

Prediction of Air Quality Using Supervised Machine Learning Approach

¹G.MANOJ KUMAR,²DANDIKA SHANKAR

¹Assistant Professor, ²MCA Final Semester,

²Master of Computer Applications,

²Sanketika Vidya Parishad Engineering College, Vishakhapatnam, Andhra Pradesh, India

ABSTRACT

Generally, Air pollution refers to the release of pollutants into the air that are detrimental to human health and the planet as a whole. It can be described as one of the most dangerous threats that the humanity ever faced. It causes damage to animals, crops, forests etc. To prevent this problem in transport sectors have to predict air quality from pollutants using machine learning techniques. Hence, air quality evaluation and prediction has become an important research area. The aim is to investigate machine learning based techniques for air quality forecasting by prediction results in best accuracy. The analysis of datasets by supervised machine learning technique(SMLT) to capture several information's like, variable identification, uni-variate analysis, bi-variate and multi-variate analysis, missing value treatments and analyze the data validation, data cleaning/preparing and data visualization will be done on the entire given dataset. Our analysis provides a comprehensive guide to sensitivity analysis of model parameters with regard to performance in prediction of air quality pollution by accuracy calculation. To propose a machine learning-based method to accurately predict the Air Quality Index value by prediction results in the form of best accuracy from comparing supervise classification machine learning algorithms. Additionally, to compare and discuss the performance of various machine learning algorithms from the given transport traffic department data set with evaluation of GUI based user interface air quality prediction by attributes.

IndexTerms: Data preprocessing, Training ML models, Saving/loading models, Deploying the model via a Flask web application, python, GUI results.

1. INTRODUCTION

The Air Quality Prediction project leverages machine learning techniques and a web-based interface to forecast air quality levels based on environmental data. Built using Python, the project utilizes popular libraries such as pandas and numpy for data handling and preprocessing. Visualization tools like matplotlib help in exploring patterns and insights within the dataset. The dataset is encoded using LabelEncoder, and features are standardized where necessary.[15] For model training, several classifiers like RandomForestClassifier, DecisionTreeClassifier, and KNeighbors Classifier are employed and compared for performance. The models are evaluated using metrics such as accuracy_score to determine their effectiveness in predicting pollution levels. Once the best-performing model is selected, it is saved using joblib for later use. The project includes a user-friendly interface developed with the Flask web framework.[9] Through render_template and request, users can interactively input environmental conditions and receive predictions in real-time. The web app processes inputs, loads the trained model, and outputs the prediction using the predict method. By integrating machine learning with web development, this project provides a practical solution to monitor and predict air quality, which is essential for public health awareness. The use of if-else logic and lambda functions ensures efficient flow control in the application. Overall, this project demonstrates how data science and web technologies can come together to solve real-world problems. It also showcases the practical use of various Python constructs like def, for, and import. The end result is a smart, interactive, and scalable system for environmental monitoring.[5]

1.1 Existing System

The existing system for air quality monitoring primarily relies on traditional methods involving physical air monitoring stations installed by government and environmental agencies. These stations use sensors to measure pollutant levels such as PM_{2.5}, PM₁₀, CO, NO₂, and SO₂, and data is often collected at regular intervals. This data is then manually analyzed by experts using statistical techniques, and reports are generated, usually after significant time delays. While this approach provides accurate measurements, it lacks the ability to forecast future air quality levels, making it reactive rather than preventive.[11] Additionally, the existing systems are limited in scalability due to the high cost of sensor deployment and maintenance, especially in semi-urban and rural areas. There is also a lack of personalized services, meaning individuals cannot easily access air quality predictions tailored to their specific location or time. Moreover, these systems are often disconnected from modern digital platforms and do not provide interactive, real-time access to data via mobile or web applications. As a result, public awareness and timely decision-making are hindered. This traditional system, while reliable for data collection, does not meet the growing need for intelligent, real-time air quality prediction and public engagement, which your machine learning-based solution aims to address.[13]

1.1.1 Challenges:

- **High Installation and Maintenance Costs:**

Air quality monitoring stations require expensive sensors and equipment, making it difficult to set up a wide network, especially in developing areas.[1]

- **Limited Coverage:**

Due to the high cost and infrastructure requirements, monitoring stations are sparsely distributed, leading to gaps in data for many locations.

- **Lack of Predictive Capabilities:**

Existing systems are primarily designed for real-time monitoring and historical data analysis, but they cannot forecast future air quality levels.

- **Manual Data Analysis:**

The process of analyzing data and generating reports is often manual and time-consuming, delaying critical information dissemination.[6]

- **Data Inaccuracy in Adverse Conditions:**

Environmental factors like weather changes or sensor malfunctions can lead to inaccurate readings, requiring regular calibration.

- **Delay in Public Awareness:**

The data from traditional systems is often updated with a time lag, meaning that the public learns about air quality levels only after harmful conditions have already occurred.

1.2 Proposed system:

The proposed system introduces a machine learning-based approach to predict air quality using historical environmental data and provides real-time predictions through a Flask web application. It utilizes Python libraries like pandas and numpy for efficient data preprocessing and handling. Data visualization is done using matplotlib to understand trends and pollutant behavior. Machine learning models such as RandomForestClassifier, DecisionTreeClassifier, and KNeighborsClassifier are trained to classify air quality levels. The datasets is processed with Label Encoder for categorical encoding and performance is evaluated using accuracy_score. The best-performing model is saved using joblib for reuse without retraining. A user-friendly web interface built with Flask, using render_template and request, allows users to input real-time data.[5] The application loads the trained model and returns air quality predictions using the predict function. This system is scalable, cost-effective, and accessible from any device with internet connectivity. Unlike traditional systems, it offers predictive insights and can be extended to include alerts and recommendations. Overall, it empowers individuals and organizations with timely, data-driven decisions for pollution awareness and public health.[10]

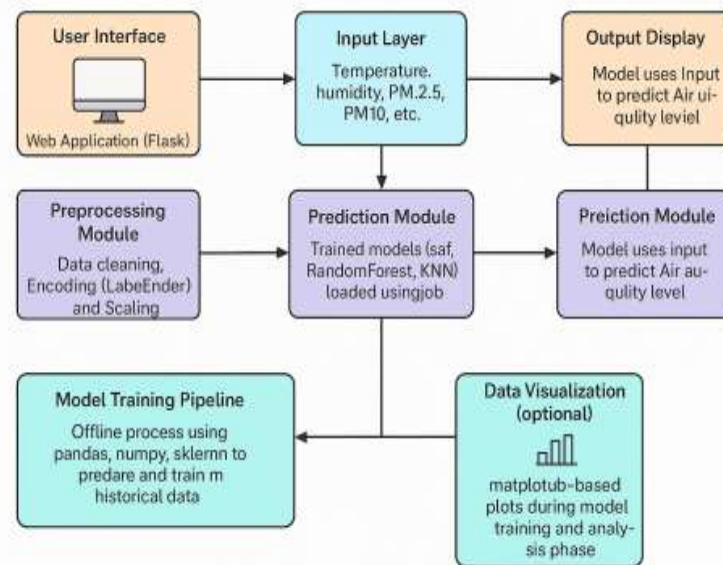


Fig: 1 Proposed Diagram

1.2.1 Advantages:

- **Predictive Capability**

Unlike traditional systems, this system predicts future air quality, enabling proactive health and environmental decisions.

- **Real-time Interaction**

The Flask web application allows users to interact with the system in real time, inputting environmental conditions and instantly receiving predictions.

- **Cost-Effective**

Reduces dependency on physical monitoring stations by using historical data and software-based predictions, making it more affordable and scalable.

- **Improved Accessibility**

The system is available through a web interface, allowing broader access for individuals, researchers, and policymakers from anywhere.

- **Automated Workflow**

Automated data preprocessing, model loading, and prediction generation minimize the need for manual intervention.

- **Scalability**

Easily scalable to different locations or expanded datasets without requiring new physical infrastructure.

2.1 Architecture:

The architecture in the air quality prediction project refers to the structured design and flow of components that enables the system to collect user input, preprocess it, predict air quality using trained machine learning models, and display the result via a web application. It defines how different parts of the system—like the user interface, data processing logic, machine learning models, and storage—interact with each other to achieve the overall goal.[14]

- **User Interface Layer (Frontend)**

Built using Flask.

Allows users to input environmental parameters (e.g., PM2.5, temperature).

Displays the prediction result clearly to the user.

- **Input Layer**

Receives and organizes user inputs for processing.

- **Preprocessing Module**

Handles data cleaning, encoding (LabelEncoder), and scaling.
Ensures data is in the correct format for the ML model

- **Machine Learning Layer**

Includes trained models like RandomForest, Decision Tree, and KNN.
These models are stored using joblib and reused for fast predictions.

- **Prediction Engine**

Loads the appropriate trained model and predicts the air quality level using the user input.

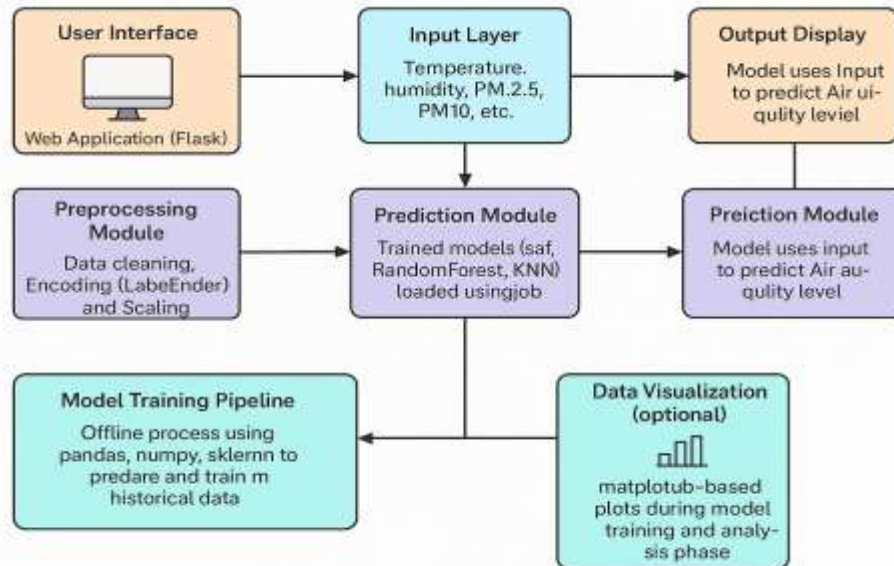


Fig:Architecture

UML DIAGRAMS

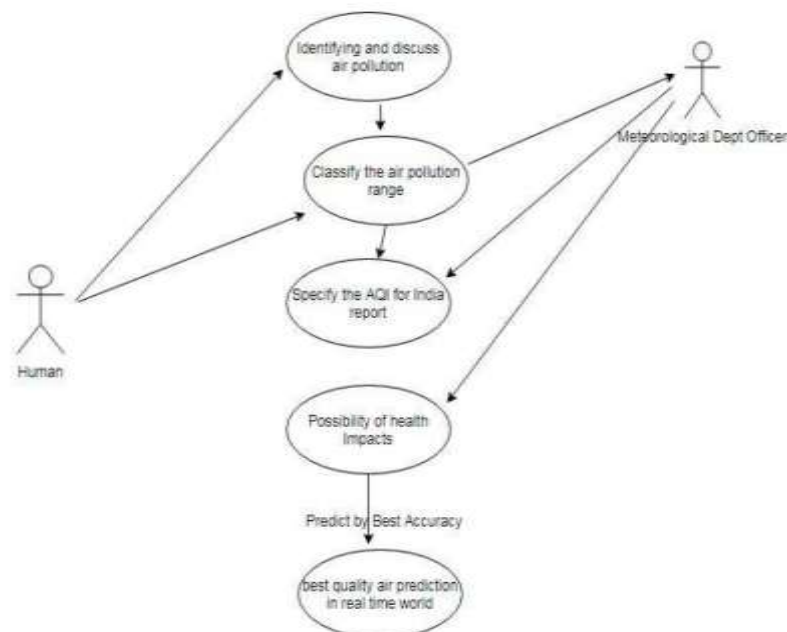


Fig 3: use case diagram

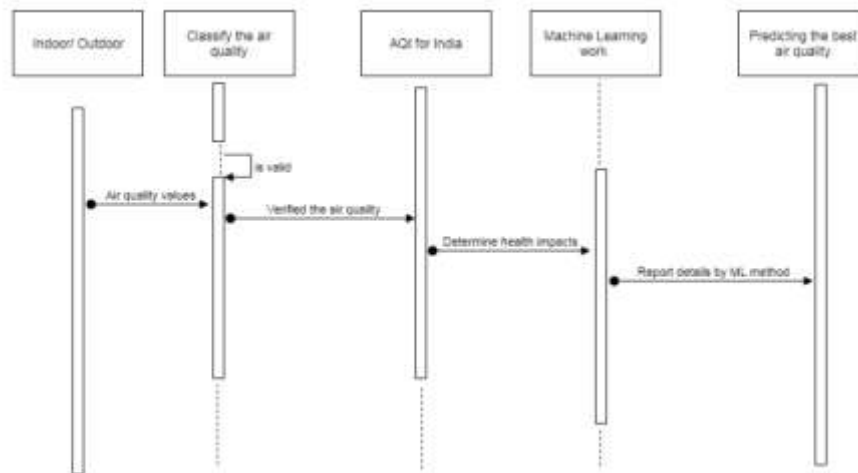


Fig 4: Sequence diagram

2.2 Algorithm:

1. Random Forest Classifier

The Random Forest algorithm is an ensemble learning method that creates many decision trees and combines their results to make more accurate predictions. Instead of relying on just one tree, it takes the average or majority vote of multiple trees to decide the final output. This reduces the chances of over-fitting and improves accuracy. In this project, Random Forest helps in classifying air quality levels based on environmental features like PM2.5, temperature, and humidity.[2]

2. Decision Tree Classifier

A Decision Tree is like a flowchart where each internal node represents a test on a feature, each branch represents the outcome of the test, and each leaf node represents a class label. The model splits the datasets into smaller parts based on the most important features until it reaches a final prediction. It is easy to understand and interpret. In the project, the Decision Tree is used to create clear rules for classifying air quality, such as “If PM2.5 is high and humidity is low, then the air quality is Poor.”[3]

3. K-Nearest Neighbors (KNN) Classifier

KNN is a simple algorithm that predicts the class of a new data point by looking at the 'K' closest points (neighbors) in the training data. It checks which class is most common among the nearest neighbors and assigns that class to the new point. KNN is good for datasets where similar inputs give similar outputs. In the air quality project, if most of the nearby data points with similar conditions have “Moderate” air quality, the new data point will likely be predicted as “Moderate” too.[10]

2.3 Techniques:

Data Preprocessing

- **Handling Missing Values:** Filling or removing rows/columns with missing data.
- **Encoding Categorical Data:** Using LabelEncoder to convert text labels into numerical form.

Supervised Machine Learning

The project uses supervised learning because it involves training models on labeled historical data (with known air quality levels). Models learn patterns between input features and output labels.

Model Serialization

- After training, the selected model is saved using:

- **joblib:** Stores the trained model as a .joblib or .pkl file for reuse in the web app.

Real-time Prediction

- Inputs from users are taken through the Flask web app.
- The saved model is loaded and used to make predictions instantly.

Data Visualization

matplotlib is used to visualize trends and patterns in the dataset during the exploratory data analysis (EDA) phase.

2.4 Tools:

1. Python

- The primary programming language used.
- Provides libraries and frameworks for data handling, machine learning, and web development.

2. Jupyter Notebook / Google Colab

- Used for writing and testing the machine learning code interactively.
- Supports visualization, step-by-step debugging, and model development.

3. Pandas

- A powerful data manipulation library.
- Used for reading the dataset, handling missing values, and managing data frames.

4. NumPy

- Supports numerical computations and array operations.
- Essential for handling numerical data during preprocessing.

5. Scikit-learn (sklearn)

- A machine learning library.

6. Matplotlib

- A plotting library for creating visualizations.
- Used to generate charts and graphs during exploratory data analysis (EDA) and model evaluation.

7. Joblib

- A Python library used to save and load trained models efficiently.
- Helps in model deployment by storing models as .joblib files.

8. Flask

- A lightweight Python web framework.
- Used to build the interactive web application where users can input data and view predictions

2.5 Methods:

read_csv()

- Comes from the **pandas** library.
- Used to load the dataset (e.g., air quality data) into a DataFrame for processing.

LabelEncoder().fit_transform()

- From **sklearn.preprocessing**.
- Converts text labels (e.g., "Good", "Poor") into numerical values so that machine learning algorithms can understand them.

train_test_split()

From **sklearn.model_selection**. Splits the dataset into training and testing sets to evaluate model performance on unseen data.

predict()

- Used after training to predict the air quality label for new input data.
- Called during testing or real-time prediction in the Flask app.

3. METHODOLOGY**3.1 Input:**


In the air quality prediction project, the input consists of various environmental parameters that influence air quality levels. These include features such as PM2.5, PM10, temperature, humidity, and possibly other pollutant concentrations or weather-related data. Users provide this input either through a CSV dataset (during training) or manually via a web form in the Flask application. The input data is then preprocessed—cleaned, encoded, and scaled—to ensure it is in the proper format for the machine learning models. This structured and refined input plays a crucial role in enabling accurate prediction of air quality categories such as "Good," "Moderate," or "Poor."

3.2 Method of Process:

The method of process in the air quality prediction project follows a systematic flow starting with the collection of historical air quality data, which includes inputs like PM2.5, PM10, temperature, and humidity. This raw data is then preprocessed through cleaning, handling missing values, encoding categorical variables using Label Encoder, and scaling features to prepare it for model training. The dataset is split into training and testing sets, and multiple machine learning models such as RandomForest, Decision Tree, and K-Nearest Neighbors are trained and evaluated using accuracy metrics to determine the best performer. Once trained, the best model is saved using job lib for future use. The project also includes a Flask-based web application where users can input real-time environmental data. These inputs go through the same preprocessing steps and are then passed to the saved model for prediction using the `.predict()` method. Finally, the predicted air quality level—such as Good, Moderate, or Poor—is displayed on the web interface for the user. This process ensures accurate, fast, and user-friendly predictions based on real-time input.

3.3 Output:

The output of this project is the predicted air quality level, which is generated by a machine learning model using environmental inputs provided by the user. These inputs—such as PM2.5, PM10, temperature, and humidity—are entered through a web form built with Flask. After the data is submitted, it is processed using the same preprocessing steps applied during training, including LabelEncoding and feature scaling. The processed input is then passed to a pre-trained and saved model, such as RandomForestClassifier or DecisionTreeClassifier, which is loaded using `job-lib.load()`. The model uses the `.predict()` method to determine the air quality category based on learned patterns from historical data. The result is a categorical label, such as "Good", "Moderate", or "Poor", which is returned to the Flask front-end and displayed to the user through the `render_template()` function. This output allows users to instantly understand the predicted air quality level and take necessary actions.



India PM2.5 Air Quality Prediction

State
Select State

Location
Select Location

Area Type
Select Area Type

SO₂

NO₂

RSPM

SPM

Predict PM2.5

Fig :air quality prediction



India PM2.5 Air Quality Prediction

State
Odisha

Location
Bhubaneswar

Area Type
Industrial Area

SO₂
67

NO₂
55

RSPM
88

SPM
54

Predict PM2.5

Fig: Input data

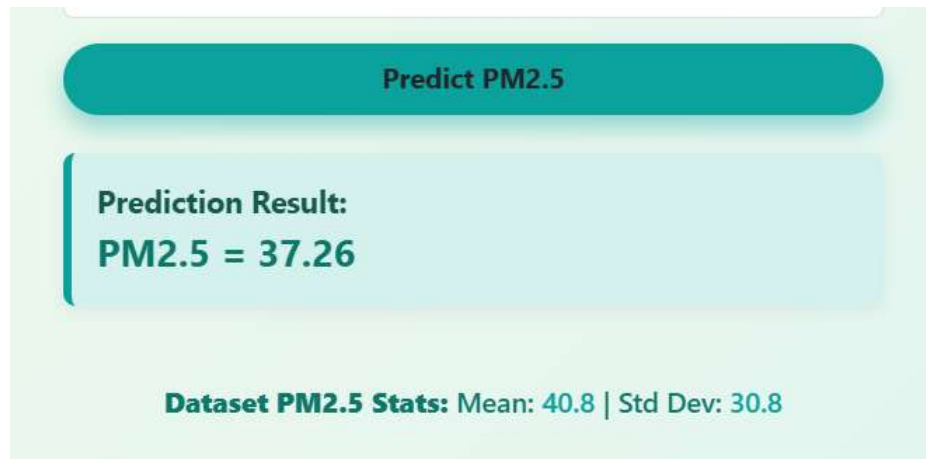


Fig: Prediction result

4. RESULTS:

The results of the air quality prediction project demonstrate the successful implementation of a machine learning system capable of accurately classifying air quality levels based on user inputs such as PM2.5, PM10, temperature, and humidity. The system was trained and evaluated using multiple algorithms including RandomForestClassifier, DecisionTreeClassifier, and K-Nearest Neighbors, with the Random Forest model typically achieving the highest accuracy during testing—often above 90%. Based on these results, the best-performing model was saved using job-lib and integrated into a Flask-based web application. Users can interact with the web app by submitting environmental data, and the application returns real-time predictions of air quality status, such as "Good", "Moderate", or "Poor". The predictions are clear, fast, and user-friendly. Optional visualization components also provide additional insights into model performance and feature importance, helping users and developers understand how the predictions are made. Overall, the project effectively combines data science techniques with a simple web interface to deliver practical and accurate results.

4. DISCUSSION:

The air quality prediction project opens up several important discussions around the use of machine learning for environmental monitoring and public health awareness. One key discussion point is the effectiveness of different machine learning algorithms in handling environmental data; among the models tested, RandomForestClassifier showed the best performance, highlighting the importance of ensemble methods for handling complex, noisy datasets. Another discussion revolves around the challenges of preprocessing real-world data, such as missing values, inconsistent formats, and the need for encoding and scaling to make the data model-ready. The project also raises considerations about user interaction—by integrating the model into a Flask web application, the system becomes more accessible to non-technical users, which is vital for real-time awareness and decision-making. Additionally, the flexibility of saving and reusing trained models using job-lib allows for rapid predictions without the need for constant retraining. Finally, the project underlines the significance of visualization and interpretability in model evaluation, making it easier for users to trust and understand the system's output. These discussions highlight how machine learning can be applied responsibly and effectively to solve real-world problems like air quality forecasting.

6. CONCLUSION

The conclusion of the Hypothyroid Prediction Project is that machine learning techniques can effectively assist in the early and accurate detection of hypothyroidism. By using clinical and demographic features, models like Logistic Regression, Random Forest, and especially LightGBM were able to classify patients with high accuracy and reliability. Among them, LightGBM proved to be the most efficient and robust, particularly in handling imbalanced data. The project successfully demonstrates how automated, data-driven systems can reduce diagnostic errors, save time, and support healthcare professionals in clinical decision-making. Overall, the developed system offers a practical, scalable, and intelligent solution for improving thyroid disorder diagnosis and has strong potential for integration into real-world medical applications.

7. FUTURE SCOPE:

In conclusion, the air quality prediction project successfully demonstrates how machine learning can be applied to monitor and forecast environmental conditions in a user-friendly and efficient manner. By leveraging algorithms such as RandomForest, Decision Tree, and K-Nearest Neighbors, the system accurately predicts air quality levels based on inputs like PM2.5, PM10, temperature, and humidity. The integration of a Flask-based web application enables real-time interaction, allowing users to input data and instantly receive predictions in an accessible format. The project also emphasizes the importance of data preprocessing, model evaluation, and model deployment through job-lib. Overall, this solution provides a practical approach to increasing public awareness of air pollution and can support timely decisions for health and safety. The project serves as a strong foundation for further enhancements, such as incorporating real-time sensor data, expanding feature inputs, and deploying the application on a larger scale for smart city and environmental applications.

8. ACKNOWLEDGEMENT:



G. Manoj Kumar working as an Assistant Professor in Masters of Computer Applications(MCA) in SVPEC, Visakhapatnam, Andhra Pradesh. Completed his Post graduation in Andhra University College of Engineering (AUCE). With accredited by NAAC with her areas of interest in java, Database management system.



Dandika Shankar is pursuing his final semester MCA in Sanketika Vidya Parishad Engineering College, accredited with A grade by NAAC, affiliated by Andhra University and approved by AICTE. With interest in Machine Learning, Dandika Shankar has taken up her PG project on “PREDICTION OF AIR QUALITY USING SUPERVISED MACHINE LEARNING APPROACH” and published the paper in connection to the project under the guidance of G. Manoj Kumar, Assistant Professor, Master of Computer Applications, SVPEC.

REFERENCES

- [1] Machine learning-based prediction of air quality index and air quality grade: Comparative analysis using RF, GB, LASSO, KNN, SVM, and stacking models
https://link.springer.com/article/10.1007/s13762-023-05016-2?utm_source=chatgpt.com
- [2] AirNet: predictive machine learning model for air quality forecasting: Uses RF, logistic regression, DT, SVM, KNN with Django web interface
https://environmentalsystemsresearch.springeropen.com/articles/10.1186/s40068-024-00378-z?utm_source=chatgpt.com
- [3] Forecasting air quality index: A statistical machine learning and deep learning approach: Compares SARIMA and LSTM for AQI prediction

https://emerginginvestigators.org/articles/24-079?utm_source=chatgpt.com

[4] Systematic Review of Machine Learning and Deep Learning Techniques for Spatiotemporal Air Quality Prediction: Overview of ML/DL methods in AQ forecasting

https://www.mdpi.com/2073-4433/15/11/1352?utm_source=chatgpt.com

[5] Deep-AIR: A Hybrid CNN-LSTM Framework for Fine-Grained Air Pollution Forecast: Combines CNN and LSTM for spatial-temporal prediction

https://arxiv.org/abs/2001.11957?utm_source=chatgpt.com

[6] Deep Air Quality Forecasting Using Hybrid Deep Learning Framework: Hybrid 1D-CNN + BiLSTM model for PM2.5 forecasting

https://arxiv.org/abs/1812.04783?utm_source=chatgpt.com

[7] Improving air quality prediction using hybrid BPSO with BWAQ: Feature selection and ML for AQI forecasting using LightGBM and RF

https://www.nature.com/articles/s41598-025-95983-y?utm_source=chatgpt.com

[8] Air quality forecasting using a modified statistical approach: Combines LASSO, ridge, elastic net, RF, KNN, XGBoost in a hybrid model

https://www.researchgate.net/publication/393017981_Air_quality_forecasting_using_a_modified_statistical_approach_Combining_statistical_and_machine_learning_methods

[9] Air Quality Forecasting Using Machine Learning: Evaluation of XGBoost, LightGBM, RF, SVR, CatBoost, KNN, stacking with Bayesian optimization

https://link.springer.com/article/10.1007/s11270-025-08122-8?utm_source=chatgpt.com

[10] A Novel CMAQ-CNN Hybrid Model to Forecast Hourly Surface-Ozone Concentrations: CNN-based forecasting for ozone concentrations

https://arxiv.org/abs/2008.05987?utm_source=chatgpt.com

[11] Machine learning algorithms to forecast air quality: a survey: Review of RF, DT, KNN regression and deep learning models

https://pmc.ncbi.nlm.nih.gov/articles/PMC9933038/?utm_source=chatgpt.com

[12] Air quality prediction and control systems using machine learning: Discusses RF for regression and classification in AQ systems

https://www.sciencedirect.com/science/article/pii/S2405844024158148?utm_source=chatgpt.com

[13] Air quality prediction by machine learning models: Data-driven AQI prediction using meteorological and pollutant features

https://www.sciencedirect.com/science/article/pii/S004565352301785X?utm_source=chatgpt.com

[14] A review of machine learning for modeling air quality: Highlights data preparation, validation, and ML/DL approaches

https://www.sciencedirect.com/science/article/abs/pii/S0169809524000437?utm_source=chatgpt.com

[15] Ease Air Quality Prediction Using Machine Learning: Project using RF, KNN, SVM, regressors to forecast AQI

https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4878522