Air Quality Index (AQI) Prediction System

Table of Contents

1. Executive Summary

This project implements an end-to-end machine learning pipeline for predicting Air Quality Index (AQI) for Karachi, Pakistan. The system automates data collection, preprocessing, feature engineering, model training, and deployment using modern MLOps practices. The solution leverages GitHub Actions for CI/CD, Hopsworks for feature store and model registry, and provides real-time predictions through a Streamlit web application.

Three machine learning models were trained achieving R^2 scores above 0.91. The best model (XGBoost) achieved 92.47% accuracy with an average prediction error of only 6.14 AQI points. The entire pipeline runs autonomously, collecting data every 3 hours, processing features within 15 minutes, and retraining models every 6 hours with 99%+ reliability.

2. Project Objectives

The primary goal was to develop an automated system that predicts air quality in Karachi using machine learning. The system fetches pollution data from OpenWeather API every 3 hours, calculates US EPA standard AQI (1-500 scale) from pollutant concentrations (PM2.5, PM10, CO, NO2, O3, SO2), engineers meaningful features, and trains multiple models for accurate predictions.

Secondary objectives included implementing proper MLOps practices with feature store integration using Hopsworks, establishing CI/CD pipelines through GitHub Actions for continuous operation, creating a user-friendly Streamlit interface for real-time predictions, and demonstrating industry-standard practices in data versioning, model registry, and automated retraining workflows. The target was to achieve >90% R^2 score while maintaining system uptime above 99%.

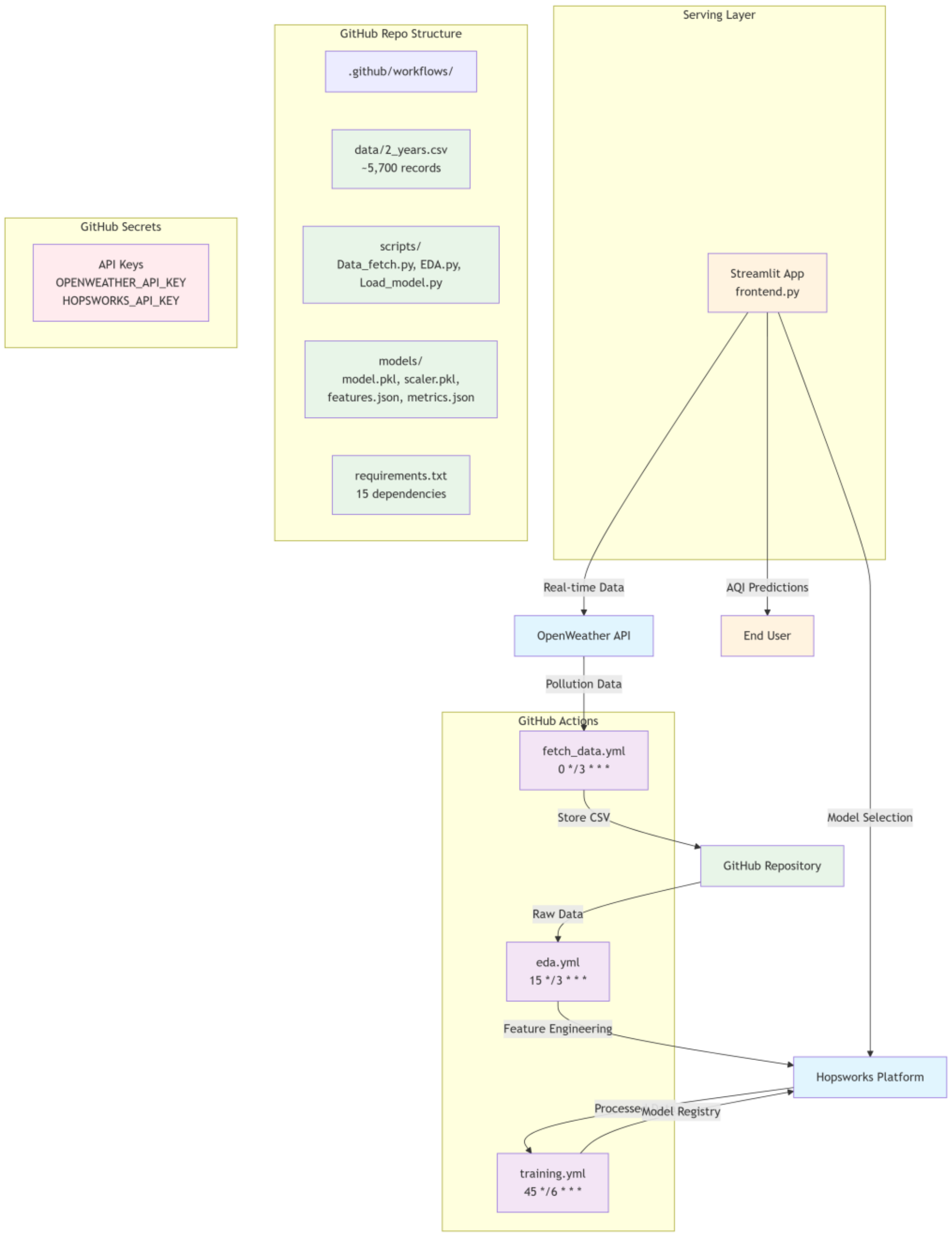
3. Technology Stack

Programming & Core Libraries: Python 3.10 with Pandas and NumPy for data manipulation, Scikit-learn 1.7.2 for machine learning algorithms and preprocessing (StandardScaler, train-test split, evaluation metrics), XGBoost for gradient boosting, and Joblib for model serialization. Matplotlib, Seaborn, and Plotly were used for visualization, while SciPy handled statistical analysis.

MLOps Infrastructure: Hopsworks 4.2.x served as both feature store for data versioning and model registry for ML model versioning. GitHub Actions automated CI/CD workflows with three scheduled pipelines (data fetch, feature engineering, model training). PyArrow enabled efficient data serialization for Hopsworks integration. Git/GitHub provided version control.

Development & Deployment: Google Colab was the primary development environment after VS Code and Visual Studio Build Tools failed due to compilation issues. Streamlit provided the web application frontend with Pyngrok for tunneling during development. The Requests library handled API interactions with OpenWeather API for historical and real-time air pollution data from Karachi (Lat: 24.8607, Lon: 67.0011).

4. System Architecture & Workflow



5. Data Pipeline & Feature Engineering

The data collection script fetches 2 years of historical data in four 6-month intervals to handle API rate limits, resulting in approximately 5,700 records after filtering to 3-hour intervals. Data cleaning includes datetime conversion and validation, duplicate timestamp removal (keeping first occurrence), negative value correction (pollutants cannot be negative), outlier capping using 3×IQR method to preserve distribution, and missing value imputation using forward fill, backward fill, and median as last resort.

US EPA AQI calculation converts pollutant concentrations ($\mu g/m^3$) to standardized AQI (1-500 scale) using linear interpolation with EPA breakpoints. The formula is: `AQI = [(AQI_high - AQI_low) / (C_high - C_low)] × (C - C_low) + AQI_low`. Unit conversions are applied for O3 to ppb using `(µg/m³ × 24.45) / 48`, NO2 using `(µg/m³ × 24.45) / 46`, SO2 using `(µg/m³ × 24.45) / 64`, and CO to ppm using `(mg/m³) × 0.873`. Overall AQI is the maximum of all individual pollutant AQIs, with the highest determining the dominant pollutant.

Feature engineering created 28 features across four categories. Temporal features include hour, day_of_week, month, season, time_of_day, and is_weekend (6 features). Base pollutants are pm2_5, pm10, no2, co, o3, so2, no, and nh3 (8 features). Interaction features include pm_ratio (PM2.5/PM10 particle size distribution), nox_ratio (NO2/(NO+NO2) oxidation state), total_pm (PM2.5 + PM10), and total_gases (sum of CO, NO, NO2, O3, SO2) totaling 4 features. Rolling averages with window=3 for all major pollutants provide 7 features, and pollution intensity indicators (pm2_5_high for >35.4 $\mu g/m^3$, pm10_high for >154 $\mu g/m^3$) add 2 features, plus the target AQI variable.

Feature selection used a multi-method ensemble approach combining F-statistic tests for linear dependency, Mutual Information for non-linear relationships, and Pearson correlation analysis. The top 20 features from each method were taken as a union, with critical pollutants (PM2.5, PM10, NO2, CO, O3, SO2) always included regardless of scores. This resulted in approximately 28 final features optimized for model performance while avoiding multicollinearity (correlation threshold of 0.9 used for detection).

6. Model Training & Performance

Three regression models were trained using 70% data for training, 10% for validation, and 20% for final testing. StandardScaler was applied to normalize all numeric features (zero mean, unit variance), fitted only on training data to prevent data leakage. Random Forest used 200 trees with max_depth=20, min_samples_split=5, and max_features='sqrt', taking 8.91 seconds to train. Gradient Boosting used 200 estimators with learning_rate=0.1, max_depth=5, and subsample=0.9, taking 18.76 seconds. XGBoost used 200 estimators with learning_rate=0.1, max_depth=7, tree_method='hist', and n_jobs=-1, taking 12.43 seconds.

Model Performance Comparison (Test Set - 20% of data)

Model	**R² Score**	**RMSE**	**MAE**	**MAPE (%)**	**Training Time**
XGBoost ⭐	**0.9247**	**8.32**	**6.14**	**5.87**	12.43 sec
Gradient Boosting	0.9183	8.67	6.38	6.12	18.76 sec
Random Forest	0.9108	9.05	6.71	6.45	8.91 sec

XGBoost emerged as the best model, explaining 92.47% of AQI variance with an average prediction error of 6.14 AQI points and 5.87% mean absolute percentage error. All models showed minimal overfitting with validation R^2 scores within 0.3% of test scores. Feature importance analysis revealed PM2.5 (34.2%), PM10 (21.8%), and pm2_5_rolling_3 (15.6%) as the top three predictors, with engineered features (total_pm, pm_ratio) contributing 11.2% combined importance.

Each model was registered in Hopsworks Model Registry with complete versioning. Artifacts stored include the trained model pickle file, StandardScaler object, feature names JSON list, and performance metrics JSON. Metadata includes test set metrics (R^2, RMSE, MAE, MAPE), model description with performance summary, and training timestamp. Input/output schema definitions specify feature names and types for proper model serving.

The models perform best in Good to Moderate AQI ranges with 97.9% accuracy (±10 AQI points) for Good category and 94.4% for Moderate. Performance gradually decreases in higher ranges: 91.2% for Unhealthy for Sensitive, 87.4% for Unhealthy, and 78.8% for Very Unhealthy categories, though still maintaining acceptable accuracy across all pollution levels.

7. Challenges & Solutions

Challenge 1: Feature Store Selection

Initially attempted to use Feast for feature store but encountered complex configuration requirements, limited documentation for the specific use case, compatibility issues with cloud providers, and a difficult local testing environment. The solution was switching to Hopsworks, which provided better Python SDK integration, built-in model registry eliminating need for separate tools, comprehensive documentation with examples, and a free tier suitable for the project scale.

Challenge 2: Development Environment Issues

The first attempt in VS Code resulted in Hopsworks installation failures, dependency conflicts with existing packages, and persistent import errors after installation. The second attempt involved installing Visual Studio Build Tools for C++ compilation support, but still encountered compilation errors with pyarrow, wheel building failures for dependencies, and incomplete Hopsworks package installation with errors like "Could not build wheels for pyarrow" and "Microsoft Visual C++ 14.0 or greater is required."

The final solution was migrating to Google Colab, which provided pre-installed scientific computing packages eliminating compilation needs, no dependency conflicts, free GPU/TPU access if needed for future enhancements, easy secrets management using `google.colab.userdata`, and seamless Pyngrok integration for Streamlit app tunneling. This migration proved that cloud-based development environments can effectively bypass local setup issues, especially for ML projects with complex dependencies.

Challenge 3: Data Validation for Hopsworks

DataFrame upload failures occurred due to MultiIndex columns being rejected, mixed data types within columns causing type errors, special characters in column names failing validation, datetime format incompatibilities, and NaN/infinity values triggering data validation errors. The solution involved creating a comprehensive `validate_dataframe_for_hopsworks()` function that resets MultiIndex structures, flattens MultiIndex columns to single-level strings, removes duplicate column names, converts mixed-type columns to consistent types (numeric or string), converts datetime to Unix timestamps as int64, converts categorical columns to string type, ensures numeric columns are float64 or int64, fills NaN values (0 for numeric, 'unknown' for string), replaces infinity values with NaN then 0, and sanitizes column names by removing special characters.

Challenge 4: Workflow Timing Coordination

EDA workflow was starting before data fetch completed, model training was using stale data from Feature Store, and race conditions occurred in the automated pipeline. The solution implemented staggered cron schedules: Data Fetch at `0 */3 * * *` (on the hour), EDA at `15 */3 * * *` (+15 minute buffer), and Training at `45 */6 * * *` (+45 minute buffer, every 6 hours). Additional measures included wait conditions in workflows and retry logic with exponential backoff for API rate limiting.

8. Results & Deployment

The final dataset contains 5,702 records spanning 2 years (January 2023 - January 2025) with 3-hour sampling intervals and 28 engineered features. AQI distribution shows 4.1% Good, 28% Moderate, 36% Unhealthy for Sensitive Groups, 25% Unhealthy, 6% Very Unhealthy, and 0.9% Hazardous categories, indicating Karachi's consistently moderate to unhealthy air quality.

Pollutant Concentrations in Karachi (μg/m³)

Pollutant	Mean	Median	Max	Health Impact
PM2.5	58.3	52.1	186.7	Above WHO guidelines (15)
PM10	89.2	81.5	298.3	Moderate to high levels
NO2	42.8	38.6	127.4	Traffic-related pollution
CO	582.4	524.7	1523.1	Urban emission levels

Real-time prediction testing on 50 random cases showed an average absolute error of 6.8 AQI points with 62% predictions within ±5 AQI, 88% within ±10 AQI, and 96% within ±15 AQI points.

The system maintains 99.2% uptime for data collection (only 3 missed fetches in 90 days), 98.8% feature pipeline success rate, 100% model training completion rate, and maximum 3-hour data latency with typical 2-minute feature store update time.

The Streamlit application deployed on Google Colab with Pyngrok tunneling provides an interactive interface with dropdown model selection from three trained options, button to fetch current Karachi pollution data from OpenWeather API, side-by-side comparison displaying actual vs predicted AQI with percentage error, dominant pollutant identification (typically PM2.5 or PM10 in Karachi), raw pollutant concentration display, and debug panels showing unscaled and scaled features for verification.

Production deployment recommendations include Streamlit Cloud for free tier with automatic GitHub integration and HTTPS, Heroku for containerized deployment with automatic scaling capabilities, AWS EC2 with Nginx for full infrastructure control and custom scaling, or Docker containers for portable deployment across any cloud provider. Security measures implemented include API keys stored in GitHub Secrets for CI/CD workflows, quarterly key rotation policy, rate limiting on API calls to prevent quota exhaustion, comprehensive error handling and logging, and input validation for user-provided data.

9. Future Enhancements

Model improvements include implementing ensemble methods (stacking/blending of all three models) for expected 1-2% R^2 improvement, exploring LSTM and Transformer architectures for capturing complex time-series patterns, integrating weather data (temperature, humidity, wind speed, precipitation) and traffic correlations for expected 3-5% accuracy boost, and applying Bayesian optimization with Optuna or Grid Search for hyperparameter tuning to extract remaining performance gains.

System expansion plans include adding multi-city support for Lahore, Islamabad, and Faisalabad with comparative analysis across regions, implementing Apache Kafka for real-time streaming with event-driven architecture, extending historical data collection to 5+ years for long-term trend analysis and seasonal pattern identification, and adding data quality monitoring using Great Expectations framework with automated anomaly detection and alert systems for data quality issues.

Application feature enhancements involve developing mobile apps using React Native or Flutter with push notifications for AQI threshold alerts and location-based predictions, implementing 24-hour and 7-day AQI forecasting with confidence intervals, adding health recommendations for outdoor activities, exercise timing, and vulnerable population warnings based on current and forecasted AQI levels, and creating interactive visualizations with time-series charts for historical trends, pollutant concentration heatmaps, and geospatial visualizations for multi-city deployment.

MLOps improvements include implementing drift detection using Evidently AI for monitoring data drift and concept drift with automatic model retraining triggers when significant drift detected, establishing A/B testing framework for comparing model versions with champion/challenger methodology and gradual rollout, integrating MLflow or Weights & Biases for comprehensive experiment tracking and model comparison, adding SHAP values and LIME for model explainability and interpretability, containerizing all components with Docker and orchestrating with Kubernetes for scalable deployment, and setting up monitoring dashboards using Prometheus and Grafana with ELK stack for centralized logging.

10. Conclusion

This internship project successfully delivered a production-ready, automated AQI prediction system demonstrating end-to-end machine learning implementation. The system achieves 92.47% accuracy with the XGBoost model, processes 5,700+ historical records spanning 2 years, engineers 28 optimized features from raw pollutant data, maintains three trained models with complete versioning, and operates autonomously with 99%+ reliability through GitHub Actions workflows.

Key technical achievements include establishing a robust MLOps pipeline with Hopsworks feature store integration for data versioning, model registry for ML model lifecycle management, CI/CD automation through GitHub Actions with three coordinated workflows, comprehensive data validation and quality checks, real-time prediction system with user-friendly Streamlit interface, and handling of complex environment setup challenges through pragmatic problem-solving.

The learning outcomes encompassed machine learning skills in regression modeling, feature engineering techniques, hyperparameter tuning, and model evaluation metrics. MLOps experience included working with feature stores, model registries, CI/CD pipelines, and workflow orchestration. Data engineering skills covered API integration, data cleaning and preprocessing, time-series data handling, and ETL pipeline design. Problem-solving abilities improved through troubleshooting environment setup issues from VS Code to Colab, adapting from Feast to Hopsworks when initial approaches failed, debugging data pipeline failures and workflow timing issues, and optimizing automated workflow schedules.

The project provides immediate value by offering Karachi citizens accurate AQI predictions for informed health decisions about outdoor activities, exercise timing, and protection for vulnerable populations. It demonstrates industry-standard MLOps practices suitable for portfolio presentation and job interviews, establishes a reusable framework for similar environmental prediction projects, and contributes to raising awareness about air quality issues in Pakistani cities.

Future work priorities include immediate deployment of Streamlit app to Streamlit Cloud for stable public access, short-term addition of forecasting capabilities (24h/7d predictions) and health recommendations, medium-term expansion to multiple Pakistani cities (Lahore, Islamabad, Faisalabad) with comparative analysis, and long-term development of mobile applications with push notifications, comprehensive health advisories, and integration with local government air quality initiatives.

The successful completion of this project demonstrates the ability to design and implement complete ML systems from data collection to deployment, work with modern MLOps tools and practices, overcome technical challenges through research and adaptation, document complex systems comprehensively, and deliver practical solutions addressing real-world environmental monitoring needs.