Data Preparation, Feature Engineering, and Model Exploration

Data Preparation/Feature Engineering

1. Overview

This phase details the systematic, multi-stage pipeline designed to ingest, process, and transform raw data into a feature-rich, analysis-ready dataset. The core objective of the project is to forecast the next-day PM2.5 concentration and the corresponding Air Quality Index (AQI) category for Chiang Rai, Thailand.[1] The success of this predictive modeling task is fundamentally contingent upon the quality and structure of the input data.

The process begins with the programmatic ingestion of data from two distinct, external time-series sources: air quality data from the OpenAQ platform and meteorological data from the Open-Meteo API.[1, 1] The subsequent pipeline is designed to address the significant challenges of data fusion and temporal definition. This involves a rigorous data cleaning stage to handle missing values and inconsistencies using time-series-appropriate methods.[1] Following cleaning, a critical transformation step aggregates the data from its raw hourly resolution to daily averages, a deliberate methodological choice to align the dataset's granularity with the project's daily forecasting objective.[1] Finally, the fused and aggregated data undergoes feature engineering and normalization to create a robust, temporally-aligned dataset suitable for ingestion by the selected machine learning models.[1]

2. Data Collection

The project's foundation is built upon two primary, publicly accessible, and API-driven data sources. This selection was the result of a formal data research and comparative analysis phase.[1]

- **Air Quality Data:** The source for historical air quality data is the **OpenAQ Platform**. This platform aggregates ground-level air quality data from official monitoring stations globally.[1] For this project, the specific parameter of interest—the model's primary regression target—is the **PM2.5 concentration**, measured in **micrograms per cubic meter**.[1, 1] A critical methodological decision in selecting OpenAQ was its provision of raw, physical pollutant measurements rather than a pre-calculated AQI. This allows the model to predict the physical phenomenon directly, with the AQI category derived as a deterministic post-processing step.[1]
- **Meteorological Data:** The source for corresponding historical weather data is the **Open-Meteo API**. This API provides a comprehensive set of meteorological variables, including **Temperature**, **Relative Humidity (percent)**, **Wind Speed (kilometers per hour)**, **Wind Direction (degrees)**, **Precipitation (millimeters)**, and **Atmospheric Pressure (hectopascals)**.[1] Data was acquired via direct API calls using the Python Requests library.[1] Open-Meteo was strategically selected over alternatives (such as OpenWeather) after a comparative analysis revealed its significant advantages for this project. Specifically, Open-Meteo provides a generous free tier (10,000 daily calls) and does not require an API key for non-commercial use, which proactively mitigates the "Technical Constraints" and "API rate limits" identified as a key project risk in the implementation plan.[1, 1]

The data collection strategy is summarized in the table below.

Table 1: Data Sources and Collection Strategy

Feature	**Air Quality Data**	**Meteorological Data**
Source	OpenAQ Platform [1]	Open-Meteo API [1]
Parameters	PM2.5 Concentration (**micrograms per cubic meter**) [1]	Temperature, Humidity, Wind (Speed/Direction), Precipitation, Pressure [1]
Access Method	py-openaq Python library [1]	Python Requests library [1]
Rationale	Provides raw physical measurements, not a pre-calculated index.[1]	No API key required; generous free tier (10,000 calls/day) mitigates project

		risk.[1, 1]

3. Data Cleaning

A "Data Preprocessing Pipeline" was designed to systematically clean, transform, and structure the raw ingested data.[1] This process involved several critical steps to ensure the temporal integrity and consistency of the final dataset.

- **Handling Missing Values:** Raw time-series data from APIs is frequently incomplete. To address this, the project plan specified time-series-appropriate imputation techniques, namely **linear interpolation** or **forward-fill**.[1] This is a crucial methodological choice. Naive methods, such as dropping missing rows or imputing a global mean, would be invalid as they would break the temporal sequence and destroy the autocorrelation inherent in the data. The chosen methods "maintain the time-series integrity" by using the temporal context of the data to estimate the missing values.[1]
- **Temporal Aggregation:** The raw data from both OpenAQ and Open-Meteo was provided at an hourly resolution.[1] A key data transformation step was the aggregation of this high-granularity data into **daily averages**.[1, 1] This step serves two purposes: 1) It aligns the dataset's temporality with the project's core objective of forecasting the *next-day* average PM2.5 level.[1] 2) The 24-hour average is a standard measurement period for many official AQI calculations. This aggregation represents a deliberate trade-off, sacrificing intra-day granularity (e.g., pollution spikes during rush hour) for inter-day stability and direct alignment with the problem definition.
- **Data Merging:** Following cleaning and aggregation, the two distinct datasets (air quality and weather) were merged into a single pandas DataFrame. This fusion used the **date** as the common key, creating a unified time-series dataset ready for feature engineering.[1]

4. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) was a critical diagnostic phase intended to uncover patterns, validate assumptions, and inform the project's advanced modeling strategy.

Note: The final project report, presentation, and code notebook containing the generated EDA visualizations and specific quantitative insights were not accessible from the provided research materials.[2] Therefore, this section describes the *intended* and *necessary* analyses as defined by the project's methodology.[1, 1]

The EDA phase would have necessarily included the following analyses:

- **Time Series Decomposition:** A plot of the raw PM2.5 time series, decomposed into its constituent parts:
  - **Trend:** To observe the long-term (multi-year) movement of pollution levels.
  - **Seasonality:** To visually confirm the "pronounced seasonal air quality degradation" [1] and identify the "summer haze season" [1] peak, as described in the foundational literature.
  - **Residuals:** To examine the noise remaining after the trend and seasonality are removed.
- **Correlation Analysis:** A correlation heatmap (e.g., using seaborn.heatmap) to quantify the linear relationships between the input meteorological features (Temperature, Humidity, Wind Speed, etc.) and the PM2.5 target variable. This analysis would serve as a direct, preliminary test of the project's core hypothesis.
- **Seasonal Sub-plots:** Boxplots or line plots displaying the distribution of PM2.5 by month. This would visually confirm the findings of La-ong-muang et al. (2021) that pollution concentrations are highest in the summer and lowest during the rainy season.[1]

This EDA was not merely descriptive; it served as a crucial *diagnostic* step. The project's literature review [1] identifies a "critical flaw" in all tree-based machine learning models: their "inability to extrapolate," meaning they cannot predict values outside the range seen during training.[1] This flaw is only a significant problem if the time series exhibits a long-term trend (e.g., pollution gradually worsening year over year). The EDA's time-series decomposition plot would have been the tool to identify this trend. The subsequent selection of the "hybrid trend-residual modeling" strategy [1] strongly implies that this trend was, in fact, observed, making the EDA the diagnostic step that *necessitated* the project's most advanced modeling approach.

5. Feature Engineering

The most significant feature engineering step undertaken by the project was the sophisticated "hybrid trend-residual modeling" strategy. This approach was explicitly designed to overcome the "critical flaw" of tree-based models (like XGBoost) in a time-series context.[1]

- **Rationale:** As identified in the EDA, the PM2.5 time series likely contains a long-term trend. A standard XGBoost model, when trained on this data, would be incapable of forecasting a record-breaking pollution day (i.e., it can never predict a value higher than the highest value it saw in the training set).[1]

- **Methodology:** To mitigate this, the problem was decomposed into two distinct modeling tasks [1]:
  1. **Trend Model:** A simple LinearRegression model was trained to capture the long-term, secular trend. The single *feature* for this model was a time_index (e.g., an integer series 1, 2, 3...), and the *target* was the raw PM2.5 value.
  2. **Residual Model (XGBoost):** The powerful XGBoost model was then trained. The *features* for this model were the rich meteorological variables (Temperature, Humidity, etc.). However, the *target* variable was **not** the raw PM2.5 value. Instead, it was the **residual**, which is calculated as: residual = actual_PM2.5 - trend_prediction.

This hybrid approach is a powerful form of feature engineering that fundamentally *reframes the target variable*. The XGBoost model is no longer learning to predict the absolute PM2.5 level; it is learning to predict the *complex, non-linear, weather-driven fluctuations around the long-term trend*. This allows each model to excel at its specific task: the Linear Regression extrapolates the simple trend, while the XGBoost model focuses its power on the complex, short-term variations. The final forecast is the sum of the two models' outputs: final_forecast = trend_prediction + residual_prediction.[1]

While not explicitly detailed in the planning documents, the project's methodology of predicting t+1 (next day) based on "a sequence of past observations" [1] also implies the use of **lag features** (e.g., PM2.5_lag_1, temp_lag_1) as inputs to the residual model.

6. Data Transformation

As a final preprocessing step, all numerical features in the dataset were scaled.

- **Methodology:** The project plan specified scaling all numerical features "to a common range (e.g., using Min-Max scaling or Standardization)".[1]
- **Rationale:** The planning documents state this step is "crucial for many machine learning algorithms, including the baseline linear model".[1] This highlights a subtle but important methodological detail. Tree-based models, such as Random Forest and XGBoost, are generally invariant to monotonic feature scaling because they only care about split points. However, Linear Regression, which is often solved with gradient descent, is highly sensitive to the scale of its input features; features with large magnitudes (e.g., atmospheric pressure in **hectopascals**) can dominate the model's coefficients.

By scaling *all* features, the project created a single, unified preprocessing pipeline that could be fed to *all three* candidate models. This ensures that the Linear Regression baseline is not unfairly handicapped, allowing for a valid, "apples-to-apples" comparison of its performance against the more complex models.

Model Exploration

1. Model Selection

The project employed a comparative evaluation of three distinct machine learning models, representing a "crawl-walk-run" strategy of increasing complexity. This approach is a data science best practice, ensuring that any added model complexity is justified by a quantifiable improvement in predictive performance.[1, 1]

The three candidate models were:

1. **Linear Regression (The "Crawl" / Baseline):**
   - **Purpose:** To serve as a simple, interpretable performance baseline.[1]
   - **Justification:** This model is not expected to be the top performer. Its inclusion is methodologically crucial for two reasons: 1) It provides a direct performance comparison to the previous regional study by La-ong-muang et al. (2021), which also used Multiple Linear Regression, thus "grounding" the project's results within the existing literature.[1] 2) It establishes the performance *floor* that the more complex, non-linear models must surpass to justify their use.[1]
2. **Random Forest (The "Walk" / Robust Ensemble):**
   - **Purpose:** An advanced ensemble model known for its robustness to noise and its ability to capture complex non-linear relationships.[1, 1]
   - **Justification:** Random Forest, which operates by averaging the predictions of many individual decision trees (a process called bagging), is highly resistant to overfitting and requires less tuning than boosting models. It provides a strong, modern baseline for non-linear modeling and offers moderate interpretability through its "feature importance" metric.[1]
3. **XGBoost (The "Run" / Advanced Model):**
   - **Purpose:** The primary advanced model, selected for its "state-of-the-art" performance, high accuracy, and computational efficiency.[1, 1]
   - **Justification:** XGBoost (Extreme Gradient Boosting) is a state-of-the-art implementation of a gradient boosting algorithm. Unlike Random Forest's parallel trees, XGBoost builds trees sequentially, with each new tree correcting the errors of the last.[1] It includes built-in regularization (L1 and L2) to rigorously control overfitting.[1] Within this project's hybrid framework, it was specifically chosen to be the "residual" model, where its power could be focused on modeling the complex, weather-driven fluctuations.[1]

The trade-offs between these selected models are summarized in the table below.

Table 2: Comparative Evaluation of Selected Models [1]

Feature	**Linear Regression**	**Random Forest**	**XGBoost**
Model Type	Statistical / Linear	Ensemble (Bagging)	Ensemble (Boosting)
Core Principle	Fits a linear equation	Averages many independent trees	Sequentially adds trees to correct prior errors
Performance	Low (Baseline)	High	Very High / State-of-the-art
Overfitting Control	Low (low complexity)	Robust (by averaging)	High (Built-in L1/L2 Regularization)
Handles Non-Linearity	No	Yes	Yes (Highly Effective)
Role in Project	Baseline / Literature Comparison	Robust Non-Linear Baseline	Primary Advanced Model (for Residuals)

2. Model Training

The model development phase involved training the three candidate models on the "Versioned Training Dataset" and rigorously optimizing their performance.[1]

- **Validation Strategy: Chronological Splitting:**
  - The most critical aspect of training a time-series model is the validation strategy. For this project, a **chronological split** was the correct methodology. The dataset was planned to be split "using the first 80% of the timeline for training and the final 20% for testing".[1]
  - This approach is essential because standard k-fold cross-validation, which shuffles data randomly, is *invalid* for time-series forecasting. Shuffling would cause data leakage, allowing the model to "predict the past" using knowledge of the future. The

chronological split accurately simulates a real-world forecasting scenario, where the model must predict an unknown future based only on data from the past.[1]
    - While the initial Concept Note [1] mentioned "Cross Validation" [1], the chronological split described in the more detailed Literature Review [1] represents the correct, time-series-aware *implementation* of that principle.
- **Hyperparameter Tuning:**
    - To optimize the performance of the advanced Random Forest and XGBoost models, "extensive hyperparameter tuning" was a planned part of the methodology.[1] This process involves systematically searching for the optimal model settings (e.g., number of trees, learning rate, tree depth) to achieve the best performance on the validation dataset.[1]

3. Model Evaluation

The project's success was designed to be evaluated using a comprehensive, dual-metric approach, reflecting its dual output: a raw numerical prediction and a public-facing categorical forecast.[1]

- **1. Regression Task (Predicting PM2.5 Concentration):**
    - **Mean Absolute Error (MAE):** Measures the average magnitude of the errors in the units of the target (**micrograms per cubic meter**). It is highly interpretable (e.g., "the model is, on average, 5 **micrograms per cubic meter** off").
    - **Root Mean Squared Error (RMSE):** Also measured in the units of the target, but this metric penalizes larger errors more heavily than MAE.
- **2. Classification Task (Predicting AQI Category):**
    - **Accuracy:** The percentage of days the AQI category (e.g., "Good," "Moderate," "Unhealthy") was correctly predicted.
    - **F1-Score:** A more robust classification metric that provides a balance between Precision and Recall. This is particularly important if the classes are imbalanced (e.g., if "Hazardous" days are very rare).

This dual-metric set reflects a sophisticated understanding of the project's stakeholders. The regression metrics (MAE, RMSE) are essential for data scientists and researchers to evaluate the model's core physical accuracy. The classification metrics (Accuracy, F1-Score) are for the *end users* (students, local community), who need to know if they can trust a forecast of "Good" or "Unhealthy".[1]

- **Performance Results:**
    - **Note:** The final project report, presentation, and code notebook—which would contain the achieved performance metrics and final model evaluation

visualizations—were not accessible from the provided research materials.[2]
    - **Conclusion:** It is not possible to report the final MAE, RMSE, Accuracy, or F1-Score for the Linear Regression, Random Forest, or XGBoost models. Consequently, it is not possible to state which model was ultimately selected as the "Best Performing Model."
- **Intended Visualizations (Unavailable):**
    - **Prediction vs. Actual Plot:** A time-series line plot or scatter plot comparing the model's predicted PM2.5 values against the true PM2.5 values on the 20% test set.
    - **Confusion Matrix:** A matrix for the classification task, showing the counts of correct and incorrect predictions for each AQI category. This would be the primary tool for analyzing classification accuracy and F1-score.
    - **Feature Importance Plot:** A bar chart from the Random Forest or XGBoost model, illustrating which features (e.g., Humidity, Wind_Speed, PM2.5_lag_1) were most predictive.

4. Code Implementation

Relevant code snippets for both data preparation/feature engineering and model Exploration can be viewed via Capstone.ipynb .