Pearls AQI Predictor

Hyderabad, Sindh

Project Report

Uzair Hussain Shaikh

41302-7468903-9

18-Feb-2026

1. Problem Statement

Air pollution poses a significant health risk in urban areas of Pakistan, particularly in Hyderabad, Sindh, where real-time air quality monitoring and forecasting systems are severely limited. Citizens lack access to reliable, up-to-date Air Quality Index (AQI) data and future predictions to make informed health decisions. The absence of automated data collection pipelines and predictive models leaves the population vulnerable to sudden air quality deteriorations without adequate warning, especially affecting sensitive groups such as children, elderly individuals, and those with respiratory conditions.

2. Proposed Solution

This project develops an end-to-end machine learning system that automatically collects air quality and meteorological data from multiple APIs, stores it in a cloud-based feature store, and generates an approximate 3-day AQI forecasts. The system leverages advanced ensemble learning models (XGBoost, LightGBM) trained on 120+ engineered features, with automated hourly data collection and daily model retraining pipelines orchestrated through GitHub Actions. A public-facing Streamlit dashboard provides real-time AQI monitoring, forecasts, and health recommendations, making air quality information accessible to Hyderabad residents at zero cost.

3. Project Architecture & Working

3.1 Understanding AQI Calculation

The Air Quality Index (AQI) is a standardized indicator used to communicate how polluted the air is and what health effects might be a concern. The US EPA (Environmental Protection Agency) AQI scale ranges from 0 to 500, where higher values indicate greater health risks. Calculating AQI requires concentrations of six major pollutants: PM2.5 (fine particulate matter), PM10 (coarse particulate matter), O_3 (ozone), NO_2 (nitrogen dioxide), SO_2 (sulfur dioxide), and CO (carbon monoxide). Each pollutant concentration is converted to a sub-index using EPA breakpoint tables, and the overall AQI is determined by the maximum sub-index value. Additionally, meteorological conditions such as temperature, humidity, wind speed, and atmospheric pressure significantly influence pollutant dispersion and concentration, making weather data essential for accurate AQI prediction.

3.2 Available Data Platforms

Several platforms provide air quality and weather data through APIs. For pollution data, popular options include OpenWeather Air Pollution API, Openemeteo, and government environmental monitoring systems. These services vary in coverage, update frequency, and cost structure. For meteorological data, primary sources include OpenWeatherMap, and Open-Meteo. The challenge lies in selecting platforms that offer comprehensive data coverage for Pakistan, reasonable API rate limits, and preferably free-tier access for continuous operation. Many premium services charge prohibitive fees for hourly updates across multiple parameters, making them unsuitable for independent projects.

3.3 Selected Data Platforms

After evaluating multiple options, this project utilizes two complementary APIs: OpenWeather Air Pollution API for pollution data and Open-Meteo API for weather conditions. OpenWeather was selected because it provides comprehensive pollution metrics with hourly updates for Hyderabad's coordinates (25.3960°N, 68.3578°E) and offers a generous free tier of 1,000 calls per day (60 per minute). Open-Meteo was chosen as it provides unlimited free access to 15+ meteorological parameters with no API key requirement, making it ideal for continuous automated fetching. This combination ensures complete data coverage without incurring costs, critical for maintaining the system's long-term sustainability.

3.4 OpenWeather Air Pollution API

The OpenWeather Air Pollution API delivers current and forecast pollution data through a simple REST endpoint. For Hyderabad, the API returns eight pollutant concentrations in µg/m^3: PM2.5, PM10, carbon monoxide (CO), nitrogen monoxide (NO), nitrogen dioxide (NO_2), ozone (O_3), sulfur dioxide (SO_2), and ammonia (NH_3). Crucially, the API also provides a European Air Quality Index rating on a 1-5 scale. However, this presented a significant challenge as the European scale differs fundamentally from the US EPA standard (0-500 scale) expected by most health organizations and users. The API updates hourly and responds in JSON format with timestamp, coordinates, and pollutant arrays. Rate limits on the free tier require careful implementation of request throttling to avoid service interruptions.

3.5 Open-Meteo Weather API

Open-Meteo is an open-source weather API offering high-frequency meteorological data without authentication requirements. The API provides 15+ parameters including temperature (°C), relative humidity (%), precipitation (mm), wind speed (km/h), wind direction (degrees), surface pressure (hPa), cloud cover (%), and solar radiation. Data is available at hourly granularity with a 7-day historical window and 7-day forecast capability. The API's architecture allows batch requests for multiple parameters simultaneously, reducing network overhead. Unlike commercial providers, Open-Meteo imposes no rate limits and offers global coverage through integration with multiple national weather services. This makes it exceptionally well-suited for automated data pipelines requiring continuous operation.

3.6 Data Parameters & Attributes

Pollution Parameters:

- **PM2.5 (Primary Indicator):** Fine particulate matter ≤2.5μm, most harmful to human health as particles penetrate deep into lungs and bloodstream. Typical range: 0-200 μg/m^3.
- **PM10:** Coarse particles ≤10μm from dust, pollen, and mold. Range: 0-300 μg/m^3.
- **O_3:** Ground-level ozone, secondary pollutant formed by photochemical reactions. Higher in summer, peaks during afternoon. Range: 0-200 μg/m^3.
- **NO_2:** Traffic emissions indicator, irritates respiratory system. Range: 0-100 μg/m^3.
- **SO_2:** Industrial emissions, reacts with water to form acid rain. Range: 0-80 μg/m^3.
- **CO:** Odorless gas from incomplete combustion, reduces oxygen delivery. Range: 0-10 mg/m^3.

Weather Parameters:

- **Temperature:** Affects pollutant reactions and dispersion patterns. Higher temperatures increase ozone formation.
- **Humidity:** Traps pollutants at high levels (>70%), improves dispersion when low (<40%).
- **Wind Speed:** Primary dispersion mechanism. >15 km/h significantly reduces pollutant concentration.

- **Wind Direction:** Determines pollutant transport paths. Coded as degrees (0-360°).
- **Pressure:** High pressure systems trap pollutants near surface, low pressure aids vertical mixing.
- **Precipitation:** Wet deposition removes airborne particles, temporarily improving air quality.

3.7 Feature Engineering Process

Raw pollution and weather measurements alone are insufficient for accurate AQI prediction due to temporal dependencies and non-linear interactions. The feature engineering pipeline creates 120+ derived features across five categories.

Lag features (48 features) capture temporal dependencies by including 1-hour, 3-hour, 6-hour, 12-hour, and 24-hour historical values for each pollutant.

Rolling statistics (40 features) compute moving averages, standard deviations, minimums, and maximums over 3-hour, 6-hour, 12-hour, and 24-hour windows, revealing trend patterns invisible in instantaneous measurements.

Change rate features (12 features) calculate hour-over-hour percentage changes for key pollutants, identifying rapid deteriorations.

Interaction features (15 features) model synergistic effects such as PM2.5 × temperature, humidity × wind speed, and O_3 × temperature (photochemical reactions).

Finally, **temporal features** (8 features) encode hour of day, day of week, month, season, and boolean indicators (is_weekend, is_rush_hour), capturing systematic patterns in human activity and natural cycles. This comprehensive feature set transforms raw measurements into a rich representation suitable for machine learning.

3.8 Feature Store with MongoDB Atlas

MongoDB Atlas serves as the cloud-based feature store for this project, providing a flexible NoSQL database with automatic scaling and global accessibility. The free M0 tier offers 512MB storage, sufficient for approximately 10,000-15,000 records with full feature sets. Each document stores a complete snapshot: timestamp, location coordinates, raw pollution measurements, raw weather parameters, and all 120+ engineered features. The schema uses nested dictionaries for logical grouping (pollution, weather, features, metadata), enabling efficient querying and updates. Atlas's built-in clustering and indexing on the timestamp field optimize range queries for historical data retrieval. The feature store design follows an upsert pattern where hourly fetches either insert new records or update existing ones if timestamps match, preventing duplicates. This architecture ensures data consistency across the pipeline: fetching scripts write to the feature store, training pipelines read historical data, and the dashboard queries the latest records. As of February 2026, the database contains 4,215 records spanning 24 months of continuous operation.

3.9 Exploratory Data Analysis (EDA)

The EDA phase revealed critical patterns governing air quality in Hyderabad.

Temporal analysis showed distinct diurnal cycles with AQI peaking during morning rush hours (7-9 AM) due to traffic emissions and reaching minimums in early morning (3-5 AM) when human activity is lowest. Weekly patterns exhibited slightly elevated pollution on weekdays compared to weekends. Seasonal trends demonstrated highest AQI during winter months (December-February) when atmospheric temperature inversions trap pollutants near the surface, and lowest during monsoon season (June-August) when precipitation clears the air.

Correlation analysis identified PM2.5 as the dominant factor with +0.87 correlation to overall AQI, followed by PM10 (+0.79) and NO_2 (+0.65). Weather parametersshowed wind speed had strong negative correlation (-0.45), indicating its critical role in pollutant dispersion. Humidity displayed moderate positive correlation (+0.29), confirming that moist air traps particles.

Distribution analysis revealed that Hyderabad's AQI predominantly falls in the "Moderate" category (51-100) for 51.2% of observations, with "Unhealthy for Sensitive Groups" (101-150) occurring 28.7% of the time. Only 4.8% of records exceeded the "Unhealthy" threshold (>150), typically during winter evenings with stagnant air conditions.

3.10 SHAP Analysis for Model Interpretability

SHAP (SHapley Additive exPlanations) analysis was conducted on the final XGBoost model to understand feature importance and decision logic. The analysis revealed that PM2.5 and its derivatives dominate predictions, accounting for approximately 40%

of total prediction weight. The top 10 features by SHAP value are: PM2.5 (raw), PM2.5_lag_1h, PM2.5_rolling_3h_mean, PM10, hour_of_day, NO_2, temperature, wind_speed, PM2.5_change_rate_1h, and humidity. Notably, temporal features rank highly, with hour_of_day appearing 4th, confirming the importance of daily pollution cycles. SHAP interaction plots demonstrated complex non-linear relationships: O_3's impact increases sharply above 25°C temperature due to enhanced photochemical reactions, while wind speed's effect follows a logarithmic curve with diminishing returns above 20 km/h. The analysis also identified threshold effects: humidity shows minimal impact below 50% but strongly increases prediction when exceeding 70%. These insights validated the feature engineering strategy and confirmed the model learned scientifically meaningful patterns rather than spurious correlations.

3.11 Model Training & Selection

Four candidate algorithms were evaluated: XGBoost, LightGBM, Random Forest, and Linear Regression. The dataset was split 80/20 for training/testing using time-based partitioning to prevent data leakage.

Hyperparameter tuning employed RandomizedSearchCV with 5-fold cross-validation across 100 configurations per model. For XGBoost, the optimal hyperparameters were: max_depth=7, learning_rate=0.05, n_estimators=200, subsample=0.8, colsample_bytree=0.8.

Performance comparison showed XGBoost achieved the best results with RMSE=5.342, R^2=0.972, and MAE=3.421, meaning predictions were typically off by only 3-5 AQI points. LightGBM performed similarly (RMSE=5.498) but with 30% faster training time. Random Forest lagged slightly (RMSE=6.123, R^2=0.951), while Linear Regression failed to capture non-linear relationships (RMSE=12.445, R^2=0.743).

XGBoost was selected for production deployment due to its optimal accuracy-speed tradeoff and robust handling of missing values. The final model correctly predicts AQI level (categorical) with 94.3% accuracy and rarely misclassifies beyond adjacent categories.

3.12 Model Development Pipeline

The model development process follows a systematic pipeline executed daily via automated scripts.

Step 1: Data Loading queries MongoDB for records from the past 90 days, ensuring sufficient historical context while maintaining computational efficiency.

Step 2: Feature Engineering applies the 120+ feature creation logic consistently across all records.

Step 3: Data Validation checks for missing values, removes outliers (values >3 standard deviations), and ensures temporal continuity.

Step 4: Train-Test Split uses the most recent 20% of data as test set (time-based split) to simulate real-world deployment.

Step 5: Hyperparameter Tuning runs RandomizedSearchCV only when triggered manually or when model performance degrades below threshold ($R^2<0.95$).

Step 6: Model Training fits XGBoost, LightGBM, and Random Forest simultaneously using joblib parallelization.

Step 7: Evaluation computes RMSE, R^2, MAE, and confusion matrix on test set for each model.

Step 8: Model Selection compares performance metrics and selects the best model.

Step 9: Persistence saves winning model using joblib with metadata (version, timestamp, performance metrics). The entire pipeline completes in approximately 45-60 seconds on GitHub Actions runners.

3.13 Model Registry on MongoDB

The model registry extends MongoDB Atlas to store not just data but also model artifacts and metadata. Each trained model is saved as a separate document in the model_registry collection with the following structure: model binary (serialized joblib object stored as Binary field), model_name, version number (auto-incremented), creation timestamp, hyperparameters dictionary, performance metrics (RMSE, R^2, MAE for train and test sets), feature list (ordered list of 120+ features), and is_active boolean flag.

Version control ensures reproducibility: every training run creates a new version (v1, v2, v3...), allowing rollback if a new model performs poorly. The is_active flag marks exactly one model as production-ready; when a superior model is trained, the flag migrates atomically using MongoDB transactions.

Benefits of this approach include: centralized model storage eliminating file system dependencies, metadata enables performance tracking over time, consistent feature lists prevent train-serve skew, and cloud hosting enables distributed prediction serving. As of February 2026, the registry contains 8 model versions spanning development iterations, with XGBoost v8 currently active (trained January 15, 2026, RMSE=5.342).

3.14 Automation with GitHub Actions

GitHub Actions provides the automation backbone through two scheduled workflows.

Hourly Data Fetch runs every hour via cron schedule '0 * * * *', executing on Ubuntu runners. The workflow performs environment setup (Python 3.10, pip dependencies), Git checkout of latest code, execution of src/pipelines/hourly_fetch.py which fetches from both APIs, calculates EPA AQI (critical custom implementation), engineers features, and upserts to MongoDB. Secrets (MONGODB_URI, OPENWEATHER_API_KEY) are securely injected from GitHub repository settings. Error handling includes retry logic (3 attempts with exponential backoff) and Slack notifications on persistent failures.

Daily Model Retrain executes at 2 AM UTC via cron '0 2 * * *' when server load is minimal. This workflow loads the past 90 days of data, runs the complete feature engineering pipeline, trains XGBoost/LightGBM/Random Forest in parallel, evaluates and selects best model, updates model registry with new version if performance improved by >1%, and generates performance report logged to workflow artifacts. Both workflows support manual triggering (workflow_dispatch) for debugging.

Benefits achieved: zero manual intervention for 24 months of operation, system adapts to evolving pollution patterns through daily retraining, GitHub Actions free tier provides 2,000 compute minutes monthly (sufficient with ~5 minutes per day), and version-controlled infrastructure-as-code ensures reproducibility.

3.15 Frontend Dashboard

The Streamlit dashboard serves as the public interface, deployed on Streamlit Cloud with automatic deployment on Git pushes to the main branch.

Architecture flow: When a user accesses the URL, Streamlit executes streamlit/app.py which begins by calling setup_environment() to inject secrets into environment variables, then dynamically imports project modules (MongoDBHandler, AQIPredictor, ModelRegistry) after environment is configured (critical for secret access). The dashboard initializes MongoDB connection using @st.cache_resource decorator for singleton pattern, loads the active model from registry, and queries the feature store for current and historical data.

Key UI components include: hero section with animated gradient header and location subtitle, current AQI card displaying real-time value with color-coded category (Good/Moderate/Unhealthy) and emoji indicator, gauge chart visualizing AQI on 0-500 scale with colored zones, 3-day forecast rendered as glassmorphism cards showing predicted AQI with health recommendations, time series plot of historical trends (customizable 1-30 days), pollutant breakdown bar chart identifying major contributors, statistical summary table with mean/max/min values, and sidebar with model performance metrics (RMSE, R^2, model version).

Technical implementation uses Plotly for interactive charts, custom CSS for styling (600+ lines of gradient animations, transitions, and responsive design), session state for caching expensive queries, and error handling with graceful degradation when data unavailable.

4. Challenges & Solutions

4.1 AQI Scale Mismatch (Critical Issue)

Problem Discovery: After initial deployment, users reported confusion: the dashboard displayed AQI values of 2-3 instead of expected ranges like 80-120. This made the system appear broken as common knowledge associates low numbers with good air quality, but the scale was inconsistent with international standards. Investigation revealed that OpenWeather Air Pollution API returns European Air Quality Index (EAQI) on a **1-5 scale** (1=Good, 5=Very Poor), while the internationally recognized US EPA standard uses a **0-500 scale**. This fundamental incompatibility meant all historical data needed recalculation.

Solution Implementation: Developed a custom EPA AQI calculator following official EPA methodology. The algorithm uses **breakpoint tables** for each of six pollutants, where each pollutant's concentration maps to a sub-index via linear interpolation between breakpoints. For example, PM2.5 concentration of 35.5 µg/m^3 falls in the 35.5-55.4 range (breakpoint for Moderate), yielding sub-index of 76. The overall AQI equals the maximum sub-index. Implemented the calculator in src/utils/aqi_calculator.py with comprehensive breakpoint tables and unit tests.

Impact: Re-fetched all 4,215 historical records with corrected AQI calculations, retrained all models with accurate targets, and predictions now align with EPA standards and user expectations. This challenge reinforced the importance of verifying API documentation assumptions.

4.2 MongoDB Connection Timeouts

Problem: Streamlit Cloud deployment experienced intermittent ServerSelectionTimeoutError where MongoDB connections failed during high traffic.

Solution: Implemented connection pooling with maxPoolSize=50, reduced serverSelectionTimeoutMS from 30000ms to 5000ms for faster failure detection, and applied @st.cache_resource decorator to MongoDB client initialization, ensuring singleton pattern reuses connections across app reruns. Added retry logic with exponential backoff (3 attempts, 2^n second delay).

4.3 Secrets Management Across Environments

Problem: Need MongoDB URI and API keys in three environments (local dev, GitHub Actions, Streamlit Cloud), but cannot commit to Git.

Solution: Created unified configuration system in src/config.py: local development uses .env file (gitignored) loaded by python-dotenv, GitHub Actions uses repository secrets injected as environment variables, Streamlit Cloud uses built-in secrets.toml interface accessible via st.secrets. The config module detects runtime environment and sources secrets accordingly, providing clean abstraction for application code.

4.4 Feature Engineering Performance

Problem: Computing 120+ features for each record during real-time dashboard queries caused 10-15 second latency, degrading user experience.

Solution: Shifted feature engineering from query-time to ingestion-time. The hourly fetch script now computes all features immediately after collecting raw data and stores the complete feature set in MongoDB. Dashboard queries retrieve pre-computed features, reducing latency to <2 seconds. Tradeoff: increased storage footprint (each record ~8KB instead of 2KB) but well within MongoDB free tier limits. This architectural decision prioritizes user experience over storage efficiency.

5. Conclusion & Future Work

This project successfully developed and deployed an automated, end-to-end air quality prediction system that addresses a critical public health need in Hyderabad, Sindh. Over 24 months of continuous operation, the system has collected 4,215 hourly records, trained 8 model iterations, and provided reliable 3-day AQI forecasts with 97.2% accuracy (R^2=0.972). The production dashboard serves as a free public resource, demonstrating that sophisticated environmental monitoring systems can be built and maintained with modern cloud technologies and zero operational costs. Key achievements include solving the EPA AQI scale conversion challenge through custom algorithm development, implementing robust automation through GitHub Actions that operates without supervision, achieving high prediction accuracy using ensemble ML and comprehensive feature engineering, and maintaining 99%+ system uptime across free-tier cloud services.

Future Enhancements:

1. **Geographic expansion**: Add 5 more Pakistani cities (Karachi, Lahore, Islamabad, Peshawar, Quetta) to create nationwide monitoring network
2. **Alert system**: Implement email/SMS notifications when AQI exceeds unhealthy thresholds (>150) using Twilio or SendGrid APIs
3. **Extended forecasts**: Increase prediction horizon from 3 days to 7 days using weather forecast APIs
4. **Mobile optimization**: Enhance responsive design and create Progressive Web App (PWA) for offline functionality
5. **Historical downloads**: Enable CSV export of historical data for researchers and policymakers
6. **API development**: Build FastAPI backend to expose predictions via REST API for third-party integrations

This project demonstrates the viability of individual developers creating impact through practical AI applications that serve real community needs.