Optimizing LSTM Architectures for Air Quality Prediction: A Comparative Study of Metaheuristic Algorithms-Based Optimizers

Soham Pahari  
University of Petroleum and Energy Studies

**Abstract— Air pollution is one of the most critical challenges of the 21st century, exacerbated by urbanization and population growth, leading to significant global concern. To address this issue, numerous researchers have explored various methodologies for accurately predicting air pollutant concentrations. This paper presents a comparative analysis of the most effective prediction methods, aiming to identify optimal combinations for enhanced performance. Motivated by nature-inspired solutions, we investigate the potential of integrating metaheuristic algorithms with recurrent neural networks (RNNs), particularly Long Short-Term Memory (LSTM) models. While traditional ARIMA models have proven effective in time series analysis, recent advancements in RNN architectures suggest that LSTMs can outperform them, although there remains considerable room for improvement in parameter optimization. We evaluate the effectiveness of five algorithms—Dung Beetle Algorithm, Quantum Swarm Algorithm, Hybrid Genetic Algorithm, Ant Colony Algorithm, and Gravitational Algorithm—to develop a hybrid metaheuristic algorithm that leverages the strengths of these approaches while minimizing their weaknesses. A systematic study evaluates the effectiveness of these algorithms using datasets sourced from various open repositories to ensure a comprehensive analysis. Our findings aim to enhance pollution forecasting accuracy and contribute to more effective environmental management\_strategies.**

INTRODUCTION

India is home to some of the most polluted cities in the world. Rapid population growth, urbanization, and a combination of other major and minor factors are worsening the problem. The Air Quality Index (AQI) serves as a numerical indicator of pollution levels, primarily driven by four key pollutants: PM2.5, PM10, NOx, CO, and ozone. This harmful mix has been linked to numerous health issues, including respiratory conditions like bronchitis, emphysema, and asthma, as well as cardiovascular, neurological, and other complex diseases. Air pollution is a highly complex field of study due to the various linear and nonlinear factors involved. [1,2]

In an effort to understand the behaviour of air pollution, various predictive models have been proposed over time. These range from traditional statistical models to machine learning and deep learning approaches.

Traditional studies on time series always suggest using Auto Regressive (AR), Auto Regressive Integrated Moving Average (ARIMA) models, the Gray model, and in some cases, Multiple Linear Regression models [3,4]. These models are very effective at detecting linearity; however, when there are nonlinear variables, they fail to proceed further. This is the reason these models always neglect residuals and are mainly efficient in forecasting trend and seasonal components.

In the recent era of advanced machine learning, there has been considerable research on detecting nonlinear components, some of which have outperformed traditional models. One paper discussed the development of a machine learning-based empirical model to predict the laminar combustion rate of AQI under extreme conditions. Another paper focused on adding multiple features and worked on mitigating convergence issues. Gu [13]. proposed a new hybrid prediction model for PM2.5 prediction, which outperformed other models in the field. The application of hybrids has significantly contributed to future approaches.

Deep learning models are much more adaptive and efficient at recognizing patterns than machine learning models. One paper discussed a comprehensive study of different DL methods and their effectiveness in predicting air pollution and forecasting. Wu[14] researched a hybrid DL-based model to predict next-hour AQI.

Some scholars have worked on combinational models to optimize parameters. Different algorithms offer different perspectives to optimize various areas, which helps achieve more accurate results and capture complex and minor patterns in residuals. Here, we must mention Kshirsagar, who explored how neural networks, regression, and hybrid models play a role in the analysis, prediction, and mitigation of air pollution, considering the most recent developments and research in the field. Zhang [11] and Li took a more advanced approach, introducing Long Short-Term Memory (LSTM) from Recurrent Neural Networks (RNNs) and also applying Convolutional Neural Networks (CNNs). The addition of these two powerful deep learning methods transformed time series prediction.

More work on Zhang and Li’s findings suggests that something further should be explored in the field of optimization. Some researchers have begun using different algorithms and combinations to achieve the most optimized parameters, reaching the lowest possible point in the error space. This introduced nature-inspired algorithms—metaheuristic algorithms. Akilandeswari [7] initiated this work by adding the Grey Wolf Optimizer in LSTM hyperparameter tuning, which outperformed previous methods. Particle Swarm Optimizer (PSO), Dung Beetle Optimizer (DBO), and Gravitational Search Algorithm (GSA) have also shed light on this field. Each new model challenges the previous one in different aspects. Combining this with linear models like ARIMA and SARIMAX makes the models more effective and advanced.

The focus of this paper is: **(1)** We tried to predict each component with a suitable method, applying all the linear components in ARIMA and SARIMAX models, and separating all nonlinear components into deep learning methods; **(2)** we developed a combined metaheuristic model inspired by DBO-PSO-GA-GSA-RDO.

MATERIALS AND METHODS

DECOMPOSITION

We used LOESS (Locally Estimated Scatterplot Smoothing) to decomposed this data. As this data was showing a additive flow we decomposed them in additive decomposition.

This gives us Trend (), Seasonal () and Residuals ()

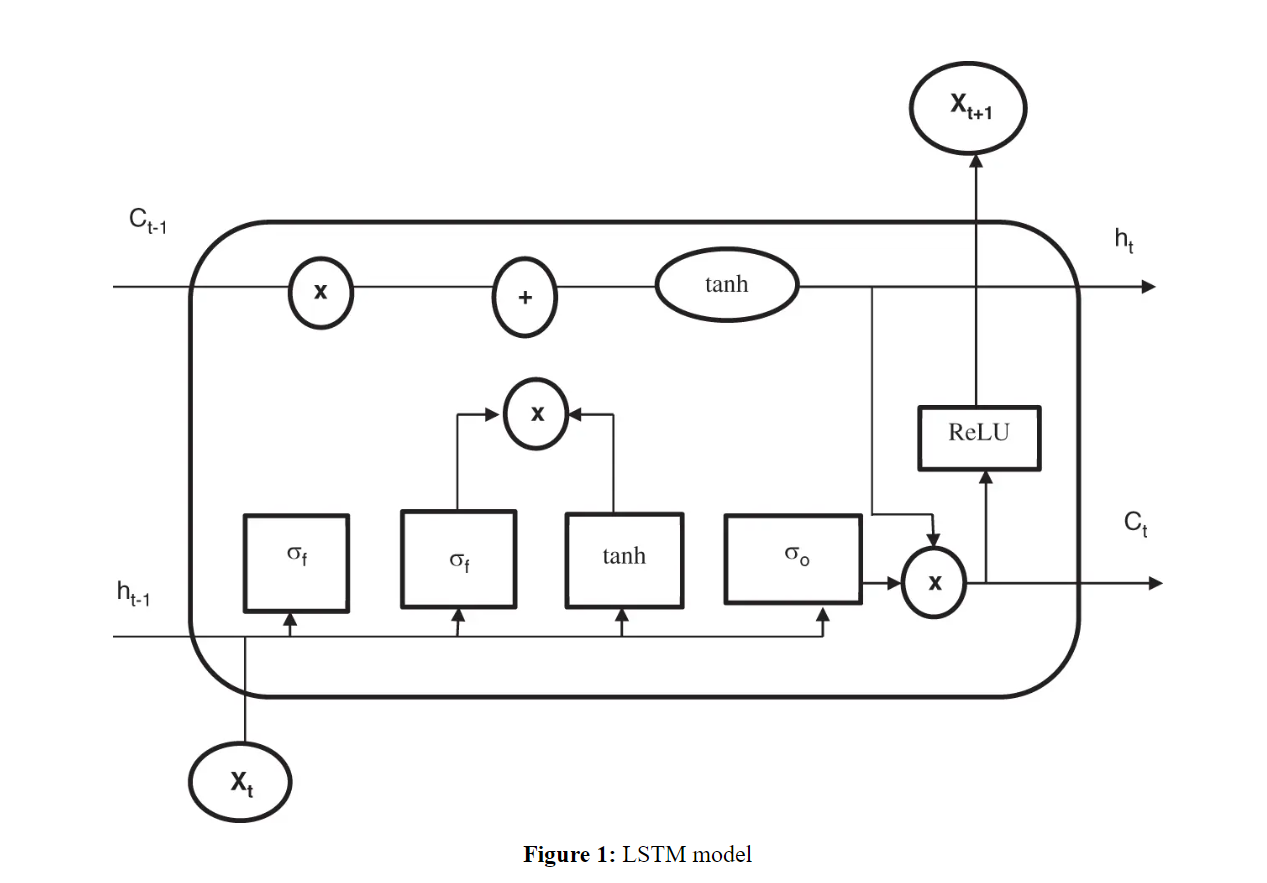
Linear Component Analysis

AutoRegressive Integrated Moving Average (ARIMA) is a widely used statistical method for time-series forecasting, particularly effective for modeling linear relationships in time-series data. The ARIMA model is denoted as ARIMA (), where represents the number of lag observations in the AutoRegressive (AR) part, refers to the number of differencing steps needed to make the series stationary (Integration), and signifies the number of lagged forecast errors in the Moving Average (MA) part. The general form of the ARIMA model is expressed as:

Here, is the actual value at time *t*, *c* is a constant,  are the coefficients for the AR part, ​ are the coefficients for the MA part, and  represents the error at time *t*. In our analysis, ARIMA was identified as the best fit for forecasting, where p=2 indicates the use of two lagged observations in the AR part, d=2 reflects that the series was differenced twice to achieve stationarity, and q=16 incorporates sixteen lagged forecast errors in the MA part.

LSTM

Long Short-Term Memory (LSTM) is a specialized form of recurrent neural network (RNN) that excels in handling time-series data and sequential tasks, such as forecasting and natural language processing. Unlike traditional RNNs, which struggle with learning long-term dependencies due to the vanishing gradient problem, LSTMs utilize a gating mechanism that helps preserve information over longer time periods. These gates—input, forget, and output—control the flow of information through the LSTM cell, allowing it to retain or discard information as needed.[6]



An LSTM cell is composed of the following components:

1. Forget Gate: This gate decides which information should be discarded from the cell state.
2. Input Gate: This gate determines which new information will be added to the cell state.
3. Cell State Update: A candidate cell state ​ is created, which is modulated by the input gate.

The cell state is updated as follows:

1. Output Gate: This gate controls what information from the cell state will be output.

Finally, the hidden state is updated:

Here, is the input at time , is the hidden state from the previous time step, ​ is the cell state, and and are the sigmoid and hyperbolic tangent activation functions, respectively. The weight matrices ​, , ​, and are learned during training, and the biases and are added to each gate's computation.

By leveraging this structure, LSTM models are able to capture both short- and long-term dependencies, making them highly effective for time-series forecasting and other sequential tasks.

DUNG BEETLE OPTIMIZER (DBO)

The Dung Beetle Optimizer (DBO) is inspired by the behavior of dung beetles, which push balls of dung to use as food or breeding chambers. The optimization algorithm is based on the natural tendency of these beetles to align their movements with celestial bodies (such as the sun or moon) to move in a straight line, which ensures they reach their destination efficiently. DBO balances exploration (searching for new solutions) and exploitation (focusing on promising solutions) by mimicking how beetles adjust their movement based on external guidance and random perturbations.[12]

Mathematical Model:

At each iteration, the position of a beetle is updated based on two main factors:

1. Attraction towards the best-known solution: This mimics the beetle's alignment with a celestial body.
2. Randomness for exploration: This introduces some degree of randomness in the beetle's movement, helping avoid local minima.

The position update for a dung beetle can be described as:

where:  is the current position of a dung beetle at iteration *t*, is the best solution found so far, controls the strength of attraction towards ,   adds randomness to promote exploration, is a random number, typically drawn from a uniform distribution.

The process repeats, with the beetles moving closer to the optimal solution while randomly exploring the search space to avoid getting stuck in local optima.

PARTICLE SWARM OPTIMIZATION (PSO)

PSO is a population-based optimization algorithm inspired by the social behavior of birds or fish, where each individual (called a "particle") adjusts its trajectory based on its own experiences and the experiences of its neighbors. Particles "fly" through the solution space, seeking the optimal solution by updating their velocity and position iteratively.[9]

Mathematical Model:

The PSO algorithm works by adjusting the velocity and position of each particle based on two main influences:

1. Personal best position: The best position the particle has visited so far.
2. Global best position: The best position found by the entire swarm.

The velocity and position of each particle are updated using the following equations:

here: is the velocity of particle at iteration t, is the position of particle at iteration t, ​ is the particle's personal best position, is the global best position found by the swarm, w is the inertia weight that controls the balance between exploration and exploitation, and are cognitive and social acceleration coefficients that control the influence of personal and global best positions, and are random numbers between 0 and 1.

The algorithm iteratively moves particles towards the global optimum by balancing local and global exploration.

GENETIC ALGORITHM (GA)

The Genetic Algorithm (GA) is a metaheuristic inspired by the process of natural selection and genetics. It mimics biological evolution, where a population of candidate solutions (chromosomes) evolves over time through processes like selection, crossover (recombination), and mutation.[16]

Mathematical Model:

The steps in GA are:

1. Selection: Select individuals from the population based on their fitness, with fitter individuals having a higher probability of being selected. This mimics "survival of the fittest."

Where ) is the probability of selecting individual i, () is the fitness of individual , and is the population size.

1. Crossover: Two parent solutions are selected to produce offspring through recombination. This is analogous to genetic recombination in biological reproduction.

The offspring inherits traits from both parents.

1. Mutation: Introduce small, random changes to the offspring to maintain diversity in the population and explore new areas in the solution space.
2. Replacement: Replace part of the old population with new offspring, creating the next generation. The algorithm repeats these steps until the population converges to a solution with a desired fitness or after a predefined number of generations.

GA balances exploration (through crossover and mutation) and exploitation (through selection) to search the solution space effectively.

GRAVITATIONAL SEARCH ALGORITHM (GSA)

GSA is an optimization algorithm inspired by Newtonian gravity. In GSA, each candidate solution is modeled as an object with mass, and the gravitational attraction between objects drives the search process. The heavier an object (i.e., the better the solution), the stronger its pull-on other objects.[8]

Mathematical Model:

The force acting on a mass iii due to another mass is calculated as:

where: is the gravitational force exerted on object by object at time , is the gravitational constant, which decreases over time to reduce exploration, and are the masses of objects I and j, is the Euclidean distance between objects and j, and are the positions of objects and , ϵ is a small constant to avoid division by zero.

The acceleration of object iii is then calculated as:

where is the total gravitational force on object .

The velocity and position of each object are updated as follows:

The algorithm iteratively updates the velocities and positions of the objects, with the heavier objects pulling others towards them. The process continues until convergence.

RED DEER ALGORITHM (RDA)

The Red Deer Algorithm (RDA) is based on the behavior of red deer during mating season. It focuses on two key phases: roaring competition and harem competition. In the roaring phase, dominant males (stags) are selected based on the loudness of their roars (fitness), and in the harem competition, these stags compete for hinds (females), influencing the positions of other individuals.[10]

Mathematical Model:

1. Roaring Competition: Stags roar to attract females, with the probability of becoming dominant based on fitness. Stags with louder roars (better fitness) have a higher chance of becoming dominant.

where is the probability of stag iii being dominant, is its fitness, and NNN is the population size.

1. Harem Competition: After selecting the dominant stags, they compete for the harem (group of females). The hinds follow the dominant stags, updating their positions based on the leader’s position:

where ​ is the position of the dominant stag, and β are control parameters that balance the attraction to the leader and random exploration.

This process mimics natural competition and selection, allowing the population to explore the search space and converge to optimal solutions based on the dominance of individuals in the population.

ALGORITHM STEPS OF PROPOSED OPTIMIZER:

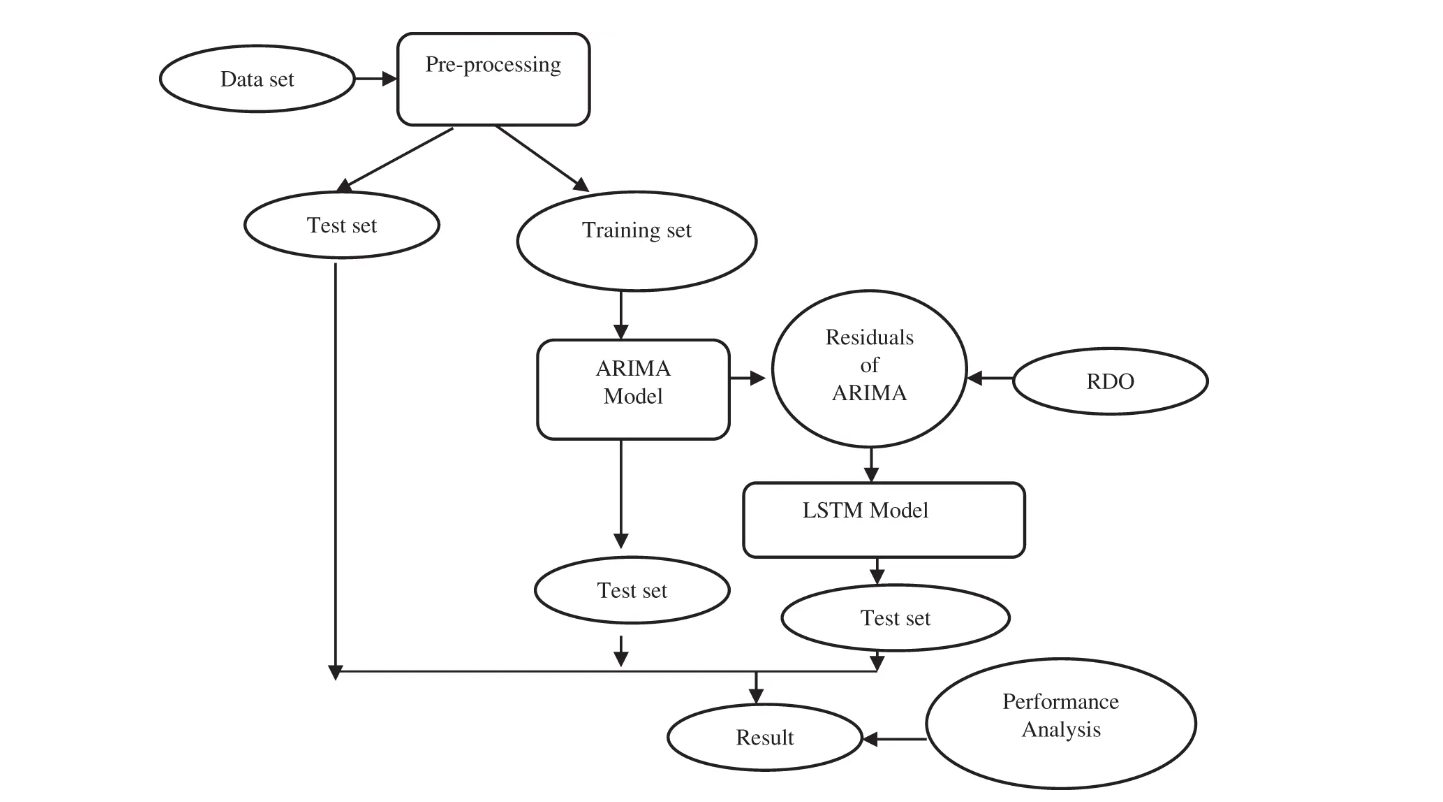
1. Initialize Population: Randomly generate a population of individuals.
2. Fitness Evaluation: Calculate the fitness for each individual based on the objective function.
3. Apply Hybrid Update Mechanism:

* GA: With probability ​, perform crossover and mutation.
* PSO: With probability ​, update velocity and position of individuals using the PSO velocity update equation.
* GSA: With probability ​, apply the gravitational force and acceleration update.
* RDA: With probability ​, apply random drift to the population.
* DBO: With probability ​, move individuals in the direction of the global best using the DBO position update.

1. Selection and Elitism: Carry the best individual from the current generation to the next generation (elitism).
2. Termination: Repeat steps 2-4 for a pre-specified number of generations or until convergence.
3. Mathematical Summary of Update Rules:

For individual in the population, at each iteration , the position updated using a hybrid rule:

DBO-PSO-GA-GSA-RDO ARCHITECTURE





DATA IDNTERPRETATION AND TRANSFORMATION

AREA OF STUDY

To check and verify our model performance we took AQI data from different cities across India, but due to computation cost we only applied it on Delhi NCR region.

The data used in this project has been sourced from the Central Pollution Control Board (CPCB), which is the official portal of the Government of India. The CPCB has made the data publicly available and can be accessed at their website: [https://cpcb.nic.in](https://cpcb.nic.in/).

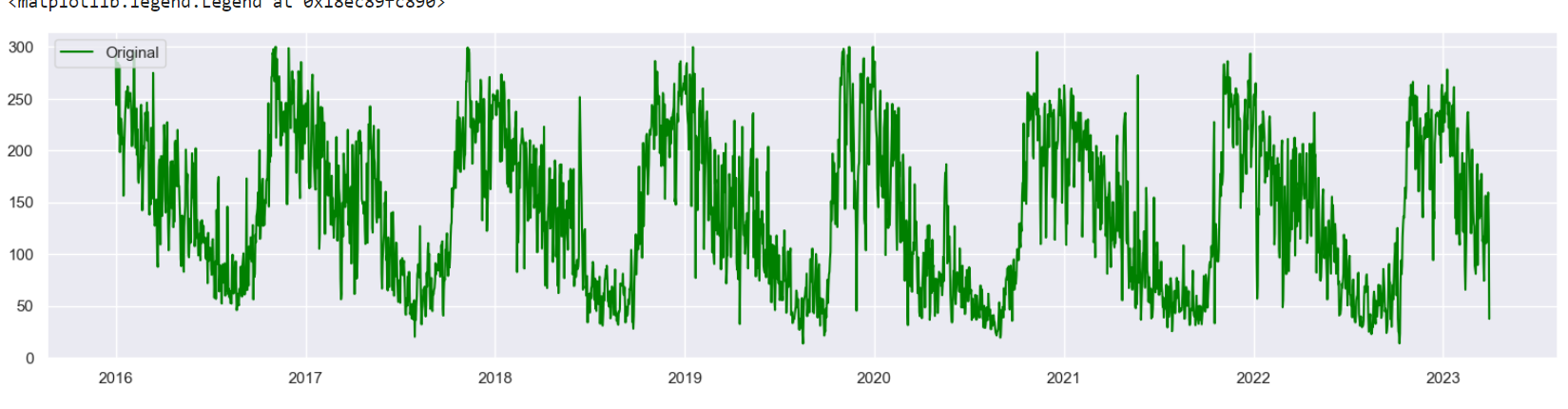
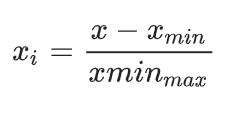


Fig: PM2.5 vs Time

DATA PROCESSING

We divided our data into three-part, 80 percent for training, 5 percent for validation and 15 percent for testing. To enhance the model training, normalized that data in range of [0, 1].



And during model evaluation transformed that to the actual readings:

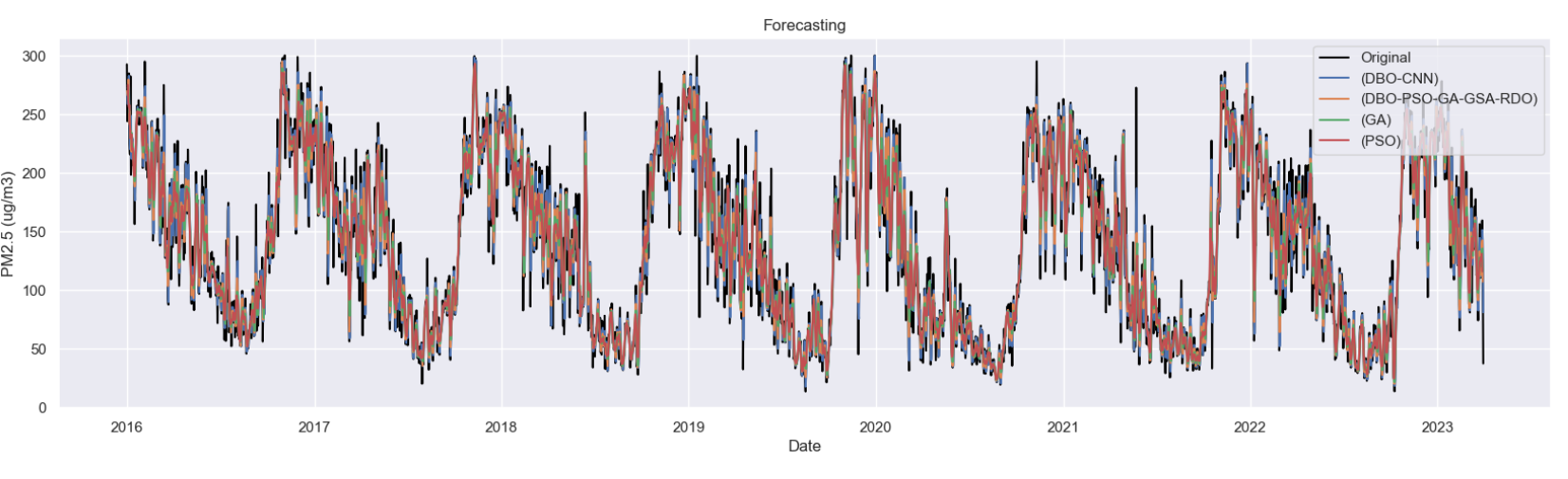


For model evolution we perform standard indicators: root mean square error RMSE, coefficient of determination R2, and mean absolute error MAE:

MODEL PREDICTION & RESULTS

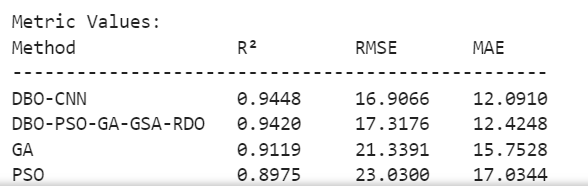
FORECAST COMPARISON AND ANALYSIS

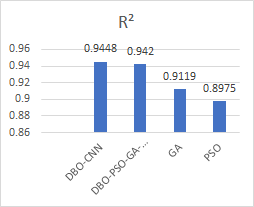
During forecasting, we composed linear and non-linear components as part of the study. Which shows us that DBO-CNN model outperform our model. Still we achieved a notable result.

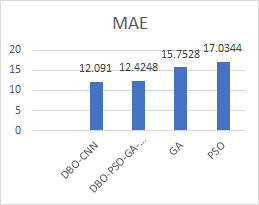


RESULT AND ANALYSIS

In Delhi our model achieved 0.9420, 17.3176 and 12.4248.







DISCUSSION AND CONCLUSION

Air pollution is a big challenge for human civilization. Prediction of this kind of phenomena will help us to stay aware when to go outside. Traditional models are unable to catch the underline patters. Our approach shows us that this may be the way to get more accurate model.

Although DBO-CNN outperform this model, but using CNN this model can achieve a better result in future.

Due to low power computation machine, we avoid other factors but they play a key role in this study. Including those factors in the model may change the game.

REFERENCES

1. <https://doi.org/10.1016/j.matpr.2020.08.187>
2. <https://doi.org/10.1016/j.sbspro.2014.10.056>
3. <https://doi.org/10.1016/j.jclepro.2020.121169>
4. <https://doi.org/10.1016/j.atmosenv.2012.06.004>
5. <https://doi.org/10.1016/j.jclepro.2022.134656>
6. https://doi.org/10.1007/s10462-020-09838-1
7. <https://doi.org/10.1007/s00500-023-07997-1>
8. <https://doi.org/10.1016/j.ins.2009.03.004>
9. <https://doi.org/10.1186/s40537-024-00926-5>
10. DOI:10.32604/csse.2022.024303
11. <https://doi.org/10.1016/j.atmosenv.2022.119347>
12. <https://doi.org/10.1038/s41598-023-36620-4>
13. <https://doi.org/10.1016/j.neucom.2021.09.051>
14. [httpswu://doi.org/10.1038/s41598-023-39838-4](https://doi.org/10.1038/s41598-023-39838-4)
15. <https://doi.org/10.1016/j.egyr.2022.02.188>
16. https://doi.org/10.1007/s10710-024-09493-3