Air Quality Prediction using ML Techniques

Manas Goyal 2022MT11918 Nilay Sharma 2022MT12007

Pratyush Sharma 2022MT61970 Vagesh Mahajan 2022MT11260

April 20, 2025

Introduction

- Predicting air quality is vital for public health and environmental planning.
- We use Machine Learning models to forecast AQI, focusing on PM2.5 levels.
- Beyond sensor data, virtual monitoring (e.g. satellites & remote sensing) improves prediction.
- To preserve data privacy, we explore Federated Learning for distributed model training while sharing only the parameters.

Machine Learning Models for AQI Prediction

- The study by [1] compares various ML models Support Vector Machines, Random Forests, and Deep Neural Networks for AQI prediction.
- **Feature Engineering:**
  - Utilizes historical pollutant levels, meteorological features (humidity), and temporal variables (hour of day). These features significantly improve the model's ability to capture environmental patterns.
- **Model Architecture:**
  - Deep Neural Networks outperformed traditional models due to their capability in modeling complex, non-linear relationships. Details include layer depth, activation functions (ReLU, tanh), and optimizers (Adam, SGD).
- **Model Validation:**
  - Implements k-fold cross-validation to ensure robustness. Hyperparameter tuning techniques like grid search and random search were employed.

Federated Learning for AQI Monitoring

- The study by [2] proposes a Federated Learning framework to address the privacy risks associated with centralized data storage.
- **Decentralized Learning Process:**
  - Each participating entity (e.g., city, agency) trains a local model on its private dataset.
  - Only model parameters or gradients are shared with a central aggregator, not the raw data.
- **Privacy Preservation Techniques:**
  - *Differential Privacy:* Adds controlled noise to updates to prevent leakage of sensitive info.
  - *Secure Aggregation:* Ensures that individual updates are encrypted and only aggregated results are visible.

Virtual Monitoring and Real-World Data Challenges

- The study by [3] emphasizes the importance of augmenting real-world monitoring systems with virtual sensors to overcome spatial limitations.
- **Data Integration:**
  - Incorporates data from ground-based sensors, satellite remote sensing, and meteorological sources to improve coverage.
- **Data Preprocessing:**
  - Reduces noise using methods like wavelet transforms.
  - Handles missing data using interpolation techniques. Applies dimensionality reduction via PCA and ICA for better feature representation.
- **Ensemble Learning Strategies:**
  - Combines predictions from multiple models (e.g., bagging, boosting, stacking) to handle uncertainties in environmental data.

Data Collection and Initial Cleaning

- The dataset used was obtained from Kaggle [4] and contains:
  - Air quality indicators: PM2.5, PM10, NO2, CO, Benzene, etc.
  - Meteorological data: Temperature, Relative Humidity, Wind Speed.
- **Data Cleaning:**
  - Removed irrelevant and unnamed columns not contributing to prediction tasks.
- **Time Formatting:**
  - Replaced dots ('.') with colons (':') in the `Time` column for valid time formatting.
  - Created a new `DateTime` column by merging `Date` and `Time`.
  - Set `DateTime` as index to facilitate time series operations.

Imputation, Scaling, and Feature Engineering

- **Imputation:**
  - Median imputation applied to handle remaining missing values.
  - Chosen for its robustness to outliers compared to mean imputation.
- **Scaling and Normalization:**
  - Standard scaling applied to ensure each feature has zero mean and unit variance.
  - Formula used: $x' = \frac{x-\mu}{\sigma}$ where x' is the normalized value, μ is the mean, and σ is the standard deviation.
- **Feature Engineering:**
  - Extracted temporal features from `DateTime`:- Hour of the day, Day, Month, and Day of the week.
  - These features help capture seasonal and time-based trends in pollutant levels.

Gradient-Boosting: LightGBM & XGBoost

LightGBM

- Uses leaf-wise, histogram splitting for faster training and lower memory use.
- Scales to large datasets with parallel and GPU support.
- Objective:

$$\hat{f} = \arg\min_{f} \mathbb{E}_{y,x} L\big(y,\ f(x)\big).$$

XGBoost

- Optimized GBM with built-in regularization and handling of sparse inputs.
- Requires one-hot encoding for categoricals; supports out-of-core and distributed training.
- Update rule at iteration t:

$$\hat{y}_i^t = \hat{y}_i^{t-1} + f_t(x_i).$$

Boosting Variants: CatBoost & AdaBoost

CatBoost

- Builds "oblivious" symmetric trees for fast bit-wise indexing.
- Reduces target-leakage in categoricals via random-permutation encoding:

$$\bar{x}_{ik}^{\sigma} = \frac{\sum_{j \neq i,\, x_{jk}^{\sigma} = x_{ik}^{\sigma}} y_j + a}{\sum_{j \neq i,\, x_{jk}^{\sigma} = x_{ik}^{\sigma}} 1 + a}.$$

AdaBoost Regressor

- Iteratively reweights samples to focus on hard-to-predict points.
- Combines weak learners into a strong ensemble by weighted majority vote.

Bagging Ensembles: RF & Extra Trees

Random Forest

- Aggregates many decision trees trained on bootstrap samples.
- Each split considers a random subset of features, reducing correlation.
- Prediction: $\hat{y} = \frac{1}{K}\sum_{k=1}^{K} h_k(x)$.

Extra Trees

- Randomizes split thresholds as well as feature choice for even greater variance reduction.
- Shares the same averaging equation as RF.

Decision Trees & SVM Regression

Decision Tree

- Splits data to maximize reduction in target standard deviation.
- Split criterion: $\sigma(T) - \sum_i \frac{|T_i|}{|T|} \sigma(T_i)$.

Support Vector Regression

- Fits a "flat" function within an ε-insensitive tube around data.
- Prediction: $f(x) = \sum_{n=1}^{N} (\alpha_n - \alpha_n^*) \, G(x_n, x) + b$.

Linear Models: OLS, Ridge & Lasso

Linear Regression

- Fits $Y = a + bX$ by minimizing squared errors.
- Closed-form:

$$b = \frac{n \sum xy - (\sum x)(\sum y)}{n \sum x^2 - (\sum x)^2}, \quad a = \frac{\sum y - b \sum x}{n}.$$

Ridge Regression

- Adds ℓ_2 penalty $\lambda \sum_j \beta_j^2$ to shrink coefficients.

$$\hat{\beta}^{ridge} = \arg\min_{\beta} \Big\{ \sum_i (y_i - X_i \beta)^2 + \lambda \sum_j \beta_j^2 \Big\}.$$

Lasso Regression

- Uses ℓ_1 penalty $\lambda \sum_j |\beta_j|$ to induce sparsity (feature selection).

$$\hat{\beta}^{lasso} = \arg\min_{\beta} \Big\{ \sum_i (y_i - X_i \beta)^2 + \lambda \sum_j |\beta_j| \Big\}.$$

Seasonal ARIMA (SARIMA): Components

AR (Autoregressive)

- Models y_t as a function of its own past values:

$$\phi(B)\, y_t = (1 - \phi_1 B - \cdots - \phi_p B^p)\, y_t.$$

I (Integrated)

- Removes non-stationarity via differencing: $(1 - B)$ for trend, $(1 - B^s)$ for seasonality.

MA (Moving Average)

- Models y_t via past error terms:

$$\theta(B)\, \varepsilon_t = (1 + \theta_1 B + \cdots + \theta_q B^q)\, \varepsilon_t.$$

Seasonal ARIMA (SARIMA)

Seasonal AR(P), MA(Q) & I(D)

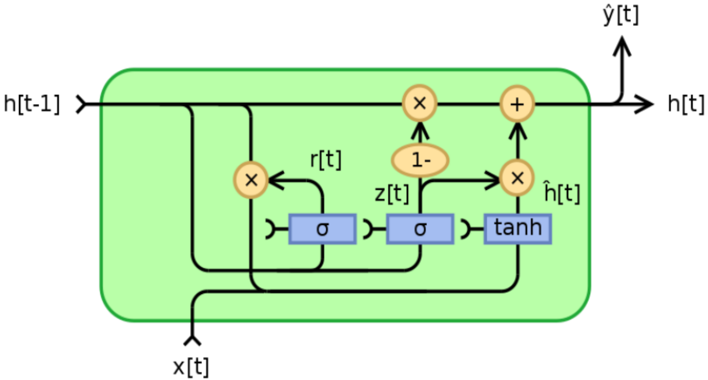
- Seasonal AR: $(1 - \Phi_1 B^s - \cdots - \Phi_P B^{Ps})$
- Seasonal MA: $(1 + \Theta_1 B^s + \cdots + \Theta_Q B^{Qs})$
- Seasonal differencing: $(1 - B^s)^D$
- Seasonal period s: e.g., 12 for monthly data with annual cycle.

$$(1 - \phi_1 B)(1 - \Phi_1 B^s)(1 - B)(1 - B^s)\, y_t = (1 + \theta_1 B)(1 + \Theta_1 B^s)\, \varepsilon_t$$

- $B\, y_t = y_{t-1}$ (backshift operator)
- ϕ_i, θ_j: non-seasonal AR/MA parameters
- Φ_k, Θ_l: seasonal AR/MA parameters
- ε_t: white-noise error

Gated Recurrent Unit (GRU): Overview

- Simplified LSTM variant: merges forget and input gates into an *update gate*.
- Fewer parameters → faster training, less vanishing-gradient issues.
- Two gates:
  - **Update gate** z_t: how much past state to keep.
  - **Reset gate** r_t: how much past to forget.



GRU Cell Equations

GRU cell updates:

$$z_t = \sigma\big(W_z[x_t,\ h_{t-1}]\big) \tag{1}$$
$$r_t = \sigma\big(W_r[x_t,\ h_{t-1}]\big) \tag{2}$$
$$\tilde{h}_t = \tanh\big(W_h[x_t,\ r_t \odot h_{t-1}]\big) \tag{3}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{4}$$

Variables:

$$x_t : \text{ input at time } t, \quad h_{t-1} : \text{ previous hidden state,}$$
$$z_t,\ r_t : \text{ update/reset gates,} \quad \tilde{h}_t : \text{ candidate state,}$$
$$W_z,\ W_r,\ W_h : \text{ weight matrices,} \quad \sigma : \text{ sigmoid function.}$$

GRU-based AQI Prediction

Objective: Predict Air Quality Index (AQI) using temporal pollutant data.

Model Setup:

- **Input:** Daily pollutant concentrations.
- **Architecture:** Single-layer GRU with 50 hidden units.
- **Training:** 100 epochs, learning rate $= 0.001$.
- **Loss Function:** Mean Squared Error (MSE).
- **Optimizer:** Adam.

ARIMA-DBO-RF Hybrid Model: Overview

Motivation: Combine linear statistical modeling with nonlinear machine learning, optimized via metaheuristics, for superior time series prediction (e.g., AQI forecasting).

Model Components:

1. **ARIMA:** Captures linear patterns and trends.
2. **DBO:** Optimizes Random Forest hyperparameters.
3. **Random Forest (RF):** Models complex nonlinear residuals.

Time Series Decomposition:

$$x_t = L_t + N_t$$

where L_t is the linear component modeled by ARIMA and N_t is the nonlinear component learned by RF.

Dung Beetle Optimization (DBO): Key Mechanisms

Inspired by: Natural behavior of dung beetles—rolling, dancing, foraging, reproducing, and stealing.

DBO Operations:

- **Ball Rolling:** Updates position based on past motion and deviation from worst location.

$$x_i(t+1) = x_i(t) + \alpha k x_i(t-1) + b\Delta x$$

- **Dancing (Obstacle):** Changes direction to bypass obstacles.

$$x_i(t+1) = x_i(t) + \tan(\theta) \cdot |x_i(t) - x_i(t-1)|$$

- **Reproduction:** Dynamically adjusts egg-laying zone.

$$B_i(t+1) = X^* + b_1(B_i - Lb^*) + b_2(B_i - Ub^*)$$

- **Feeding & Stealing:** Fine-tune exploration of the search space.

$$x_i(t+1) = x_i(t) + C_1(x_i - Lb^*) + C_2(x_i - Ub^*)$$

Model Integration & Benefits

Final Prediction:

- Combine ARIMA's linear output and RF's nonlinear residual output.
- DBO ensures the RF is well-optimized for generalization.

Hyperparameters Tuned by DBO:

- Number of trees
- Maximum depth
- Min samples split
- Min samples per leaf

Advantages of ARIMA-DBO-RF:

- Accurate modeling of both linear and nonlinear time series behavior.
- Intelligent and adaptive parameter optimization.
- Reduced manual tuning effort.
- Higher prediction accuracy and robustness.

Data Processing & Modeling Pipeline

Data Cleaning & EDA

- Removal of NaNs
- Formatting of date/time
- Exploratory plots: histograms, heatmaps, time series
- Outlier detection via boxplots & scatter plots

Model Evaluation

- Train/test split (70:30) with feature scaling
- Metrics: MAE, RMSE, R^2
- Hyperparameter tuning:
  - Optuna for GRU
  - Dung Beetle Optimization for RF

Model Performance Summary

- Best for NO_2: Extra Trees ($MAE = 26.8$, $R^2 0.858$).
- Best for CO: Extra Trees ($MAE 13.75$, $R^2 0.807$).
- Benzene and NOx: near-perfect by tree ensembles.

Model Performance for NO_2 Concentration Prediction

NO_2 Data Traits:

- Highly non-linear (traffic, weather, time interactions)
- Right-skewed with sharp spikes (heteroscedastic)
- Locally sensitive to perturbations (e.g., wind shifts)

Best Performers:

- **Extra Trees (MAE=26.8)**: Random splits handle noise/outliers
- **Random Forest (MAE=27.7)**: Non-linear spike isolation
- **Boosted Trees (XGBoost/LightGBM/CatBoost)**: Residual-focused for spikes
- **Hybrid (ARIMA-DBO-RF)**: Seasonality + nonlinear residuals
- **GRU (MAE=16.58)**: Learns temporal dynamics (caution: $R^2 = 0.72$ suggests overfitting)

Model Performance for NO_2 Concentration Prediction

Common Failures:

- **Linear Models (MAE≈80)**: Cannot capture non-linearity
- **Seasonal ARIMA**: Misinterprets spikes as noise
- **SVR (MAE=64.1)**: Poor scalability with skewed data
- **AdaBoost (MAE=112.7)**: Over-emphasizes outliers
- **Single Decision Tree (MAE=27.8)**: Overfits without ensemble

Key Insight:

- NO_2 requires models that:
  - Handle sharp non-linearities (tree ensembles)
  - Are robust to skewed distributions (randomized splits)
  - Capture temporal dynamics (GRU/hybrids)

Model Performance for CO Concentration Prediction

CO Data Characteristics:

- Narrow value range with low variance
- Smooth bimodal cycles (morning/evening traffic)
- Fewer extreme outliers (long atmospheric lifetime)

Top Performing Models:

- **Extra Trees (MAE=13.75)**: Random splits handle noise well
- **Random Forest (MAE=14.46)**: Captures time×traffic interactions
- **Boosted Trees (XGBoost/LightGBM)**: Refine traffic peak predictions
- **Hybrid (ARIMA-DBO-RF)**: Combines seasonality + nonlinear residuals
- **GRU (MAE=0.614)**: Learns bimodal patterns (needs regularization)

Model Performance for CO Concentration Prediction

Common Limitations:

- **Linear Models (MAE≈55)**: Cannot capture traffic peaks
- **Seasonal ARIMA (MAE=55.13)**: Too rigid for day-to-day variations
- **Single Decision Tree (MAE=10.91)**: Overfits frequent values
- **SVR (MAE=34.92)**: Oversensitive to parameter tuning
- **AdaBoost (MAE=54.67)**: Overweights minor deviations

Key Insights:

- CO's stability favors:
  - Ensemble methods (ET/RF) for temporal partitioning
  - Hybrid models for cyclical patterns
  - Regularized deep learning for sequential data
- Avoid models that:
  - Assume perfect periodicity (ARIMA)
  - Lack noise robustness (SVR/AdaBoost)

Model Performance Insights for Benzene

Benzene Data Characteristics:

- Highly periodic (strong diurnal/seasonal cycles)
- Low variability and few irregularities
- Smooth patterns with minimal noise

Top Performing Models:

- **Random Forest (MAE=0.0173)**: Perfectly bins cyclical patterns
- **Extra Trees (MAE=0.0268)**: Robust to minor fluctuations
- **Single Decision Tree (MAE=0.0188)**: Memorizes periodic structure
- **Hybrid (ARIMA-DBO-RF)**: Combines seasonality + residual modeling
- **Seasonal ARIMA (MAE=0.8150)**: Naturally fits smooth cycles

Model Performance Insights for Benzene

Strong Alternatives:

- **XGBoost/LightGBM (MAE≈0.08)**: Residual fitting
- **Linear/Ridge (MAE≈0.82)**: Capture basic seasonality

Underperformers:

- **SVR (MAE=2.5972)**: Struggles with periodicity
- **AdaBoost (MAE=1.3720)**: Overcorrects minor deviations
- **Lasso (MAE=1.1855)**: Underfits due to L1 regularization
- **GRU (MAE= 4.185)**: Potential overfitting (R^2=0.7116)

Key Takeaways:

- Benzene's regularity allows near-perfect prediction
- Tree-based methods excel at binning cyclical patterns
- Simple models (linear/ARIMA) perform surprisingly well
- Complex methods offer diminishing returns

Optimal Models for NO_x Prediction

NO_x Data Characteristics:

- Moderate nonlinearity (traffic, industry, weather)
- Strong seasonal/diurnal cycles + episodic spikes
- Winter inversions and rush-hour peaks

Top Performing Models:

- **CatBoost (MAE=51.81)**: Ordered boosting prevents leakage
- **XGBoost (MAE=57.08)**: Handles residual spikes
- **Extra Trees (MAE=58.57)**: Robust to noise via random splits
- **Hybrid (ARIMA-DBO-RF)**: Combines seasonality + nonlinear residuals

Model Performance Insights for NO_x

Specialized Approaches:

- **GRU (MAE=0.614)**: Captures complex temporal patterns
- **Seasonal ARIMA (MAE=125.98)**: Only handles smooth cycles

Limited Effectiveness:

- **Linear Models (MAE≈124.8)**: Miss nonlinearities
- **SVR (MAE=130.94)**: Oversensitive to parameters
- **AdaBoost (MAE=135.70)**: Overweights rare events
- **Single Decision Tree (MAE=72.54)**: Overfits without ensemble

Key Takeaways:

- Requires models that handle both:
  - Regular cycles (seasonal/diurnal)
  - Episodic spikes (traffic/industrial events)
- Hybrid and ensemble methods dominate
- Simple linear/seasonal models fail on extremes

Our journey as a team

- The project involved implementing 14 machine learning and time-series models for air quality prediction.
- Workload was distributed equitably among four team members.
- Tasks included data preprocessing, EDA, model building, hyperparameter tuning, literature review, and analysis.
- Each member owned key components and contributed collaboratively to the final pipeline.

Manas' Contributions

- Sourced and prepared the Kaggle Air Quality dataset.
- Handled missing values, standardized features, and pollutant-wise normalization.
- Performed detailed EDA to discover trends and correlations.
- Implemented Ridge Regression, SVM, Random Forest, and Extra Trees Regressor.
- Analyzed model performance pollutant-wise and validated using domain knowledge.

Pratyush's Contributions

- Performed model centric EDA for their respective models.
- Implemented AdaBoost, Decision Tree, and Lasso Regression.
- Focused on model interpretability and boosting strategies.
- Studied literature on virtual station modeling and imputation.
- Cross-validated pollutant-specific results, especially for NO_2, NO_X, C_6H_6, and CO.

Vagesh's Contributions

- Performed model centric EDA for their respective models.
- Implemented XGBoost, CatBoost, and LightGBM with tuned hyperparameters.
- Focused on gradient boosting ensemble models and generalization.
- Reviewed literature on hybrid optimization and virtual monitoring stations.
- Contributed to performance comparison across ensemble models.

Nilay's Contributions

- Performed model centric EDA for their respective models.
- Developed time-series models: Linear Regression, SARIMA, GRU, and hybrid ARIMA-DBO-RF.
- Experimented with lag variables and seasonality.
- Studied ARIMA-CNN-LSTM-DBO models for hybrid forecasting.
- Implemented DBO for Random Forest hyperparameter optimization.
- Architected the final pipeline integrating statistical and machine learning models.

Collaborative Efforts

- Joint literature review and result interpretation.
- Comparative analysis of model behavior across pollutants.
- Report writing, experimental validation, and performance evaluation were collective tasks.
- Emphasis on seasonal, spatial, and pollutant-specific trends in results.

Future Directions

- Optimize federated learning for scalability.
- Integrate satellite and IoT sensor data.
- Combine physical models with ML for interpretability.

References

- Study on ML models for PM2.5 prediction (ScienceDirect)
- Air-quality prediction based on the ARIMA-DBO-LSTM
- Virtual monitoring station techniques (ScienceDirect)
- Kaggle Air Quality Dataset