Project Report
Pearls AQI Predictor for Karachi

This project builds an end-to-end pipeline to predict air quality for Karachi using one year of hourly weather and pollution data. I chose Karachi because I live here. The work covers data collection, cleaning, exploratory data analysis, feature engineering, model training and evaluation, model registration, and simple plans to run the pipeline on a schedule and show results in a Streamlit dashboard.

Data collection and preparation

I collected hourly weather and air-quality readings for Karachi from Open-Meteo's archive and air-quality API. The script loops month by month from 1 January 2024 to 31 December 2024 to avoid timeouts, merges weather and pollutant data on timestamp, and converts the time column to a proper datetime. After merging, I renamed columns, added the fixed Karachi coordinates, and saved the combined data. The final backfill produced 8,784 hourly rows, which equals 365 days by 24 hours.

I then created a clean dataset with consistent names, making it ready to compute features. This same dataset was also uploaded to Hopsworks as a Feature Group named air_quality_features so that features can be read later for model training and for any downstream use.

Feature engineering

From the raw data I added simple time features such as hour of day, day of week, month, and a weekend flag. I also created features that help a model understand short-term changes, like one-hour PM2.5 differences and rolling averages over 6 hours and 24 hours. I added lag features for PM2.5 at 1 hour, 3 hours, and 24 hours to capture short-term history. I derived interactions like temperature times humidity, and simple wind features including a small "wind inverse" term, and sine and cosine transforms of wind direction. The engineered feature table was saved back to disk for training and also kept in the Feature Store for reuse.

Alongside features, I computed an Air Quality Index (AQI) value for each hour using US-EPA breakpoints for PM2.5 and PM10 and chose the worse of the two. I also mapped AQI to categories such as Good, Moderate, and Unhealthy. Over the year, the average AQI was about 89, and most hours were Moderate, with a smaller share in Unhealthy for Sensitive Groups and Unhealthy. This helped me understand the target behavior before training.

Exploratory data analysis

I checked dataset size and summary statistics to confirm ranges look reasonable. I plotted the distribution of PM2.5 and a time plot of PM2.5 over the year for Karachi. This showed seasonality and daily swings that match expectations for an urban coastal city with traffic patterns and changing weather. These quick checks confirmed that the data merged correctly and that there were no obvious gaps after the monthly fetch-and-merge process.

Training and evaluation

For predicting PM2.5 directly from the feature group, I trained a Random Forest Regressor with a standard train-test split. On the held-out test set, the model reached a Mean Absolute Error of about 2.33 µg/m³, a Root Mean Squared Error of about 3.41 µg/m³, and an R^2 score of about 0.928. I saved this model as `pm25_rf_model.pkl`. These numbers show the model explains most of the variation in PM2.5 on test data while keeping average errors low in absolute terms.

I also trained a Ridge Regression model as a simple linear baseline. It performed worse than the Random Forest, with a higher error and a lower R^2 score. This confirmed that non-linear models capture the short-term dynamics better for this task. Both models were registered in the Hopsworks Model Registry with their metrics and descriptions, but the Random Forest is the preferred model for deployment.

Next, I trained a second set of models to predict AQI directly from the engineered features that include lags, rolling statistics, and weather interactions. I tried Random Forest and Gradient Boosting. The code prepares the data by dropping rows without AQI, selecting the feature columns that exist, splitting into train and test, and then fitting both models. This experiment helps compare tree ensembles for AQI, similar to PM2.5.

Pipelines and scheduling

To make this work repeatable, I describe the steps as two pipelines. First is the feature pipeline. It fetches raw weather and pollutant data from external APIs, builds time features and derived features such as rolling means and one-hour changes, and writes these processed features into the Feature Store. I also backfilled one full year to create a strong training set. Second is the training pipeline. It reads historical features and targets from the Feature Store, trains several models such as Random Forest and Ridge Regression, evaluates them using RMSE, MAE, and R^2, and then stores the best model in the Model Registry. In a simple automation plan, the feature pipeline can run each hour to keep data fresh, and the training pipeline can run daily to incorporate the most recent data and update the model if it improves.GitHub Actions can orchestrate these runs.

Application and dashboard

For sharing results, I keep it simple. The web app layer loads the saved model and can also read features from CSV when needed. It computes AQI predictions for the next three days by applying the trained model to recent feature values or forecasted drivers if available. The focus is to make this accessible for a Streamlit dashboard so that anyone can explore current conditions and short-term forecasts with clear charts and labels.

Conclusion

This project builds a clear path from hourly data collection to useful AQI predictions for Karachi. It gathers one year of data, engineers meaningful features, validates patterns with simple EDA, trains and evaluates models with strong scores, and registers the best model. It also lays out simple scheduling and a Streamlit view so the system can update on its own and present predictions in an easy interface.