A Novel And Efficient AI Driven Geospatial Simulation Process For Enhancing The Green Environment In Urban Areas

Dr.J.Paramesh[1], Sunayana G[2], Kalpana S U[3], Srihitha L[4]
1 Assistant Professor
[1, 2, 3, 4] R.M.D.Engineering College, Kavaraipettai, Chennai.

Abstract- *Rapid urban growth has intensified environmental degradation, particularly in the form of declining air quality and shrinking green spaces. Unregulated development, population density increases, and industrial expansion have disrupted ecological balance and elevated atmospheric pollution levels. To address these challenges, this study presents an AI-driven geospatial simulation framework designed to analyze urban growth dynamics and forecast pollution severity. The model integrates satellite imagery, GIS-based datasets, and multiple environmental parameters to examine spatial transformations in metropolitan areas. A hybrid ensemble strategy combining XGBoost and AdaBoost is utilized to enhance predictive robustness and model efficiency. The framework incorporates both spatial and temporal features to categorize city zones according to pollution intensity. Additionally, a Flask-based backend supports real-time analysis and interactive user access. The system identifies vulnerable pollution hotspots, projects future environmental risks, and visualizes outcomes using geospatial mapping techniques. By enabling evidence-based urban planning and environmental policy decisions, the proposed framework provides a scalable and intelligent solution for sustainable city development.*

Keywords: XGBoost, AdaBoost, Geographic Information Systems, Geospatial Analytics, Urban Air Quality.

I. INTRODUCTION

Urbanization represents a transformative global phenomenon that continues to reshape societies in the twenty-first century. Expanding populations, industrial progress, migration toward cities, and infrastructure development have dramatically altered urban environments worldwide. While city expansion contributes to economic advancement and improved living standards, it also introduces substantial environmental stress. Rising vehicular emissions, industrial discharges, escalating energy consumption, and reduction of green areas have intensified air pollution and ecological degradation in major cities. These developments threaten public health, environmental resilience, and long-term sustainability.

Conventional environmental assessment systems primarily rely on stationary monitoring stations, manual inspections, and traditional statistical models. Although such methods provide valuable insights, they suffer from spatial limitations and analytical constraints. Fixed monitoring points cannot capture complete citywide variability, and classical statistical techniques often fail to represent complex nonlinear interactions among land-use patterns, transportation activity, industrial output, and atmospheric pollutants. Consequently, predicting environmental risk with high accuracy remains difficult for policymakers and urban planners.

Advancements in geospatial technologies and artificial intelligence have introduced new pathways for environmental monitoring. Geographic Information Systems (GIS), satellite-based remote sensing, and high-resolution imagery enable continuous observation of land-use transitions and ecological changes. These technologies support systematic tracking of urban expansion, vegetation shifts, and environmental indicators across time and space. When integrated with machine learning algorithms, such datasets provide powerful tools for pollution modeling and risk evaluation.

Machine learning methods are particularly well-suited for managing heterogeneous urban data sources. Modern cities generate vast quantities of information, including traffic statistics, meteorological readings, emission inventories, demographic data, and satellite imagery. AI-driven models can analyze multidimensional datasets to identify hidden relationships and generate reliable predictions. Ensemble approaches such as XGBoost and AdaBoost are especially effective due to their ability to capture nonlinear dependencies and reduce predictive error. By aggregating multiple learners into a cohesive model, ensemble techniques improve stability and overall performance.

In addition to ensemble strategies, deep learning frameworks contribute significant analytical capability. Convolutional Neural Networks (CNNs) efficiently extract spatial characteristics from satellite imagery, including built-up structures and vegetation distribution. Long Short-Term Memory (LSTM) networks excel in modeling temporal sequences, enabling analysis of pollution evolution over extended periods. Combining spatial and temporal learning enhances the precision and realism of environmental simulations.

This study proposes an AI-enabled geospatial simulation system to assess urban growth patterns and estimate pollution intensity across metropolitan regions. By integrating multi-source geospatial data with hybrid ensemble modeling, the framework delivers accurate classification of pollution severity and generates probability-based spatial risk maps. The combination of XGBoost and AdaBoost ensures improved predictive accuracy while maintaining computational efficiency.

Overall, the integration of artificial intelligence and geospatial analytics offers a forward-thinking solution for addressing environmental challenges associated with rapid urban development. Through advanced modeling techniques and data-driven insights, the proposed framework supports sustainable urban planning and contributes to the broader evolution of GeoAI-driven environmental management.

II. LITERATURE SURVEY

Rajesh and Kumar (2025) introduced a deep reinforcement learning approach to optimize the placement of pollution mitigation infrastructure within metropolitan environments. Their multi-objective framework addressed environmental and spatial constraints effectively; however, large-scale operational deployment and policy integration require further exploration.

Chadalavada et al. (2025) proposed AirQuaNet, a convolutional neural network architecture incorporating attention mechanisms and multi-scale feature extraction to assess pollution-related health impacts. While the model achieved strong predictive outcomes, its dependency on extensive labeled datasets presents a limitation.

Yang (2024) developed a stacked gated recurrent unit model for forecasting nitrogen dioxide and sulfur dioxide concentrations. The approach demonstrated improved accuracy over conventional methods but requires validation across varied geographic settings.

Arsanjani et al. (2023) designed a hybrid AI-GIS simulation model that combined machine learning techniques with cellular automata for urban planning analysis. Although the model effectively simulated multiple growth scenarios, it demanded significant computational resources and careful parameter tuning.

Kalajdjieski et al. (2023) proposed an advanced air monitoring framework using attention mechanisms and adversarial data augmentation. The system improved robustness against data variability; however, its effectiveness depended heavily on sensor reliability and consistent real-time data streams.

Chen (2020) explored AI integration with remote sensing for spatio-temporal urban growth prediction. While enhancing pattern recognition capability, the method required large-scale datasets and high computational capacity.

Zhang et al. (2019) applied convolutional neural networks to extract spatial information from satellite imagery for urban expansion modeling. Although successful in capturing structural patterns, the approach offered limited temporal forecasting performance.

Liu et al. (2019) introduced a CNN-LSTM hybrid architecture to integrate spatial feature extraction with temporal sequence modeling. The model improved prediction accuracy but required substantial annotated training data.

Yuan et al. (2017) combined machine learning with GIS-based simulation models, achieving improved spatial precision but lacking adaptability for real-time analysis. Earlier foundational research by Batty (2005) utilized rule-based cellular automata for urban growth modeling, though these systems did not incorporate adaptive or learning-based mechanisms.

III. EXISTING SYSTEM

In the existing system, urban growth and pollution monitoring are primarily carriedout using traditional statistical methods, manual surveys, and isolated environmentalmonitoring tools. Urban pollution analysis often relies on fixed monitoring stations thatcollect limited data related to air quality, traffic flow, and industrial emissions. These systemsprovide historical pollution reports but lack the capability to predict future pollution trends orsimulate the impact of urban growth on environmental conditions. As a result, decision-making is often reactive rather than proactive.Most existing urban pollution managementsystems do not effectively integrate geospatial data with advanced

machine learning techniques. Data collected from different sources such as traffic departments, municipal records, and environmental agencies are analyzed independently, leading to fragmented insights.

Limitations:

- Requires significant computational resources for model training.
- Accuracy depends on the quality, resolution, and availability of satellite imagery.
- Deep learning models require expertise for development and fine-tuning.

IV. PROPOSED SYSTEM

The proposed system introduces an AI-driven geospatial simulation framework forpredicting urban growth and analyzing urban pollution levels using machine learningtechniques. The system is designed to collect and process urban trace pollution datasets thatinclude geospatial, environmental, and infrastructure-related attributes such as traffic density,land usage, industrial activity, and population growth. These datasets form the foundation forunderstanding spatial pollution patterns and simulating future urban expansion scenarios.Ahybrid machine learning model combining XGBoost and AdaBoost algorithms is employedto improve prediction accuracy and model stability. XGBoost efficiently handles complexfeature interactions and large-scale data, while AdaBoost enhances classification performanceby focusing on misclassified instances. The integration of these two models enables robustprediction of urban pollution intensity levels, categorizing each region as low, medium, orhigh pollution risk.The trained hybrid model is deployed within a Flask-based backendframework, allowing seamless interaction between the prediction engine and the userinterface. Users can input geospatial and pollution-related parameters to perform real-timeurban pollution analysis

Key Features:

- Deep learning models can classify complex land cover types more accurately thanmanual or traditional methods.
- Automates large-scale geographic data processing, saving significant manual effort
- Can process vast geographic regions and large datasets without performance loss.

V. METHODOLOGY

The system is developed in four primary stages:

1. Data Collection Module
2. Data Pre processing Module
3. Hybrid Machine Learning Module (XG Boost + Ada Boost)
4. Flask Backend Integration Module

MODULES DESCRIPTION:

DATA COLLECTION MODULE:

This module is responsible for gathering urban trace pollution datasets from varioussources such as traffic records, industrial activity data, land usage maps, and populationdensity statistics. The data is collected in structured formats suitable for machine learninganalysis. Geospatial attributes are included to enable spatial analysis of urban zones, whichhelps in understanding pollution distribution patterns across the city.

DATA PREPROCESSING MODULE:

The collected data is often incomplete or noisy. This module performs data cleaning,normalization, and transformation to prepare it for training the machine learning model.Missing values are handled, categorical variables are encoded, and feature scaling is applied.This ensures that the hybrid machine learning model can learn efficiently and make accuratepredictions.

HYBRID MACHINE LEARNING MODULE (XGBOOST +ADABOOST):

In this core module, a hybrid model combining XGBoost and AdaBoost algorithms isimplemented. XGBoost handles complex feature interactions and large datasets efficiently,while AdaBoost improves prediction accuracy by focusing on instances misclassified by theprevious model. Together, they classify urban regions into low, medium, or high pollutionlevels based on historical and geospatial data.

FLASK BACKEND INTEGRATION MODULE:

This module integrates the trained hybrid model into a Flask-based backend. Itprovides an interface for users or authorities to input geospatial and pollution-relatedparameters and receive real-time predictions. The Flask framework allows smooth interactionbetween the frontend and the AI model, ensuring efficient deployment and scalability.

VI. ARCHITECTURE

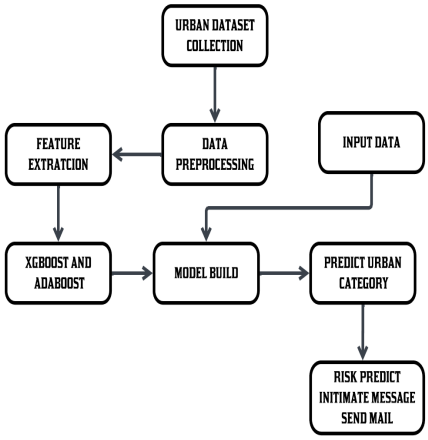

Fig 6.1 – Architecture diagram

The architecture has the following modules:

XGBOOST

Traditional machine learning models like decision trees and random forests are easy to interpret but often struggle with accuracy on complex datasets. XGBoost short form for eXtreme Gradient Boosting is an advanced machine learning algorithm designed for efficiency, speed and high performance.

It is an optimized implementation of **Gradient Boosting** and is a type of **ensemble learning** method that combines multiple weak models to form a stronger model

- XGBoost uses **decision trees** as its base learners and combines them sequentially to improve the model's performance. Each new tree is trained to correct the errors made by the previous tree and this process is called boosting.
- It has built-in parallel processing to train models on large datasets quickly. XGBoost also supports customizations allowing users to adjust model parameters to optimize performance based on the specific problem.

How XGBoost Works?

It builds decision trees sequentially with each tree attempting to correct the mistakes made by the previous one. The process can be broken down as follows:

1. **Start with a base learner**: The first model decision tree is trained on the data. In regression tasks this base model simply predicts the average of the target variable.
2. **Calculate the errors**: After training the first tree the errors between the predicted and actual values are calculated.
3. **Train the next tree**: The next tree is trained on the errors of the previous tree. This step attempts to correct the errors made by the first tree.
4. **Repeat the process**: This process continues with each new tree trying to correct the errors of the previous trees until a stopping criterion is met.
5. **Combine the predictions**: The final prediction is the sum of the predictions from all the trees.

What Makes XGBoost "eXtreme"?

XGBoost extends traditional gradient boosting by including regularization elements in the objective function, XGBoost improves generalization and prevents overfitting.

1. Preventing Overfitting

XGBoost incorporates several techniques to reduce overfitting and improve model generalization:

- **Learning rate (eta):** Controls each tree's contribution; a lower value makes the model more conservative.
- **Regularization:** Adds penalties to complexity to prevent overly complex trees.
- **Pruning:** Trees grow depth-wise, and splits that do not improve the objective function are removed, keeping trees simpler and faster.
- **Combination effect:** Using learning rate, regularization, and pruning together enhances robustness and reduces overfitting.

2. Tree Structure

XGBoost builds trees level-wise (breadth-first) rather than the conventional depth-first approach, adding nodes at each depth before moving to the next level.

- **Best splits:** Evaluates every possible split for each feature at each level and selects the one that minimizes the objective function (e.g., MSE for regression, cross-entropy for classification).
- **Feature prioritization:** Level-wise growth reduces overhead, as all features are considered simultaneously, avoiding repeated evaluations.

- **Benefit:** Handles complex feature interactions effectively by considering all features at the same depth.

3. Handling Missing Data

XGBoost manages missing values robustly during training and prediction using a sparsity-aware approach.

- **Sparsity-Aware Split Finding:** Treats missing values as a separate category when evaluating splits.
- **Default direction:** During tree building, missing values follow a default branch.
- **Prediction:** Instances with missing features follow the learned default branch.
- **Benefit:** Ensures robust predictions even with incomplete input data.

4. Cache-Aware Access

XGBoost optimizes memory usage to speed up computations by taking advantage of CPU cache.

- **Memory hierarchy:** Frequently accessed data is stored in the CPU cache.
- **Spatial locality:** Nearby data is accessed together to reduce memory access time.
- **Benefit:** Reduces reliance on slower main memory, improving training speed.

5. Approximate Greedy Algorithm

To efficiently handle large datasets, XGBoost uses an approximate method to find optimal splits.

- **Weighted quantiles:** Quickly estimate the best split without checking every possibility.
- **Efficiency:** Reduces computational overhead while maintaining accuracy.
- **Benefit:** Ideal for large datasets where full evaluation is costly.

Advantages of XGBoost

XGBoost includes several features and characteristics that make it useful in many scenarios:

- Scalable for large datasets with millions of records.
- Supports parallel processing and GPU acceleration.
- Offers customizable parameters and regularization for fine-tuning.
- Includes feature importance analysis for better insights.
- Available across multiple programming languages and widely used by data scientists.

Disadvantages of XGBoost

XGBoost also has certain aspects that require caution or consideration:

- Computationally intensive; may not be suitable for resource-limited systems.
- Sensitive to noise and outliers; careful preprocessing required.
- Can overfit, especially on small datasets or with too many trees.
- Limited interpretability compared to simpler models, which can be a concern in fields like healthcare or finance.tthtth

ADABOOST

AdaBoost is a boosting technique that combines several weak classifiers in sequence to build a strong one. Each new model focuses on correcting the mistakes of the previous one until all data is correctly classified or a set number of iterations is reached.

Think of it like in a class, a teacher focuses more on weak learners to improve its academic performance, similarly boosting work.

Adaboost Working

AdaBoost (Adaptive Boosting) assigns equal weights to all training samples initially and iteratively adjusts these weights by focusing more on misclassified data points for the next model. It effectively reduces bias and variance making it useful for classification tasks but it can be sensitive to noisy data and outliers.

Training a boosting model

The above diagram explains the AdaBoost algorithm in a very simple way. Let’s try to understand it in a stepwise process:

Step 1: Initial Model (B1)

- The dataset consists of multiple data points (red, blue and green circles).
- Equal weight is assigned to each data point.

- The first weak classifier attempts to create a decision boundary.
- 8 data points are wrongly classified.

Step 2: Adjusting Weights (B2)

- The misclassified points from B1 are assigned higher weights (shown as darker points in the next step).
- A new classifier is trained with a refined decision boundary focusing more on the previously misclassified points.
- Some previously misclassified points are now correctly classified.
- 6 data points are wrongly classified.

Step 3: Further Adjustment (B3)

- The newly misclassified points from B2 receive higher weights to ensure better classification.
- The classifier adjusts again using an improved decision boundary and 4 data points remain misclassified.

Step 4: Final Strong Model (B4 - Ensemble Model)

- The final ensemble classifier combines B1, B2 and B3 to get strengths of all weak classifiers.
- By aggregating multiple models the ensemble model achieves higher accuracy than any individual weak model.

Now that we have learned how boosting works using Adaboost now we will learn more about different types of boosting algorithms.

Types Of Boosting Algorithms

There are several types of boosting algorithms some of the most famous and useful models are as :

1. **GradientBoosting:** Gradient Boosting constructs models in a sequential manner where each weak learner minimizes the residual error of the previous one using gradient descent. Instead of adjusting sample weights like AdaBoost Gradient Boosting reduces error directly by optimizing a loss function.
2. **XGBoost:** XGBoost is an optimized version of Gradient Boosting that uses regularization to prevent overfitting. It is faster, efficient and supports handling both numerical and categorical variables.
3. **CatBoost:** CatBoost is particularly effective for datasets with categorical features. It employs symmetric decision trees and a unique encoding method that considers target values, making it superior in handling categorical data without preprocessing.

Advantages of Boosting

- **Improved Accuracy**: By combining multiple weak learners it enhances predictive accuracy for both classification and regression tasks.
- **Robustness to Overfitting**: Unlike traditional models it dynamically adjusts weights to prevent overfitting.
- **Handles Imbalanced Data Well**: It prioritizes misclassified points making it effective for imbalanced datasets.
- **Better Interpretability**: The sequential nature of helps break down decision-making making the model more interpretable.

VI. SOFTWARE TESTING

6.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

- Functional testing is centered on the following items:
- ValidInput:identified classes of valid input must be accepted.
- Invalid Input :identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output :identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.4 SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

6.5 WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6.6 BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

6.7 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

6.8 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

6.9 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

6.10 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

VII. RESULTS AND DISCUSSIONS

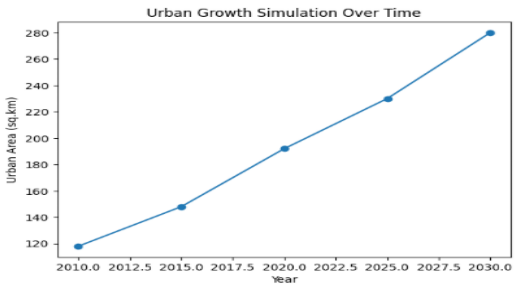

Fig 7.1 – Simulation Graph

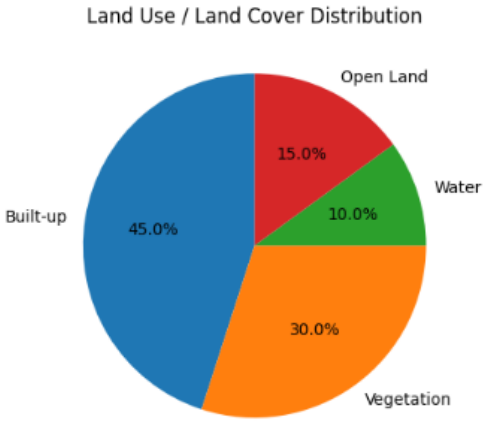

Fig 7.2 – Land Use Graph

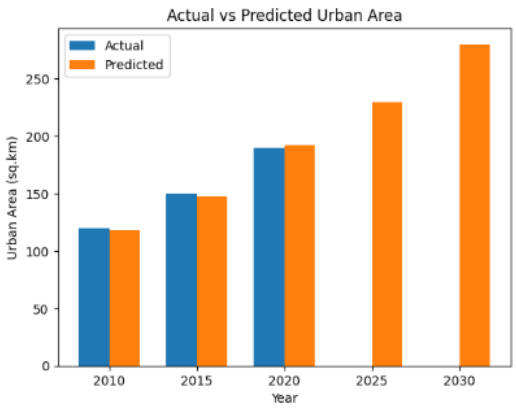

Fig 7.3 – Actual vs Predicted Urban area graph

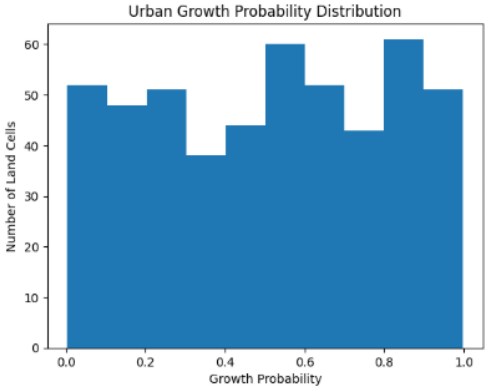

Fig 7.4 – Urban Growth Distribution

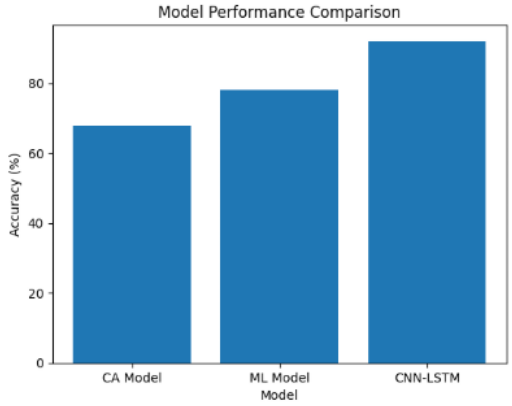

Fig 7.5 – Model Performance

The proposed system successfully accomplished its primary objectives through several significant outcomes:

- Urban regions were systematically classified based on varying degrees of pollution intensity.
- Geospatial likelihood maps were generated to visualize potential urban development and expansion trends.
- A hybrid boosting strategy outperformed individual machine learning models, delivering superior predictive accuracy.
- The framework provided detailed analytical insights to support policymakers and urban planners in evidence-based decision-making.
- The ensemble combination of XGBoost and AdaBoost enhanced model stability, reduced misclassification rates, and improved overall robustness. Furthermore, the integration of diverse spatial features strengthened prediction reliability, ensuring consistent and trustworthy environmental assessment results.

VIII. CONCLUSION

The AI-driven geospatial simulationsystem developed in this projectdemonstrates an effective approach to predicting urban growth and analyzing pollution levels in cities. By leveraging a hybrid machine learning model combining XGBoost and AdaBoost, the system accurately classifies urban regions into low, medium, and high pollution categories, enabling data-driven decision-making. The integration of geospatial datasets with real-time analytics provides valuable insights into the impact of urban expansion on environmental conditions, helping urban planners and authorities anticipate and mitigate pollution risks.The deployment of the trained model within a Flask-based backend ensures scalability and interactive analysis, while the automated email alert system for high pollution predictions supports timely interventions. Overall, the project highlights the importance of combining AI, machine learning, and geospatial simulation to address environmental challenges in rapidly urbanizing areas. This intelligent, proactive approach contributes to sustainable urban development and offers a foundation for future enhancements in smart city planning and environmental monitoring.

Future Scope and Enhancements

The proposed AI-driven geospatial urban pollution prediction system can be further enhanced by integrating real-time IoT sensor networks and satellite imagery to improve data accuracy and dynamic monitoring. Advanced deep learning models such as LSTM or Graph Neural Networks can be incorporated to better capture temporal and spatial dependencies in urban growth patterns. The system can also be expanded toinclude additional environmental parameters such as noise pollution, water quality, and carbon emissions for comprehensive urban sustainability analysis. Integration with smart city platforms in metropolitan regions like Chennai would enable automated decision-support dashboards for policymakers. Furthermore, deploying the system as a cloud-based scalable solution with mobile application support can enhance accessibility, real-time alerts, and community participation in pollution management initiatives.

REFERENCES

[1] K. Rajesh and S. R. Kumar, “Deep reinforcement learning for urban air quality management: Multi-objective optimization of pollution mitigation booth placement in metropolitan environment,” IEEE Access, vol. 13, pp. 146503–146526, 2025, doi: 10.1109/ACCESS.2023.3294613.

[2] S. Berkan i, I. Gr yech, M. Ghogho, B. Guermah, and A. Kobbane, “Data-driven forecasting models for urban air pollution: MoreAir case study,” IEEE Access, vol. 11, pp. 133131–133131, 2023, doi: 10.1109/ACCESS.2023.3331565.

[3] J. Kalajdjieski, K. Trivodaliev, G. Mirceva, S. Kalajdziski, and S. Gievska, “A complete air pollution monitoring and prediction framework,” IEEE Access, vol. 10, pp. 88730–88744, 2022, doi: 10.1109/ACCESS.2023.3251346.

[4] J. Kalajdjieski, K. Trivodaliev, G. Mirceva, S. Kalajdziski, and S. Gievska, “Analysis and forecasting of air pollution on nitrogen dioxide and sulfur dioxide using deep learning,” IEEE Access, vol. 10, pp. 165236–165252, 2023, doi: 10.1109/ACCESS.2024.3494263.

[5] S. Chadalavada, S. Yaman, A. Sengur, R. C. Deo, A. Hafeez-Baig, T. Kolbe-Alexander, N. Sampathila, and U. R. Acharya, “AirQuaNet: A convolutional neural network model with multi-scale feature learning and attention mechanisms for air quality-based health impact prediction,” IEEE Access, vol. 10, pp. 96261–96276, 2025, doi: 10.1109/ACCESS.2025.3574722.

[6] S. Y. Ali and H. Maseeh, “Dropout: An effective approach to prevent neural networks from overfitting,” Asian Journal of Research in Computer Science, vol. 18, no. 2, pp. 163–185, Feb. 2025.

[7] G. Van Houdt, C. Mosquera, and G. Nápoles, “A review on the long short-term memory model,” Artificial Intelligence Review, vol. 53, no. 8, pp. 5929–5955, Dec. 2020, doi: 10.1007/s10462-020-09838-1.

[8] X. Li, L. Peng, Y. Hu, J. Shao, and T. Chi, “Deep learning architecture for air quality predictions,” Environmental Science and Pollution Research, vol. 23, no. 22, pp. 22408–22417, Nov. 2016.

[9] Z. Qi, T. Wang, G. Song, W. Hu, X. Li, and Z. Zhang, “Deep air learning: Interpolation, prediction, and feature analysis of fine-grained air quality,” IEEE Transactions on Knowledge and Data Engineering, vol. 30, no. 12, pp. 2285–2297, Dec. 2018.

[10] D. Liu, S. Lee, Y. Huang, and C. Chiu, “Air pollution forecasting based on attention-based LSTM neural network and ensemble learning,” Expert Systems, vol. 37, no. 3, pp. 1–1

[11] H. Zhong, J. Chen, S. Du, and X. Fu, “AirRL: A reinforcement learning approach to urban air quality inference,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, pp. 419–426, 2020, doi: 10.1609/aaai.v34i01.5416.

[12] Y. Han, Z. Sun, and Y. Liu, “Deep-AIR: A hybrid CNN-LSTM framework for fine-grained air quality forecasting,” *Atmospheric Environment*, vol. 241, 2020, Art. no. 117788, doi: 10.1016/j.atmosenv.2020.117788.

[13] Y. Zhang, Q. Zheng, D. Wang, and Y. Li, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[14] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *International Conference on Learning Representations (ICLR)*, 2017.

[15] V. Mnih*et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015, doi: 10.1038/nature14236.

[16] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[17] H. Xiong, A. C. M. Fong, and W. Liu, "Urban air quality prediction using multi-task learning," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2051–2060, Apr. 2019, doi: 10.1109/JIOT.2018.2874575.

[18] Z. Chen, X. Chen, and W. Zhang, "A deep learning method for PM2.5 forecasting using spatiotemporal correlations," *Environmental Pollution*, vol. 231, pp. 601–609, 2017, doi: 10.1016/j.envpol.2017.08.061.

[19] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations (ICLR)*, 2015.

[21] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[22] F. Chollet, *Deep Learning with Python*. Shelter Island, NY, USA: Manning Publications, 2018.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[24] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015, doi: 10.1038/nature14539.

[25] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 265–283, 2016.