Executive Summary and Business Context

Business Context and Strategic Importance

Rapid urbanization and increasingly stringent environmental regulations have created an urgent demand for reliable air quality monitoring and forecasting systems. Pollutants such as CO, NOx, and Benzene pose serious public health risks, contribute to regulatory non-compliance, and impose economic costs through increased healthcare spending and reduced workforce productivity.

Strategic Objectives

This project addresses the Urban Environmental Intelligence Challenge by building a Kafka-based streaming analytics platform that ingests sensor data in real-time, performs advanced data analysis, and predicts the Carbon Monoxide level.

Key Findings

- There are strong peaks during daily commute hours, differences in weekday/weekend levels, and seasonal trends that highlight the predictable structure in emissions.
- The XGBoost model achieved the best predictive accuracy, significantly outperforming SARIMA and baselines.
- Short-term lags and daily cycles are dominant predictors, confirming persistence and periodicity in pollution patterns.
- The Kafka streaming architecture reliably ingests, cleans, and processes data in real time, ensuring deployment feasibility. This is important due to the high volume of data

Recommendations

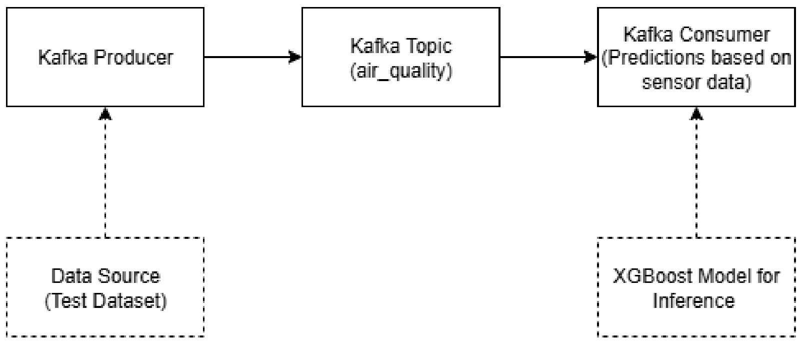
- Deploy forecast-driven early warning systems to notify at-risk populations by giving push alerts on high CO days, especially during commute hours
- Use the predictive insights to optimize traffic light scheduling, and introduce a congestion-based toll pricing system
- Integrate forecasts into workplace and school planning, reducing productivity loss from pollution-driven health impacts.

Technical Architecture and Infrastructure Implementation

Kafka Ecosystem Design

- Deployed Kafka in KRaft mode via Docker for a simplified setup without ZooKeeper.
- Producer streams dataset rows with a fixed delay, simulating real sensor emissions.
- Consumer subscribes to Kafka topics, cleans faulty values, applies feature engineering, and performs model inference.

Architecture Diagram


Kafka Producer
Kafka Topic (air_quality)
Kafka Consumer (Predictions based on sensor data)
Data Source (Test Dataset)
XGBoost Model for Inference

Producer Design

- Streams rows from the test dataset into the Kafka topic
- Lightweight design (no cleaning/transformation) ensures low latency and high throughput, and follows the Single Responsibility Principle

Consumer Design

- Cleans records (datetime parsing, faulty value replacement).
- Maintains a rolling history_df to enable lag and rolling window feature creation.
- Applies the same create_features pipeline as training to ensure feature consistency.
- Runs inference with saved XGBoost model and logs CO predictions vs actual values received from the sensor via Kafka.

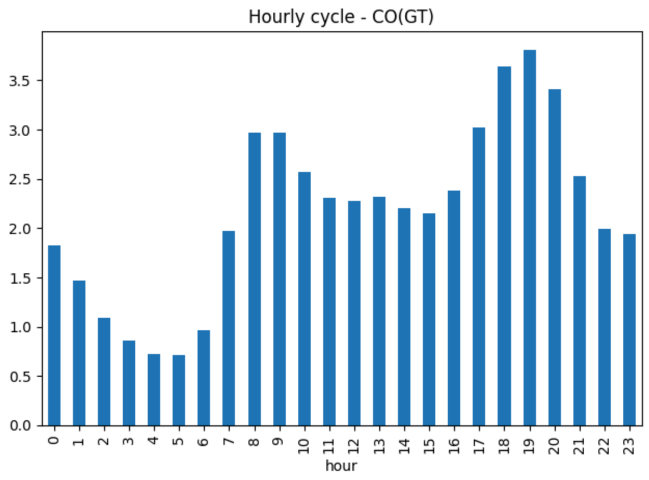
Infrastructure Challenges and Solutions

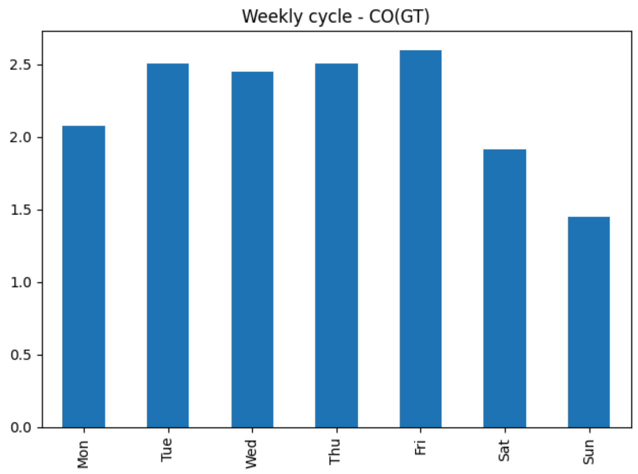
- Datetime serialization: Resolved by converting Pandas timestamps to ISO 8601 in the producer.
- Warm-up dependency: Introduced warm-up steps in the consumer to ensure that the lag/rolling features are available before inference. This is an important step to support the XGBoost model with the feature engineering performed

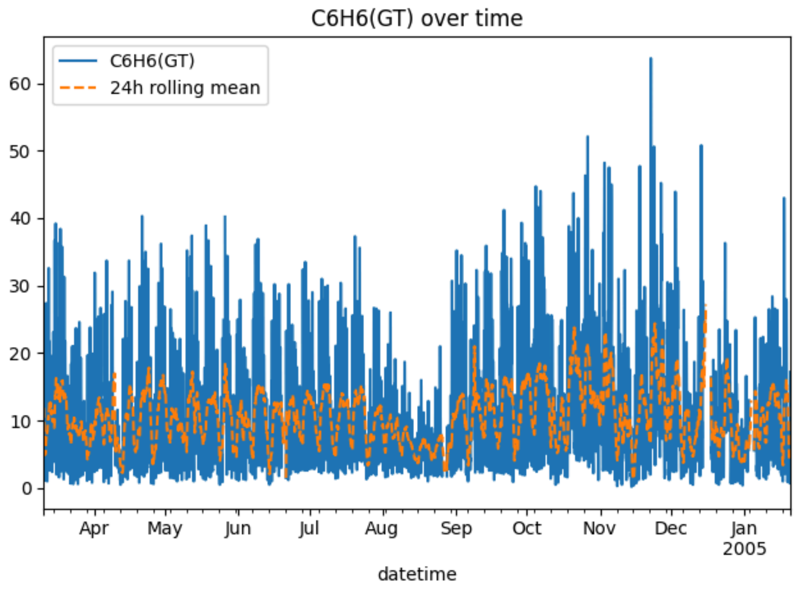
Data Intelligence and Pattern Analysis

Temporal Patterns

- Daily patterns: CO peaks at 7 am - 9 am and and 5 - 8 pm, which aligns with regular commuter traffic
- Weekly patterns: There are elevated levels on weekdays and lower on weekends
- Seasonal variation: Higher pollutant levels in colder months due to heating and inversions which show the seasonal dependency

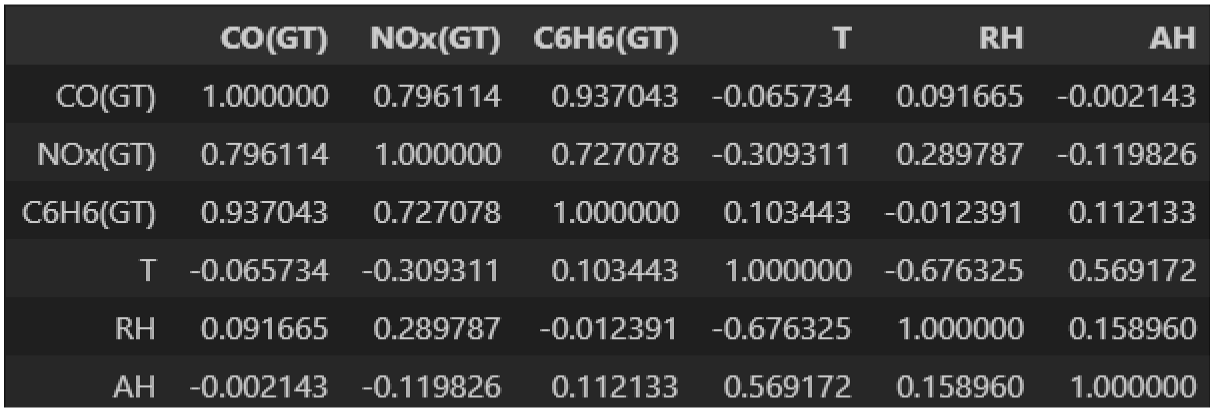






Pollutant Correlations

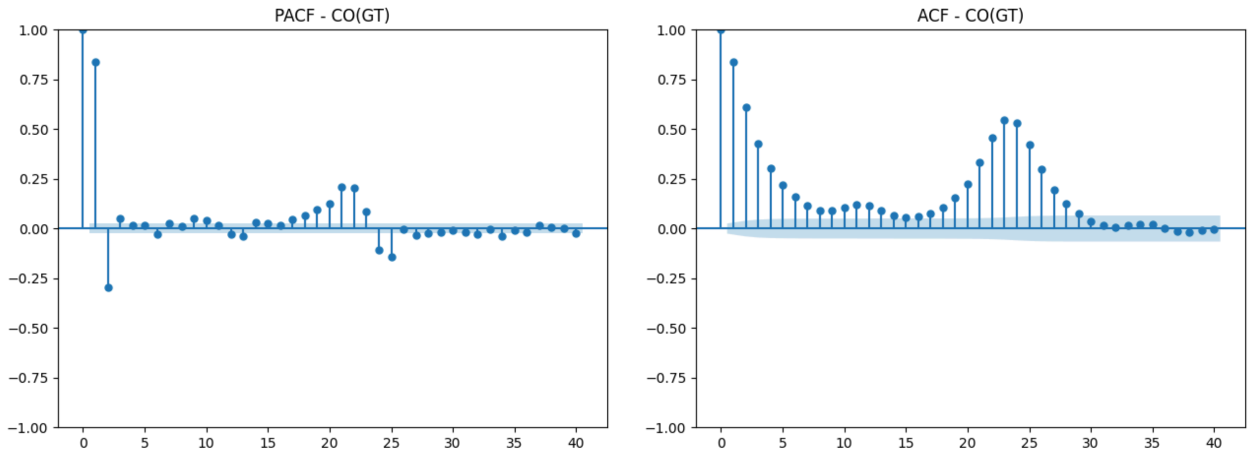
- Strong correlation between CO and C6H6 (≈0.93) and CO and NOx (≈0.79).
- Negative correlation with temperature, indicating pollution accumulation in cold weather.

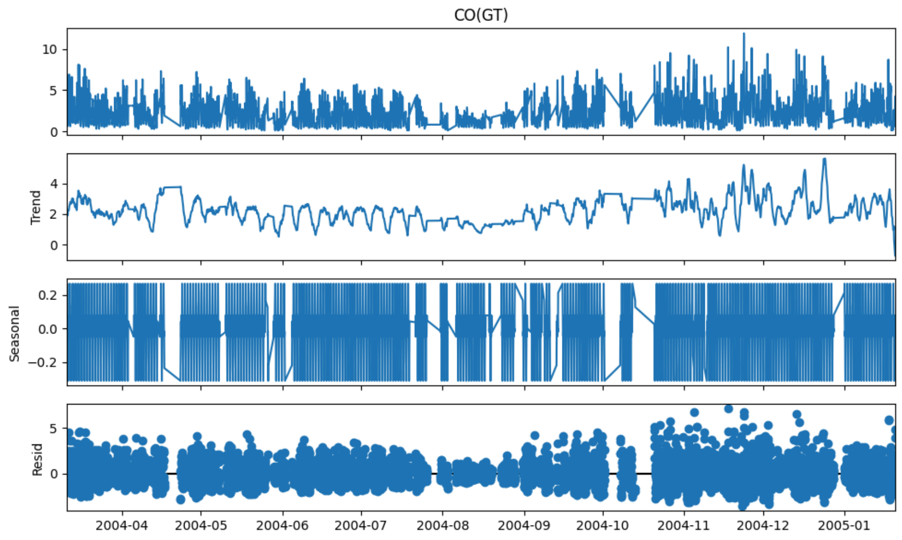


	CO(GT)	NOx(GT)	C6H6(GT)	T	RH	AH
CO(GT)	1.000000	0.796114	0.937043	-0.065734	0.091665	-0.002143
NOx(GT)	0.796114	1.000000	0.727078	-0.309311	0.289787	-0.119826
C6H6(GT)	0.937043	0.727078	1.000000	0.103443	-0.012391	0.112133
T	-0.065734	-0.309311	0.103443	1.000000	-0.676325	0.569172
RH	0.091665	0.289787	-0.012391	-0.676325	1.000000	0.158960
AH	-0.002143	-0.119826	0.112133	0.569172	0.158960	1.000000

Statistical Insights

- ACF/PACF plots reveal significant autocorrelations at 24-hour lags, confirming daily cycles.
- Seasonal decomposition shows trend, seasonality, and anomalies in pollutant data.
- Outlier analysis highlights episodic pollution events (traffic surges, industrial activity)





Business Implications

- Spikes in CO during commute indicate elongated high exposure periods.
- Strong correlations between pollutants suggest shared traffic sources. Specific policies can be created to target them

Predictive Analytics and Model Performance

Feature Engineering Approach

The predictive success of this problem depends on creating features that capture both the short-term effects (daily effects) and the long-term seasonal effects, as identified through the EDA.

- Lag features: 1h, 2h, 24h:
  - Air quality measurements reflect high autocorrelation, ie, past pollution levels are highly predictive of short-term values. Hence:
    - Lag 1h and 2h capture short-term effects
    - Lag 24h captures daily periodicity to address the longer autocorrelation observed in ACF plots
- Rolling averages: 3h, 24h:
  - Since the real-world sensor data is noisy, spikes due to anomalies and outliers within the short term have to be smoothed to preserve long-term trends. Hence:
    - A 3h rolling mean provides short-term smoothing to deal with short-term pollution spikes
    - 24h rolling mean handles specific days, which are outliers due to other factors
- Cyclical Time Encoding:
  - Standard hours of the day are categorical, but time is inherently cyclic. To capture this periodicity, sine and cosine transformations were applied
    - Hour of day captures daily cycles such as commute time peaks and nighttime lows
    - Day of week captures differences in weekdays and weekends
- Imputation Strategy - Forward Fill
  - Missing values were present across multiple features, especially in the sensors for pollutants. The missing values were imputed using forward fill, ie, using the last known valid value. This aligns with the real production context, where valid past readings are available but future readings are not

Together, these features transform the raw data into a rich representation of short-term persistence as well as long-term periodicity. This feature set enables the XGBoost model to capture complex nonlinear interactions between pollutants, time, and environmental conditions

Model Development Methodology

- XGBoost: Gradient boosting with tuned hyperparameters (n_estimators=500, learning_rate=0.05, max_depth=5, subsample=0.8)
  - The XGBoost model was tuned to 500 estimators, a learning rate of 0.05, and a maximum depth of 5, which provided a strong balance between capturing complex pollutant dynamics and avoiding overfitting. Subsample and colsample_bytree values of 0.8 introduced randomness to reduce variance and improve robustness, while forward-fill imputation ensured stable feature availability. These settings were chosen after observing that shallower trees underfit and deeper

ones overfit, while a smaller learning rate consistently improved generalization on the time-based validation split.

- SARIMA with daily seasonality (m=24), and selected order (2,1,1)(1,0,1,24)
    - The SARIMA model was configured using pmdarima.auto_arima, which selected order (2,1,1) with seasonal order (1,0,1,24). The daily seasonal parameter (m=24) reflected the clear 24-hour cycles observed in autocorrelation and decomposition analysis. Although this specification matched the temporal structure of the data, pollutant dynamics proved too irregular and nonlinear for SARIMA, leading to weaker validation performance compared to tree-based methods.

- Seasonal Naive (t-24) for baseline - predicts the observation of the same hour on previous day

Performance Results

Model	RMSE	MAE	R^2
XGBoost	0.9	0.58	0.7
SARIMA	1.84	1.54	-0.23
Seasonal	1.69	1.24	-0.04

Business Interpretation

- XGBoost's accuracy (RMSE ≈0.9) implies forecasts are typically within ±1 ppm of actual CO concentrations.
- This accuracy supports early warning systems for respiratory health risk management.
- Forecasts can help city officials anticipate pollution spikes, enabling proactive interventions (e.g., traffic restrictions).

Production Deployment Strategy

- Real-time inference via consumer pipeline with consistent feature engineering.
- Monitoring of predictions vs actuals to track performance drift
- Retraining strategies:
    - Schedule-based: weekly/monthly retraining to capture the changes in seasonality
    - Performance-triggered: retrain if R^2 drops below the threshold to capture any changes in the overall pattern
    - Canary deployment to validate new models before full rollout.

Strategic Conclusions and Future Enhancements

Project Limitations

- The dataset is restricted to a single city. This will not allow it to generalize to different types of cities and towns due to the differences in industrialization in each place
- The time window is limited and very old. It will not reflect the patterns in the pollution levels now in the current day and age
- While XGBoost delivers good performance, the use of a deep learning model like LSTM can be explored, as they may capture temporal dependencies well
- The architecture was validated on a local environment. It is important to test the system on a cloud native setup with actual sensors to account for the data drift and delays in production systems

Lessons Learned

- Streaming pipelines must ensure feature consistency between training and inference. This is especially important due to the variability in real-world data.
- Simplicity in producer/consumer design increases reliability. By keeping the producer lightweight and making it responsible for only the transmission of data, we can scale both the systems independently and the overall system remains easy to debug
- Statistical models (SARIMA) underperform in complex, noisy, nonlinear environments despite being suited for seasonal data. This is due to the variability in the features. This reiterates the importance of tree-based models to capture non-linearities

Future Enhancements

- Deep learning models like LSTMs or Transformers could further improve accuracy by learning complex temporal dependencies.
- Deploying on cloud-native platforms for scalability to allow for ingestion from thousands of sensors across various data sources(sensors)
- Visualizing forecasts, anomalies, and trends in an accessible format will bridge the gap between technical outputs and actionable insights for policymakers
- Forecasts should trigger alerts when pollution levels exceed safety thresholds. This can be done by pushing the predictions to another Kafka topic, which can notify vulnerable populations, inform traffic control measures, or even trigger regulatory interventions automatically.

Appendix

- Source for git ignore - https://github.com/github/gitignore/blob/main/Python.gitignore
- Reference for setting up Kafka - https://docs.docker.com/guides/kafka/#prerequisites
- AI tools like ChatGPT, Github Copilot, and Claude to assist in debugging code, generating .md templates, and learning about new topics like SARIMA
    - I tested all code generated manually to ensure it works as intended and spent time questioning the LLMs to understand why it suggested things a certain way