10 Pearls

AirSense Karachi – AI-Powered AQI Intelligence

February 18, 2026

Overview

AirSense Karachi is an AI-powered platform designed to monitor, predict, and forecast the Air Quality Index (AQI) in real time for the city of Karachi. It integrates environmental data collection, predictive modeling, hazard alert generation, and interactive visualization through a Streamlit dashboard.

The system collects pollutant and meteorological data (PM2.5, PM10, CO, NO_2, SO_2, Ozone, temperature, wind speed) via the Open-Meteo API. Temporal and derived features, such as AQI change rate, are engineered to enhance prediction accuracy. All data and models are stored in **MongoDB Atlas**, ensuring scalability, versioning, and real-time access.

For prediction and forecasting, multiple models are implemented, including **Ridge Regression, Random Forest, XGBoost, and Prophet** for time-series AQI trends. The best-performing model is automatically selected based on standard regression metrics (MAE, RMSE, R^2) and deployed for live monitoring.

The platform provides real-time AQI monitoring, 3-day forecasting, and dynamic hazard alerts, while the Streamlit dashboard enables interactive exploration of historical trends, model comparisons, and scenario simulations. AirSense Karachi is designed for scalability, future extension to other cities, and potential integration with IoT sensors for city-wide air quality intelligence.

Goals

The primary goals of AirSense Karachi are to:

- **Real-Time AQI Prediction** – Provide accurate, instant AQI predictions using environmental sensor data to inform citizens about current air quality conditions.
- **3-Day AQI Forecasting** – Enable proactive planning by predicting AQI trends for the next three days based on recent environmental and temporal data using Prophet.
- **Hazard Alert Generation** – Automatically classify AQI levels into categories such as Safe, Unhealthy, Very Unhealthy, and Severe Hazard, providing actionable alerts in real time.
- **Model Evaluation and Selection** – Implement multiple machine learning and time-series models, compare their performance, and automatically select the best model for deployment.
- **Data and Model Management** – Maintain scalable storage of features, alerts, model versions, and evaluation metrics in MongoDB Atlas, ensuring traceability and reproducibility.
- **Interactive Dashboard** – Provide a user-friendly Streamlit interface for real-time AQI monitoring, historical trend visualization, model comparison, and scenario testing.

Methodology

AirSense Karachi follows a **three-layer approach** combining data acquisition, predictive modeling, and user-facing visualization.

1. Data Acquisition and Preprocessing

- AQI and environmental data are collected via the **Open-Meteo API**, including pollutants (PM2.5, PM10, CO, NO_2, SO_2, Ozone) and meteorological parameters (temperature, wind speed).

- Temporal features (hour, day, month, day of week) and derived features such as AQI change rate are engineered.
- Data cleaning handles missing values, converts datetime formats, and aligns feature order for model consistency.
- All preprocessed data is stored in **MongoDB Atlas**, enabling scalable and real-time access.

2. Predictive Modeling and Forecasting

- Multiple models are implemented for AQI prediction: **Ridge Regression, Random Forest, XGBoost**, and **Prophet** for time-series forecasting.
- The training pipeline automates feature preparation, model training, and evaluation using **MAE, RMSE, and R^2**.
- The best-performing model is automatically selected and registered in MongoDB for deployment.
- Prophet enables 3-day AQI forecasting, providing multi-step predictions based on historical trends and recent environmental data.

3. Hazard Alert System

- AQI thresholds classify air quality into Safe, Unhealthy, Very Unhealthy, or Severe Hazard categories.
- Alerts are stored in MongoDB and displayed dynamically on the Streamlit dashboard, enabling real-time hazard monitoring.

4. Interactive Visualization and Deployment

- The **Streamlit dashboard** allows users to explore live AQI readings, simulate environmental scenarios, visualize historical trends, and compare model performance.
- CI/CD pipelines using **GitHub Actions** automate feature extraction, model training, evaluation, and deployment.

This methodology ensures **accurate AQI prediction, reliable forecasting, automated hazard alerts**, and a scalable platform for real-time environmental monitoring.

Specifications

1. Hardware Specifications

- Local development: Any modern PC or laptop capable of running Python-based pipelines
- Recommended RAM: 16GB+ for smooth model training and forecasting
- Storage: Minimum 50GB for datasets, model files, and MongoDB backups

2. Software Specifications

- Programming Language: Python 3.11
- Libraries and Frameworks:
    - Data Processing: `pandas`, `numpy`
    - Machine Learning: `scikit-learn`, `xgboost`
    - Time-Series Forecasting: **`prophet`**
    - Visualization: `plotly`, `matplotlib`, `streamlit`
- Database: MongoDB Atlas (cloud-hosted NoSQL)
- Development Environments: VS Code, PyCharm, Google Colab
- CI/CD: GitHub Actions for automated pipelines
- Operating System: Windows

3. Functional Specifications

- Real-time AQI prediction and hazard alert generation
- 3-day AQI forecasting using Prophet and ML models
- Historical AQI trend visualization and analysis
- Model evaluation and comparison (Ridge Regression, Random Forest, XGBoost, Prophet)
- Interactive Streamlit UI for scenario simulation, monitoring, and model explainability
- Automatic storage of features, alerts, model versions, and evaluation metrics in MongoDB Atlas

System Architecture

The AirSense Karachi platform is structured into three main layers: **Data Layer**, **Model Layer**, and **Application/UI Layer**. Each layer plays a specific role in ensuring reliable, real-time AQI prediction, visualization, and analytics. The system is designed for scalability, modularity, and easy maintenance.

1. Data Layer

Purpose: Collect, process, and store raw and engineered data for prediction models.

Components & Workflow:

1. **Data Collection**
    - **Air Quality Data:** Pulled from Open-Meteo's air quality API for Karachi (PM2.5, PM10, CO, NO_2, SO_2, O_3).
    - **Weather Data:** Pulled from Open-Meteo's historical weather API (temperature, wind speed).
    - Data is fetched using Python `requests` and converted into **pandas DataFrames**.
2. **Data Cleaning & Normalization**
    - Column names standardized (e.g., `pm2_5` → `pm25`).
    - Data merged on timestamp for unified features.
3. **Storage**
    - Raw and feature-engineered data is stored in **MongoDB Atlas**.
    - Collections:
        - `raw_data`: Stores merged raw air quality & weather data.
        - `features`: Stores feature-engineered data ready for modeling.
4. **Feature Engineering Pipeline (`feature_pipeline.py`)**
    - Creates **time-based features**: hour, day, month, weekday, weekend.
    - Pollutant rolling statistics: mean, standard deviation, lag features.
    - Weather features: rolling mean and lag.
    - Multi-pollutant AQI proxy computed as weighted sum of pollutants.
    - AQI trend features: rolling mean and lag values.

- Resulting features stored back in MongoDB for modeling.

2. Model Layer

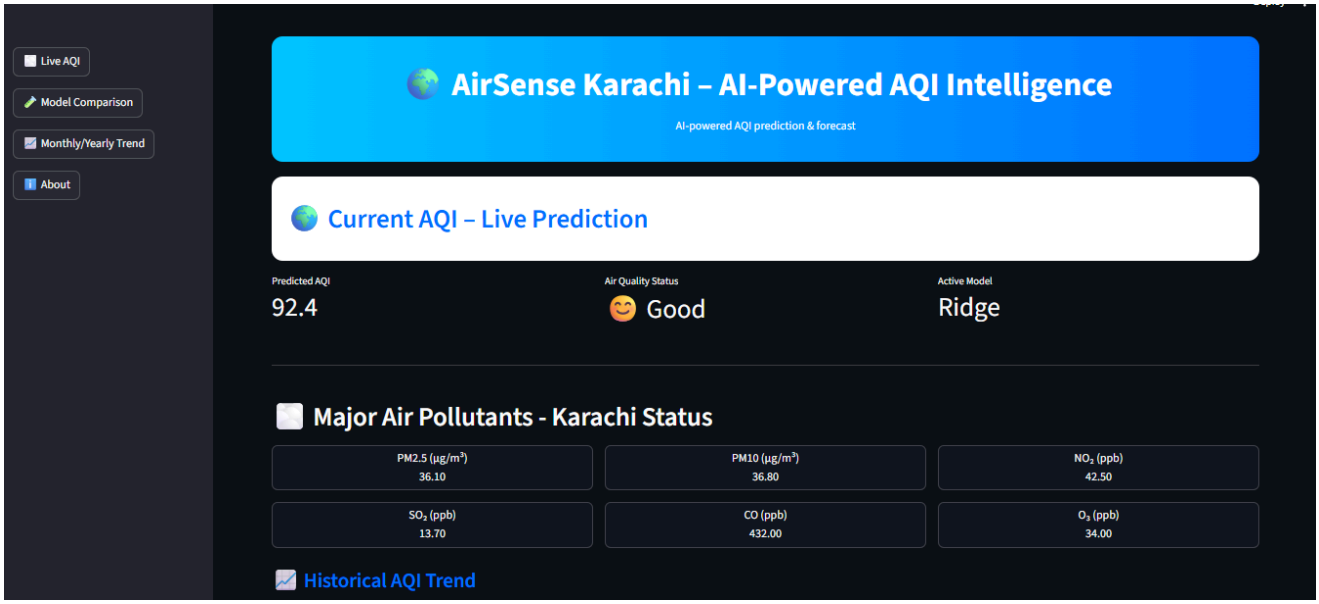
Purpose: Train, evaluate, and store machine learning and time-series models for AQI prediction.

Components & Workflow:

1. **Training Pipeline (`training_pipeline.py`)**
   - Loads feature data from MongoDB.
   - Sorts by timestamp to preserve time-series integrity.
   - Creates **target variable**: next-hour AQI (`AQI_target`).
   - Removes columns that may cause leakage.
   - Splits dataset into **train (80%)** and **test (20%)**.
2. **Models Trained:**
   - **Ridge Regression:** Regularized linear model.
   - **Random Forest Regressor:** Ensemble tree model capturing non-linear patterns.
   - **XGBoost:** Gradient boosting optimized for regression tasks.
   - **Prophet:** Time-series model for AQI forecasting.
3. **Evaluation & Selection:**
   - Metrics: **MAE, RMSE, R^2, Robust Score**.
   - Best model selected based on lowest **Robust Score (RMSE)**.
   - Models saved to disk and metadata stored in `model_registry` in MongoDB.
4. **Output:**
   - Active model used for predictions in the Application/UI Layer.
   - Model metrics available for **comparison and visualization**.

3. Application/UI Layer

Purpose: Provide end-users with a **dashboard** for real-time AQI monitoring, historical trends, and forecast insights.

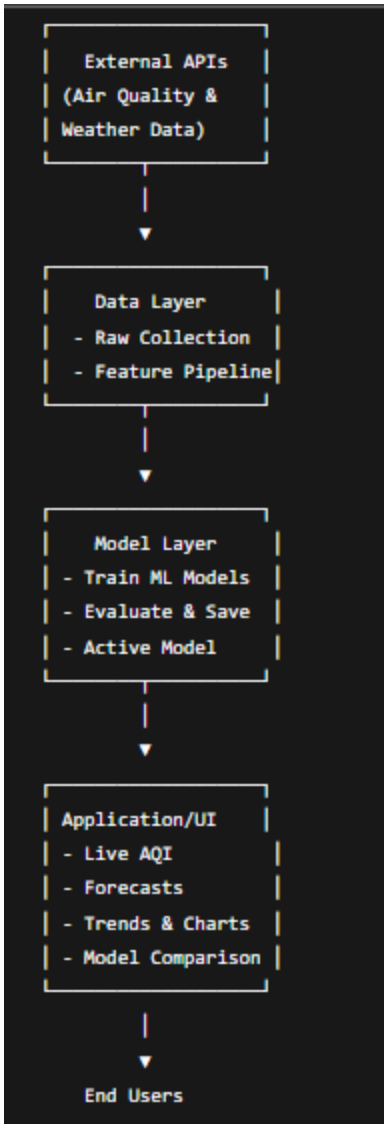


Components & Features:

1. **Framework:**
    - **Streamlit** for interactive web interface.
    - **Plotly** for dynamic charts and trend visualization.
2. **Data & Model Integration:**
    - Fetches feature data from MongoDB.
    - Loads the **active predictive model** from `model_registry`.
    - Aligns input features to model requirements for real-time prediction.
3. **Key Dashboard Features:**
    - **Live AQI Prediction:** Shows current AQI, status, and active model.
    - **Pollutant Cards:** Interactive display of PM2.5, PM10, NO_2, SO_2, CO, O_3.
    - **Historical & Forecast Charts:**
        - Past AQI trends.
        - 3-day forecast with predicted AQI and trend insights.

        - Combined chart with shaded forecast area and percentage change.
    - **Model Comparison:**
        - Displays trained models, metrics (MAE, RMSE, R^2), and highlights the best model.
    - **Monthly & Yearly Trends:**
        - Aggregated visualizations of AQI over time.
    - **About Section:** Explains platform purpose, features, and AI methodology.

4. System Flow Diagram



Flow Explanation:

1. **External APIs** provide air quality and weather data.
2. **Data Layer** cleans, merges, and transforms the data into features for prediction.
3. **Model Layer** trains multiple ML and time-series models, evaluates them, and selects the best performing one.
4. **Application Layer** fetches data and models to display live AQI, trends, forecasts, and model metrics to the user.

This **three-layer architecture** ensures:

- **Real-time predictions** using latest data.
- **Accurate forecasting** using robust models.
- **Interactive insights** for end-users with transparency on model performance.
- **Modular design** for easy extension, e.g., adding more cities, pollutants, or models.

Challenges

- **Data Leakage during Model Training**
  Problem: The Ridge regression model initially showed an RMSE of zero, indicating that the model was learning from features that contained target information.
  Solution: We removed all columns that could leak information about the target, such as `AQI`, `AQI_target`, and raw pollutant values, ensuring the model learned from independent features only.

- **LSTM Model Incompatibility and Performance Issues**
  Problem: LSTM was running locally but not supported on Python 3.11 and had very long training times, making it unsuitable for deployment.
  Solution: Replaced LSTM with the Prophet time-series model, which is fast, reliable, and supported in the deployment environment for AQI forecasting.

- **Manual CI/CD Pipeline Execution**
  Problem: The data fetching, feature computation, and model training pipeline had to be

triggered manually, increasing workload and the chance of errors.
Solution: Implemented both manual and automatic triggers for the CI/CD pipeline to allow scheduled runs and on-demand execution.

- **Incomplete or Missing Data**
  Problem: Some timestamps or pollutant readings were missing in air quality and weather datasets, disrupting feature computation.
  Solution: Added validation checks for required columns and used backfilling for rolling statistics to handle missing values without breaking the pipeline.

- **High Dimensionality from Lag and Rolling Features**
  Problem: Adding multiple lag and rolling statistics features could cause feature explosion and increase overfitting risk.
  Solution: Selected only relevant lag and rolling features, and removed constant or non-numeric columns to keep the feature set manageable.

- **Real-Time Prediction Latency**
  Problem: Loading the model and fetching data from MongoDB for every user interaction slowed down the dashboard.
  Solution: Utilized Streamlit caching (`@st.cache_resource` and `@st.cache_data`) to store models and data temporarily, improving dashboard responsiveness.

- **Managing Multiple Models and Selecting the Best Model**
  Problem: Difficulty in tracking multiple trained models and identifying the best one for deployment.
  Solution: Created a centralized MongoDB model registry to store all models, their metrics, and active status, allowing automatic selection of the best-performing model.

- **Short-Term AQI Forecast Accuracy**
  Problem: Predicting 3-day AQI forecasts required careful handling of historical data and model predictions.
  Solution: Combined historical AQI trends with model predictions to generate reliable

short-term forecasts for the dashboard.

- **Data Normalization and Visualization Consistency**
  Problem: Different APIs provided inconsistent column names and data formats, which could break the pipeline.
  Solution: Standardized all pollutant column names and timestamps across data pipelines, ensuring smooth integration and consistent visualizations in Streamlit charts.

Conclusion

AirSense Karachi successfully delivers an **AI-powered air quality monitoring and forecasting platform**, integrating feature engineering, predictive modeling, and interactive visualization. The system predicts AQI in real-time, tracks major pollutants, and provides historical and 3-day forecast insights. Challenges such as data leakage, LSTM limitations, and CI/CD automation were resolved, ensuring reliable predictions and smooth dashboard operation.

Future work includes exploring advanced time-series models (GRU/ARIMA), expanding pollutant coverage, real-time sensor integration, predictive alerts, mobile app deployment, and fully automated CI/CD retraining for continuous model updates.