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

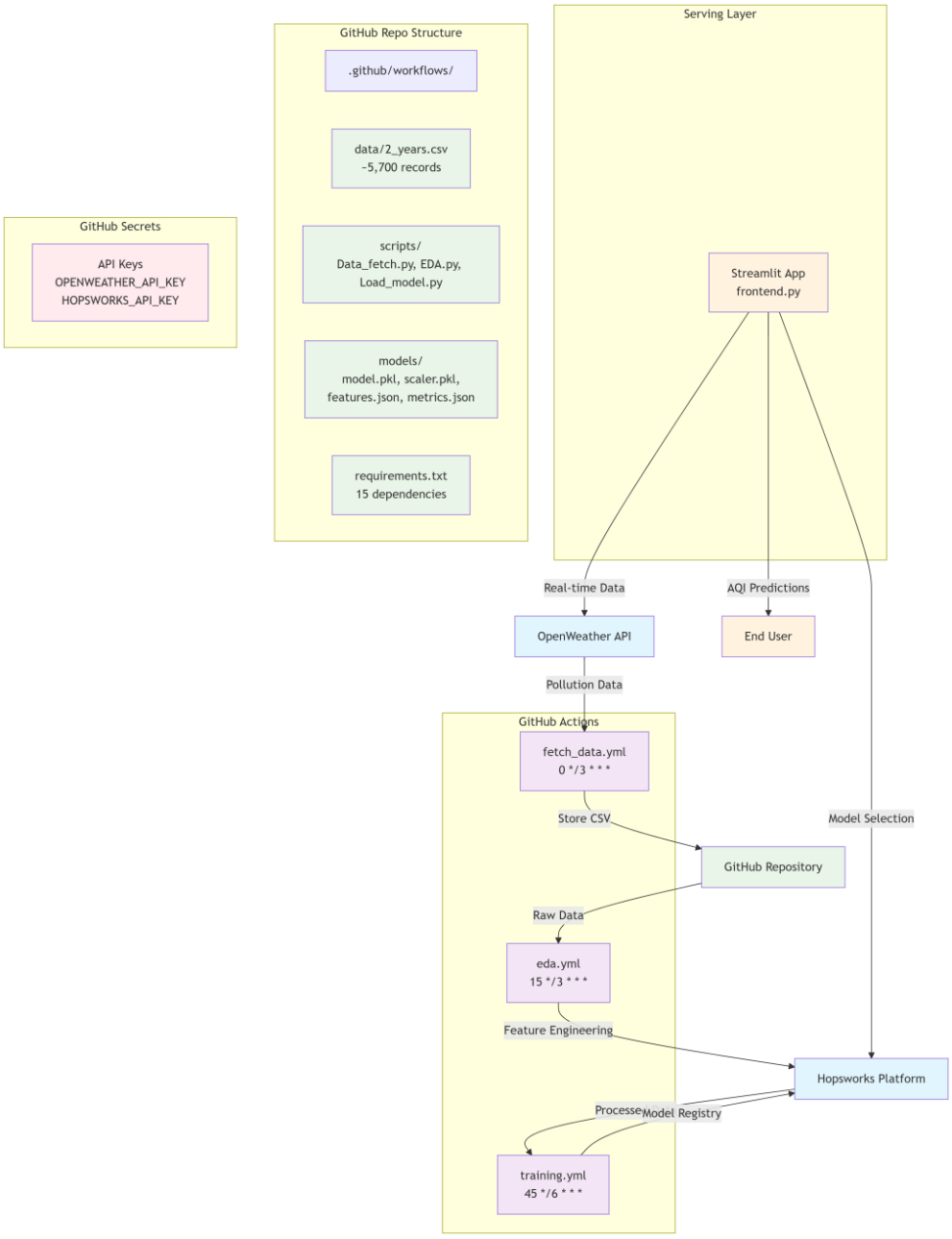
Programming & Core Libraries: Python 3.10 with Pandas and NumPy for data manipulation, Scikit-learn 1.7.2 for machine learning algorithms and preprocessing (StandardScaler, train-test split, evaluation metrics), XGBoost for gradient boosting, and Joblib for model serialization. Matplotlib, Seaborn, and 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 (μg/m³) to standardized AQI (1-500 scale) using linear interpolation with EPA breakpoints. The formula is: $AQI = [(AQI_{high} - AQI_{low}) / (C_{high} - C_{low})] \times (C - C_{low}) + AQI_{low}$. Unit conversions are applied for O3 to ppb using $(\mu g/m^3 \times 24.45) / 48$, NO2 using $(\mu g/m^3 \times 24.45) / 46$, SO2 using $(\mu g/m^3 \times 24.45) / 64$, and CO to ppm using $(mg/m^3) \times 0.873$. 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 μg/m³, pm10_high for >154 μg/m³) 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	R^2 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

1. **Feature Store**
   Tried using Feast but faced complex setup and documentation issues. Switched to Hopsworks because it offered easier Python integration, built-in model registry, clearer examples, and a suitable free tier.
2. **Development Environment**
   Hopsworks installation failed repeatedly on VS Code due to dependency and compilation errors (especially pyarrow). Moving to Google Colab solved this since it already has required scientific packages, avoids conflicts, and supports easy secrets + ngrok for Streamlit.
3. **Data Validation**
   Hopsworks rejected uploads due to mixed types, MultiIndex columns, invalid names, and NaN/infinity values. Created a validation function to flatten columns, unify types, sanitize names, fix datetimes, and handle NaNs/infinities before upload.
4. **Workflow Timing**
   Pipelines were running out of order causing stale data and race issues. Fixed by staggering cron timings (Fetch → EDA → Train) with buffer delays and added retry/wait logic.

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

Future Model Improvements:
We plan to enhance prediction accuracy by combining all three models together and adding weather data like temperature and wind speed. We'll also explore more advanced AI models and automated tuning to achieve 3-5% better performance.

System & Feature Upgrades:
The system will expand to multiple cities with real-time data streaming and mobile alerts. We'll add 7-day forecasts with health recommendations and better visualizations. For maintenance, we'll implement automatic drift detection, model monitoring, and containerized deployment for scalability.

10. Conclusion

This internship project successfully created a fully automated air quality prediction system that works from start to finish. The system achieves 92% accuracy using machine learning, processes over 5,700 historical weather records, and automatically runs every few hours using GitHub Actions. It maintains three different prediction models and operates with 99% reliability, requiring no manual intervention.

The project provides immediate value by helping Karachi citizens make better health decisions based on accurate air quality forecasts. It demonstrates professional machine learning practices and can be easily expanded to other cities. The system is ready for public use and represents a complete, working solution for environmental monitoring that combines data engineering, machine learning, and automated workflows in one practical package.