5. Proposed Work

This project presents an example of a web-based application which leverage a heuristic optimization algorithm called Jaya to predict pollution levels. The system adjusts linear regression models to estimate PM2.5, PM10, and AQI delivering a highly accurate estimate. The users can conveniently upload real-time or historical environmental data in CSV format containing 20 meteorological and urban data like temperature, relative humidity, wind speed, and a congestion index. We use a characteristic equation for each pollutant, which has assigned fixed weights. We optimize these weights using the Jaya algorithm iteratively to minimize prediction errors. After the optimization step, the model can make predictions for the concentration levels of the pollutants, and the users can download the processed data. The implementation is fully browser-based, hence, does not require a server backend which makes this system accessible, and platform-independent. The implementation is done using TensorFlow.js, which allows for real-time computations done in the browser. This novel solution provides real-time, accurate, customizable, and user-friendly pollution forecasting, which can be further developed for urban health surveillance and environmental policy monitoring.

5.1 Framework

The proposed system is a client-side web application designed to predict air pollution parameters—specifically PM2.5, PM10, and AQI—using characteristic linear equations enhanced through the Jaya optimization algorithm. Users provide input via CSV files containing historical or real-time data for 20 environmental and meteorological parameters such as temperature, humidity, wind speed, and congestion index, along with measured pollutant concentrations. For each pollutant, a predefined linear model with assigned feature weights and a bias term is optimized using the Jaya algorithm, which iteratively adjusts these parameters to minimize the mean squared prediction error. The system’s architecture is designed to perform all computations within the browser environment, ensuring rapid feedback, platform independence, and user privacy.

The architecture diagram illustrates the end-to-end workflow of the system, from data input to prediction output. The process begins with the user uploading a CSV file, which is parsed directly in the browser. The parsed data feeds into the initial characteristic equations, which act as baseline linear models for PM2.5, PM10, and AQI. These equations serve as the foundation for Jaya optimization, which refines the weights and biases to fit the specific dataset. Once optimization is complete, the updated weights are passed to a prediction module implemented in TensorFlow.js. This module computes predicted pollutant values for each row in the dataset. Finally, a CSV file containing both actual and predicted values is generated and made available for download. All stages—data parsing, optimization, prediction, and file generation—are executed in-browser without server-side interaction, ensuring efficiency and ease of use.

The system is built entirely using HTML, CSS, and JavaScript, with TensorFlow.js serving as the computational backend for prediction logic. The Jaya optimization algorithm is implemented natively in JavaScript to allow seamless integration with the browser interface. As shown in the architecture diagram, the “Frontend Browser” block encapsulates all core functionalities: data preprocessing, application of characteristic equations, Jaya optimization, and prediction using TensorFlow.js. This design eliminates the need for external servers or APIs, enhancing performance and security. After optimization and prediction, results are presented in a downloadable CSV file, allowing users to store or further analyze the data. This lightweight and modular architecture provides a scalable foundation for real-time air quality prediction tools, particularly in resource-constrained or decentralized deployment scenarios.

5.2 Algorithm

Algorithm 1: Jaya Optimization for Weight Tuning in Pollution Prediction

Input:

- Dataset D with features X₁ to X₂₀ and target Y ∈ {PM2.5, PM10, AQI}

- Initial weights W₀ = [w₁, w₂, ..., w₂₀], bias b₀

- Population size N, number of iterations T

Output:

- Optimized weights W\*, bias b\*

1: Initialize population P of N candidate solutions, each as [w₁, ..., w₂₀, b]

2: for t = 1 to T do

3: For each candidate i ∈ P:

4: Evaluate fitness\_i = Mean Squared Error over dataset D

5: Identify best and worst candidates in P based on fitness

6: For each candidate i ∈ P:

7: For each dimension j ∈ [1,...,21]: // 20 weights + bias

8: r₁, r₂ ← Random numbers in [0,1]

9: xᵢⱼ ← xᵢⱼ + r₁ × (bestⱼ − |xᵢⱼ|) − r₂ × (worstⱼ − |xᵢⱼ|)

10: Update candidate i if new fitness improves

11: end for

12: Return best solution [W\*, b\*] from final population P

Description of the Algorithm

The Jaya optimization algorithm is employed in this work to fine-tune the weights and bias of linear characteristic equations used for predicting PM2.5, PM10, and AQI values. As a population-based, parameter-free metaheuristic, Jaya iteratively refines a set of candidate solutions without requiring algorithm-specific control parameters such as crossover or mutation rates. Each candidate represents a set of weights and bias, and its fitness is evaluated using mean squared error against the input dataset. In each iteration, candidates are updated by moving towards the best solution and away from the worst, guided by random factors to maintain diversity. This process continues until a specified number of iterations is reached, after which the best-performing solution is selected as the optimized model. The lightweight nature of Jaya makes it particularly suitable for in-browser execution, as implemented in this work using JavaScript and TensorFlow.js for real-time, client-side pollution prediction.