IILM UNIVERSITY

Air Quality Prediction Using Machine Learning

SUBMITTED BY:

Girish Kumar Yadav
Garvit Jain
Veer Singhi

SUBMITTED TO:

Mrs Vijaya Choudhary
(Project Guidance Mentor)

Teacher's Signature:

Air Quality Prediction Using Machine Learning

Software Requirements Specification

Table of Contents

1. Introduction

This Software Requirements Specification (SRS) document provides a comprehensive overview of the requirements for the Air Quality Prediction System, a machine learning-based software designed to predict the Air Quality Index (AQI) and classify air quality categories based on sensor data. The system leverages advanced algorithms to process historical air quality data, offering valuable insights for environmental monitoring, public health, and urban planning.

1.1 Purpose

The purpose of this SRS is to clearly define the functional and non-functional requirements for the Air Quality Prediction System. This document serves as a blueprint for software engineers, data scientists, environmental researchers, and project stakeholders to ensure the system is developed, implemented, and tested in alignment with specified goals. It aims to facilitate clear communication among all parties involved and establish a foundation for successful project execution.

1.2 Scope

The Air Quality Prediction System is a standalone software product with the following objectives:

- Data Processing: Ingest and preprocess air quality data from the AirQualityUCI.csv dataset, which contains sensor measurements such as CO, NO2, and ozone levels.
- Machine Learning: Utilize Random Forest, AdaBoost, and Logistic Regression models to predict AQI values and classify air quality into categories (e.g., Good, Moderate, Unhealthy).
- Applications: Support environmental monitoring, public health initiatives, and policy-making by providing accurate AQI predictions within 1 second per query, achieving an R^2 score of at least 0.95 for regression models and an accuracy of at least 85% for classification models.

Exclusions:

- The system does not collect real-time data from physical sensors.
- It does not include a graphical user interface, though it saves models for potential front-end integration.
- Hardware integration with IoT devices is out of scope for the current version.

The system is designed to be extensible, allowing future enhancements such as real-time data processing or web-based interfaces.

1.3 Definitions, Acronyms, and Abbreviations

- AQI: Air Quality Index, a standardized measure of air pollution levels.
- CO(GT): Carbon Monoxide concentration (Ground Truth).
- NO2(GT): Nitrogen Dioxide concentration.
- PT08.S1(CO): Sensor response to CO.

- NMHC(GT): Non-Methane Hydrocarbons concentration.
- C6H6(GT): Benzene concentration.
- PT08.S2(NMHC): Sensor response to NMHC.
- NOx(GT): Nitrogen Oxides concentration.
- PT08.S3(NOx): Sensor response to NOx.
- PT08.S4(NO2): Sensor response to NO2.
- PT08.S5(O3): Sensor response to Ozone.
- T: Temperature in Celsius.
- RH: Relative Humidity in percentage.
- AH: Absolute Humidity.
- ML: Machine Learning.
- RF: Random Forest, an ensemble learning method.
- AB: AdaBoost, a boosting algorithm.
- LR: Logistic Regression, a classification algorithm.
- NaN: Not a Number, indicating missing data.
- SRS: Software Requirements Specification.

1.4 References

1. UCI Machine Learning Repository: Air Quality Dataset. Available at: https://archive.ics.uci.edu/dataset/360/air+quality.
2. IEEE Guide to Software Requirements Specifications, IEEE Std 830-1998.
3. Scikit-learn Documentation: https://scikit-learn.org/stable/.
4. Pandas Documentation: https://pandas.pydata.org/.
5. Joblib Documentation: https://joblib.readthedocs.io/.
6. GeeksforGeeks AQI Using Python: https://www.geeksforgeeks.org/predicting-air-quality-index-using-python/

These references provide foundational data, standards, and technical documentation critical to the system's development.

1.5 Overview

This SRS is structured to provide a clear and systematic description of the Air Quality Prediction System:
- Section 2: Offers a general description, covering the product's perspective, core functions, user profiles, constraints, and assumptions.
- Section 3: Details specific requirements, including external interfaces, functional requirements, non-functional requirements, and design constraints.

This organization ensures that all aspects of the system are thoroughly documented and easily accessible to stakeholders.

2. General Description

This section provides a high-level overview of the Air Quality Prediction System, outlining its context, functionality, and operational considerations.

2.1 Product Perspective

The Air Quality Prediction System is a standalone software solution developed in Python, leveraging machine learning to analyze air quality sensor data. It integrates seamlessly with open-source libraries such as Pandas for data manipulation and Scikit-learn for model training. The system processes historical data from the AirQualityUCI.csv dataset, producing AQI predictions and classifications. While currently a backend solution, it is designed with extensibility in mind, enabling potential integration with front-end interfaces (e.g., web or mobile apps) or IoT devices for real-time monitoring in future iterations. The system operates independently of external hardware, relying solely on pre-collected CSV data.

2.2 Product Functions

The Air Quality Prediction System performs the following core functions:

- Data Ingestion: Loads the AirQualityUCI.csv file, which contains 9357 rows of sensor data including pollutants (CO, NO2, ozone) and environmental factors (temperature, humidity).
- Data Preprocessing: Cleans the dataset by removing invalid columns, handling missing values (replacing -200 with NaN and imputing with column means), and ensuring data consistency.
- Model Training: Trains Random Forest and AdaBoost models for AQI regression and Logistic Regression for AQI classification into categories such as Good, Moderate, or Unhealthy.
- Model Persistence: Saves the trained Random Forest model as aqi_model.pkl for reuse in predictions or integration with external applications.
- Performance Evaluation: Outputs detailed metrics, including R^2 score for regression models, mean absolute error, and classification accuracy, to assess model effectiveness.

These functions enable the system to deliver reliable air quality insights for environmental and public health applications.

2.3 User Characteristics

The system caters to the following user groups:

- Data Scientists: Professionals with expertise in Python and machine learning, responsible for developing, testing, and refining the models. They require familiarity with libraries like Scikit-learn and Pandas.
- Environmental Researchers: Individuals studying air quality trends for academic or public health purposes. They need basic Python knowledge to run the system and interpret outputs.

- Software Developers: Engineers integrating the system's backend into larger applications, such as web platforms or mobile apps. They should understand model serialization (e.g., .pkl files) and API integration.

No advanced air quality domain knowledge is required, as the system abstracts complex data processing into user-friendly outputs.

2.4 General Constraints

The system operates under the following constraints:
- Data Source: Limited to processing the AirQualityUCI.csv dataset, which uses semicolon separators and commas for decimals.
- Hardware: Requires standard hardware (e.g., 4GB RAM, 1GHz CPU) with Python 3.11 installed.
- No Real-Time Capability: Does not support direct sensor integration or real-time data collection.
- Dataset Format: Assumes consistent CSV structure (e.g., specific columns like CO(GT), NO2(GT)). Any deviation may cause preprocessing errors.

These constraints ensure the system remains lightweight and focused on its core objectives.

2.5 Assumptions and Dependencies

Assumptions:
- The AirQualityUCI.csv dataset is available, contains valid sensor readings, and follows the expected format.
- Missing values in the dataset are consistently marked as -200, allowing standardized preprocessing.
- The operating environment supports Python 3.11 and required libraries without compatibility issues.

Dependencies:
- Python Libraries: Pandas (≥1.5) for data processing, Scikit-learn (≥1.0) for machine learning, NumPy for numerical operations, and Joblib (≥1.0) for model serialization.
- Dataset: The system requires the AirQualityUCI.csv file to function, as it is the sole data source.

These assumptions and dependencies are critical to the system's successful operation and must be validated during deployment.

3. Specific Requirements

This section details the specific requirements for the Air Quality Prediction System, ensuring all are traceable, verifiable, and unambiguous.

3.1 External Interface Requirements

3.1.1 User Interfaces

- UI-001: The system shall provide a command-line interface (CLI) for executing the Python script, allowing users to initiate data processing and model training.
- UI-002: The system shall display performance metrics (e.g., R^2 score, classification accuracy) in the console for user review.
- UI-003: The system shall save the trained Random Forest model as aqi_model.pkl, enabling future use in predictions or integration with external applications.

The CLI is designed to be intuitive, requiring minimal user input beyond specifying the CSV file path.

3.1.2 Hardware Interfaces

- HI-001: The system shall operate on standard hardware configurations, such as a CPU with 4GB RAM and 1GHz processing speed, ensuring accessibility on common devices.
- HI-002: No direct hardware interfaces (e.g., air quality sensors) are required, as the system processes pre-collected data from the AirQualityUCI.csv file.

This approach minimizes hardware dependencies, making the system portable across various platforms.

3.1.3 Software Interfaces

- SI-001: The system shall interface with Python 3.11, ensuring compatibility with modern Python environments.
- SI-002: The system shall use Pandas (version ≥1.5) for efficient data loading, cleaning, and preprocessing.
- SI-003: The system shall use Scikit-learn (version ≥1.0) for implementing and training machine learning models (Random Forest, AdaBoost, Logistic Regression).
- SI-004: The system shall use Joblib (version ≥1.0) for serializing and saving trained models to disk.

These software interfaces ensure robust data handling and model performance.

3.1.4 Communications Interfaces

- CI-001: The system shall not require network communication, as it processes local CSV files and operates offline.
- CI-002: If integrated with a front-end application in the future, the system shall support file-based model loading (e.g., reading aqi_model.pkl) without requiring network protocols.

This offline design enhances security and simplifies deployment.

3.2 Functional Requirements

3.2.1 Data Collection and Preprocessing

3.2.1.1 Introduction

This feature enables the system to load and preprocess air quality data from the AirQualityUCI.csv file, ensuring it is clean and ready for machine learning. Preprocessing is critical to handle inconsistencies, such as missing values or invalid entries, which could degrade model performance.

3.2.1.2 Inputs

- DC-001: The system shall accept a CSV file (AirQualityUCI.csv) with semicolon-separated values and commas as decimal points.
- DC-002: The CSV file shall include columns: Date, Time, CO(GT), PT08.S1(CO), NMHC(GT), C6H6(GT), PT08.S2(NMHC), NOx(GT), PT08.S3(NOx), NO2(GT), PT08.S4(NO2), PT08.S5(O3), T, RH, AH.

3.2.1.3 Processing

- DC-003: The system shall load the CSV file using Pandas, specifying the correct separator (semicolon) and decimal (comma) settings.
- DC-004: The system shall remove the last two unnamed columns (Unnamed: 15, Unnamed: 16), which contain NaN values and are irrelevant to analysis.
- DC-005: The system shall truncate the dataset to 9357 rows, discarding any trailing rows with all NaN values.
- DC-006: The system shall replace -200 values (indicating missing data) with NaN to standardize handling.
- DC-007: The system shall impute NaN values with the mean of the respective column, preserving data integrity.

3.2.1.4 Outputs

- DC-008: The system shall produce a cleaned Pandas DataFrame with 9357 rows and 15 columns, free of missing values.

3.2.1.5 Error Handling

- DC-009: The system shall display an error message if the CSV file is not found or incorrectly formatted (e.g., wrong separator).
- DC-010: The system shall log preprocessing errors (e.g., invalid data types) to the console for debugging.

This preprocessing pipeline ensures the dataset is robust and suitable for accurate model training.

3.2.2 AQI Prediction Using Regression

3.2.2.1 Introduction
This feature enables the system to predict AQI values using Random Forest and AdaBoost regression models. AQI prediction is a core functionality, providing numerical estimates of air pollution levels for environmental monitoring.

3.2.2.2 Inputs
- AP-001: The system shall use features: CO(GT), NO2(GT), PT08.S5(O3), T, RH, AH, which are critical indicators of air quality.
- AP-002: The system shall compute AQI as the average of CO(GT), NO2(GT), and PT08.S5(O3) for training and evaluation purposes.

3.2.2.3 Processing
- AP-003: The system shall split the dataset into 80% training and 20% testing sets, using a random_state=42 for reproducibility.
- AP-004: The system shall train a Random Forest Regressor with default Scikit-learn parameters, leveraging its robustness to noisy data.
- AP-005: The system shall train an AdaBoost Regressor with default parameters, enhancing prediction accuracy through boosting.
- AP-006: The system shall evaluate models using R^2 score and mean absolute error, providing comprehensive performance insights.

3.2.2.4 Outputs
- AP-007: The system shall output an R^2 score for Random Forest, expected to be ≥ 0.99, indicating high prediction accuracy.
- AP-008: The system shall output an R^2 score for AdaBoost, expected to be ≥ 0.93, reflecting reliable performance.
- AP-009: The system shall output an R^2 score for Logistic Regression, expected to be ≥ 0.87, reflecting reliable performance.

3.2.2.5 Error Handling
- AP-010: The system shall display an error if model training fails (e.g., due to insufficient data or feature mismatches).
- AP-011: The system shall log prediction errors to the console, aiding in troubleshooting.

This feature ensures precise AQI predictions, supporting applications in public health and environmental policy.

3.2.3 AQI Classification Using Logistic Regression
3.2.3.1 Introduction
This feature enables the system to classify AQI into categories (e.g., Good, Moderate, Unhealthy) using Logistic Regression, providing interpretable air quality assessments for end-users.

3.2.3.2 Inputs
- AC-001: The system shall use the same features as AQI prediction: CO(GT), NO2(GT), PT08.S5(O3), T, RH, AH.
- AC-002: The system shall categorize AQI into: Good (≤50), Moderate (≤100), Unhealthy for Sensitive (≤150), Unhealthy (≤200), Very Unhealthy (≤300), Hazardous (>300).

3.2.3.3 Processing
- AC-003: The system shall encode AQI categories using Scikit-learn's LabelEncoder for compatibility with Logistic Regression.
- AC-004: The system shall split the dataset into 80% training and 20% testing sets (random_state=42).
- AC-005: The system shall train a Logistic Regression model with max_iter=1000 to ensure convergence.
- AC-006: The system shall evaluate the model using accuracy and a classification report (precision, recall, F1-score) for each category.

3.2.3.4 Outputs
- AC-007: The system shall output an accuracy score, expected to be ≥0.85, indicating reliable classification.
- AC-008: The system shall output a classification report detailing performance across all AQI categories.

3.2.3.5 Error Handling
- AC-009: The system shall display a warning if Logistic Regression fails to converge within 1000 iterations.
- AC-010: The system shall log classification errors to the console for debugging.

This feature enhances the system's utility by providing user-friendly air quality classifications.

3.3 Non-Functional Requirements

3.3.1 Performance
- NF-PERF-001: The system shall process a single AQI prediction within 1 second on standard hardware, ensuring real-time usability.
- NF-PERF-002: The system shall complete training of all models (Random Forest, AdaBoost, Logistic Regression) within 5 minutes for a dataset of 9357 rows, balancing efficiency and accuracy.

3.3.2 Reliability

- NF-REL-001: The system shall achieve an R^2 score of ≥0.99 for Random Forest regression, ensuring high prediction accuracy.
- NF-REL-002: The system shall achieve a classification accuracy of ≥0.87 for Logistic Regression, guaranteeing reliable AQI categorization.

3.3.3 Availability
- NF-AVAIL-001: The system shall be available 99% of the time when executed in a stable Python environment, minimizing downtime.
- NF-AVAIL-002: The system shall handle unexpected crashes gracefully, displaying clear error messages to aid recovery.

3.3.4 Security
- NF-SEC-001: The system shall not expose sensitive data, as it processes anonymized sensor data from AirQualityUCI.csv.
- NF-SEC-002: The system shall ensure saved model files (e.g., aqi_model.pkl) are not corrupted during serialization, maintaining data integrity.

3.3.5 Maintainability
- NF-MAIN-001: The system code shall adhere to PEP 8 style guidelines, enhancing readability and ease of updates.
- NF-MAIN-002: The system shall include inline comments and documentation for all major functions, facilitating future maintenance.

3.3.6 Portability
- NF-PORT-001: The system shall run on any operating system supporting Python 3.11, including Windows, Linux, and macOS.
- NF-PORT-002: The system shall be portable across Python environments with the required libraries installed, ensuring flexibility.

3.4 Design Constraints
- DC-001: The system shall use only open-source Python libraries (Pandas, Scikit-learn, NumPy, Joblib) to ensure accessibility and cost-effectiveness.
- DC-002: The system shall process data strictly in the AirQualityUCI.csv format (semicolon-separated, comma as decimal) to maintain compatibility.
- DC-003: The system shall not require proprietary software or hardware, ensuring broad deployability.
- DC-004: The system shall adhere to Scikit-learn’s API for model training and evaluation, leveraging standardized machine learning workflows.