A Machine Learning Approach for Predicting Particulate Matter Concentration in Nairobi County

By

Christine Lorraine Eyinda

148454

Submitted in Partial Fulfilment of the Requirements for the Degree of Master of Science in Data Science and Analytics at Strathmore University

Institute of Mathematical Sciences

Strathmore University

Nairobi, Kenya

June, 2025

Declaration and Approval

Declaration

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the dissertation contains no material previously published or written by another person except where due reference is made in the dissertation itself.

Student's Name: **Christine Lorraine Eyinda**

Sign: _____ Date: 26/05/2025

Approval

The dissertation of **Christine Lorraine Eyinda** was reviewed and approved by the following:

Dr. John Olukuru,
Head of Data Science and Analytics
iLabAfrica, Strathmore Univeristy.

Dr. Kennedy Senagi,
Lecturer, Insititute of Mathematical Sciences,
Strathmore Univeristy.

Dr. Godfrey Madigu,
Dean, Institute of Mathematical Sciences,
Strathmore University.

Prof. Bernard Shibwabo,
Director of Graduate Studies,
Strathmore University.

Abstract

Nairobi experiences elevated particulate matter($PM_{2.5}$) concentrations due to motor vehicle emissions, biomass fuel use, industrial processes, and open waste burning, leading to significant public health risks. This study developed a predictive tool for $PM_{2.5}$ concentrations in Nairobi using machine learning, integrating real-time weather data and historical pollution trends. Using the CRISP-DM framework, the research incorporated data from low-cost air quality sensors, supplemented by meteorological data from OpenWeatherMap. After rigorous preprocessing—such as missing value imputation, outlier treatment, and resampling to an hourly interval—exploratory data analysis revealed diurnal and seasonal patterns, with peak pollution levels observed during dry seasons and weekday rush hours. Correlation analysis showed weak negative relationships between $PM_{2.5}$ and temperature ($r = 0.07$), wind speed ($r = 0.10$), and dew point ($r = 0.05$), indicating pollutant dispersion under warmer, windier conditions. Minor positive correlations with pressure ($r = 0.03$) and humidity ($r = 0.01$) suggested that stable atmospheric conditions slightly increase $PM_{2.5}$ levels. Other factors like visibility, rainfall, and wind direction had minimal impact. Spatial analysis identified significant hotspots along high-traffic corridors, notably the Nairobi Expressway/Southern Bypass ($z = 2.80$, $p = 0.016$) with average $PM_{2.5}$ of 26.01 $\mu g/m^3$, and coldspots in higher elevation areas like Kilimani ($z = 2.61$, $p = 0.001$), where $PM_{2.5}$ averaged below 12 $\mu g/m^3$. This reflects the combined influence of local emissions and topography on pollution distribution. A LightGBM quantile regression model achieved strong predictive performance (RMSE: 2.821, R^2: 0.915). A Streamlit web app was developed to provide interactive forecasts and AQI categorizations, offering a valuable tool for air quality management and public health planning in Nairobi.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AAP Ambient Air Pollution

AOD Aerosol Optical Depth

AQI Air Quality Index

ANN Artificial Neural Networks

BAM-1022 Beta Attenuation Monitor

BC Black Carbon

CAAQMS Continuous Ambient Air Quality Monitoring Stations

CO Carbon Monoxide

Conditional GANs Conditional Generative Adversarial Networks

ConvLSTM Convolutional Long Short-Term Memory

EDA Exploratory Data Analysis

ESRI Environmental Systems Research Institute

eCO Equivalent Carbon Monoxide

GAN Generative Adversarial Network

GCN Graph Convolutional Network

GRU Gated Recurrent Unit

HAP Household Air Pollution

LME Linear Mixed Effects

LR Linear Regression

LightGBM Light Gradient Boosting Machine

LSTM Long Short-Term Memory

MAE Mean Absolute Error

MODIS Moderate Resolution Imaging Spectroradiometer

NOx Nitrogen Oxide

OC Organic Carbon

O3 Ozone

PM Particulate Matter

RF Random Forest

RMSE Root Mean Squared Error

ROS Reactive Oxygen Species

SS2.5 Sea Salt

SO2 Sulfur Dioxide

SO4 Sulphates

SVR Support Vector Regression

TPS Typical Particle Size

TVOC Total Volatile Organic Compounds

VOCs Volatile Organic Compounds

WHO World Health Organisation

XGBoost Extreme Gradient Boosting

Acknowledgments

I offer my sincerest gratitude to God for His unwavering guidance and strength throughout this journey. It is through His grace that I have been able to complete this project.

I am deeply grateful for the guidance and supervision provided by my esteemed supervisor Dr. John Olukuru throughout this journey. I would also like to convey my heartfelt appreciation to my supervisor and lecturer, Dr. Kennedy Senagi, for his important support and supervision throughout this research. His experience and insights have helped shape the direction and success of my research.

My gratitude extends to Strathmore University, particularly the Institute of Mathematical Sciences (SIMS), and the @iLabAfrica research centre. I am especially thankful to @iLabAfrica for the financial support provided, which played a vital role in enabling me to complete my studies. Their commitment to delivering world-class education and imparting practical and technical knowledge has been invaluable to my academic and professional growth.

Dedication

This dissertation is lovingly dedicated to my dear parents,whose boundless love and silent sacrifices have lit my path. Their unwavering guidance and steadfast support have been the quiet pillars beneath my dreams. To my grandmother and grandfather,whose tender hearts and enduring faith have been my refuge and my inspiration through every step of this journey. By their enduring love, I find the roots that ground me and the wings that help me soar.

Chapter 1: Introduction

1.1 Background

Public health and environmental sustainability are at risk due to air pollution, which is a significant global health concern (Racheeti, 2024). Even while our understanding of its negative impacts has advanced significantly, its impact is still growing and need immediate attention (Racheeti, 2024). There are two types of air pollution: ambient air pollution (AAP) and household air pollution (HAP).

HAP causes millions of early deaths every year and is a major worldwide health risk, especially in developing nations (Kaba et al., 2019). This kind of pollution is common in countries with limited resources, particularly when biomass fuels made from crop waste, animal manure, wood, or charcoal are used for domestic energy (Kaba et al., 2019). The resulting HAP contains contaminants such as particulate matter and carbon monoxide, which contribute to both HAP and AAP (J, 2019). HAP affects all life stages, from preconception to old age, with multi-systemic health implications on the respiratory, cardiovascular, endocrine, and neurological systems (Apte and Salvi, 2016). HAP can be up to ten times worse than outdoor air pollution due to pollutant accumulation in enclosed places(Bhole and Mesham, 2017). By promoting a shift to cleaner fuels and lowering household air pollution, recent initiatives like the Global Alliance for Clean Cookstoves offer hope for complete solutions to this critical health issue (Amegah and Jaakkola, 2016).

AAP consists of a complex mixture of thousands of components (Mannucci and Franchini, 2017). Among these, airborne particulate matter (PM) and gaseous pollutants, including ozone, nitrogen dioxide (NO), volatile organic compounds (such as benzene), carbon monoxide (CO), and sulfur dioxide (SO), pose the greatest health risks. Fossil fuel combustion releases primary pollutants into the atmosphere, such as soot particles and sulfur and nitrogen oxides (Mannucci and Franchini, 2017). Power generation, domestic heating, industrial operations, and motorized road traffic are the main sources of these primary particles. When primary pollutants react in the atmosphere, secondary pollutants like PM and ozone are created (Mannucci and Franchini, 2017).

According to the World Health Organization (WHO), 6.7 million premature deaths occur

each year as a result of ambient and household air pollution (Guo et al., 2020). An estimated 4.2 million premature deaths were attributed to AAP globally in 2019. This resulted from exposure to fine particulate matter, which is linked to respiratory and cardiovascular conditions as well as cancers (Guo et al., 2020).

Significant variations in pollution-related risk factors were found among various age groups, according to the Global Burden of Disease 2021 assessment (Metrics, 2024). For children under five, household air pollution from solid fuels is the fourth leading cause, signifying considerable health hazards linked to the utilization of solid fuels for cooking and heating. Ambient particulate matter pollution also posed a serious concern for this age group, ranking eighth. In contrast, among older adults, ambient particulate matter pollution ranked fourth, underscoring its severe impact on health, while household air pollution from solid fuels remained significant, ranking eighth overall. These results demonstrate how pollution has a negative impact on people of all ages and adds considerably to the overall burden of disease (Metrics, 2024). Ischemic heart disease and stroke alone were responsible for nearly 37% of premature deaths caused by outdoor air pollution in 2019, according to WHO estimates, while 18% and 23% of the deaths were caused by acute lower respiratory infections and chronic obstructive pulmonary disease, respectively (Guo et al., 2020).

Ajit et al evaluated the air quality in Nairobi, Addis Ababa, and Kampala, with an emphasis on air pollution from transportation. Drivers and commuters, especially those utilizing buses and motorbike taxis, were found to be exposed to dangerously high levels of air pollution in both Kampala and Addis Ababa. In Nairobi, workers near roadside buildings also faced high pollution exposure. This study emphasized that traffic emissions are a significant source of urban air pollution in East Africa, posing health risks to large commuting populations. Air pollution causes about 19,112 deaths in Kenya each year, 6,672 of which are children (Tékouabou et al., 2022). An estimated 40% of Nairobi's $PM_{2.5}$ concentration is caused by motor vehicles, 25% by unlawful dumping and open garbage burning, 15% by dirty biomass fuels, and another 15% by industrial activities (Matara et al., 2024).

Existing prediction models based on machine learning estimate $PM_{2.5}$ concentrations using historical data from specific prediction sites (Medhi and Gogoi, 2024). However, these

models often overlook the spatial relationships between the prediction site and surrounding monitoring locations, resulting in less accurate predictions, particularly in uncertain conditions. As evidence linking air pollution to adverse health outcomes continues to grow, there is an urgent need to develop sustainable and innovative approaches to address this critical issue. This research aims to contribute to the literature on air quality prediction in Nairobi, providing a promising avenue for facilitating timely public health interventions.

1.2 Problem Statement

Air pollution, particularly the presence of fine particulate matter ($PM_{2.5}$), poses a significant global public health risk, contributing to millions of premature deaths annually. In Kenya, this issue is severe with thousands of deaths,including those of children, attributed to air pollution. Nairobi experiences elevated $PM_{2.5}$ concentrations due to vehicular emissions, waste burning, biomass fuel use, and industrial processes. Given the growing evidence linking $PM_{2.5}$ exposure to severe health outcomes, there is an urgent need for innovative solutions to provide accurate predictions in order to facilitate better planning and also inform offsetting $PM_{2.5}$ concentration using carbon credits.

(Zhang et al., 2022) states that regional diversity, temporal correlations, and spatial correlations are the three main challenges to $PM_{2.5}$ prediction. Traditional statistical models, which are frequently used in current research, might not adequately represent the complexity of $PM_{2.5}$ dynamics, particularly in settings that are rapidly urbanizing. Additionally, machine learning models often overlook how $PM_{2.5}$ concentrations at one location are influenced by nearby areas, missing critical spatial relationships that affect air quality. Therefore, for effective public health interventions and to mitigate the negative health effects of air pollution, it is crucial to develop strong, data-driven predictive models that can capture both the spatial and temporal dynamics of $PM_{2.5}$ concentrations, tailored to Nairobi's unique urban environment.

1.3 General Objective

The primary focus of this study was to develop accurate, data-driven predictive models for $PM_{2.5}$ in Nairobi, enabling real-time air quality monitoring and evidence-based environmental policy interventions.

1.4 Specific Objectives

This research aimed to address the following objectives:

(a) To analyse the spatial correlation of $PM_{2.5}$ concentration and meteorological factors.

(b) To fit predictive models for estimating $PM_{2.5}$ concentration.

(c) To compare the performance of different machine learning algorithms in predicting $PM_{2.5}$.

(d) To optimize the best performing model

(e) To deploy the best model.

1.5 Research Questions

The research questions addressed in this study were as follows:

(a) What is the spatial relationship of $PM_{2.5}$ and meteorological factors?

(b) What is the best approach to developing a predictive model for $PM_{2.5}$ concentration?

(c) Which machine learning algorithms perform best in predicting $PM_{2.5}$ concentrations, and how do their performances compare?

(d) Which parameters can optimize the performance of the best machine learning algorithm for $PM_{2.5}$ prediction?

(e) Which deployment strategies are most effective for implementing the trained machine learning model?

1.6 Scope of the Study

The scope of this study will focus on the predictive modeling of $PM_{2.5}$ concentrations in Nairobi, Kenya. The research intends to explore key spatial relationships and develop predictive models. A potential limitation of this study will be the limited availability of real-time data.

1.7 Research Justification

The ability to accurately predict $PM_{2.5}$ concentrations represents a critical advancement beyond conventional weather forecasting, addressing a pressing environmental health challenge. Unlike meteorological predictions that track atmospheric conditions, $PM_{2.5}$ modeling specifically accounts for the complex interplay between weather patterns and anthropogenic emission sources,including vehicular traffic, industrial operations, and biomass burning,that disproportionately impact urban air quality. This predictive capacity carries profound implications for public health protection and sustainable urban development.

By generating localized, high-resolution forecasts, these models enable timely interventions such as targeted pollution advisories for vulnerable populations and dynamic traffic management during peak exposure periods. Furthermore, they provide an evidence base for long-term policy decisions regarding emission controls, land use planning, and environmental regulation. In Nairobi's rapidly urbanizing context, where traditional monitoring approaches struggle to capture the city's unique pollution dynamics, this research offers a vital tool for mitigating the substantial disease burden attributed to airborne particulate matter while supporting the development of cleaner, healthier urban environments.

Chapter 2: Literature Review

This chapter reviews the theoretical and empirical literature related to the prediction of $PM_{2.5}$ concentrations, with a focus on health impacts and the application of machine learning models.

2.1 Theoretical Literature Review

Air pollution, according to the World Health Organization, is the contamination of indoor or outdoor spaces by physical, chemical, or biological elements that change the atmosphere's inherent properties. Air pollution is defined as a composition of harmful substances of anthropogenic and natural origin that alter the natural characteristics of the atmosphere to render it harmful to human welfare(Mudu et al., 2023). Thinking about the various kinds of contaminants that are in the atmosphere is one approach to comprehend air pollution. These can be categorized into two primary types: PM and gaseous pollutants(Maknae, 2024).

Dust, smoke, soot, pollen, and metals are examples of solid or liquid particles suspended in the air, collectively known as PM (Maknae, 2024). PM comes in a variety of sizes, but the most dangerous are those with diameters less than 10 micrometers (PM_{10}) or 2.5 micrometers ($PM_{2.5}$). These microscopic particles can penetrate deeply into the lungs, causing cancer, cardiovascular and respiratory disorders, and premature death (Maknae, 2024). This provides the justification of predicting $PM_{2.5}$ implemented in this study.

Gases that are regarded as contaminants include ozone (O_3), nitrogen dioxides (NO_2), sulfur dioxide (SO_2), carbon monoxide (CO), and volatile organic compounds (VOCs) (Maknae, 2024). When they combine with one another and with other particles in the air, secondary pollutants such as smog and acid rain can be formed. Moreover, gaseous pollutants can cause damage to the central nervous system, respiratory and cardiovascular systems, skin, and eyes (Maknae, 2024).

2.1.1 Air Pollution and Human Health

The concept of human health is complex and includes social, mental, and physical well-being. It's influenced by various factors, including our environment, socioeconomic status, lifestyle choices, and access to healthcare(Wongpakdee et al., 2023). As discussed in the

previous section, one factor that poses a threat on human health is air pollution. Several theoretical frameworks have been developed to explain the pathways through which air pollution, particularly $PM_{2.5}$, impacts human health. These models provide a foundation for understanding the mechanisms behind pollutant-induced health outcomes and inform the empirical approach of this study.

The Particulate Matter theory, first proposed by Dr. Richard P. Poirot in 2007, describes how $PM_{2.5}$, which are tiny particles in the air, enter the respiratory system and cause ailments like bronchitis, asthma, and cardiovascular disorders (Maknae, 2024). This theory provides a strong empirical justification for forecasting $PM_{2.5}$ as anticipating high pollution episodes can support early warnings and targeted policy interventions. According to Dr. Sarah L. Stevenson's Oxidative Stress Theory, pollutants such as ozone and nitrogen oxides cause oxidative stress, which results in reactive oxygen species (ROS) that harm cells and cause neurological, cardiovascular, and respiratory condition (Maknae, 2024). Furthermore, according to Dr. Maria T. Hernandez's 2015 Inflammatory Response Theory, air pollutants cause a systemic inflammatory response, with immune cells and pro-inflammatory cytokines being crucial in the development of respiratory disorders, lung damage, and cardiovascular issues(Maknae, 2024).

These theoretical perspectives inform the present study by supporting the relevance of monitoring and predicting $PM_{2.5}$ levels. Recurring elevations in $PM_{2.5}$ could plausibly trigger these biological responses, thereby increasing public health risks.

2.1.2 Factors influencing Particulate Matter Concentration

Recent studies have explored various factors influencing $PM_{2.5}$ concentrations, particularly in urban environments. The geographic detector approach was used in a study done in China to evaluate how weather and anthropogenic precursors affected $PM_{2.5}$ levels in various cities (Jing et al., 2020). Significant spatiotemporal differences were found, and the main affecting factor, temperature, was found to account for 27% of $PM_{2.5}$ concentrations per year. In southern China, precipitation was found to be a significant factor, whereas in northern China, temperature was more important. In winter, anthropogenic precursors, especially ammonia, had a stronger impact on $PM_{2.5}$ levels in northern China than in other seasons, highlighting seasonal variability in pollutant behavior. The study

emphasized the interaction between factors, with the combination of ammonia and temperature explaining up to 46% of $PM_{2.5}$ concentrations. These results underscored the complex interplay between meteorological conditions and human activities, providing a valuable foundation for understanding the drivers of $PM_{2.5}$ formation and informing targeted pollution control strategies(Jing et al., 2020).

A comprehensive study examined the spatial-temporal trends of $PM_{2.5}$ pollution in Africa's subregions and the continent as a whole from 1980 to 2021. The study analyzed monthly aerosol species concentrations from MERRA-2 reanalysis datasets, including sulphates (SO_4), organic carbon (OC), black carbon (BC), $Dust_{2.5}$, and sea salt (SS2.5) (Ouma et al., 2024). The findings indicated a positive trend in $PM_{2.5}$ across most of the regions apart from western and central Africa, where small negative trends were observed. Over northern, western, and central Africa, $Dust_{2.5}$ was the prevailing aerosol contributor, whereas over eastern and southern Africa, SO_4 and OC were the major contributors, respectively. The study discovered that $PM_{2.5}$ trends were strongly connected to wind speed, whereas precipitation and temperature exhibited weak or negative associations. Socioeconomic characteristics also suggested that locations with a high population density and little plant cover had high $PM_{2.5}$ concentrations, whereas vegetated areas had low levels. Topographically, low-lying basins accumulated deposited $PM_{2.5}$, especially in northern and western Africa (Ouma et al., 2024).

The sources and seasonal patterns of fine particulate matter ($PM_{2.5}$) in urban Sub-Saharan Africa are not well understood. A study in Nairobi from 2008 to 2010 measured $PM_{2.5}$ levels at an urban and suburban site (Gaita et al., 2014). The average $PM_{2.5}$ concentration was found to be $21 \pm 9.5, \mu g/m^3$ in the urban area and $13 \pm 7.3, \mu g/m^3$in the suburban area(Gaita et al., 2014). On 29% of days at the urban site and 7% at the suburban site, daily $PM_{2.5}$ levels were higher than the WHO 24-hour recommendation. About 80% of the components found were black carbon, iron, sulfur, and chlorine. Five sources of $PM_{2.5}$ were found via positive matrix factorization analysis: traffic, industry, combustion, mineral dust, and a mixed factor of biomass burning, secondary aerosol, and aged sea salt. Of these, traffic and mineral dust accounted for almost 74% of concentrations. Seasonal variations were observed in most sources, except for traffic, and weekdays had higher $PM_{2.5}$ levels.

2.2 Empirical Literature Review

This section reviews empirical research on various machine learning architectures, types of data and general approaches employed in previous studies for predicting $PM_{2.5}$ concentrations.

2.2.1 Models and Data used for $PM_{2.5}$ prediction

Conventional prediction models for air pollutants primarily rely on deterministic and statistical methods. Deterministic models, which rely on historical data to simulate emission and diffusion processes, often struggle with incomplete knowledge and insufficient data, resulting in imprecise and unstable predictions(Ding et al., 2021). Statistical methods, while commonly used, can also face challenges in accurately predicting air pollution due to the complex and variable nature of time-series air quality data(Morapedi and Obagbuwa, 2023). To address these limitations, machine learning algorithms have emerged as powerful tools for improving air pollution prediction accuracy.

In order to address concerns about air quality near heavy traffic, Matara's study (Matara et al., 2024) sought to categorize particulate matter ($PM_{2.5}$) concentrations along the Nairobi Expressway into "Acceptable" or "Unacceptable" categories based on EPA guidelines. In addition to atmospheric temperature and humidity, the dataset included PM1, $PM_{2.5}$, PM4, PM10, Typical Particle Size (TPS), Total Volatile Organic Compounds (TVOC), and equivalent carbon monoxide (eCO). $PM_{2.5}$ levels surpassing 12 $\mu g/m^3$ were considered undesirable. Air quality was classified using a number of machine learning classifiers, including XGBoost, Random Forest, Extra Trees, K-Nearest Neighbor, Naive Bayes, and Binary Logistic Regression. XGBoost performed the best with an F1-score = 0.455. Feature selection was conducted out using the Boruta Algorithm with Random Forest, which indicated humidity as the most important variable impacting $PM_{2.5}$ concentrations. Although it offered insightful information on air quality monitoring, the study did not use satellite imagery and instead relied on real-time data from Open-Seneca sensors, which were collected at one-second intervals. However, the research demonstrated the capability of simple machine learning models in environmental analysis. Real-time data application and innovative feature selection methods are strong points, yet a closer examination of differences in model performance and study expansion beyond the ex-

pressway may enhance the contribution of the study.

(Morapedi and Obagbuwa, 2023) evaluated $PM_{2.5}$ levels in South African cities, emphasizing their effects on public health and air quality. In order to analyze pollution trends, the dataset contained important characteristics such as location, population statistics, and $PM_{2.5}$ concentrations. $PM_{2.5}$ levels were predicted using machine learning models such as Cat Boost Regressor and Extreme Gradient Boosting Regressor, while the Air Quality Index (AQI) status was predicted using K-Nearest Neighbor, Logistic Regression, Support Vector Machine, Decision Tree, and Random Forest Classifier. Random Forest and Decision Tree performed best for AQI, while the Cat Boost Regressor was found to be the most effective for $PM_{2.5}$ prediction. The findings are significant for informing environmental health policies and also demonstrated the capability of simple machine learning algorithms in environmental analysis. However, limitations include generalizability to other regions.

The work by (Njeru et al., 2023) tackled the crucial problem of forecasting $PM_{2.5}$ levels in Nairobi City, which is necessary to comprehend urban air quality and its consequences for public health. The dataset contained factors like meteorological data, ground $PM_{2.5}$ measurements from five locations utilizing calibrated Alpha Sense OPC-N2 sensors, and satellite aerosol optical depth (AOD) data. Simple linear regression, multiple linear regression, and a linear mixed effects (LME) model were the three machine learning techniques used. The LME model gave the best performance, achieving an RMSE of 4.797. The results indicated high concentrations of $PM_{2.5}$ in the vicinity of traffic roads, urban areas, industrial areas, and informal settlements, and lower concentrations in the proximity of forest areas. This research highlights the potential of using 1 km^2 resolution satellite AOD for reliable daily $PM_{2.5}$ predictions.The strength of the study includes the comprehensive dataset and the LME model's ability to capture daily variability in $PM_{2.5}$ concentrations. However, traditional statistical methods often produce poor estimates due to the complexities of time-series data.

The study by (Muthukumar et al., 2022) addressed the challenging problem of forecasting $PM_{2.5}$ levels in the broader Los Angeles County area. With a focus on satellite images of nitrogen dioxide, which plays a major role in the chemical events that lead to $PM_{2.5}$ generation, the dataset comprised variables such satellite data, meteorological

data, and ground-based sensor data.They employed a Convolutional Long Short-Term Memory (ConvLSTM) model to learn both spatial and temporal correlations in remote-sensing and satellite data after learning spatial correlations in meteorological data using a Graph Convolutional Network (GCN).The outcomes showed how well this combination strategy worked to solve the spatiotemporal difficulties in $PM_{2.5}$ prediction. The strength of the study is seen in the creative use of temporal and geographical correlations, which improves prediction accuracy by capturing the intricate relationships between environmental factors throughout time and place. However, ConvLSTMs demand large volumes of spatiotemporal data and are computationally intensive making them less practical for deployment in resource-limited contexts(Muthukumar et al., 2022).

Aldaweesh's study employed four machine learning models—Support Vector Regression (SVR), Linear Regression (LR), Random Forest (RF), and Artificial Neural Networks (ANN)—to estimate hourly particulate matter concentrations ($PM_{2.5}$) in China(AlDaweesh, 2019). Reliable stations, such as U.S. diplomatic installations, provided the meteorological data, which included temperature, humidity, wind speed, pressure, and $PM_{2.5}$ levels. Two methods were used to test the models: one used data from a single city for training and testing, while the other used data from several cities to generalize model performance. With an R-squared value between 0.67 and 0.78, Random Forest fared better than the other models, whereas Linear Regression did the poorest because of the poor correlations between $PM_{2.5}$ and other characteristics. In order to reduce dimensionality, the dew point variable was eliminated, and moderate relationships were discovered with pressure, wind speed, and seasonality. The study came to the conclusion that, despite their high processing costs, sophisticated models such as Random Forest and Neural Networks are better suited for forecasting $PM_{2.5}$ when interactions are non-linear. The poor performance of Linear Regression demonstrated its limitations in complicated environmental predictions(AlDaweesh, 2019).

In the study by (Zamani Joharestani et al., 2019), the main goal was to find important characteristics for forecasting $PM_{2.5}$ levels in Tehran's urban area using machine learning techniques like deep learning, Random Forest (RF), and Extreme Gradient Boosting (XGBoost). Feature permutation was employed for deep learning, even though RF and XGBoost come with built-in techniques for feature importance detection. Furthermore,

XGBoost was used in a recursive feature elimination, with Mean Absolute Error (MAE) acting as the standard for eliminating superfluous features. Historical $PM_{2.5}$ data, wind speed, visibility, temperature, height, and day of the year were important variables. XGBoost achieved an R-squared of 0.81, MAE of 9.92 µg/m^3, and RMSE of 13.58 µg/m^3, outperforming both RF and deep learning models. In order to forecast $PM_{2.5}$ concentrations, the study combined meteorological and ground-based climatic data with satellite Aerosol Optical Depth (AOD) data. Interestingly, while AOD was initially included as a key feature, the study concluded that AOD did not significantly enhance $PM_{2.5}$ prediction.

According to a recent study by (Medhi and Gogoi, 2024), the Prob $PM_{2.5}$ model predicts $PM_{2.5}$ concentrations using Conditional Generative Adversarial Networks (GANs) and data from Continuous Ambient Air Quality Monitoring Stations (CAAQMS) in Delhi and Guwahati. The model takes into account pollutants like NO2, SO2, and $PM_{2.5}$ as well as climatic factors like temperature, wind speed, and relative humidity. Following a correlation study between these features and $PM_{2.5}$, adversarial training was carried out using the Conditional GAN framework. In contrast to more conventional models such as LSTM and GRU, Prob $PM_{2.5}$ showed better performance.

Although Conditional GANs have demonstrated strong capabilities in modeling the probabilistic distribution of $PM_{2.5}$ data, particularly by capturing nonlinear and high-dimensional relationships (Medhi and Gogoi, 2024), their practical application in real-world settings remains constrained. These models require substantial computational resources, careful hyperparameter tuning, and are prone to training instability. In contrast, ensemble models such as Random Forest, XGBoost, and LightGBM offer robust performance while being easier to implement and interpret as illustrated by (Zamani Joharestani et al., 2019; Morapedi and Obagbuwa, 2023).

In the context of this study, where resource efficiency, scalability, and model transparency were essential for informing local policy and public health decisions, ensemble methods presented a more practical solution. By adopting ensemble tree-based approaches, the study achieved a balance between predictive accuracy and operational feasibility, without sacrificing interpretability critical for policy advocacy.

2.2.2 Dashboard Deployment Strategies

The literature on dashboards used for air quality monitoring is reviewed in this section.(Jing et al., 2019) categorizes dashboards into three types based on their role: operational, analytical, and strategic. When incorporating geospatial data, these categories can be further defined: Operational Dashboards provide descriptive measurements of a smart city using indicators derived from original geospatial data and other relevant data. These indicators offer insights into the current state of the city; Analytical Dashboards utilize spatial analytics to infer patterns and relationships from geospatial data, providing a diagnostic tool for smart cities; Strategic Dashboards which are predictive dashboards that utilize existing patterns and input data into models to forecast the future state of a smart city.

(Hananto and Putra, 2018) developed a dashboard system for monitoring $PM_{2.5}$ levels. The building of Surabaya college, Stikom, was equipped with a $PM_{2.5}$ sensor installed with an Airbox Edimax device. For the control operations, data acquisition, and data transmission to the cloud, the device was connected to a Wi-Fi connection. Air quality measuring data was collected and stored in the cloud via an open data platform. Then, within a week of the study, the data was successfully collected in JSON format. A dashboard system was developed to monitor the level of air pollution from this data. The aim of this dashboard was to provide users at Stikom Surabaya with the capability to monitor the air quality within the college building. From the dashboard system visualization, it was noted that the $PM_{2.5}$ level varied throughout the day.

The research of (Bachechi et al., 2020) developed the TRAFAIR Air Quality dashboard which was deployed to visualize and share air quality data for the city of Modena. The dashboard offered a collection of visual analytics, including dynamic and interactive graphics to show and convey data on the city's air quality state in real time, as well as statistics and trends. The dashboard's main features were its ability to handle massive amounts of data streaming in from multiple sensors and having both spatial and temporal dimensions; its ability to process complex data, including time series maps produced by models or interpolation processes; and its scalability and ease of portability to other cities.

2.3 Research Gaps

The literature review reveals extensive research on particulate matter prediction and the use of machine learning algorithms for predictive analytics. However, significant gaps remain, particularly in the application of advanced models for $PM_{2.5}$ prediction in African countries and the incorporation of spatial correlations.

The study by (Matara et al., 2024) exhibits notable limitations, including a lack of spatial correlation analysis, which could enhance understanding of how $PM_{2.5}$ concentrations vary across different geographical areas, particularly in urban settings like Nairobi. Additionally, the research is geographically constrained to the Nairobi Expressway, potentially overlooking broader regional pollution patterns and sources. Furthermore, the study did not incorporate advanced modeling techniques that could improve prediction accuracy and provide a more comprehensive analysis of air quality dynamics. Addressing these gaps could lead to more effective air quality management strategies and a better understanding of particulate matter distribution in urban environments.

In the study by (Njeru et al., 2023), notable gaps were identified. The research relied solely on statistical models, which, as noted by (Morapedi and Obagbuwa, 2023), often produce poor estimates, particularly in time series data. Moreover, it did not account for the spatial correlation between meteorological factors and $PM_{2.5}$ concentrations. (Morapedi and Obagbuwa, 2023) further emphasized that future research could leverage real-time data analysis through cloud computing to enhance model performance. Evaluating models on larger datasets, such as Nitrogen Dioxide (NO_2), Ozone (O_3), Sulfur Dioxide (SO_2), and Carbon Monoxide (CO), and adding deep learning and ensemble techniques for $PM_{2.5}$ and PM10 prediction would make the analysis more robust.

2.4 Conceptual Framework

The conceptual framework is a structured representation of the key principles, variables, and relationships guiding this study, derived from interdisciplinary academic foundations in environmental science, machine learning, and data engineering as illustrated in Figure 2.1. It suggests that $PM_{2.5}$ concentration is dynamically influenced by meteorological factors (e.g., temperature, humidity) and anthropogenic pollution sources, which collectively form the input domain.

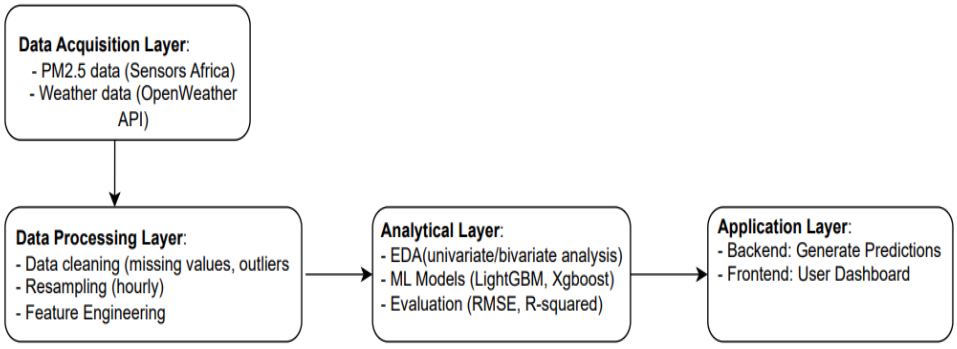


Figure 2.1: Conceptual framework

This framework operationalizes these relationships through a four-layer pipeline: (1) Data Acquisition, integrating sensor measurements ($PM_{2.5}$, weather) with OpenWeather API forecasts; (2) Data Processing, involving resampling, outlier removal, and feature engineering (e.g., lag features); (3) Analytical Modeling, where ensemble techniques (LightGBM, XGBoost, stacking) predict $PM_{2.5}$ with uncertainty quantification; and (4) Application, delivering actionable insights via visualizations and confidence intervals. By grounding this workflow in CRISP-DM methodology and feature-based time-series forecasting, the framework not only clarifies causal linkages but also ensures reproducibility and scalability for air quality prediction tools.

Chapter 3: Methodology

This study used a unique methodology based on the Cross-Industry Standard Process for Data Mining (CRISP-DM) framework. The CRISP-DM methodology offers a structured and iterative process that comprehensively covers all stages of the data science project lifecycle—from understanding business objectives to implementation and evaluation of results (Parganiha et al., 2022). The steps outlined in this chapter are designed to create a robust framework for predicting $PM_{2.5}$ concentrations, ultimately contributing to environmental health and policy formulation.

3.1 Data Understanding

This research utilized data from Sensors Africa, a pan-African citizen science project funded by Innovate Africa. The initiative employs low-cost sensors to monitor air, water, and sound pollution, providing valuable insights for communities (Tékouabou et al., 2022). The open-source sensor data was accessible through Open Africa Datasets and includes measurements from the DHT22 sensor for humidity and temperature, as well as the PMS5003 for particulate matter (PM_1, $PM_{2.5}$, and PM_{10}). The dataset covered the period from January 2018 to March 2024 and focuses on Nairobi, Kenya. This study also incorporated meteorological data which are well-regarded in air pollution research (Zamani Joharestani et al., 2019).

3.2 Data Preparation

The data preparation stage addressed the challenges posed by using multiple data sources. We systematically cleaned the data to transform it from its raw state into a high-quality format suitable for modeling.This process included handling missing values through appropriate imputation techniques and addressing outliers to ensure data integrity. To ensure regular intervals for time series data, the data was resampled to an hourly interval. The data was then merged with hourly weather data obtained from Open Weather.

3.3 Exploratory Data Analysis

This study conducted a comprehensive exploratory data analysis (EDA). This step involved univariate analysis, which focused on examining a single variable to understand

its distribution and characteristics, such as mean, median, and mode. Bivariate analysis was employed to explore the relationship between two variables, helping to identify correlations or associations. EDA provided insights into the underlying trends in the data and helped refine the model by highlighting important features crucial for the prediction task. Additionally we analysed the spatial correlation of $PM_{2.5}$ and meteorological factors. The pearson correlation coeffecient was used with its formula given in Equation 1.

$$r = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum(X_i - \bar{X})^2 \cdot \sum(Y_i - \bar{Y})^2}} \tag{1}$$

In addition, this study utilized the Getis–Ord Gi* statistic to identify $PM_{2.5}$ hotspots in Nairobi . The Gi* statistic uses Equation 2 as mentioned by Environmental Systems Research Institute (ESRI) to identify the hotspots areas (Hassan and Bhuiyan, 2023).

$$G_i^* = \frac{\sum_{j=1}^{n} w_{i,j}x_j - \bar{X}\sum_{j=1}^{n} w_{i,j}}{s\sqrt{\frac{n\sum_{j=1}^{n} w_{i,j}^2 - \left(\sum_{j=1}^{n} w_{i,j}\right)^2}{n-1}}} \tag{2}$$

where: x_j is the value of feature j; $w_{i,j}$ is the spatial weight between feature i and j; n is the number of features; $\bar{X} = \frac{1}{n}\sum_{j=1}^{n} x_j$; $s = \sqrt{\frac{\sum_{j=1}^{n} x_j^2}{n} - (\bar{X})^2}$

Gi* statistic produces z-scores and p-value. A higher z-score and a small p-value of a cluster signify the hottest spot while a negative z-score and a small p-value present the coldest area (Hassan and Bhuiyan, 2023).

3.4 Data Preprocessing

Before modeling, additional preprocessing was done to make the data ready for machine learning. This steps involves feature selection and feature engineering. Moreover, feature scaling was also performed before fitting the LSTM model.

3.5 Modeling

In this study, we aimed to predict the concentration of $PM_{2.5}$, a crucial indicator of air quality using machine learning algorithms. Figure 3.1 illustrates the machine learning approach.

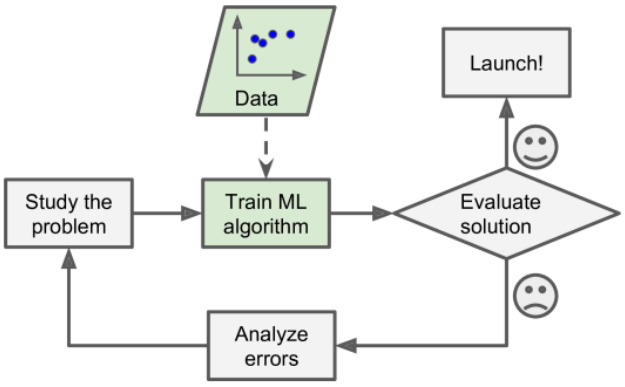


Figure 3.1: Machine Learning Approach

The dataset was split into training, validation and testing datasets to facilitate model evaluation. Since the training set was made up of labeled data justifies the supervised learning approach adopted by this study.

3.5.1 Ensemble Machine Learning Algorithms

This study implemented ensemble methods to predict $PM_{2.5}$. According to (Géron, 2019), the methods integrates several base algorithms to construct better predictive performance than the base algorithms of a single tree. By combining diverse models, ensemble methods can capture a wider range of patterns and make more accurate predictions on unseen data. This study implemented the following ensemble methods: Boosting (Gradient Boosting, Xgboost, LightGBM),Bagging (Random Forest) and stacking ensemble.

Bagging

This approach entails using the same training algorithm for every predictor, but to train them on different random subsets of the training set (Géron, 2019). When sampling is performed with replacement, this method is called bagging. An illustration of how the bagging technique works is illustrated in Figure 3.2.This study adopted the random forest model which is made up of several decision trees and the training set is sampled with replacement hence a bagging technique. The random forest model is defined in Equation 3

$$\hat{y} = \frac{1}{B}\sum_{b=1}^{B} T_b(x) \tag{3}$$

where B is the number of trees, and $T_b(x)$ is the prediction of the b-th tree.

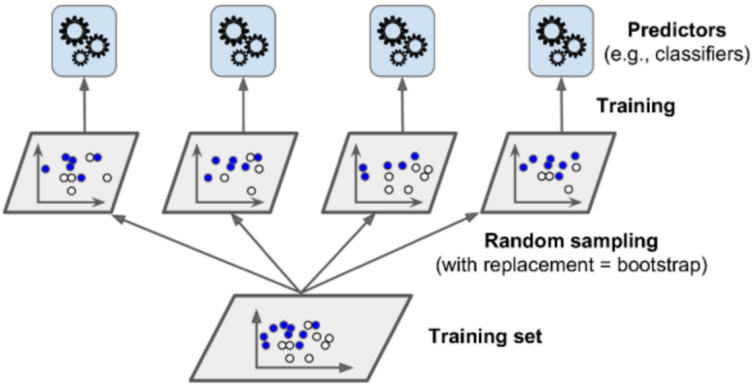


Figure 3.2: Bagging Approach

Boosting

Ensemble learning refers to any method that sequentially combines multiple weak learners (e.g., shallow decision trees) into a strong learner, improving predictive accuracy and robustness (Géron, 2019). Among ensemble techniques, boosting methods iteratively train models to correct errors from previous iterations. This study adopts three state-of-the-art boosting algorithms for $PM_{2.5}$ prediction:Gradient Boosting (GBM),Extreme Gradient Boosting(Xgboost), Light Gradient Boosting Machine(LightGBM). An illustration of how the boosting technique works is shown in Figure 3.3

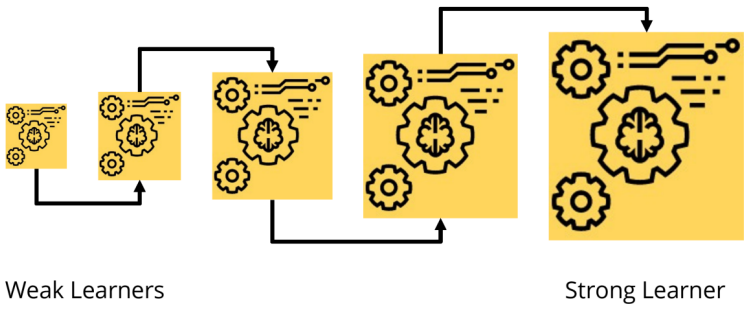


Figure 3.3: Boosting Approach

The respective equations for this models is given below:

(a) **Gradient Boosting**

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x) \tag{4}$$

where $F_m(x)$ is the model at step m, $h_m(x)$ is the weak learner, and γ_m is the step size.

(b) **Xgboost**

$$\mathcal{L}(\phi) = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k) \tag{5}$$

where $\Omega(f_k)$ is the regularization term to prevent overfitting.

(c) **LightGBM**

$$\mathcal{L}(\phi) = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k) \tag{6}$$

where $\Omega(f_k)$ is the regularization term to prevent overfitting.

Stacking Ensemble

This approach combines mutiple regressors via a meta-regressor. Each of the regressors predicts a value and then a final predictor called a meta learner takes these predictions as inputs and makes the final prediction as illustrated in Figure 3.4.The regressors implemented in this study include: Xgboost and LightGBM while the meta-regressor was a linear regression model.

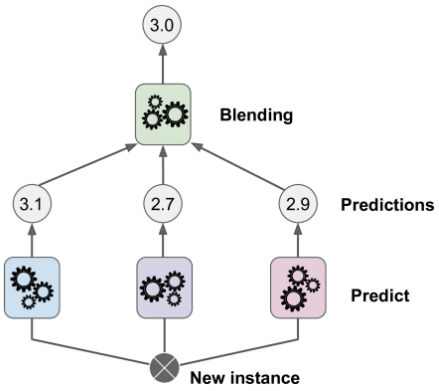


Figure 3.4: Stacking Approach

3.5.2 Neural Network Achitecture

A standard neural network consists of multiple layers of linked nodes, as illustrated in Figure 3.6. All of the neurons in the network possess the characteristics outlined for perceptrons(Géron, 2019). There is also a different number of the hidden layers that are utilized to link the input layers and the output layers. A neural network with one hidden layer is known as an artificial neural network as depicted in Figure 3.5.

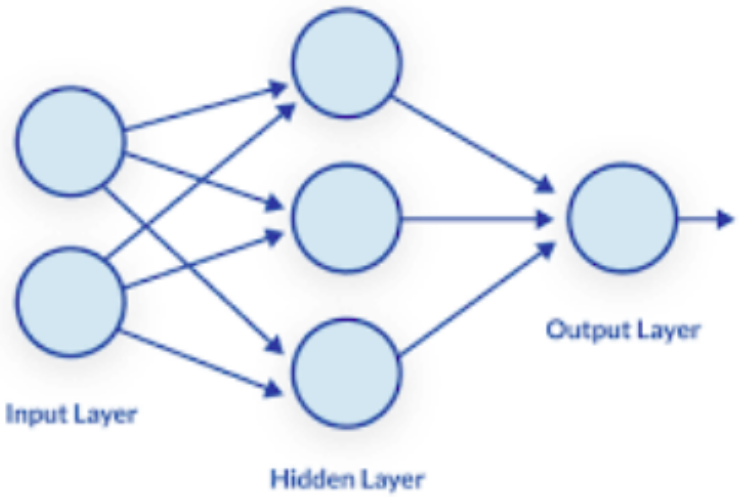


Figure 3.5: Artificial Neural Network Architecture

When the hidden layer is more than one we have the multilayer perceptron(Géron, 2019) as shown in Figure 3.6.

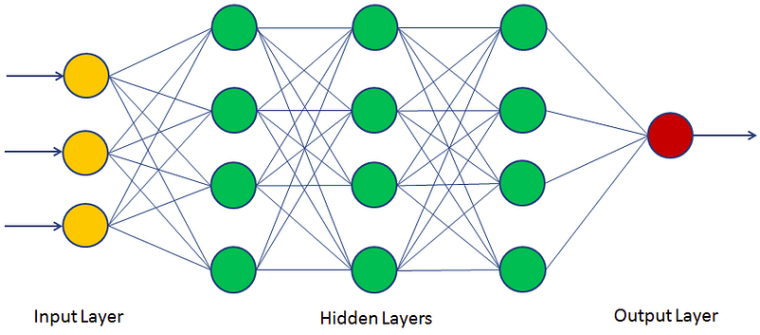


Figure 3.6: Multi-layer Perceptron Architecture

Perceptron and Activation Function

Application of an activation function to a neural network introduces non-linearity. As a result,the neural network is able to learn,classify and model complicated data (Géron, 2019). A Neural Network without an activation function would simply be a linear re-

gression model which has less power to learn complex functional mappings from data. Activation functions should also be differentiable (Géron, 2019).

ReLU Activation function was recently proved to have six times improvement in convergence from Tanh function. The mathematical form of this equation is very simple and efficient as illustrated in Figure 3.7. ReLU avoids and rectifies vanishing gradient problem.

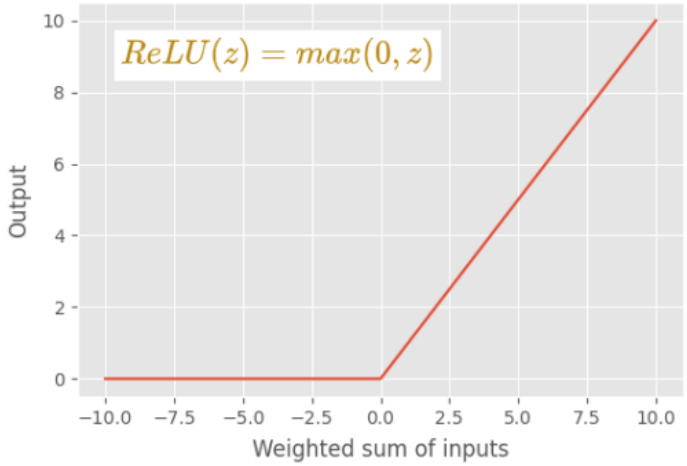


Figure 3.7: ReLU Activation Function

In the first hidden layer, the ReLU activation function is applied to the weighted output from the input layer (Cohen, 2021):

$$R(H_i^{(1)}) = \max(0, H_i^{(1)}) \tag{7}$$

From the first hidden layer into the second layer:

$$H_i^{(2)} = \sum_{j=1}^{n} W_{ij}^{(2)} \cdot R(H_j^{(1)}) + b_i^{(2)} \tag{8}$$

In the second hidden layer, the ReLU activation function is applied:

$$R(H_i^{(2)}) = \max(0, H_i^{(2)}) \tag{9}$$

From the second hidden layer into the output layer:

$$\hat{Y}_i = \sum_{j=0}^{n} W_{ij}^{(3)} \cdot R(H_j^{(2)}) + b_i^{(3)} \tag{10}$$

The output you get is dependent on the type of problem. If it is a classification problem the softmax function is applied (Géron, 2019). The function converts the weighted sum values into probabilities that sum to one.

Long Short-Term Memory (LSTM) networks

This study implemented an LSTM which is an important neural network widely applied in sequential data is the Recurrent Neural Network (RNN). Unlike other neural networks, RNNs focus on the relationships between input and output data. The basic structure of an RNN is shown in Figure 3.8.

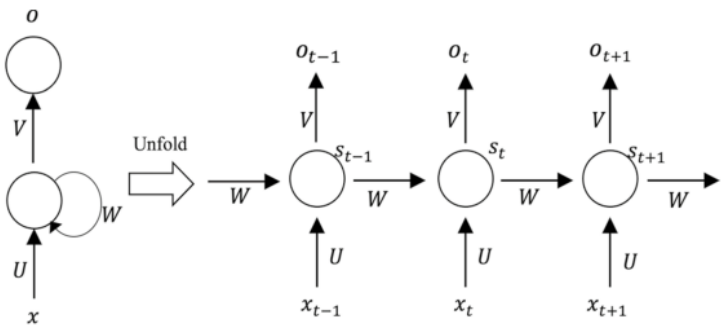


Figure 3.8: The structure of a simple recurrent neural network

As shown in Figure 3.8, x denotes input data, o denotes output data, U represents the weight matrix from the input layer to the hidden layer, V represents the weight matrix from the hidden layer to the output layer, W represents the weight matrix from the hidden layer to itself, and s is the state value of the hidden layer. However, the vanishing gradient problem often occurs during the training process of RNNs, leading to the training parameters being reduced to zero (Ding et al., 2021). To address this issue, the long- and short-term memory (LSTM) model was introduced to mitigate the problem of vanishing gradient. Figure 3.9 shows the specific network structure of the LSTM model.

Unlike the internal structure of RNN, the state of LSTM is controlled by an input gate i_t, a forget gate f_t, and an output gate o_t. Among them, the forget gate is designed to discard information from the memory cell (Ding et al., 2021). The forget gate mechanism receives the output value h_{t-1} from the previous time step and the input value x_t at the

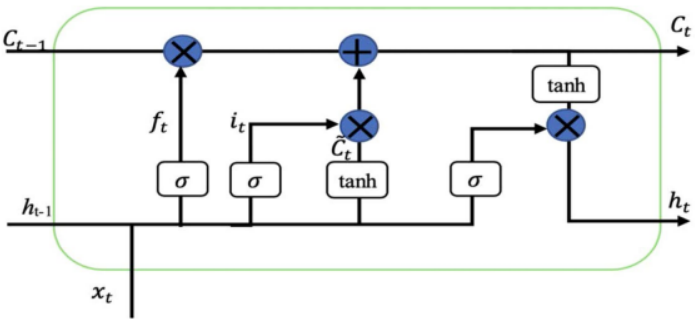


Figure 3.9: The structure of an LSTM model

current time. Then, a probability value C_{t-1} is calculated through the sigma function, which is used to determine the retention of the unit state at the previous time. Also, the input gate is responsible for updating new information to the cell state. Specifically, the probability of state update is controlled according to the output value of the σ function, and then a new input value C_t is generated through the tanh function. The output gate controls the output of the external state h_t according to the internal state C_t at the current time. The specific process can be described in Equation 11 - Equation 16.

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{11}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{12}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{13}$$

$$\tilde{C}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{14}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{15}$$

$$h_t = o_t \odot \tanh(C_t) \tag{16}$$

where W_f, W_i, W_o, and W_c represent the weight matrices for input vector x_t. U_f, U_i, U_o, and U_c denote the weight matrices from the previous state to the hidden state. b_f, b_i, b_o, and b_c are bias weights. $\odot$ represents the multiplication of the matrix. x_t is the input vector at time t. h_t denotes the output vector at time t. C_t represents the cell status at time t.

3.6 Evaluation and Optimization

The evaluation of model performance utilized metrics specific to regression problems. We propose to adopt the following evaluation metrics:

3.6.1 Root Mean Squared Error(RMSE)

The standard deviation of the residuals (errors) between the expected and actual values is known as the RMSE, and it is provided in Equation 17. According to (Muinde et al., 2023), it is currently calculated by finding the square root of the mean of the squared residuals. Good model performance is indicated by a lower RMSE, while poor model performance is shown by a greater RMSE.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2} \tag{17}$$

3.6.2 R-squared

R-squared, formulated in Equation 18, assesses the percentage of variance in the predicted variable that is accounted for by the predictor independent variables (Muinde et al., 2023).

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{18}$$

The evaluation process was implemented using Python, with the assistance of key libraries for machine learning and data analysis. Specifically, the Scikit-learn library was used for model evaluation, regression analysis, and hyperparameter tuning. This library provides a comprehensive set of functions to compute the metrics discussed above. Additionally, hyperparameter optimization via Random Search was used to refine the models further

(Géron, 2019). By implementing this evaluation in Python with Scikit-learn, we achieved robust and optimized models.

3.7 Deployment

This research implemented a dashboard using a web application via Streamlit. Additionally, interactive visualizations were incorporated to enhance user engagement and data interpretation within the dashboard.

Chapter 4: System Design and Architecture

4.1 Introduction

This chapter provides a thorough explanation of the Nairobi $PM_{2.5}$ Prediction Tool's system architecture and design. The $PM_{2.5}$ Prediction Tool is a web-based application that provides 24-hour air quality prediction for Nairobi, Kenya. The system combines:

1. Machine learning models (LightGBM with quantile regression)
2. Real-time weather data integration (OpenWeatherMap API)
3. Interactive visualization capabilities
4. User-friendly interface with input validation

4.2 System Overview

The system architecture for the Nairobi $PM_{2.5}$ Prediction Tool shown in Figure 4.1 illustrates the workflow of the air quality forecasting system. It is designed for optimal efficiency and user accessibility, beginning with a Streamlit-based web interface where environmental analysts input historical $PM_{2.5}$ data and select forecast parameters.

Initially, a user submits a forecast request via the web application, specifying a target date and time and the most recent 5 hours of $PM_{2.5}$ readings. These inputs are validated client-side to ensure they fall within Nairobi's geographical bounds and acceptable value ranges. The application then packages this data with metadata (hour, week, year) into a structured JSON payload.

This payload is processed by the Python backend , which first checks for real-time weather data via the OpenWeather API. The backend then combines these weather features (temperature, humidity, wind speed) with the user-provided $PM_{2.5}$ history to create a feature vector. The feature vector is fed into three pre-trained LightGBM models:

1. Main Model: Predicts median $PM_{2.5}$ concentration
2. Upper Quantile Model: Generates the 95th percentile bound
3. Lower Quantile Model: Calculates the 5th percentile bound

These models, trained on historical Nairobi air quality data with quantile regression,

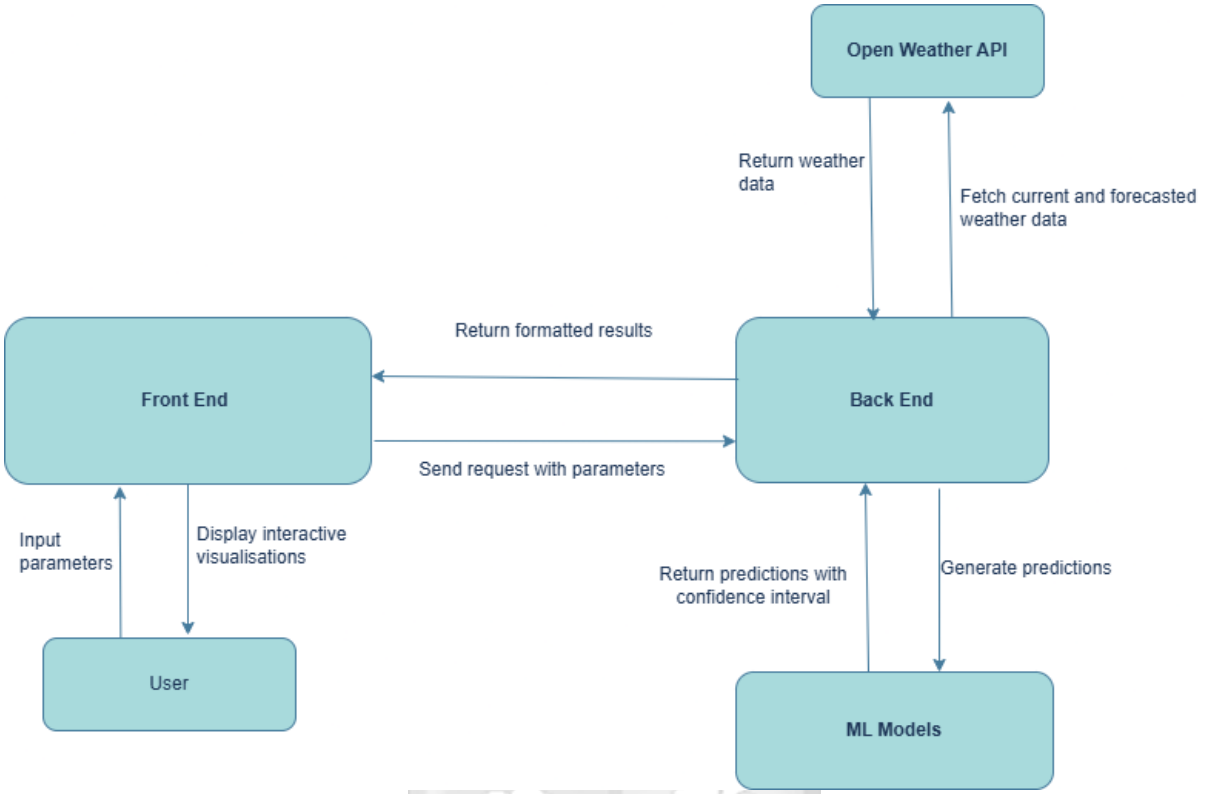


Figure 4.1: Architecture of the $PM_{2.5}$ Prediction System

process the input in parallel. The results are post-processed to assign AQI categories (Good, Moderate, Unhealthy) based on EPA standards, with each prediction interval color-coded for visual interpretation.The backend returns the forecast as a JSON response containing 24 hours of predictions with confidence intervals. The Streamlit frontend then visualizes this data through: interactive plotly charts showing temporal trends; tabular displays with AQI color-coding and downloadable CSV reports. This architecture ensures reliable operation even with intermittent API connectivity while maintaining interpretability for non-technical users.

4.3 Frontend Development

The frontend of the $PM_{2.5}$ Prediction Tool was developed using Streamlit, a Python framework designed for rapid deployment of data-centric web applications. This section details the interface design, user interaction workflows, and robustness mechanisms implemented to ensure usability and reliability.

4.3.1 User Interface Design

The interface adopts a basic, task-focused design, prioritizing clarity and accessibility. Navigation is facilitated through a sidebar menu, implemented using streamlit_option_menu, allowing users to seamlessly switch between the Home, Methodology, and Prediction modules as shown in Figure 4.2.

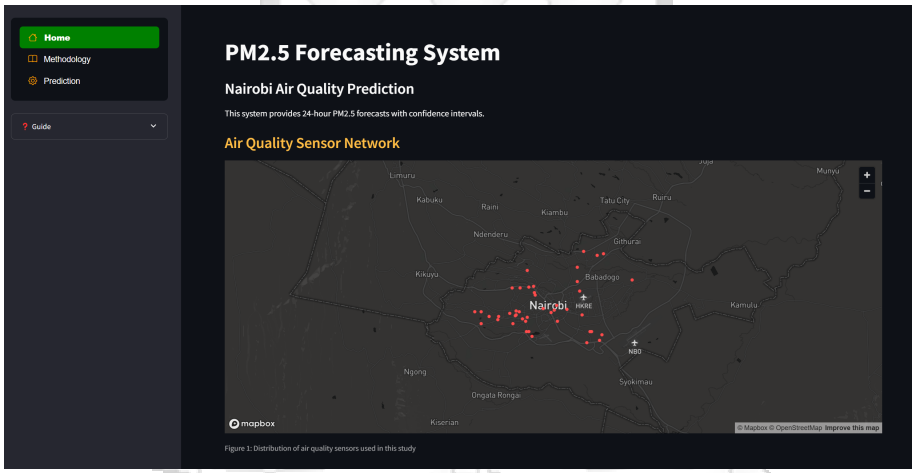


Figure 4.2: Web Application Navigation Bar

Input controls ensure data integrity and user-friendliness. The date and time selection employs restrictive widgets, such as date_input with a five-day limit, reflecting the OpenWeatherMap API's forecast constraints. Additionally, $PM_{2.5}$ history data is entered through a structured five-column grid of numeric inputs, enforcing data completeness as illustrated in Figure 4.3. Advanced options, like weather data sources, are kept hidden in expandable sections to keep the interface clean and uncluttered. Users can view them only when needed.

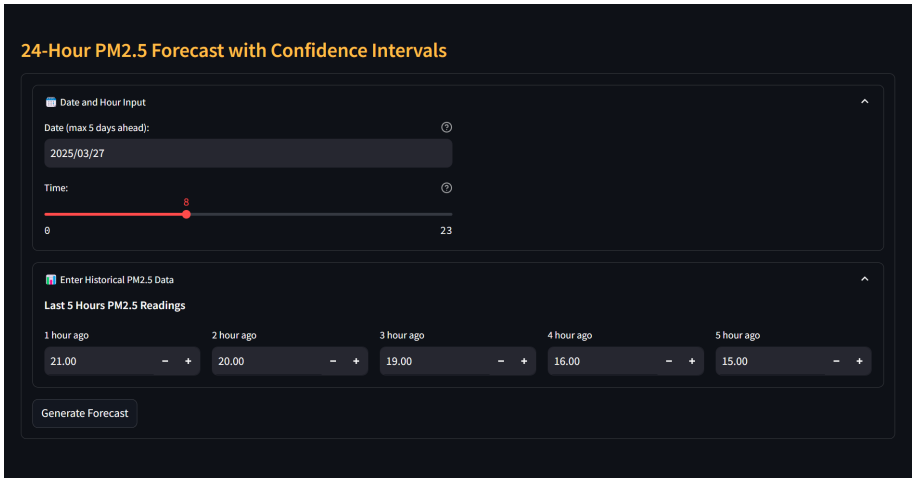


Figure 4.3: Frontend Interface

4.3.2 Error Handling and User Feedback

To enhance user experience and minimize frustration, the system incorporated mechanisms to guide corrective actions and ensure data integrity. Date validation was implemented ensuring users can only select dates within the valid forecast period—from the current day up to thirt days in the past. To prevent errors, dates outside this range are automatically made inactive, preventing users from selecting unavailable options.

To manage API failures effectively, the system provides real-time alerts when weather data defaults to alternative APIs. If the API is unavailable, users are informed that the displayed weather data is using an alternative weather APIs. A warning message is displayed in such cases (Figure 4.4).

```
except Exception as api_error:
    error_msg = str(api_error)
    if hasattr(api_error, 'response'):
        error_msg += f" (Status: {api_error.response.status_code})"

    st.warning(f"⚠ Using alternative API(meteostat)")
```

Figure 4.4: API Error Handling

The system also enhances usability by incorporating dynamic processing states. A progress bar (st.progress) visually tracks forecast generation, keeping users informed about system activity. Additionally, toast notifications (st.toast) confirm successful API calls, providing instant feedback and ensuring a seamless interaction with the platform(Figure 4.5). Through these features, the system proactively prevents errors, guides

users toward corrective actions, and maintains transparency in data processing.

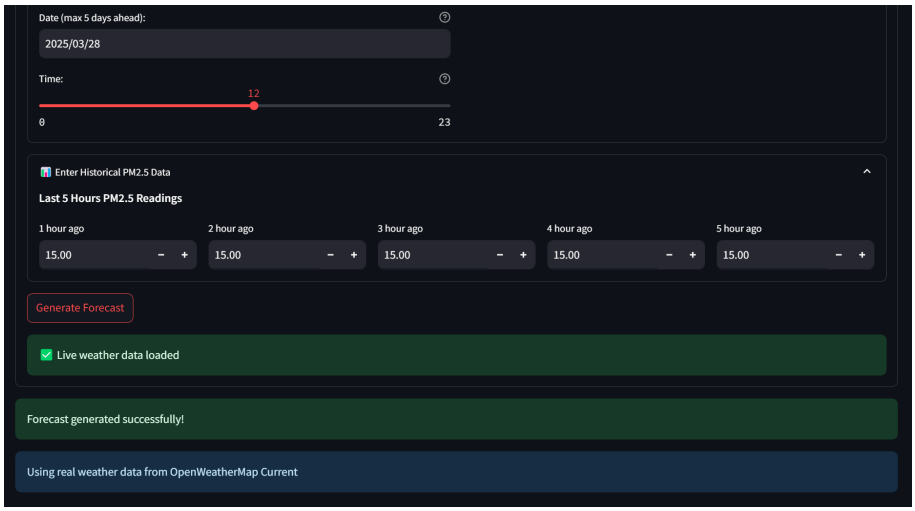


Figure 4.5: Confirmation of successful API calls

4.4 Backend Development

The $PM_{2.5}$ Prediction Tool is built around a structured set of core components that facilitate efficient model serving, data processing, and API interaction.

4.4.1 Model Layer

This study utilized the LightGBM ensemble, consisting of three quantile regression models: a main model, an upper 95% confidence bound model, and a lower 5% confidence bound model. These models work together to provide more reliable and uncertainty-aware predictions illustrated in Figure 4.6. To enhance efficiency, caching mechanisms are implemented using the @st.cache_resource decorator. This optimization ensures that models are loaded only once per session, reducing unnecessary computations and improving response times

4.4.2 Data Processing pipeline

All the data preprocessing steps were defined in the data_preprocessing script. The system employed feature engineering techniques, such as creating lag-based features to capture historical $PM_{2.5}$ trends. Additionally, temporal features such as hour, week, and year are extracted from timestamps to incorporate seasonal and time-dependent patterns into the predictions.

```python
@st.cache_resource
def load_models():
    """Load all LightGBM models (main + quantile)"""
    try:
        models = load('lightgbm_models.joblib')
        return {
            'main': models['main_model'],
            'upper': models['upper_model'],
            'lower': models['lower_model']
        }
    except Exception as e:
        st.error(f"Model loading failed: {str(e)}")
        return None
```

Figure 4.6: Loading and caching LightGBM Model

4.4.3 API Integration

To account for meteorological influences on air quality, the system integrates real-time data from the OpenWeatherMap API, fetching both current conditions and past conditions(in the past 30 days) for temperature, humidity, wind speed, wind direction (degrees), pressure, maximum temperature, and dew point. If API connectivity fails, the tool defaults to using alternative APIs i.e Meteostat. This dual-source approach ensures reliability, with cached API responses (15-minute intervals) minimizing latency while fallback simulations trigger user alerts.

Chapter 5: System Implementation and Testing

5.1 Introduction

This chapter documents the practical implementation of the $PM_{2.5}$ forecasting system, translating the theoretical design from Chapter 4 into a functional tool for environmental analysis. It includes interface screenshots that demonstrate the user workflow, data validation processes to ensure accuracy, and testing outcomes that verify system reliability. Additionally, result visualizations are presented to showcase prediction outputs. The chapter emphasizes how the system meets the needs of environmental scientists and policymakers through an intuitive design and robust technical implementation.

5.2 User Interface Design

The Streamlit-based interface, shown in Figure 5.1, is specifically designed for environmental analysts, providing an intuitive and efficient workflow for $PM_{2.5}$ forecasting. To use the tool, environmental analysts should navigate to the prediction tab, select a forecast start date and hour for real-time forecasting, and choose a start hour in a 24-hour format. They then input $PM_{2.5}$ measurements from the past five hours (in $\mu g/m^3$) before clicking "Generate Forecast" to obtain predictive results.

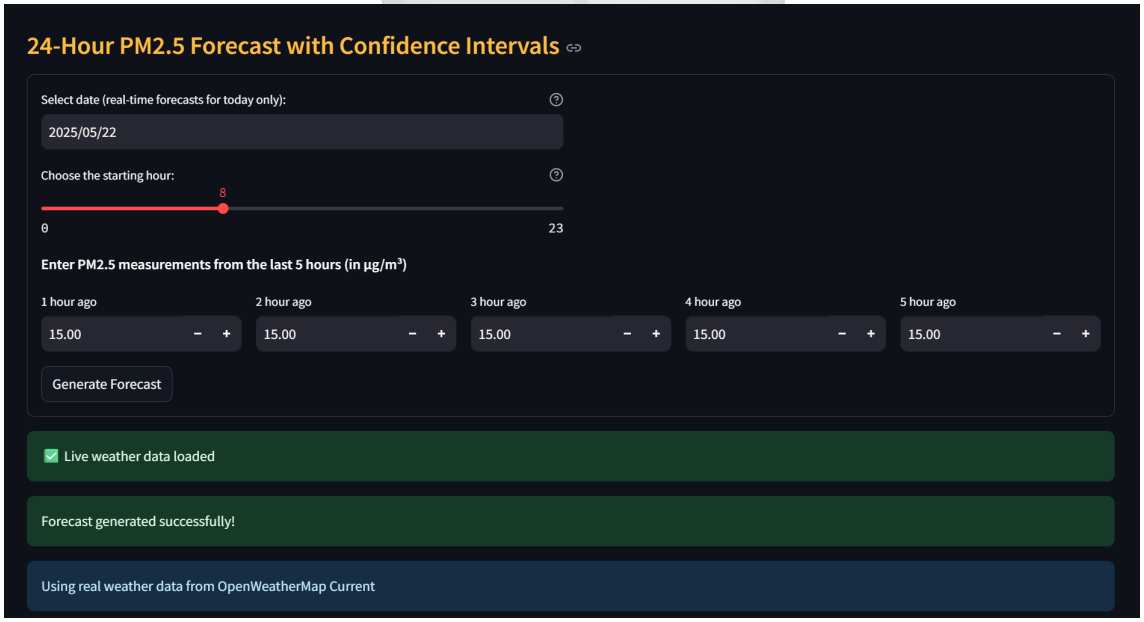


Figure 5.1: User input

Finally, in the output phase, the forecast results are displayed in an interactive chart,

allowing users to analyze trends visually as shown in Figure 5.2 . Additionally, the system provides an option to download the forecast data in a CSV format for further analysis or reporting as shown in Figure 5.3.

Figure 5.2: Interactive chart output

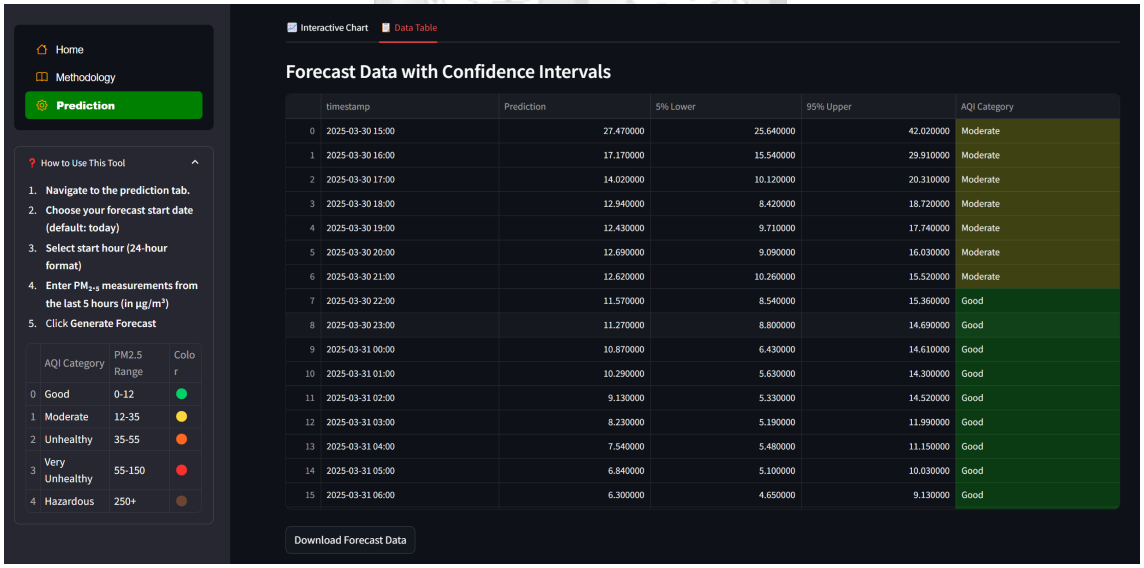


Figure 5.3: Forecasted Data Table output

5.3 Input Validation

The system enforces validation checks to ensure data integrity and model reliability. To maintain forecasting accuracy, the system enforces strict date and time rules. Users cannot select a forecast date in the future,since the max_date is set to the current date (Figure 5.4). The system is designed to generate predictions exclusively for the current day, as real-time weather data integration ensures maximum accuracy for immediate forecasting needs. Users attempting to select future dates receive an automated prompt: 'Forecasting is limited to the current day only'.

```
with col1:
    # Enhanced date input with tomorrow as default
    min_past_days=30
    min_date = today + timedelta(days=min_past_days)
    max_date = today
    default_date = today    # Default to today

    forecast_date = st.date_input(
        "Choose your forecast start date (any date from today through the past:",
        min_value=min_date,
        max_value=max_date,
        value=default_date,
        #help="Weather data available for up to 5 days in advance"
    )

    # Show warning if trying to forecast beyond API limits
    if forecast_date > today + timedelta(days=5):
        st.warning("Note: Forecasting is limited to the current day only")
```

Figure 5.4: Validating date and time inputs

To prevent unrealistic input values, $PM_{2.5}$ measurements are restricted between 0 and 100 µg/m^3. The system enforces a minimum value of 0 µg/m^3 and a maximum of 100 µg/m^3 using built-in input constraints as shown in Figure 5.5. Furthermore, the form requires a complete history of the past five hours $PM_{2.5}$ readings before submission is allowed.

```python
# PM2.5 history inputs (formerly in expander)
st.markdown("**Enter PM2.5 measurements from the last 5 hours (in µg/m³)**")
pm_cols = st.columns(5)
pm_history = []
for i, col in enumerate(pm_cols, 1):
    with col:
        pm_history.append(
            st.number_input(f"{i} hour ago",
                            min_value=0.0,
                            max_value=100.0,
                            value=15.0,
                            key=f"pm_{i}")
        )
```

Figure 5.5: Validating $PM_{2.5}$ Measurements

5.4 Prediction Results

When an environmental analyst submits the forecast request by clicking the "Generate Forecast" button, the system processes the inputs and displays results in a dedicated output section as shown in section 5.3. The results are presented through two integrated components: interactive visualisation and tabular data that can be downloaded. Based on Figure 5.6, the dataset contains several key columns that provide detailed insights into the $PM_{2.5}$ forecasting process:

timestamp	prediction	upper_95	lower_05	aqi_category	data_source
3/30/2025 15:00	27.46915348	42.02367376	25.64490924	Moderate	OpenWeatherMap Forecast
3/30/2025 16:00	17.17132941	29.90516869	15.53840828	Moderate	OpenWeatherMap Forecast
3/30/2025 17:00	14.02075864	20.31032308	10.11605498	Moderate	OpenWeatherMap Forecast
3/30/2025 18:00	12.93850882	18.7150004	8.416351359	Moderate	OpenWeatherMap Forecast
3/30/2025 19:00	12.43325813	17.7360915	9.71172378	Moderate	OpenWeatherMap Forecast

Figure 5.6: Sample of the forecasted data

This columns include:

1. **timestamp**: The date and time of the prediction.
2. **prediction**: The predicted median $PM_{2.5}$ concentration in micrograms per cubic meter ($\mu g/m^3$).
3. **upper_95**: The upper bound of the 95% confidence interval for the prediction.
4. **lower_05**: The lower bound of the 5% confidence interval for the prediction.
5. **aqi_category**: The air quality index (AQI) category, which classifies the air quality based on EPA standards (e.g., Good, Moderate, Unhealthy).

6. **data_source**: The origin of the weather data, which could be from an API or simulated data.

Interpretation of one data instance

For the prediction made on 30-Mar-2025 at 1500hrs, the model forecasts a moderate air quality level, with a $PM_{2.5}$ concentration of 27.47 µg/m^3. The confidence range for this prediction is relatively narrow, between 25.64 and 42.02 µg/m^3, with a 90% confidence level. The AQI category assigned to this prediction is "Moderate," indicating that the air quality is acceptable, but there may be some health concerns for sensitive individuals.The narrow confidence interval, with a range of ±8 µg/m^3, suggests a high level of certainty in the model's prediction for this specific hour. This indicates that the forecast is reliable, and the variability in $PM_{2.5}$ concentrations is relatively low for this time frame.

While initial designs considered simulated meteorological data as a fallback for unavailable forecasted $PM_{2.5}$ values, this approach was ultimately discarded due to the compounding uncertainties inherent in modeling dynamic weather variables (e.g., wind, precipitation). Simulated atmospheric conditions could introduce significant error propagation into $PM_{2.5}$ predictions, undermining reliability. Instead, the system relies exclusively on empirical data from OpenWeatherMap API or Meteostat to maintain traceability and accuracy. This shift prioritizes direct observational inputs over theoretical approximations.

5.5 Prediction Error Over time

The Root Mean Squared Error (RMSE), a measure of the model's prediction performance, typically decreases as the time horizon increases as shown in Figure 5.7.

Forecast Decay Evaluation:

Output is truncated. View as a scrollable element

	Horizon (hours)	RMSE
0	1	3.212207
1	2	13.117849
2	3	11.394784
3	4	11.623428
4	5	10.677853
5	6	11.154854
6	7	11.159481
7	8	11.135391
8	9	11.113731
9	10	11.123239
10	11	11.253830
11	12	11.301409

Figure 5.7: Prediction Decay Evaluation

For a 1-hour forecast, the RMSE was 3.21 µg/m^3, indicating high accuracy suitable for short-term predictions. As the forecast horizon extended to 2 hours, the RMSE increased to 13.12 µg/m^3, which is expected due to the growing uncertainty associated with weather conditions. Despite the increase in error over time, the error stabilized at around 11.4 µg/m^3 for horizons between 3 to 12 hours indicating that the model can provide consistent operational forecasts within a reasonable margin of error.

Chapter 6: Discussion of Results

6.1 Introduction

The objective of this research was to develop a machine learning approach for particulate matter prediction in Nairobi County. This section provides a detailed discussion of the results obtained from the study which includes data preprocessing, analysis of the data, model building and optimization.

6.2 Data Understanding

The particulate matter data covered the period of 1 January 2018 to 19 March 2024. Sensor ID, Sensor-type, Lon, Lat, date, value-type, and value are the only seven features present in the original data.The value column contains the corresponding value for the value-type column. The data set was pivoted using the python pivot_table() function to access the indicators in the value-type column. The resulting data is illustrated in Table 6.1:

Table 6.1: Dataset Attributes

#	Attribute	Description
1	Sensor-ID	Unique identifier of the sensor
2	Sensor-type	Type of sensor used
3	Lon	Longitude of the sensor location
4	Lat	Latitude of the sensor location
5	Timestamp	Time when data was collected
6	Location	Numeric code indicating the sensor's place
7	$PM_{2.5}$	$PM_{2.5}$ concentration recorded
8	PM_{10}	PM_{10} concentration recorded
9	Humidity	Recorded humidity value
10	Temperature	Recorded temperature value

The data set contained approximately 27 million records, aggregated at the second level. Data collection intervals varied significantly, with sensors recording measurements at an average interval of 12.45 seconds, ranging from 0 seconds to a maximum of 2,678,435.92

```
Mean interval: 12.449209576999872 seconds
Minimum interval: 0.0 seconds
Maximum interval: 2678435.92008 seconds
```

Figure 6.1: Data Collection Time Interval Statistics

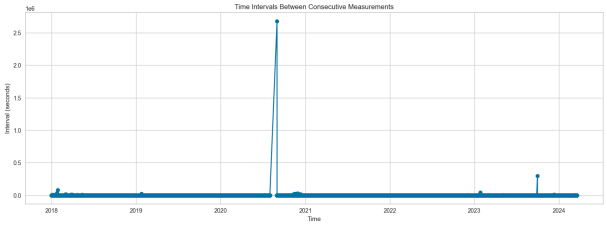


Figure 6.2: Data Collection Time Interval Graph

seconds. This inconsistency suggested the presence of missing data. The results are illustrated in Figure 6.1 and Figure 6.2.

6.3 Data preprocessing

6.3.1 Missing Value Treatment

As illustrated in Figure 6.3, humidity and temperature contained more than 50% missing values and $PM_{2.5}$ contained 43.19% missing values.

```
pm2.5          43.196500
humidity       56.832979
temperature    56.832975
dtype: float64
```

Figure 6.3: Missing values in the entire dataset

Using the missing values assumptions, this study investigated the presence of missing values in the data set.

Missing Not at Random

Using the "groupby" function in python the data was grouped based on the sensor types and the following pattern on missing values was identified. The results are shown in Figure 6.4.

sensor_type	pm2.5	temperature	humidity
DHT22	99.999891	0.001353	0.001361
SDS011	1.583034	98.437073	98.437073
SPH0645_MIC	99.443672	99.582754	99.582754
UltimateGPS	100.000000	100.000000	100.000000
pms5003	0.003223	99.999870	99.999870

Figure 6.4: Missing values by sensor type in (%)

The presence of missing values in specific columns was found to be attributed to the functionality of different sensors used in data collection. Since each sensor is designed to measure specific parameters, it does not provide data for unrelated attributes, leading to missing values that are Missing Not At Random (MNAR). The following observations explain this phenomenon:

(a) **DHT22 Sensor:** measures humidity and temperature, which explains the missing values in the $PM_{2.5}$ column.

(b) **SDS011 Sensor:** measures particulate matter (PM) concentration, leading to missing values in the temperature and humidity columns.

(c) **UltimateGPS Sensor:** Collects location attributes but does not measure air quality parameters, hence the missing values in the dataset.

(d) **SPH0645_MIC Sensor:** A digital MEMS microphone used for audio applications, unrelated to air quality measurements, explaining the missing values in relevant columns.

(e) **PMS5003 Sensor:** Measures particulate matter (PM) concentration, resulting in missing values in the temperature and humidity columns.

To address missing values in the dataset, a targeted approach was applied based on the functionality of each sensor. Since the DHT22 sensor does not measure $PM_{2.5}$, all records where $PM_{2.5}$ was either missing or recorded as a value were removed. Similarly, for the SDS011 and PMS5003 sensors, which measure particulate matter concentration, all records containing values or missing data in the humidity and temperature columns were dropped. Additionally, data from the SPH0645_MIC sensor, which is unrelated to air

quality measurements, and the UltimateGPS sensor, which provides location attributes (but was dropped as latitude and longitude columns were already available), were entirely removed from the dataset. This approach ensured that only relevant and reliable measurements were retained for analysis.

Missing Completely at Random

The remaining missing values shown in Figure 6.5 as a result of the correct sensors not collecting the required data. i.e SDSO11 not collecting data on $PM_{2.5}$. This could have resulted due to faulty sensors.

sensor_type	sensor_type	pm2.5	temperature	humidity
DHT22	0.0	100.0	49.999998	50.000002
SDS011	0.0	0.0	100.000000	100.000000
pms5003	0.0	0.0	100.000000	100.000000

Figure 6.5: Remaining Missing values by sensor type in (%)

Due to the previous pre-processing, all $PM_{2.5}$ measurements were missing (100%) for DHT22 sensors. More notably, approximately 50% of temperature and humidity readings were missing from DHT22 records, suggesting either intermittent sensor failures. Conversely, SDS011 and PMS5003 sensors showed complete data coverage for their designed purpose (0% missing $PM_{2.5}$ values), but as expected, contained no temperature or humidity measurements (100% missing) since these parameters fall outside their measurement capabilities. Based on these findings, we implemented two key data cleaning actions:

(a) Complete removal of all DHT22 sensor records due to their high missingness rate in all the columns

(b) Elimination of temperature and humidity columns from SDS011/PMS5003 datasets since these sensors cannot provide these measurements.

The final curated data set exclusively comprised particulate matter measurements ($PM_{2.5}$) from SDS011 and PMS5003 sensors, ensuring all retained data aligns with the sensors' designed capabilities and the study's air quality focus.

6.3.2 Outlier Treatment

Outliers are data points that are past the 3-standard deviation point. A boxplot was used to detect the outliers in the particulate matter column as illustrated in Figure 6.6 and Figure 6.7

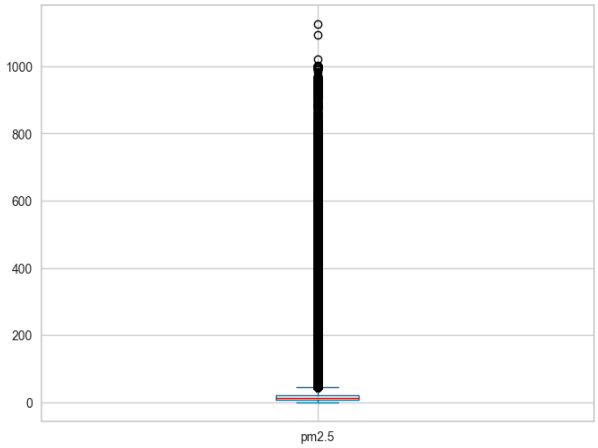


Figure 6.6: Boxplot with outliers

	latitude	longitude	pm2.5
count	1.574861e+07	1.574861e+07	1.574861e+07
mean	-1.299111e+00	3.681585e+01	1.831565e+01
std	4.099719e-02	5.979603e-02	2.939010e+01
min	-1.446000e+00	3.665800e+01	-1.000000e+00
25%	-1.298000e+00	3.679100e+01	7.000000e+00
50%	-1.298000e+00	3.679100e+01	1.375000e+01
75%	-1.289000e+00	3.682500e+01	2.200000e+01
max	-1.192000e+00	3.700100e+01	1.124400e+03

Figure 6.7: Descriptive Statistics

A number of anomalous $PM_{2.5}$ measurements were found in the dataset, including extreme positive outliers (maximum: 1124.400 µg/m^3), possibly from instrument malfunctions) and negative values (minimum:(-1.000 µg/m^3), suggesting sensor errors). These outliers disproportionately affected central tendency measures as evidenced by the right-skewed distribution (mean: (18.32 µg/m^3), which exceeds the median (13.75µg/m^3)

To ensure data validity and robustness, some preprocessing steps were applied to the $PM_{2.5}$ values. Negative values, which are physically implausible, were replaced with the median value (13.750 µg/m^3) to maintain data integrity without introducing artificial distortions. Extreme values were capped using the interquartile range (IQR) method, setting the maximum $PM_{2.5}$ value to 44.500 µg/m^3. The overall data distribution improved, as seen in Figure 6.8.

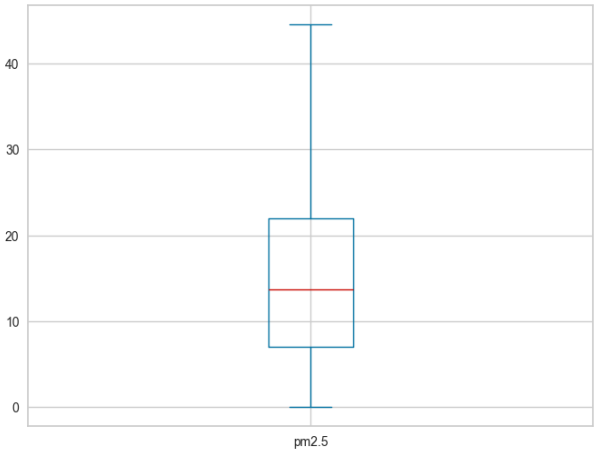


Figure 6.8: Boxplot without outliers

6.3.3 August 2020 Imputation

In the Sensors Africa repository, there was no available data for August 2020, as illustrated in the 2020 time series plot (Figure 6.9).

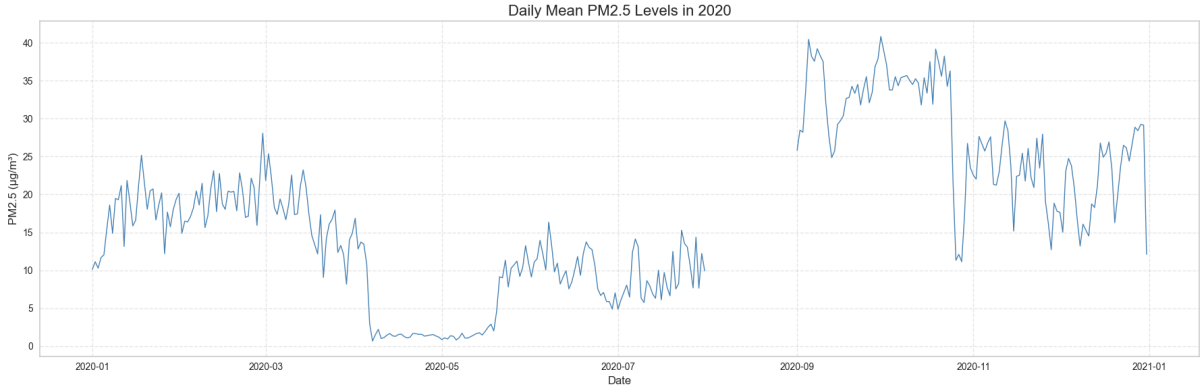


Figure 6.9: Daily Mean $PM_{2.5}$ levels in 2020

Given that 2020 was marked by the unique occurrence of COVID-19, it was more appropriate to use the patterns observed in the same year to impute the missing data for

August. To better understand the structure of the dataset, time series decomposition was performed to identify the trend and seasonality for the month of August, which aided in determining a suitable imputation strategy.

Before proceeding, however, it was necessary to determine whether an additive or multiplicative decomposition was more suitable. This was done by visually inspecting the reconstructed data from both models to assess which best captured the underlying patterns. The results of this comparison are illustrated in Figure 6.10 and Figure 6.11

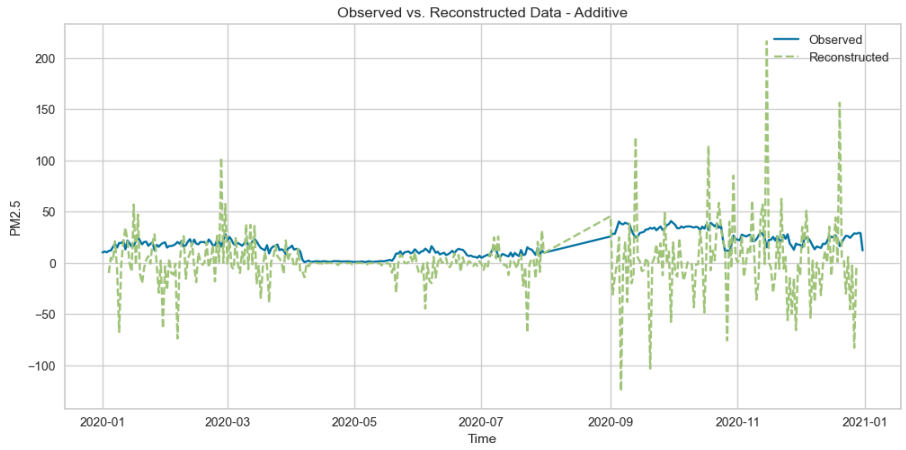


Figure 6.10: Additive Reconstructed Data

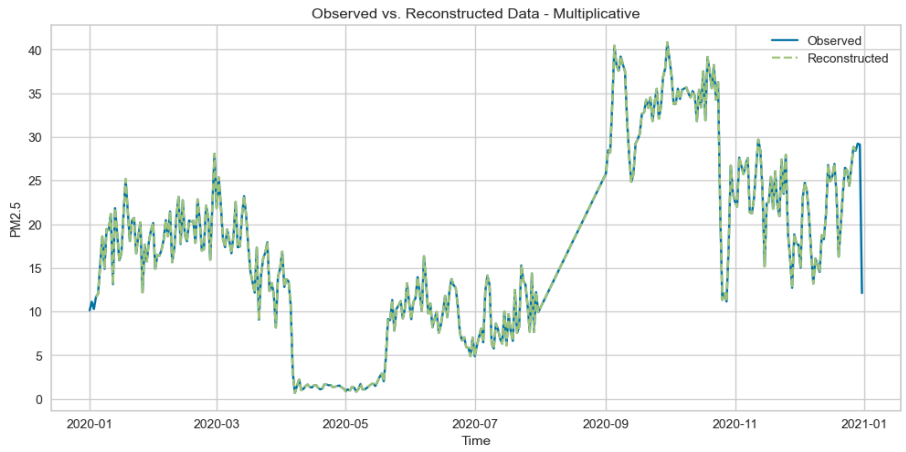


Figure 6.11: Multiplicative Reconstructed Data

From Figure 6.11, the reconstructed data from the multiplicative model closely matched the observed data. As a result, the trend and seasonal components of the multiplicative

model was used to impute August 2020 values. The results are illustrated in Figure 6.12 and Figure 6.13

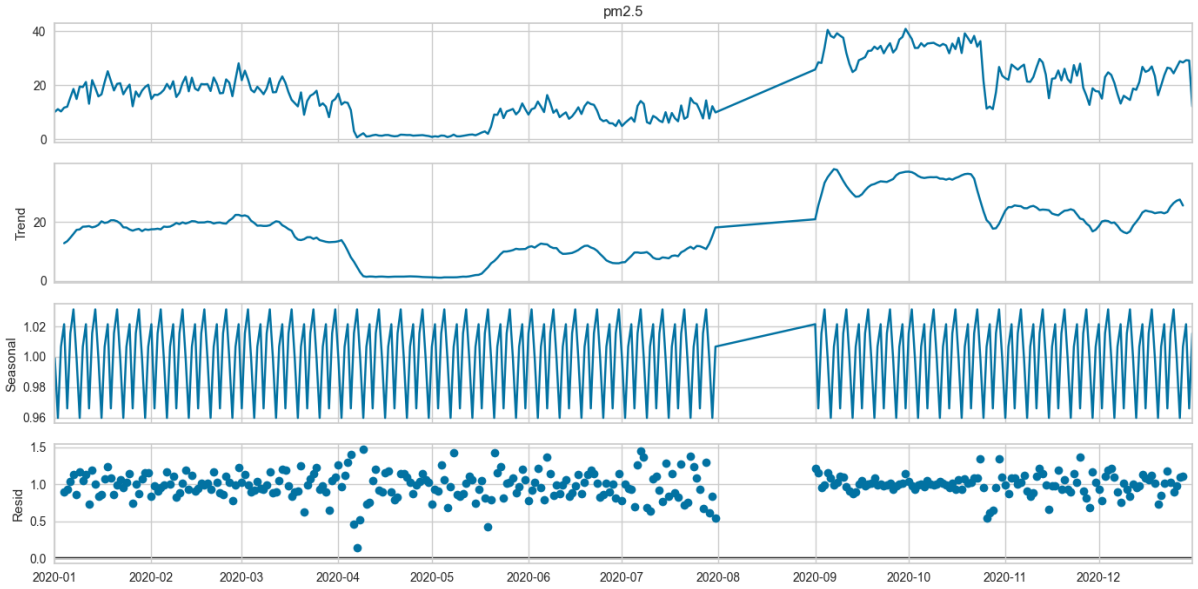


Figure 6.12: Before Reconstruction

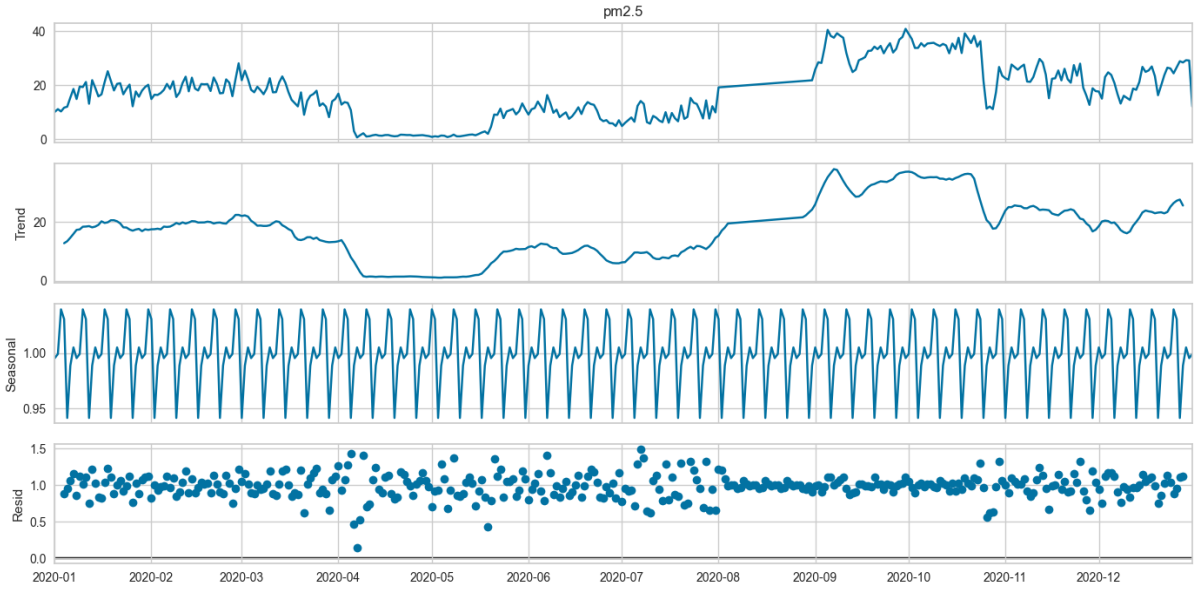


Figure 6.13: After Reconstruction

The imputed values, now align with the overall trend and seasonality of the data.

6.3.4 Resampling

A dataset qualifies as a time series only when observations are collected at regular temporal intervals. Following the imputation of August 2020 values, we re-examined the collection frequency. The temporal distribution shown in Figure 6.14 and Figure 6.15 revealed irregular sampling intervals, with some observations recorded beyond the 12 mark.

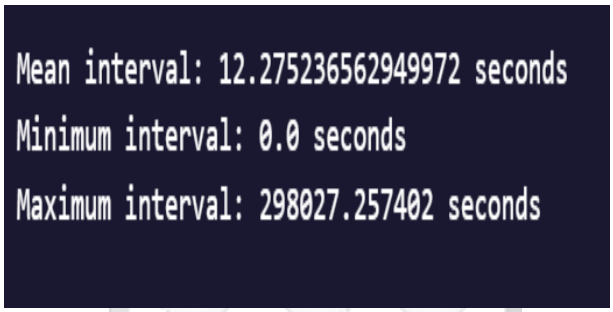


Figure 6.14: Time interval statistics

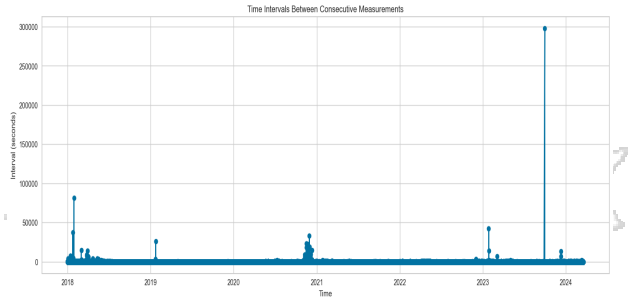


Figure 6.15: Time Interval Graph

To establish proper time series characteristics, the data was resampled to 1-hour intervals. The results after resampling are illustrated in Figure 6.16 and Figure 6.17

```
Mean interval: 3600.0 seconds
Minimum interval: 3600.0 seconds
Maximum interval: 3600.0 seconds
```

Figure 6.16: Time interval statistics

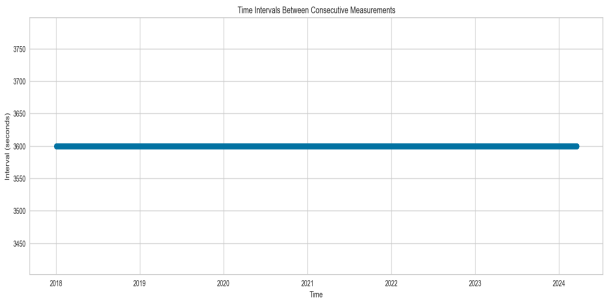


Figure 6.17: Time Interval Graph

6.3.5 Data Enrichment

With the data aggregated at the hourly level, it was merged with historical weather data obtained from OpenWeather, which was also aggregated at an hourly frequency. This integration ensured consistency between the datasets, allowing for more comprehensive analysis. The merged dataset is illustrated in Figure 6.18

Merged Data (using concat):

	pm2.5	temp	visibility	dew_point	feels_like	temp_min	temp_max	pressure	humidity	wind_speed	wind_deg	rain_1h	clouds_cover	weather_main
2018-01-01 00:00:00+00:00	16.556757	18.16	9999.0	13.04	17.91	16.71	18.16	1014.0	72.0	2.60	20.0	NaN	40.0	Clouds
2018-01-01 01:00:00+00:00	17.695122	17.16	9999.0	12.08	16.81	16.71	17.16	1014.0	72.0	2.06	10.0	NaN	40.0	Clouds
2018-01-01 02:00:00+00:00	25.462857	16.16	9999.0	12.13	15.84	15.71	16.16	1014.0	77.0	2.60	360.0	NaN	40.0	Clouds

Figure 6.18: Merged Data

6.4 Exploratory Data Analysis

This section contains results from the visual exploration of the data. Several studies relate the most significant indicators of particulate matter concentrations in the air to their typical concentrations hourly, daily, monthly, and even annually in order to gain a better understanding of these indicators.

6.4.1 Particulate Matter Data Analysis

According to Figure 6.19, $PM_{2.5}$ levels remained relatively high and consistent from Monday to Saturday, with the highest concentrations observed on Friday. In contrast, Sunday recorded the lowest $PM_{2.5}$ levels, suggesting a possible reduction in pollution on that day. This pattern may indicate changes in human activity or traffic emissions over the course of the week in Nairobi.

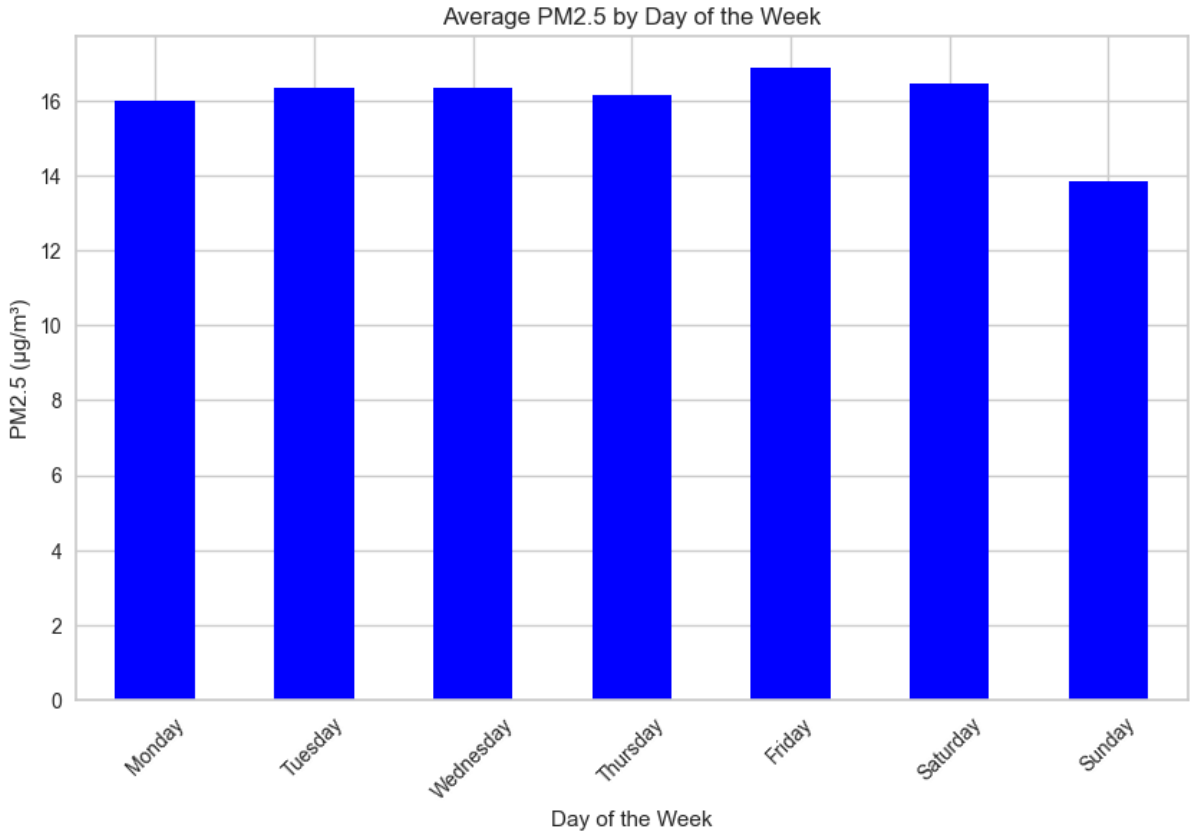


Figure 6.19: Average $PM_{2.5}$ by Day of Week

According to Figure 6.20, $PM_{2.5}$ levels exhibit distinct diurnal and seasonal patterns. Diurnally, the concentrations are lowest around midnight (00:00) and midday (1200hrs

–1300hrs). However, two notable peaks are observed: one at 0400hrs, where $PM_{2.5}$ levels rise sharply, and a second, higher peak at 1700hrs, reaching 19.58 µg/m^3. After 1700hrs, the levels gradually decline into the night. This pattern suggests that early morning and late afternoon $PM_{2.5}$ levels are influenced by factors such as traffic, industrial activity, and meteorological conditions, which affect pollutant dispersion.

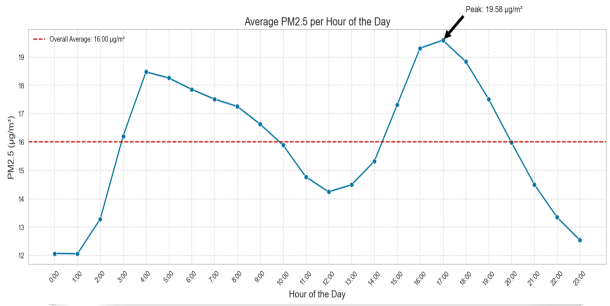


Figure 6.20: Average $PM_{2.5}$ by Hour of the day

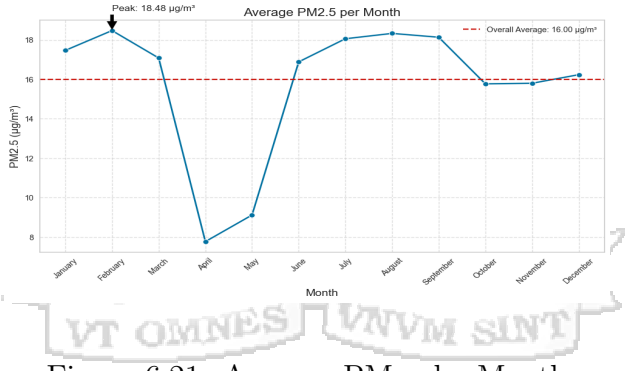


Figure 6.21: Average $PM_{2.5}$ by Month

As illustrated in Figure 6.21, February, part of Nairobi's hot and dry season, experiences the highest $PM_{2.5}$ concentrations, likely due to low rainfall and reduced atmospheric dispersion, leading to the accumulation of dust, vehicle emissions, and industrial pollution. In contrast, $PM_{2.5}$ levels experience a sharp drop in April, the wettest month in Nairobi, due to the long rains. The increased rainfall helps clear pollutants from the air, significantly reducing $PM_{2.5}$ concentrations. After the rainy season, pollution levels begin to rise again as the ground dries up. The cool dry season from June to September is marked by higher suspended pollutants in the air, which can contribute to increasing $PM_{2.5}$ levels. This seasonal variation highlights the impact of weather patterns on air quality in

Nairobi, with rainfall acting as a cleansing mechanism, and dry seasons fostering pollutant accumulation.

6.4.2 Spatial Correlation and Hotspot Analysis

The spatial relationship between $PM_{2.5}$ and meteorological factors, as represented in the Figure 6.22, indicated weak correlations across various parameters. $PM_{2.5}$ showed a slight negative correlation with temperature ($r = -0.07$), wind speed ($r = -0.10$), and dew point ($r = -0.05$), suggesting that higher temperatures and stronger winds may help disperse pollutants, reducing $PM_{2.5}$ concentrations. Conversely, there was a minor positive correlation with pressure ($r = 0.03$) and humidity ($r = 0.01$), implying that stable atmospheric conditions and higher humidity might slightly contribute to increased $PM_{2.5}$ levels. Other factors, such as visibility ($r = -0.05$), rainfall ($r = -0.04$), and wind direction ($r = -0.08$), exhibit negligible correlations, indicating their limited influence.

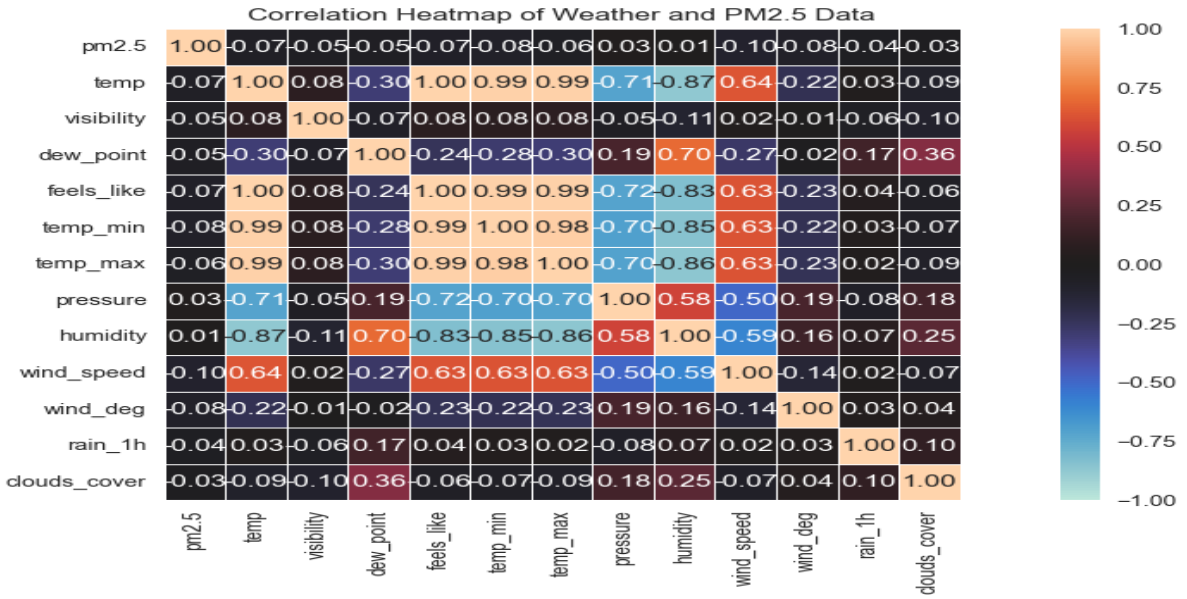


Figure 6.22: Correlation Analysis

To further investigate spatial patterns, hotspot analysis was conducted using a Getis-Ord Gi statistic*, revealing statistically significant clustering of $PM_{2.5}$ as shown in Figure 6.23:

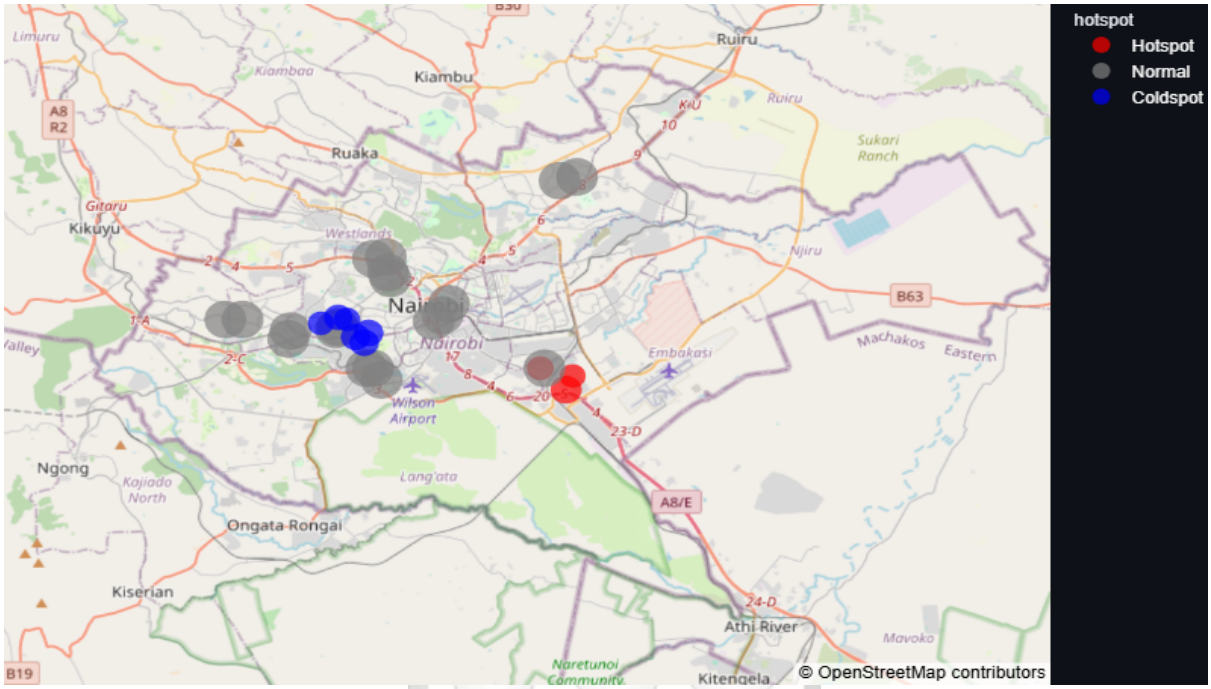


Figure 6.23: Significant Hotspot Analysis

1. **Hotspots**: (z-score $> +1.96$, $p < 0.05$): Areas with significantly elevated $PM_{2.5}$ levels, indicating pollution clustering. The primary hotspot ($z = 2.80$, $p = 0.016$) was located along the Nairobi Expressway/Southern Bypass (latitude: -1.327, longitude: 36.882,elevation: 1635m), with an average $PM_{2.5}$ concentration of 26.01 $\mu g/m^3$. Two additional hotspots were identified near this region, reinforcing the impact of high-traffic corridors on air quality.

2. **Coldspots** (z-score < -1.96, $p < 0.05$): Areas with significantly lower $PM_{2.5}$ levels, suggesting cleaner air. The most pronounced coldspot ($z = -2.61$, $p = 0.001$) was observed in Kilimani (latitude: -1.298, longitude: 36.791,elevation:1757m), with $PM_{2.5}$ averages below 12 $\mu g/m^3$. The higher elevation of these regions may contribute to lower pollutant retention.

These results highlight the spatial heterogeneity of $PM_{2.5}$ distribution across Nairobi's variable topography. Pollution hotspots cluster in low-elevation urban infrastructure (e.g., Southern Bypass at 1635m) where traffic emissions dominate, while coldspots align with higher-elevation areas like Kilimani (1757m) that benefit from enhanced atmospheric dispersion. The analyses demonstrate that meteorological factors alone cannot explain $PM_{2.5}$ patterns; instead, localized anthropogenic sources (e.g.traffic) interacting with to-

pographic features drive the observed spatial clustering. This underscores the need for location-specific air quality management strategies that account for both human activity and geographic context.

6.5 Modeling

This section contains results on the feature engineering, model building and hyperparameter tuning.

6.5.1 Feature Engineering

To fit a time series problem using machine learning algorithms, it is crucial to create lag features. In this study, five lag features were generated to help the model predict future $PM_{2.5}$ concentrations based on past values. These lag features are essential for time-series forecasting as they allow the model to capture temporal dependencies in the data.

In addition to the lag features, time-based features were created to capture seasonal and temporal patterns in $PM_{2.5}$ concentrations. These included features such as hour, day, and month. The hour feature accounts for daily variations in $PM_{2.5}$ levels, which can be influenced by factors like traffic, industrial activity, and weather conditions. The day feature helps capture weekly patterns, such as reduced pollution on weekends when industrial activities are lower. Lastly, the month feature captures seasonal trends, such as higher pollution during winter due to temperature inversions or increased heating activities. These time-based features, combined with lag features, provide a comprehensive understanding of the patterns in $PM_{2.5}$ concentrations, improving the forecasting model.

6.5.2 Feature Selection

Feature selection was done using a random forest model. The results are depicted in Figure 6.24. The RandomForestRegressor feature importance analysis reveals that temporal patterns dominate $PM_{2.5}$ prediction, with autoregressive lags (lag_1 to lag_5) collectively contributing 60% of predictive power. The hour of day emerges as the second most influential feature, highlighting strong diurnal pollution patterns. Among meteorological factors, dew point and temperature show greater importance than humidity, while wind speed and direction demonstrate moderate influence. Notably, extreme temperature measures (temp_max) show minimal predictive value compared to current conditions.

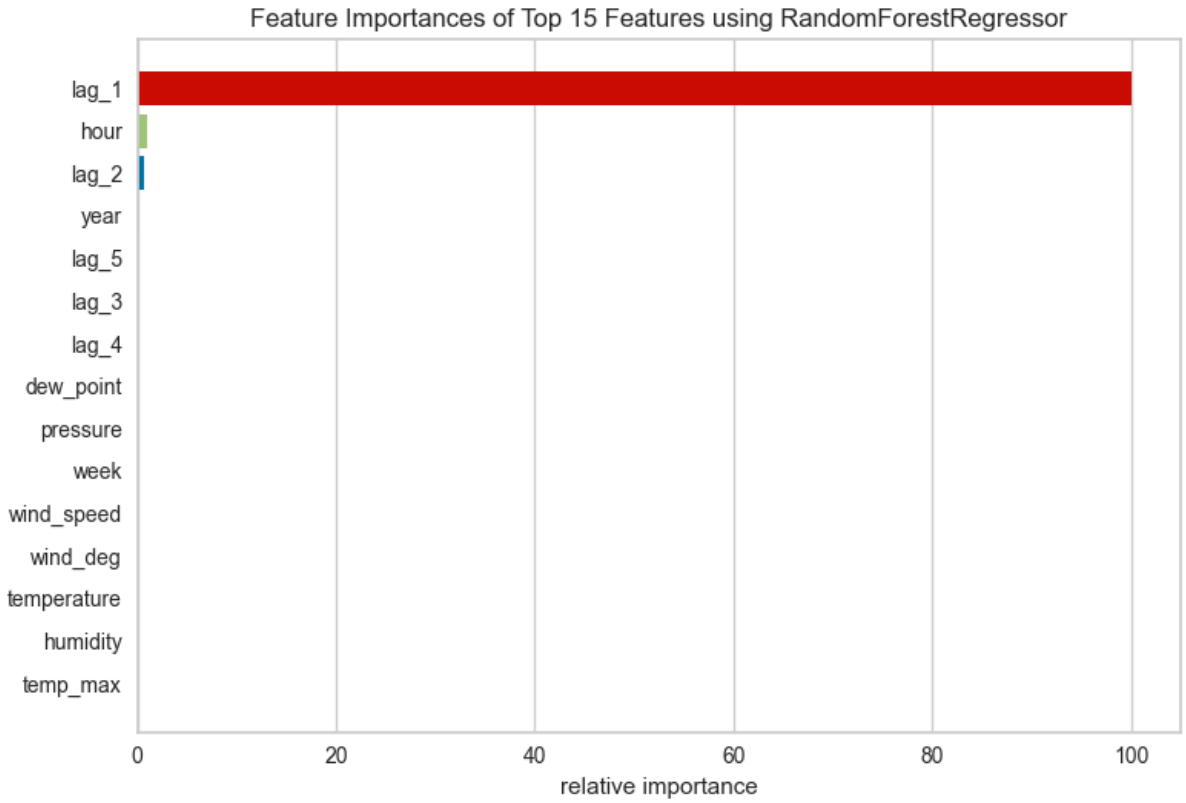


Figure 6.24: Feature Selection

6.5.3 Model Selection and Optimization

This study employed ensemble models, including Random Forest, XGBoost, LightGBM, Gradient Boosting, and a Stacking Ensemble technique. Initial training used default hyperparameters to establish baseline performance (Figure 6.25 & Figure 6.26). The Stacking Ensemble showed the best generalization on test data, motivating further optimization via hyperparameter tuning.

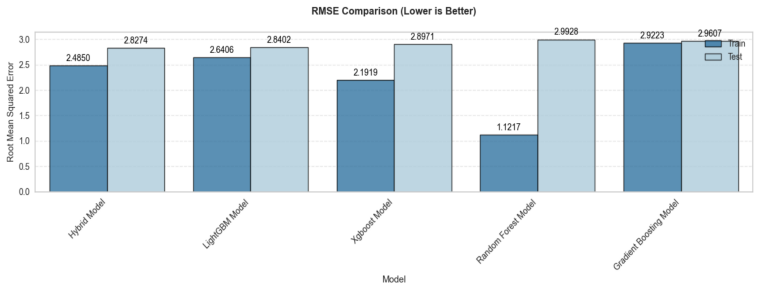


Figure 6.25: RMSE before optimization

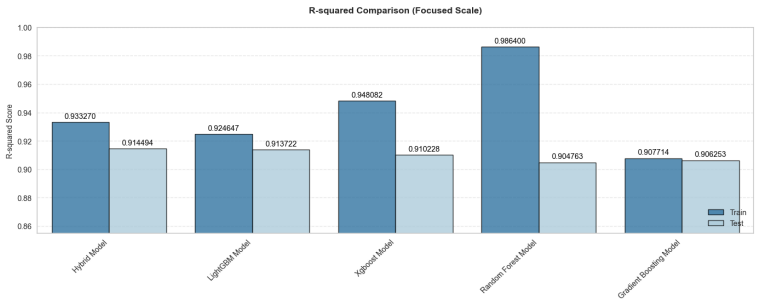


Figure 6.26: R-squared before optimization

Before hyperparameter tuning, the Random Forest model achieved the highest performance on the training data, while the Stacking Ensemble model demonstrated the best generalization performance on the test data.

Ensemble model Optimization

To ensure computational efficiency, RandomizedSearchCV (50 iterations, 5-fold CV, negative MSE scoring) was employed for all models.The search space for each algorithm was designed to cover common practical ranges while prioritizing parameters known to influence bias-variance tradeoffs (e.g., tree depth, regularization, subsampling). A fixed random state of 56 was used to ensure consistent results. Below are the explored ranges, tuning strategies, and final selected values (Table 6.2 & Table 6.3), justified by cross-validation performance:

Table 6.2: Hyperparameter Tuning Table

Model	**Hyperparameters Explored**	**Final Selection**
LightGBM	max_bin:[255] n_estimators: [50] max_depth: [5] min_samples_split: [2] min_data_in_leaf: [1] boosting_type: [gbdt, dart] learning_rate: [0.001, 0.1] reg_alpha: [0,0.1,0.2] reg_lambda: [0,0.1,0.2] subsample: [0.7,0.8,0.9, 1.0] colsample_bytree:[0.7,0.8,0.9,1.0]	max_bin:[255] n_estimators: [200] max_depth: [None] min_samples_split: [10] min_data_in_leaf: [4] boosting_type: [gbdt] learning_rate: [0.05] reg_alpha: [0.1] reg_lambda: [0] subsample: [0.9] colsample_bytree:[1.0]
Random Forest	n_estimators: [50,100,150,200] max_depth: [5,10,15,20, None] min_samples_split: [2,5,10] min_samples_leaf: [1,2,4] max_features: [sqrt, log2, None] bootstrap: [True, False]	n_estimators: [200] max_depth: [20] min_samples_split: [2] min_samples_leaf: [4] max_features: [None] bootstrap: [True]
Gradient Boosting	n_estimators: [50,100,150,200] learning_rate: [0.001,0.01,0.05,0.1] max_depth: [3,5,7,9] min_samples_split: [2,5,10] min_samples_leaf: [1,2,4] subsample: [0.7,0.8,0.9,1.0] max_features: [sqrt, log2, None]	n_estimators: [100] learning_rate: [0.1] max_depth: [7] min_samples_split: [10] min_samples_leaf: [2] subsample: [0.7] max_features: [None]

Table 6.3: Hyperparameter Tuning Table Continuation

Model	**Hyperparameters Explored**	**Final Selection**
XGBoost	n_estimators: [50,100,150,200] learning_rate: [0.001,0.01,0.05,0.1] max_depth: [3,5,7,9] subsample: [0.7,0.8,0.9,1.0] colsample_bytree: [0.7,0.8,0.9,1.0] gamma: [0,0.1,0.2] reg_alpha: [0,0.1,0.5] reg_lambda: [0,0.1,1.0]	n_estimators: [200] learning_rate: [0.1] max_depth: [7] subsample: [1.0] colsample_bytree: [0.8] gamma: [0.1] reg_alpha: [0.1] reg_lambda: [1.0]

Following hyperparameter tuning and model stacking, the Stacked Ensemble model demonstrated superior performance, achieving an MAE of 1.933, RMSE of 2.821, and R^2 of 0.915 as shown in (Figure 6.27 & Figure 6.28). It outperformed all standalone models on test data while maintaining robust training performance, suggesting an optimal bias-variance tradeoff.

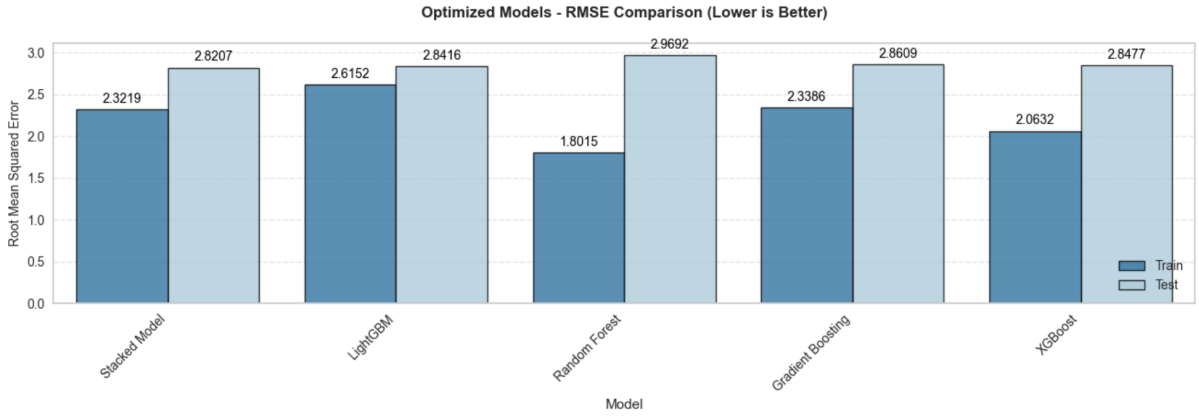


Figure 6.27: RMSE after optimization

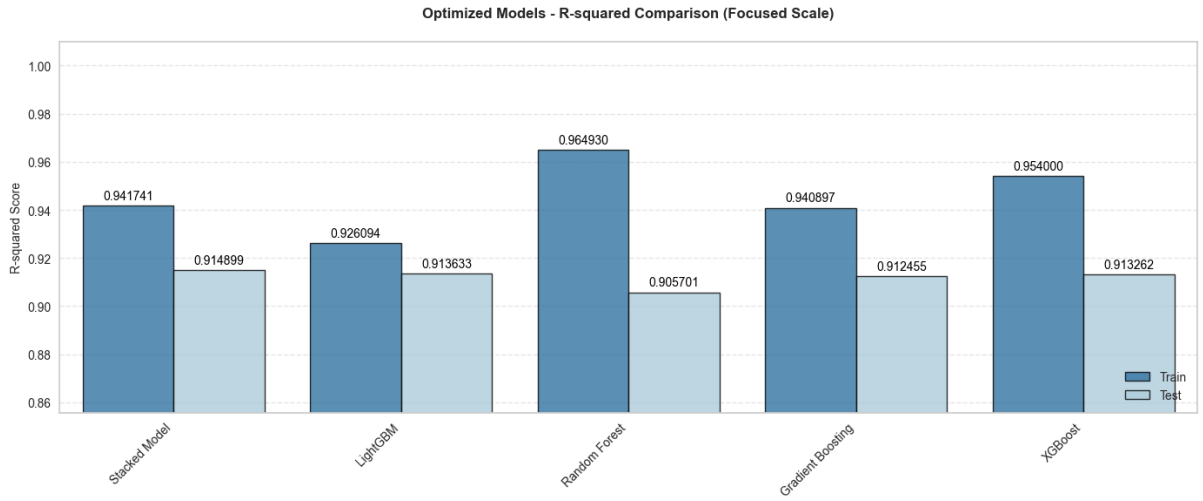


Figure 6.28: R-squared after optimization

The stacking ensemble utilized LightGBM and XGBoost as base learners, combined via a meta-learner (linear regression). The architecture of the fitted stacked model is illustrated in Figure 6.29.

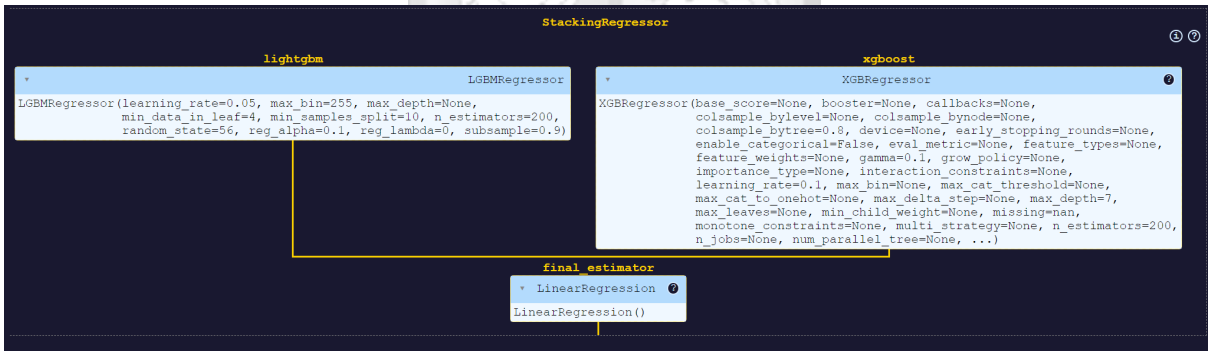


Figure 6.29: Fitted Stacked Regressor

Neural Network Evaluation

This study conducted sequential evaluations of LSTM architectures, beginning with a baseline model featuring a single LSTM layer (32 units) with linear output, trained using the Adam optimizer (learning rate=0.001) with early stopping (patience=8), which achieved a training RMSE of 3.782 (R^2=0.839) and test RMSE of 4.707 (R^2=0.670).To address the noticeable train-test gap, we implemented an enhanced architecture with two LSTM layers (64/32 units) and 20% dropout regularization. While this modification slightly improved test performance (RMSE=4.665, R^2=0.676), the model still exhibited

signs of overfitting as evidenced by as illustrated in Figure 6.30 and Figure 6.31. This suggests that while dropout effectively moderated the overfitting (compared to the baseline's 20% performance drop), the LSTM architecture remained fundamentally prone to overfitting, likely due to the complexity of the network.

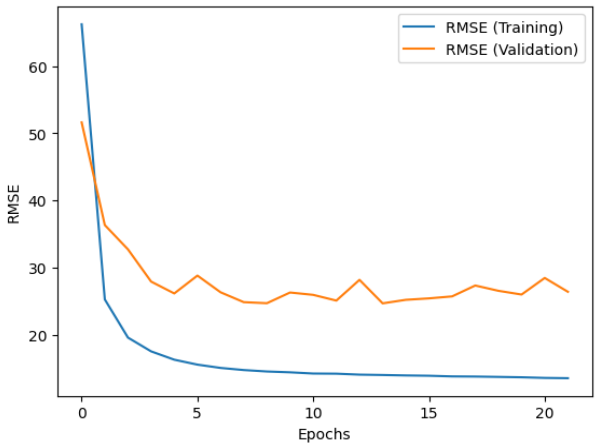


Figure 6.30: Simple LSTM Performance

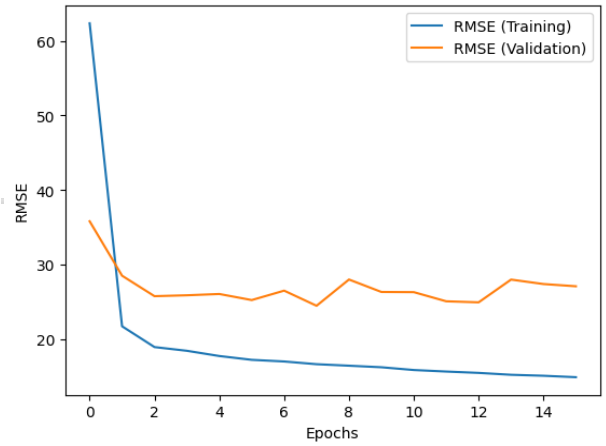


Figure 6.31: Regularized LSTM Performance

6.5.4 Final Model Selection

This study demonstrated that ensemble methods outperformed the LSTM neural network, achieving superior predictive accuracy Though the LSTM showed potential for sequential pattern recognition, its requirement for extensive tuning and computational resources made it less practical for this application compared to the robust, deployment-

ready ensemble approaches. The Stacked Ensemble model demonstrated superior performance, achieving an MAE of 1.933, RMSE of 2.821, and R^2 of 0.915. Since stacked models require maintaining multiple sub-models and meta-learners, leading to increased computational overhead, this study deployed the LightGBM model, which was the second-best model with an MAE of 1.93, RMSE of 2.841, and R-squared of 0.913.

6.6 System Design and Deployment

Chapter 4 of this research study provides a detailed description of the system design and architecture. The results of system implementation and testing are discussed in Chapter 5.

Chapter 7: Conclusions, Recommendations and Future Work

7.1 Conclusion

In this research, we successfully developed and deployed a LightGBM-based quantile regression model to predict $PM_{2.5}$ concentrations in Nairobi, Kenya. The system integrates real-time weather data from OpenWeatherMap with historical air quality measurements, processed through a robust data pipeline that handles missing values, outliers, and temporal inconsistencies. The deployed model demonstrated strong performance, achieving an RMSE of 3.21 µg/m^3 for 1-hour forecasting and 11.3 µg/m^3 for a 12-hour forecast horizon, with an R^2 of 0.913 on the test set.

Analysis of meteorological factors revealed weak correlations with $PM_{2.5}$, with temperature, wind speed, and dew point showing slight negative relationships, suggesting that higher temperatures and stronger winds may aid in pollutant dispersion. Conversely, pressure and cloud cover exhibited minor positive correlations, indicating that stable atmospheric conditions and cloudier weather might slightly contribute to $PM_{2.5}$ accumulation. Other factors, such as visibility, rainfall, and wind direction, showed minimal influence. These findings highlight the role of meteorological dynamics in pollutant dispersion and accumulation, though their impact varies based on local environmental conditions.

Additionally, the Streamlit-based web application provides environmental analysts with interactive 24-hour forecasts featuring confidence intervals and AQI-categorized outputs based on EPA standards. This work bridges the gap between theoretical machine learning and practical air quality management, offering a scalable tool for environmental analysts and policymakers.

7.1.1 Model Lifecycle Management

To ensure sustained model performance, this study will establish a structured lifecycle management framework. The deployed LightGBM model will undergo quarterly retraining cycles, with monthly monitoring for data and concept drift using statistical tests (e.g., Kolmogorov-Smirnov) on feature distributions. Model updates will follow a staged rollout protocol, including shadow deployment and A/B testing for at least two weeks before full production release. Performance degradation beyond predefined thresholds

(e.g., 10% increase in RMSE) will trigger immediate retraining. These procedures address critical operational requirements while balancing computational costs with model accuracy, ensuring reliable performance in real-world applications.

The implementation leverages version control and metadata tracking to maintain reproducibility and facilitate auditability, essential for both technical maintenance and regulatory compliance.This framework provides a practical approach to model maintenance, acknowledging the trade-offs between retraining frequency, computational resources, and performance stability in production environments.

7.2 Recommendations

To facilitate real-world adoption, this $PM_{2.5}$ prediction tool should be piloted with municipal authorities through a phased approach, beginning with a 3-month validation period in select urban districts. Embedding the tool into environmental monitoring and public health workflows can enhance air quality assessment and decision-making at the city level. Key enhancements necessary for effective implementation include mobile integration via a lightweight app or SMS interface to disseminate 1–6 hour forecasts for emergency alerts,especially in communities near pollution hotspots, allowing for timely issuance of AQI-driven health advisories. An automated alerting system tied to threshold breaches, such as those based on WHO $PM_{2.5}$ guidelines, should be integrated with existing public health platforms to ensure broad accessibility and actionable communication. Dynamic model updates through a municipal data pipeline that incorporates real-time traffic patterns, industrial emissions, and meteorological data will improve prediction accuracy and account for seasonal variability. Longer-range 24-hour forecasts can inform strategic planning activities, such as optimizing mobile monitoring routes or scheduling construction and dust-control measures during low-wind conditions.

Successful deployment will require cross-sector collaboration: environmental agencies for expanding sensor coverage and reducing dependence on manual data entry; urban planners for implementing spatial interventions such as green infrastructure in predicted pollution hotspots; and public health teams for conducting targeted outreach and resource allocation. Institutional barriers such as, data-sharing protocols and outdated IT systems, must be proactively addressed through policy frameworks that mandate standardized

data formats and interdepartmental coordination. The system's scalability will depend on expanding IoT sensor networks and reducing reliance on manual inputs. These investments should be supported by cost-benefit analyses that highlight potential healthcare savings, such as decreased asthma-related hospital admissions. Lastly, integrating historical data for long-term trend analysis will support the evaluation of policy effectiveness and continuous improvement of urban pollution mitigation strategies.

7.3 Future Work

This study identified several critical directions for advancement: although Sentinel-5P satellite data was evaluated to improve spatial coverage, its daily temporal resolution posed significant challenges when integrated with hourly ground measurements, risking the introduction of artificial patterns that could undermine model accuracy. Future implementations should prioritize emerging remote sensing technologies with native hourly resolution capabilities. The model's geographic scalability could be enhanced through transfer learning approaches, enabling adaptation to other regions while maintaining prediction reliability. While technology alone cannot solve air quality challenges, systems like this empower environmental professionals to make smarter, more targeted decisions - ultimately leading to cleaner air and healthier urban populations.

References

AlDaweesh, S. A. (2019). Predicting hourly particulate matter (pm2.5) concentrations using meteorological data. In *2019 International Conference on Computing, Electronics Communications Engineering (iCCECE)*, pages 136–140.

Amegah, A. K. and Jaakkola, J. J. K. (2016). Household air pollution and the sustainable development goals. *Bulletin of the World Health Organization*, 94:215 – 221.

Apte, K. K. and Salvi, S. (2016). Household air pollution and its effects on health. *F1000Research*, 5.

Bachechi, C., Desimoni, F., Po, L., and Casas, D. M. (2020). Visual analytics for spatio-temporal air quality data. In *2020 24th International Conference Information Visualisation (IV)*, pages 460–466.

Bhole, D. and Mesham, P. (2017). Implications of household air pollution in india on health: need of health technology. *Int J Healthc Educ Med Informatics*, 4:18–22.

Cohen, S. (2021). The basics of machine learning: strategies and techniques. *Artificial Intelligence and Deep Learning in Pathology*.

Ding, C., Wang, G., Zhang, X., Liu, Q., and Liu, X. (2021). A hybrid cnn-lstm model for predicting pm2.5 in beijing based on spatiotemporal correlation. *Environmental and Ecological Statistics*, 28(3):503–522.

Gaita, S. M., Boman, J., Gatari, M. J., Pettersson, J. B. C., and Janhäll, S. (2014). Source apportionment and seasonal variation of pm 2.5 in a sub-saharan african city: Nairobi, kenya. *Atmospheric Chemistry and Physics*, 14:9977–9991.

Guo, C., Liu, G., and Chen, C.-H. (2020). Air pollution concentration forecast method based on the deep ensemble neural network. *Wireless Communications and Mobile Computing*, 2020:8854649.

Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly. Appears in Collections: Information technology.

Hananto, V. R. and Putra, I. G. N. A. W. (2018). A dashboard system for monitoring air pollution in surabaya based on pm2.5. *Journal of Information Systems Engineering and Business Intelligence*.

Hassan, M. S. and Bhuiyan, M. A. H. (2023). Spatiotemporal mapping and modeling hotspot of pm2. 5 in the central part of bangladesh. *Journal of Hyperspectral Remote Sensing v*, 13(1):13–23.

J, P. (2019). Household air pollution and the integrated exposure-response function for the global burden of disease estimates: have we filled the gap? *Environmental Epidemiology*, 3.

Jing, C., Du, M., Li, S., and Liu, S. (2019). Geospatial dashboards for monitoring smart city performance. *Sustainability*, 11(20).

Jing, Z., Liu, P., Wang, T., Song, H., Lee, J., Xu, T., and Xing, Y. (2020). Effects of meteorological factors and anthropogenic precursors on pm2.5 concentrations in cities in china. *Sustainability*, 12(9).

Kaba, M., Wilkinson, R., Phillips, D. I. W., and Levene, D. S. (2019). Improving household air quality: The neglected cultural dimension. *Ethiopian Journal of Health Development*, 33.

Maknae, C. (2024). The impacts of air pollution on human health: A critical literature review. *American Journal of Natural Sciences*.

Mannucci, P. M. and Franchini, M. (2017). Health effects of ambient air pollution in developing countries. *International Journal of Environmental Research and Public Health*, 14.

Matara, C. M., Nyambane, S. O., Yusuf, A. O., Ochungo, E. A., and Khattak, A. (2024). Classification of particulate matter (pm) concentrations using feature selection and machine learning strategies. *LOGI – Scientific Journal on Transport and Logistics*, 15(1):85–96.

Medhi, S. and Gogoi, M. (2024). Pm2.5 concentration prediction using generative adversarial network: A novel approach. *Journal of Applied and Natural Science*.

Metrics, H. (2024). Global burden of disease 2021. *Findings from the GBD 2021 Study*.

Morapedi, T. D. and Obagbuwa, I. C. (2023). Air pollution particulate matter (pm2. 5) prediction in south african cities using machine learning techniques. *Frontiers in Artificial Intelligence*, 6.

Mudu, P., Adair-Rohani, H., deSouza, P., Gumy, S., de Sá, T. H., Lewis, J., Mwaura, A., Shairsingh, K., Vert, C., Williams, K., et al. (2023). Tracking urban health policies: a conceptual framework with special focus on air pollution in african cities.

Muinde, J., Tanga, C. M., Olukuru, J., Odhiambo, C., Tonnang, H. E. Z., and Senagi, K. (2023). Application of machine learning techniques to discern optimal rearing conditions for improved black soldier fly farming. *Insects*, 14(5).

Muthukumar, P., Cocom, E., Nagrecha, K., Comer, D., Burga, I., Taub, J., Calvert, C. F., Holm, J., and Pourhomayoun, M. (2022). Predicting pm2.5 atmospheric air pollution using deep learning with meteorological data and ground-based observations and remote-sensing satellite big data. *Air Quality, Atmosphere & Health*, 15(7):1221–1234.

Njeru, M. N., Mwangi, E., Kaniu, I., and Gatari, M. (2023). Application of remote sensing in determining particulate matter pollution. In *2023 IEEE Radio and Antenna Days of the Indian Ocean (RADIO)*, pages 1–2.

Ouma, Y. O., Keitsile, A., Lottering, L., Nkwae, B., and Odirile, P. (2024). Spatiotemporal empirical analysis of particulate matter pm2.5 pollution and air quality index (aqi) trends in africa using merra-2 reanalysis datasets (1980–2021). *Science of The Total Environment*, 912:169027.

Parganiha, V., Shukla, S. P., and Sharma, L. K. (2022). Developing decision making and risk mitigation: Using crisp-data mining. *Data Mining and Machine Learning Applications*, pages 281–315.

Racheeti, P. (2024). The silent killer: Addressing air pollution as a public health emergency- review. *International Journal For Multidisciplinary Research*.

Tékouabou, S. C., Chenal, J., Azmi, R., Diop, E. B., Toulni, H., and de Padoue Nsegbe, A. (2022). Towards air quality particulate-matter monitoring using low-cost sensor data

and visual exploration techniques: case study of kisumu, kenya. *Procedia Computer Science*, 215:963–972. 4th International Conference on Innovative Data Communication Technology and Application.

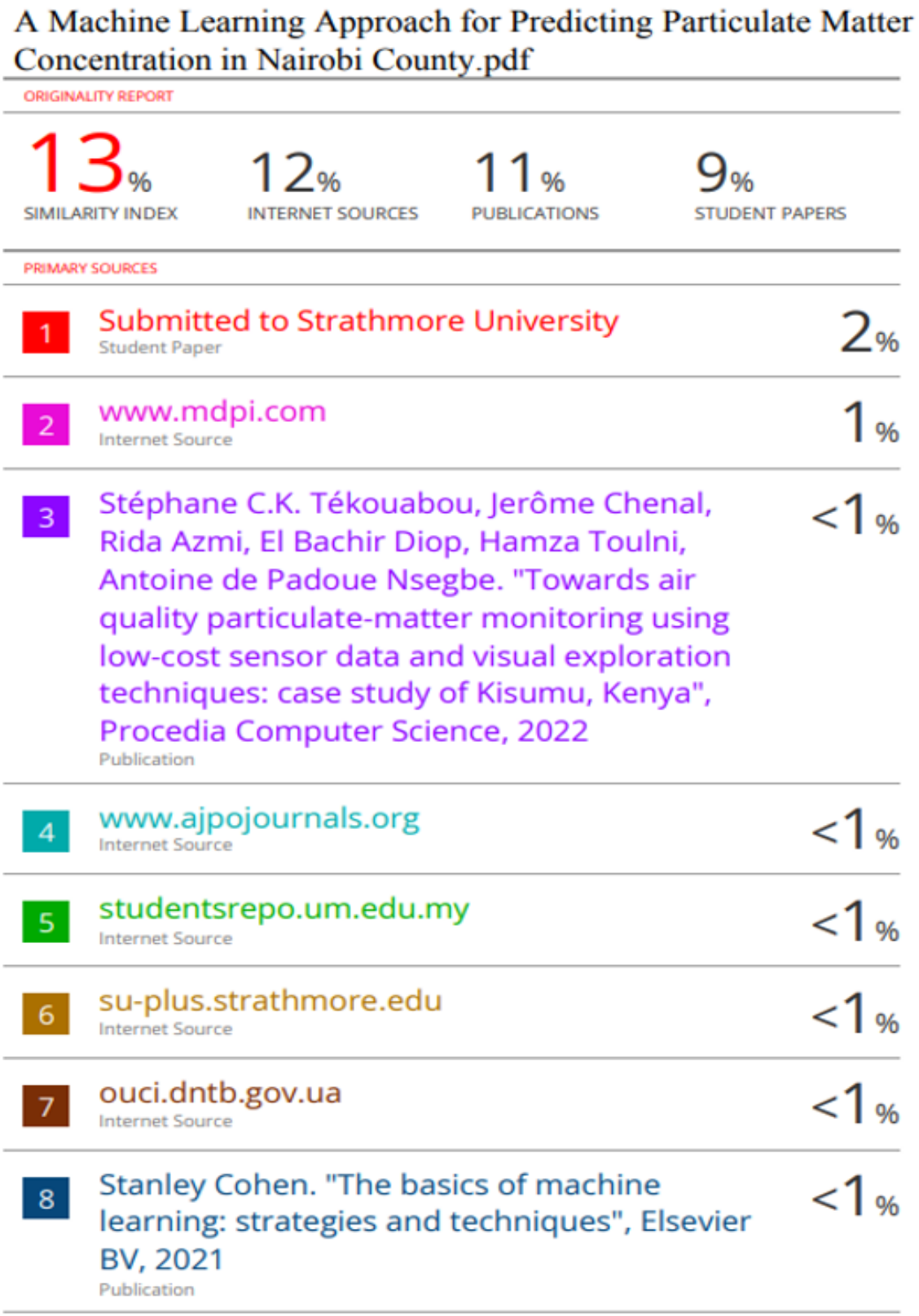
Wongpakdee, S., Nanta, P., and Kutintara, B. (2023). Concept and factors affecting well-being as perceived by bangkok homebuyers. *Environment-Behaviour Proceedings Journal*, 8(24):61–67.

Zamani Joharestani, M., Cao, C., Ni, X., Bashir, B., and Talebiesfandarani, S. (2019). Pm2.5 prediction based on random forest, xgboost, and deep learning using multisource remote sensing data. *Atmosphere*, 10(7).

Zhang, A., Chen, S., Zhao, F., and Dai, X. (2022). Good environmental governance: Predicting pm2.5 by using spatiotemporal matrix factorization generative adversarial network. In *Frontiers in Environmental Science*.

Appendices

Appendix A: Similarity Report



A Machine Learning Approach for Predicting Particulate Matter Concentration in Nairobi County.pdf

ORIGINALITY REPORT

13%	12%	11%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Strathmore University Student Paper	2%
2	www.mdpi.com Internet Source	1%
3	Stéphane C.K. Tékouabou, Jerôme Chenal, Rida Azmi, El Bachir Diop, Hamza Toulni, Antoine de Padoue Nsegbe. "Towards air quality particulate-matter monitoring using low-cost sensor data and visual exploration techniques: case study of Kisumu, Kenya", Procedia Computer Science, 2022 Publication	<1%
4	www.ajpojournals.org Internet Source	<1%
5	studentsrepo.um.edu.my Internet Source	<1%
6	su-plus.strathmore.edu Internet Source	<1%
7	ouci.dntb.gov.ua Internet Source	<1%
8	Stanley Cohen. "The basics of machine learning: strategies and techniques", Elsevier BV, 2021 Publication	<1%

9	jpoll.ut.ac.ir Internet Source	<1 %
10	www.research-collection.ethz.ch Internet Source	<1 %
11	Submitted to Universiti Tunku Abdul Rahman Student Paper	<1 %
12	unhabitat.org Internet Source	<1 %
13	ir.lib.hiroshima-u.ac.jp Internet Source	<1 %
14	www.frontiersin.org Internet Source	<1 %
15	www.coursehero.com Internet Source	<1 %
16	Xin Zhu, Qingcai Chen, Tong Sha, Yue Yin, Jinwen Li, Zimeng Zhang, Jiale Ding, Tengfei Xu. "Unexpected Changes in Occurrence and Sources of Chromophoric Dissolved Organic Matter in PM2.5 Driven by the Clean Air Action over Xi'an, China", Atmospheric Environment, 2025 Publication	<1 %
17	Submitted to The University of Manchester Student Paper	<1 %
18	Submitted to University of South Africa (UNISA) Student Paper	<1 %
19	www.icipe.org Internet Source	<1 %

Appendix B: Ethical Clearance Confirmation

Strathmore UNIVERSITY

20th February 2025

Ms Eyinda Christine,
christine.eyinda@strathmore.edu

Dear Ms Eyinda,

RE: A Machine Learning Approach for Predicting Particulate Matter Concentration in Nairobi County

This is to inform you that SU-ISERC has reviewed and **approved** your above **SU-masters** proposal. Your application reference number is **SU-ISERC2574/25.** The approval period is from **20th February 2025 to 19th February 2026.**

This approval is subject to compliance with the following requirements:

i. Only approved documents including (informed consents, study instruments, MTA) will be used.
ii. All changes including (amendments, deviations, and violations) are submitted for review and approval by SU-ISERC.
iii. Death and life-threatening problems and serious adverse events or unexpected adverse events whether related or unrelated to the study must be reported to SU-ISERC within 72 hours of notification.
iv. Any changes anticipated or otherwise that may increase the risks or affected safety or welfare of study participants and others or affect the integrity of the research must be reported to SU-ISERC within 72 hours.
v. Clearance for the export of biological specimens must be obtained from relevant institutions.
vi. Submission of a request for renewal of approval at least 60 days prior to the expiry of the approval period. Attach a comprehensive progress report to support the renewal.
vii. Submission of an executive summary report within 90 days of completion of the study to SU-ISERC.

Before commencing your study, you will be expected to obtain a research license from National Commission for Science, Technology, and Innovation (NACOSTI) https://research-portal.nacosti.go.ke/ and obtain other clearances needed.

Yours sincerely,

Mr Ambrose Rachier,
Chairperson; SU-ISERC

Ole Sangale Rd, Madaraka Estate. PO Box 59857-00200, Nairobi, Kenya. Tel +254 (0)703 034000
Email admissions@strathmore.edu www.strathmore.edu

Appendix C: Main.py

```
import streamlit as st
import pandas as pd
from streamlit_option_menu import option_menu
from data_preprocessing import (read_data, create_lag_features,
                                train_test_split_data, load_models,
                                validate_nairobi_location, geocode_location,fetch_weather)
from visuals import (get_aqi_category, get_aqi_color, display_forecast_results)
import numpy as np
from datetime import datetime, timedelta, time
import math
import random
import plotly.graph_objects as go

# Page configuration
st.set_page_config(page_title="Nairobi PM2.5 Prediction Tool", page_icon="đ$", layout="wide")

# Sidebar navigation
with st.sidebar:
    selected = option_menu(None, ["Home","Methodology", "Prediction"],
                           icons=['house', 'book','gear'],
                           menu_icon="cast", default_index=0,
                           styles={
                               "icon": {"color": "orange", "font-size": "16px"},
                               "nav-link": {"font-size": "15px", "text-align": "left",
"margin":"0px", "--hover-color": "#eee"},
                               "nav-link-selected": {"background-color": "green"},
                           })

    with st.expander("â How to Use This Tool"):
        st.markdown("""
        1. **Navigate to the prediction tab.**
        2. **Choose your forecast start date (default: today)**
        3. **Select start hour (24-hour format)**
        4. **Enter PMâ.â measurements from the last 5 hours (in ÂIg/mÂł)**
        5. Click **Generate Forecast**
        """)
        st.table(pd.DataFrame({
            "AQI Category": ["Good", "Moderate", "Unhealthy", "Very Unhealthy", "Hazardous"],
            "PM2.5 Range": ["0-12", "12-35", "35-55", "55-150", "250+"],
            "Color": ["đ˜", "đĄ", "đ ", "đ´", "đ¤"]
        }))


if selected == "Home":
    st.title("PM2.5 Forecasting System")
    st.markdown("""
    ### Nairobi Air Quality Prediction
    This system provides 24-hour PM2.5 forecasts with confidence intervals.
    """)


    st.subheader(":orange[Air Quality Sensor Network]")

    # Sensor coordinates data
    sensor_locations = {
        'Latitude': [-1.33, -1.327, -1.322, -1.32, -1.316, -1.316, -1.316, -1.316,
                     -1.306, -1.306, -1.303, -1.303, -1.301, -1.3, -1.298, -1.297,
                     -1.297, -1.296, -1.295, -1.295, -1.293, -1.292, -1.291, -1.291,
                     -1.291, -1.29, -1.289, -1.288, -1.287, -1.283, -1.27, -1.267,
                     -1.265, -1.261, -1.261, -1.26, -1.259, -1.253, -1.251, -1.239,
                     -1.235, -1.22, -1.218, -1.215],
        'Longitude': [36.866, 36.882, 36.797, 36.885, 36.79, 36.793, 36.87, 36.872,
                      36.733, 36.773, 36.789, 36.829, 36.754, 36.785, 36.791, 36.743,
                      36.755, 36.776, 36.777, 36.86, 36.769, 36.821, 36.725, 36.733,
                      36.781, 36.777, 36.825, 36.841, 36.811, 36.828, 36.801, 36.8,
                      36.857, 36.772, 36.782, 36.793, 36.799, 36.854, 36.923, 36.791,
                      36.854, 36.879, 36.887, 36.862]
```

```
    }

    sensor_df = pd.DataFrame(sensor_locations)
    st.map(sensor_df, latitude='Latitude', longitude='Longitude', color='#FF4B4B', size=15,
zoom=10)
    st.caption("Figure 1: Distribution of air quality sensors used in this study")

elif selected == "Prediction":
    st.markdown("### :orange[24-Hour PM2.5 Forecast with Confidence Intervals]")

    # Initialize session state for predictions if it doesn't exist
    if 'forecast_results' not in st.session_state:
        st.session_state.forecast_results = None

    # Load models
    models = load_models()
    if not models:
        st.stop()

    # Fixed Nairobi coordinates
    NAIROBI_COORDS = (-1.286389, 36.817223)
    today = datetime.now().date()  # Define 'today' variable

    # Compact form with border
    with st.form('pm25_form', border=True):
        # Date and Hour Input section (formerly in expander)
        col1, col2 = st.columns(2)

        with col1:
            # Enhanced date input with tomorrow as default
            min_date = today
            max_date = today + timedelta(days=5)  # 5-day forecast limit
            default_date = today  # Default to today

            forecast_date = st.date_input(
                "Choose your forecast start date (any date from today through [today + 5
days]:",
                min_value=min_date,
                max_value=max_date,
                value=default_date,
                help="Weather data available for up to 5 days in advance"
            )

            # Show warning if trying to forecast beyond API limits
            if forecast_date > today + timedelta(days=5):
                st.warning("Note: Forecast limited to 5 days ahead")

            start_hour = st.slider("Choose the starting hour:", 0, 23, 8, help="Use the 24-
hour clock system")  # Default to 8 AM

        with col2:
            # Weather data handling
            if forecast_date == today + timedelta(days=1):  # If forecasting tomorrow
                st.info("""
                **Weather Data Note:**
                Using today's latest observed weather as baseline,
                with tomorrow's forecasted weather patterns
                """)

        # Location display (fixed to Nairobi)
        # st.markdown(f"**Location:** Nairobi (Lat: {NAIROBI_COORDS[0]:.4f}, Lon:
{NAIROBI_COORDS[1]:.4f})")

        # PM2.5 history inputs (formerly in expander)
        st.markdown("**Enter PM2.5 measurements from the last 5 hours (in ÂÏg/mÂł)**")
        pm_cols = st.columns(5)
        pm_history = []
```

```
        for i, col in enumerate(pm_cols, 1):
            with col:
                pm_history.append(
                    st.number_input(f"{i} hour ago",
                                    min_value=0.0,
                                    max_value=100.0,
                                    value=15.0,
                                    key=f"pm_{i}")
                )

        submitted = st.form_submit_button("Generate Forecast")

    if submitted:
        try:
            # Get weather data (either current or forecast)
            if forecast_date == today + timedelta(days=1):  # Tomorrow's forecast
                current_weather = fetch_weather(*NAIROBI_COORDS, datetime.now())
                forecast_weather = fetch_weather(*NAIROBI_COORDS,
                                                 datetime.combine(forecast_date,
time(start_hour, 0)))
                weather = {
                    **current_weather,
                    **{k: forecast_weather[k] for k in ['temperature', 'wind_speed',
'humidity']}
                }
            else:  # Today or other dates
                weather = fetch_weather(*NAIROBI_COORDS, datetime.combine(forecast_date,
time(start_hour, 0)))
                if weather.get('data_source') == 'Simulated Data':
                    st.error("â ď¸ Using simulated weather data (API unavailable)")
                else:
                    st.success("â Live weather data loaded")

            # API status feedback
            if weather.get('data_source', '').startswith('OpenWeatherMap'):
                st.toast("đ Data successfully retrieved from API", icon="â")

            base_time = datetime.combine(forecast_date, time(start_hour, 0))
            hours = [base_time + timedelta(hours=i) for i in range(24)]

            predictions = []
            current_lags = pm_history.copy()

            with st.spinner("Generating forecast..."):
                progress_bar = st.progress(0)

                for i, hour in enumerate(hours):
                    # Use the weather data we already fetched
                    features = {
                        'lag_1': current_lags[-1],
                        'lag_2': current_lags[-2],
                        'lag_3': current_lags[-3],
                        'lag_4': current_lags[-4],
                        'lag_5': current_lags[-5],
                        'hour': hour.hour,
                        'week': hour.isocalendar()[1],
                        'year': hour.year,
                        'dew_point': weather['dew_point'],
                        'wind_speed': weather['wind_speed'],
                        'wind_deg': weather['wind_deg'],
                        'pressure': weather['pressure'],
                        'humidity': weather['humidity'],
                        'temperature': weather['temperature'],
                        'temp_max': weather['temp_max']
                    }

                    input_df = pd.DataFrame([features])
```

Appendix D: Data preprocessing.py

```
import streamlit as st
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from joblib import load
import os
import requests
import math
import random
from datetime import datetime
from geopy.geocoders import Nominatim
from geopy.exc import GeocoderTimedOut, GeocoderServiceError
import time
from dotenv import load_dotenv
load_dotenv()

NAIROBI_BOUNDS = {
    'min_lat': -1.47,
    'max_lat': -1.15,
    'min_lon': 36.65,
    'max_lon': 37.05
}

@st.cache_data
def read_data():
    historical_data = pd.read_csv(

'https://raw.githubusercontent.com/Lorraine254/Data/refs/heads/main/nairobi_pm25_weather_historical.csv',

        parse_dates=['Unnamed: 0']
    )
    historical_data.rename(columns={'Unnamed: 0': 'timestamp'}, inplace=True)
    historical_data['timestamp'] = pd.to_datetime(historical_data['timestamp']).dt.tz_localize(None)
    historical_data['hour'] = historical_data['timestamp'].dt.hour
    historical_data['year'] = historical_data['timestamp'].dt.year
    historical_data['week'] = historical_data['timestamp'].dt.isocalendar().week
    return historical_data

@st.cache_data
def create_lag_features(df, lag=5):
    for i in range(1, lag+1):
        df[f'lag_{i}'] = df['pm2.5'].shift(i)
    df.dropna(inplace=True)
    return df

@st.cache_data
def train_test_split_data(df):
    df_lagged = create_lag_features(df, lag=5)
    X = df_lagged.drop(columns=['pm2.5'])
    y = df_lagged['pm2.5']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    return X_train, X_test, y_train, y_test

@st.cache_resource
@st.cache_resource
@st.cache_resource
def load_models():
    """Load all LightGBM models (main + quantile)"""
    try:
        models = load('lightgbm_models.joblib')
        return {
            'main': models['main_model'],
            'upper': models['upper_model'],
            'lower': models['lower_model']
        }
    except Exception as e:
        st.error(f"Model loading failed: {str(e)}")
        return None

def fetch_weather(lat, lon, target_time):
    """Fetch weather data with robust error handling"""
    # Fixed Nairobi coordinates
    NAIROBI_COORDS = (-1.286389, 36.817223)
```

```
    # Always use Nairobi coordinates regardless of input
    lat, lon = NAIROBI_COORDS

    try:
        # For current weather (remove time parameter)
        if target_time.date() == datetime.now().date():
            response = requests.get(
                "https://api.openweathermap.org/data/2.5/weather",
                params={
                    'lat': lat,
                    'lon': lon,
                    'appid': os.getenv('OPENWEATHER_API_KEY'),
                    'units': 'metric'
                },
                timeout=10
            )
            response.raise_for_status()
            data = response.json()

            return {
                'dew_point': data.get('main', {}).get('dew_point', 15),
                'wind_speed': data.get('wind', {}).get('speed', 3),
                'wind_deg': data.get('wind', {}).get('deg', 0),
                'pressure': data.get('main', {}).get('pressure', 1013),
                'humidity': data.get('main', {}).get('humidity', 50),
                'temperature': data.get('main', {}).get('temp', 20),
                'temp_max': data.get('main', {}).get('temp_max', 23),
                'data_source': 'OpenWeatherMap Current'
            }
        # For forecast data (up to 5 days)
        else:
            response = requests.get(
                "https://api.openweathermap.org/data/2.5/forecast",
                params={
                    'lat': lat,
                    'lon': lon,
                    'appid': os.getenv('OPENWEATHER_API_KEY'),
                    'units': 'metric',
                    'cnt': 40  # Number of timestamps (5 days in 3-hour intervals)
                },
                timeout=10
            )
            response.raise_for_status()
            data = response.json()

            # Find the closest forecast time to our target
            closest = min(data['list'], key=lambda x: abs(datetime.fromtimestamp(x['dt']) -
target_time))

            return {
                'dew_point': closest.get('main', {}).get('dew_point', 15),
                'wind_speed': closest.get('wind', {}).get('speed', 3),
                'wind_deg': closest.get('wind', {}).get('deg', 0),
                'pressure': closest.get('main', {}).get('pressure', 1013),
                'humidity': closest.get('main', {}).get('humidity', 50),
                'temperature': closest.get('main', {}).get('temp', 20),
                'temp_max': closest.get('main', {}).get('temp_max', 23),
                'data_source': 'OpenWeatherMap Forecast'
            }

    except Exception as api_error:
        error_msg = str(api_error)
        if hasattr(api_error, 'response'):
            error_msg += f" (Status: {api_error.response.status_code})"

        st.warning(f"â  Using simulated Nairobi data (API error: {error_msg[:150]})")

        # Generate realistic Nairobi-like simulated data
        hour = target_time.hour
        return {
            'dew_point': 12.0 + 4*math.sin(hour/24*2*math.pi),
            'wind_speed': 2.5 + random.uniform(-0.5, 0.5),
```

```
            'wind_deg': random.randint(90, 270),
            'pressure': 1015 + random.randint(-5, 5),
            'humidity': 60 + int(15*math.sin(hour/12*math.pi)),
            'temperature': 18.0 + 6*math.sin(hour/24*2*math.pi),
            'temp_max': 22.0 + 4*math.sin(hour/24*2*math.pi),
            'data_source': 'Simulated Nairobi Data'
        }

@st.cache_data
def validate_nairobi_location(lat, lon):
    return (NAIROBI_BOUNDS['min_lat'] <= lat <= NAIROBI_BOUNDS['max_lat'] and
            NAIROBI_BOUNDS['min_lon'] <= lon <= NAIROBI_BOUNDS['max_lon'])

@st.cache_data
def geocode_location(location_name):
    geolocator = Nominatim(user_agent="nairobi_pm25_app")
    max_retries = 3
    retry_delay = 1

    for attempt in range(max_retries):
        try:
            location = geolocator.geocode(location_name + ", Nairobi, Kenya")
            if location:
                return location.latitude, location.longitude
            return None
        except (GeocoderTimedOut, GeocoderServiceError) as e:
            if attempt == max_retries - 1:
                st.error(f"Geocoding failed: {str(e)}")
                return None
            time.sleep(retry_delay)
    return None
```

Appendix E: Visuals.py

```
import plotly.express as px
import plotly.graph_objects as go
from datetime import datetime
import streamlit as st
import pandas as pd



def plot_model_results(df, metric):
    """Create comparison plots for models"""
    if metric == 'rmse':
        title = "RMSE Comparison"
        metric_cols = ['train_rmse', 'test_rmse']
        y_title = "RMSE"
    elif metric == 'r2':
        title = "RÂ² Comparison"
        metric_cols = ['train_r2', 'test_r2']
        y_title = "RÂ² Score"
    else:
        raise ValueError(f"Unsupported metric: {metric}")

    # Melt the DataFrame for plotting
    plot_df = df.melt(
        id_vars=['model_name'],
        value_vars=metric_cols,
        var_name='dataset',
        value_name=metric
    )

    # Clean up dataset names
    plot_df['dataset'] = plot_df['dataset'].str.replace('train_', 'Training
').str.replace('test_', 'Test ')

    fig = px.bar(
        plot_df,
        x='model_name',
        y=metric,
        color='dataset',
        barmode='group',
        title=title,
        text_auto='.3f'
    )
    fig.update_layout(
        xaxis_title="Model",
        yaxis_title=y_title,
        legend_title="Dataset"
    )
    return fig

def get_aqi_category(pm25):
        """Classify PM2.5 into AQI categories"""
        if pm25 < 12:
            return "Good"
        elif pm25 < 35:
            return "Moderate"
        elif pm25 < 55:
            return "Unhealthy for Sensitive Groups"
        elif pm25 < 150:
            return "Unhealthy"
        elif pm25 < 250:
            return "Very Unhealthy"
        else:
            return "Hazardous"

def get_aqi_color(pm25):
    """Get color for AQI category"""
    if pm25 < 12:
```

```
        return '#00E400'  # Green
    elif pm25 < 35:
        return '#FFFF00'  # Yellow
    elif pm25 < 55:
        return '#FF7E00'  # Orange
    elif pm25 < 150:
        return '#FF0000'  # Red
    elif pm25 < 250:
        return '#8F3F97'  # Purple
    else:
        return '#7E0023'  # Maroon

def display_forecast_results(results_df):
    st.success("Forecast generated successfully!")

    # Show data source information
    if 'data_source' in results_df.columns:
        source = results_df['data_source'].iloc[0]
        if source == 'Simulated Data':
            st.warning("""
            **Note:** Using simulated weather data because:
            - Historical API only covers last 5 days
            - Try recent dates for real weather data
            """)
        else:
            st.info(f"Using real weather data from {source}")

    # Create two tabs for different views
    tab1, tab2 = st.tabs(["đ Interactive Chart", "đ Data Table"])

    with tab1:
        fig = go.Figure()

        # Confidence interval band (shown first so it's behind)
        fig.add_trace(go.Scatter(
            x=results_df['timestamp'],
            y=results_df['upper_95'],
            name='95th Percentile',
            line=dict(width=0),
            fillcolor='rgba(0, 100, 255, 0.2)',
            fill='tonexty'
        ))

        fig.add_trace(go.Scatter(
            x=results_df['timestamp'],
            y=results_df['lower_05'],
            name='5th Percentile',
            line=dict(width=0),
            showlegend=False
        ))

        # Main prediction line
        fig.add_trace(go.Scatter(
            x=results_df['timestamp'],
            y=results_df['prediction'],
            name='Predicted PM2.5',
            line=dict(color='blue', width=2)
        ))

        fig.update_layout(
            title='PM2.5 Forecast with 90% Confidence Interval',
            xaxis_title='Time',
            yaxis_title='PM2.5 (ÃIg/mÅł)',
            hovermode="x unified",
            height=600,
            showlegend=True
        )
```

```
        st.plotly_chart(fig, use_container_width=True)

    with tab2:
        st.markdown("### Forecast Data with Confidence Intervals")

        # Format the dataframe for display
        display_df = results_df.copy()
        display_df['Prediction'] = display_df['prediction'].round(2)
        display_df['95% Upper'] = display_df['upper_95'].round(2)
        display_df['5% Lower'] = display_df['lower_05'].round(2)
        display_df['AQI Category'] = display_df['aqi_category']

        # Create a style function with more subtle highlighting
        def highlight_aqi(row):
            color = get_aqi_color(row['Prediction'])
            # Lighten the color by adding opacity
            light_color = color + '33'  # Adds 20% opacity (33 in hex)
            return [f'background-color: {light_color}' if col == 'AQI Category' else '' for
col in row.index]

        # Apply styling only to the AQI Category column
        st.dataframe(
            display_df[['timestamp', 'Prediction', '5% Lower', '95% Upper', 'AQI
Category']].style.apply(
                highlight_aqi,
                axis=1
            ),
            height=600
        )
```