Erasmus Mundus Joint Master

Artificial Intelligence for Sustainable Societies (AISS)

Applied Programming for Data Science

May 2025

Deniz Yener
a22407862@ulusofona.pt

Mohsen Hassan Nejad
a22407814@ulusofona.pt

Software Engineering Report

Abstract

This project is developed as part of the "Applied Programming" course in the Artificial Intelligence for Sustainable Societies Master's program. It explores air quality in Beijing from 2010 to 2014, focusing on predicting fine particulate matter (PM2.5) and how it evolves over time in relation to meteorological and seasonal factors. The dataset is a time series, with hourly readings of PM2.5 and weather conditions such as temperature, wind speed, and pressure.

Dataset: https://archive.ics.uci.edu/dataset/381/beijing+pm2+5+data

Contents

1 Introduction

1.1 Problem Statement

Coming from highly polluted cities like Tehran and Izmir, we've seen firsthand how hard it can be for people, especially families with children, the elderly, and those with health conditions, to know whether it's safe to go about their lives. In winter particularly, this uncertainty can have serious health consequences.

Our project is driven by the idea of making pollution information more accessible and visual. The aim is simple: **to let users check upcoming air quality in their city through a categorized graph that shows whether it's safe or not to be outside**. Behind that simple interface, we use machine learning to predict PM2.5 levels based on historical weather and pollution data, as well as feature-engineered variables.[11]

1.2 Main Goals and Objectives

Predictive Modelling: The primary goal of this project is to develop a machine learning system that can accurately predict PM2.5 levels in Beijing based on historical air quality and meteorological data. This system aims to provide accessible and visual pollution information to help users, particularly vulnerable populations like children, the elderly, and those with health conditions, make informed decisions about outdoor activities. [20]

Specific objectives include:

Predictive Modelling for Weather and Time Variables: RRebuilding ML models using exclusively meteorological and time variables, excluding lag, rolling-window, and VMD-volatility features. The goals are as follows:

1. Evaluate model performance using only weather and time variables.
2. Identify key features affecting pollution (e.g.wind direction, temperature, time of the day for the model.

Classification of Air Quality Levels: Classify air quality levels (e.g., Good, Moderate, Unhealthy) based on AQI standards, making predictions easier to interpret. [18, 7]

Feature Importance Analysis: Identify the most influential features (e.g., wind direction, temperature, time of day) to better understand what drives pollution variability.

Web Application Development: Create a user-friendly web application that presents forecasted air quality information in an accessible visual format.

1.3 Software Project Management

Team: Deniz Yener, Mohsen Hassan Nejad.

Workflow: The project was developed by a team of two contributors, Deniz Yener and Mohsen Hassan Nejad, as part of the Artificial Intelligence for Sustainable Societies (AISS) Master's program.

The workflow followed an iterative data science methodology:

- Problem definition and dataset selection
- Exploratory data analysis
- Data wrangling and cleaning
- Feature engineering
- Model development and evaluation
- Web application implementation

Risks: Careful attention was paid to proper time series train-test splitting and feature engineering to prevent information from the future from influencing predictions about the past. Model complexity was balanced against performance requirements to ensure the final application could provide timely predictions.

The PM2.5 target variable contained missing values that required appropriate handling strategies to maintain data integrity. Measures were taken to ensure the model would perform well on unseen data and not just the training dataset.

1.4 Project Workflow

The project followed a structured data science workflow consisting of the following key phases:

Exploratory Data Analysis (EDA): Uncovering patterns and trends in the data. [17]

- Initial data exploration and understanding
- Analysis of PM2.5 characteristics and distribution
- Comparison with Air Quality Index standards
- Identification of seasonality patterns
- Examination of relationships between numerical features

Data wrangling and cleaning: Addressing missing values and outliers to ensure data quality

- Handling of missing values

Feature engineering: Creating meaningful input variables to prepare the dataset for modeling.

- PM2.5 Outlier Detection and Flagging
- Categorical and Temporal Feature Engineering
- Creation of time-based features (day of week, month, season)
- Development of lagged features to capture temporal patterns

Machine learning modeling and evaluation: Training and assessing models to predict PM2.5, classify air quality, and identify key influencing factors.[16, 14]

- Defining the target variable and features
- Time Series data split
- Preprocessor Setup
- Pipeline Creation
- Hyperparameter tuning
- Performance comparison using appropriate metrics

Weather and Time Modeling: Training and assessing time and weather variables for the models to predict PM2.5 levels and identify key influencing factors.

- Defining the target variable and features
- Time Series data split for weather and time variables'
- Preprocessor Setup
- Pipeline Creation
- Hyperparameter tuning
- Cross-validation for robust evaluation
- Performance comparison using appropriate metrics for weather and time variables'

WebApp that uses our models: Creating a demo of a real-world application of our models.

- Design of the user interface
- Implementation of air prediction functionality [**?**]
- Visualization of air quality forecasts
- Deployment and testing

Conclusion and Limitations: Considerations and limitations of our project and the possible future steps.

1.5 Glossary of Terms

- **PM2.5**: Particulate matter with a diameter of 2.5 micrometers or less. These fine particles can penetrate deep into the lungs and even enter the bloodstream, posing significant health risks.
- **Air Quality Index (AQI)**: A standardized system for conveying air pollution levels to the public, typically categorizing air quality as Good, Moderate, Unhealthy for Sensitive Groups, Unhealthy, Very Unhealthy, or Hazardous.
- **Feature Engineering**: The process of creating new variables from existing data to improve model performance.
- **Lagged Features:** Variables created by shifting time series data backward in time, allowing models to learn from past values.
- **Rolled Features:** captures the volatility of pollution levels over the last day, a proxy for trend predictability.
- **pm2.5lag1** Yesterday's air still hangs around— the last-hour reading is the single best clue about the next hour.
- **ExtremePM2.5False** Indicates stable air quality; helps model avoid false alarms when pollution hasn't spiked recently.
- **Extreme PM2.5True** Highlights known spikes in pollution—reinforces caution when forecasting next steps.
- **WinDirV** North-south wind vector—captures directional dispersion of pollution plumes.
- **DewP** Dew point correlates with humidity; more moisture often means greater particle suspension.
- **pm2.5roll24std** Captures the volatility of pollution levels over the last day—a proxy for trend predictability.
- **WinDirU** East-west wind vector—complements 'WinDirV' for full 2D wind pattern impact.
- **pm2.5roll24mean** Averages out noise in hourly readings, providing a smoothed pollution trend baseline.
- **DewP**: Dew point temperature, the temperature at which air becomes saturated with water vapor.
- **Temp**: Ambient temperature measured in Celsius.
- **Press**: Atmospheric pressure.
- **WinDir**: Combined wind direction, categorized as NW (northwest), NE (northeast), SE (southeast), and CV (calm variable).
- **WindSpeed**: Cumulated wind speed.
- **HoursofSnow**: Binary indicator for snow (0 = No Snow, 1 = Snow).
- **HoursofRain**: Binary indicator for rain (0 = No Rain, 1 = Rain).

2 Software System Features and Requirements

The software system for the Beijing Air Quality Analysis project is designed to meet the critical need for predicting PM2.5 levels and delivering accessible air quality information. Our system aims to serve a diverse user range, from residents of Beijing to those in Tehran, providing an intuitive platform for visualizing air quality forecasts for the upcoming days. [19]

For residents and health-sensitive individuals, the system offers clear visualizations and predictions of PM2.5 levels. These features help users to make informed decisions regarding outdoor activities and potential exposure risks.

3 Project Resources

This project represents ongoing work in developing a comprehensive air quality prediction system. All project materials, including our complete collaborative notebook, source code, and additional resources, are available in our GitHub repository.

GitHub Repository The complete project, including all code, data preprocessing scripts, model implementations, and our collaborative Jupyter notebook, can be found at:

`https://github.com/TechNejad/ml-project_beijing-air-quality/tree/main`

Project Presentation Our detailed project presentation, covering methodology, results, and future directions, is available at:

`https://docs.google.com/presentation/d/1mSbRu8jtuZOiwOL07-CrmGrHxqbF5-_9c8YpH1RTzf8/edit?usp=drive_link`

Web Application Demo A video demonstration of our ongoing web application development can be viewed at:

`https://drive.google.com/file/d/1Bvj4HjrfFD-e8dgSwDNGaSzz7irqHGc_/view?usp=drive_link`

The web application aims to provide real-time air quality predictions and interactive visualizations based on the models developed in this study.

3.1 Functional Requirements

The functional requirements define the specific capabilities and services that the system must provide. Each requirement is assigned a unique identifier, priority level, and brief description.

ID	Requirement	Priority	Description
FR1	Data Preprocessing	High	The system preprocesses the Beijing PM2.5 dataset, handling missing values and preparing features for model training.
FR2	PM2.5 Level Prediction	High	The system predicts PM2.5 concentration levels based on historical data and meteorological features.
FR3	Air Quality Classification	High	The system classifies predicted PM2.5 levels according to Air Quality Index standards (Good, Moderate, Unhealthy).
FR4	Feature Importance Analysis	Medium	The system identifies and ranks features that most significantly influence PM2.5 levels.
FR5	Temporal Pattern Recognition	Medium	The system recognizes and accounts for temporal patterns (daily, weekly, seasonal) in PM2.5 levels.
FR6	Visualization Generation	High	The system generates visualizations of predicted air quality levels and historical patterns.
FR7	Web Interface Provision	High	The system provides a user-friendly web interface for accessing predictions and visualizations.
FR8	Model Performance Reporting	Medium	The system reports performance metrics of the predictive models.
FR9	Data Update Capability	Low	The system allows for updates with new air quality data as it becomes available.
FR10	Multiple Model Comparison	Low	The system enables comparison between different predictive models.

Functional Requirements for Air Quality Monitoring System

3.2 Interface Requirements

Web Application Interface: The web application interface we are employing utilizes Streamlit, a modern and intuitive framework for creating web applications with Python. This interface is accessible through standard web browsers and is designed to present predicted PM2.5 levels in a clear and visually engaging format.

By utilizing Streamlit's capabilities, the interface incorporates color coding that aligns with Air Quality Index (AQI) standards, providing users with an immediate understanding of air quality conditions. The responsive design has an accessible and user-friendly interface suitable for both desktop and mobile devices. [3]

Software Interface Requirements: For data processing, we utilized Python libraries for: Data manipulation (pandas, numpy) Machine learning (scikit-learn, scipy.stats, GridSearchCV) Visualization (matplotlib, seaborn)

3.3 Nonfunctional Requirements

- Project analysis and results
- Project model evaluation metric results
- Future work and research
- Web Application that uses our best model

4 Software Architecture and Construction

4.1 Datasets and Exploratory Data Analysis

The Beijing PM2.5 dataset from the UCI Machine Learning Repository forms the foundation of this project. This dataset contains hourly air quality readings and meteorological data collected from Beijing between January 1, 2010, and December 31, 2014, comprising 43,824 hourly records.

The dataset was selected for several key reasons:

1. It provides continuous hourly data over multiple years, ideal for time series analysis and prediction
2. It combines both pollution measurements (PM2.5) and meteorological features, enabling comprehensive modeling
3. It represents a real-world urban environment with significant pollution challenges

The dataset includes the following variables:

Variable	**Description**	**Type**
No	Row index	Discrete Numerical
year	Year of record (2010-2014)	Discrete Numerical
month	Month of record (1-12)	Discrete Numerical
day	Day of record (1-31)	Discrete Numerical
hour	Hour of record (0-23)	Discrete Numerical
pm2.5	PM2.5 concentration ($\mu g/m^3$)	Continuous Numerical
DEWP	Dew point temperature (°C)	Continuous Numerical
TEMP	Temperature (°C)	Continuous Numerical
PRES	Atmospheric pressure (hPa)	Continuous Numerical
cbwd	Combined wind direction (NW, NE, SE, CV)	Categorical
Iws	Cumulated wind speed (m/s)	Continuous Numerical
Is	Snow occurrence (0=No, 1=Yes)	Binary
Ir	Rain occurrence (0=No, 1=Yes)	Binary

Target Variable, PM2.5, Characteristics:

PM2.5 values are with 83.2 percent below 200µg/m^3. Extreme values above 600µg/m^3 are **very rare** (0.05) but impactful. This complexity suggests the need for **special treatment** of outliers and possibly **log transformation for modeling.**

Air Quality Index (AQI) and Pollution Patterns: **The AQI comparisons follow official air quality standards from:**

- US EPA (Environmental Protection Agency), which defines PM2.5 thresholds for AQI categories based on health impacts [EPA AQI Factsheet, 2012].
- Chinese HJ 633-2012, issued by China's Ministry of Ecology and Environment, outlining PM2.5 breakpoints and pollution severity categories [HJ 633-2012 official regulation]. [10, 1, 2]
- World Health Organization (WHO) guidelines, revised in 2021, which recommend a daily average PM2.5 limit of 15 µg/m^3 and an annual average of 5 µg/m^3, based solely on health risk evidence.

4.2 Methods and algorithms

The development of the air quality prediction system involved a comprehensive set of data science methods and techniques:

1. **Data Preprocessing Techniques**: [13] The only missing value belonged to our target variable, PM2.5, where significant data gaps exist before 2012, with over 1300 missing hours. After 2013, missing values are rare and well-distributed. Our first reaction was to drop pre-2012 data because it contained noticeable gaps. After several discussions and an audit, we realized that the total hours before 2012: 17520, and the missing PM2.5 hours before 2012: 1397. Percentage of missing data: 7.97 percent. Thus, we decided to impute the missing values to recover the missing data gaps meaningfully.

   We applied a multi-step strategy balancing temporal structure and data availability: [22]

   (a) Drop fully-missing segments (e.g., the first 24 hours of 2010).

   (b) Interpolate missing values within each day, separately for morning and evening periods. To maintain the natural daily patterns and prevent errors from interpolating across different days, we split the PM2.5 Data into two parts for interpolation:

   Morning hours: From 00:00 to 14:00 **Evening hours:** From 15:00 to 23:00

   This approach helps maintain the structure within each day and avoids incorrect trends that might arise from interpolating across day boundaries.

   (c) Fall back to median imputation per month/hour/year where interpolation was not possible.

   (d) Preserve the original values and construct a unified 'PM2.5-filled' variable for downstream modeling.

   These steps aim to maximize data coverage while respecting diurnal and seasonal patterns in PM2.5 levels.

2. **Feature Engineering**:

   Several derived features were created to enhance model performance:

   - Temporal features: day of week, month, season, hour of day
   - Lagged features: previous 1, 3, 6, 12, and 24 hours of PM2.5 values
   - Rolling statistics: moving averages and standard deviations over various windows
   - Cyclical encoding of time features using sine and cosine transformations

3. **Outlier Detection And Temporal Trends**:

   In this section, we apply two outlier detection techniques, namely IQR and Z-score, on key environmental variables such as 'PM2.5', 'Dew Point', 'Temperature', 'Pressure', and 'Wind Speed'. This helps us identify anomalies in the dataset and understand the overall distribution shape before feature normalization or modeling.

   - IQR Method: Flags extreme values that fall outside 1.5× the interquartile range.
   - Z-score Method: Detects statistical outliers based on deviation from the mean.

   To handle extreme WindSpeed values, we apply a winsorization technique by capping all values above the **99th percentile (269.12 m/s)**.[9]

   (a) A new column 'WindSpeed-Winsorized' is created.
   (b) All values above the threshold are replaced with **269.12**, maintaining the structure while dampening extreme distortions.

   We thereby reduced the influence of implausible spikes on downstream analysis while keeping the time-series structure intact.

4. **Variational Mode Decomposition (VMD)**: To better understand the temporal structure of pollution spikes, we apply Variational Mode Decomposition (VMD). VMD is a signal processing technique that decomposes a complex time series into a set of **oscillatory components (modes)**, each representing different frequency patterns.

   VMD helps us:

   - Isolate high-frequency spikes (e.g. sudden smog events).
   - Separate baseline seasonal patterns from volatile behavior.
   - Identify structurally distinct "pollution regimes" in the data.

   Feature importance is a crucial concept in machine learning, particularly in tree-based models. It refers to techniques that assign a score to input features based on their usefulness in predicting a target variable.

   This process helps us distinguish between noise and meaningful patterns, and **isolate spikes as separate signal components**. We now have two distinct ways of representing pollution extremes in the data:

   (a) **Extreme-PM2.5** a value-based flag (e.g., PM2.5 $>$ 700 µg/m^3)
   (b) **Extreme-Event-VMD** a structure-based flag derived from VMD Mode 1 dynamics

Each captures a different kind of signal, one based on pollutant level, the other on pattern behavior. This layered experimentation helps assess whether temporal structure (captured by VMD) can enhance prediction beyond raw concentration spikes. Which we will test in the feature importance section.

5. **Categorical and Temporal Feature Engineering:**

   In this section we explore rare weather events of 'HoursOfRain' and 'HoursOfSnow' and wind direction transformation.

   (a) A rolling feature transforms **a single-hour signal into a cumulative one**, tracking **recent conditions over time**.

   (b) This gives the model richer temporal context and can reveal **slow-building effects** that isolated hourly values may miss.

   Although 'HoursOfRain' and 'HoursOfSnow' appear weak based on distribution and correlation, **exploratory modeling** shows that rolling versions can add subtle but real predictive value.

6. **Transforming Wind Direction into Continuous Features using Sine and Cosine:** Wind direction initially appears as categorical labels to accurately represent

   wind direction, we can convert the directions into two continuous numbers using sine and cosine functions.

   This method **preserves the circular nature of wind direction** and shows how different directions relate to each other. Here is how it works: These transformations yield two continuous features, capturing directional data effectively:

   Before applying sine and cosine transformations, we convert categorical wind directions to numerical degrees following the **Standard Meteorological Convention**.

Most trigonometric functions in programming languages expect input in radians. We need to convert the wind direction from degrees to radians:

$$radians = degrees \times \left(\frac{\pi}{180}\right) \tag{1}$$

the following step is to calculate sine and cosine values for each wind direction value (in radians).

- 'WindDir-sin:' captures East-West variations.
- 'WindDir-cos:' captures North-South variations.

Benefits of this Approach:

Preserves **circular relationships between directions**. Avoids issues with directional wrap-around (e.g., 0° = 360°). Facilitates improved performance in machine learning models by enabling better capture of directional effects on PM2.5.

In time series forecasting, for environmental variables like PM2.5 concentration, it is crucial to account for **temporal dependencies**: the idea that the current time period is influenced by **previous periods**.

1. **Lag Features**: These represent the PM2.5 value from previous time period such as 1, 2, 3, 6, 12, 24 hours ago. Lag features help the model train on past values by introducing previous observations as new input variables.
2. **Rolling Features**: Such as rolling means and rolling standard deviations, summarize the recent history of the series, helping the model understand local trends or volatility.

The ' pm2.5-lag' variable creates separate columns that consider how the pollution levels were 1,2,3,6,12, and 24 hours before, to provide information about how pollution levels have evolved over time.

We will test and compare their importance and use them later in our modeling stage. The 'pm2.5-roll24mean ' variable enables the model to learn about the 24-hour trends, local fluctuations, and volatility in pollution levels.

Together, 'pm2.5-lag' and 'pm2.5-roll24' window features enhance the model's ability to learn the temporal structures and dynamic behavior of PM2.5 concentrations, ultimately improving the robustness and accuracy of forecasts.

Shifting the "Extreme-Event-VMD" to keep the model honest:

Our previously created "Extreme-Event-VMD" variable tags hours when PM2.5 jumps wildly using a true/false boolean. Why shift it: The 'Extreme-Event-VMD' flag reflects the current volatility of PM2.5, which is directly related to the value we are trying to predict.

This creates data leakage, where the model has access to "future" knowledge during training, leading to artificially high performance. The solution is to move the flag **one time-step into the past**. Thereby, the model now sees only **yesterday's** volatility when forecasting today, thus we preserve the real-world causality.

5 Experimenting with Models

1. **Python libraries** [12]

Listing 1: Python imports for machine learning models

```
# Data Processing and Pipeline Tools
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer

# Data Pre-processing
from sklearn.preprocessing import StandardScaler, OneHotEncoder

# Model Selection and Validation
from sklearn.model_selection import train_test_split,
GridSearchCV, TimeSeriesSplit

# Regression Models
from sklearn.linear_model import LinearRegression,
Ridge, Lasso
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
import xgboost as xgb

# Model Evaluation Metrics
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score, make_scorer

# Utilities
import joblib
import warnings
warnings.filterwarnings('ignore')
```

2. **Defining the Target Variable and Features**
   - Our initial step for building effective models is defining our target variable and predictor features.
   - Our goal is to predict the concentration of PM2.5 particulate matter, which is our dependent variable.
3. **Time Series Data Splitting**

   In **time series modeling**, preserving chronological integrity is crucial for reliable and accurate predictions. Unlike conventional machine learning methods, random splits cannot be applied due to **temporal dependencies.**

   (a) Chronological Splitting: We partitioned the dataset sequentially into training and testing subsets based on chronological order. This approach prevents future information from accidentally influencing the model training phase. By doing so we are maintaining the integrity of our temporal data.

(b) Selection of Split Date: The split boundary is set at **"January 1, 2014."** This date provides a sufficiently large and recent test set, which allows us to evaluate seasonal patterns and trends much better and robustly evaluate our model's reliable performance.

4. **Preprocessor Setup**

   In this section, we set up a **data preprocessing pipeline** to prepare our data for analysis. This pipeline applies specific transformations to each column, using a **ColumnTransformer** to ensure each type of data is processed appropriately for accurate model interpretation. Our preprocessor acts as the smart kitchen, and ColumnTransformer applies StandardScaling for every number and applies One-Hot-Encoding for every label data. As a result, we will have a neat numeric table where the model handles apples with apples.

   Numerical Features: We are applying a standard scaling method to normalize the numerical features. This is a necessary step for model convergence and improving the stability of our predictions.

   Categorical Features: We are encoding the categorical features to change them into numerical representations. This way, the categorical features will be included in model training.

   The preprocessing pipeline learns transformations from the training dataset and applies these learned parameters to the test dataset. We are then able to create an unbiased model evaluation and preserve the temporality of the data and prevent data leakage.

5. **Model Pipelines**

   In this section, we combine our preprocessing steps from 3.3 and use them to create individual pipelines for each predictive model;

   - preprocessor → **RandomForest**

   - preprocessor → **XGBoost**

   - preprocessor → **SVR**

   - preprocessor → **KNN**

6. **Hyperparameter tuning** [21] In time series, the challenge is **predicting future values** without knowing the actual outcomes until they occur. To address this, we used historical data for tuning, tweaking hyperparameters to increase predictive accuracy. We employed **'GridSearchCV'** and **'TimeSeriesSplit'**, tailored for temporal data, to ensure robust model performance. [14, 15]

6 Weather and Time Modeling

In this section we reconstructed our modeling process from previous section but now purely using **meteorological signals** such as temperature, humidity, wind, pressure, etc. leaving out the lag, rolling-window, and VMD-volatility features we engineered in §3.

Goal is to build and evaluate the same four models (RF, XGB, SVR, KNN) on weather-only inputs to see **how far ecology-only can take us.**

Takeaway:

1. Which weather variables standout most as predictors.
2. A clean performance baseline for later comparison.

The steps taken in this section and their explanations more or less mirror section 3.

7 Project Analysis and Results

7.1 Project model evaluation

We evaluated each of our tuned models, Random Forest, XGBoost, SVR, and KNN, against the test dataset (‘Xtest‘, ‘ytest‘) to have an unbiased estimate on future, unseen data.[6, 5]

We employed the standard metrics to evaluate our model’s performance:

1. **Mean Absolute Error (MAE):** Quantifies average prediction error magnitude, irrespective of direction.
2. **Mean Squared Error (MSE):** Emphasizes larger errors by squaring deviations, penalizing substantial mispredictions.
3. **Root Mean Squared Error (RMSE):** Provides error magnitude in the original units, facilitating interpretability.
4. **Coefficient of Determination (R^2):** Indicates the proportion of variance explained by the model, reflecting how well the model captures underlying data patterns.

7.2 Project model evaluation metric results

Systematic evaluation of machine learning models is super important for accurately predicting their future performance and having reliable and well interpretable predictions. We have encountered a very common issue in time series forecasting, especially with tree-based models like **Random Forest and XGBoost**. When we included lagged rolling hour features for visualising feature importance, the plots show that pm2.5lag1 (and potentially pm2.5lag2, pm2.5lag3) completely dominates the feature importance, pushing all other **features’ importances to near zero, making them visually indistinguishable.**

Due to strong Autocorrelation, the current **PM2.5 value is extremely correlated with the PM2.5 value from the previous hour or day.** The model finds this relationship very easy and powerful to exploit. Tree-based models are greedy. They will pick the feature that provides the biggest reduction in error (or impurity) at each split. When pm2.5lag1 offers such a strong signal, the model will prioritize it heavily, often making subsequent splits on other features less impactful in terms of overall importance.[4]

8 Conclusion and Future Remarks

8.1 Limitations

Potential for Data Leakage: We have carefully created lag and rolling features to avoid data leakage; however, due to the preprocessing pipeline's nature, subtle leakage through improperly handled shifted features remains a potential risk, possibly affecting the result of performance metrics.

Computational Efficiency: Hyperparameter tuning using GridSearchCV and Time Series Splits significantly increased computational demands. Affecting scalability and practical applicability for real-time scenarios.

External Factors: Our current model does not explicitly consider additional pollution sources, such as industrial emissions, traffic density, or regional holidays, celebrations, etc. This is a limiting factor for predicting PM2.5 levels during unusual pollution events.

8.2 Future work and research

Extended Feature Set: Explore additional environmental, socio-economic, and industrial emission features such as satellite-derived Aerosol Optical Depth (AOD), traffic volume, and industrial activity levels to enhance predictive accuracy and robustness.

Deep Learning Approaches: Investigate deep learning methods such as LSTM and Transformer-based models, capable of capturing long-term dependencies and complex temporal patterns more effectively, potentially boosting predictive accuracy further.

Spatial-Temporal Modeling: Expand the project to incorporate multiple locations across Beijing, developing spatial-temporal prediction models capable of capturing regional variations and interdependencies in air pollution.

8.3 Conclusion

In this project, we aimed to accurately forecast PM2.5 particulate matter concentrations in Beijing by leveraging advanced machine learning techniques. Utilizing air pollutant measurements obtained from monitoring stations across the city, we implemented four distinct machine learning algorithms: **Random Forest, XGBoost, Support Vector Regression (SVR), and K-Nearest Neighbor (KNN)**. Of these approaches, the **RandomForest** model proved to be particularly effective, delivering robust and reliable predictive performance.

The study incorporated various data inputs, including meteorological parameters, traffic-related data, and pollutant measurements from nearby monitoring stations, covering the period from January 1, 2010, to December 31, 2014. A comprehensive feature engineering approach—**integrating lag features, rolling statistical measures, and seasonality indicators**—was crucial in enhancing model performance.

The findings clearly indicated that pollutant data from preceding hours significantly influence the accuracy of future PM2.5 predictions. Additionally, we assessed whether meteorological variables alone could provide adequate predictive capability for pollutant concentrations, independent of engineered lag and rolling features. Our analysis revealed that weather variables, when used exclusively, lack sufficient predictive power, highlighting the indispensable role of feature-engineered variables.

9 References

References

[1] AQI scale comparison – AQICN. (2015, March 20). A comparison of worldwide Air Quality Scales – Part 1. AQICN. https://aqicn.org/faq/2015-03-20/a-comparison-of-worldwide-air-quality-scales-part-1/

[2] Bulut, A. (2022). Predict Pollution of Beijing with Regression Model [Kaggle notebook]. Kaggle. https://www.kaggle.com/code/alibulut1/predict-pollution-of-beijing-with-regression-model

[3] Chinese AQI standard (HJ 633-2012) – Ministry of Environmental Protection of the People's Republic of China. (2012). Environmental air quality index (AQI) technical regulation – Trial (HJ 633-2012). China Environmental Science Press.

[4] CrossValidated (Stack Exchange). (n.d.). *How can I generate a time series with autocorrelation at lags other than 1?* CrossValidated. Retrieved May 25, 2025, from https://stats.stackexchange.com/questions/410534/how-can-i-generate-a-time-series-with-autocorrelation-at-lags-other-than-1

[5] DataCamp. (n.d.). *Winsorized mean tutorial.* Retrieved May 25, 2025, from https://www.datacamp.com/tutorial/winsorized-mean

[6] DataScience Stack Exchange. (n.d.). Time-series hyper-parameter tuning. DataScience Stack Exchange. Retrieved May 25, 2025, from https://datascience.stackexchange.com/questions/106346/time-series-hyperparameter-tuning

[7] EPA-style AQI formula discussion – PurpleAir Community. (2019, June 18). How to calculate the EPA PM2.5 AQI [Online forum post]. PurpleAir Community. https://community.purpleair.com/t/how-to-calculate-the-epa-pm2-5-aqi/87

[8] Esri. (n.d.). Understanding outliers in time-series analysis. In ArcGIS Pro tool reference: Space-time pattern mining. Esri. Retrieved May 25, 2025, from https://pro.arcgis.com/en/pro-app/latest/tool-reference/space-time-pattern-mining/understanding-outliers-in-time-series-analysis.htm

[9] GeeksforGeeks. (n.d.). *Winsorization.* Retrieved May 25, 2025, from https://www.geeksforgeeks.org/winsorization/

[10] Li, Q., Zhang, H., Cai, X., Song, Y., & Zhu, T. (2021). *The impacts of the atmospheric boundary layer on regional haze in North China.* **NPJ Climate and Atmospheric Science, 4**, Article 9. https://doi.org/10.1038/s41612-021-00165-y

[11] Mahmud, S., Ridi, T. B. I., Miah, M. S., Sarower, F., & Elahee, S. (2022). Implementing machine learning algorithms to predict particulate matter (PM 2.5): A case study in the Paso del Norte region. Atmosphere, 13(12), 2100. https://doi.org/10.3390/atmos13122100

[12] mlcourse.ai. (n.d.). Gradient boosting. Retrieved May 26, 2025, from https://mlcourse.ai/book/topic10/topic10_gradient_boosting.html

[13] nnjjpp. (2025). Pipelines for preprocessing: A tutorial [Kaggle notebook]. Kaggle. https://www.kaggle.com/code/nnjjpp/pipelines-for-preprocessing-a-tutorial

[14] Scikit-learn Developers. (n.d.). Grid search and randomized search for hyper-parameter tuning. In scikit-learn: Machine learning in Python (Version 1.x). Retrieved May 25, 2025, from https://scikit-learn.org/stable/modules/grid_search.html

[15] Scikit-learn Developers. (n.d.). Model evaluation: Quantifying the quality of predictions. In scikit-learn: Machine learning in Python. Retrieved May 26, 2025, from https://scikit-learn.org/stable/getting_started.html#model-evaluation

[16] Scikit-learn Developers. (n.d.). sklearn.model_selection.TimeSeriesSplit (Version 1.x). In scikit-learn: Machine learning in Python. Retrieved May 25, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html

[17] Tan, A. (2022, February 12). Forecasting_Air_Pollution (Version archived) [GitHub repository]. GitHub. https://github.com/at-tan/Forecasting_Air_Pollution

[18] U.S. EPA break-points – U.S. Environmental Protection Agency. (2025, May 21). AQI breakpoints. Air Quality System. https://aqs.epa.gov/aqsweb/documents/codetables/aqi_breakpoints.html

[19] Wang, L., Liu, J., Gao, Z., Li, Y., Huang, M., Fan, S., Zhang, X., Yang, Y., Miao, S., Zou, H., Sun, Y., Chen, Y., & Yang, T. (2019). *Vertical observations of the atmospheric boundary layer structure over Beijing urban area during air pollution episodes.* **Atmospheric Chemistry and Physics, 19**, 6949–6967. https://doi.org/10.5194/acp-19-6949-2019

[20] Yin, P.-Y. (2025). A review on $PM_{2.5}$ sources, mass prediction, and association analysis: Research opportunities and challenges. Sustainability, 17(3), 1101. https://doi.org/10.3390/su17031101

[21] Zhang, X. (2019). time_series_forecasting_pytorch [GitHub repository]. GitHub. https://github.com/zhangxu0307/time_series_forecasting_pytorch

[22] Zhao, X. J., Zhang, X. L., Xu, X. F., Xu, J., Meng, W., & Pu, W. W. (2009). *Seasonal and diurnal variations of ambient $PM_{2.5}$ concentration in urban and rural environments in Beijing.* **Atmospheric Environment, 43**, 2893–2900. https://doi.org/10.1016/j.atmosenv.2009.03.009