print("Model accuracy on test is:", accuracy_score(Y_test2, test_preds3))
print('-'*50)

print('KappaScore is:', metrics.cohen_kappa_score(Y_test2, test_preds3))
```

```
from sklearn.metrics import classification_report

print("               Classification Report")

print("")

cr = classification_report(Y_test2, test_preds3)

print(cr)


def predict_aqi_category_decision_tree():

   pm25 = float(input("Enter PM2.5 concentration: "))

   pm10 = float(input("Enter PM10 concentration: "))

   co = float(input("Enter CO concentration: "))

   nox = float(input("Enter NOx concentration: "))

   o3 = float(input("Enter O3 concentration: "))

   predicted_aqi_category = DT2.predict([[pm25, pm10, co, nox, o3]])

   return predicted_aqi_category[0]


predicted_aqi_category_dt = predict_aqi_category_decision_tree()

print("Predicted AQI category using Decision Tree Classifier:",
predicted_aqi_category_dt)
```

RANDOM FOREST CLASSIFIER

```
RF=RandomForestClassifier().fit(X_train2, Y_train2)


train_preds4 = RF.predict(X_train2)
```

```
print("Model accuracy on train is:", accuracy_score (Y_train2, train_preds4))

test_preds4 = RF.predict(X_test2)

print("Model accuracy on test is:", accuracy_score(Y_test2, test_preds4))

print('-'*50)

print('KappaScore is:', metrics.cohen_kappa_score(Y_test2, test_preds4))

RF_pred=RF.predict(X_test2)
print(RF_pred.tolist())

print(Y_test2.tolist())

plt.figure(figsize=(3,3))
sns.heatmap(data=confusion_matrix(y_true=Y_test2, y_pred=RF_pred),
        annot=True,
        fmt='4d',
        cbar=False,
        cmap=plt.cm.Blues,
        xticklabels=class_labels,
```

```
        yticklabels=class_labels)
plt.title(label='Confusion Matrix Between Different Categories of Aqi')
plt.show()


from sklearn.metrics import classification_report
print("               Classification Report")
print("")
cr = classification_report(Y_test2, test_preds4)
print(cr)


def predict_aqi_category_random_forest():
   pm25 = float(input("Enter PM2.5 concentration: "))
   pm10 = float(input("Enter PM10 concentration: "))
   co = float(input("Enter CO concentration: "))
   nox = float(input("Enter NOx concentration: "))
   o3 = float(input("Enter O3 concentration: "))
   predicted_aqi_category = RF.predict([[pm25, pm10, co, nox, o3]])
   return predicted_aqi_category[0]


predicted_aqi_category_rf = predict_aqi_category_random_forest()
print("Predicted AQI category using Random Forest Classifier:",
predicted_aqi_category_rf)
```

K NEAREST NEIGHBOURS

```
KNN = KNeighborsClassifier().fit(X_train2, Y_train2)

train_preds5 = KNN.predict(X_train2)

print("Model accuracy on train is:", accuracy_score(Y_train2, train_preds5))

test_preds5 = KNN.predict(X_test2)

print("Model accuracy on test is:", accuracy_score(Y_test2, test_preds5))
print('-'*50)

print('KappaScore is:', metrics.cohen_kappa_score(Y_test2, test_preds5))

from sklearn.metrics import classification_report
print("               Classification Report")
print("")
cr = classification_report(Y_test2, test_preds5)
print(cr)

def predict_aqi_category_knn():
   pm25 = float(input("Enter PM2.5 concentration: "))
   pm10 = float(input("Enter PM10 concentration: "))
```

```
    co = float(input("Enter CO concentration: "))
    nox = float(input("Enter NOx concentration: "))
    o3 = float(input("Enter O3 concentration: "))
    predicted_aqi_category = KNN.predict([[pm25, pm10, co, nox, o3]])
    return predicted_aqi_category[0]

predicted_aqi_category_knn = predict_aqi_category_knn()
print("Predicted AQI category using K Nearest Neighbors (KNN) Classifier:", predicted_aqi_category_knn)

predicted_aqi_category_knn = predict_aqi_category_knn()
print("Predicted AQI category using K Nearest Neighbors (KNN) Classifier:", predicted_aqi_category_knn)

predicted_aqi_category_knn = predict_aqi_category_knn()
print("Predicted AQI category using K Nearest Neighbors (KNN) Classifier:", predicted_aqi_category_knn)
```

CLASSIFICATION MODELS EVALUATION/ANALYSIS

```
import matplotlib.pyplot as plt
import numpy as np

train_accuracy = [0.559938585095075, 1.0, 1.0, 0.8470532656194638]
```

```
test_accuracy = [0.5564478034955125, 0.7699574870099197,
0.810108644307983, 0.788379782711384]

kappa_scores = [0.3425322323627409, 0.6726627853186109,
0.7272327797330834, 0.6964489867507531]


models = ['Logistic Regression', 'Decision Tree', 'Random Forest', 'K-Nearest
Neighbors']


num_models = len(models)


bar_width = 0.25


index = np.arange(num_models)


plt.figure(figsize=(12, 5))


plt.bar(index - bar_width, train_accuracy, bar_width, color='blue', alpha=0.7,
label='Train Accuracy')

plt.bar(index, test_accuracy, bar_width, color='orange', alpha=0.7, label='Test
Accuracy')

plt.bar(index + bar_width, kappa_scores, bar_width, color='green', alpha=0.7,
label='Kappa Score')


plt.xlabel('Models')

plt.ylabel('Scores')

plt.title('Model Performance Comparison')
```

```
plt.xticks(index, models)
plt.legend()

plt.tight_layout()
plt.show()
```

END OF REPORT