EnviroScan:AI-Powered Pollution Source Identifier using Geospatial Analytics

Project Statement:

Pollution monitoring systems typically measure pollutant levels but fail to identify the specific sources, limiting the ability of authorities and urban planners to take targeted and effective actions. This project leverages machine learning, weather data, and geospatial analytics to predict the likely source of pollution—such as industrial activity, vehicular traffic, agricultural burning, or natural causes. The system ingests real-time pollution sensor data, weather parameters, and surrounding location features to classify pollution sources, generate geospatial heatmaps, and issue alerts for high-risk zones, thereby supporting data-driven environmental decision-making.

Outcomes:

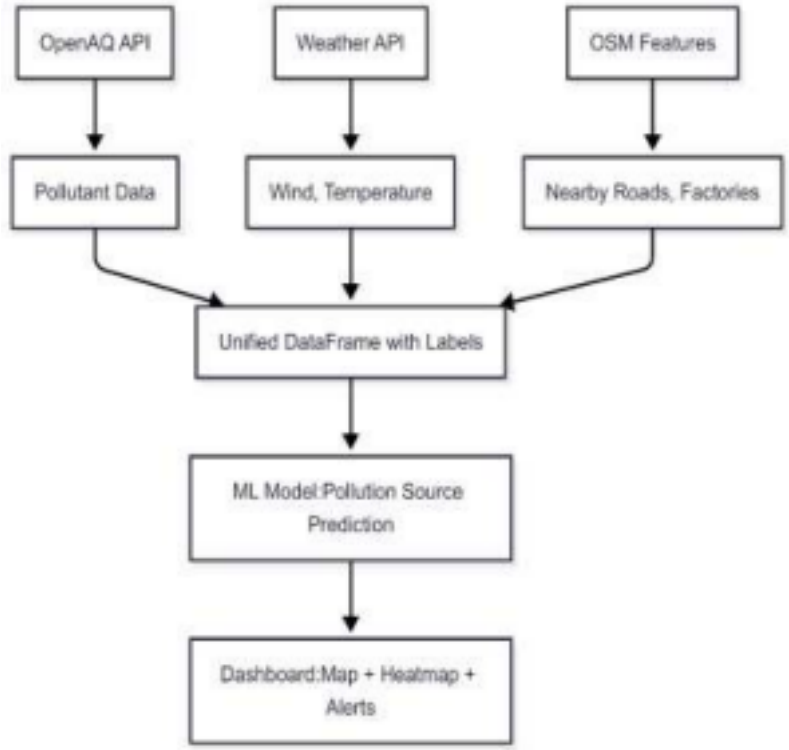
- Predict likely sources of pollution (e.g., industrial, vehicular).
- Display real-time pollution hotspots and risk zones on interactive maps.
- Trigger pollution alerts based on threshold exceedance and source confidence.
- Enable data-driven policy-making and urban planning.
- Generate reports and visualizations for environmental agencies.

Modules to be implemented

- Data Collection from APIs and location databases
- Data Cleaning and Feature Engineering
- Source Labeling and Simulation
- Model Training and Source Prediction
- Geospatial Mapping and Heatmap Visualization
- Real-Time Dashboard and Alerts
- Final Documentation and Presentation

System Architecture for Pollution Source Identifier using AI and Geospatial Analytics

Data Flow and Machine Learning Workflow for Pollution Source Identifier



Week-wise module implementation and high-level requirements with output screenshots

Milestone 1: Week 1-2

Module 1: Data Collection from APIs and location databases

- Collect air quality data (PM2.5, PM10, NO_2, CO, SO_2, O_3) from the OpenAQ API for selected locations. • Collect weather data (temperature, humidity, wind speed, wind direction) from the OpenWeatherMap API.
- Extract nearby physical features such as roads, industrial zones, dump sites, and agricultural fields using OpenStreetMap via OSMnx.
- Tag each data point with latitude, longitude, timestamp, and source API metadata. • Store the collected data in structured CSV/JSON format for preprocessing and modeling.

Module 2: Data Cleaning and Feature Engineering

• Remove duplicate entries and invalid records from raw datasets.
• Handle missing values using interpolation or mean/median imputation.
• Standardize timestamps, GPS coordinates, and pollutant units.
• Normalize pollutant and weather values for consistent model input scaling.
• Calculate spatial proximity features like distance to nearest road, industry, or dump site. • Derive temporal features such as hour of day, day of week, or season to capture pollution patterns. • Combine datasets (pollution, weather, location features) into a unified, feature-rich DataFrame.

Milestone 2: Week 3–4

Module 3: Source Labeling and Simulation

• Define rules to label pollution sources based on proximity and contextual features: • Close to main road + high NO_2 = Vehicular
• Near factory + high SO_2 = Industrial
• Near farmland + dry season + high PM = Agricultural
• Apply heuristics to assign labels like Vehicular, Industrial, Agricultural, Burning, or Natural. • Simulate labeled training data if ground-truth labels are not available.
• Validate labeling logic using domain knowledge and expert references.
• Prepare final labeled dataset for model training.

Module 4: Model Training and Source Prediction

• Split the labeled dataset into training and test sets (e.g., 80/20 split).
• Train classification models such as:
• Random Forest
• XGBoost
• Decision Tree
• Use pollutant concentrations, weather, and proximity features as input variables. • Predict the target variable: pollution_source.
• Tune model hyperparameters using GridSearchCV or RandomizedSearchCV. • Evaluate model performance using accuracy, precision, recall, F1-score, and confusion matrix. • Export the trained model using joblib or pickle for integration in the dashboard. **Milestone 3: Week 5–6**

Module 5: Geospatial Mapping and Heatmap Visualization

• Load predictions and location data into an interactive map interface.
• Use Folium or geopandas to create dynamic pollution heatmaps.
• Overlay source-specific markers (e.g., ►► = Industrial, ►► = Vehicular) on the map.
• Visualize high-risk zones with colored gradients based on pollutant severity. • Allow filtering by date, location, and predicted source category.
• Export or embed maps into the web dashboard for user interaction.

Module 6: Real-Time Dashboard and Alerts

• Build an interactive dashboard using Streamlit for real-time user access.

• Include input fields for city, coordinates, or time range selection.
• Display:
• Pollution prediction results with source labels and confidence
• Real-time alerts when pollution crosses safe thresholds
• Charts showing pollutant trends over time
• Pie charts showing predicted source distribution
• Embed the map with dynamic layers and heatmap overlays.
• Add a download option for users to export daily/weekly pollution reports. • Optionally integrate email/SMS alerts for critical conditions.

Milestone 4: Week 7–8

Module 7: Final Documentation and Presentation

• Document all data sources and APIs used in the project.
• Describe preprocessing steps, cleaning logic, and feature engineering methods. • Explain the model architecture, feature selection, and tuning process.
• Include evaluation metrics and interpretation of results.
• Add system architecture and data flow diagrams.
• Prepare the final project report in PDF or Markdown format.
• Create presentation slides summarizing methodology, visuals, and outcomes. • Upload project to GitHub with complete code, documentation, and README.
• (Optional) Deploy the Streamlit dashboard on Streamlit Cloud or Hugging Face Spaces.

Evaluation Criteria

Milestone 1 Evaluation (Week 1–2):

• Successful integration of pollution data from OpenAQ and weather data from OpenWeatherMap. • Effective extraction of location-based features using OpenStreetMap or OSMnx. • Proper storage and organization of collected datasets in a unified format (CSV/JSON). • Clean handling of missing values, duplicates, and coordinate inconsistencies. • Clear documentation of the data collection and preprocessing pipeline.

Milestone 2 Evaluation (Week 3–4):

• Accurate implementation of spatial and contextual feature engineering.
• Correct calculation of distances to nearby pollution sources (e.g., roads, factories, dumps). • Logical and consistent labeling of pollution sources using rule-based or simulated methods. • Balanced and well-distributed labeled dataset prepared for model training.
• Clear visualizations showing feature relationships and label distributions.

Milestone 3 Evaluation (Week 5–6):

• Correct training and evaluation of machine learning classification models (e.g., Random Forest, XGBoost). • Use of appropriate metrics such as accuracy, precision, recall, F1-score, and confusion matrix. • Effective hyperparameter tuning and model optimization using cross-validation techniques. • Export of best-performing model using joblib or pickle.

• Clear tracking of model performance and feature importance analysis.

Milestone 4 Evaluation (Week 7–8):

• Functional and user-friendly Streamlit dashboard or Flask app displaying real-time predictions. • High-quality geospatial visualizations (heatmaps, marker maps) of pollution levels and predicted sources. • Proper implementation of pollution alerts and confidence scores on the UI.
• Availability of interactive plots and data download options in the dashboard.
• Comprehensive final documentation with system architecture, data flow diagram, and user instructions. • Well-structured GitHub repository with source code, trained models, and deployment guide.