# African Air Quality Prediction

AI7101, Machine Learning with Python

| Full Name | Student ID |
|---|---|
| Truong Vu | 25010973 |
| Tung Do | 25010940 |
| Khoi Ho | 25010972 |

Lecturer: Maxim Panov

جامعة محمد بن زايد للذكاء الاصطناعي
MOHAMED BIN ZAYED UNIVERSITY OF ARTIFICIAL INTELLIGENCE

Abu Dhabi, October 3, 2025

# Contents

# 1 | Introduction

## 1.1 | Background

Ambient air pollution, especially fine particulate matter with aerodynamic diameter $\leq 2.5\mu m$ ($PM_{2.5}$), is among the world's most significant environmental health risks. The State of Global Air 2024 report [7] estimates 8.1 million deaths in 2021 attributable to air pollution, making it the second-leading global risk factor for death. In response to epidemiological evidence at lower concentrations, the WHO 2021 Air Quality Guidelines [13] recommend an annual $PM_{2.5}$ level of 5 $\mu g/m^3$.
In Africa, due to biomass and waste burning, traffic and industry, seasonal agricultural fires, and dust outbreaks, the $PM_{2.5}$ level has surpassed the recommended value since 2019 (Fig 1.1). However, due to high costs of installing and maintaining measurement devices, there is not enough devices yet to monitor the air quality on regions that are not equipped with sensors. Thus, there is an urgent need to estimate air pollution in a larger area with restricted access to these devices. Solving this problem can support health interventions and policy decisions in the continent.

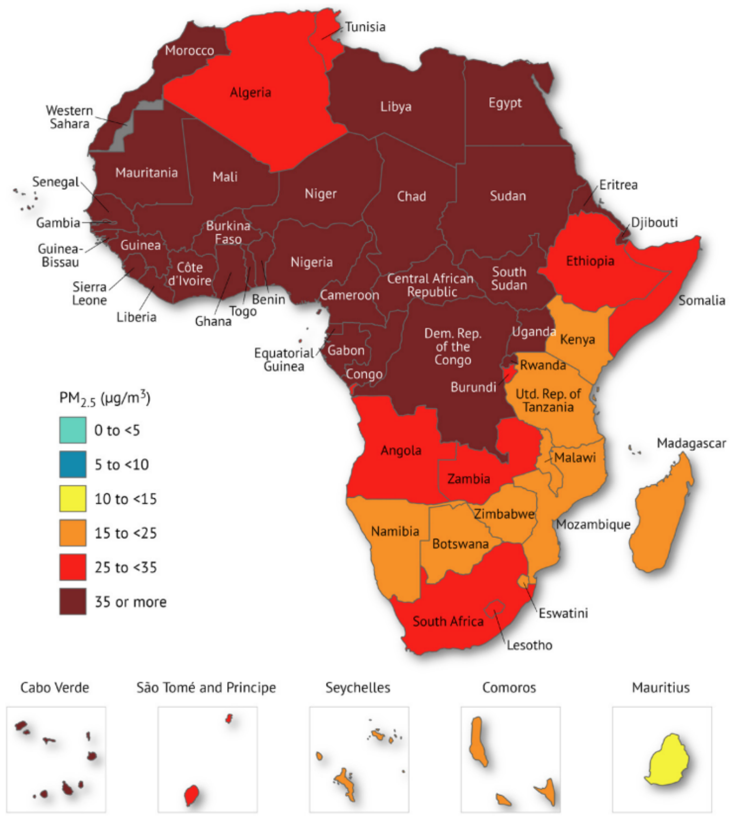

**Figure 1.1:** Population-weighted annual average $PM_{2.5}$ in African countries. Source: [8]

## 1.2 | Our problem

In our problem, we estimate the $PM_{2.5}$ based on Aerosol Optical Depth (AOD) measurements across 8 African cities. The AOD measurements are collected via Sentinel-5P's Google API [6] while corresponding $PM_{2.5}$ values are collected via AirQo platform [1].
We formulate this as a **regression** problem, where ADO measures are used to predict $PM_{2.5}$ values.

# 2 | Exploratory Data Analysis

## 2.1 | Dataset

Our dataset is provided by Zindi [2]. The data is formatted in the form of a train set and a test set. The train set consists of data from 4 cities: Kampala, Lagos, Nairobi and Bujumbura; while the cities in the test set are Kisumu, Gulu, Accra and Yaounde. In each set, there are 79 features, however the $PM_{2.5}$

labels are only available for the train set.
We can categorize the features into two groups:

- Metadata: Collection site ID, coordinates; city data; timestamps for data collection
- Measurements: Density; Cloud fraction; Spherical angle of sensors; etc. across different pollutants (SO2, CO, NO2, O3, HCHO,...)

Loading the data is as simple as

```
train, x_test = pd.read_csv("train.csv"), pd.read_csv("test.csv")
x_train, y_train = train.iloc[:, :-1], train['pm2_5']
```

## 2.2 | Data imbalance

In the dataset, the number of data points for each city varies. This applies to both the training set and the test set, however the effect is more significant on the training set (Fig 2.1). This has a significant effect, whose results will be described later in section 7.3.

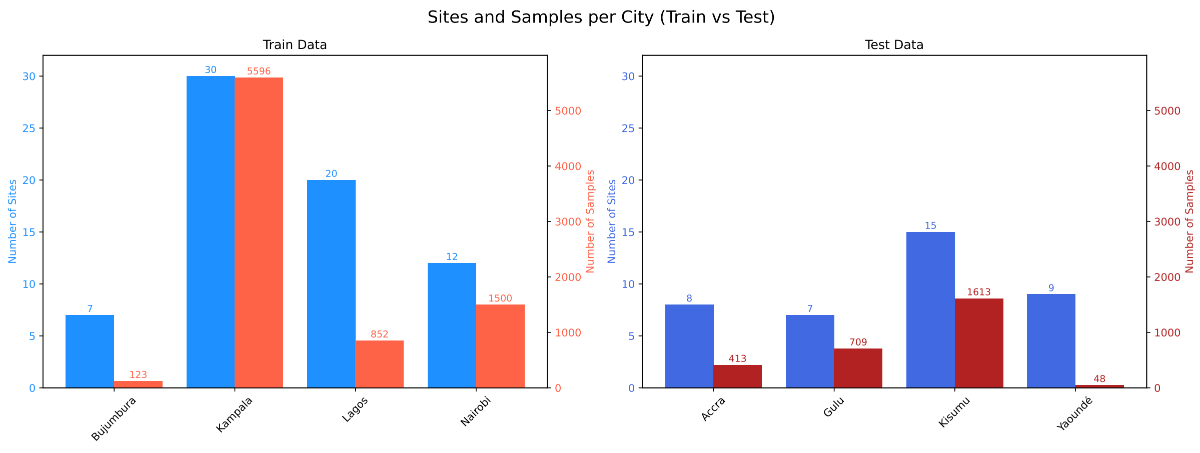

**Figure 2.1:** Number of collection sites and data samples per city in our dataset

The timestamps of data collection is also not equal between cities and between the two sets (Fig 2.2). Aside from the fact that the cities are separated in train/test which means there are no past data, the sparsity and inequality is also a factor to why we did **not** take time series modelling into consideration. On the other hand, while the timestamps are sparse in the latter half, they are similar in all cities, which is a signal that time information will be important features.

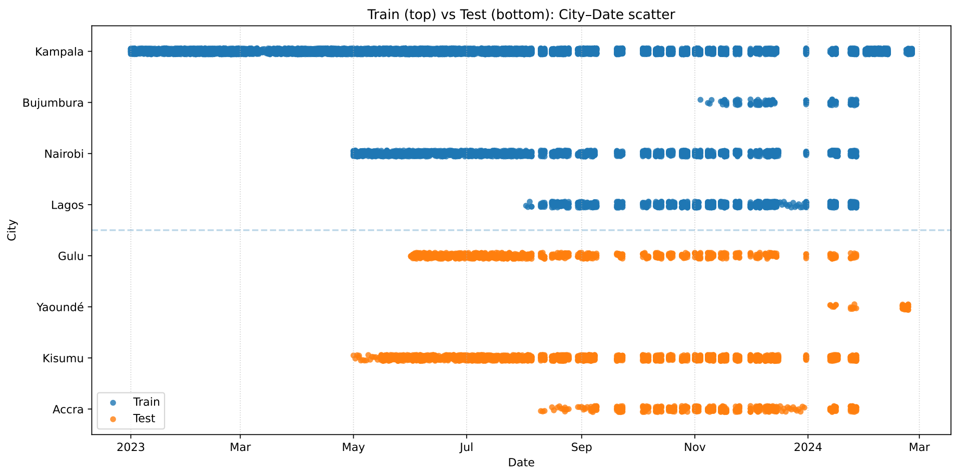

**Figure 2.2:** When were the AOD measures captured in both sets?

## 2.3 | Null values

A large portion of samples in **both** the train set and the test set includes null values. As shown in Fig 2.3, a majority of data is missing, and may be unrecoverable, which we will eventually remove (`uvaerosollayerheight`)

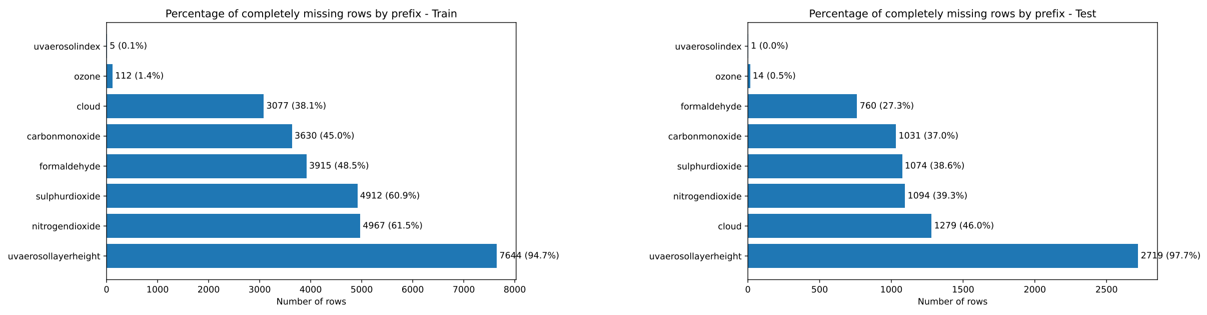

**Figure 2.3:** Proportion of null values per feature on the train/test set

While this simulates the lack of devices in real-life data collection, the large portion of missing data makes it more difficult for accurate modelling. To deal with this, we come up with an imputation strategy in section 3.2.
We also checked wether there are any patterns of null data in the dataset. In our best findings, there are no significant patterns, which suggests that the null data are intentionally injected by the organizers.

## 2.4 | Target variable

We noticed the target variable is heavily right skewed (Fig 2.4), therefore we proposed two methods to help training more stable: log scaling and clipping in section 7.9.

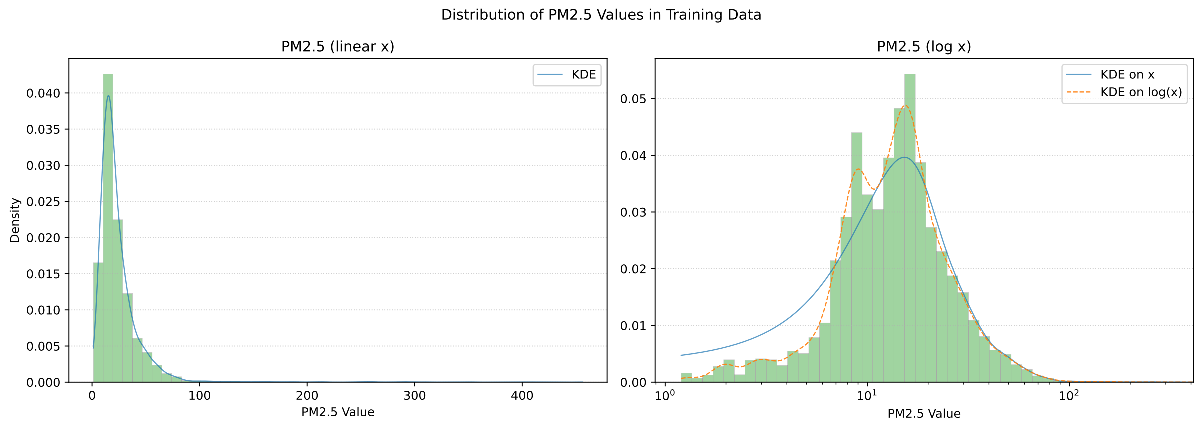

**Figure 2.4:** Distribution of $PM_{2.5}$ value in train set

Moreover, in Fig 2.5, when we look at each city in individual, they have quite the same shape as the whole population. Two points to note here is that Bujumbura does not have quite the shape because its sample size is too small; and most of the outliers ($PM_{2.5} > 100\mu g/m^3$) appeared in Lagos.
Another interesting observation in Fig 2.6 is that in our dataset, the timestamps agrees with scientific findings related to a higher $PM_{2.5}$ value during winter in other regions of the world. Furthermore, while our data contains only measurements from 10a.m. to 2p.m., it seems that the $PM_{2.5}$ value is more varied at noon compared to before and after. This consolidates our point that temporal information can help models understand the context better to provide more accurate predictions.

## 2.5 | Feature correlation analysis

For our dataset, the number of initial features is 79. Removing categorical variables leaves us with 70 features. We now analyze their correlation with each other and with the target variable, which is shown in a rather unclear correlation matrix (Fig 2.7 left).

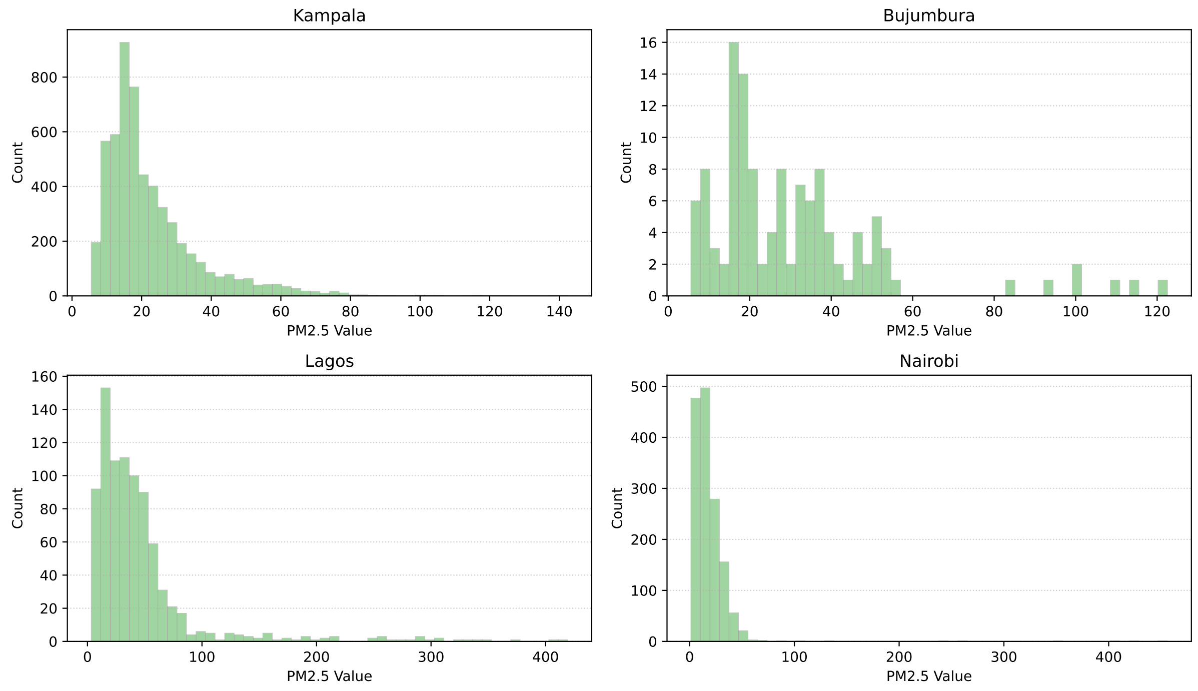

**Figure 2.5:** Distribution of $PM_{2.5}$ value in train set, by city

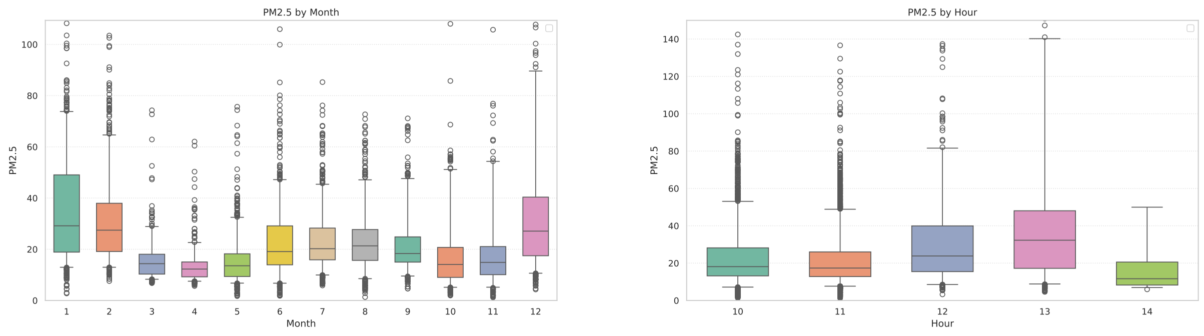

**Figure 2.6:** Distribution of $PM_{2.5}$ by month and by hour at collection point

We saw a repeated patterns of values 1 on the correlation figure, and the reason they appear is most of the AOD values are measured on the same sensors, or similarly-calibrated ones, thus their values are mostly similar or differ by just a fraction of a degree, which is negligible. Based on this, we can fill all the null data in `_angle` and `_altitude` columns and then remove duplicated columns. This helps us reduce 70 numerical features to 38, with a more complex correlation matrix (Fig 2.7 right). We denote this the **unify** feature engineering strategy.
Next, we still observe some highly correlated values. They are `cloud` columns, which has `base` and `top` values due to their 3D structure, and it is no doubt they are highly correlated: a cloud with higher base is more likely to have a higher peak than others. The story goes the same for atmospheric `pressure` feature. Therefore, we construct new features that calculate the gap between `base` and `top`, and call it the **cloud residual** strategy.

After the proposed processing, we are now left with features with lower, insignificant correlation to the target variable (Table 7.4). However, the top 5 features with significantly higher correlations than the rest are still AOD measurements for pollutants (CO, NO2), suggesting that an increase in those values can lead to worse air quality.

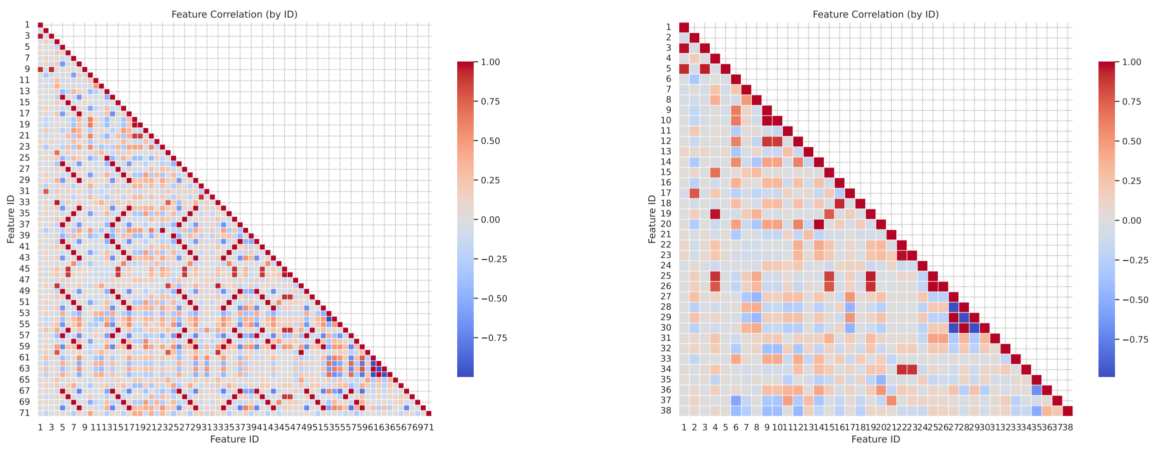

**Figure 2.7:** Feature correlation before and after sensor spherical angle removal

**Table 2.1:** Features after EDA-based processing ranked by correlation with target variable

| Abs(corr) | Feature |
|---|---|
| 0.422418 | `carbonmonoxide_co_column_number_density` |
| 0.403459 | `nitrogendioxide_tropospheric_no2_column_number_density` |
| 0.398677 | `nitrogendioxide_no2_column_number_density` |
| 0.395134 | `nitrogendioxide_no2_slant_column_number_density` |
| 0.327197 | `nitrogendioxide_absorbing_aerosol_index` |
| 0.227742 | `solar_azimuth_angle` |
| 0.199219 | `formaldehyde_tropospheric_hcho_column_number_density` |
| 0.199143 | `altitude` |
| ... | ... |

# 3 | Feature Engineering

Feature Engineering was the phase dedicated to transforming raw and often incomplete data into a set of highly predictive variables. This was particularly crucial due to the complexity of combining time-series, geo-spatial, and sparse measurement data.

## 3.1 | Target Variable Preprocessing: Clipping for Robustness

The raw $PM_{2.5}$ concentration data exhibited a heavily skewed distribution with extreme positive outliers, which can disproportionately influence the RMSE loss function.

- **Technique: Value Clipping (Capping)**
- **Implementation:** The maximum $PM_{2.5}$ value was clamped, or **clipped**, at the **97% quantile** ($\approx 65\mu g/m^3$). All values above this threshold were set equal to the threshold.
- **Rationale:** Clipping stabilizes the training process of the regression model. Since the **RMSE** metric heavily penalizes large errors, clipping outliers prevents the model from obsessively trying to fit rare extreme values, leading to better generalization on the bulk of the data. The chosen threshold corresponds to the World Health Organization's (WHO) warning zone, making the model practically relevant up to a critical public health level.

## 3.2 | Missing Value Imputation: The Three-Tier Time-Series Strategy

Many of the key atmospheric measurement features (e.g., $SO_2, CO, NO_2$) from the Sentinel-5P satellite are highly sparse (due to cloud cover or revisit time). A structured imputation pipeline was used to handle these null values while preserving temporal and spatial relationships.

1. **Within-Group Forward Fill (ffill):** Fills a missing value with the previous valid observation within a specific group defined by city and location. This assumes that $PM_{2.5}$ levels are highly persistent in the short term at a given sensor location.
2. **Within-Group Backward Fill (bfill):** If ffill fails (e.g., at the very start of a time series), the missing value is filled with the next valid observation in the time series for that location.
3. **Global Median Fallback:** If both temporal fills fail, the remaining nulls are imputed using the overall **median** of that feature across the entire dataset. The median is preferred over the mean as it is more robust to the dataset's inherent outliers.

### Example of Sequential Imputation (ffill and bfill)

The imputation strategy for the $PM_{2.5}$ project first groups the data by location (City/Site ID) and then applies sequential filling within each group to handle missing satellite measurement features, such as CO (Carbon Monoxide) column density.

### Raw Data (Grouped by Site)

Consider a short time-series snippet for a single sensor Site A with missing values (NaN):

Table 3.1: Initial Data Segment for CO at Site A

| Time (Hour) | Site | CO Density |
|---|---|---|
| H1 | Site A | 4.50 |
| H2 | Site A | NaN |
| H3 | Site A | NaN |
| H4 | Site A | 4.85 |
| H5 | Site A | NaN |
| H6 | Site A | 5.10 |

### Step 1: Forward Fill (ffill)

The ffill operation propagates the last observed valid value forward to fill subsequent NaN entries.

$$X_t = X_{t-1} \quad \text{if } X_t = \text{NaN and } X_{t-1} \neq \text{NaN}$$

Table 3.2: Result after Forward Fill (ffill)

| Time (Hour) | Site | CO Density |
|---|---|---|
| H1 | Site A | 4.50 |
| H2 | Site A | 4.50 ($\leftarrow$H1 value) |
| H3 | Site A | 4.50 ($\leftarrow$H2 imputed value) |
| H4 | Site A | 4.85 |
| H5 | Site A | 4.85 ($\leftarrow$H4 value) |
| H6 | Site A | 5.10 |

### Step 2: Backward Fill (bfill)

The bfill operation is applied to the data set resulting from Step 1. It is primarily used to fill any NaN values that remained at the beginning of the time series (or after a long gap if the ffill was applied before grouping, which is not the case here).
In our example, all NaN values were successfully filled by ffill. However, if the first row (H1) had been missing, bfill would use the next valid value (H4, if H2 and H3 were also NaN) to fill it.
For the value at H3 in the ffill data above, if it was still NaN, bfill would fill it using the value from H4 (4.85).

$$X_t = X_{t+1} \quad \text{if } X_t = \text{NaN and } X_{t+1} \neq \text{NaN}$$

**Table 3.3:** Result after Backward Fill (bfill) - No change in this example

| **Time (Hour)** | **Site** | CO Density |
|---|---|---|
| H1 | Site A | 4.50 |
| H2 | Site A | 4.50 |
| H3 | Site A | 4.50 |
| H4 | Site A | 4.85 |
| H5 | Site A | 4.85 |
| H6 | Site A | 5.10 |

## 3.3 | Augmented Features: Location and Temporal Components

New features were synthesized to capture non-linear relationships with time and space.

- **Composite Location Feature:**

  ```
  location = site_latitude || site_longitude
  ```

  A unique categorical feature was created by concatenating the latitude and longitude of the sensor site. This allows the model to learn a specific residual bias for each physical sensor location, accounting for hyper-local environmental factors not captured by satellite data.

- **Temporal Features:** Features like **Month**, **ISO Week**, **Day of Week**, and a **Weekend Indicator** were extracted from the measurement timestamp. These capture crucial meteorological and human-behavioral patterns (e.g., lower weekend industrial activity, seasonal wind/rain patterns).

- **Categorical Encoding:** The new categorical features (e.g., Location_ID, Month) were converted into integers using **Label Encoding**. This is often effective for tree-based models (like XGBoost and CatBoost) as the integer values are simply treated as discrete labels in the splitting criteria.

## 3.4 | Feature Selection: Permutation Importance

A key step was to select the optimal subset of features, which was done via a sophisticated, model-specific technique.

- **Technique: Permutation Feature Importance**

- **Mechanism:** After a model is trained, the importance of a feature is measured by the drop in the model's score (RMSE) when the values of that single feature are randomly shuffled (permuted) on the validation set.

- **Advantage:** Unlike simpler impurity-based methods, permutation importance is:

  1. **Model-Agnostic:** It directly measures a feature's impact on the final performance metric (RMSE).
  2. **Realistic:** It accounts for the feature's role in the entire model's context, including interactions with other features.

- **Implementation:** The top $K = 40$ features with the highest permutation importance scores were selected. CatBoost is selected as the base model. The project conclusion notes that a secondary step of correlation-based filtering to remove highly-correlated (redundant) features was attempted but ultimately found to add little value, confirming the robustness of the core Permutation Importance strategy.

## 4 | Cross-Validation Procedure

The project's objective—generalizing air quality prediction across Africa—necessitated a specialized cross-validation approach to prevent data leakage and provide an unbiased estimate of the final model's out-of-sample RMSE.

### 4.1 | Technique: Group K-Fold Cross-Validation (GKF)

Standard K-Fold CV randomly splits data, which would place rows from the same city in both the training and validation sets. This creates leakage and an optimistically low RMSE. Therefore, we utilize Group K-Fold CV as follows (see Table 4.1).

- **Grouping Key: city**
- **Mechanism:** GKF ensures that all data points belonging to a specific **group** (in this case, the **city** or a more granular `site_id`) are contained entirely within *either* the training set *or* the validation set for any given fold.
- **Rationale for Location Separation:** The goal is to predict $PM_{2.5}$ in a **new, unseen city**. By holding out entire cities, the GKF simulates this generalization challenge, yielding an RMSE that realistically reflects the model's performance in a new geographical area.

Table 4.1: Illustration of Group K-Fold Cross-Validation by City

| Fold | Kampala | Nairobi | Lagos | Bujumbura |
|---|---|---|---|---|
| **Fold 1** | Validation | Train | Train | Train |
| **Fold 2** | Train | Validation | Train | Train |
| **Fold 3** | Train | Train | Validation | Train |
| **Fold 4** | Train | Train | Train | Validation |

### 4.2 | Cross-Validation Pipeline

The cross-validation pipeline includes the following steps:

1. **Split and Feature Selection Isolation:** For each fold, GKF separates the data. Then, feature selection is performed **only on the training data of that fold**. This prevents the validation set's target variable from influencing the selection of features, which is a major source of leakage.
2. **Fold Evaluation:** The chosen model is trained on the current fold's training data and evaluated on the held-out validation data to obtain a **fold-specific RMSE**.
3. **Final Performance Estimate:** The reported performance is the **Mean RMSE averaged across all K folds**.

## 5 | Modelling

The final modeling strategy was built on the superior performance of tree-based ensemble methods, combined via a simple **averaging** technique for robustness.

- **Models Used: CatBoost** [10], **LightGBoost** [9], and **XGBoost** [4] (all Gradient Boosting Decision Trees), alongside **SVR** [5] (Support Vector Regression).
- **Rationale:** Ensemble methods combine predictions from multiple models. This is a common practice in machine learning to **reduce prediction variance** and produce a more stable, robust output than any single model could achieve alone. Since Gradient Boosting models and SVR learn in fundamentally different ways, their errors are often uncorrelated, making their combination highly effective.

## 6 | Hyperparameter Search

Hyperparameter tuning is the process of selecting the optimal set of parameters for the base models. This is crucial for controlling model complexity and avoiding overfitting.

### 6.1 | Optimization Strategy: Bayesian Optimization

Given the computational cost of training the ensemble models within the GKF structure, an efficient search technique is required:

- **Method: Bayesian Optimization (via Optuna** [3]**)**
- **Rationale:** Bayesian Optimization uses a probabilistic surrogate model to intelligently select the next set of hyperparameters to test. By balancing **exploration** (searching uncertain regions) and **exploitation** (searching high-performing regions), it finds an optimal configuration in significantly fewer expensive trials compared to Grid Search or Random Search.
- **Goal:** Minimize the Mean RMSE reported by the inner GKF validation loop.

In this project, we use 5 trials for each search. The best model found is then trained with the full training data and evaluated on the test data.

### 6.2 | Key Hyperparameters Tuned

The primary parameters targeted for optimization include:

- **Boosting Models:** learning rate and max depth. The search range of the learning rate is $[0.01, 1]$. The search range of max depth is $[3, 12]$.
- **Lasso:** $\alpha$. The search range for $\alpha$ is $[0.0001, 1]$.
- **SVR:** $C, \epsilon$. The search range for $C$ is $[0.1, 10]$ and for $\epsilon$ is $[0.01, 1]$.

## 7 | Experiments

We conducted a sequence of experiments to understand the effects of feature selection strategies, dimensionality reduction, feature engineering, ensemble modelling, and target preprocessing on model performance. Unless otherwise stated, the baseline model used for experiments was Support Vector Regression (SVR) with default hyperparameters in `sklearn`.

### 7.1 | Baseline Setup

The baseline configuration used an SVR without feature selection and target preprocessing. The feature engineering of the baseline is described in the aforementioned section, but without date augmentation. This achieved private and public leaderboard scores of 17.892 and 14.101, respectively. This setup served as the comparison point for subsequent experiments.

### 7.2 | Cross-Validation: Hyperparameter Search

An advantage of using Optuna [3] compared to Grid Search is that we avoid searching on a sparse space. Instead a continuous space is searched on and the results iteratively improve, as shown in Fig 7.1.

### 7.3 | Cross-Validation: Results

Compared to the test results in the following sections, and compared to the labels in the training set, we can observe a relatively high RMSE. Fig 7.2 explains why this occurs: a very high RMSE when the validation fold is for Lagos. The factor that justify for this problem, despite a high volume of training data, is that the data in Lagos has frequent outliers. This is shown in section 2.4.
This also shows that data quality is greater than quantity: when equipped with diverse data, models can still predict with a smaller error even when the number of training samples is limited (Kampala).

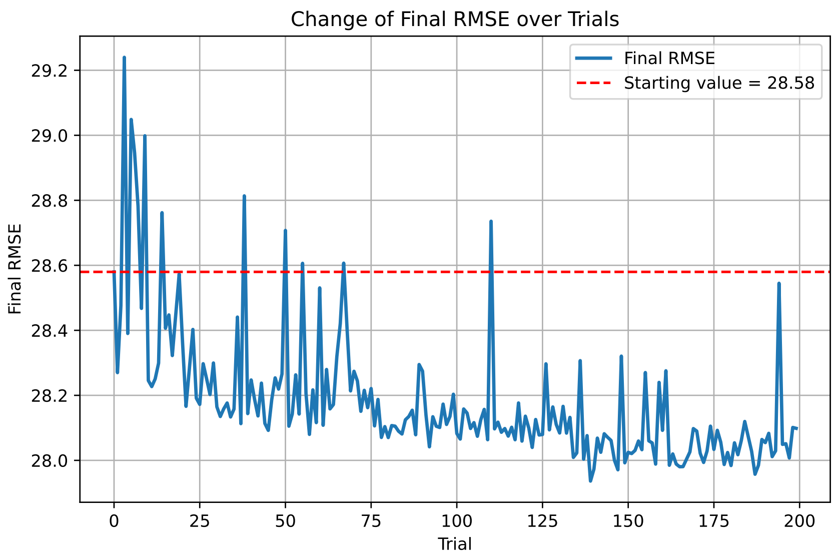

**Figure 7.1:** RMSE on test set after $n$ trials of hyperparameter search

## 7.4 | Ensemble Models

We next explored model ensembling. As seen in Table 7.1, combining SVR with tree-based models consistently improved results, while adding Lasso to the ensemble did not help. The best ensemble combined SVR, CatBoost, LightGBM, and XGBoost, matching the baseline private/public balance (17.892 / 14.101).

**Table 7.1:** Performance of ensemble models.

| Ensemble model | | | | | Private score | Public score |
|---|---|---|---|---|---|---|
| SVR [5] | CatBoost [10] | LightGBM [9] | XGBoost [4] | Lasso [11] | | |
| ✓ | | | | | 19.394 | 15.866 |
| ✓ | ✓ | | | | 18.159 | 14.254 |
| ✓ | ✓ | ✓ | | | 18.066 | 14.233 |
| ✓ | ✓ | ✓ | ✓ | | **17.892** | **14.101** |
| ✓ | ✓ | ✓ | ✓ | ✓ | 18.099 | 14.297 |

## 7.5 | Feature Selection Strategies

Several feature selection techniques were evaluated to reduce redundancy and highlight the most predictive variables. As shown in Table 7.2, permutation-based importance yielded the best results, improving the public score to 14.068. ANOVA and CatBoost feature importance provided modest results, while Lasso-based selection underperformed.

**Table 7.2:** Comparison of feature selection strategies.

| Feature selection | Private score | Public score |
|---|---|---|
| Baseline | 17.892 | 14.101 |
| CatBoost [10] | 17.987 | 14.074 |
| ANOVA [12] | 17.931 | 14.196 |
| Lasso [11] | 18.250 | 14.454 |
| Permutation | **17.855** | **14.068** |

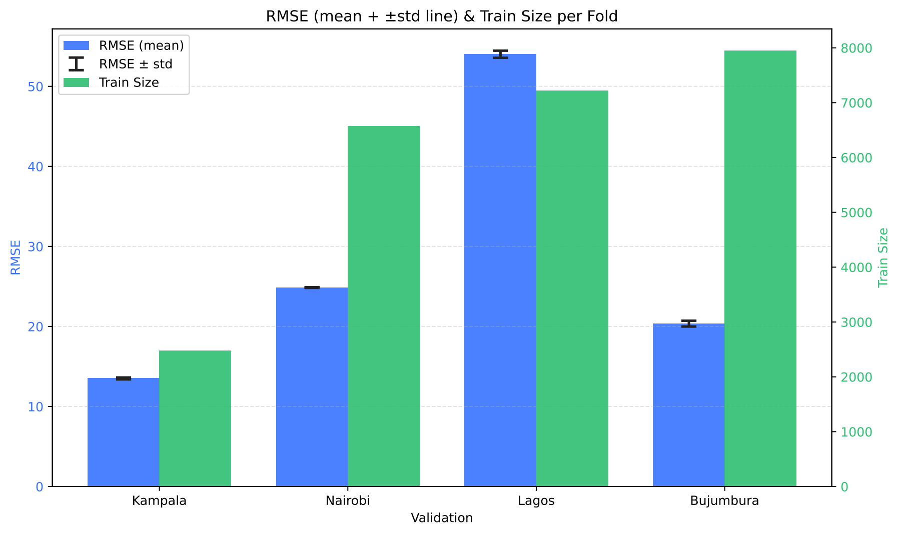

**Figure 7.2:** RMSE and training size on each fold of cross-validation

## 7.6 | Number of Selected Features

Using permutation importance, we further tuned the number of top features retained. Table 7.3 shows that keeping the top 40 features produced the most stable balance between private and public scores. Selecting too few (20) reduced private performance, while selecting too many (60) slightly degraded generalization.

**Table 7.3:** Effect of number of top features retained.

| Num top features | Private score | Public score |
|---|---|---|
| 20 | 18.009 | **14.026** |
| 40 | **17.855** | 14.068 |
| 60 | 17.883 | 14.132 |

## 7.7 | Correlation Threshold Filtering

We tested different thresholds for correlation-based feature removal. As seen in Table 7.4, the loosest setting (keeping all features up to correlation 1.0) performed best. Stricter thresholds (0.75, 0.90) removed too many features and slightly hurt performance. This result indicates that using permutation feature selection is already enough.

**Table 7.4:** Correlation threshold filtering.

| Correlation threshold | Private score | Public score |
|---|---|---|
| 0.75 | 17.957 | 14.081 |
| 0.90 | 17.921 | 14.094 |
| 1.00 | **17.855** | **14.068** |

## 7.8 | Feature Engineering Ablations

To assess the value of different engineered features, we compared variants of the baseline. The details for the **unify** and **cloud residual** are explicitly mention in section 2.5. Results are summarized in Table 7.5. Dropping the location feature or unifying features degraded performance. Augmenting the date with additional temporal components produced the best improvement (public score = 14.035). Adding cloud pressure residuals also underperformed relative to baseline.
So why did the proposed feature engineering technique failed? First of all, dropping location features is a certain information loss, while the unify strategy may not be meaningful as we configured to drop features anyway. Finally, cloud residual information can be uninformative to our models.

Table 7.5: Impact of feature engineering strategies.

| Feature Engineering | Private score | Public score |
|---|---|---|
| Baseline | 17.855 | 14.068 |
| + drop location | 18.332 | 14.594 |
| + augment date | **17.828** | **14.035** |
| + unify | 18.084 | 14.359 |
| + cloud residual | 17.944 | 14.179 |

## 7.9 | Target Preprocessing

Finally, we tested transformations of the target variable ($PM_{2.5}$) suggested in section 2.4. As Table 7.6 shows, clipping extreme outliers improved performance considerably, lowering the public score to 13.813. In contrast, log-scaling alone or log-scaling with clipping degraded results, suggesting that simple clipping was the most effective stabilization.

Table 7.6: Impact of target preprocessing.

| Target preprocessing | | Private score | Public score |
|---|---|---|---|
| Log Scale | Clip | | |
| - | - | 17.828 | 14.035 |
| - | ✓ | **17.770** | **13.813** |
| ✓ | - | 18.031 | 14.172 |
| ✓ | ✓ | 18.188 | 14.270 |

## 7.10 | Summary

Overall, permutation-based feature selection with 40 features, correlation threshold of 1.0, date augmentation, and outlier clipping produced the best generalization performance. Ensemble learning with tree-based models further stabilized results. These experiments demonstrate the importance of careful preprocessing and validation design for robust $PM_{2.5}$ prediction.

# Conclusion

In this project, we tackle the problem of predicting the component of $PM_{2.5}$ on city-scale given satellite-aided metrics. This can help African countries in operational monitoring, public health, and policy making. In a short period of time, we achieved a score equivalent to the top 10% of participants, showing a promising room for improvements.
With extra data collection, our problem can be extended on a larger scale, especially for regions with high $PM_{2.5}$ such as Abu Dhabi. Also, our problem can be formulated as a time series problem in case of more dense timestamps in the dataset, which can add additional predictive values and more temporal insights.
Finally, we would like to thank Prof. Maxim Panov and all the TAs of the course AI7101 to provide us with an enriching and intellectually stimulating learning experience.

# Contributions

- **Khoi Ho:** Problem formulation, EDA, cross-validation results.
- **Truong Vu:** Feature engineering, modelling, cross-validation.
- **Tung Do:** Experiment, ablation study, conclusion.

## 8 | References

[1] AirQo. https://www.airqo.net/home. Makerere University project. Accessed 2025-10-03.

[2] AirQo African Air Quality Prediction Challenge. https://zindi.africa/competitions/airqo-african-air-quality-prediction-challenge. Competition page; accessed 2025-10-03.

[3] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, 2019.

[4] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[5] Harris Drucker, Christopher J Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. *Advances in neural information processing systems*, 9, 1996.

[6] Google Earth Engine Team. Sentinel-5P Datasets in Earth Engine. https://developers.google.com/earth-engine/datasets/catalog/sentinel-5p. Earth Engine Data Catalog. Accessed 2025-10-03.

[7] Health Effects Institute. State of Global Air 2024. https://www.stateofglobalair.org/resources/report/state-global-air-report-2024. Produced in partnership with UNICEF; the State of Global Air is a collaboration between HEI and IHME's Global Burden of Disease project.

[8] Health Effects Institute. The State of Air Quality and Health Impacts in Africa. https://www.stateofglobalair.org/sites/default/files/documents/2022-10/soga-africa-report.pdf.

[9] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[10] Liudmila Ostroumova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In *Neural Information Processing Systems*, 2017.

[11] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

[12] Ben James Winer, Donald R Brown, Kenneth M Michels, et al. *Statistical principles in experimental design*, volume 2. Mcgraw-hill New York, 1971.

[13] World Health Organization. WHO global air quality guidelines: particulate matter (PM2.5 and PM10), ozone, nitrogen dioxide, sulfur dioxide and carbon monoxide. https://www.who.int/publications/i/item/9789240034228. Licence: CC BY-NC-SA 3.0 IGO.