Predicting Air Quality Severity for Proactive Health and Equipment Protection in Hospital Environments: A Machine Learning Approach

1^{st} Amit K. Karn
MSc Data Science and Computational Intelligence
Softwarica College of IT and E-commerce (Coventry University)
Kathmandu, Nepal
240754@softwarica.edu.np

***Abstract*—Air pollution in urban areas, especially in geographically locked regions like Kathmandu, poses a critical threat to its inhabitants' health and presents a significant risk to vulnerable patients and sensitive infrastructure within healthcare facilities. This report introduces a machine learning-driven approach to predict air quality severity in Kathmandu, specifically highlighting its application within hospital environments for informed proactive safeguarding of public health and sensitive healthcare equipment by integrating an embedded meteorological sensor with IoT for real-time air quality predictions. Particulate Matter (PM2.5) consists of a mixture of tiny particles suspended in the air, originating from various sources such as construction, forest fires, and vehicular emissions. These particles have been reportedly linked to severe respiratory diseases, including asthma, bronchitis, and other chronic obstructive pulmonary diseases (COPD). The methodology encompasses the analysis of raw meteorological and air quality data from the Kathmandu Valley, collected from publicly available repositories and recorded via several ground stations within the city. The collected and preprocessed data is then utilized for training various machine learning models, including Logistic Regression, Random Forest, and eXtreme Gradient Boost (XGBoost). Throughout all classes, both Random Forest and XGBoost consistently outperformed Logistic Regression. The weighted average accuracy for both Random Forest and XGBoost was found to be 73**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Air pollution is one of the most common yet critical factors that is responsible for the ongoing decline in the global population's health. Developing cities like Kathmandu Valley have consistently ranked among the cities with the worst air quality globally, as a consequence of rapid urbanization and various local factors [1]. The primary cause of this persistent pollution in Nepal, and consequently in Kathmandu, include rapid urbanization, vehicular emissions, brick kilns, and transboundary pollution [2]. Additionally, the poor waste and sewage management can lead to frequent increase in the concentrations of harmful particulate matter and other pollutants in the city, which often gets trapped close to the land along with the cold air due to the topographical structure of the valley [3]. These particulate matters, primarily PM2.5 have reportedly been linked with harmful effects on the public health, specially with respiratory and cardiovascular systems [4]. The annual PM2.5 levels of Kathmandu have been recorded at five times the WHO safe limit, which reduces an average of 4 years from the resident's life span. The city is frequently ranked among the world's most polluted cities - for instance, Kathmandu experienced unhealthy air quality on 75 out of 90 days [5]. Apart from public health, air pollution also tends to damage sensitive hospital equipment which can lead to faulty readings and expensive maintenance costs [6]. Thus, to proactively address these challenges, specially within the critical hospital environments, the need for accurate and timely prediction of severe air quality has been growing. Machine learning techniques can help in analyzing the complex meteorological data against the observed PM2.5 concentrations and identifying the key contributing factors of severe air pollution [4], [15]. Therefore, this project aims to develop a reliable machine learning approach to predict the severity of air quality within the hospital environments, facilitating timely interventions for informed decision-making and prevention of risks to public health and sensitive equipment damage.

II. Literature Review

A systematic literature review was performed to identify and analyze the recent articles and research works related to effects of air quality in healthcare facilities and usage of different modern machine learning techniques to address the issue.

The Oizom article, "Hospital Air Quality: Sources, Risks & Control Measures," sheds light on how the air inhaled by the patients and the staff of a hospital is susceptible to pollution from both the indoor as well as outdoor conditions within a hospital's vicinity. The article also goes through various sources of these contaminants that could potentially originate within a hospital such as smoke and chemical byproducts released from anesthesia and surgical procedures, practices from routine cleaning and disinfection, along with raw materials for infrastructure development. All these collectively release volatile organic compounds (VOCs) into the hospital's air. The pollutants that are commonly found in hospitals could range from Carbon Dioxide (CO2) and several VOCs to biological contaminants like bioaerosols and extremely small ultrafine particles (UFPs) with the traces of toxic gases, formaldehyde, chlorine-based compounds from cleaning agents and heavy metals. The article then also addresses the several obvious outdoor contributors to the hospital's poor air quality, including emissions from vehicles, nearby construction activities, pollutants originating from nearby industrial areas, and the presence of parking facilities. If not ventilated properly, these external pollutants could easily penetrate the hospital area, further degrading the air quality. Exposure to these pollutants are associated with significant health risks, which is more dangerous for vulnerable populations like patients suffering from respiratory conditions like asthma, chronic obstructive pulmonary disease (COPD), along with those with weaker immune systems. Poor air quality can thus prolong hospital stays, or worse, increase mortality risks. Apart from patients, the article also implies the potential risks of poor air quality to sensitive hospital equipment as they rely on clean air to function properly, but prolonged exposure to excessive pollutants can cause malfunctions disrupting hospital operations [6].

III. Problem and Dataset

A. Air Pollution in Hospital Environments

Air pollution can be defined as the prolonged stay of one or more contaminants in the atmosphere, such as smoke, gas, dust, mist, odour, vapor or fumes in such a quantity that can potentially be harmful to human health, primarily related to respiratory [13]. According to the World Health Organization (WHO), air pollution was the second leading cause of noncommunicable diseases (NCDs) after tobacco, contributing to almost 85% of the 6.7 million total deaths in 2019. These NCDs include asthma, chronic obstructive pulmonary disease (COPD), stroke ischemic heart disease, lung cancer, and diabetes [7]. People react differently to air pollution. Children and older people along with those with pre-existing health conditions are more sensitive to the impacts of air pollution. Additionally, individuals belonging to the most deprived socioeconomic groups with little to no access to sophisticated medical care commonly suffer from poorer health, exacerbating their vulnerability [8].

Most hospitals these days require substantial energy consumption to operate. It's a necessity for hospitals to operate 24 hours a day, and 7 days a week to provide vital medical care within an urban society. While in operation, they need all their facilities such as air conditioning, waste management, imaging equipment, disease control and temperature adjustment mechanisms to be in working condition [10]. As a result hospitals consume a disproportionate amount of energy when compared to smaller-sized commercial or residential buildings [9]. These indoor factors contribute significantly to poor air quality in hospitals and need to be mitigated by following the official guidelines for building design, building operation, installing High-Efficiency HVAC systems, utilizing natural ventilation strategies, installing insulation in walls, air sealing, incorporating passive solar design and installing heat recovery ventilation systems (HRVs) [11].

In addition to the aforementioned indoor factors, several outdoor sources also heavily contribute to the degradation of hospitals' air quality. In a 2018 study conducted by the Department of Environmental and Occupational Health at Kung University, Taiwan, researchers determined that fine and coarse particles originating from outdoor sources were found to be the major contributors to the hospital's poor air quality, regardless of air conditioning type and working area [12]. While numerous toxins that are contaminated through air can harm our health in one way or another, certain distinct pollutants have been strongly linked to causing serious health issues such as particulate matter (PM), carbon monoxide (CO), nitrogen dioxide (NO2), sulfur dioxide (SO2), and ozone (O3). Even among these, fine particulate matter is the most significant source of health risks because of their ability to penetrate deep into lungs, enter the bloodstream and travel to various organs [13].

B. Particulate Matter (PM)

The World Health Organization (WHO) defines particulate matter as the "particles in the air, including dust, dirt, soot, smoke, and liquid droplets." These tiny particles, often originating from sources like vehicles burning fossil fuels or power plants using coal, are really small - usually less than 10 micrometers (PM10) or even tinier, less than 2.5 micrometers (PM2.5). At about 1/30th of a human hair, these particulate matters can easily penetrate deep into lungs and bloodstream which makes it so dangerous [7].

This paper aims to classify the concentration of PM2.5 into standard AQI categories denoting the severity of the air quality. The air quality is considered as "severe" if the 24-hour AQI exceeds 150, corresponding to the border between "Unhealthy for Sensitive Groups" and "Unhealthy" levels (using U.S. AQICN conventions, roughly PM2.5 $>$ 55 µg/m³ for a 24 hour average). The threshold is chosen at a level where vulnerable patients can have adverse effects from even short term exposure, thus requiring hospitals to start taking protective measures. Also during periods of forest fire episodes, the AQI of Kathmandu increases well beyond 150, ranging from 235 to 265 at various stations, which falls in the "Very Unhealthy" to "Hazardous" range [14]. Such extreme levels would need advanced warning systems in the hospitals to safeguard its patients and sensitive equipment.

C. Data Sources and Collection

For this study, three types of data were separately collected - historical air quality data, historical meteorological data and historical holiday data.

1) Air Quality Data: The air quality in Nepal is monitored through several ground monitoring stations and is regulated by the Department of the Environment under the Ministry of Forests and Environment of Nepal. According to the air quality status report of Nepal 2020, there are 27 monitoring stations all over Nepal measuring the following significant parameters: PM1, PM2.5, PM10 and Total Suspended Particulates (TSP) [16]. These ground monitoring stations have many limitations such as limited coverage, spatial variability, and expensive and complex equipment [15]. Therefore, the data obtained from these sources comes with a lot of missing values and irregularities.

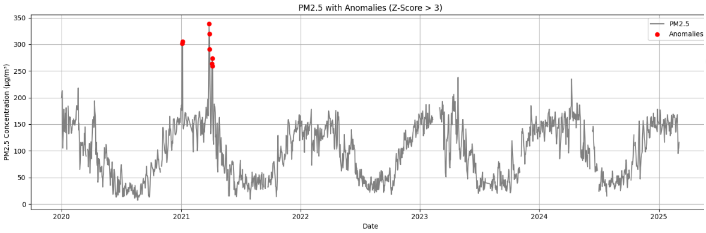


Fig. 1: PM2.5 TimeSeries January 2020 To April 2025

The time series plot of PM2.5 above shows its cyclic nature due to its dependence on the seasons and consequently weather. The anomalies like spikes in early January 2021 was when Kathmandu experienced unprecendented air quality deterioration. On the night of January 4, the Air Quality Index (AQI) at the US Embassy monitoring station exceeded 500 and remained above this level from 10PM until 5AM of the next morning. The sudden spikes were attributed to a combination of factors, including winter inversion, vehicular emissions, wildfires and cross-border industrial pollution [41], [42]. Similarly, in March 2021, Nepal faced extensive wildfires, with incidents reported in 73 districts. These fires were exacerbated by ususually dry winter, with precipitation levels significantly below average. The resulting smoke led to hazardous air quality levels across the country, including Kathmandu [43], [44].

2) Meteorological Data: Historical meteorological datasets are crucial while predicting the concentrations of particulate matters in air. Conventionally, these datasets are regulated by the Department of Hydrology and Meteorology, Babarmahal, Kathmandu, Nepal, and upon following a data request payment procedure, the dataset can be retrieved [17]. But for the purpose of this study, the historical meteorological data of Kathmandu was scrapped from a website called www.wunderground.com [18]. The data obtained was a 30 minutes interval meteorological data of Kathmandu from 2020-01-05 to 2025-04-25, which can aid greatly in developing a time series forecast model for predicting weather in a shorter time window. These data are the aggregated values from the data collected from 4 different weather stations' spots in Kathmandu - Tribhuvan International Airport, Nakhu Bazar, New Road, Thaiba and Patan Dhoka [18]. The key features of the meteorological dataset are temperature, dew point, humidity, wind, wind speed, pressure, and condition. Some of these factors like temperature, humidity, and wind are known to influence air quality [19], [20] - for instance, high winds can disperse or wash out pollutants, whereas temperature inversions and low wind speeds can exacerbate pollution accumulation. Below are the time series patterns of some of the fetures from January 2020 to April 2025:

3) Holiday Data: The impact of pollution cannot be predicted accurately if the holidays are not accounted for. Most of the major sources of air pollution like vehicular emissions

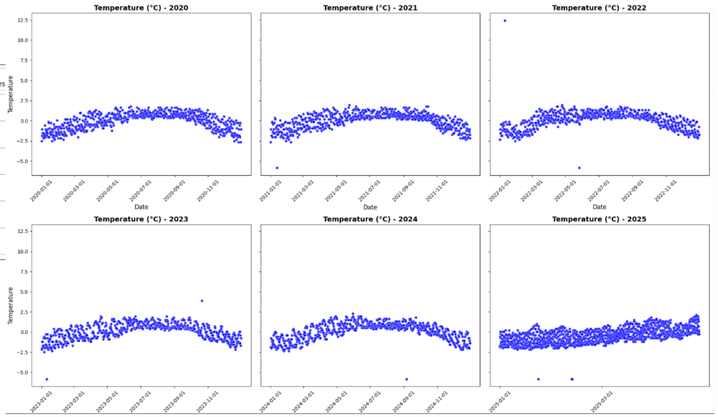


Fig. 2: Temperature TimeSeries January 2020 To April 2025

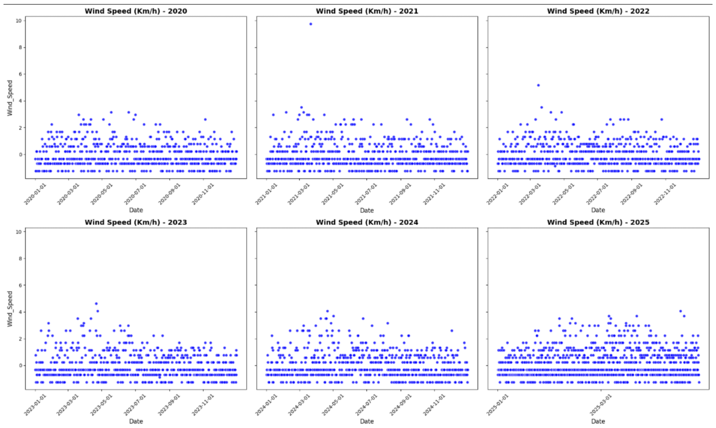


Fig. 3: Wind Speed TimeSeries January 2020 To April 2025

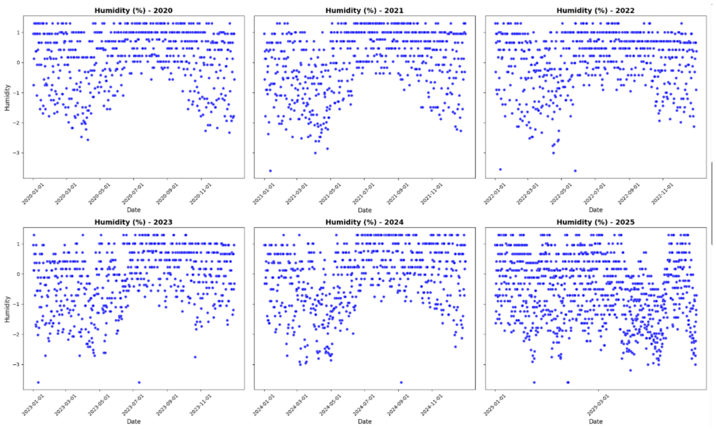


Fig. 4: Humidity TimeSeries January 2020 To April 2025

and industrial waste are drastically lessened depending on the type of holiday. Some holidays are optional, whereas some are mandatory. In Nepal most of the institutions give a compulsory weekend holiday on Saturday with some giving on Sundays too. The holiday dataset was scrapped from www.timeanddate.com website from 2020-01-01 to 2025-04-25 [21]. The columns obtained are date, name and type which was then put against the year. These holidays did not account for the weekend holidays, so it was later added to the data set with Sundays being marked as optional for Nepal's context.

All three of the datasets collected were compiled by coinciding their timestamp to get further insights. Each row in the final compiled dataset includes the timestamp (in local time) and the corresponding observed meteorological and air quality metrics (PM2.5) along with whether the day was a compulsory or optional holiday. The raw data contained numerous missing values and anomalies (perhaps due to sensor downtime or power outages) which required thorough data cleaning and preprocessing before moving onto the next process.

TABLE I: Summary of Dataset Characteristics

Characteristic	Details
Meteorological	**Key Features:** Timestamp, Temp., Dew Point, Humidity, Wind Dir./Speed, Pressure, Condition **Timeframe:** Jan 2020 – Apr 2025 **Source:** https://www.wunderground.com/ (scraped) **Coverage:** Aggregated from various locations in/around Kathmandu **Target:** PM2.5 Class
Air Quality	**Key Features:** Date, PM2.5, PM10 **Timeframe:** 2017 – Apr 2025 **Source:** https://aqicn.org/ (downloaded) **Coverage:** US Embassy Station in Kathmandu
Holidays	**Key Features:** Date, Name, Type **Timeframe:** Jan 2020 – Apr 2025 **Source:** https://www.timeanddate.com/ (scraped) **Coverage:** Kathmandu-specific

IV. Methods

To forecast the air quality severity, three machine learning models were trained on the compilation of historical air quality, meteorological, and holiday data - Logistic Regression (LR), Extreme Gradient Boosting (XGBoost), and Random Forest (RF). These models predicted one of the predefined PM2.5 level severity classes by learning from a wide range of features. The selection of these models was based on their ability to handle complex, non-linear relationships within environmental datasets and their proven efficacy in classification tasks.

A. Logistic Regression (LR)

Logistic Regression is one of the most widely used methods for binary or multi-class classification problems [22]. It is a supervised machine learning algorithm which is used to predict the probability of whether a given set of features belongs to a class or not [23]. In the context of this project, LR will be used to predict the predefined PM2.5 class based on the provided meteorological features such as temperature, humidity, wind speed, and pressure. It does so by using a sigmoid or softmax function to transform a combination of meteorological features into a probability score for each of the PM2.5 class, and finally results in the class with the highest predicted probability. Since the target variable in this context has more than three possible types, the softmax function is used in place of the sigmoid function [23]. For K classes, the softmax function will be:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

B. Random Forest (RF)

Random Forest is an ensemble learning method for classification and regression problems that works by constructing and merging multiple decision trees during training and resulting in the mode of the classes of individual trees [24]. Random forest is a robust algorithm known for its high accuracy and ability to handle complex, non-linear relationships in data, which makes it suitable for prediction of the PM2.5 class [25]. Random Forest can implicitly capture complex interactions for example - a combination of low humidity and temperature can lead to higher levels of pollution due to the pollutants getting trapped in the air.

C. Extreme Gradient Boosting (XGBoost)

XGBoost, an optimized version of Gradient Boosting, is a type of ensemble learning method designed for efficiency, speed, and high performance. Its strength lies in combining multiple weak models to form a stronger model. XGBoost uses Decision Trees as its base learner models, then combines them sequentially, which improves the model's performance. Each new model is trained to minimize the errors made by its predecessor, and this process is called boosting. Additionally, the built-in parallel processing leverages the quick training of very large datasets. Mathematically speaking, XGBoost is an iterative process that starts with an initial prediction. Then, each additional model is designed to minimize the errors of that prediction sequentially. A general formula to represent XGBoost is:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i)$$

Where:

- $\hat{y}_i$ is the final predicted value for the i-th data point
- K is the number of trees in the ensemble
- $f_k(x_i)$ represents the prediction of the k-th tree for the i-th data point.

V. Experimental Setup

A machine learning project's success hinges on a carefully designed experimental setup, including data manipulation, model training, and evaluation.

A. Data Acquisition and Initial Processing

The datasets used for this project have been compiled from 3 different raw data sources - meteorological dataset from www.wunderground.com [18], air quality dataset from 'www.aqicn.com' [26], and holiday dataset from www.timeanddate.com

[21]. The meteorological dataset is a collection of about 88,000 rows which contains the weather data such as timestamp, temperature, wind speed, and so on, recorded at every 30 minute interval. Next, the AQI data (about 1800 rows) contains date, PM2.5 and PM10 values. Similarly, the holiday dataset containing date, name, and type was also gathered. These datasets of csv file format were loaded as Pandas Dataframe into the notebook of Google Colaboratory, which is a free cloud-based service from Google which provides free access to Google's computing resources, including GPUs and TPUs, which makes it a powerful tool for high resource intensive tasks like machine learning, and data science [27].

B. Data Preprocessing

Data preprocessing is a critical phase of training machine learning models which ensures the quality and suitability of data before feeding into the models.The steps include:

1) **Handling Missing Values:** The very initial examination of the dataset revealed missing values in PM10 and wind features from the air quality and the meteorological datasets respectively. Missing values are a common problem in real-world datasets, especially in air quality studies [28] and when it comes to dealing with sources like weather and pollutants monitoring stations, primarily because of power outages and inadequate or costly maintenance in developing nations like Nepal [29]. The strategy used to handle missing wind values was using the backfill method. Backfilling is a common technique used to impute missing values in datasets of a time series nature, where previous data points are used to fill the gaps [30]. PM10 was simply dropped as it had no use for this project.
2) **Feature Engineering:** The acquired dataset contains quite a number of columns that's practically useless for this project. PM10 from the air quality dataset was removed because this project focuses on predicting PM2.5. The precipitation and wind gust fields from the meteorological dataset were removed as they had no variance whatsoever, possibly because of faulty sensors. The columns were calibrated to their standard units like temperatures from Fahrenheit to Celcius and pressures from inches in Mercury (inHg) to hectopascals (hPa). Meteorological parameters like temperature and wind speed are known to be highly important in the dispersion of pollutants. There were some anomalies in the meteorological data which were removed using upper and lower caps that occurred from the year 2020 to 2025. For example - a minimum of 1.6 °C and a maximum of 35.3 °C were reported on 18th December, 2020 and 15th June, 2024 respectively in Kathmandu [31], [32]. The categorical wind direction data was mapped into numerical degrees which enhances the model performance. Some features like 'is windy', 'wind sin', and 'wind cos' were derived which reduced the categorical columns significantly. The rows were then sorted in place by date. The 30-minute interval meteorological dataset was then aggregated into daily data in order to be merged with the daily air quality dataset by common dates. For aggregation, the mean of numerical columns, the maximum of binary columns, and the mode of the categorical column were taken. Finally, the merged dataset was compiled and the numeric PM2.5 was classified according to the following ranges:

TABLE II: PM2.5 Class Ranges (μg/m^{3})

PM2.5 Class	**Range (μg/m^{3})**
Low	0 - 35.4
Moderate	35.5 - 55.4
Unhealthy	55.5 - 150.4
Very Unhealthy	150.5 - 250.4
Hazardous	$>$ 250.4

VI. Model Training and Validation

Three classification models - Logistic Regression, Random Forest and XGBoost are trained to predict the PM2.5 class. These models underwent rigorous training along with a proper validation strategy to ensure their robustness.

- **Train-Test Split:** The dataset processed in the earlier stages was split into training, testing and validation sets, using a 70:15:15 ratio. The model is initially trained on the training set and tuned using the validation set until it starts giving optimal results. The model is then finally tested on the unseen test set.
- **Cross-Validation:** K-fold cross-validation strategy was implemented to mitigate the risk of overfitting of the models [34]. This helps in providing a more reliable estimate of model performance. This process involves partitioning the training data into predefined K subsets, then training the model on K-1 subsets, and finally validating on the remaining subset. The process is then repeated K times for each of the three models. The average performance score across all the folds gives a more balanced evaluation metric for the models.
- **Hyperparameter Tuning:** Each of the machine learning models has a set of hyperparameters, which can be adjusted to find an optimal configuration. A technique called grid search was used to systematically test a combination of predefined sets of hyperparameters for each model (LR, RF, XGB) to identify the optimal configuration for each model that gives the best performance on the validation set [35]. Below are the best hyperparameters for each of the models:

TABLE III: Best Hyperparameters

Model	**Best Parameters**
Logistic Regression	’classifier__C’: 0.1
Random Forest	’classifier__max_depth’: 10, ’classifier__n_estimators’: 200
XGBoost	’classifier__learning_rate’: 0.1, ’classifier__max_depth’: 3, ’classifier__n_estimators’: 200

A. Software and Hardware Environment

Machine learning training can be an extremely hardware resource-intensive process. Thus, the Google Colab environment was utilized to perform the entire project development, including data preprocessing, model training, and evaluation. Using the interactive capabilities of the Python programming language along with its rich set of libraries for machine learning such as `pandas`, `numpy`, `scikit-learn` (`sk-learn`), and `matplotlib`, the entire model development process was concluded.

B. Model Deployment

After completion of the model development process, the model file along with its final stages of feature engineering was dumped into a pickle file. This file was then used in a web app built using Pyhton's Streamlit library to further test the model's performance.

VII. Results

The evaluation of the trained machine learning models was primarily based on their ability to accurately predict the PM2.5 class from a given set of meteorological data points. The performance of each selected algorithm - Logistic Regression (LR), Random Forest (RF), and Extreme Gradient Boost (XGBoost) - was assessed using a group of appropriate metrics that reflects the classification of PM2.5 class prediction.

The key performance indicators for a classification task include Precision, Recall, F1-score, and Accuracy.

- **Precision:** The proportion of positive identifications that were actually true.
- **Recall:** The proportion of actual positives that were correctly identified.
- **F1-score:** The harmonic mean of the precision and recall, which provides a balanced measure especially useful when dealing with imbalanced classes.
- **Accuracy:** The measure that represents the overall correctness of the model's predictions.

Since the classes of air quality are typically imbalanced in nature (Hazardous and Unhealthy classes are typically less frequent than Good or Moderate ones), the F1-score is the most suitable candidate for evaluating air quality severity.

The results of the evaluation of the models trained on the preprocessed meteorological and air quality dataset are summarized below:

TABLE IV: Final Test Evaluation Metrics

Model	Metric	Value
Random Forest	Accuracy	0.77
	Macro Avg Precision	0.71
	Macro Avg Recall	0.69
	Macro Avg F1-Score	0.70
	Weighted Avg F1-Score	0.77
XGBoost	Accuracy	0.77
	Macro Avg Precision	0.74
	Macro Avg Recall	0.70
	Macro Avg F1-Score	0.72
	Weighted Avg F1-Score	0.77
Logistic Regression	Accuracy	0.68
	Macro Avg Precision	0.56
	Macro Avg Recall	0.60
	Macro Avg F1-Score	0.57
	Weighted Avg F1-Score	0.69

As shown in the table above, both XGBoost and Random Forest models demonstrated comparable and the highest overall performance on the validation set, achieving an accuracy of approximately 72.5% and 72.9% respectively. Logistic Regression, on the other hand, showed a lower accuracy of 64.1%. With class 3 (Very Unhealthy) showing the best performance (81% F1-score), Random Forest exhibited a balanced performance across most classes, having comparatively better precision, recall, and F1-scores than the other models. Similarly, XGBoost also performed well on the majority of classes (also having the highest 81% for class 3). However, it performed poorly for class 0 compared to Random Forest, with notably lower precision and recall values. Lastly, Logistic Regression showed the weakest performance across the majority of classes, and especially for class 0, it had a very low precision of 21% despite having a higher recall of 57%. Logistic Regression also performed weakly on class 3 compared to the other two models, with a recall of just 65%, which lowered its F1-score to 74%.

Finally, the macro-averaged F1-scores of Random Forest (69%) and XGBoost (64%) indicate that they performed in a balanced way across all classes compared to Logistic Regression (55%). The weighted averages of both Random Forest and XGBoost were found to be 73%, which shows their similarity in overall accuracies.

VIII. Discussion and Conclusion

The findings of this research demonstrate the feasibility of using machine learning models to predict the severity of air quality in Kathmandu, with a specific goal of taking timely proactive measures to safeguard hospital environments. Both the ensemble models, Random Forest and XGBoost, consistently outperformed Logistic Regression across the different classes, indicating their usefulness for this complex predictive task. Their balanced performance, as reflected in their precision, recall, and F1-score, hints at their ability to accurately identify the severity of air quality, which is crucial for guiding timely interventions.

Air pollution in Kathmandu has been a serious public health concern, exacerbated by the city's unique and challenging geographical factors. Therefore, the findings of these findings become quite substantial for hospital environments in Kathmandu. The ability to predict air quality severity can help accelerate our transition from a reactive approach to a proactive one by allowing hospitals to take necessary precautions. For example, if a Hazardous air quality level is predicted within a reasonable window, hospital administrators can take necessary precautions to limit the outdoor exposure of vulnerable patients, as well as adjusting the ventilation systems for filtered air recirculation or to temporarily restrict all non-essential outdoor activities of patients and staff.

Furthermore, the prediction should be adjusted in accordance with equipment protection. Sensitive medical equipment is susceptible to damage from contamination by particulate matter and other pollutants. With the ability to anticipate episodes of poor air quality, hospitals can implement measures such as enabling high-efficiency particulate air (HEPA) filters, increasing positive pressure in critical areas, or performing informed maintenance on HVAC systems [11]. These timely proactive management strategies for hospital equipment ultimately extend their lifespan, reduce operational costs, and ensure the continuous functionality of critical healthcare infrastructure. This project thus addresses a serious and prevalent real-world issue, especially for developing countries crippled by severe air pollution, providing solutions that could potentially impact the long-term operational resilience and patient safety within vulnerable healthcare environments.

This research aims to contribute to a much broader field by practically demonstrating a practical application of machine learning, which can eventually help to bridge a critical data gap in environmental monitoring. It is acknowledged that the lack of reliable and consistent long-term records of air pollution over Kathmandu makes projects like these challenging, which is what makes this project particularly valuable, as it leverages already scarce resources to predict air quality severity. The methodologies used can be laid out as a foundational framework for regions with very limited direct pollutant monitoring infrastructures.

Despite the promising results, this study has several limitations. The primary reason behind the inaccuracies of the resultant model can be attributed to the disparity among the sources of the meteorological and air quality datasets. The air quality dataset was collected from a particular local ground station at a certain location, whereas the meteorological data is an aggregation from multiple ground stations within Kathmandu. Thus, datasets collected from the same locality would be much more accurate at predicting air quality severity than the model developed in this project.

The potential of the model can further be realized by installing it on embedded IoT devices within hospital vicinities and monitoring live air quality. The embedded IoT devices shall be directly connected to sensors that can capture weather data like temperature, humidity, and wind speed. Along with records from the past few days or months (lagged features), the device would then be able to predict the real-time severity of air quality with significantly higher accuracies than achieved in this project.

In conclusion, this research successfully showcases the ability of machine learning in predicting air quality severity in Kathmandu, offering a critical tool for proactively safeguarding public health and hospital equipment. It demonstrates how complex environmental data can be transformed into actionable insights and contribute to enhancing patient safety, operational efficiency, and the overall resilience of healthcare systems in urban areas.

References

[1] R.S.A. Usmani, T.R. Pillai, I.A.T. Hashem, M. Marjani, R.B. Shaharudin, and M.T. Latif, "Air pollution and cardiorespiratory hospitalization, modeling and analysis using artificial intelligence techniques," Research Square, 2021. doi:10.21203/rs-330603/v1.

[2] S. Ali, D. Gurung, and R. Pokhrel, "Causes and Consequences of Air Pollution in Nepal," Institute of Development Studies (IDS), 2024.

[3] R.S.A. Usmani, T.R. Pillai, I.A.T. Hashem, M. Marjani, R.B. Shaharudin, and M.T. Latif, "Air pollution and cardiorespiratory hospitalization, modeling and analysis using artificial intelligence techniques," Research Square, 2021. doi:10.21203/rs-330603/v1.

[4] A. Kumar, R. Singh, and P. Sharma, "Machine Learning Approach for Air Quality Prediction in Smart Cities Using a Neural Network Logistic Leaf-Based Model," SSRN Electronic Journal, 2024. Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5091438

[5] ICIMOD, "Kathmandu choked on polluted air for 75 of the last 90 days," April 3, 2025. Available at: https://www.icimod.org/press-release/kathmandu-choked-on-polluted-air-for-75-of-the-last-90-days/

[6] Y. Bhavsar, "oizom," Oizom, Mar. 03, 2025. Available at: https://oizom.com/hospital-air-quality-sources-risks-control-measures/

[7] World Health Organization, "Health consequences of air pollution on populations," June 25, 2024.

[8] European Environment Agency, "How air pollution affects our health," [Online]. Available at: https://www.eea.europa.eu/en/topics/in-depth/air-pollution/eow-it-affects-our-health. [Accessed: May 27, 2025].

[9] R. Ji and S. Qu, "Investigation and evaluation of energy consumption performance for hospital buildings in China," Sustainability, vol. 11, no. 6, p. 1724, Mar. 2019, doi: 10.3390/su11061724.

[10] A. González González, J. García-Sanz-Calcedo, and D. Rodríguez Salgado, "Evaluation of energy consumption in German hospitals: Benchmarking in the public sector," Energies, vol. 11, no. 9, p. 2279, Sep. 2018, doi: 10.3390/en11092279.

[11] X. Zhang, Y. Yang, and J. Kim, "Retrofitting for Improving Indoor Air Quality and Energy Efficiency in the Hospital Building," Sustainability, vol. 15, no. 4, art. no. 3464, 2023, doi: 10.3390/su15043464.

[12] C.-C. Chan et al., "Contribution of Indoor- and Outdoor-Generated Fine and Coarse Particles to Indoor Air in Taiwanese Hospitals," Aerosol Air Qual. Res., vol. 18, no. 4, pp. 864–874, Apr. 2018, doi: 10.4209/aaqr.2017.12.0552.

[13] World Health Organization, "Health impacts of air pollution," [Online]. Available at: https://www.who.int/teams/environment-climate-change-and-health/air-quality-energy-and-health/health-impacts. [Accessed: May 27, 2025].

[14] Online Khabar, "Kathmandu ranked second most polluted city globally," [Online]. Available at: https://english.onlinekhabar.com/kathmandu-ranked-second-most-polluted-city-globally.html. [Accessed: May 28, 2025].

[15] A. Gautam and A. K. Singh, "Applying machine learning algorithms to estimate PM$_{2.5}$ using satellite data and metrological data," May 2024, doi: 10.13140/RG.2.2.11787.36647. [Online]. Available at: https://www.researchgate.net/publication/381166847_Applying_machine_learning_algorithms_to_estimate_PM_25_using_satellite_data_and_metrological_data. [Accessed: May 28, 2025].

[16] Department of Environment, Government of Nepal, "Main-report-60120-1660561765.pdf," [Online]. Available at: https://doenv.gov.np/progressfiles/Main-report-60120-1660561765.pdf. [Accessed: May 28, 2025].

[17] Department of Hydrology and Meteorology, Nepal, "Meteoro-logical data request payment procedure," (2019-2021).

[18] Weather Underground, "Historical Weather in Kathmandu, Nepal - May 27, 2025," [Online]. Available at: https://www.wunderground.com/history/daily/np/kathmandu/VNKT. [Accessed: May 28, 2025].

[19] F. Ghasemian, M. Khorrami, and A. Shahsavani, “Investigating Meteorological Factors Influencing Pollutant Concentrations and Copernicus Atmosphere Monitoring Service (CAMS) Model Forecasts in the Tehran Metropolis,” Atmosphere, vol. 16, no. 3, p. 264, 2025, doi: 10.3390/atmos16030264.

[20] I. Logothetis, G. Giakoumakis, A. Mitsotakis, and P. Grammelis, “Air pollution and the impact of meteorological factors in air quality of Chalki Island during a winter period of 2023-2024,” EGU General Assembly 2025, EGU25-5695, 2025. doi: 10.5194/egusphere-egu25-5695.

[21] Time and Date, “Holidays and Observances in Nepal in 2020,” [Online]. Available at: https://www.timeanddate.com/holidays/nepal/2020. [Accessed: May 28, 2025].

[22] M. Maalouf, “Logistic Regression in Data Analysis: An Overview,” International Journal of Data Analysis Techniques and Strategies (IJDATS), vol. 3, no. 3, pp. 281–299, 2011.

[23] GeeksforGeeks, “Understanding Logistic Regression,” [Online]. Available at: https://www.geeksforgeeks.org/understanding-logistic-regression/. [Accessed: May 28, 2025].

[24] L. Breiman and A. Cutler, “Random Forest Algorithm in Machine Learning,” [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/#h-what-is-random-forest. [Accessed: June 1, 2025].

[25] M. Méndez, M. G. Merayo, and M. Núñez, “Machine learning algorithms to forecast air quality: a survey,” doi: 10.1007/s10462-023-10424-4, Feb. 16, 2023.

[26] The World Air Quality Index Project, “US Embassy, Kathmandu, Nepal Air Pollution: Real-time Air Quality Index,” [Online]. Available: https://aqicn.org/city/nepal/kathmandu/us-embassy/. [Accessed: May 24, 2025].

[27] Google Colaboratory, “Google Colab,” [Online]. Available: https://colab.research.google.com/. [Accessed: June 1, 2025].

[28] A. R. Alsaber, J. Pan, and A. Al-Hurban, “Handling complex missing data using random forest approach for an air quality monitoring dataset: A case study of Kuwait environmental data (2012 to 2018),” International Journal of Environmental Research and Public Health, vol. 18, no. 3, p. 1333, 2021.

[29] S. Bakken et al., “Standards in action: historical and current perspectives,” Journal of the American Medical Informatics Association, vol. 30, no. 12, pp. 2077–2082, 2023.

[30] H. Zhang, Z. Li, Y. Xue, X. Chang, J. Su, P. Wang, Q. Guo, and H. Sun, “A Stochastic Bi-level Optimal Allocation Approach of Intelligent Buildings Considering Energy Storage Sharing Services,” IEEE Trans. Consum. Electron., vol. 70, pp. 5142–5153, 2024.

[31] Republica, “Kathmandu Valley’s temperature plunges to 1.6 degree Celsius,” [Online]. Available at: https://myrepublica.nagariknetwork.com/news/kathmandu-valley-s-temperature-plunges-to-1-6-degree-celsius. [Accessed: June 1, 2025].

[32] Khabarhub, “Cold wave grips Nepal, temperature dips to -3 degree Celsius,” [Online]. Available at: https://english.khabarhub.com/2024/15/361114/. [Accessed: Dec. 27, 2024].

[33] Deepchecks Team. “Normalization in Machine Learning.” [Online]. Available: https://www.deepchecks.com/glossary/normalization-in-machine-learning/. [Accessed: June 1, 2025].

[34] M. Stone, “Cross-validatory choice and assessment of statistical predictions,” *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 36, no. 2, pp. 111–147, 1974. doi: 10.1111/j.2515-3107.1974.tb01034.x.

[35] Y. Bengio, “Practical recommendations for gradient-based training of deep architectures,” arXiv preprint arXiv:1206.5533, 2012.

[36] G. van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[37] W. McKinney, “Data Structures for Statistical Computing in Python,” in *Proceedings of the 9th Python in Science Conference*, 2010, pp. 56–61.

[38] C.R. Harris et al., “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[39] Streamlit. “Streamlit — The fastest way to build and share data apps.” [Online]. Available: https://streamlit.io/. [Accessed: June 1, 2025].

[40] F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[41] Nepali Times, “Kathmandu air pollution hits record high,” Jan. 5, 2021. [Online]. Available: https://nepalitimes.com/latest/kathmandu-air-pollution-hits-record-high. [Accessed: June 1, 2025].

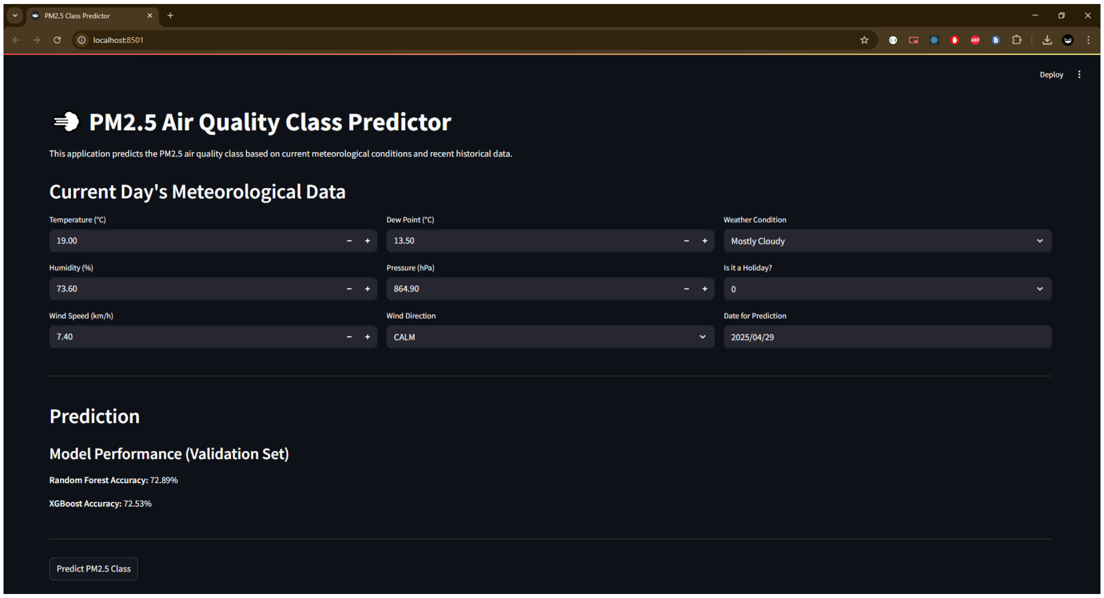
[42] The Record, “Kathmandu’s AQI crosses 500 mark,” Jan. 5, 2021. [Online]. Available: https://www.recordnepal.com/kathmandus-aqi-crosses-500-mark. [Accessed: June 1, 2025].

[43] Kathmandu Post, “Haze and smoke turn Kathmandu most polluted; low visibility affects flights,” Mar. 27, 2021. [Online]. Available: https://kathmandupost.com/climate-environment/2021/03/27/haze-and-smoke-turn-kathmandu-most-polluted-low-visibility-affects-flights. [Accessed: June 1, 2025].

[44] ScienceDirect, “Atmospheric Pollution Research: Impact of wildfires on air quality in Nepal,” 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1309104221002345. [Accessed: June 1, 2025].

APPENDIX

https://github.com/guts982/ml-pm25-prediction-using-classification



PM2.5 Air Quality Class Predictor

This application predicts the PM2.5 air quality class based on current meteorological conditions and recent historical data.

Current Day's Meteorological Data

Temperature (°C): 19.00

Dew Point (°C): 13.50

Weather Condition: Mostly Cloudy

Humidity (%): 73.60

Pressure (hPa): 864.90

Is it a Holiday?: 0

Wind Speed (km/h): 7.40

Wind Direction: CALM

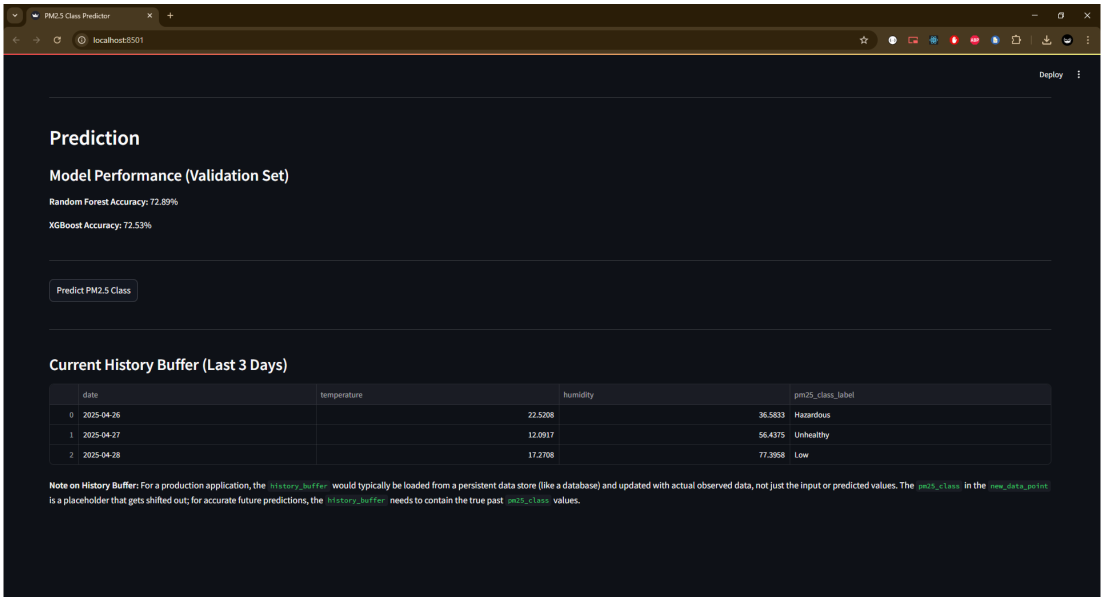
Date for Prediction: 2025/04/29

Prediction

Model Performance (Validation Set)

Random Forest Accuracy: 72.89%

XGBoost Accuracy: 72.53%



Prediction

Model Performance (Validation Set)

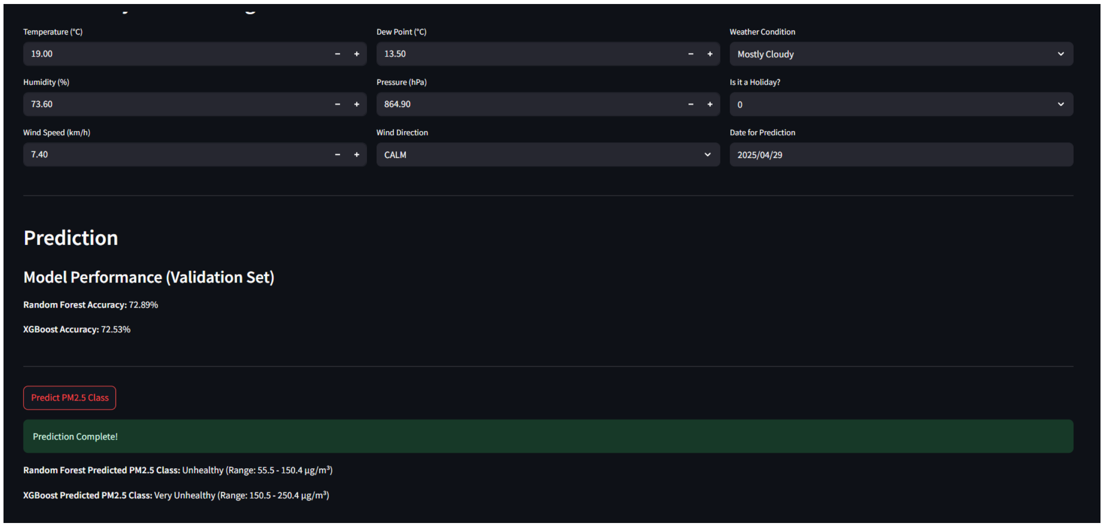
Random Forest Accuracy: 72.89%

XGBoost Accuracy: 72.53%

Current History Buffer (Last 3 Days)

	date	temperature	humidity	pm25_class_label
0	2025-04-26	22.5208	36.5833	Hazardous
1	2025-04-27	12.0917	56.4375	Unhealthy
2	2025-04-28	17.2708	77.3958	Low

Note on History Buffer: For a production application, the `history_buffer` would typically be loaded from a persistent data store (like a database) and updated with actual observed data, not just the input or predicted values. The `pm25_class` in the `new_data_point` is a placeholder that gets shifted out; for accurate future predictions, the `history_buffer` needs to contain the true past `pm25_class` values.



Temperature (°C)
19.00

Dew Point (°C)
13.50

Weather Condition
Mostly Cloudy

Humidity (%)
73.60

Pressure (hPa)
864.90

Is it a Holiday?
0

Wind Speed (km/h)
7.40

Wind Direction
CALM

Date for Prediction
2025/04/29

Prediction

Model Performance (Validation Set)

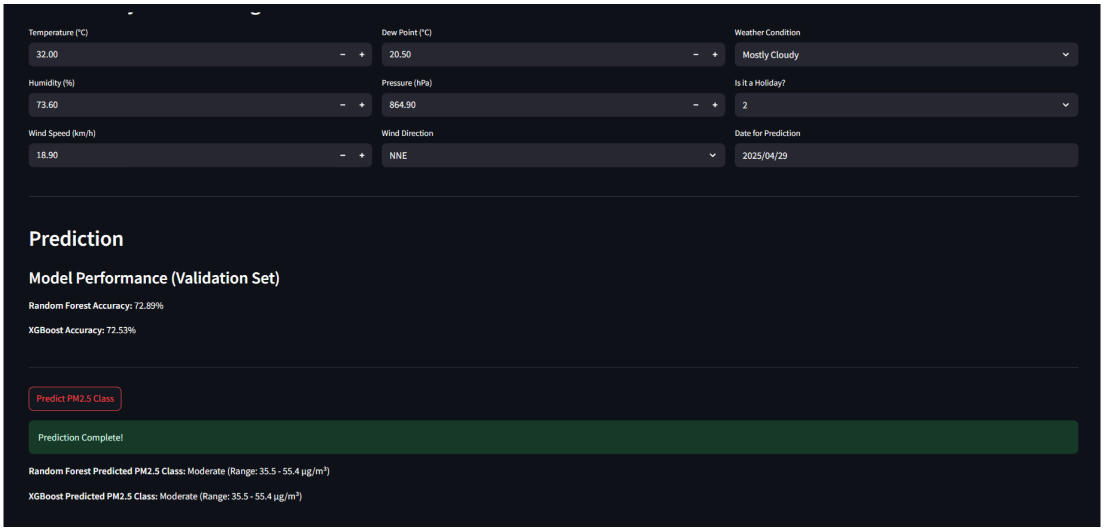
Random Forest Accuracy: 72.89%

XGBoost Accuracy: 72.53%

Prediction Complete!

Random Forest Predicted PM2.5 Class: Unhealthy (Range: 55.5 - 150.4 µg/m³)

XGBoost Predicted PM2.5 Class: Very Unhealthy (Range: 150.5 - 250.4 µg/m³)



Temperature (°C)
32.00

Dew Point (°C)
20.50

Weather Condition
Mostly Cloudy

Humidity (%)
73.60

Pressure (hPa)
864.90

Is it a Holiday?
2

Wind Speed (km/h)
18.90

Wind Direction
NNE

Date for Prediction
2025/04/29

Prediction

Model Performance (Validation Set)

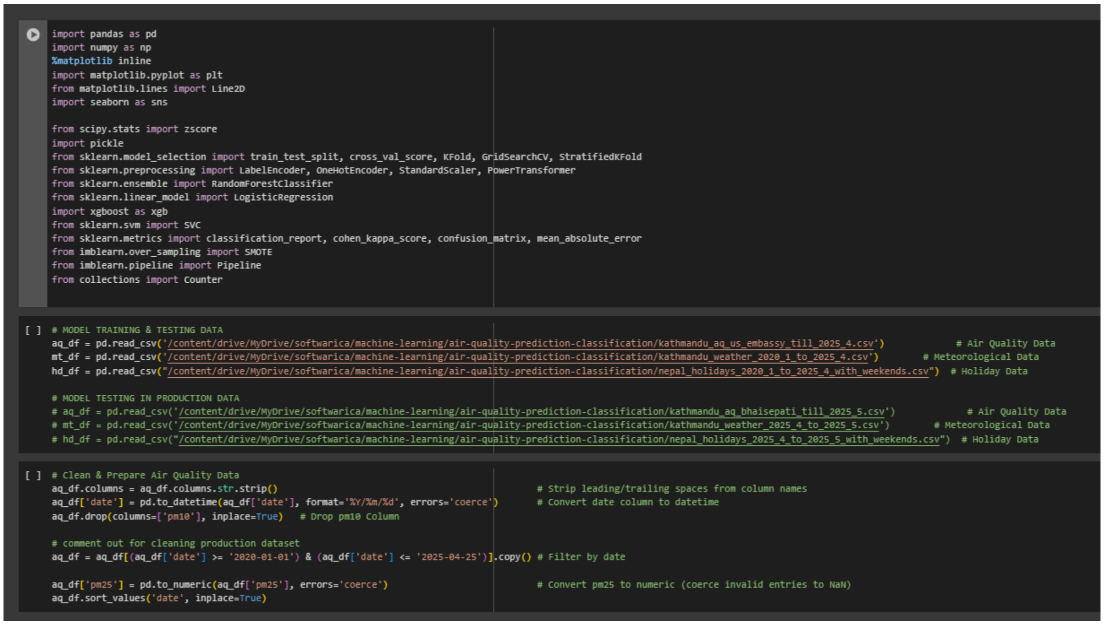
Random Forest Accuracy: 72.89%

XGBoost Accuracy: 72.53%

Prediction Complete!

Random Forest Predicted PM2.5 Class: Moderate (Range: 35.5 - 55.4 µg/m³)

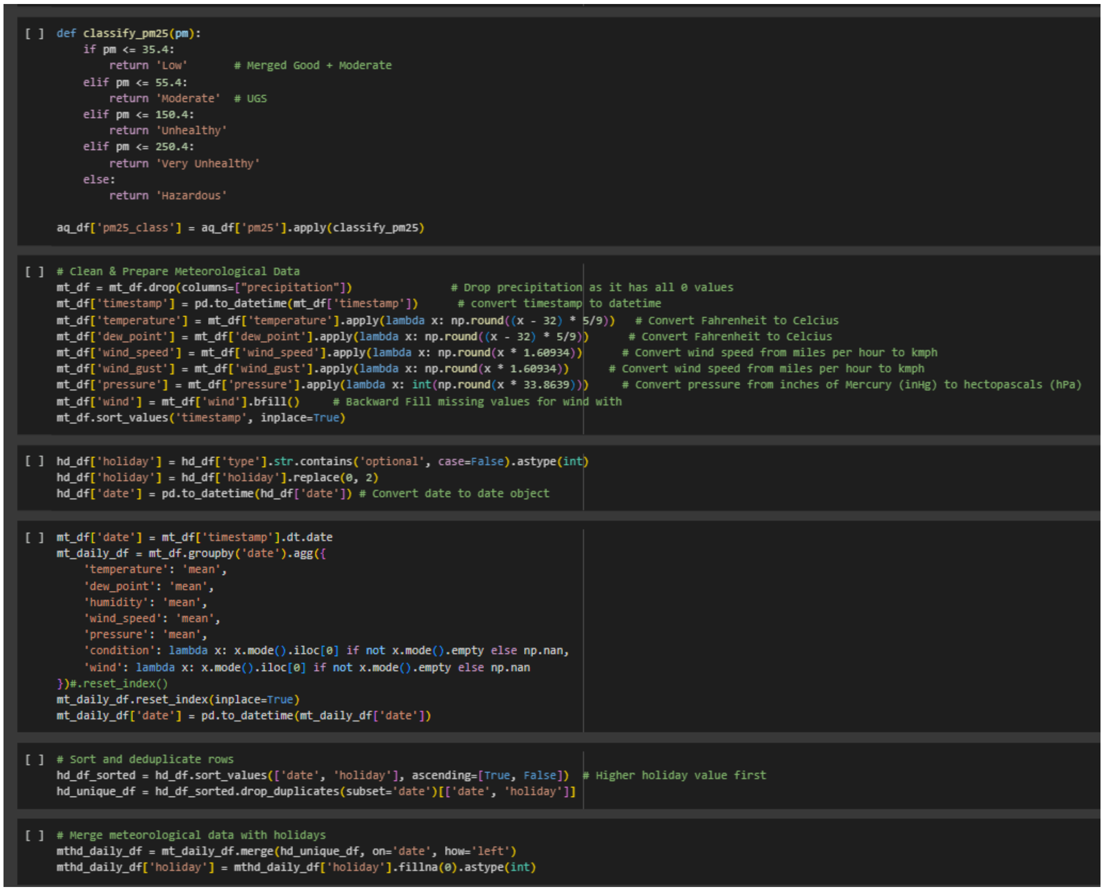
XGBoost Predicted PM2.5 Class: Moderate (Range: 35.5 - 55.4 µg/m³)



```
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D
import seaborn as sns

from scipy.stats import zscore
import pickle
from sklearn.model_selection import train_test_split, cross_val_score, KFold, GridSearchCV, StratifiedKFold
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler, PowerTransformer
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
import xgboost as xgb
from sklearn.svm import SVC
from sklearn.metrics import classification_report, cohen_kappa_score, confusion_matrix, mean_absolute_error
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline
from collections import Counter
```

```
[ ] # MODEL TRAINING & TESTING DATA
    aq_df = pd.read_csv('/content/drive/MyDrive/softwarica/machine-learning/air-quality-prediction-classification/kathmandu_aq_us_embassy_till_2025_4.csv')           # Air Quality Data
    mt_df = pd.read_csv('/content/drive/MyDrive/softwarica/machine-learning/air-quality-prediction-classification/kathmandu_weather_2020_1_to_2025_4.csv')        # Meteorological Data
    hd_df = pd.read_csv("/content/drive/MyDrive/softwarica/machine-learning/air-quality-prediction-classification/nepal_holidays_2020_1_to_2025_4_with_weekends.csv")  # Holiday Data

    # MODEL TESTING IN PRODUCTION DATA
    # aq_df = pd.read_csv('/content/drive/MyDrive/softwarica/machine-learning/air-quality-prediction-classification/kathmandu_aq_bhaisepati_till_2025_5.csv')           # Air Quality Data
    # mt_df = pd.read_csv('/content/drive/MyDrive/softwarica/machine-learning/air-quality-prediction-classification/kathmandu_weather_2025_4_to_2025_5.csv')        # Meteorological Data
    # hd_df = pd.read_csv("/content/drive/MyDrive/softwarica/machine-learning/air-quality-prediction-classification/nepal_holidays_2025_4_to_2025_5_with_weekends.csv")  # Holiday Data
```

```
[ ] # Clean & Prepare Air Quality Data
    aq_df.columns = aq_df.columns.str.strip()                                            # Strip leading/trailing spaces from column names
    aq_df['date'] = pd.to_datetime(aq_df['date'], format='%Y/%m/%d', errors='coerce')       # Convert date column to datetime
    aq_df.drop(columns=['pm10'], inplace=True)   # Drop pm10 Column

    # comment out for cleaning production dataset
    aq_df = aq_df[(aq_df['date'] >= '2020-01-01') & (aq_df['date'] <= '2025-04-25')].copy() # Filter by date

    aq_df['pm25'] = pd.to_numeric(aq_df['pm25'], errors='coerce')                         # Convert pm25 to numeric (coerce invalid entries to NaN)
    aq_df.sort_values('date', inplace=True)
```



```python
def classify_pm25(pm):
    if pm <= 35.4:
        return 'Low'       # Merged Good + Moderate
    elif pm <= 55.4:
        return 'Moderate'  # UGS
    elif pm <= 150.4:
        return 'Unhealthy'
    elif pm <= 250.4:
        return 'Very Unhealthy'
    else:
        return 'Hazardous'

aq_df['pm25_class'] = aq_df['pm25'].apply(classify_pm25)
```

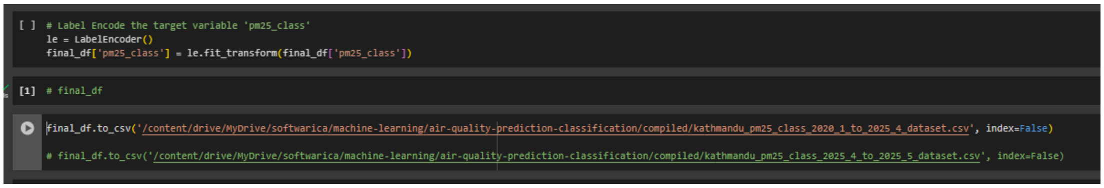
```python
# Clean & Prepare Meteorological Data
mt_df = mt_df.drop(columns=["precipitation"])               # Drop precipitation as it has all 0 values
mt_df['timestamp'] = pd.to_datetime(mt_df['timestamp'])      # convert timestamp to datetime
mt_df['temperature'] = mt_df['temperature'].apply(lambda x: np.round((x - 32) * 5/9))   # Convert Fahrenheit to Celcius
mt_df['dew_point'] = mt_df['dew_point'].apply(lambda x: np.round((x - 32) * 5/9))      # Convert Fahrenheit to Celcius
mt_df['wind_speed'] = mt_df['wind_speed'].apply(lambda x: np.round(x * 1.60934))      # Convert wind speed from miles per hour to kmph
mt_df['wind_gust'] = mt_df['wind_gust'].apply(lambda x: np.round(x * 1.60934))      # Convert wind speed from miles per hour to kmph
mt_df['pressure'] = mt_df['pressure'].apply(lambda x: int(np.round(x * 33.8639)))     # Convert pressure from inches of Mercury (inHg) to hectopascals (hPa)
mt_df['wind'] = mt_df['wind'].bfill()     # Backward Fill missing values for wind with
mt_df.sort_values('timestamp', inplace=True)
```

```python
hd_df['holiday'] = hd_df['type'].str.contains('optional', case=False).astype(int)
hd_df['holiday'] = hd_df['holiday'].replace(0, 2)
hd_df['date'] = pd.to_datetime(hd_df['date']) # Convert date to date object
```

```python
mt_df['date'] = mt_df['timestamp'].dt.date
mt_daily_df = mt_df.groupby('date').agg({
    'temperature': 'mean',
    'dew_point': 'mean',
    'humidity': 'mean',
    'wind_speed': 'mean',
    'pressure': 'mean',
    'condition': lambda x: x.mode().iloc[0] if not x.mode().empty else np.nan,
    'wind': lambda x: x.mode().iloc[0] if not x.mode().empty else np.nan
})#.reset_index()
mt_daily_df.reset_index(inplace=True)
mt_daily_df['date'] = pd.to_datetime(mt_daily_df['date'])
```

```python
# Sort and deduplicate rows
hd_df_sorted = hd_df.sort_values(['date', 'holiday'], ascending=[True, False]) # Higher holiday value first
hd_unique_df = hd_df_sorted.drop_duplicates(subset='date')[['date', 'holiday']]
```

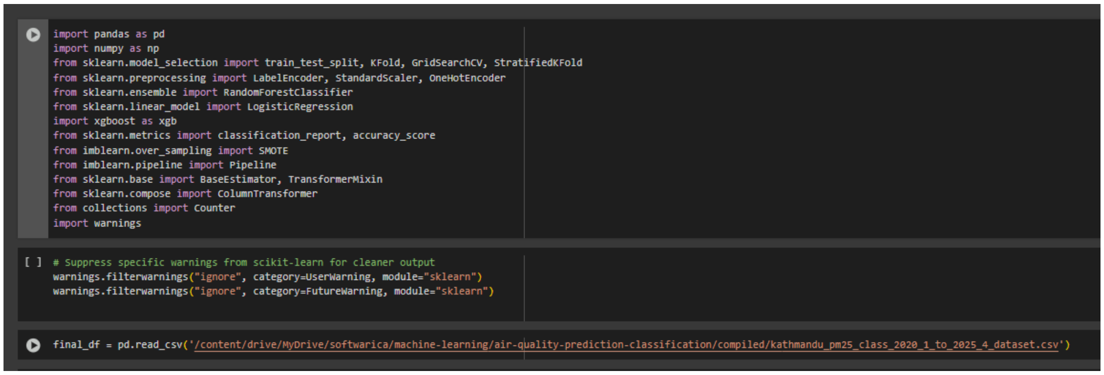
```python
# Merge meteorological data with holidays
mthd_daily_df = mt_daily_df.merge(hd_unique_df, on='date', how='left')
mthd_daily_df['holiday'] = mthd_daily_df['holiday'].fillna(0).astype(int)
```



```python
# Label Encode the target variable 'pm25_class'
le = LabelEncoder()
final_df['pm25_class'] = le.fit_transform(final_df['pm25_class'])
```

```python
# final_df
```

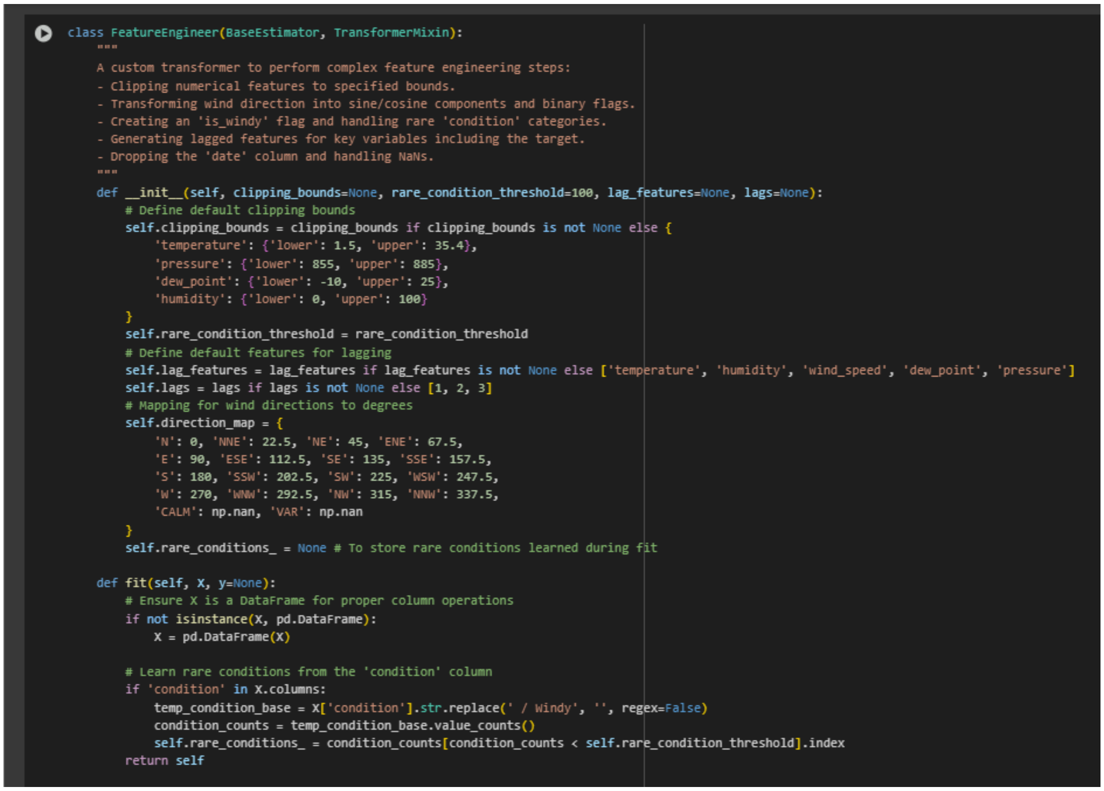
```python
final_df.to_csv('/content/drive/MyDrive/softwarica/machine-learning/air-quality-prediction-classification/compiled/kathmandu_pm25_class_2020_1_to_2025_4_dataset.csv', index=False)

# final_df.to_csv('/content/drive/MyDrive/softwarica/machine-learning/air-quality-prediction-classification/compiled/kathmandu_pm25_class_2025_4_to_2025_5_dataset.csv', index=False)
```



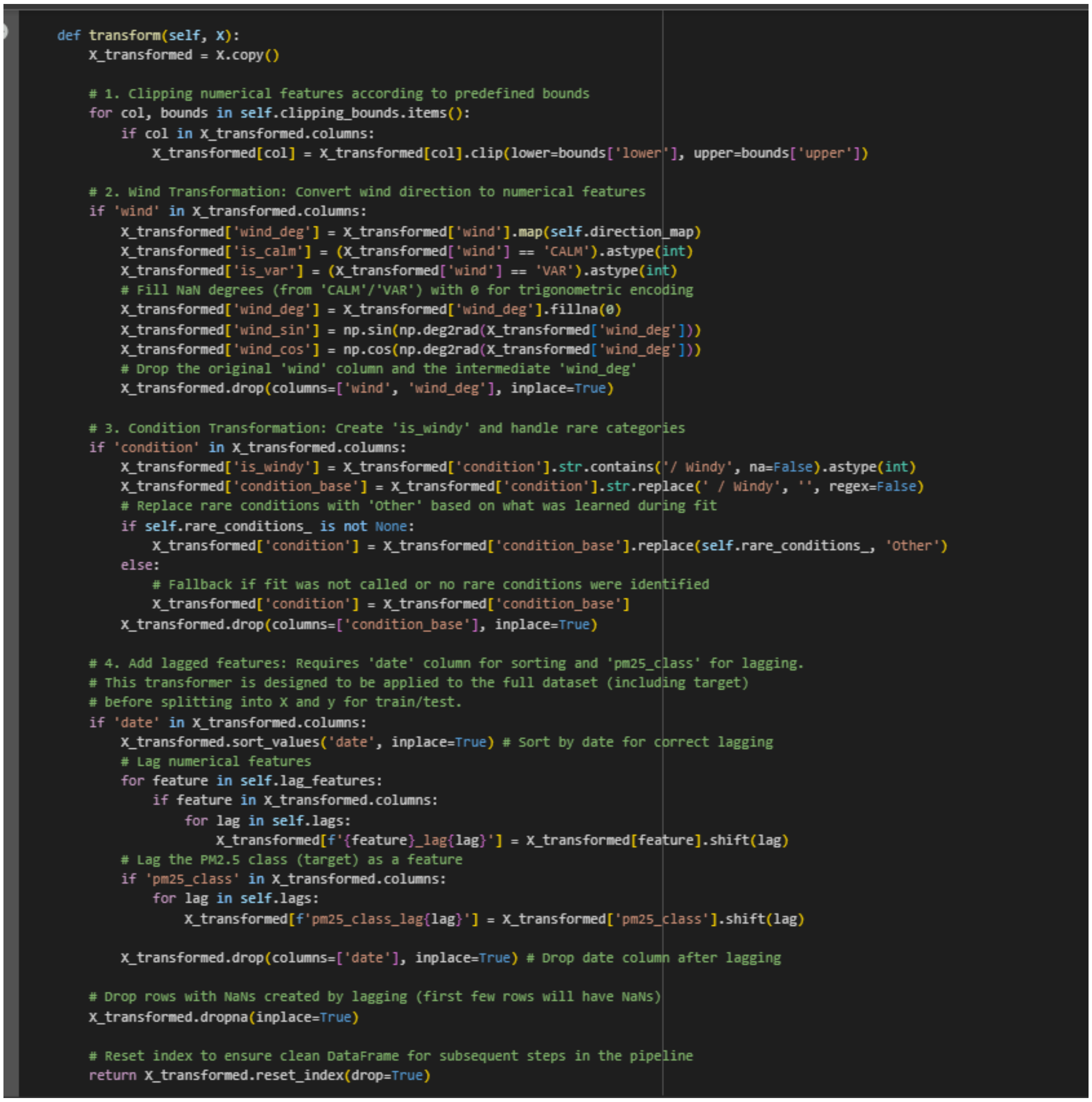
```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, KFold, GridSearchCV, StratifiedKFold
from sklearn.preprocessing import LabelEncoder, StandardScaler, OneHotEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
import xgboost as xgb
from sklearn.metrics import classification_report, accuracy_score
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.compose import ColumnTransformer
from collections import Counter
import warnings
```

```python
# Suppress specific warnings from scikit-learn for cleaner output
warnings.filterwarnings("ignore", category=UserWarning, module="sklearn")
warnings.filterwarnings("ignore", category=FutureWarning, module="sklearn")
```

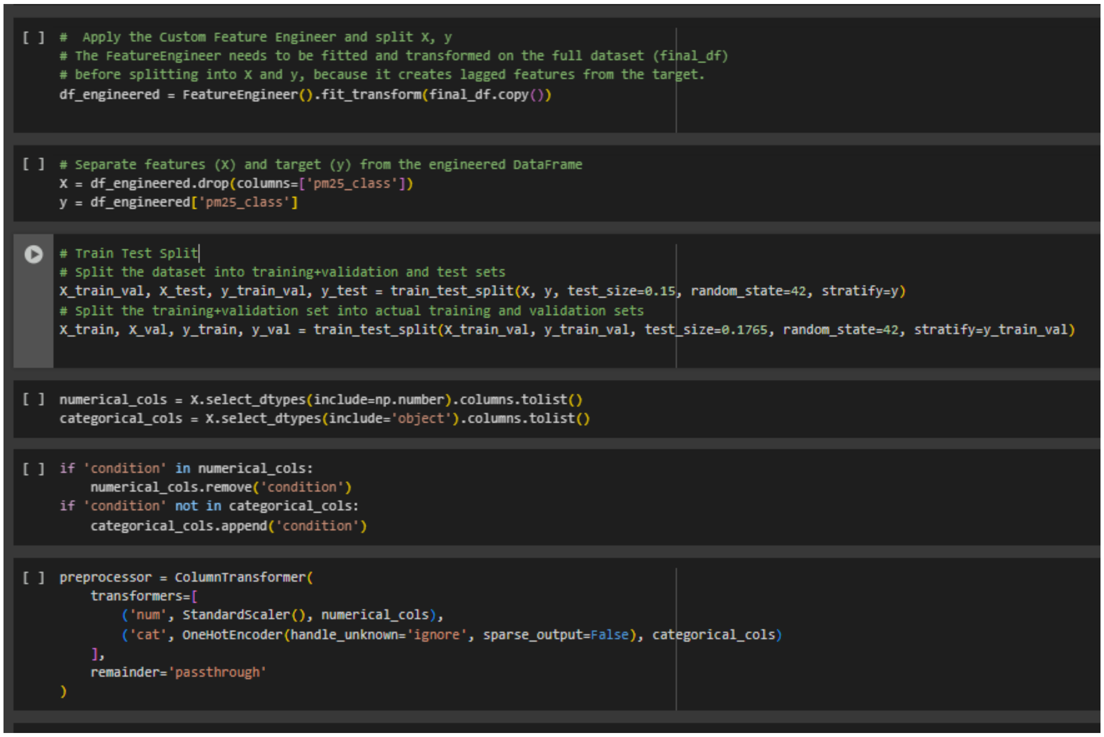
```python
final_df = pd.read_csv('/content/drive/MyDrive/softwarica/machine-learning/air-quality-prediction-classification/compiled/kathmandu_pm25_class_2020_1_to_2025_4_dataset.csv')
```



```python
class FeatureEngineer(BaseEstimator, TransformerMixin):
    """
    A custom transformer to perform complex feature engineering steps:
    - Clipping numerical features to specified bounds.
    - Transforming wind direction into sine/cosine components and binary flags.
    - Creating an 'is_windy' flag and handling rare 'condition' categories.
    - Generating lagged features for key variables including the target.
    - Dropping the 'date' column and handling NaNs.
    """
    def __init__(self, clipping_bounds=None, rare_condition_threshold=100, lag_features=None, lags=None):
        # Define default clipping bounds
        self.clipping_bounds = clipping_bounds if clipping_bounds is not None else {
            'temperature': {'lower': 1.5, 'upper': 35.4},
            'pressure': {'lower': 855, 'upper': 885},
            'dew_point': {'lower': -10, 'upper': 25},
            'humidity': {'lower': 0, 'upper': 100}
        }
        self.rare_condition_threshold = rare_condition_threshold
        # Define default features for lagging
        self.lag_features = lag_features if lag_features is not None else ['temperature', 'humidity', 'wind_speed', 'dew_point', 'pressure']
        self.lags = lags if lags is not None else [1, 2, 3]
        # Mapping for wind directions to degrees
        self.direction_map = {
            'N': 0, 'NNE': 22.5, 'NE': 45, 'ENE': 67.5,
            'E': 90, 'ESE': 112.5, 'SE': 135, 'SSE': 157.5,
            'S': 180, 'SSW': 202.5, 'SW': 225, 'WSW': 247.5,
            'W': 270, 'WNW': 292.5, 'NW': 315, 'NNW': 337.5,
            'CALM': np.nan, 'VAR': np.nan
        }
        self.rare_conditions_ = None # To store rare conditions learned during fit

    def fit(self, X, y=None):
        # Ensure X is a DataFrame for proper column operations
        if not isinstance(X, pd.DataFrame):
            X = pd.DataFrame(X)

        # Learn rare conditions from the 'condition' column
        if 'condition' in X.columns:
            temp_condition_base = X['condition'].str.replace(' / Windy', '', regex=False)
            condition_counts = temp_condition_base.value_counts()
            self.rare_conditions_ = condition_counts[condition_counts < self.rare_condition_threshold].index
        return self
```



```python
def transform(self, X):
    X_transformed = X.copy()

    # 1. Clipping numerical features according to predefined bounds
    for col, bounds in self.clipping_bounds.items():
        if col in X_transformed.columns:
            X_transformed[col] = X_transformed[col].clip(lower=bounds['lower'], upper=bounds['upper'])

    # 2. Wind Transformation: Convert wind direction to numerical features
    if 'wind' in X_transformed.columns:
        X_transformed['wind_deg'] = X_transformed['wind'].map(self.direction_map)
        X_transformed['is_calm'] = (X_transformed['wind'] == 'CALM').astype(int)
        X_transformed['is_var'] = (X_transformed['wind'] == 'VAR').astype(int)
        # Fill NaN degrees (from 'CALM'/'VAR') with 0 for trigonometric encoding
        X_transformed['wind_deg'] = X_transformed['wind_deg'].fillna(0)
        X_transformed['wind_sin'] = np.sin(np.deg2rad(X_transformed['wind_deg']))
        X_transformed['wind_cos'] = np.cos(np.deg2rad(X_transformed['wind_deg']))
        # Drop the original 'wind' column and the intermediate 'wind_deg'
        X_transformed.drop(columns=['wind', 'wind_deg'], inplace=True)

    # 3. Condition Transformation: Create 'is_windy' and handle rare categories
    if 'condition' in X_transformed.columns:
        X_transformed['is_windy'] = X_transformed['condition'].str.contains('/ Windy', na=False).astype(int)
        X_transformed['condition_base'] = X_transformed['condition'].str.replace(' / Windy', '', regex=False)
        # Replace rare conditions with 'Other' based on what was learned during fit
        if self.rare_conditions_ is not None:
            X_transformed['condition'] = X_transformed['condition_base'].replace(self.rare_conditions_, 'Other')
        else:
            # Fallback if fit was not called or no rare conditions were identified
            X_transformed['condition'] = X_transformed['condition_base']
        X_transformed.drop(columns=['condition_base'], inplace=True)

    # 4. Add lagged features: Requires 'date' column for sorting and 'pm25_class' for lagging.
    # This transformer is designed to be applied to the full dataset (including target)
    # before splitting into X and y for train/test.
    if 'date' in X_transformed.columns:
        X_transformed.sort_values('date', inplace=True) # Sort by date for correct lagging
        # Lag numerical features
        for feature in self.lag_features:
            if feature in X_transformed.columns:
                for lag in self.lags:
                    X_transformed[f'{feature}_lag{lag}'] = X_transformed[feature].shift(lag)
        # Lag the PM2.5 class (target) as a feature
        if 'pm25_class' in X_transformed.columns:
            for lag in self.lags:
                X_transformed[f'pm25_class_lag{lag}'] = X_transformed['pm25_class'].shift(lag)

        X_transformed.drop(columns=['date'], inplace=True) # Drop date column after lagging

    # Drop rows with NaNs created by lagging (first few rows will have NaNs)
    X_transformed.dropna(inplace=True)

    # Reset index to ensure clean DataFrame for subsequent steps in the pipeline
    return X_transformed.reset_index(drop=True)
```



```python
#  Apply the Custom Feature Engineer and split X, y
# The FeatureEngineer needs to be fitted and transformed on the full dataset (final_df)
# before splitting into X and y, because it creates lagged features from the target.
df_engineered = FeatureEngineer().fit_transform(final_df.copy())
```

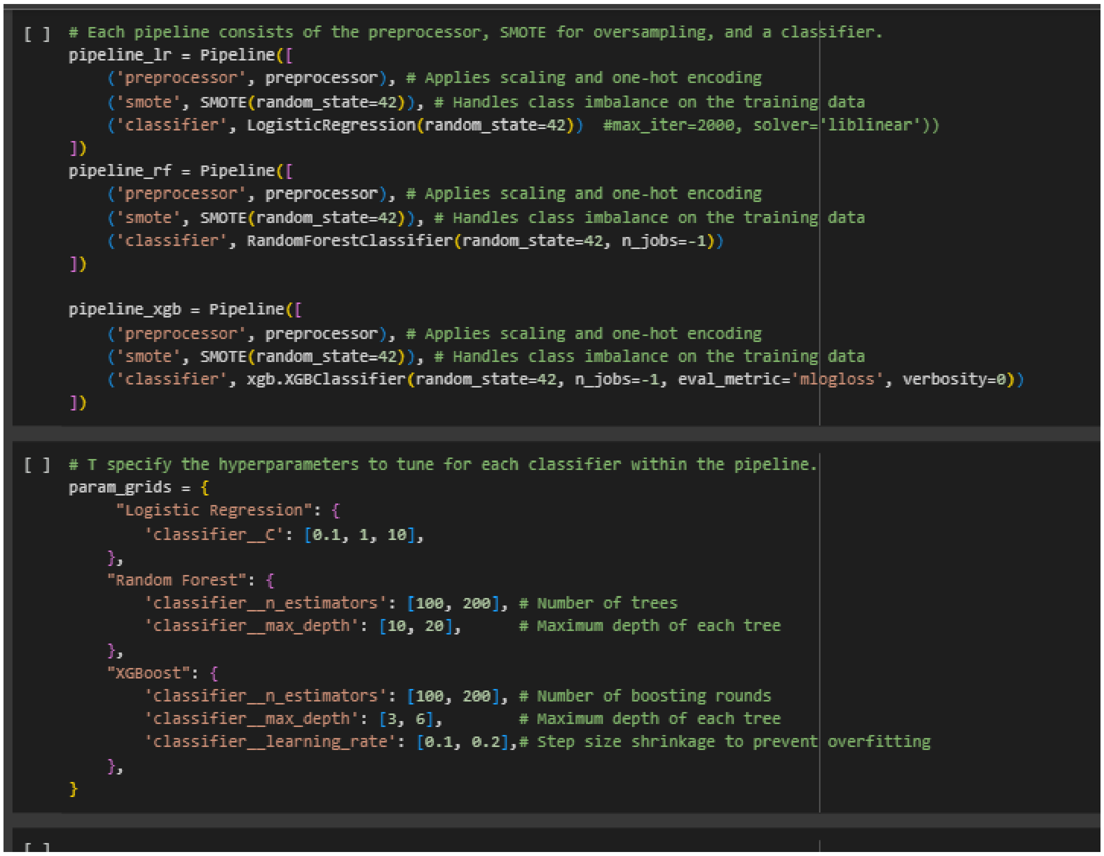
```python
# Separate features (X) and target (y) from the engineered DataFrame
X = df_engineered.drop(columns=['pm25_class'])
y = df_engineered['pm25_class']
```

```python
# Train Test Split
# Split the dataset into training+validation and test sets
X_train_val, X_test, y_train_val, y_test = train_test_split(X, y, test_size=0.15, random_state=42, stratify=y)
# Split the training+validation set into actual training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X_train_val, y_train_val, test_size=0.1765, random_state=42, stratify=y_train_val)
```

```python
numerical_cols = X.select_dtypes(include=np.number).columns.tolist()
categorical_cols = X.select_dtypes(include='object').columns.tolist()
```

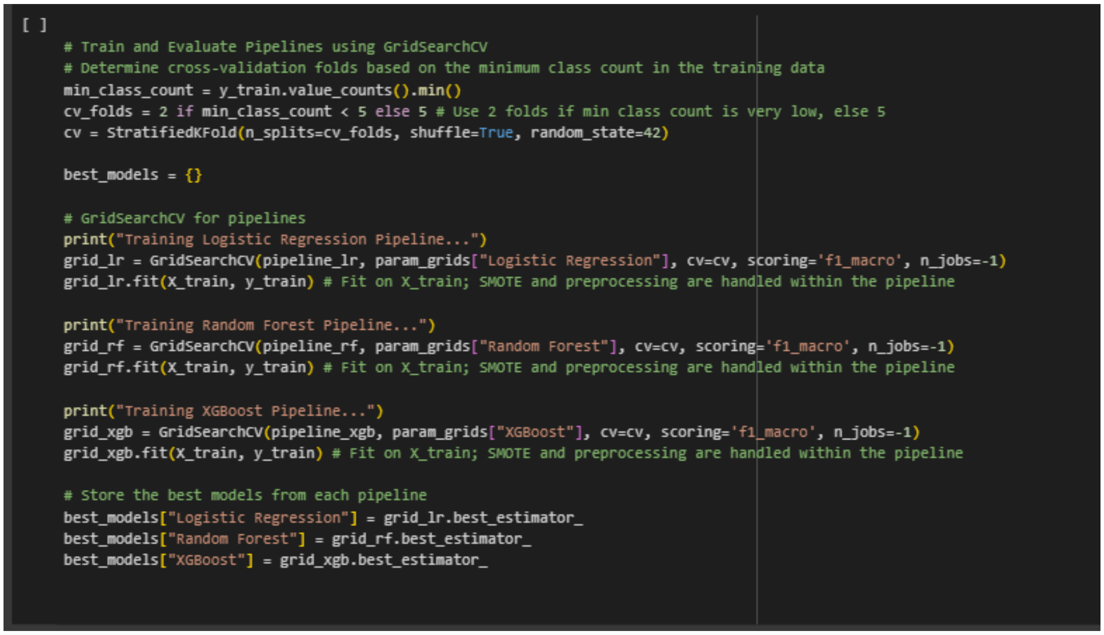
```python
if 'condition' in numerical_cols:
    numerical_cols.remove('condition')
if 'condition' not in categorical_cols:
    categorical_cols.append('condition')
```

```python
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_cols),
        ('cat', OneHotEncoder(handle_unknown='ignore', sparse_output=False), categorical_cols)
    ],
    remainder='passthrough'
)
```

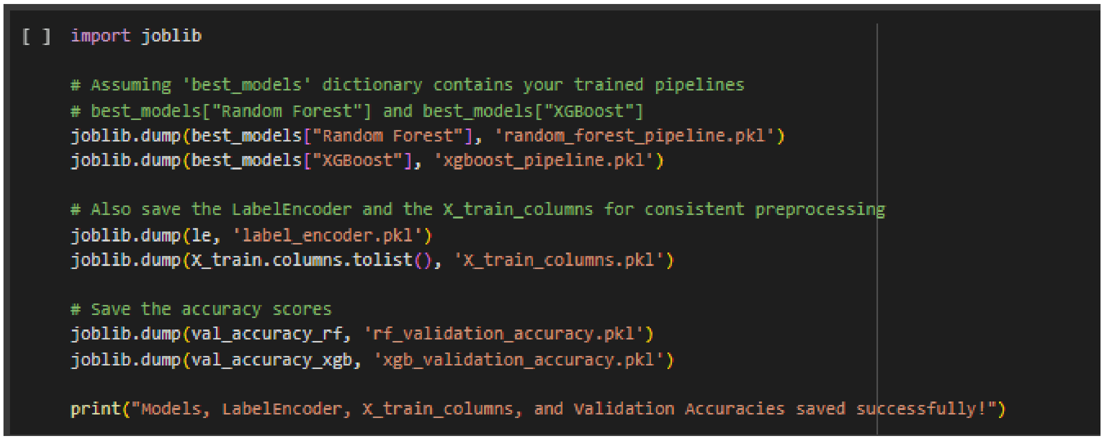


```python
# Each pipeline consists of the preprocessor, SMOTE for oversampling, and a classifier.
pipeline_lr = Pipeline([
    ('preprocessor', preprocessor), # Applies scaling and one-hot encoding
    ('smote', SMOTE(random_state=42)), # Handles class imbalance on the training data
    ('classifier', LogisticRegression(random_state=42))  #max_iter=2000, solver='liblinear'))
])
pipeline_rf = Pipeline([
    ('preprocessor', preprocessor), # Applies scaling and one-hot encoding
    ('smote', SMOTE(random_state=42)), # Handles class imbalance on the training data
    ('classifier', RandomForestClassifier(random_state=42, n_jobs=-1))
])

pipeline_xgb = Pipeline([
    ('preprocessor', preprocessor), # Applies scaling and one-hot encoding
    ('smote', SMOTE(random_state=42)), # Handles class imbalance on the training data
    ('classifier', xgb.XGBClassifier(random_state=42, n_jobs=-1, eval_metric='mlogloss', verbosity=0))
])
```

```python
# T specify the hyperparameters to tune for each classifier within the pipeline.
param_grids = {
     "Logistic Regression": {
        'classifier__C': [0.1, 1, 10],
    },
    "Random Forest": {
        'classifier__n_estimators': [100, 200], # Number of trees
        'classifier__max_depth': [10, 20],      # Maximum depth of each tree
    },
    "XGBoost": {
        'classifier__n_estimators': [100, 200], # Number of boosting rounds
        'classifier__max_depth': [3, 6],        # Maximum depth of each tree
        'classifier__learning_rate': [0.1, 0.2],# Step size shrinkage to prevent overfitting
    },
}
```



```
[ ]
    # Train and Evaluate Pipelines using GridSearchCV
    # Determine cross-validation folds based on the minimum class count in the training data
    min_class_count = y_train.value_counts().min()
    cv_folds = 2 if min_class_count < 5 else 5 # Use 2 folds if min class count is very low, else 5
    cv = StratifiedKFold(n_splits=cv_folds, shuffle=True, random_state=42)

    best_models = {}

    # GridSearchCV for pipelines
    print("Training Logistic Regression Pipeline...")
    grid_lr = GridSearchCV(pipeline_lr, param_grids["Logistic Regression"], cv=cv, scoring='f1_macro', n_jobs=-1)
    grid_lr.fit(X_train, y_train) # Fit on X_train; SMOTE and preprocessing are handled within the pipeline

    print("Training Random Forest Pipeline...")
    grid_rf = GridSearchCV(pipeline_rf, param_grids["Random Forest"], cv=cv, scoring='f1_macro', n_jobs=-1)
    grid_rf.fit(X_train, y_train) # Fit on X_train; SMOTE and preprocessing are handled within the pipeline

    print("Training XGBoost Pipeline...")
    grid_xgb = GridSearchCV(pipeline_xgb, param_grids["XGBoost"], cv=cv, scoring='f1_macro', n_jobs=-1)
    grid_xgb.fit(X_train, y_train) # Fit on X_train; SMOTE and preprocessing are handled within the pipeline

    # Store the best models from each pipeline
    best_models["Logistic Regression"] = grid_lr.best_estimator_
    best_models["Random Forest"] = grid_rf.best_estimator_
    best_models["XGBoost"] = grid_xgb.best_estimator_
```



```
[ ]  import joblib

     # Assuming 'best_models' dictionary contains your trained pipelines
     # best_models["Random Forest"] and best_models["XGBoost"]
     joblib.dump(best_models["Random Forest"], 'random_forest_pipeline.pkl')
     joblib.dump(best_models["XGBoost"], 'xgboost_pipeline.pkl')

     # Also save the LabelEncoder and the X_train_columns for consistent preprocessing
     joblib.dump(le, 'label_encoder.pkl')
     joblib.dump(X_train.columns.tolist(), 'X_train_columns.pkl')

     # Save the accuracy scores
     joblib.dump(val_accuracy_rf, 'rf_validation_accuracy.pkl')
     joblib.dump(val_accuracy_xgb, 'xgb_validation_accuracy.pkl')

     print("Models, LabelEncoder, X_train_columns, and Validation Accuracies saved successfully!")
```