Journal of Computer and Electrical Sciences (JCES)- Volume-1 Issue-4, April 2026
ISSN: 3049-2602

AIR QUALITY PREDICTION USING MACHINE LEARNING

L.Madhuri[1], Mrs. N. V. L. Manaswini[2]

[1]Student, Department of Computer science and Engineering

Andhra Loyola Institute of Engineering and Technology, Andhra Pradesh, India
[2]Associate Professor, Department of Computer Science and Engineering

Andhra Loyola Institute of Engineering and Technology, Andhra Pradesh, India

Abstract: This work presents the design and implementation of an Air Quality Prediction System using machine learning techniques. The system predicts Air Quality Index (AQI) values using regression models and classifies air quality status into categories such as Good, Moderate, Unhealthy for Sensitive Groups, and Unhealthy using classification models. The dataset incorporates multiple pollutant concentrations including SO_2, NO_2, CO, O_3, PM10, and PM2.5 along with meteorological parameters. Multiple ensemble algorithms including Decision Tree, AdaBoost, Gradient Boosting, and XGBoost are implemented and evaluated. Data preprocessing including missing value handling, label encoding, and feature scaling is performed. The trained models are integrated into a Flask-based web application for real-time AQI prediction.

Keywords: Air Quality Index, Machine Learning, XGBoost, Gradient Boosting, AdaBoost, Flask, AQI Prediction

1. INTRODUCTION

Air quality monitoring and prediction have become critical components in environmental management and public health systems. Rapid industrialization, urbanization, and increased vehicular emissions have significantly contributed to air pollution, making it necessary to develop intelligent systems that can analyze and predict air quality effectively. The Air Quality Index (AQI) is a standardized indicator used to measure the level of air pollution and its potential impact on human health. It is derived from multiple pollutant concentrations such as sulfur dioxide (SO_2), nitrogen dioxide (NO_2), carbon monoxide (CO), ozone (O_3), particulate matter (PM10 and PM2.5), and others.

The proposed system focuses on predicting AQI values and categorizing air quality status using machine learning techniques. Two major problem types are addressed in this project: regression and classification. Regression models are used to predict continuous AQI values, whereas classification models categorize the AQI into predefined levels such as Good, Moderate, Unhealthy, etc.

Data preprocessing plays a vital role in this system. Real-world air quality datasets often contain missing values, inconsistent formats, and noise. Techniques such as handling missing values, converting categorical variables into numerical formats, and feature scaling using StandardScaler are applied to ensure data quality and model performance. Label encoding is used to convert categorical AQI status into numerical labels suitable for machine learning algorithms.

The system employs multiple machine learning algorithms to improve prediction accuracy and robustness. Decision Tree models are used as baseline models due to their interpretability and simplicity. Ensemble techniques such as AdaBoost, Gradient Boosting, and XGBoost are implemented to enhance performance by combining multiple weak learners into a strong predictive model.

Model evaluation is carried out using appropriate metrics. For regression, metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R^2 Score are used. For classification, performance is evaluated using accuracy, precision, recall, F1-score, and confusion matrix. Finally, the system integrates machine learning models into a Flask web application, enabling users to input environmental parameters and obtain AQI predictions instantly.

2. LITERATURE SURVEY

Air quality prediction has been extensively studied in recent years due to increasing environmental concerns and the need for intelligent monitoring systems. Various researchers have proposed different machine learning and statistical approaches to predict AQI and analyze pollution patterns.

https://doi.org/10.5281/zenodo.19612360

Traditional methods for air quality prediction relied heavily on statistical techniques such as linear regression, time series analysis, and autoregressive models. While these methods provided baseline results, they were limited in handling complex nonlinear relationships between pollutants and environmental factors.

With the advancement of machine learning, researchers began applying algorithms such as Decision Trees, Support Vector Machines (SVM), Random Forests, and Neural Networks to improve prediction accuracy. Ensemble learning techniques such as AdaBoost and Gradient Boosting improved prediction performance significantly by combining multiple weak learners. XGBoost, an advanced implementation of Gradient Boosting, has been widely adopted due to its efficiency, scalability, and superior performance, including regularization techniques that reduce overfitting.

Deep learning approaches, including Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN), have also been explored for air quality prediction. These models capture complex patterns in large datasets but require significant computational resources. Feature engineering and selection have been identified as crucial factors influencing model performance. Recent research also focuses on integrating machine learning models with web-based applications using frameworks like Flask or Streamlit for real-time prediction.

3. PROPOSED SYSTEM

3.1 Existing System

Traditional air quality monitoring systems primarily rely on manual data collection and basic statistical analysis. Government agencies deploy monitoring stations that measure pollutant concentrations but lack predictive capabilities. Statistical models such as linear regression and time series forecasting were used with significant limitations including assumed linear relationships, failure to capture complex interactions, and limited scalability. Most existing systems also lacked automation in data preprocessing, classification of air quality into meaningful categories, and user-friendly interfaces for general users.

3.2 Proposed System

The proposed Air Quality Prediction System addresses the limitations of existing systems by integrating advanced machine learning techniques, automated data processing, and a Flask-based web interface. The system predicts both AQI values (regression) and air quality categories (classification) using multiple ensemble algorithms. Key features include:

- Automated data preprocessing pipeline handling missing values, encoding, and scaling
- Dual-model approach with separate regression and classification models
- Data balancing techniques for equal class representation
- Comprehensive model evaluation using MAE, MSE, RMSE, R^2, precision, recall, F1-score
- Model persistence using pickle and joblib for reuse without retraining
- Flask-based web application for real-time predictions

4. METHODOLOGY

The methodology of the Air Quality Prediction System follows a structured pipeline ensuring accurate and efficient prediction of AQI values and categories.

The first step is data collection, where the air quality dataset containing pollutant attributes and environmental factors is obtained. The dataset includes features such as SO_2, CO, O_3, O_3_8hr, PM10, PM2.5, NO_2, NOx, NO, wind speed, wind direction, CO_8hr, PM2.5_avg, PM10_avg, and SO_2_avg.

Data preprocessing is the next crucial step. Missing values are identified and removed to maintain data integrity. All columns are converted into numerical format. Categorical variables such as air quality status are encoded using LabelEncoder. Feature selection is performed to retain only relevant attributes.

The dataset is divided into training and testing sets using train-test split. Feature scaling is applied using StandardScaler. Multiple algorithms are then implemented and trained: Decision Tree, AdaBoost, Gradient Boosting, and XGBoost for both regression and classification tasks.

After training, models are evaluated using appropriate metrics. Regression models are assessed using MAE, MSE, RMSE, and R^2 score, while classification models are assessed using precision, recall, F1-score, and confusion matrix. The best-performing models are then saved using pickle and joblib and integrated into a Flask-based web application for real-time predictions.

https://doi.org/10.5281/zenodo.19612360

5. SYSTEM REQUIREMENTS

5.1 Software Requirements

Operating System: Windows / Linux / macOS; Programming Language: Python; Libraries Used: NumPy, Pandas, Matplotlib, Seaborn, Scikit-learn, XGBoost, Joblib/Pickle; Framework: Flask for web integration; IDE/Tools: VS Code / Jupyter Notebook / PyCharm; Database (Optional): MySQL; Browser: Chrome / Edge.

5.2 Hardware Requirements

Processor: Minimum Intel i3 or higher; RAM: Minimum 4 GB (8 GB recommended); Storage: At least 10 GB free space; GPU (Optional): For faster training; Internet: Required for deployment and updates.

6. IMPLEMENTATION

6.1 System Architecture

The Air Quality Prediction System follows a modular and layered architecture divided into four main layers:

The Data Layer handles input data including the air quality dataset stored in CSV format. The Processing Layer performs all preprocessing tasks: handling missing values, converting data types, encoding categorical variables using LabelEncoder, and scaling features using StandardScaler. Data balancing is also applied in this layer.

The Model Layer is the core component consisting of multiple machine learning models for both regression and classification. Algorithms such as Decision Tree, AdaBoost, Gradient Boosting, and XGBoost are implemented, trained using preprocessed data, and saved using pickle or joblib. The Presentation Layer provides a Flask-based web interface where users input environmental parameters and receive predictions in a user-friendly format.

6.2 Model Development

The model creation process involves training multiple machine learning algorithms using preprocessed air quality data. Separate datasets are prepared for regression and classification tasks.

Decision Tree models are initially trained as baseline models. Ensemble methods including AdaBoost, Gradient Boosting, and XGBoost are then implemented. Each model is trained using training data and evaluated on testing data with hyperparameters tuned for optimal performance.

6.2.1 Key Code Snippets

```
from xgboost import XGBRegressor
xgb_reg = XGBRegressor(n_estimators=100, learning_rate=0.1, max_depth=5,
random_state=42)
xgb_reg.fit(X_train_reg_scaled, y_train_reg)
y_pred_xgb = xgb_reg.predict(X_test_reg_scaled)

from xgboost import XGBClassifier
xgb_clf = XGBClassifier(n_estimators=100, learning_rate=0.1, max_depth=5,
    objective='multi:softprob', num_class=5, random_state=42)
xgb_clf.fit(X_train_cla_scaled, y_train_cla)
```

6.3 Flask Web Application

Flask is used to integrate machine learning models with a user-friendly interface, enabling real-time AQI prediction. Flask follows a simple architecture based on routes and templates. Routes are defined for the home page, prediction page, and result display page. Pre-trained models, scalers, and encoders are loaded at application startup using pickle or joblib. Template rendering using Jinja2 allows dynamic prediction results to be displayed on web pages.

6.4 Results

The proposed system was successfully implemented and tested. XGBoost achieved the best performance in both regression and classification tasks:

- Decision Tree Regression: $R^2 = 0.91$, RMSE = 8.2
- AdaBoost Regression: $R^2 = 0.93$, RMSE = 6.8
- Gradient Boosting Regression: $R^2 = 0.96$, RMSE = 5.1
- XGBoost Regression: $R^2 = 0.98$, RMSE = 3.7

https://doi.org/10.5281/zenodo.19612360

- XGBoost Classification Accuracy: 100% on balanced dataset

The Classification Report for XGBoost showed precision, recall, and F1-score of 1.00 for all classes (Good, Moderate, Unhealthy for Sensitive Groups, Unhealthy). The confusion matrices confirmed minimal misclassification. Sample prediction output: Predicted AQI Value: 138.58, Predicted Status: Unhealthy for Sensitive Groups.

7. TEST CASES

7.1 Testing Methodology

The system was tested using four levels: Unit Testing (individual modules), Integration Testing (combined module interaction), System Testing (complete application workflow), and User Acceptance Testing (end-user evaluation). Edge cases including invalid inputs, zero values, and extreme pollutant concentrations were also verified.

7.2 Test Case Table

TC ID	Test Scenario	Input	Expected Output	Actual Output	Status
TC01	Load Dataset	CSV file	Dataset loaded	Same	Pass
TC02	Handle Missing Values	Null data	Removed/cleaned	Same	Pass
TC03	Encode Status	Categorical data	Numeric labels	Same	Pass
TC04	Feature Scaling	Raw data	Scaled data	Same	Pass
TC05	Train Regression Model	Training data	Model trained	Same	Pass
TC06	Train Classification Model	Training data	Model trained	Same	Pass
TC07	Predict AQI Value	Valid input	AQI value	Same	Pass
TC08	Predict AQI Status	Valid input	Category label	Same	Pass
TC09	Invalid Input Handling	Text input	Error message	Same	Pass
TC10	Confusion Matrix Display	Model output	Graph displayed	Same	Pass
TC11	Save Model	Trained model	File saved	Same	Pass
TC12	Load Model	Saved file	Model loaded	Same	Pass
TC13	Flask Route Access	URL access	Page loads	Same	Pass
TC14	Form Submission	User input	Prediction shown	Same	Pass
TC15	Scaling Consistency	New data	Proper scaling	Same	Pass
TC16	Label Decoding	Encoded label	Original label	Same	Pass
TC17	XGBoost Regression	Training data	AQI predicted	Same	Pass
TC18	XGBoost Classification	Training data	Status predicted	Same	Pass
TC19	AdaBoost Regression	Training data	AQI predicted	Same	Pass
TC20	AdaBoost Classification	Training data	Status predicted	Same	Pass

https://doi.org/10.5281/zenodo.19612360

8CONCLUSION

The Air Quality Prediction System developed in this project demonstrates the effective use of machine learning techniques in environmental monitoring and analysis. The system successfully implements both regression and classification models to predict AQI values and categorize air quality status. Multiple ensemble algorithms including Decision Tree, AdaBoost, Gradient Boosting, and XGBoost were implemented and evaluated.

Among the implemented algorithms, XGBoost demonstrated superior performance for both regression ($R^2 = 0.98$) and classification (Accuracy = 100% on balanced data), confirming the effectiveness of ensemble methods for AQI prediction. The Flask-based web application makes the system accessible to both technical and non-technical users, enabling real-time AQI predictions from environmental input parameters.

Future enhancements include integration with IoT sensors for real-time data, application of deep learning models such as LSTM for temporal patterns, geographical visualization using maps, mobile application development, and deployment on cloud platforms such as AWS or Google Cloud.

REFERENCES

1. Scikit-learn Documentation – https://scikit-learn.org

2. XGBoost Documentation – https://xgboost.readthedocs.io

3. Python Official Documentation – https://docs.python.org

4. Flask Documentation – https://flask.palletsprojects.com

5. Research Papers on Air Quality Prediction (IEEE, Springer)

6. Environmental Protection Agency (EPA) AQI Guidelines

7. Kaggle Datasets – Air Quality Data

8. NumPy and Pandas Documentation – https://numpy.org, https://pandas.pydata.org

9. Chen, T. and Guestrin, C., "XGBoost: A Scalable Tree Boosting System," KDD 2016.

10. Friedman, J.H., "Greedy Function Approximation: A Gradient Boosting Machine," Annals of Statistics, 2001.

https://doi.org/10.5281/zenodo.19612360