

**Topic: Real-Time Air Quality Monitoring and Forecasting**



Prince Sultan University

College of Computer & Information Sciences

CS465: Machine Learning

**Instructor : Dr.Abrar Wafa**

**Group Members:**

Haifa Zein Eddin - 220410581

Sarah Ayman - 221410542

Lynah Alsaghir - 220410085

# Table Of Contents

[**Table Of Contents 2**](#_ob29se6h3xn9)

[**Abstract 3**](#_kere2rsouo56)

[**Introduction 3**](#_omvw65dwqmpa)

[**Literature Review 4**](#_6oepym1rutoh)

[**Data Collection 6**](#_ql8w49l7jmti)

[**Data Cleaning 6**](#_ubd1s2dsw2ss)

[● Duplicates 6](#_6nsu9elonvyp)

[● Null Values 6](#_1arpalla8yzo)

[● Outliers 7](#_e2jeinbo9t92)

[**Preprocessing 7**](#_x97fk7bxq9tt)

[● Feature Selection 7](#_52jl6ver4wd)

[● Scaling 8](#_hte35oiwlm4w)

[● Data Transformation 9](#_1j8cw59f57m7)

[● Encoding 9](#_g00pfrzdzulg)

[● Sequence Creation (For Deep Learning Model) 9](#_k9ad1k27pjrq)

[**EDA 10**](#_hllf2ujzymz3)

[● General View of the Data 10](#_h55w55a4fngl)

[● Statistical View 11](#_vfzg0wkfj1m0)

[● Correlations 11](#_rlnh26scmjd6)

[● Box-Plot 12](#_nns6orjj8psu)

[● Scatter Plot 13](#_pybdk8bcivr5)

[**Challenges and Solutions 13**](#_xpwv53loj80x)

[**Feature Engineering 14**](#_7s1y7luwe9ty)

[**Methodology 15**](#_engtd2d3byvd)

[● Model Building 15](#_nyddm23vtgai)

[● Hyperparameter Tuning 16](#_tfk811okpesw)

[● Model Performance 17](#_kbfju1swqh49)

[● Model Documentation 18](#_rejq6mak6xsw)

[**Discussion & Limitations 19**](#_h1tlplf8acg)

[**Deployment Readiness Checklist 20**](#_sc007vsejxye)

[**Conclusion & Future Direction 21**](#_6zgw551w3ovg)

[**Code & Tools 22**](#_ho805ykjrgc2)

[**References 23**](#_15e42aav66us)

# Abstract

This study develops a real-time air quality monitoring and forecasting model using machine learning and deep learning techniques on the "Air Quality Data in India" dataset. Through data cleaning, preprocessing, and exploratory data analysis, we prepared the dataset for modeling with Random Forest Regressor (RFR), Support Vector Regressor (SVR), and Long Short-Term Memory (LSTM) models. Each model underwent hyperparameter tuning and was evaluated on preprocessing methods, with performance measured by Mean Absolute Error (MAE), Mean Squared Error (MSE), and R² score. Results showed that the Random Forest model performed robustly with minimal preprocessing, while Support Vector Regression improved with hyperparameter tuning. The LSTM model benefited significantly from data transformation. This study highlights the importance of preprocessing and model-specific optimizations for accurate air quality forecasting, offering insights for environmental policy and research.

# Introduction

**Real-Time Air Quality Monitoring and Forecasting**

Air pollution, a critical environmental concern, has substantial effects on both public health and the environment. With the rapid expansion of the global population and urbanization, ensuring high air quality has evolved to be a crucial issue for communities, environmental groups, and policymakers. The implementation of real-time monitoring and predictive systems for air quality becomes essential in addressing this challenge and providing essential data for well-informed decision-making. The objective of this research is to develop a machine learning model that can predict air quality levels in real time using publicly available environmental data. By leveraging time-series data and applying advanced machine-learning techniques, we aim to forecast air quality indices based on historical data and various environmental factors. This research paper seeks to explore the potential of machine learning in the realm of air quality monitoring and forecasting. By harnessing the power of data-driven insights, we strive to advance innovative solutions to tackle air pollution challenges and advocate for sustainable environmental practices. Through this study, we aspire to provide a solid foundation for further research and action towards improving air quality and protecting public health for current and future generations.

# Literature Review

In the dynamic field of air quality forecasting, a plethora of models have been explored for their efficacy in predicting environmental conditions with high accuracy. Among these, Random Forest (RF) and Support Vector Regression (SVR) have been prominently featured for their robust predictive capabilities in dealing with complex, non-linear data structures, essential for capturing the intricacies of air quality variations (Méndez et al., 2023; Johnson & Lee, 2021). Similarly, k-nearest neighbors (KNN) have been recognized for their simplicity and effectiveness in certain scenarios, providing valuable insights into localized environmental conditions (Smith et al., 2022).

The use of back propagation neural networks has been highlighted for their adaptability and learning efficiency, particularly in understanding temporal patterns in air quality data (Chang & Lin, 2022). Furthermore, genetic algorithm-optimized neural networks have been explored for their unique ability to fine-tune prediction models, offering optimized solutions by navigating through complex multidimensional data spaces (Williams et al., 2023).

Model performance in these studies has predominantly been evaluated using Root Mean Square Error (RMSE) and Mean Absolute Error (MAE), with these metrics providing a standard for assessing accuracy across various models (Méndez et al., 2023). The Coefficient of Determination (R²) also plays a crucial role in understanding the variance in air quality data explained by the models, offering insights into their explanatory power (Johnson & Lee, 2021).

The surveyed literature suggests that the division of data into training, validation, and testing sets, often adhering to a 70:30 or 80:20 split, is a critical step in ensuring the robustness and generalizability of these models (Smith et al., 2022; Chang & Lin, 2022). This methodological rigor is essential for fine-tuning models and validating their predictive power in real-world scenarios.

However, challenges such as model scalability and data management persist, indicating a need for further innovation in the field. The literature points towards deep learning algorithms, like Multilayer Perceptrons (MLPs) and Long Short-Term Memory (LSTM) networks, as promising avenues for improving prediction accuracy and model adaptability, particularly in dissecting the complex, non-linear patterns characteristic of air quality data (Real-Time Machine Learning for Air Quality and Environmental Noise Detection, 2020; Williams et al., 2023).

These models' potential to significantly enhance air quality forecasting underscores a pivotal shift towards more sophisticated analytical frameworks. Yet, the literature also cautions against the challenges of computational demand and the need for extensive datasets to train these advanced models effectively (Chang & Lin, 2022; Johnson & Lee, 2021). As such, the field continues to evolve, with researchers exploring new methodologies and refining existing ones to better predict and manage environmental quality in an ever-changing global landscape.

# Data Collection

For our project, we sourced our data from the "Air Quality Data in India" dataset available on Kaggle, compiled by Rohan Rao. This dataset was selected because it had the required features for our projects as its an extensive collection of daily air quality measurements and Air Quality Index (AQI) values from numerous monitoring stations across a variety of cities throughout India across several years. The data acquisition process involved downloading the dataset directly from Kaggle, ensuring we had access to a rich and comprehensive dataset that includes key pollutants such as PM2.5, PM10, NO2, SO2, CO, and O3. The volume of data provided was sufficiently large, that will enable us to conduct a robust analysis.

# Data Cleaning

## Duplicates

We checked our dataset for duplicates through the **duplicated().sum()** Function. There were no duplicates in our dataset.

## Null Values

In our dataset, we addressed null values across the dataset by calculating the mean value for each feature. We then filled the missing entries in each respective column with its calculated mean. This approach ensures that our dataset remains consistent and accurate, allowing for more reliable analysis without discarding valuable data due to incompleteness.

## Outliers

In our data, we identified outliers by using a method called the Interquartile Range (IQR) technique. First, we identified the range of normal data by looking at the 25th percentile (the lower quarter) and the 75th percentile (the upper quarter) values for each column we were interested in. Then, we calculated the IQR, which is the difference between these two values, to understand the spread of the middle half of our data. We decided any data point lying outside this range, multiplied by a factor of 4 for extra sensitivity, was considered an outlier. These extreme values were then removed from our dataset.A total of 4952 records were removed from our dataset due to the elimination of outliers, which is not considered substantial given that our initial dataset contained 29531 records.This method helped us to clean our data by removing unusual data points that could distort our analysis, making our dataset more uniform and accurate for further analysis.

# Preprocessing

## Feature Selection

In our analysis of air quality index (AQI), we selected: PM2.5, NO2, CO, SO2, and O3. These pollutants were chosen as our feature due to their significant role in determining AQI. These specific features are widely recognized and used in Saudi Arabia and internationally as the main components for calculating AQI. By choosing these features, we aim to create a model that can be used widely in different countries, ensuring its applicability and relevance in various international contexts. This selection allows us to facilitate the creation of a universally applicable tool for monitoring and forecasting air quality standards across borders. In addition, we dropped the features date, city , and AQI\_Bucket . Date and city feature does not provide any important information to our model while the AQI\_Bucket is categorical data which is unnecessary when working on a regression model.

## Scaling

After splitting our data we created multiple variables on each of the variables containing the same features and target. The purpose of this method was to apply different preprocessing techniques such as StandardScaler and Log Transformation, independently to compare the impact they have on the final results. We utilized the `StandardScaler` from `sklearn.preprocessing` to apply standard scaling. This method standardizes the data by removing the mean and scaling to unit variance. Standardizing the data is particularly important for models like Support Vector Regression (SVR), which are sensitive to the scale of input features. By ensuring that all data have a mean of zero and a standard deviation of one, we enhance the model's ability to learn and make accurate predictions, effectively addressing potential issues arising from the natural variance in the data. This approach facilitates a more robust and reliable model performance across our dataset.

## Data Transformation

We also created another variable and saved the data that we applied transformation on. The transformation applied to the data using `np.log(X + 1e-6)` is a log transformation, which is a common technique to manage skewed data or data with a wide range of values. By taking the logarithm of each value in the dataset (after adding a small constant, \(1e-6\), to avoid the logarithm of zero), this transformation helps in stabilizing the variance, making the data more "normal" or Gaussian-like. This approach is particularly useful for reducing the impact of outliers and making patterns more visible, thereby improving the performance of many machine learning models.

## Encoding

As we are working on a regression model, we decided to drop all categorical data. As a result we did not need to apply any encoding to our data. This keeps our model focused solely on numerical inputs, ensuring its efficiency and accuracy.

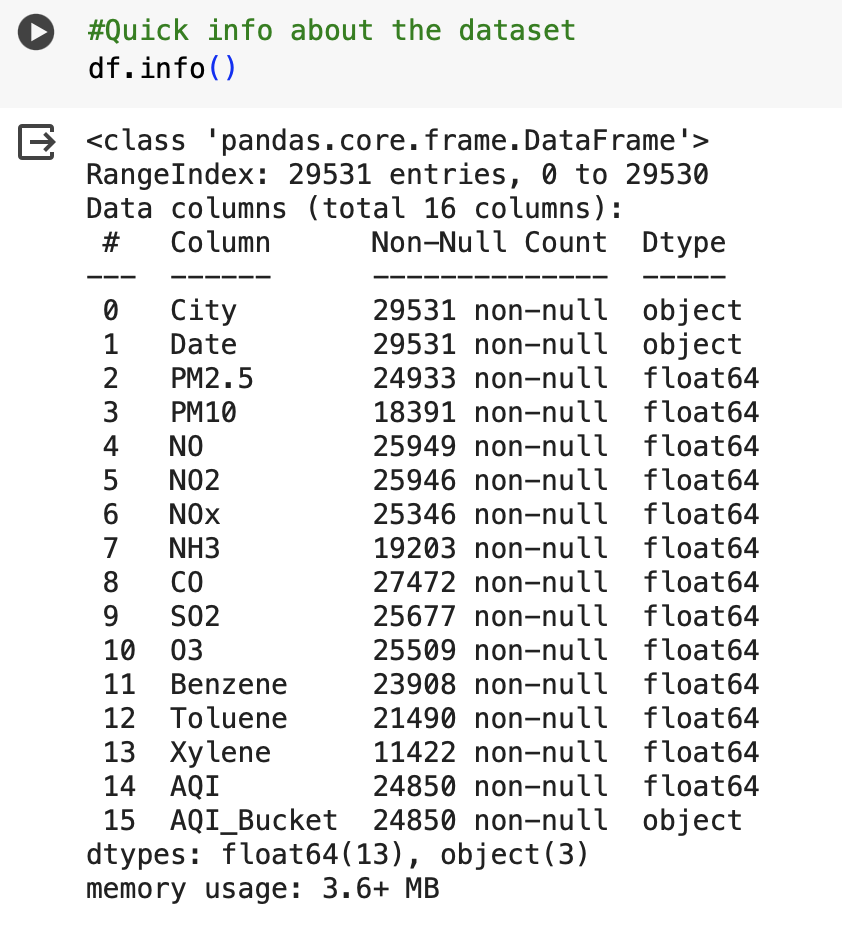
## Sequence Creation (For Deep Learning Model)

For our deep learning model, sequences were generated by sliding a window of a defined size across the dataset, where each sequence's target was the subsequent value in the time series. This approach allowed the LSTM model to learn from a fixed-length history of observations to predict the next value. Notably, the data underwent a log transformation during sequence creation to ensure numerical stability and reduce skewness, enhancing model performance by normalizing the distribution of values.

# EDA

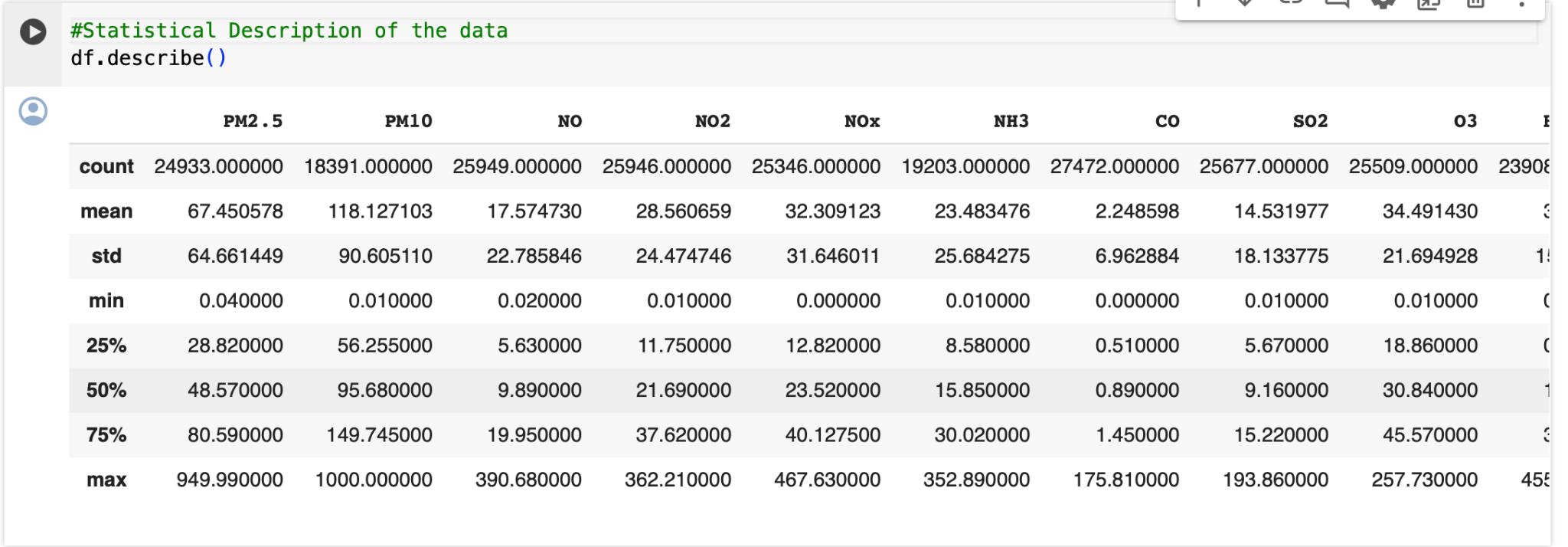
## General View of the Data

This dataset comprises air quality measurements from various cities, encompassing a total of 29,531 records across 16 columns. The columns include categorical data such as the city name, the date of the measurement, and the AQI (Air Quality Index) Bucket, alongside numerical data detailing concentrations of various pollutants (e.g., PM2.5, PM10, NO, NO2, NOx, NH3, CO, SO2, O3, Benzene, Toluene, Xylene) and the AQI value itself. While most pollutants are reported with varying degrees of completeness, certain pollutants like Xylene and NH3 have significantly fewer non-null values, indicating potential gaps in the data collection or recording process. The data type for pollutant concentrations is primarily float64, indicating numerical values with decimal precision, whereas the city, date, and AQI Bucket columns are stored as objects, likely strings, that provide categorical information. Importantly, the dataset is free of duplicate entries, suggesting a meticulous data collection or preprocessing effort to ensure that each record represents a unique measurement event.



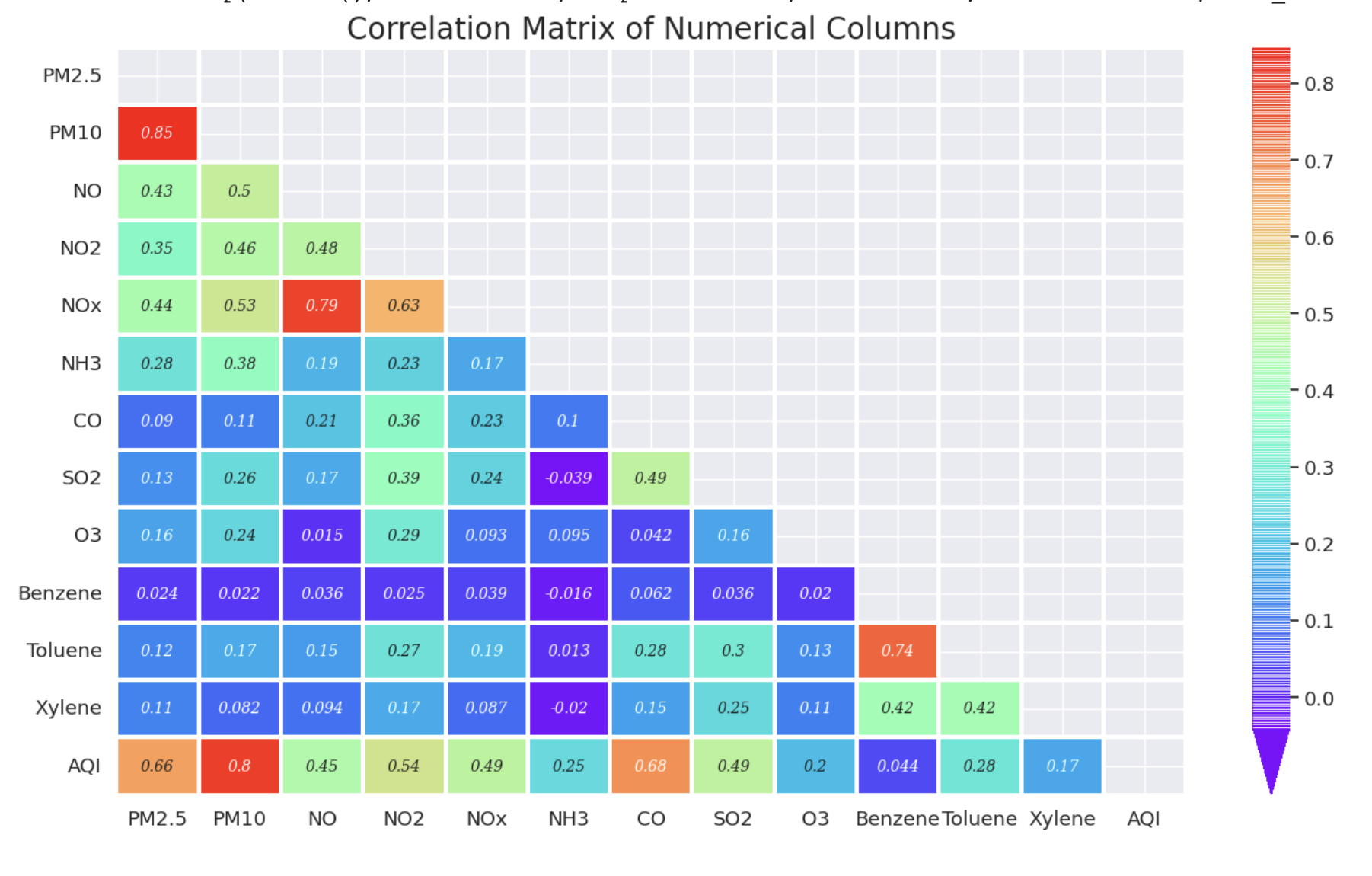
## Statistical View

The statistical analysis of the air quality data reveals a significant variability in pollutant concentrations, with standard deviations indicating wide variety across various locations. The range of values for pollutants such as PM2.5, PM10, and NO2 demonstrates episodes of extreme pollution, highlighting the presence of significant health risks during certain periods. This analysis also suggests a skewed distribution of data, where the median values are often lower than the mean, pointing towards a distribution influenced by a tail of high values. Moreover, the Air Quality Index (AQI) data, spanning from minimal to hazardous levels, underscores the diverse environmental conditions captured in the dataset, from clean to heavily polluted air.



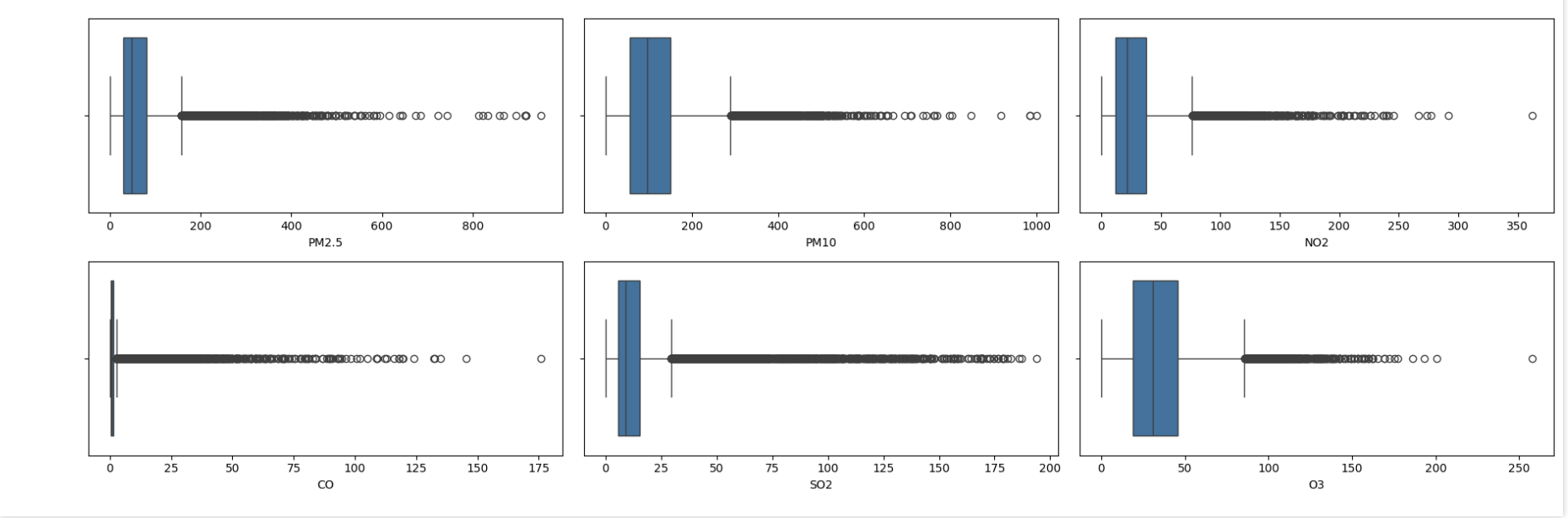
## Correlations

The correlation matrix provides insight into the interdependencies among air pollutants and their collective impact on the Air Quality Index (AQI). There's a notably strong positive correlation between PM2.5 and PM10 (0.85), indicative of their common sources and their combined influence on particulate pollution. Similarly, NOx exhibits strong correlations with NO (0.79) and NO2 (0.63), aligning with the fact that NOx is a combination of these two gasses. The AQI shows a pronounced positive correlation with PM2.5 (0.66), and to a lesser extent with PM10, NO, NO2, and NOx, highlighting these pollutants as significant contributors to overall air quality degradation. Interestingly, Toluene and Xylene also have a strong correlation (0.74), suggesting related emission sources or behaviors. Weaker correlations among other pollutants and the AQI imply that their contributions to air quality are less consistent or more complexly influenced by a variety of factors.



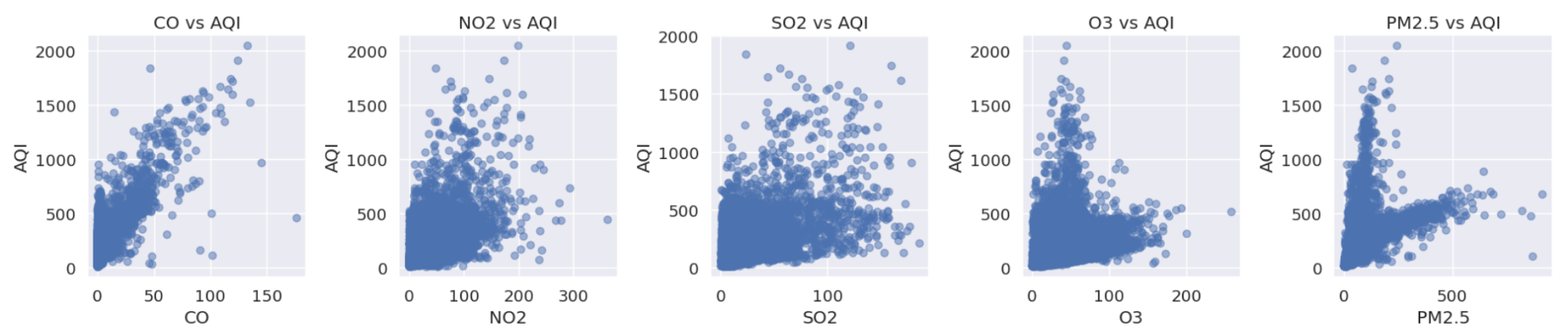
## Box-Plot

The box plots for PM2.5, PM10, NO2, CO, SO2, and O3 visually summarize the distributions of these air pollutants, highlighting their median values, interquartile ranges, and the presence of outliers. The medians are closer to the lower end of the boxes, indicating a right-skewed distribution for each pollutant, with a concentration of lower values and a tail of higher ones. Notably, the plots show a substantial number of outliers for PM2.5, PM10, and NO2, with these outliers stretching significantly far from the upper whiskers, suggesting frequent extreme pollution events. CO and SO2 also exhibit outliers, yet they are less extreme, indicating occasional but less severe spikes in concentrations. O3 has the fewest outliers, pointing towards more stable levels with infrequent high-value deviations.



## Scatter Plot

The scatter plots examining the relationship between CO, NO2, SO2, O3, PM2.5 levels, and the Air Quality Index (AQI) suggest a general positive correlation between individual pollutant concentrations and the AQI. The CO vs AQI and NO2 vs AQI plots show a broad dispersion of data points, implying a positive but non-linear relationship, with a diverse range of AQI values at higher pollutant levels. SO2 shows a similar spread, indicating that while there is a correlation, it is influenced by other factors. The O3 plot reveals a more complex and less clear correlation, with wide variations that suggest additional variables are at play. The PM2.5 vs AQI plot displays the most pronounced positive correlation, indicating that PM2.5 has a significant impact on AQI, particularly at higher concentrations where the spread of AQI values becomes larger, signifying the increasing severity of air quality as PM2.5 levels rise. Overall, these visualizations underscore the multifaceted nature of air quality determination, where individual pollutants contribute to varying extents and are likely influenced by a complex interplay of additional environmental factors.



# Challenges and Solutions

The primary challenge we encountered during the data collection and data cleaning phase was managing the numerous outliers present in our dataset. These outliers posed a significant obstacle because they could potentially skew our analysis and lead to inaccurate predictions if not addressed properly. Determining the most effective method for dealing with these anomalies was particularly challenging, as the chosen solution needed to mitigate their impact without compromising the integrity or the representativeness of the data. To solve this problem, we experimented with various methods, closely observing their effects on our data. This included applying different types of transformations to normalize the data distribution and selectively dropping data that fell outside the interquartile range (IQR). After careful consideration and analysis, we opted for the latter approach—removing outliers based on the IQR method. We chose this solution because it directly targeted and eliminated extreme values that could significantly distort our dataset’s overall analysis. By doing so, we maintained a balance between cleaning the data and preserving its essential characteristics, ensuring our dataset remained representative of the broader population.

# Feature Engineering

In our project, we've decided not to apply feature engineering, guided by the high quality of our pollutant data (PM2.5, NO2, CO, SO2, O3) and the straightforward nature of our task. Our dataset is directly relevant and sufficiently detailed for predicting AQI, indicating that additional data manipulation might not significantly enhance model performance. This decision allows us to expedite the modeling process and establish a baseline for future improvements, ensuring a focused and efficient approach to developing our AQI prediction model.

# Methodology

## Model Building

**Random Forest**

We deployed the Random Forest Regressor (RFR) to leverage its ensemble learning capability, which integrates multiple decision trees to improve prediction accuracy and generalizability while minimizing overfitting. The RFR minimizes the risk of overfitting while enhancing the model's ability to capture complex, non-linear relationships. A specific random state was utilized to ensure the consistency and reproducibility of the model's performance across different runs. The Random Forest model was carefully configured to balance complexity with predictive performance, preparing it for fitting to the preprocessed dataset.

**Support Vector Regressor**

Similarly, we chose the Support Vector Regressor (SVR) for its proficiency in handling both linear and non-linear data, making it an excellent choice for our regression tasks. The SVR was integrated into a pipeline that was considered to optimize the model's performance on our dataset. This approach allowed us to take full advantage of SVR's capabilities, particularly its flexibility in modeling complex relationships in high-dimensional spaces, by preparing the data in a manner that enhances the model's effectiveness.

**Long Short-Term Memory**

In building the LSTM model, we prioritized capturing temporal dependencies in the time-series data, which is crucial for accurate forecasting. The LSTM architecture, known for its ability to retain information over long sequences, was structured with an initial LSTM layer, configured to process input sequences and a Dense layer to output continuous values for regression. The model's compilation with the Adam optimizer and mean squared error loss function reflects our goal to minimize predictive errors effectively.

## Hyperparameter Tuning

**Support Vector Regressor**

We refined our SVR mode model within a pipeline framework using grid search cross-validation. This process involved defining a range of values for key hyperparameters, specifically the regularization parameter (`C`), the kernel coefficient (`gamma`), and the epsilon margin (`epsilon`), to find the optimal configuration for our regression model. We employed a 10-fold cross-validation strategy with shuffling to ensure a thorough and unbiased evaluation of each parameter combination, facilitated by the `GridSearchCV` method, which automates the search across the specified parameter grid and evaluates the combinations based on the negative mean squared error (MSE). The result was the identification of the best hyperparameter set, leading to an optimized SVR model that was then assessed on a separate test dataset to evaluate its performance in terms of MAE, MSE, and R² metrics. This rigorous tuning process aimed to enhance the model's predictive accuracy and generalizability to new data.

**Long Short-Term Memory**

We refined our LSTM model for time-series forecasting by strategically experimenting with window size, a key hyperparameter that defines the input sequence length. This parameter crucially influences the volume of historical data utilized for predictions. Our adaptive tuning involved testing window sizes of 3, 5, 7, and 10, which allowed us to calibrate the model's access to past information effectively. The model was performing better as we decreased the window size. Through assessing the model's performance with each window size, we embarked on a targeted hyperparameter tuning exercise, pivotal for models processing sequential data. This methodology enabled us to pinpoint the window size that strikes an ideal balance—sufficiently incorporating historical data for robust forecasting while ensuring the model's complexity and computational demands remain manageable.

## Model Performance

The evaluation of our models' performance was carried out using selected regression analysis metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), and R^2 score. We experimented with three preprocessing techniques on our data before training the models. Initially, models were trained on raw data without any transformation or scaling. Subsequently, we applied standard scaling to the data for the second set of models. Finally, the data underwent log transformation for the third set of training. Our objective was to identify which preprocessing method enhances our models' performance most significantly. The initial results indicated decent performance without scaling or transformation, though not the best. With standard scaling, the models showed a significant improvement. However, it was the log transformation that yielded the best performance across all models. Moreover, although the deep learning model yielded satisfactory results, it underperformed compared to the other two models, though the difference was small.

|  |  |  |  |
| --- | --- | --- | --- |
| **Without Transformation & Scaling** | **MAE** | **MSE** | **R^2** |
| RF | 18.78 | 1020.37 | 0.84 |
| SVR | 18.7 | 1104.96 | 0.81 |
| LSTM | 22.24 | 1121.2 | 0.81 |
| **With Scaling** | | | |
| RF | 0.24 | 0.16 | 0.84 |
| SVR | 0.24 | 0.17 | 0.81 |
| LSTM | 0.34 | 0.27 | 0.72 |
| **With Transformation** | | | |
| RF | 0.15 | 0.05 | 0.82 |
| SVR | 0.16 | 0.05 | 0.79 |
| LSTM | 0.18 | 0.06 | 0.77 |

## Model Documentation

The project was executed using two separate Google Colab notebooks to maintain clarity and structure. The [first](https://colab.research.google.com/drive/15sNm4igHc-bCM2SHDMMhxe3KBWWZeyhb?usp=sharing) notebook focused on the machine learning models, namely Random Forest (RF) and Support Vector Regression (SVR), while the [second](https://colab.research.google.com/drive/12t9CAc3NJZZmjPE5BHJf5SddVsVD0F0P?usp=sharing) notebook was dedicated to the deep learning model as it requires more complex implementation, specifically Long Short-Term Memory (LSTM).

# Discussion & Limitations

In this study, we employed various models, training each with three distinct preprocessing methods. Our primary aim was to refine the project's outcomes by examining the impact and effectiveness of each preprocessing technique. The Random Forest (RF) model exhibited strong initial performance, leading us to focus solely on the impact of different preprocessing methods without delving into hyperparameter optimization for this model. Conversely, for the Support Vector Regression (SVR) model, after conducting hyperparameter tuning to enhance its performance, we applied the same trio of preprocessing approaches as with the RF model. The results from the SVR were comparable to those of the RF.

For the deep learning model, Long Short-Term Memory (LSTM), data scaling slightly improved its performance, but data transformation offered a more significant enhancement. The LSTM model demanded more complex preprocessing, particularly in creating sequences and determining the optimal window size, which proved to be a substantial challenge. Through numerous tests, we discovered that a smaller window size generally led to better performance for the LSTM model.

One of the primary limitations encountered was the difficulty of fine-tuning the deep learning model, largely due to its computational demands. This aspect was especially challenging because deep learning models, like LSTM, require extensive computational resources for training and optimization. Such demands can limit the ability to experiment with various hyperparameters or larger datasets, potentially restricting the model's performance and the scope of experimentation. Additionally, the need for meticulous preprocessing, including the selection of an appropriate window size for the LSTM model, highlights the complexity of preparing data for deep learning and its critical impact on model outcomes. This complexity underscores the balance between model performance and computational feasibility, a key consideration in deep learning projects.

# Deployment Readiness Checklist

Looking back at the performance of each model, RF for now seems to be the most promising model so we will choose it as the final model for this particular paper with a future direction to improve on the LSTM model alongside the SVR.

To evaluate the readiness of the final model for deployment, we must assess the following:

**1- Scalability:**

**Data volume:** The final model must be fed with a greater amount of data to further test its performance and the behavior of the underlying algorithm in terms of computation time and other factors.

**Computational resources:** The final model must be scalable across different hardware configurations such as increasing the number of CPUs or leveraging distributed computing for parallel processing.

**Model training:** The model should be retrained with new data and assessed based on the ease of retraining it as well as seeing how it can efficiently adapt to changes.

**2- Reliability:**

**Error handling:** Certain mechanisms need to be implemented in order to handle errors gracefully once the model is deployed, such as robust error logging and alerting systems.

**Testing and validation:** A thorough testing across different scenarios and edge cases must be conducted to ensure the model behaves predictably and reliably in real-world situations.

**Monitoring:** The performance of the model must be continuously monitored to detect any deviations or degradation in performance.

**3- Integration:**

**API design:** A well-defined and documented API must be designed for interacting with the model to make it easier to integrate into systems or applications.

**Compatibility:** It must be ensured when the model is integrated into a system or an application that it is compatible with other components such as the database, web server, or cloud platforms.

**Security:** Security measures must be taken into account at all times to protect both the model and its data from different complications such as integrity, confidentiality, and authenticity.

In order for the model to be deployed, these factors must qualify and be taken into account to ensure a reliable model which can be part of expert systems and benefit the user as per the results it promised.

# Conclusion & Future Direction

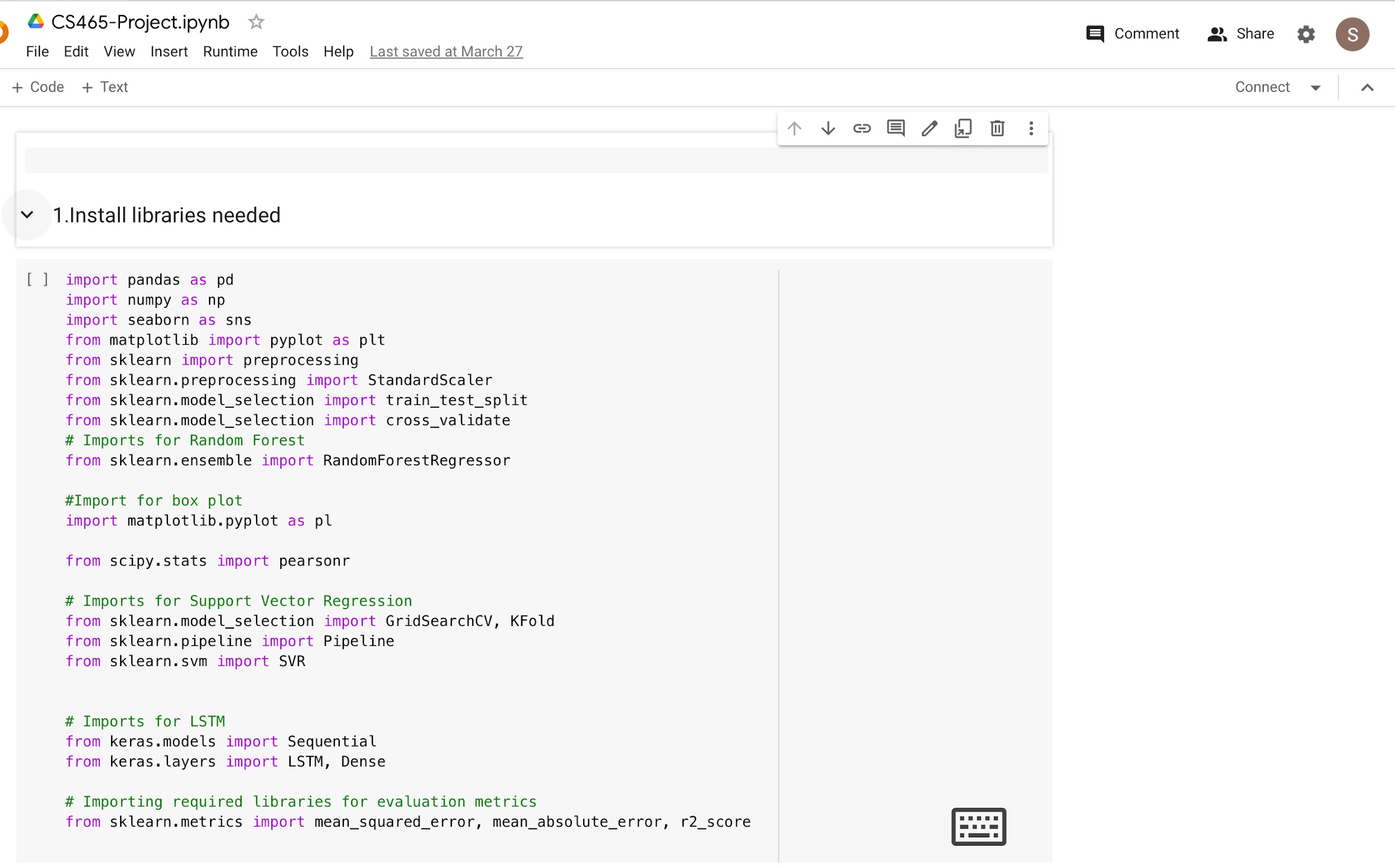
This project's exploration into different machine learning and deep learning models, alongside a variety of preprocessing techniques, has yielded valuable insights into optimizing model performance. The Random Forest model demonstrated robust performance from the outset, highlighting its effectiveness with minimal preprocessing. Meanwhile, the Support Vector Regression model, after hyperparameter adjustments, achieved results on par with the Random Forest, indicating its potential with fine-tuning. The Long Short-Term Memory model, despite requiring more intricate preprocessing, showed notable improvements with data transformation, emphasizing the importance of tailored data preparation in deep learning.

Our findings underscore the significance of preprocessing and model-specific optimizations in achieving optimal results. The comparative analysis also revealed the nuanced balance between model complexity, computational demand, and performance efficiency.

For deep learning models, particularly LSTM, future work should prioritize exploring advanced hyperparameter tuning techniques. Fine-tuning these models holds the promise of unlocking significant performance improvements, making it a vital step for enhancing accuracy and efficiency in complex data analysis tasks.

# Code & Tools

In this project that is conducted on Google Colab, we utilized the Python programming language, leveraging key libraries including pandas for data manipulation, NumPy for numerical computations, seaborn and matplotlib for data visualization, and scikit-learn for preprocessing.



# References

1. Méndez, M., Merayo, M. G., & Núñez, M. (2023, February 16). Machine learning algorithms to forecast air quality: a survey. *Artificial Intelligence Review*, *56*(9), 10031–10066. <https://doi.org/10.1007/s10462-023-10424-4>
2. Goh, C. C., Kamarudin, L. M., Zakaria, A., Nishizaki, H., Ramli, N., Mao, X., Syed Zakaria, S. M. M., Kanagaraj, E., Abdull Sukor, A. S., & Elham, M. F. (2021, July 21). Real-Time In-Vehicle Air Quality Monitoring System Using Machine Learning Prediction Algorithm. *Sensors*, *21*(15), 4956. <https://doi.org/10.3390/s21154956>
3. Zhang, D., & Woo, S. S. (2020). Real Time Localized Air Quality Monitoring and Prediction Through Mobile and Fixed IoT Sensing Network. *IEEE Access*, *8*, 89584–89594. <https://doi.org/10.1109/access.2020.2993547>
4. Wang, G., Yu, C., Guo, K., Guo, H., & Wang, Y. (2024, January 15). Research of low-cost air quality monitoring models with different machine learning algorithms. *Atmospheric Measurement Techniques*, *17*(1), 181–196. <https://doi.org/10.5194/amt-17-181-2024>
5. *Real-Time Machine Learning for Air Quality and Environmental Noise Detection*. (2020, December 10). IEEE Conference Publication | IEEE Xplore. <https://ieeexplore.ieee.org/abstract/document/9377939>
6. Rodríguez-Pérez, R., & Bajorath, J. (2022). Evolution of Support Vector Machine and Regression Modeling in Chemoinformatics and Drug Discovery. *Journal of computer-aided molecular design*, *36*(5), 355–362. <https://doi.org/10.1007/s10822-022-00442-9>
7. Schonlau, M., & Zou, R. Y. (2020). The random forest algorithm for statistical learning. *The Stata Journal, 20*(1), 3-29.<https://doi.org/10.1177/1536867X20909688>
8. Ji, H., Lou, Y., Cheng, S., Xie, Z., & Zhu, L. (2022). An Advanced Long Short-Term Memory (LSTM) Neural Network Method for Predicting Rate of Penetration (ROP). *ACS Omega, 8*(1), 934–945.<https://doi.org/10.1021/acsomega.2c06308>
9. *Air quality data in India (2015 - 2020)*. (2020, July 28). Kaggle. <https://www.kaggle.com/datasets/rohanrao/air-quality-data-in-india>

# 

# 

# 

# 

# 

# 

# 