Pearls AQI Predictor

Project Report

A Serverless, Feature-Store-Driven, Multi-Model System

Author: Abdul Mueed Khan

Contents

1. Project Overview

This project delivers a 72-hour (3 days) AQI forecast for Multan using a 100% serverless architecture centered on Hopsworks Feature Store and GitHub Actions automation. Multiple forecasting methods—ranging from traditional machine learning to deep learning—were developed and compared, ultimately converging on a **banded direct multi-horizon LSTM** model. The model, deployed via a FastAPI application for low-latency inference, predicts $PM_{2.5}$ and PM10 concentrations, which are used to compute the EPA AQI. The entire pipeline is automated, handling data ingestion from OpenWeather, historical feature engineering, model training, and deployment.

Key outcomes:

- **Direct LSTM (banded)** with horizon-weighted loss and target-horizon exogenous features provides strong short-term accuracy (1–12h) and reasonable mid/long performance (12–72h).
- **Feature Store-centric** design: robust Feature Groups and Feature Views ensure consistent train/serve schemas and online access.
- **Automations**: hourly forecast collector; hourly feature updates; daily retraining; artifact registration to Model Registry.
- **App**: FastAPI service serving a modern dashboard with AQI shading, and summary statistics.

2. System Architecture

2.1. Overview

- **Data sources**:
  - **OpenWeather** Current Pollution, Current Weather, Pollution Forecast, Weather Forecast APIs.
- **Feature Store (Hopsworks)**:
  - **Historic Feature Group** with engineered features for training and encoder windows.
  - **Forecast Feature Group** with 72 future hours for decoder exogenous at $t + h$.
  - **Feature Views** serving schemas.
- **Models**:
  - Direct LSTM multi-horizon (banded short/midlong)
  - Baselines: LightGBM (direct & iterated), ExtraTrees, RandomForest
  - Seq2Seq LSTM variant
- **App**: FastAPI loads banded LSTM models, fetches Feature Views, predicts 72 hours, computes AQI.
- **Automation**: GitHub Actions workflows for hourly pipelines and daily retraining (manual/external trigger via repository dispatch).

2.2 Key Components and Scripts

2.2.1 Data Pipeline and Infrustructure

- **datacollector.py:** serves as the central data ingestion system, fetching both air quality and weather data from OpenWeather with robust error handling with exponential backoff retry logic, real-time and historical data retrieval, and automatic US EPA AQI computation from raw pollutant concentrations. It is designed for production reliability through comprehensive logging and data validation. While the pipeline initially incorporated the IQAir API for AQI values, this was later deprecated in favor of calculated AQI to maintain consistency with OpenWeather data. Although the script still includes a redundant IQAir API call, the returned data is discarded and is planned to be removed in future iterations.
- **config.py:** consolidates all project-wide configuration settings, including API credentials, geographic coordinates for Multan, feature engineering parameters, AQI thresholds, and Hopsworks integration details. It adopts an environment-based configuration approach to ensure secure credential handling while maintaining flexibility across development and deployment environments.
- **hopsworksintegration.py:** this script provides a lightweight wrapper around the Hopsworks SDK for uploading engineered features to the Hopsworks Feature Store. It defines a **HopsworksUploader** class that manages basic connections, handles feature group creation with primary key (*time_str*) and event time (*time*) settings, and supports data insertion with retry logic for network operations. Designed for simplicity, it streamlines the upload process without additional complexity beyond core Hopsworks functionality.
- **featureengineering.py:** implements the complete feature engineering pipeline, incorporating cyclical time encodings, lagged variables, rolling statistics, change rates, and feature interactions. The initial design produced 127 features, largely guided by domain intuition and established practices in air pollution modeling; however, following a detailed exploratory data analysis (EDA), many of these features were pruned due to redundancy or weak predictive value, while a smaller set of targeted features was added. The resulting streamlined pipeline provides a refined and balanced feature set that improves model robustness and efficiency without unnecessary complexity.

2.2.2 Feature Engineering and Automation

- **automatedhourlyrun.py:** executes automated hourly feature updates through the feature engineering pipeline. Running on GitHub Actions, it fetches raw data from APIs, applies feature transformations, and pushes the processed features into the Hopsworks Feature group.

- **automatedhourlyrun_updated.py:** an improved version of the base hourly update script, this implementation integrates additional features and refinements identified during exploratory data analysis (EDA). It is executed via GitHub Actions, triggered by an external cron job, and ensures higher data quality, consistency, and model readiness through a more robust feature pipeline.

2.2.3 Model Training

- **lstmdirectmultihorizonv1.py:** a direct multi-output **LSTM architecture** capable of predicting all 72 forecast horizons in a single forward pass, thereby eliminating error accumulation common in autoregressive approaches. The model incorporates horizon-weighted loss functions that place greater emphasis on the critical short horizons (immediate future hours), along with horizon-dependent noise injection to better simulate forecast uncertainty. It is trained using a banded short- and mid-long horizon strategy, ensuring balanced performance across near-term and extended predictions.

While several alternative models were explored, this architecture was ultimately selected for its superior overall performance and ability to deliver the most satisfactory results.

2.3.4 Model Management and Deployment

- **model_registry_utils.py:** utility functions for seamless integration with the Hopsworks Model Registry, supporting automated model versioning, artifact management, and deployment synchronization between the registry and the production datasets directory. It streamlines the management of trained models, ensuring consistent version control and reliable deployment within the ML pipeline.
- **fastapi_app.py:** production-ready FastAPI application that serves real-time AQI forecasts through an interactive dashboard. It supports on-demand inference using cached models, incorporates robust error handling for reliability, and delivers a modern web interface with ECharts-powered visualizations for intuitive user interaction.

3. Data Management & Feature Store

3.1 Historic Feature Group

Hourly engineered features with:

- **PM block:** raw $PM_{2.5}$, pm10; rolling means/max (3/12/24h); change rates (1/6/24h).
- **Weather:** temperature, humidity, pressure, windspeed, wind direction sin/cos.
- **Time:** cyclical hour/day/month/day_of_week sin/cos.
- **Pollutants:** carbonmonoxide, ozone, sulphurdioxide, nh3.
- **Interactions:** PM×weather ($PM_{2.5}$ pressure_interaction etc.).

Primary key: timestr (floored-to-hour ISO-like string);

This feature group stores all processed and engineered features necessary for model training, ensuring a consistent and reliable data source.

Note: Time features were encoded cyclically to preserve their natural periodicity and prevent artificial discontinuities (e.g., between 23:00 and 00:00).

3.2 Forecast Feature Group

The script **automated_forecastcollector.py** pushes 72 rows (step 1..72) with:

- **Weather:** temperature, humidity, pressure, windspeed, wind direction sin/cos.
- **Time at horizon ($t + h$):** hour sin/cos, day sin/cos, month sin/cos, dayofweek sin/cos.
- **Pollutants:** non-PM pollutants, including carbonmonoxide, ozone, sulphurdioxide, nh3.

This feature group contains hourly-updated 72-hour forecasts, providing the exogenous features used in the model's input pipeline.

3.3 Feature Views & Serving

A **Feature View (FV)** is a logical representation of data in the feature store, combining one or more feature groups into a queryable and versioned dataset for training or online serving.

- **historicfv:** encoder windows (72h for short; 96h for mid/long).
- **forecastsfv**: exogenous decoder inputs for $t + 1 \ldots t + 72$.

Re-initializing FV serving to the latest training dataset version is recommended after schema-affecting changes. The app expects serving schemas to expose all required columns.

4. Exploratory Data Analysis (EDA)

Extensive EDA was conducted to understand distributions, seasonality, and relationships among PMs, weather, and co-pollutants.

4.1 Coverage and Distributions

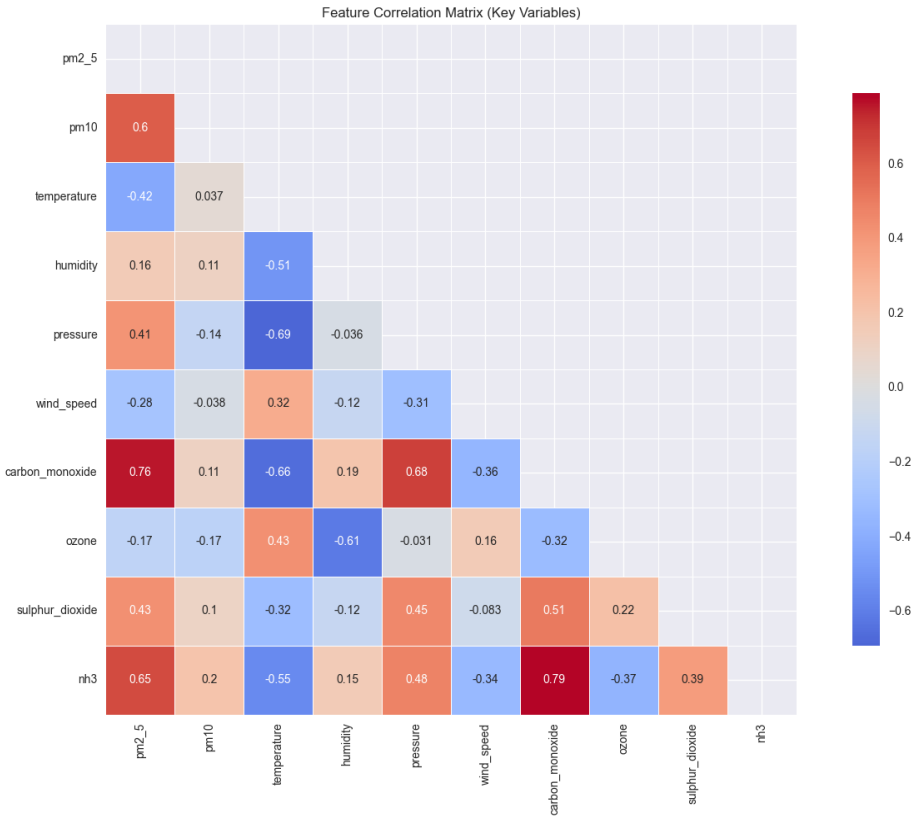
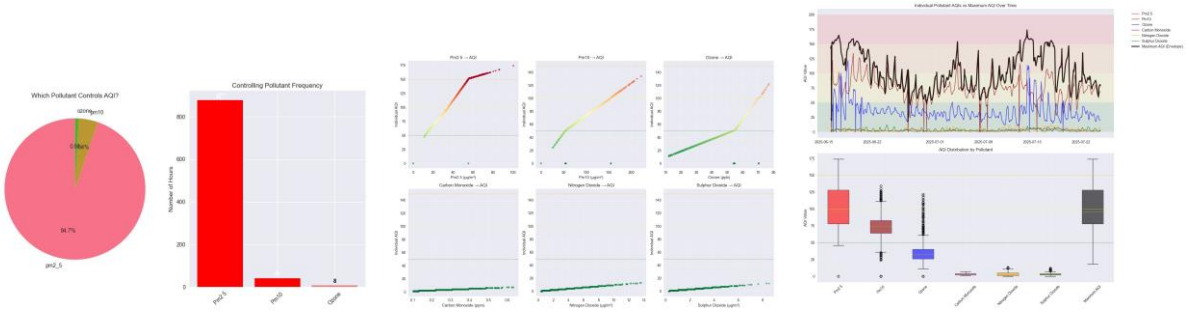


Figure 4.1: EDA: coverage and temporal structure

The correlation matrix in Figure 4.1 indicates that $PM_{2.5}$ exhibits the strongest associations with carbon monoxide, NH_3, sulphur dioxide, and atmospheric pressure. By comparison, PM_{10} demonstrates weaker correlations overall, with humidity, NH_3, and carbon monoxide emerging as the most influential factors. These observations suggest that exogenous features will play a particularly important role in forecasting $PM_{2.5}$ concentrations across different horizons.

4.2 AQI Dominance Analysis



(a) **(b)** **(c)**

Figure 4.2: AQI Dominating Pollutants

Methodology

AQI values for each pollutant were computed using the **U.S. EPA's official breakpoint system**. For each pollutant, its measured concentration was matched to the corresponding EPA-defined concentration range ("breakpoint") and linearly interpolated to obtain the pollutant-specific AQI sub-index. The pollutants included in this analysis were **$PM_{2.5}$, PM_{10}, O_3, CO, NO_2, and SO_2**; any pollutant absent for a given hour was excluded from that calculation.

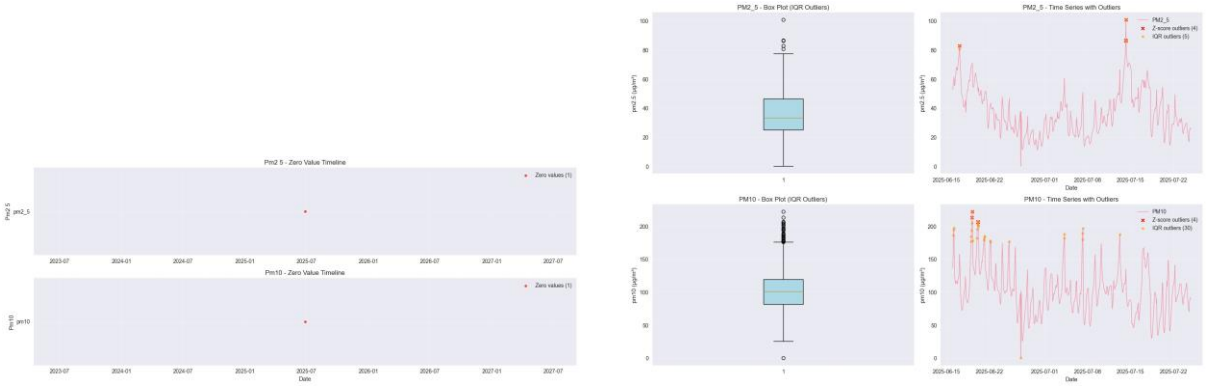
The **overall AQI** for each timestep was determined as the **maximum sub-index** among all available pollutants, with the pollutant contributing that maximum recorded as the **dominant pollutant**. This ensures that the dominance analysis aligns with EPA reporting standards and accurately reflects the primary driver of poor air quality at each point in time.

Results

Figure 4.2(a) shows the distribution of dominant pollutants over the study period. $PM_{2.5}$ emerged as the dominant pollutant for the majority of hours, followed by PM_{10} and Ozone in much smaller proportions. Ozone, NO_2, and SO_2 contributed minimally to dominance events, reflecting both their typically lower concentrations and weaker correlation with the overall AQI. These findings suggest that mitigation strategies should focus primarily on controlling fine particulate matter levels to achieve substantial improvements in AQI. As a result, $PM_{2.5}$ and PM_{10} were considered sufficient as target variables, as they contributed to over 97% of the AQI levels.

Figures 4.2(b) and 4.2(c) further confirm that $PM_{2.5}$ dominates the maximum AQI levels for the vast majority of the time, accounting for 94.8% of instances within 90% of the peak AQI.

4.3 Data Validation



(a) Zero Values **(b)** Outliers Analysis

Figure 4.3: OpenWeather Data Validation

Figure 4.3(a) shows the zero-value timelines for $PM_{2.5}$ and PM_{10} concentrations. Only a single zero reading was observed for each pollutant across the entire dataset, both occurring at the same timestamp in mid-2025. The scarcity of zero readings suggests minimal missing or faulty sensor data for these pollutants, indicating high data completeness and reliability for subsequent modeling.

Outlier analysis (**Figure 4.3(b)**) using both the **z-score** and **IQR** methods revealed notable differences in detection counts. For **$PM_{2.5}$**, four outliers were flagged by the z-score method, compared to five identified via the IQR approach. In the case of **PM_{10}**, both methods detected four outliers by z-score, but the IQR method flagged a substantially higher number (30), likely due to the distribution's skewness and the IQR method's greater sensitivity to values far from the interquartile range.

Given that pollutant concentration data often exhibit skewed, non-normal distributions, the **IQR method** was selected as the primary outlier detection approach for subsequent preprocessing. This ensures robustness against distributional asymmetry, while still allowing cross-checks with z-score results for consistency.

4.4 Distribution Analysis

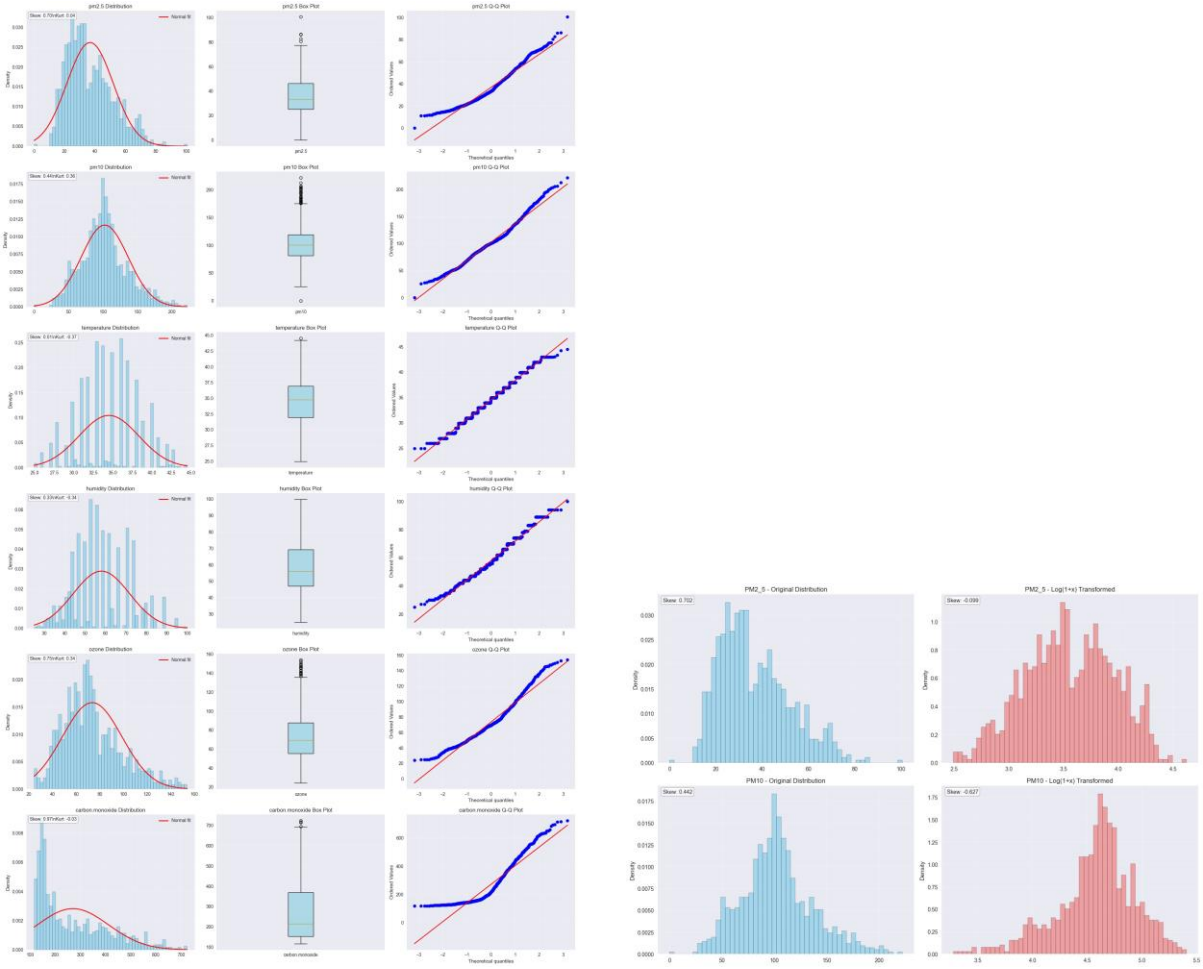


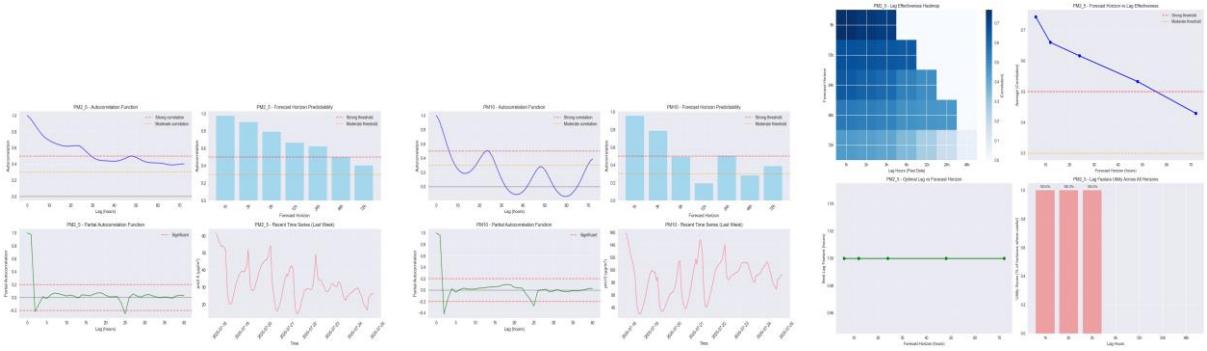
Figure 5.4

Focus: Understanding feature distributions, skewness, and normality for optimal model preprocessing and outlier interpretation.

The distribution statistics indicate that **$PM_{2.5}$** is highly skewed to the right (skewness = 2.398) with a pronounced heavy tail, suggesting the presence of extreme high-concentration events. **PM_{10}** shows moderate right skewness (0.744) and a heavy-tailed nature, though less extreme than $PM_{2.5}$. These results confirm that both pollutants deviate from normality, with $PM_{2.5}$ exhibiting stronger non-Gaussian characteristics. **Figure 5.4(b)** suggests that a logarithmic transformation would effectively reduce the skewness in **$PM_{2.5}$** , improving its normality. In contrast, the distributions of the other pollutants appear more amenable to robust scaling or standard scaling, as their skewness and kurtosis values are less extreme.

In light of these findings, **preprocessing strategies were adapted differently for the models**. For tree-based methods, $PM_{2.5}$ values were log-transformed to reduce skewness, while IQR outliers were removed to avoid unstable splits. For LSTMs, IQR outlier removal was applied across all pollutants to reduce training instability, while $PM_{2.5}$ values were retained in raw form with standard scaling, given that deep networks handle skewness more effectively.

4.5 Forecasting Performance Analysis



(a) Forecast feature sample 1 **(b)** Forecast feature sample 2 **(c)** Lag effectiveness

Figure 4.5.1: Temporaral Autocorrelation Analysis

To evaluate short-term persistence in $PM_{2.5}$ and PM10 concentrations, we analyzed the predictive strength of current PM readings and their temporal lags for forecasting future values. Results indicate that:

- **Current PM values** show strong continuity for forecasts up to **6 hours ahead**, suggesting high short-term persistence in pollution levels.
- **Lagged PM values** (past readings) are also significant predictors:
    - **1-hour lag** is the most reliable, maintaining predictive value for up to **24 hours ahead**, with exceptionally high accuracy in the **first 3 hours**.
    - **2-hour and 3-hour lags** provide moderate predictive capability, useful for shorter horizons.

Recent lags (1–3h) dominate short-term predictability; usefulness transitions toward mixed/longer lags beyond 24h. This supports banded training and horizon-weighted loss.

4.6 Weather-PM Correlation

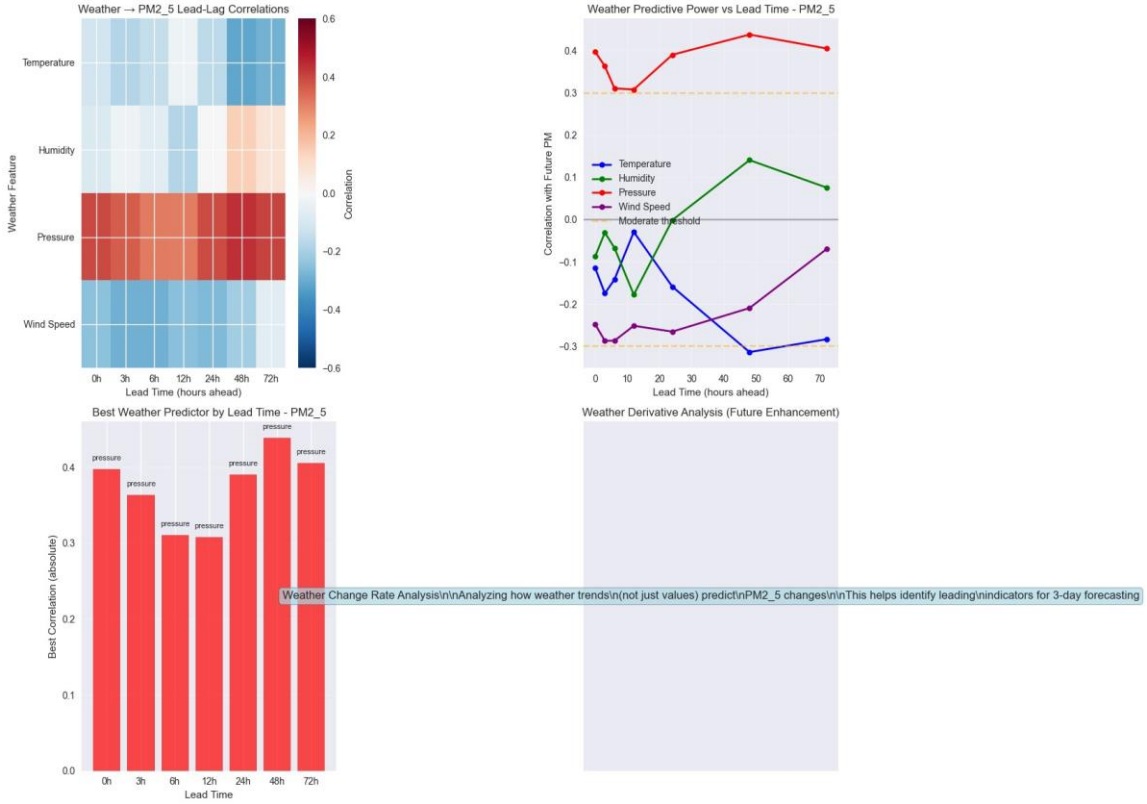


Figure 4.5.2: Weather → $PM_{2.5}$ lead–lag correlations and predictive power by lead time

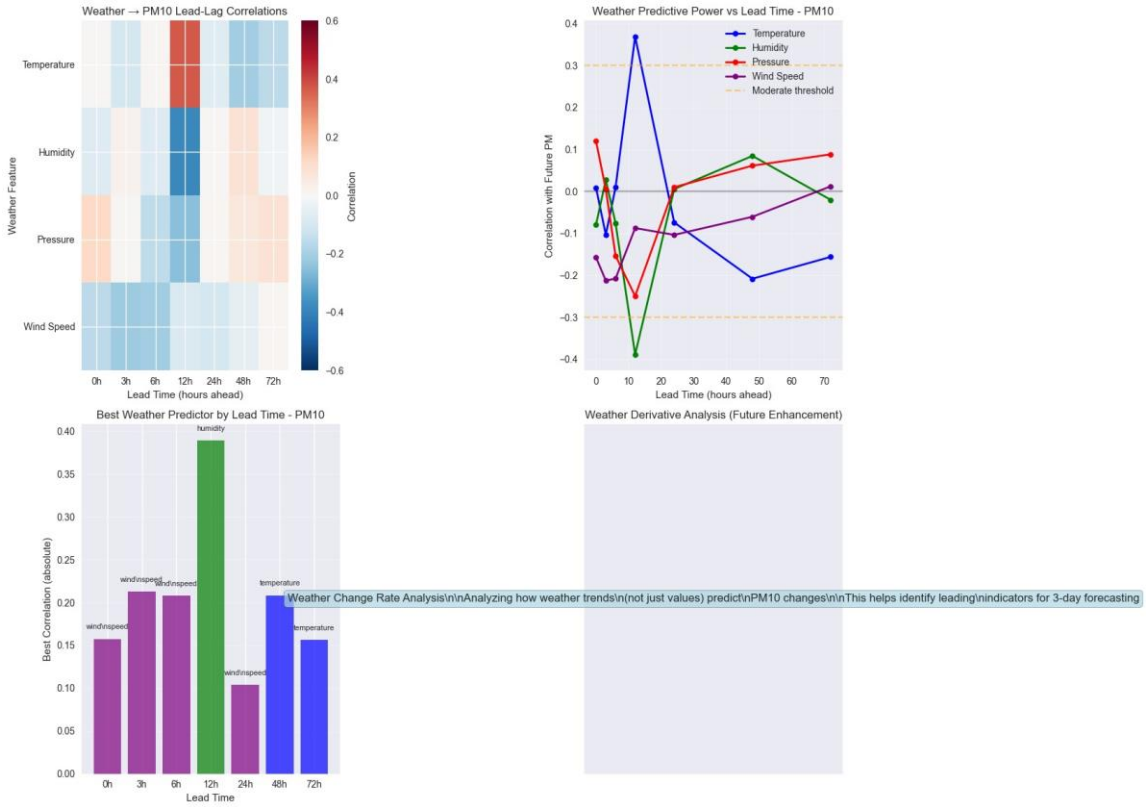
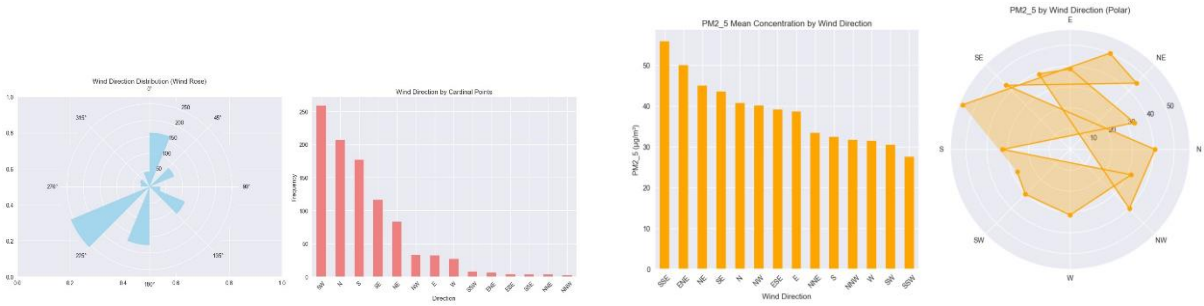


Figure 4.5.3: Weather → PM10 lead–lag correlations and predictive power by lead time

For $PM_{2.5}$, atmospheric pressure provides a stable predictive signal across all forecast horizons. For PM10, humidity and temperature emerge as the most consistent meteorological predictor, while wind speed/direction offers additional value at select horizons. Derivative and trend-based features remain a promising direction for further enhancing forecast accuracy.

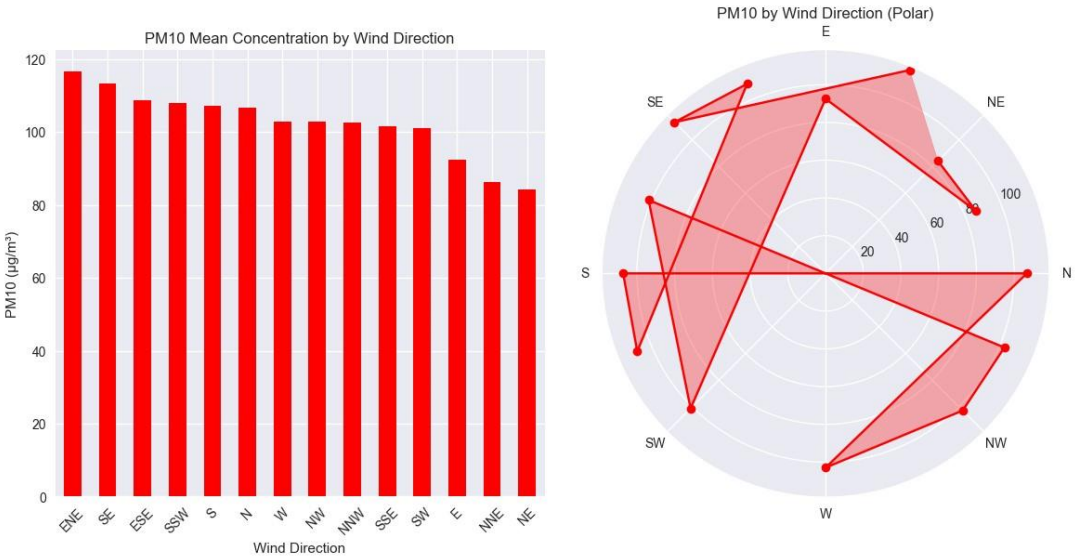
4.7 Wind Direction Effects



(a) Wind direction distribution (wind rose and counts) by wind direction (cartesian/polar)

(b) PM2.5 concentration

Figure 4.6: Prevailing winds and PM10 directional dependence



Findings.

- **Prevailing winds**: Directional frequency clusters in a few sectors, indicating dominant advection patterns.

- **Directional PM response**: Specific sectors align with higher mean PM10/$PM_{2.5}$. Encoding wind direction via sine/cosine, including wind speed, and adding PM×wind interactions captures advection/dispersion effects and reduces residual variance.

4.8 Rolling Features and Forecast Correlation

An analysis of **rolling statistics** revealed strong correlations between pollutant rolling features and future pollutant concentrations, underscoring their predictive value in time-series forecasting. For both $PM_{2.5}$ and PM_{10}, short rolling windows (e.g., 3h mean/min) demonstrated the strongest correlation with near-term forecasts (1–6 hours ahead), while longer rolling windows (12h–24h) were more informative for mid- to long-term horizons (12–72 hours ahead). This aligns with the intuition that short-term pollutant dynamics are driven by immediate fluctuations, whereas longer horizons reflect broader accumulation and dispersion trends. Notably, rolling minima for short windows provided the highest short-horizon correlations ($PM_{2.5}$: $\rho \approx 0.88$; PM_{10}: $\rho \approx 0.87$ at 1h ahead), while 24h rolling means and maxima sustained moderate correlations at 24–72h horizons. These findings validate the inclusion of rolling statistics in the feature pipeline and highlight the need to balance short- and long-window features to support multi-horizon prediction accuracy.

Although LSTM models can inherently learn temporal patterns, these rolling features were included to reinforce trend information and mitigate limitations associated with the relatively small dataset, thereby supporting more robust multi-horizon forecasting.

4.9 Pollutants Lead-Lag Correlation

The correlation analysis shows clear differentiation in the predictive utility of gaseous pollutants for forecasting PM concentrations. **NH_3 and CO emerge as the strongest early-warning indicators for PM2.5**, with average correlations of 0.488 and 0.543 respectively, both peaking at horizons below 6 hours. This suggests that secondary aerosol formation processes linked to these precursors play a critical role in short-term PM spikes. **SO_2 provides moderate predictive power**, particularly for PM2.5 within the first 12 hours, but its influence diminishes over longer horizons. **O_3 correlations are weak and inconsistent**, with slight utility only as a very short-term signal. For PM10, however, none of the gaseous pollutants demonstrate strong predictive value, with correlations largely negligible beyond minimal short-horizon signals from NH_3 and O_3.

4.10 PM-Weather Interactions

Cross-features combining particulate levels with meteorological variables were examined for their ability to anticipate future pollutant concentrations. The analysis revealed that **PM × Pressure** interactions exhibited the highest predictive strength, with correlations peaking above 0.97 at short horizons and sustaining the strongest average values across longer forecasts. **PM × Temperature** and **PM × Humidity** interactions also demonstrated notable predictive relevance, while wind speed interactions consistently showed weak association.

These results highlight how weather–pollution interactions capture important dynamics such as pollutant production, persistence, and dispersion

5. Feature Engineering

5.1 Pipeline (featureengineering.py)

The feature engineering pipeline generates a diverse set of inputs for modelling, including:

- **Time-based features** (cyclical encoding), pollutant- and weather-derived variables, time-aware lags, rolling statistics over 3/6/12/24 hours, rate-of-change metrics, and PM–weather interaction terms.

- **Missing value handling** via forward/backward fill, residual row removal, elimination of non-positive PM readings, and IQR-based clipping of outliers for key pollutants.

5.2 EDA-Driven Modifications

Following insights from **comprehensiveeda.ipynb** and **eda.ipynb**, the script **automatedhourlyrun_updated.py** was introduced with the following enhancements:

- Integration of historic records with new raw data prior to feature engineering, ensuring consistent temporal context.

- Inclusion of additional weather and pollutant lags, as well as rolling statistics, that demonstrated predictive value during EDA.

6. Models

6.1 Direct Multi-Horizon LSTM (banded)

6.1.1 Architecture

1. Encoder (Historical Context)

- **Input**: 72-hour historical window of engineered features for **Short Model (1-12h),** 96-hour window for the **Mid/Long Model (12-72h)**.

- **Architecture**: Two stacked LSTM layers with hidden sizes [128, 64], followed by Batch Normalization.

- **Rationale:** The first layer (128 units) captures short- and medium-range temporal dependencies. The second layer (64 units) compresses representations, reducing dimensionality and mitigating overfitting while retaining sufficient capacity.

- **Output**: Context vector summarizing historical patterns

- **Purpose**: Capture temporal dependencies and patterns in historical data

2. Decoder (Multi-Horizon Prediction)

- **Input**:
    - Repeated context vector (72 times)
    - Target horizon exogenous features (weather, pollutants at t+h)
    - Positional embeddings (sin/cos encoding of step position)
    - Current PM values (repeated across all steps)
- **Architecture:**
    - The context vector is repeated across all steps (RepeatVector).
    - Inputs are concatenated with horizon-specific auxiliary features.
    - A single LSTM layer with **192 units** processes the fused sequence.
    - A TimeDistributed(Dense(2)) layer outputs parallel predictions for $PM_{2.5}$ and PM_{10}.
- **Rationale:** A parallel direct decoder avoids exposure bias inherent in autoregressive approaches. 192 hidden units provide sufficient capacity to integrate historical context with per-horizon signals, without requiring deeper recurrent stacks.
- **Output**: A sequence of 72 × 2 predictions ($PM_{2.5}$, PM10 for each hour)

3. Regularization

To improve generalization and training stability, dropout with a rate of 0.3 was applied after the decoder LSTM, while a light L2 penalty (1e-4) was added on the output layer to stabilize the final pollutant mapping. **Batch Normalization** was incorporated after each LSTM layer to support more stable optimization and faster convergence. Empirical evaluation further demonstrated that BatchNorm consistently outperformed LayerNorm for this AQI forecasting task, achieving a $PM_{2.5}$ RMSE of 1.82 compared to 3.77 at the 1-hour horizon.

6.1.2 Exogenous Features and Noise Injection

Since pollutants and weather variables exhibit strong correlation with current PM levels, we incorporated them as exogenous inputs to the decoder. This allows the model to condition its forecasts on the evolving environmental state, rather than relying solely on historical pollutant sequences.

A challenge arises, however: during training we have access to *true future* pollutant and weather values, but at inference only forecasted values are available. This train–serve mismatch can cause the model to overfit to unrealistic "perfect" inputs, degrading performance when deployed.

To address this, **noise injection** was applied to exogenous features during training. The idea is to replicate the uncertainty of forecasted data, forcing the model to learn robustness under imperfect conditions. Specifically:

- **Pollutants and weather were perturbed** in standardized space with horizon-dependent Gaussian noise. Variance increased with forecast horizon, reflecting the higher uncertainty of long-range forecasts.

- **Temporal correlation** was introduced using an AR(1) process ($\rho = 0.8$), ensuring that injected errors persisted across adjacent time steps, as real forecasts do.

- **Feature-specific treatment**: pollutant inputs received larger perturbations than weather inputs, consistent with observed differences in forecast reliability.

This strategy allowed the model to benefit from informative exogenous features while avoiding brittle reliance on unavailable ground-truth future values.

6.1.3 Banded Training

The system employs a **two-model banded training strategy** designed to optimize performance across different forecast horizons.

The **Short Model** focuses on predictions within the 1–12 hour range, using **delta targets** (the change from the current PM value) rather than absolute concentrations. This approach allows the model to more easily learn short-term fluctuations, with particularly heavy emphasis on the first 6 hours through higher loss weights (8.0 for 1–3 hours, 4.0 for 4–6 hours). It leverages a **72-hour historical window** to capture recent pollutant and weather trends, aiming to maximize **immediate and short-term accuracy**.

In contrast, the **Mid/Long Model** is dedicated to forecasts between 12–72 hours ahead, predicting **absolute PM values** directly. This model uses a **balanced weighting scheme** (1.0–3.0) across horizons and incorporates a **96-hour historical context** to better capture long-range temporal dependencies.

For the delta-based short-term forecasts, predictions are reconstructed into absolute values during inference by adding the current PM level, enabling seamless integration of

both models' outputs into a unified forecast stream. This dual-model design ensures the system excels at both rapid-response short-term forecasting and more stable long-term prediction accuracy.

6.1.4 Rationale for Weights and Loss Function

To balance predictive accuracy across short and long forecast horizons, the models use a **custom horizon-weighted loss function**:

$$Loss = \frac{1}{T}\sum_{t=1}^{T}[MSE_t + 0.2MAE_t] * w_t$$

where w_t is a horizon-specific weight. This design reflects both statistical robustness and operational priorities in AQI forecasting.

Formulation (MSE + 0.2·MAE). The MSE component strongly penalizes large deviations, ensuring the model corrects major forecasting errors such as pollution spikes. The MAE term, down-weighted by a factor of 0.2, provides robustness against outliers and heavy-tailed residuals, while stabilizing gradient updates. This elastic-net style blend offers more explicit control over the L2/L1 balance than Huber loss, enabling cleaner integration with horizon weighting.

Coefficient choice (0.2). The 0.2 factor was selected empirically as a pragmatic compromise: values in the 0.1–0.3 range yielded similar performance, while higher weights over-flattened gradients at longer horizons and lower weights reduced robustness to outliers. The chosen value consistently delivered stable training and balanced short- and mid-horizon performance.

Horizon weights. Forecast steps are weighted to reflect both practical utility and error growth with horizon length. The Short Model (1–12h) emphasizes the earliest hours (weights: 8.0 for 1–3h, 4.0 for 4–6h, tapering thereafter) to maximize responsiveness in operational decision-making. The Mid/Long Model (12–72h) employs a smoother weighting scheme (e.g., 1.0 for 1–3h, 3.0 for 7–24h, 1.5 for 25–72h) to prioritize stability in day-ahead predictions while acknowledging greater uncertainty at extended horizons.

Alternative considered (Huber). While Huber loss is a strong baseline for robustness, it offers only a single transition parameter between L2 and L1 regimes. In contrast, the explicit MSE+MAE composition provides clearer interpretability and tunable balance, particularly when combined with horizon-specific weights. Empirically, the composite loss produced more stable training curves and better calibration at critical horizons.

6.2 Alternate Models

6.2.1 LightGBM

Two variants of LightGBM were explored for multi-horizon AQI forecasting:

- **Direct Multi-Output (v1)** – Implemented using MultiOutputRegressor from scikit-learn, wrapping a single LGBMRegressor instance. This allowed training a separate model for each horizon in a single unified call, enabling simultaneous prediction of all future steps. This approach started with strong short-term performance — for example, at a 1-hour horizon, results were excellent ($PM_{2.5}$: R^2 = 0.981; PM10:R^2 = 0.945). However, accuracy degraded substantially for longer horizons, with $PM_{2.5}$ performance dropping to R^2 = 0.013, and PM10 to R^2 = 0.009 at 24 hours. This indicates the model struggled to generalize well to distant forecasts.

- **Iterated Single-Output (v2)** – A standard LGBMRegressor was trained to predict only the next time step. Predictions were then recursively fed back into the model to forecast further horizons.

Horizon (hours)	PM2.5		PM10	
	RMSE	R^2	RMSE	R^2
1	1.57	0.979	6.66	0.972
6	7.02	0.581	24.76	0.618
12	10.19	0.092	34.47	0.265
24	9.06	0.234	29.44	0.468
48	10.93	-0.301	38.79	0.071
72	12.70	-0.726	45.16	-0.356

Table 1: lightgbm_multi_horizon_trainer_v2.py validation metrics

The iterative LightGBM model began with highly accurate short-term predictions at 1-hour horizon. However, performance degraded significantly over longer horizons due to errors from previous steps compounding up. Training was extremely slow due to the stepwise forecasting approach, yet the early-horizon accuracy made the results promising for short-term air quality prediction.

6.2.2 ExtraTrees and RandomForest

Both ExtraTrees and RandomForest were evaluated in multi-output configurations. While they delivered strong short-term accuracy, with R2R^2R2 values above 0.97 at the 1-hour horizon, performance degraded sharply with increasing lead time, turning negative at longer horizons (48–72h). This degradation pattern was consistent across both models, highlighting their limited ability to capture long-range temporal dependencies despite their solid early-horizon performance.

6.2.3 Seq2Seq LSTM (Full Teacher Forcing, Autoregressive Inference)

This was the first non-tree baseline and the first LSTM model trained, implemented in TensorFlow/Keras. The encoder consisted of stacked LSTMs [128,64] with Batch Normalization, while a 64-unit decoder **autoregressively** generated 72 forecast steps using only previous PM values and sinusoidal positional embeddings (no exogenous inputs). Training used full **teacher forcing**, whereas validation and inference relied on autoregressive rollout. Despite its conceptual simplicity and ability to exploit short-term pollutant dynamics without external forecasts, training proved extremely slow even on GPU, reflecting the high sequential computation cost of autoregressive LSTMs. Performance was further constrained by exposure bias and error accumulation at longer horizons.

6.3 Overfitting in Tree-Based Models

Across all evaluated tree-based approaches — including LightGBM, ExtraTrees, and RandomForest — signs of overfitting were evident despite multiple configuration adjustments. These adjustments included limiting tree depth, increasing the minimum number of samples required for splits and leaf nodes, and applying regularization parameters where supported (e.g., lambda_l1 and lambda_l2 in LightGBM). Overfitting was primarily identified by the large discrepancy between training and testing performance: training R^2 values frequently exceeded 0.95 even for long horizons, while corresponding test R^2 scores dropped sharply, sometimes becoming negative at 48–72h. This performance gap, coupled with comparatively low test-set error reduction after hyperparameter tuning, confirmed that the models were memorizing patterns in the training data rather than learning generalizable temporal relationships.

7. Evaluation & Findings

7.1 Validation Snapshots (Banded Direct LSTM)

Horizon	PM2.5 RMSE	PM2.5 R^2	PM10 RMSE	PM10 R^2
1h	1.952 ± 0.049	0.956 ± 0.002	5.809 ± 0.151	0.977 ± 0.001
6h	6.258 ± 0.132	0.651 ± 0.019	21.642 ± 0.231	0.672 ± 0.007
12h	7.321 ± 0.176	0.415 ± 0.031	24.756 ± 0.336	0.506 ± 0.013
24h	8.620 ± 0.180	0.317 ± 0.031	30.236 ± 0.771	0.381 ± 0.035
48h	8.548 ± 0.210	0.172 ± 0.041	37.101 ± 0.459	0.058 ± 0.023
72h	8.585 ± 0.073	0.173 ± 0.015	36.866 ± 0.356	−0.027 ± 0.021

Across nine independent runs of the Banded Direct LSTM model, performance was consistent at short horizons but degraded steadily over longer forecasts. At a 1-hour horizon, the model achieved **$PM_{2.5}$ RMSE = 1.95 ± 0.05** (R^2 = 0.956 ± 0.002) and **PM10 RMSE = 5.81 ± 0.15** (R^2 = 0.977 ± 0.001), indicating high accuracy and stability.

Forecast skill declined with horizon length, with $PM_{2.5}$ RMSE reaching ~8.55 and PM10

RMSE ~37.10 by 48 hours, and R^2 values approaching zero for PM10 by 72 hours, indicating little predictive power beyond two days for that pollutant. The low standard deviations suggest that variability between runs was minimal, and performance trends were robust. Compared to the tree-based models, performance for this approach showed a much slower decline and maintained strong predictive accuracy over the first 12 hours. As a result, this approach was selected for further experimentation and inference due to its stronger long-horizon stability.

7.2 SHAP (Removed from runtime; assets retained)

Decoder step-level importance was computed using SHAP offline (now removed from code for speed).

Selected images are included for reference:

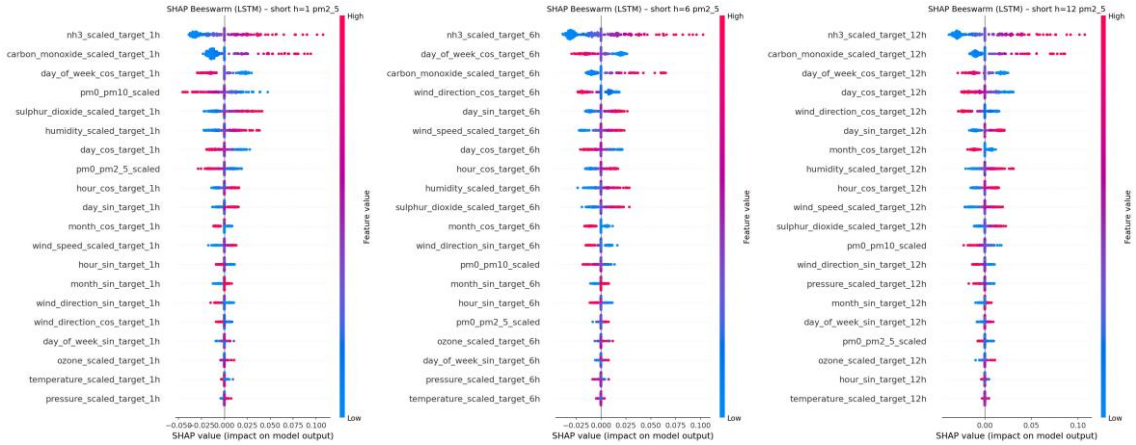


Figure 7.1(a): Short band ($PM_{2.5}$): h=1,6,12 SHAP beeswarm

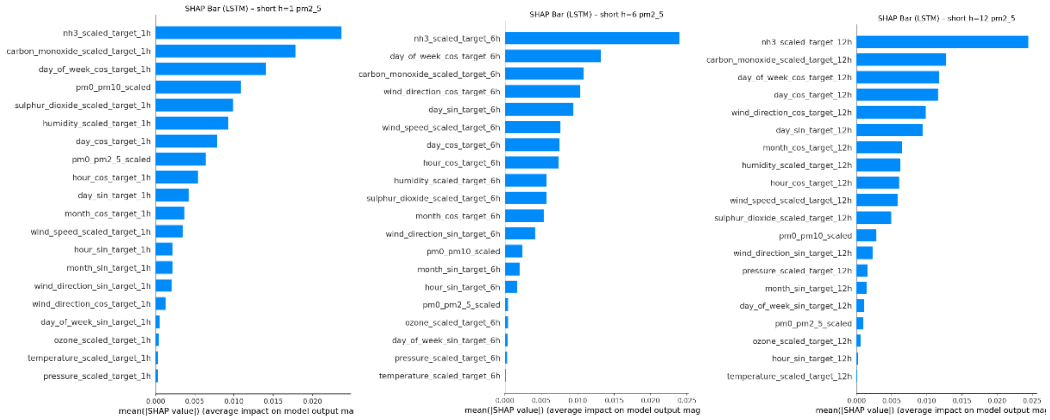


Figure 7.1(b): Short band ($PM_{2.5}$): h=1,6,12 SHAP bar

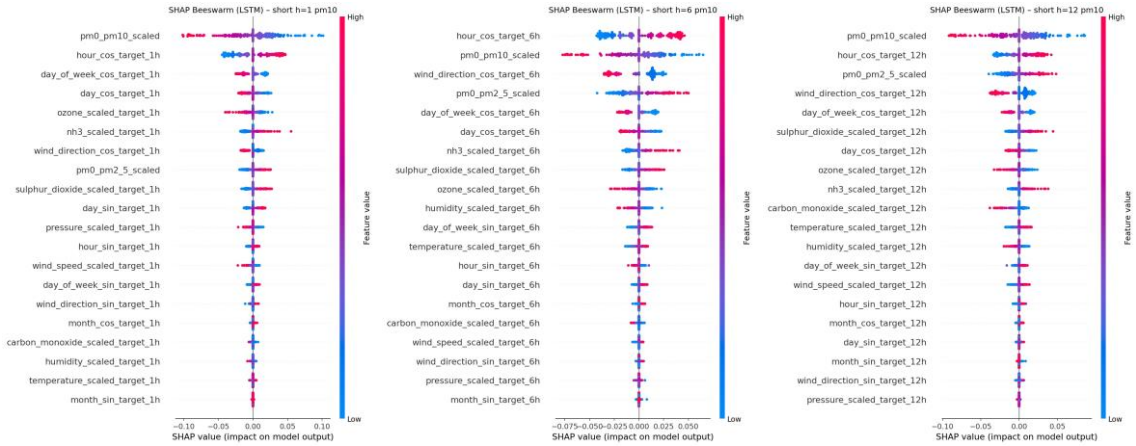


Figure 7.2: Short band (PM10): h=1,6,12 SHAP beeswarm

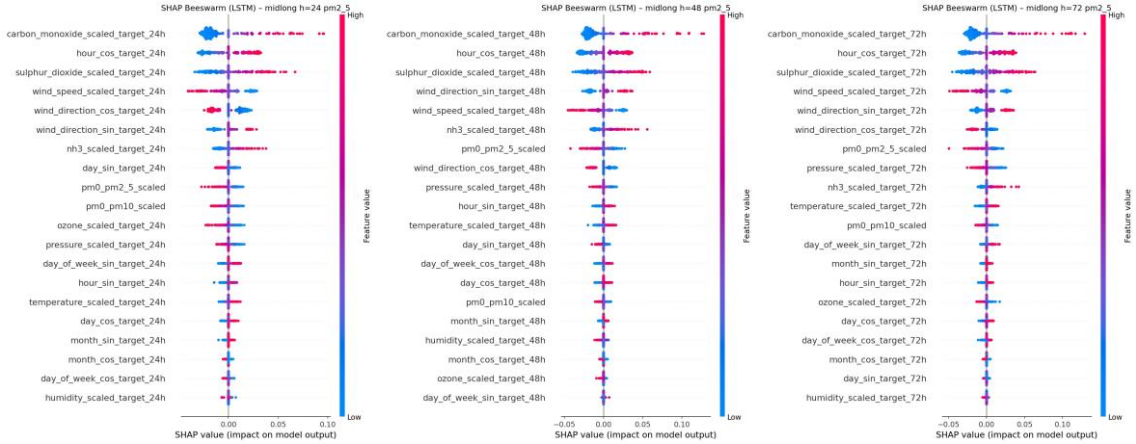


Figure 7.3: Mid/Long ($PM_{2.5}$): h=24,48,72 SHAP beeswarm

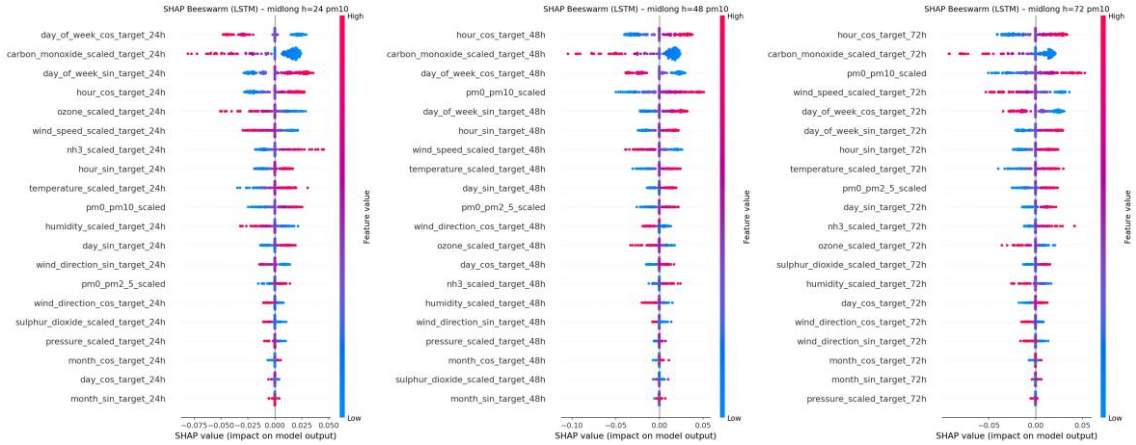


Figure 7.4: Mid/Long (PM10): h=24,48,72 SHAP beeswarm

These results indicate that pollutants such as NH_3 and carbon monoxide are highly influential in predicting $PM_{2.5}$ at shorter forecast horizons. Consequently, the accuracy of meteorological forecasts from OpenWeather plays a critical role in determining the overall precision of the model's predictions. For longer horizons, the model increasingly

relies on temporal features, reflecting a shift from immediate pollutant dependencies to broader time-related patterns.

Notes SHAP computations were slow and were intentionally removed from training scripts to keep CI/CD runtimes reasonable. The interpretability conclusions were used to confirm the value of target-horizon exogenous and certain interactions.

7.3 Training Dynamics (Loss Curves)

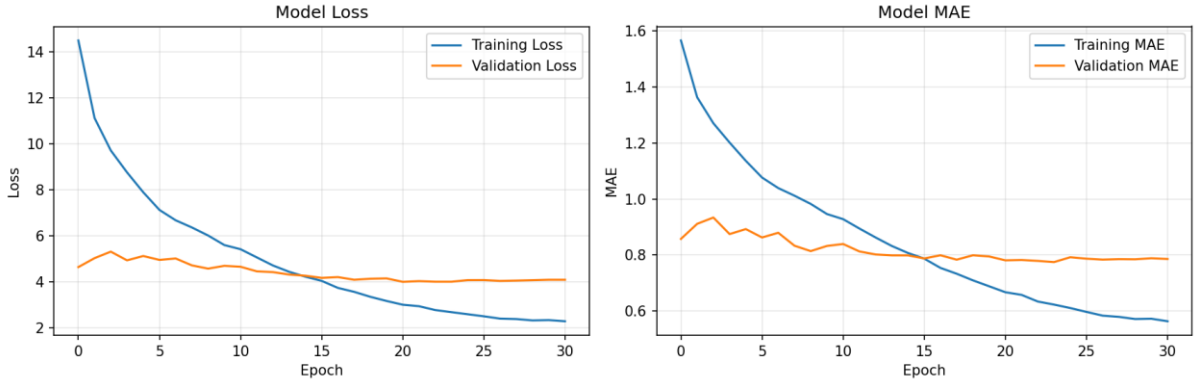


Figure 7.5: Short Model Training/Validation Loss

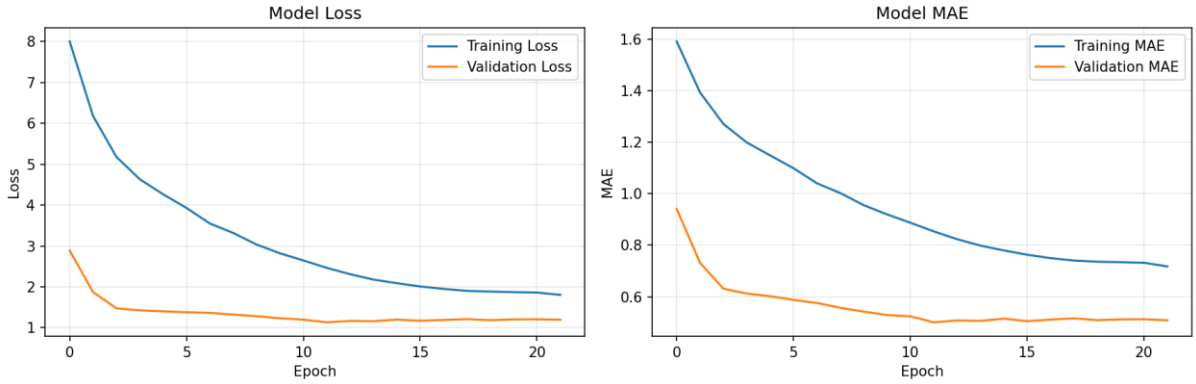


Figure 7.6: Mid/Long Model Training/Validation Loss

The **short-horizon model** converged rapidly, with training loss decreasing sharply from approximately 14.5 to 2.3 within 30 epochs (training MAE ≈1.56 → 0.56). Validation improvement, however, was limited: loss decreased only marginally (≈4.7 → 4.1), while MAE flattened (≈0.86 → 0.78). This indicates a persistent generalization gap: the model continued to exploit the strong short-horizon weighting during training, while validation performance saturated. Signs of mild overfitting appeared after roughly 12–15 epochs, consistent with the use of delta targets and aggressive weighting of early horizons.

The **mid/long-horizon model** displayed more stable convergence. Training loss declined from ≈8.0 to ≈1.8, while validation loss steadily improved from ≈2.9 to ≈1.2. Validation MAE also improved markedly (≈0.94 → 0.50), with the gap between training and

validation curves narrowing over time. This reflects a favorable bias–variance trade-off and robust generalization across longer horizons, with no evidence of divergence.

In both cases, **early stopping** safeguards against overfitting and ensures that evaluation and downstream deployment use the best-performing model weights. Together, these dynamics validate the banded setup, allowing each sub-model to contribute optimally within its respective forecast window.

8. Inference Application

8.1 FastAPI Service (fastapiapp.py)

- Loads banded models (short- and mid/long-term) from the Hopsworks Model Registry and associated datasets.
- Retrieves encoder windows from historicfv (72 h or 96 h) and decoder exogenous inputs from forecastsfv (next 72 h).
- Assembles tensors consistent with training procedures, including scaling, positional embeddings, and repeated current PM values.
- Computes AQI using EPA breakpoint calculations, with NaN handling to prevent errors from transient or missing inputs.

Image 1: Snapshot of the app

8.2 Endpoint Summary

- **/predict**: Returns arrays of $PM_{2.5}$, PM10, and AQI predictions for horizons 1–72 hours.
- **/current**: Returns current PM and AQI values using OpenWeather's real-time pollution API.
- **/**: Provides a modern SPA dashboard with AQI band shading and 24-hour statistics.

9. Automation

9.1 Forecast Collector (automated_forecast_collector.py)

This hourly job (managed as a separate workflow) performs the following tasks:

1. Fetches 72-hour weather and pollution forecasts.
2. Computes time-based features for each horizon, including sine and cosine transformations of wind direction.
3. Aligns timestamps by automatically detecting offsets and merges data on the adjusted hourly steps.
4. Pushes the resulting 72 rows to the forecast Feature Group.

9.2 Historic Hourly Updates

The **automatedhourlyrunupdated.py** script performs EDA-driven updates by:

- Combining existing historic data with new raw rows to maintain correct lag and rolling context.
- Re-engineering all features and pushing the updated data to the historic Feature Group.

The legacy *automatedhourlyrun.py* script is retained for reference.

9.3 Retraining

The daily retraining workflow is triggered once every day on GitHub Actions. It performs the following steps:

- Installs dependencies and executes lstmdirectmultihorizonv1.py with REGISTERTOHOPSWORKS=1.
- Registers model artifacts in the Hopsworks Model Registry, with optional synchronization to Datasets or Production environments.

All automation workflows are executed via GitHub Actions and triggered through cron-job.org. This approach was chosen because GitHub's built-in scheduler occasionally introduces delays, which are undesirable for our system since hourly feature updates are critical for accurate forecasting.

10. Conclusion

The banded direct LSTM approach demonstrates robust performance for short-term AQI forecasting within a scalable, serverless pipeline. Leveraging a Feature Store–centric design ensures consistent train and serve feature schemas and supports reliable online inference. However, long-horizon forecasts remain constrained by the uncertainty in exogenous inputs, and current performance may be further limited by the relatively short historical dataset available; incorporating multi-year data would help the model capture seasonal variations and improve accuracy over extended horizons. Future work includes evaluating longer encoder contexts (120–144 h) and larger decoder capacities to enhance long-horizon predictions, exploring learned positional embeddings and attention mechanisms in the decoder, and developing probabilistic forecasts (quantiles) to better express uncertainty at 48–72 h. Additionally, incorporating regime or volatility indicators, event flags such as dust storms, and potentially a mixed-ensemble approach combining LSTMs with gradient boosting models per horizon band could further improve forecast reliability.

Appendix A: Alternative Training Configurations

This appendix documents optional training configurations supported by the project but not used in the main study. These are provided for completeness and to enable reproducibility or extension of the work.

A.1 Single Model (All Horizons)

- Trains a single LSTM across all 72 horizons rather than banded short/mid-long models.
- Loss weights: 1–3h = 5.0, 4–6h = 3.0, 7–24h = 3.0, 25–72h = 1.0.
- Targets: delta targets enabled.
- Artifacts saved to temp/direct_lstm_multi_horizon/.
- Run via:

```
$env:SINGLE_RUN="1"

python lstm_direct_multi_horizon_v1.py
```

Observation: This configuration yielded weaker performance, particularly at mid and long horizons, indicating the model was unable to balance short-term accuracy against long-horizon generalization. This motivated the shift to the two-model banded setup, which provided more stable and accurate results.

A.2 Walk-Forward Cross-Validation

- Implements rolling, time-ordered folds to better approximate deployment scenarios.
- Default folds: 4 (configurable via WALK_CV_FOLDS).
- Each fold trains both short and mid/long models on growing historical windows.
- Outputs per-fold results and an averaged summary (no artifacts saved).
- Example:

```
$env:WALK_CV="1";
$env:WALK_CV_FOLDS="6"
```

- **Rationale:** Unlike random splits, walk-forward CV respects temporal ordering and avoids data leakage, providing a stronger measure of generalization. In our experiments, the folds consistently showed stable performance across horizons, reinforcing that the models are not overfitting to specific time windows.

Appendix B: Environment & Deployment Setup

Local App

- **Environment Variables (.env)**

```
HOPSWORKS_API_KEY=YOUR_HOPSWORKS_KEY

OPENWEATHER_API_KEY=YOUR_OPENWEATHER_KEY

MODEL_SHORT_NAME=direct_lstm_short          # default if
unset

MODEL_MIDLONG_NAME=direct_lstm_midlong  # default if
unset
```

- Run *pip install requirements.txt*
- Run *python fastapi_app.py*
- **Access**: http://localhost:8080/
- **Prerequisite**: models deployed to Hopsworks Model Registry (Production) under default names direct_lstm_short and direct_lstm_midlong.

Docker

- Same Environment Variables (.env) as above next to *docker-compose.yml*
- Build/Run *docker compose up -d –build*
- **Access**: http://localhost:8000/

For security reasons, the .env file itself is **not committed to GitHub**. However, a valid **Hopsworks API Key** must be provided at runtime, as it is required to authenticate and load the correct feature groups (e.g., historic_fv, forecasts_fv) used by the application. Without this key, the app cannot access the feature store or retrieve the necessary data.

Appendix C: Environment Variables (.env)

```
HOPSWORKS_API_KEY=n3409kwIl97ux3wG.zgkRLpTMYWXEJAU78aeJUd9fMudE1TWsH6
AnDbc4vDiDHwF8dPD3E1QycKcF3SLx

OPENWEATHER_API_KEY=c3cbfd274dd489401797fd118b862a5a

IQAIR_API_KEY=d9bef7b2-9dbd-4515-b28c-671056715581

REGISTER_TO_HOPSWORKS=1

MODEL_SHORT_NAME=direct_lstm_short

MODEL_MIDLONG_NAME=direct_lstm_midlong
```