Major Project Report

on

Early Warning Prediction of Environmental Parameters

Submitted by

E.Vaishnavi –20bcs044

Golla Anjaiah –20bcs048

Rishabh Gautam –20bcs112

P.Vishnu Kumar –20bcs138

Under the guidance of

Dr. Prakash Pawar

Assistant Professor, Electronics & Communication Engineering

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY DHARWAD

CERTIFICATE

It is certified that the work contained in the project report titled "Early Warning Prediction of Environmental Parameters," by "E.Vaishnavi (Roll No: 20bcs044)", "Golla Anjaiah (Roll No:20bcs048)", "Rishabh Gautam (Roll No: 20bcs112)" and "P.Vishnu Kumar (Roll No: 20bcs138)" has been carried out under my/our supervision and this work has not been submitted elsewhere for a degree.

Signature of Supervisor(s)

Name(s)

Department(s)

(Month, Year)

Contents

1 INTRODUCTION

Underground mining operations present unique challenges, particularly concerning the maintenance of safe and healthy working environments. Environmental parameters such as CO_2, CO, NO, temperature, and humidity play crucial roles in ensuring the well-being of workers and the efficiency of operations within these confined spaces. Timely monitoring and prediction of these parameters are essential for early detection of potential hazards and implementation of preventive measures.

In this project, we embark on a comprehensive analysis of environmental conditions in underground mines, focusing on time series forecasting and analysis techniques. Our goal is to leverage advanced algorithms such as ARIMA, SARIMA, and LSTM to predict future values of environmental parameters and gain insights into temporal patterns and trends. By doing so, we aim to provide actionable insights for maintaining optimal environmental conditions and mitigating potential risks.

Through this analysis, we seek to address key questions such as the identification of environmental conditions with respect to time, daily periods when conditions are categorized as Good, Satisfactory, Poor, Very poor, or Severe, and the parameters that require immediate attention for control or mitigation. Ultimately, our endeavor is to contribute to the enhancement of safety, productivity, and sustainability in underground mining operations.

2 OBJECTIVE

Our objective is to perform time series forecasting and analysis on environmental parameters including CO_2 , CO, NO, temperature, and humidity data collected from underground mines. We aim to predict future values of these parameters using time series algorithms such as ARIMA, SARIMA, and LSTM. Additionally, we will compare the forecasting results of ARIMA and SARIMA models. Furthermore, we will analyze the environmental conditions over time and identify daily time periods when the environmental condition is categorized as Good, Satisfactory, Poor, Very poor, or Severe. Finally, we will determine which parameters need to be controlled or mitigated to maintain optimal environmental conditions within the underground mines.

3 DATASET

The dataset contains DateTime and environmental parameters such as CO_2, CO, NO, Temperature and Humidity. The Dataset contains 40 days environmental parameters data collected from sensors. article gensymb

- CO_2 normal value is 400 ppm to 500 ppm
- CO normal value is 0 ppm to 15 ppm
- NO normal value is 0 ppm to 10 ppm
- Temp and hum values are less than 28 to 30C and 59.5% to 62%

CO_2 (Carbon Dioxide) is a greenhouse gas that traps heat in the atmosphere. The maximum CO_2 concentration recorded was 736 ppm, the minimum was

	DateTime	CO2	NO	temp	humi
0	12/31/2023 0:00	544	0	29.8	61.4
1	12/31/2023 0:05	546	0	29.8	61.4
2	12/31/2023 0:10	549	0	29.8	61.4
3	12/31/2023 0:15	546	0	29.6	61.4
4	12/31/2023 0:20	548	0	29.6	61.4
...	...	...	...	...	...
9211	1/31/2024 23:35	550	0	29.2	59.9
9212	1/31/2024 23:40	549	0	29.2	59.9
9213	1/31/2024 23:45	547	0	29.2	59.9
9214	1/31/2024 23:50	549	0	29.2	59.9
9215	1/31/2024 23:55	556	0	29.2	59.9

9216 rows × 5 columns

Figure 1. Dataset Details

PARAMETERS	THRESHOLD
CO2	550
CO	18
NO	13
TEMPERATURE	31
HUMIDITY	58

Figure 2. threshold limit values of parameters in underground mine.

PARAMETERS	MAXIMUM	MINIMUM	AVERAGE
CO2	736	522	573.23
CO	32	0	0.256
NO	35	0	0.529
Temperature	33	28.8	30.07
Humidity	61.4	55.3	58.4

Figure 3. minimum, maximum values, average values of each parameters

522 ppm, and the average was 573.23 ppm. **CO (Carbon Monoxide)** is a gas that can be harmful to human health at high concentrations. The maximum CO concentration recorded was 32 ppm, the minimum was 0 ppm, and the average was 0.256 ppm. **NO (Nitrogen Oxide)** gas pollutant resulting from combustion processes. The maximum NO concentration recorded was 35 ppm, the minimum was 0 ppm, and the average was 0.529 ppm. In Temperature, the maximum temperature recorded was 33 °C, the minimum was 28.8 °C, and the average was 30.07 °C. Humidity refers to the amount of moisture in the air. The maximum humidity recorded was 61.4%, the minimum was 55.3%, and the average was 58.4%.

4 TASKS

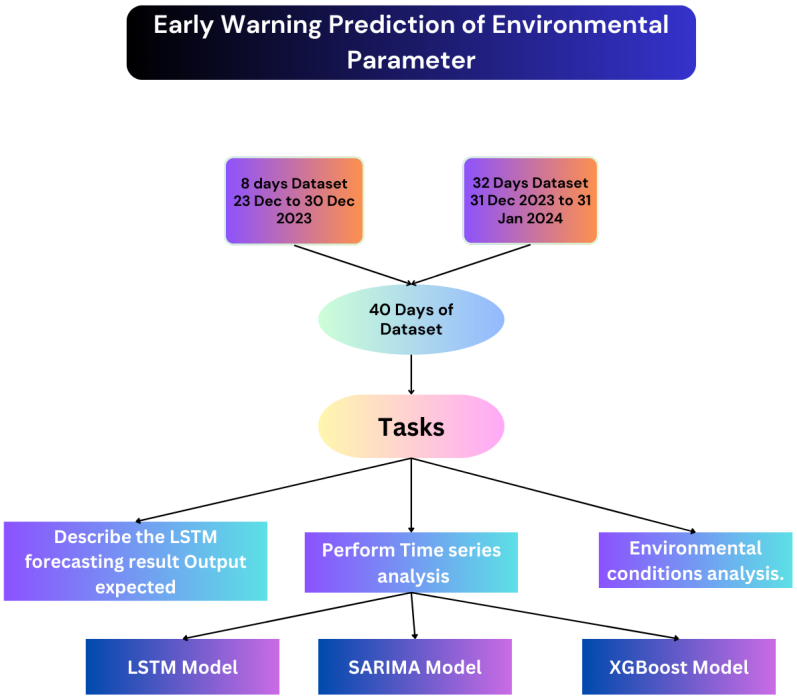


Figure 4. describes the tasks flowchart

4.1 Perform time series forecasting using Time series algorithms

- Perform Time series analysis such ARIMA, SARIMA, LSTM, and Compare results of ARIMA and SARIMA.

4.2 Describe the LSTM forecasting result Output expected.

- Based on the 40 days data, forecast the next 1 or 2 days' data similar to shown in graph and Its applicable to other parameters CO, NO and temperature and humidity also.

4.3 Environmental conditions analysis.

- Based on the parameters – CO_2, CO, NO, Temp and humidity Find out the Environmental condition with respect to Time Daily at what time the environmental condition is Good, Satisfactory, Poor, Very poor, or Severe, and which parameter need to control or mitigate

Flowcharts of Time Series Forecasting using time series algorithm and Environmental Conditions Analysis

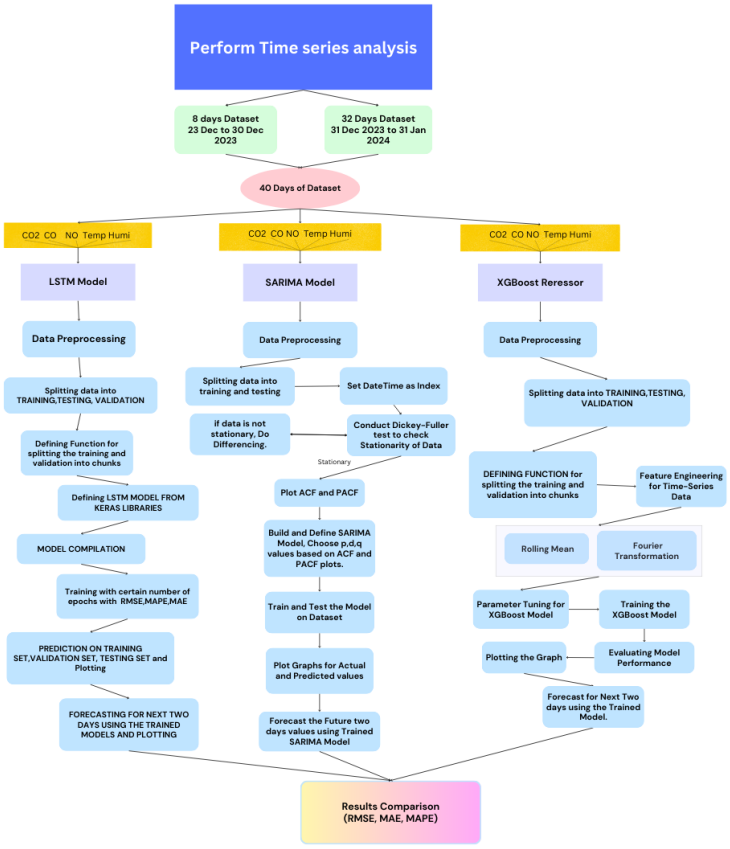


Figure 5. Describes the Time Series Forecasting using time series algorithm

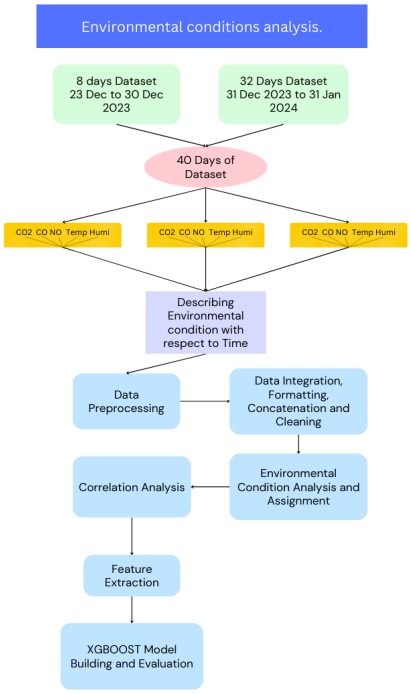


Figure 6. Describes the Prediction of Environmental Conditions Analysis

5 TIME SERIES FORECASTING ANALYSIS.

5.1 LSTM TIME SERIES FORECASTING

5.1.1 LSTM MODEL

Long Short-Term Memory (LSTM) models have garnered significant attention in the field of time series analysis for their adeptness in capturing intricate temporal dependencies. These models offer a sophisticated framework for understanding and forecasting sequential data, making them invaluable tools across various domains. At the heart of the univariate LSTM architecture lie memory cells, each equipped with hidden states and cell states. The hidden state serves as the network's memory, retaining crucial information from past time steps, while the cell state facilitates selective retention or forgetting of data, enabling the model to grasp long-term dependencies effectively.

Integral to the functionality of univariate LSTM models are gates, including the input, forget, and output gates. These gates regulate the flow of information within the model, controlling the influx of new data, determining what information to discard, and managing the output of the model's predictions. Leveraging activation functions such as the hyperbolic tangent (tanh), LSTM cells transform inputs and gate outputs, enabling the model to capture nonlinear relationships inherent in time series data. During the training phase, LSTM models update their parameters through backpropagation and gradient descent optimization, striving to minimize the disparity between predicted and actual values in the time series. Once trained, these models are capable of generating accurate forecasts for future time steps, making them indispensable tools for predictive analytics in diverse fields.

In conclusion, univariate LSTM models represent a sophisticated approach to time series analysis, leveraging their ability to capture complex temporal dependencies. By understanding the internal mechanisms of these models, practitioners can harness their power to make accurate forecasts and gain insights into sequential data patterns. With ongoing advancements in deep learning and neural network research, univariate LSTM models continue to play a pivotal role in addressing challenging time series forecasting tasks across various industries.

Data Preprocessing:

Here we have datasets for December and January month of CO_2, CO, NO, Temperature, Humidity. The datasets are concatenated for the further preprocessing. Then the combined data is sorted using python pandas library utilities and the duplicated data points also removed using pandas and the following graphs are plotted by changing the index of the combined dataset to the Datetime column of the dataset itself for the clear visualization of the data.

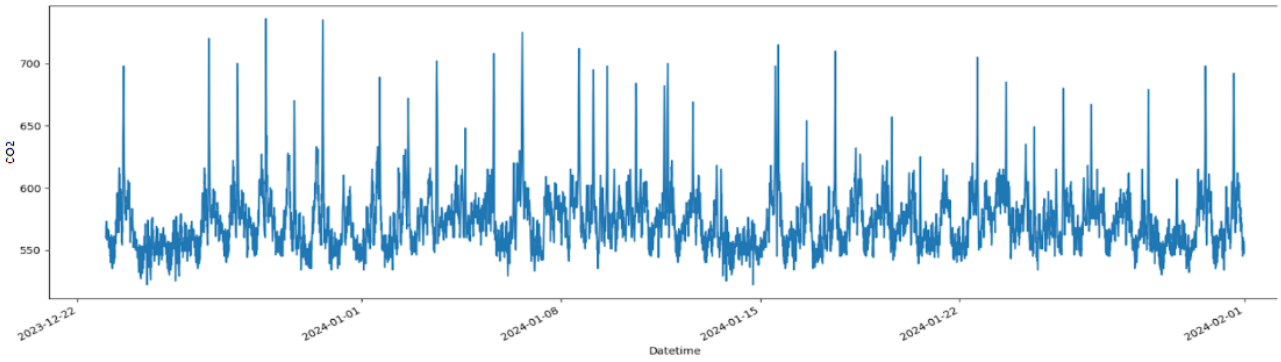


Figure 7. Visualization of the data CO_2

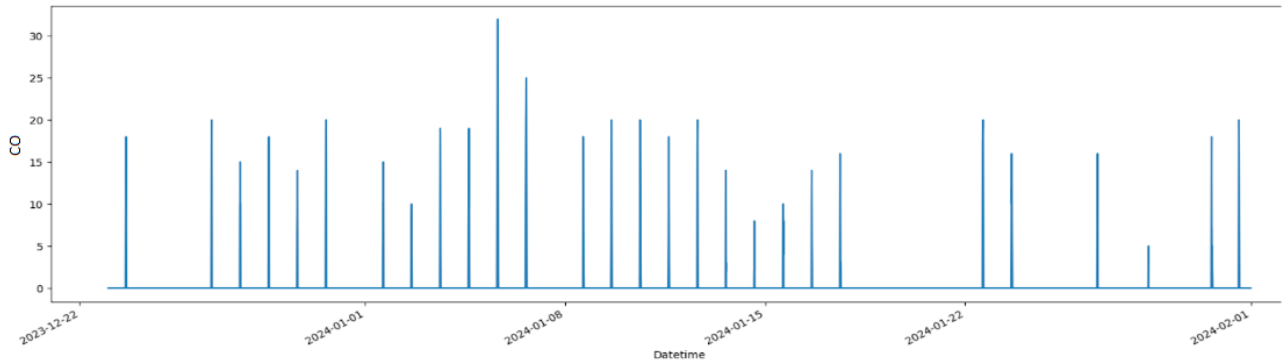


Figure 8. Visualization of the data CO

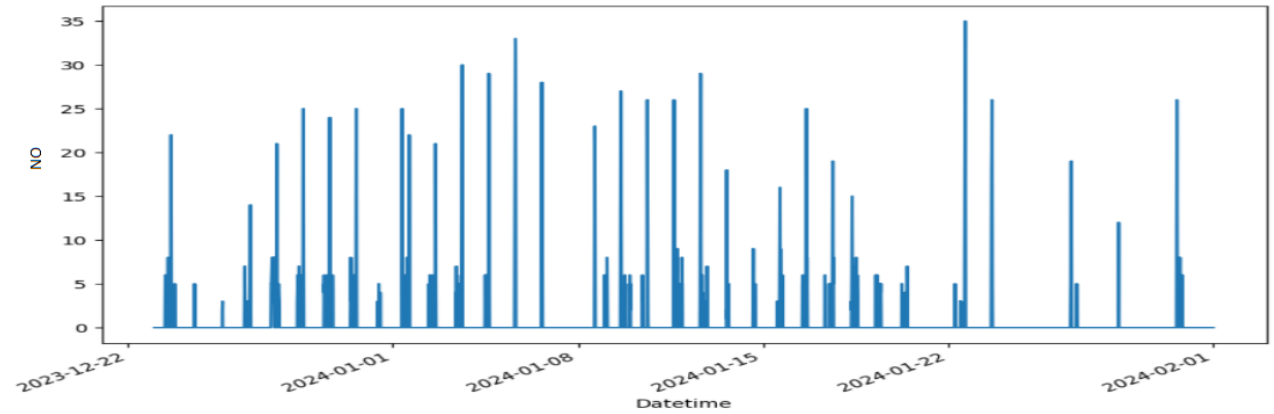


Figure 9. Visualization of the data NO

Data distribution for X and y variables using defined function:

About the function: The function is designed to facilitate the preparation of data for training LSTM (Long Short-Term Memory) models, particularly for time series analysis. It operates on a pandas DataFrame containing sequential data points, such as a time series. The function iterates over the DataFrame and creates input-output pairs, essential for training the LSTM model.

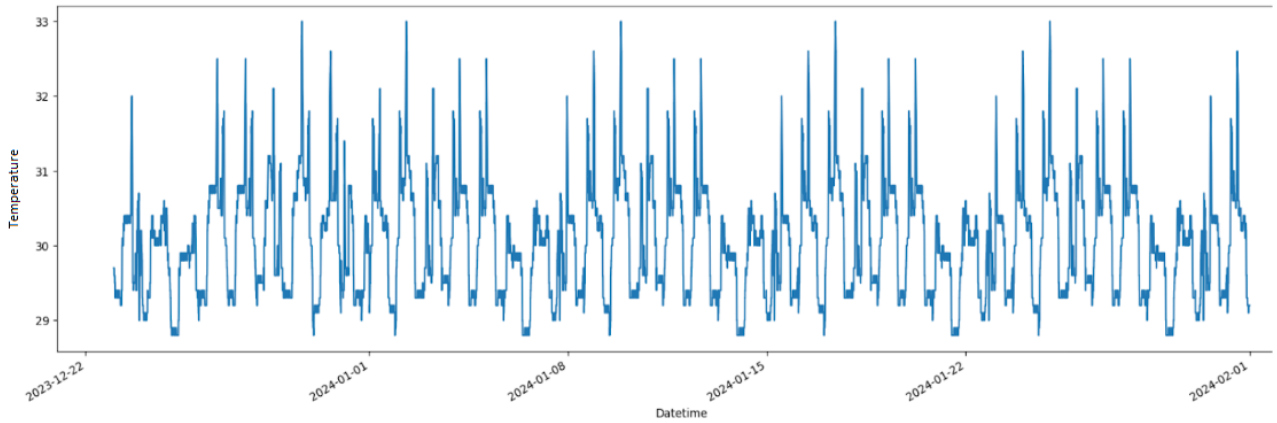


Figure 10. Visualization of the data Temperature

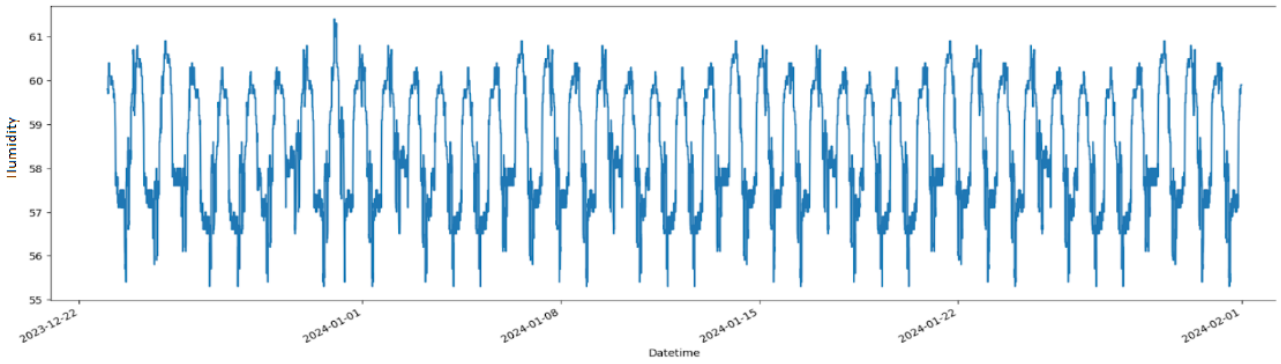


Figure 11. Visualization of the data Humidity

To achieve this, the function employs a sliding window approach. It starts by selecting a window of a specified size (defaulted to 5) from the beginning of the time series. This window represents the historical context or input sequence for a single training instance. The function then moves this window one step forward at a time, iteratively traversing the DataFrame.

For each window position, the function extracts the data points within the window and organizes them into a nested list structure. Each nested list corresponds to an individual input sequence, where each element within the nested list represents a single data point. Simultaneously, the function captures the next data point immediately following the window as the output value or label for that particular input sequence. These input-output pairs are crucial for training the LSTM model to predict future data points based on historical patterns.

Upon processing the entire DataFrame, the function returns two NumPy arrays: one containing the input sequences (X) and another containing the corresponding output values (y). These arrays can then be utilized directly in the training process of an LSTM model, providing the necessary input data and target labels for supervised learning. By encapsulating this data preparation process into a function, users can easily transform their time series data into a format suitable for LSTM model training, streamlining the overall workflow of time series analysis and prediction tasks.

Splitting Dataset for Training, Validation and Testing:

We will get X1 and y1 dataframes after using the defined function for data distribution, then the DataFrame X1 is divided into three subsets: training, validation, and testing. These subsets are crucial for training, tuning, and evaluating machine learning models, respectively.

Training Set (*X train1*, *y train1*): It contains the initial portion of the DataFrame X1, consisting of the first 5000 samples. *X train1* comprises the input features used for training the model. *y train1* contains the corresponding target labels or output values associated with the input features in *X train1*.

Validation Set (*X val1*, *y val1*): The validation set is utilized for fine-tuning the model's hyperparameters and assessing its performance during training. It consists of samples from the DataFrame X1 ranging from the 5000th to the 8999th index, totaling 4000 samples. Similar to the training set, *X val1* comprises input features, while *y val1* contains the corresponding target labels.

Testing Set (*X test1*, *y test1*):

The testing set serves as an independent dataset to evaluate the trained model's performance and generalization ability It includes the remaining samples of the DataFrame X1 starting from the 9000th index until the end. *X test1* consists of input features, while *y test1* holds the corresponding target labels. By partitioning the DataFrame X1 into these distinct subsets, practitioners can effectively train, validate, and test machine learning models, ensuring robustness and reliability in their predictive capabilities. This division enables comprehensive model assessment, aiding in the identification of potential overfitting or underfitting issues and facilitating the selection of optimal model configurations.

Use of Splitting dataset into training,validation and testing:

The division of a dataset into training, validation, and testing sets serves several crucial purposes in machine learning:

Training Set:

The training set is used to train the machine learning model's parameters. During training, the model learns the underlying patterns and relationships in the data by adjusting its parameters to minimize a predefined loss function. A larger training set typically leads to a more robust and accurate model, as it provides more diverse examples for learning.

Validation Set:

The validation set is used to fine-tune the model's hyperparameters and evaluate its performance during training. After each training epoch or batch, the model's performance is evaluated on the validation set. This helps in monitoring the model's progress and detecting overfitting. By adjusting hyperparameters based on validation set performance, such as learning rate or regularization strength, practitioners can optimize the model's performance on unseen data.

Testing Set: The testing set is used to assess the final performance of the trained model. Once the model training and hyperparameter tuning are completed, the model's performance is evaluated on the testing set, which consists of unseen data that the model has not encountered during training. The testing set provides an unbiased estimate of the model's generalization performance, reflecting how well it can make predictions on new, unseen data from the same underlying distribution as the training data. By dividing the

dataset into training, validation, and testing sets, practitioners can effectively train and evaluate machine learning models while ensuring the model's robustness and generalization ability to unseen data. This division enables the identification of potential issues such as overfitting and facilitates the selection of optimal model configurations for deployment in real-world scenarios.

5.1.2 LSTM MODEL BUILT FOR TRAINING TIME SERIES DATA

The neural network model designed here specializes in predicting carbon dioxide (CO_2,CO,NO,Temperature,Humidity) levels by leveraging historical time series data. It initiates with an input layer structured to accommodate sequences of given data, each comprising five consecutive observations. This setup allows the model to ingest contextual information crucial for discerning patterns and trends in the given data over time. Following the input layer, an essential component of the model architecture, a Long Short-Term Memory (LSTM) layer, takes precedence. Renowned for its adeptness in capturing temporal dependencies, the LSTM layer processes the input sequences, effectively capturing the complex and dynamic nature of the data.

Subsequent to the LSTM layer, the model incorporates two dense layers to further refine the learned representations and facilitate accurate predictions. The first dense layer, boasting eight units and employing the Rectified Linear Unit (ReLU) activation function, introduces non-linearity to the model's transformations. This injection of non-linearity allows the model to capture intricate relationships within the CO_2 data, enhancing its capacity to discern subtle patterns and fluctuations. Lastly, the second dense layer, serving as the output layer, consists of a single unit utilizing a linear activation function. This layer is

pivotal in generating predictions for future CO_2 levels based on the processed representations learned by the model.

In essence, the architectural arrangement of the model, comprising input, LSTM, and dense layers, equips it with the capability to effectively learn and discern meaningful patterns within the input data. By leveraging historical time series data, the model can make accurate predictions of future levels, thereby offering valuable insights for environmental monitoring and climate research endeavors. This model's robustness and predictive prowess underscore its potential utility in addressing real-world challenges pertaining to variable level forecasting and mitigation strategies.

5.1.3 LSTM MODEL COMPILATION PROCESS:

A checkpoint mechanism is implemented to save the best-performing model during the training phase. This is achieved using the ModelCheckpoint callback, which is a feature provided by TensorFlow's Keras library. The checkpoint is configured to save only the best model encountered during training, based on a predefined criterion, typically the validation loss.

During model training, after each epoch, the performance of the model on the validation set is evaluated. If the current model performs better (i.e., achieves a lower validation loss) than any previous model encountered during training, the checkpoint mechanism saves the current model's weights and architecture to a specified directory. By setting SaveBestOnly is equal to True, the checkpoint ensures that only the best-performing model is saved, preventing overwriting of previous checkpoints with inferior models.

The model is then compiled using the specified loss function (MeanSquared-

Error()), optimizer (Adam with a learning rate of 0.0001), and evaluation metrics ('mae', 'mape', and 'RootMeanSquaredError()'). Subsequently, the model is trained using the fit method, with the training data (*Xtrain1, ytrain1*) and validation data (*Xval1*, *yval1*) provided. The training process is configured to run for a specified number of epochs (450 in this case) and is augmented with the cp1 callback, which triggers the checkpoint mechanism to save the best model encountered during training.

Overall, the checkpoint mechanism ensures that the best-performing model is saved during the training process, enabling practitioners to retain the model configuration that yields optimal performance on the validation set for subsequent evaluation or deployment.

Prediction for the CO_2 Training, Validation, Testing sets using trained LSTM Model respectively:

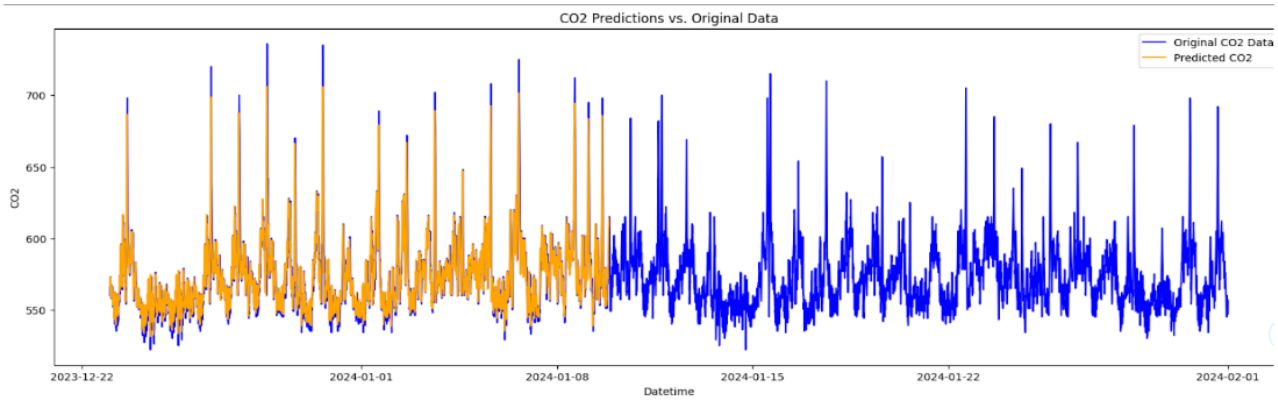


Figure 12. CO_2 prediction vs Original Data

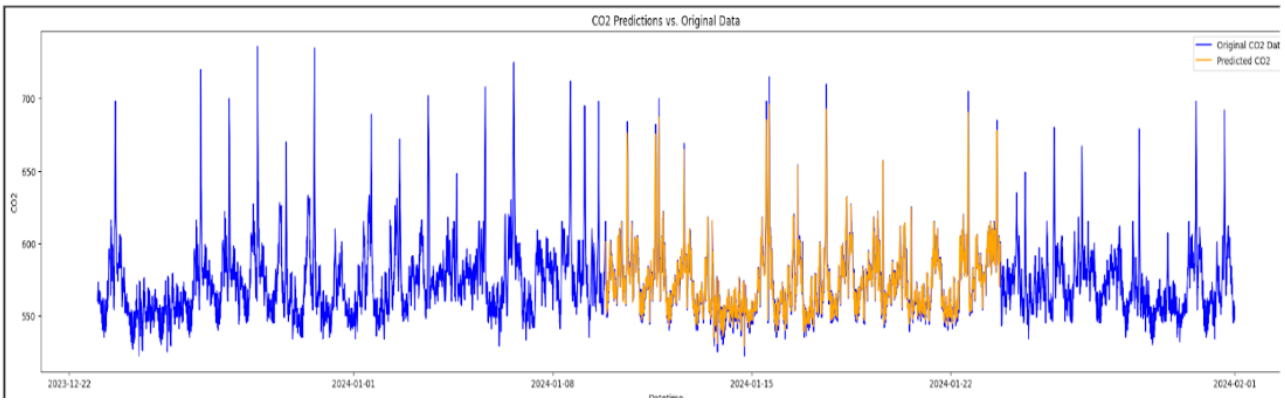


Figure 13. CO_2 prediction vs Original Data

FORECASTED DATA FOR CO_2 FOR THE NEXT TWO DAYS USING THE TRAINED LSTM MODEL represented using BLACK line:

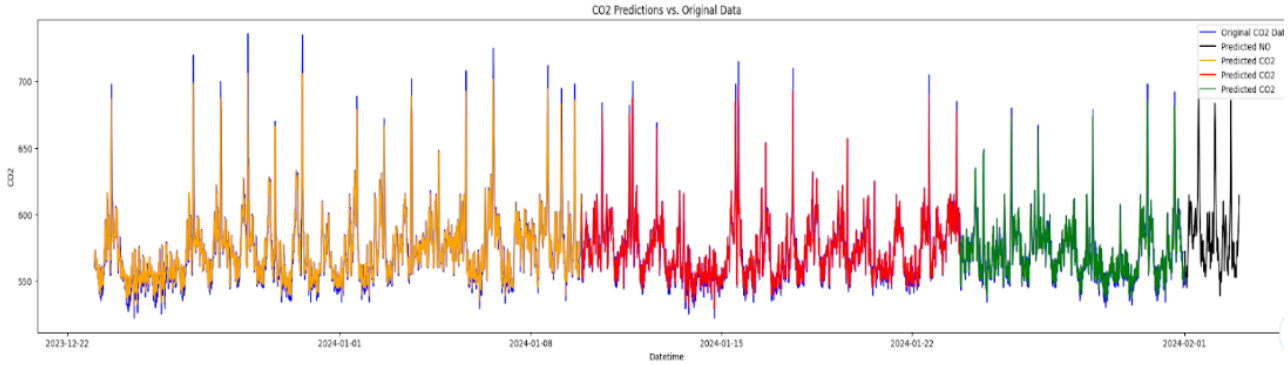


Figure 15. Forcasting data For CO_2 prediction vs original data

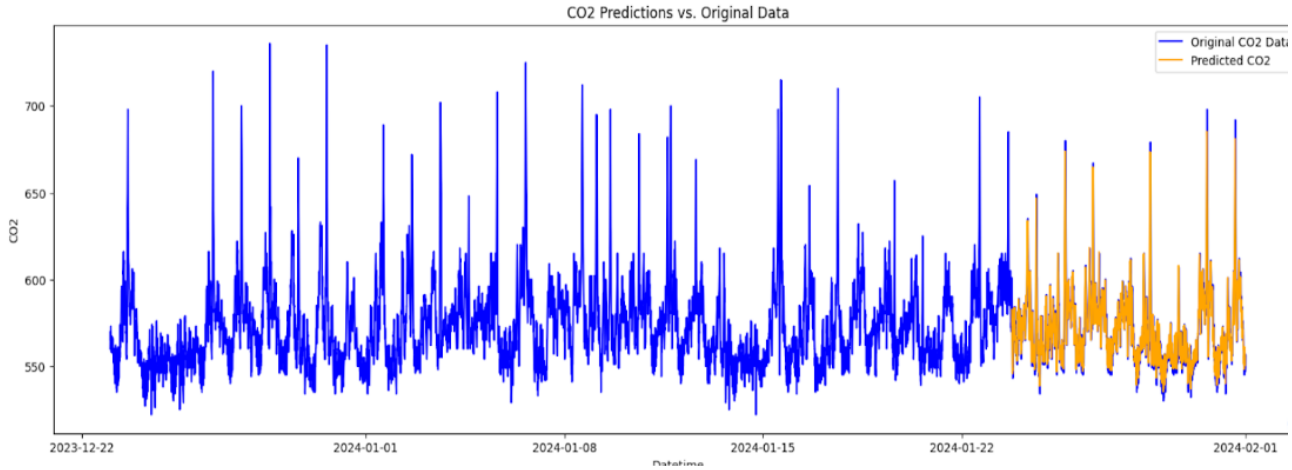


Figure 14. CO_2 prediction vs Original Data

PREDICTIONS FOR THE CO Training, Validation, Testing sets using trained LSTM Model respectively:

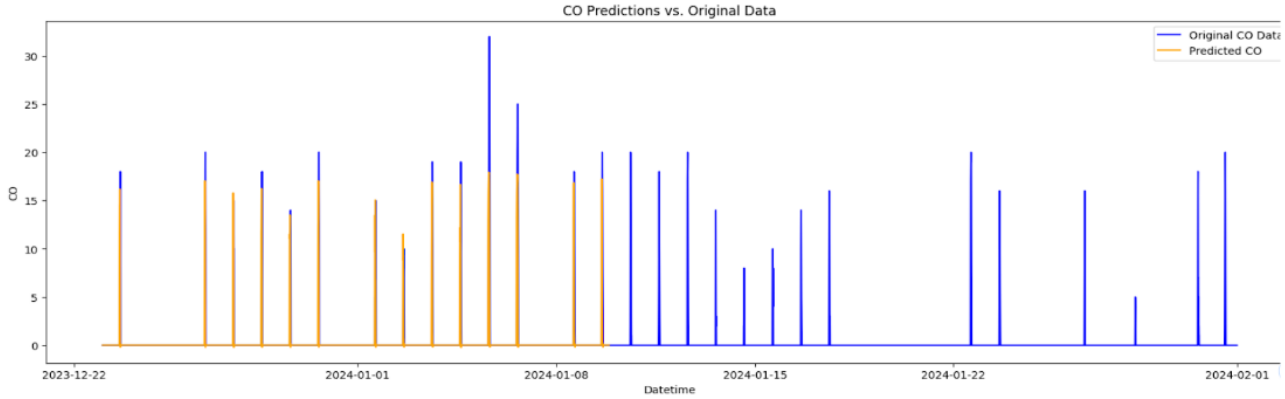


Figure 16. CO prediction vs Original Data

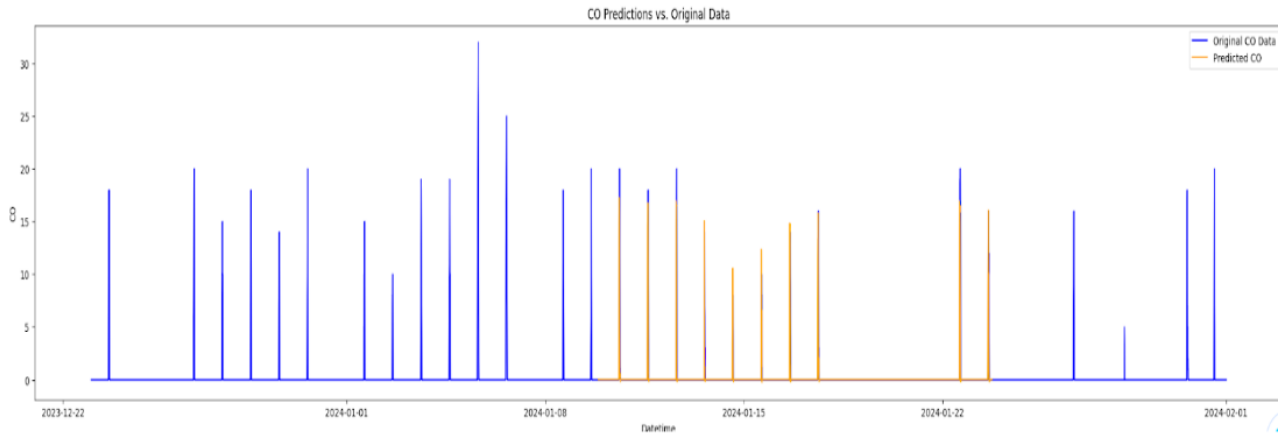


Figure 17. CO prediction vs Original Data

FORECASTED DATA FOR CO FOR THE NEXT TWO DAYS USING THE TRAINED LSTM MODEL represented using BLACK line:

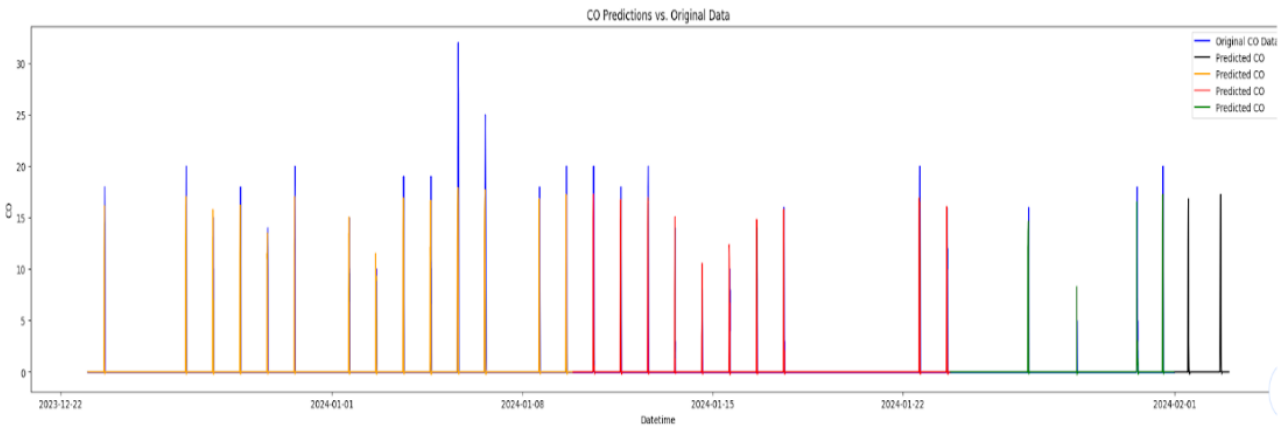


Figure 19. Forecasting data For CO for next two days prediction vs Original Data

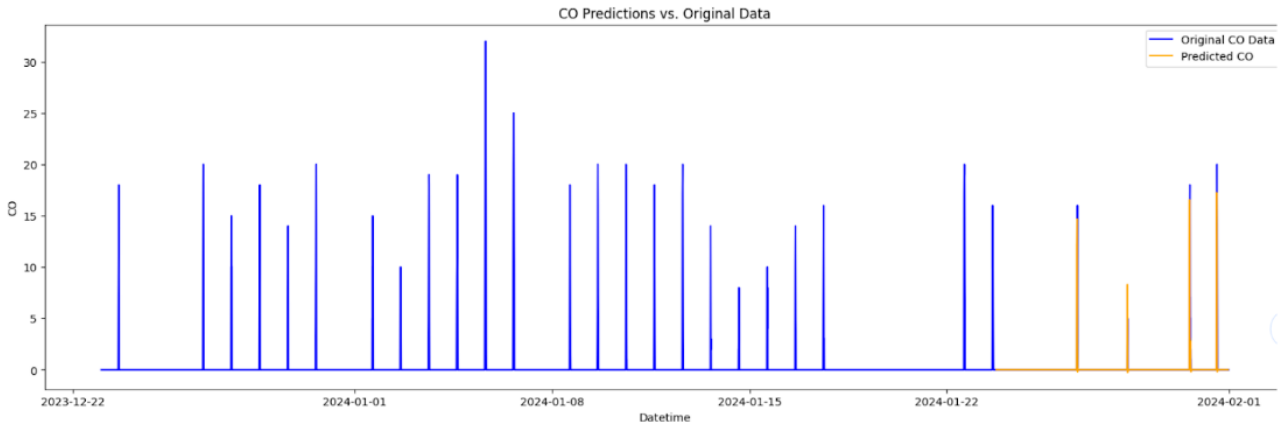


Figure 18. CO prediction vs Original Data

PREDICTIONS FOR THE NO Training, Validation, Testing sets using trained LSTM Model respectively:

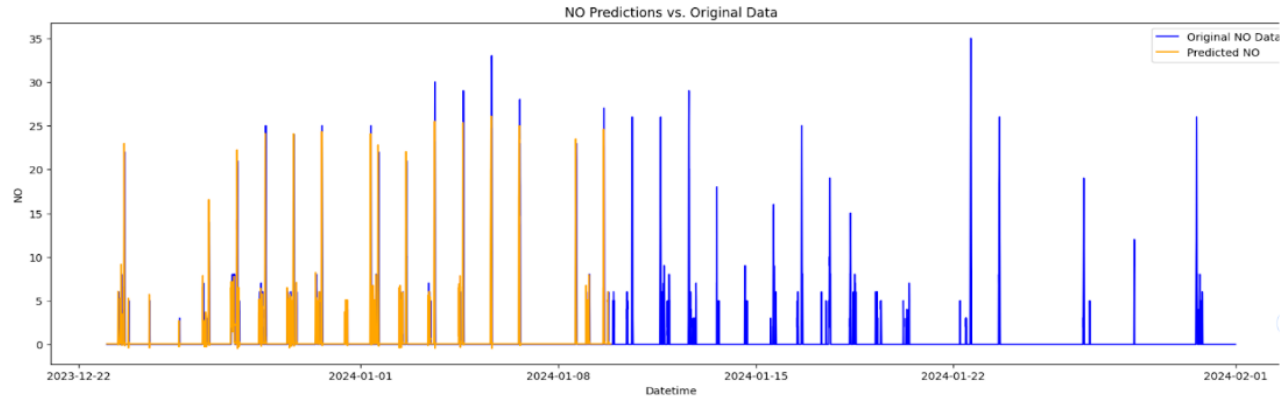


Figure 20. NO prediction vs Original Data

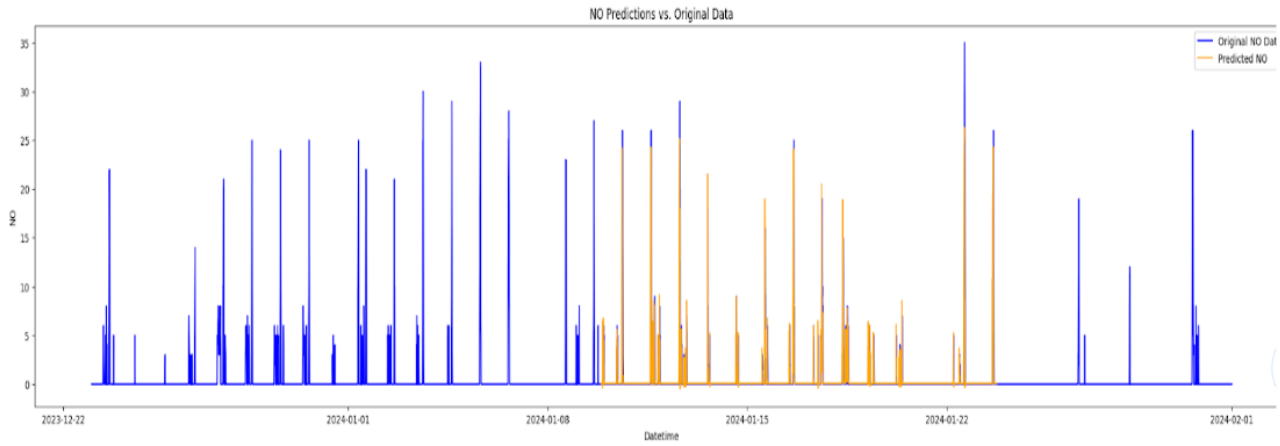


Figure 21. NO prediction vs Original Data

FORECASTED DATA FOR NO FOR THE NEXT TWO DAYS USING THE TRAINED LSTM MODEL represented using last Orange line:

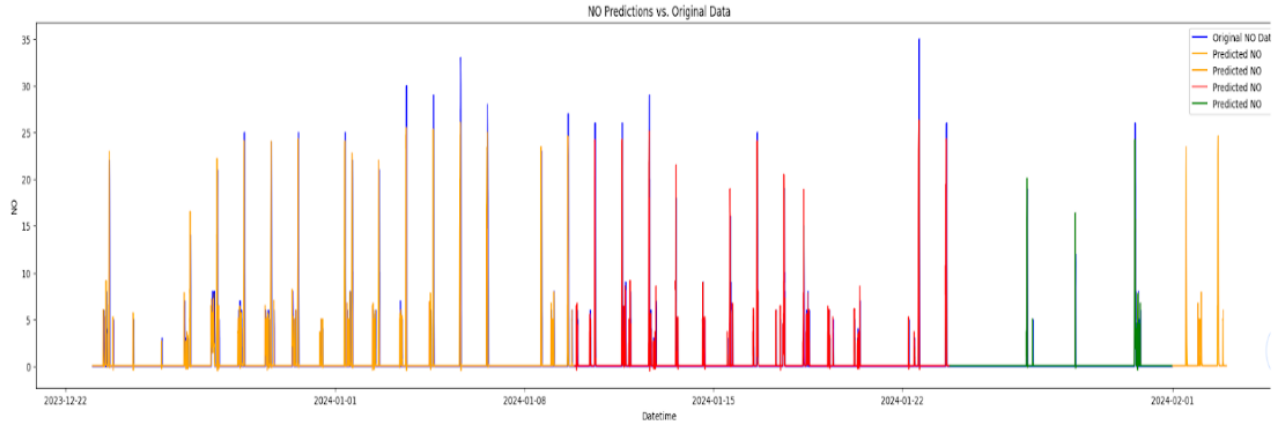


Figure 23. Forecasting data For NO for next two days prediction vs Original Data

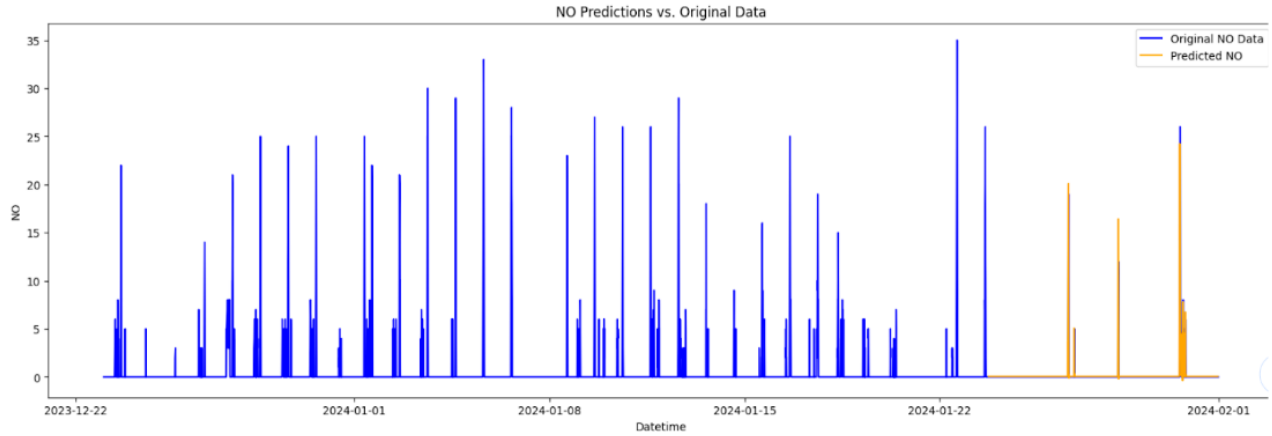


Figure 22. NO prediction vs Original Data

PREDICTIONS FOR THE Temperature Training, Validation, Testing sets using trained LSTM Model respectively:

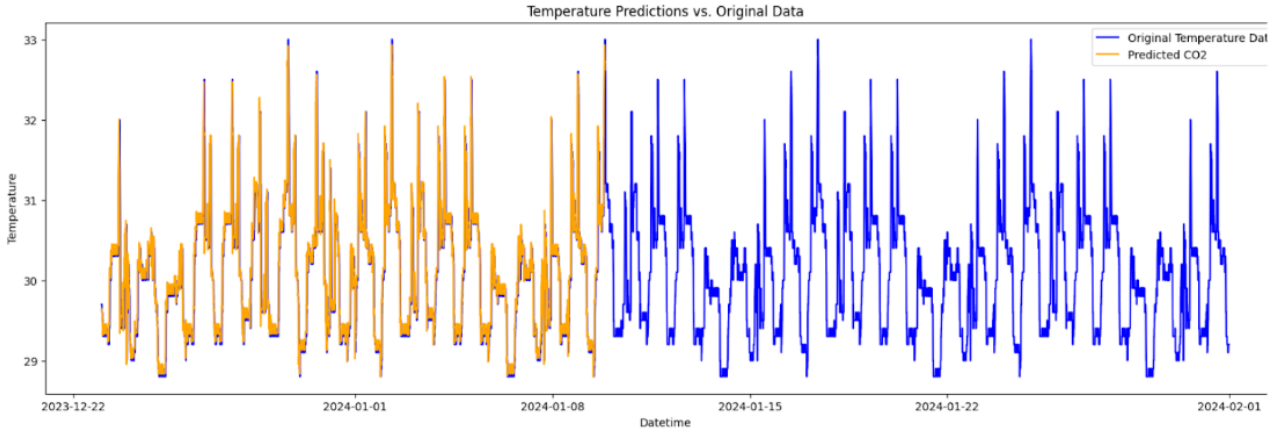


Figure 24. Temperature prediction vs Original Data

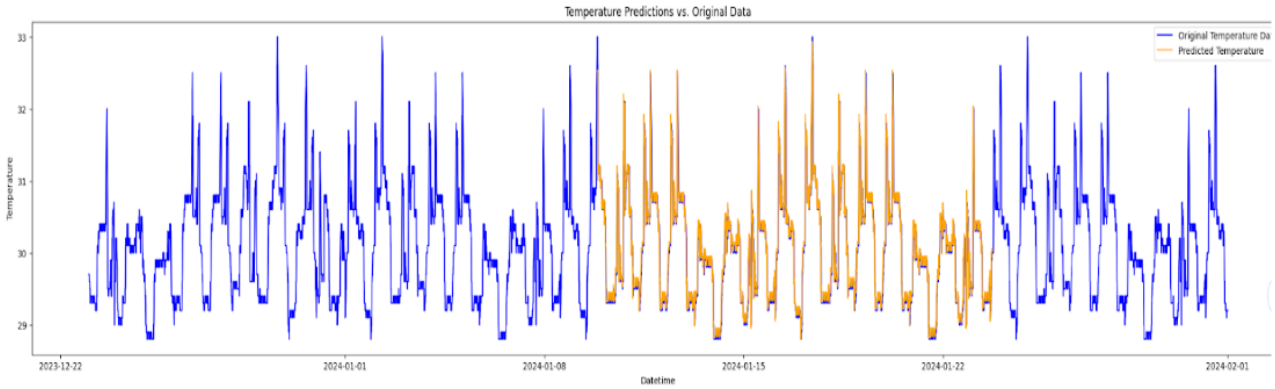


Figure 25. Temperature prediction vs Original Data

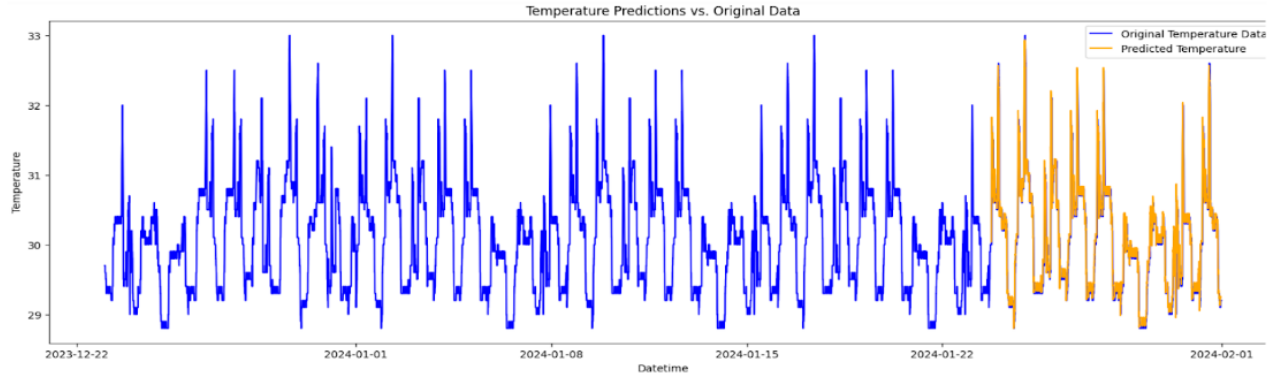


Figure 26. Temperature data prediction vs Original Data

FORECASTED DATA FOR Temperature FOR THE NEXT TWO DAYS USING THE TRAINED LSTM MODEL represented using BLACK line:

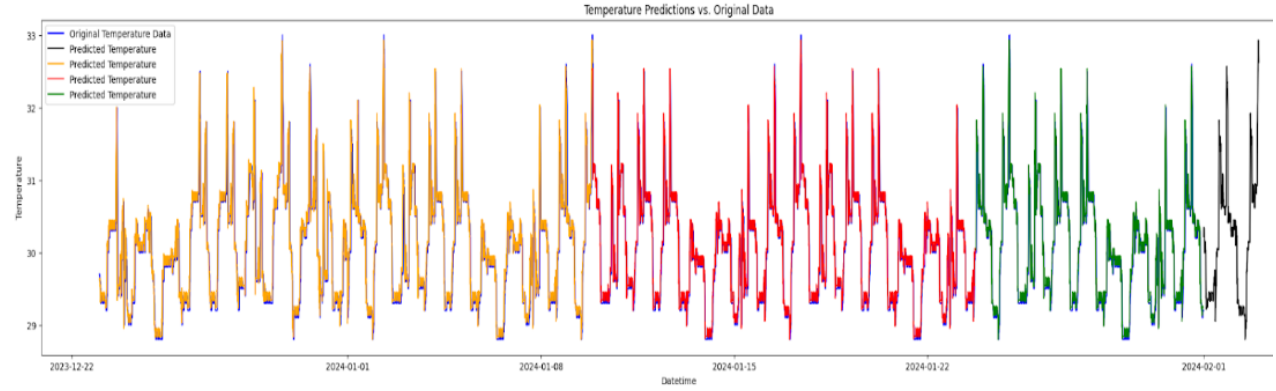


Figure 27. Forecasting Data For Temperature for next two days prediction vs Original Data

PREDICTIONS FOR THE Humidity Training, Validation, Testing sets using trained LSTM Model respectively:

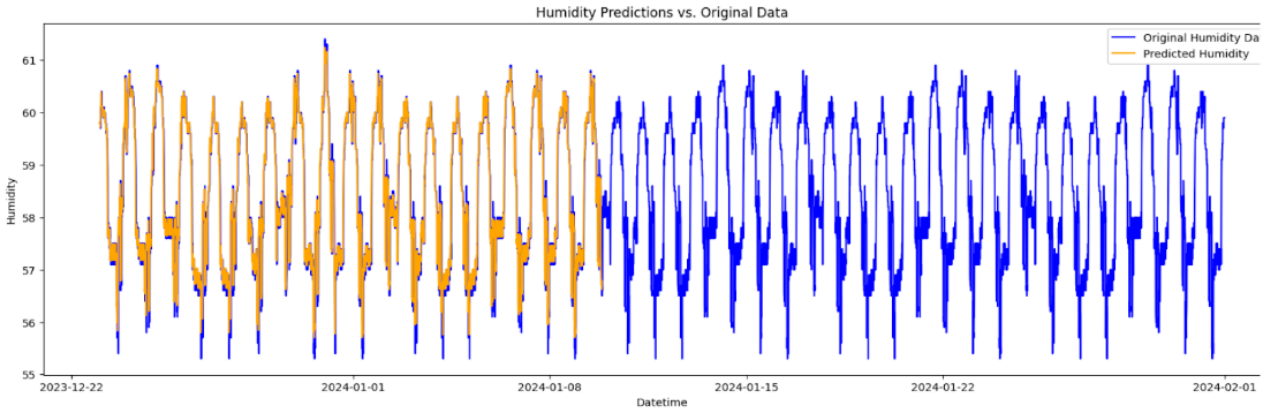


Figure 28. Humidity prediction vs Original Data

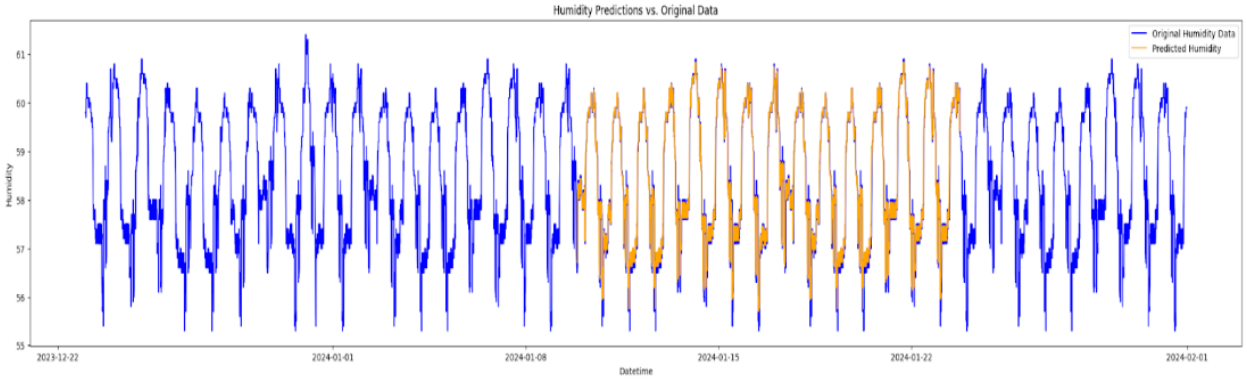


Figure 29. Humidity prediction vs Original Data

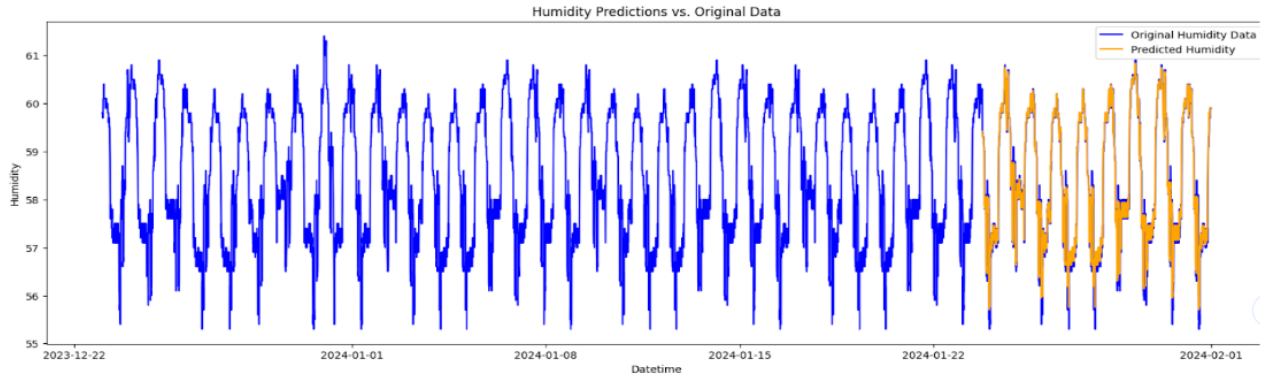


Figure 30. Humidity prediction vs Original Data

FORECASTED DATA FOR Humidity FOR THE NEXT TWO DAYS USING THE TRAINED LSTM MODEL represented using BLACK line:

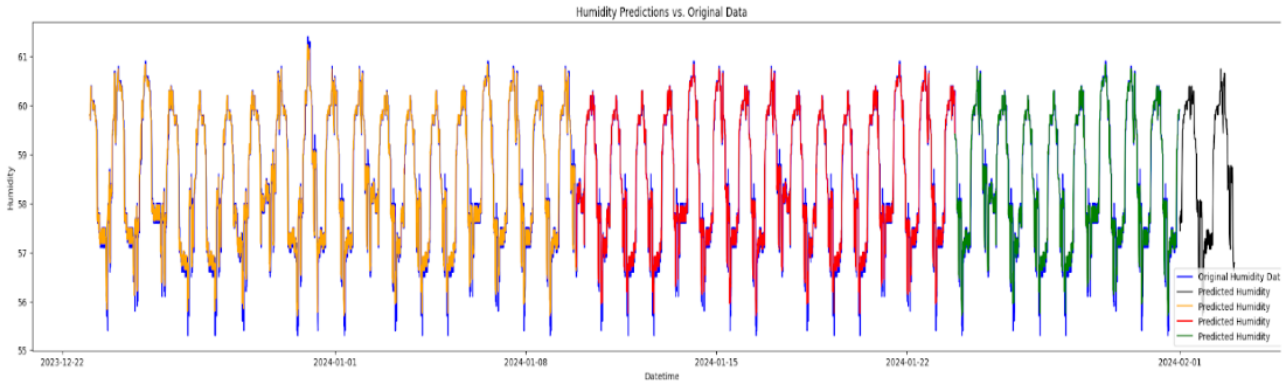


Figure 31. Forecasting Data For Humidity for next two days prediction vs Original Data

5.1.4 ERROR ANALYSIS:

VARIABLES	RMSE	MAE	MAPE
CO2	6.8972	4.8408	0.8355
CO	0.3788	0.0678	16505122
NO	0.7363	0.2202	69516928
TEMPERATURE	0.1294	0.0686	0.2262
HUMIDITY	0.3020	0.1901	0.3268

Figure 32. Error Analysis

RMSE, MAPE, and MAE serve as crucial evaluation metrics for predictive models in regression analysis or forecasting. RMSE assesses the average discrepancy between predicted and actual values, with an emphasis on larger errors due to the squared differences. MAPE offers insight into the average percentage difference between predicted and actual values, providing a relative measure of accuracy. Meanwhile, MAE quantifies the average absolute difference between predicted and actual values, offering a straightforward interpretation of error magnitude. These metrics collectively aid in understanding model performance, guiding decision-making processes, and facilitating model refinement endeavors.

5.2 SARIMA TIME SERIES FORECASTING

5.2.1 INTRODUCTION:

As it's a time series problem, we initially started with ARIMA, which is also a popular and effective modeling technique for capturing the temporal dependencies and trends present in time series data, including environmental parameters such as CO_2, CO, NO, Temp, and Humidity. In our dataset, we observed some daily or hourly trends/patterns. However, ARIMA is more suitable for data that changes over time but doesn't follow a regular pattern like seasons or cycles. It's akin to analyzing historical data to predict future trends without considering seasonal changes, such as forecasting next year's sales based solely on last year's sales and other factors. Subsequently, we explored Seasonal ARIMA (SARIMA) models, which are an extension of ARIMA designed to handle time series data with seasonal patterns. SARIMA not only considers the overall trend and random fluctuations but also accounts for the regular ups and downs that occur at certain times of the year.

Seasonal ARIMA:

1. To implement SARIMA on a dataset, the dataset should be stationary.
2. To check stationarity, we need to conduct the Dickey-Fuller test on the dataset. Check the results of the test. Here, the null hypothesis (H0) states that the data is not stationary: If p ¿0.05 =¿ accept H0, data is not stationary If p¡0.05 =¿ Strong evidence to reject null hypothesis (data is stationary)
3. If the data is not stationary, then we need to apply differencing by subtracting

each observation from its previous observation. Then, repeat step 2.

4. If the data is stationary, we can proceed to implement the SARIMA model on our dataset.

5. To fit and build the model, we need to specify three components: AR (Auto Regression), I (Integrated - for differencing), and MA (Moving Average), along with four seasonality parameters P, D, Q, and S.

6. Optimal values for these components can be chosen by analyzing two plots: the Autocorrelation Function (ACF) and the Partial Autocorrelation Function (PACF).

7. Since the Dickey-Fuller test indicated that the data is stationary, we can set d=0 (for differencing) in our model. For the seasonal counterpart of d, I, we will set D=1 as it slightly improves the model's fit compared to D=0.

8. Determine the q value for the MA(q) part of the model by inspecting the autocorrelation chart. The values are chosen based on the specific charts corresponding to each of the five parameters.

9. Similarly, we check for large spikes in the partial autocorrelation charts to determine the AR(p) value.

10. Calculate the metrics MSE, MAE, MPAE, R-Squared.

11. R-squared measures the goodness of fit of a regression model, ranging from 0 to 1. the value of 1 indicates a perfect fit, while 0 suggests no relationship between variables.

5.2.2 SARIMA MODEL FOR CO_2:

Step-1: Load data, Preprocessing, Performing Dicky-Fuller test:

```
Dickey-Fuller Test:
Test Statistic                  -9.960068e+00
p-value                          2.386616e-17
# Lags Used                      3.500000e+01
Number of Observations Used      1.135600e+04
dtype: float64
```

Figure 33. Dickey Fuller Text for CO_2

Step-2 ACF and PACF plots:

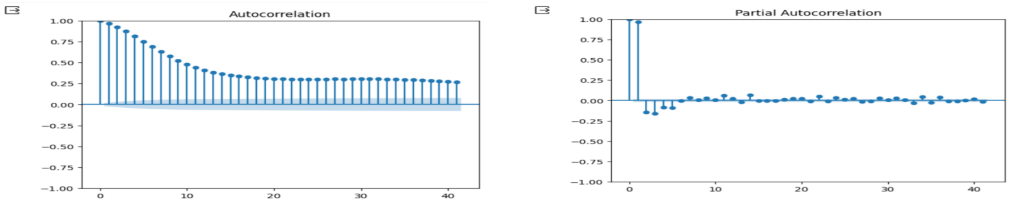


Figure 34. ACF and PACF plots for CO_2

Based on the above plots, the possible AR(p) value is up to 8. However, we observed that a value of 5 yields good results, so we selected 5 as the value of p. The seasonality P value is taken as 1.

The integrating (d) value is 0 because we found that our dataset was stationary from the beginning. The seasonality D value is taken as 1.

The Moving Average (q) value can be up to 40 as every spike is within this range. However, for our model, we obtained good results at q=2. The seasonality Q is taken as 1.

The seasonality period is 6, seasonal period to 6 means that the model considers a seasonal cycle of 6 * 5 = 30 minutes. In other words, the model accounts for seasonal patterns that repeat every 30 minutes in the dataset.

Step-3 Build and Define SARIMAX Model:

Build the SARIMA model with choosed values.

Step-4 Analyze Results:

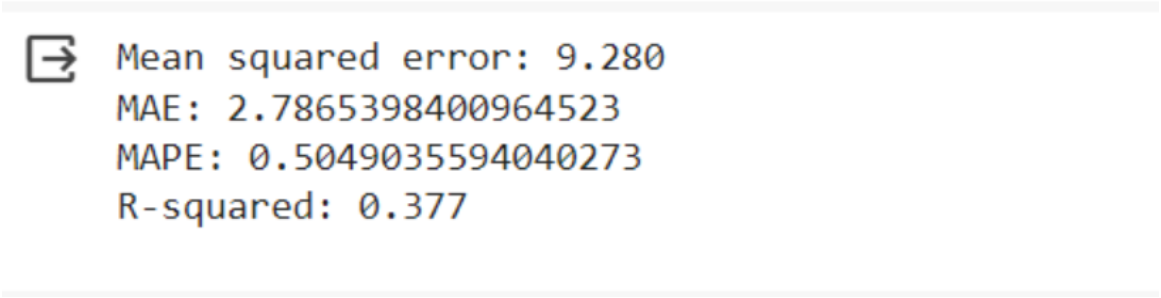


Figure 35. CO_2 Analysis using SARIMAX. MSE, MAE, MAPE and R-Squared

- The MSE of 9.280 suggests moderate accuracy in the forecasts.
- The MAE of 2.787 indicates the average magnitude of the errors in the forecasted values.
- The MAPE of 0.505percent suggests that, on average, the forecasted CO_2 values deviate from the actual values by approximately 0.5percent.
- The R-squared value of 0.377 suggests that the SARIMAX model explains approximately 37.7percent of the variance in the CO_2 values, indicating a moderate fit.

Plot

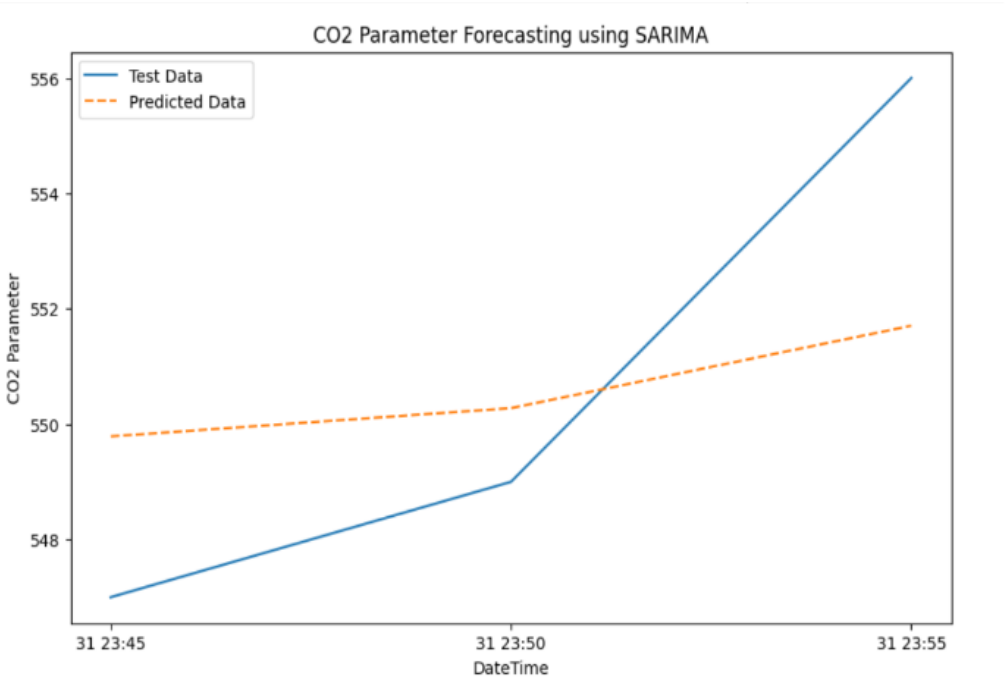


Figure 36. CO_2 Parameter Forecasting using SARIMA

Forecast CO_2 values for next 2 days (1st Feb 2024 12AM to 2nd Feb 2024 11PM):

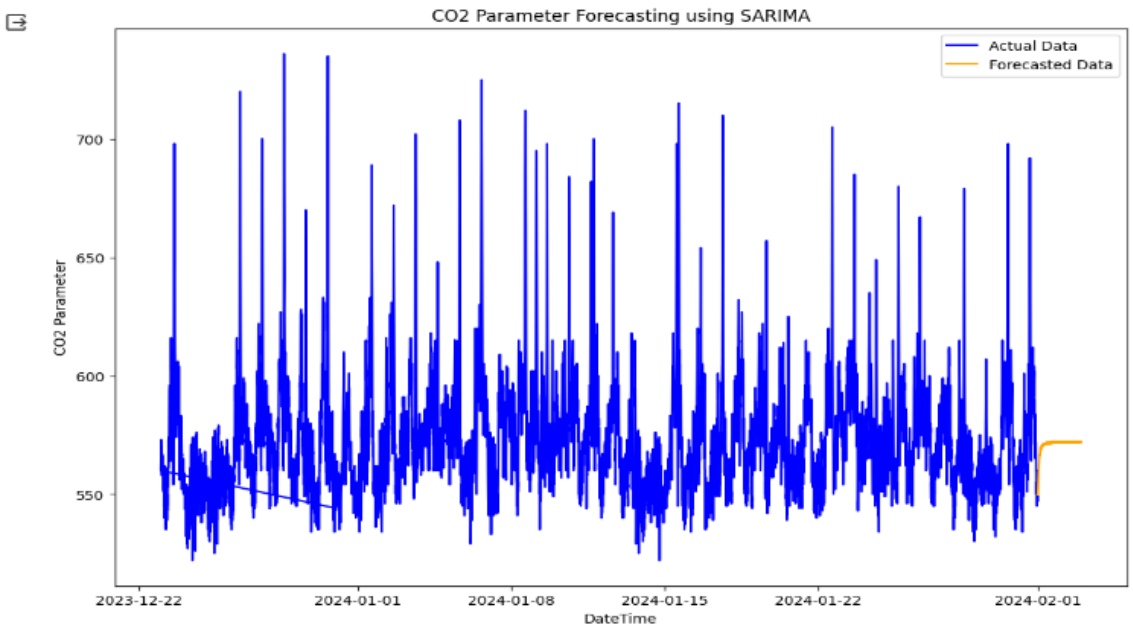


Figure 37. CO_2 Parameter Forcasting for next 2 days using SARIMA

5.2.3 SARIMA MODEL FOR CO:

Step-1 Load data, do preprocessing

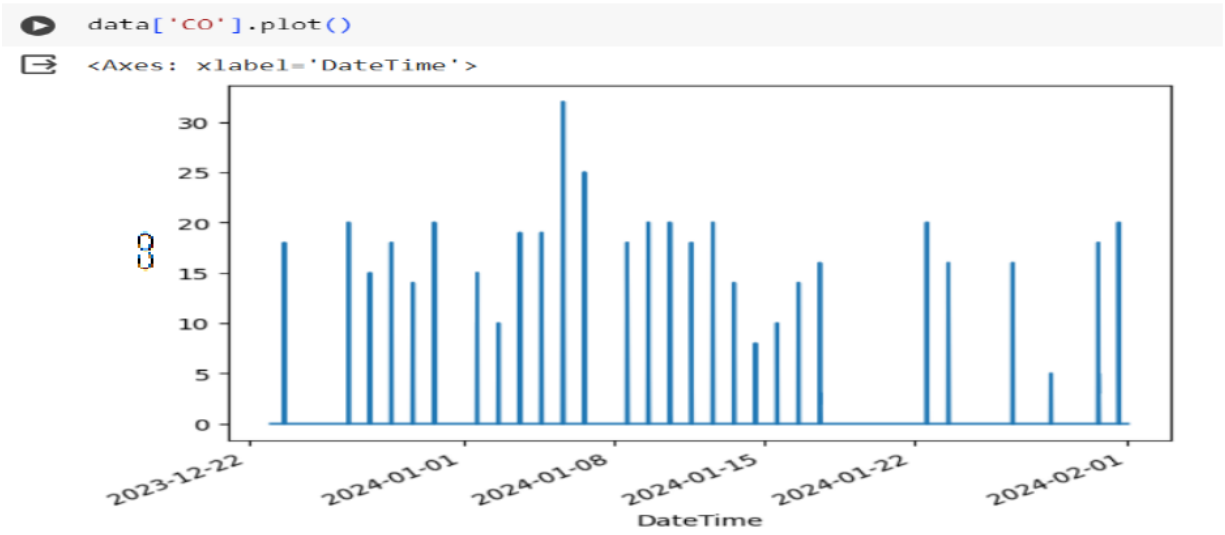


Figure 38. plotting CO data

Conduct Dickey-Fuller test:

```
Dickey-Fuller Test:
Test Statistic                  -22.524881
p-value                           0.000000
# Lags Used                      15.000000
Number of Observations Used    11376.000000
dtype: float64
```

Figure 39. Dickey Fuller Text for CO

Based on the Dickey-Fuller test results, since the p-value is less than the significance level (typically 0.05), we reject the null hypothesis, and thus, the dataset is considered stationary. Therefore, there's strong evidence to suggest

that the time series data is stationary.

Step-2 ACF and PACF plots:

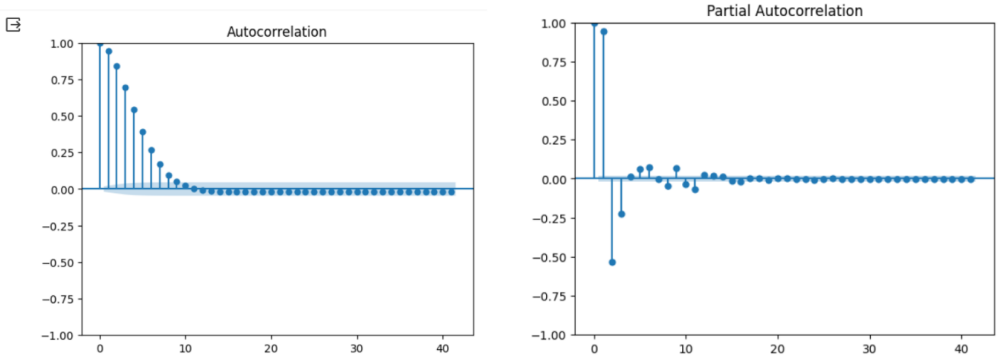


Figure 40. ACF and PACF plots for CO

Based on the above plots, the possible AR(p) value is up to 11. However, we observed that a value of 5 yields good results, so we selected 5 as the value of p. The seasonality P value is taken as 1.

The integrating (d) value is 0 because we found that our dataset was stationary from the beginning. The seasonality D value is taken as 1.

The Moving Average (q) value can be up to 8 as every spike is within this range. However, for our model, we obtained good results at q=5. The seasonality Q is taken as 1.

The seasonality period is 20, seasonal period to 20 means that the model considers a seasonal cycle of 20 * 5 = 100 minutes. In other words, the model accounts for seasonal patterns that repeat every 100 minutes in the dataset.

Step-3 Build and Define SARIMAX Model:

Build the SARIMA model with choosed values.

Step-4 Analyze results:

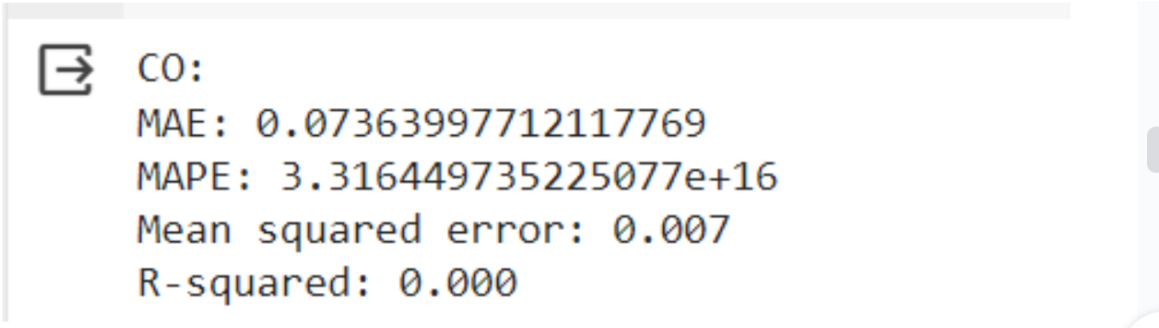


Figure 41. CO Analysis using SARIMAX. MSE, MAE, MAPE and R-Squared values

- The average magnitude of errors between expected and actual values is measured by the MAE. The mean absolute error (MAE) in this instance is 0.0736, indicating that the model's predictions often differ by 0.0736 units from the actual CO readings.
- Prediction errors are expressed by the MAPE as a percentage of the actual values. The remarkably high MAPE value of 3.316449735225077e+16 suggests notable differences between the actual and projected values, which may suggest subpar model performance.
- The mean squared error (MSE) calculates the average of the squared deviations between the expected and actual values. The MSE in this case is 0.007, meaning that the squared differences between the actual and anticipated CO values are, on average, 0.007 units.
- The percentage of the dependent variable's (CO) variance that can be predicted from the independent variables (the model's predictions) is indicated

by the R-squared value. A value of 0 denotes the absence of any variability in the data that the model can explain. The R-squared value in this instance is 0.000, indicating that no variability in the CO data is captured by the model.

Plot

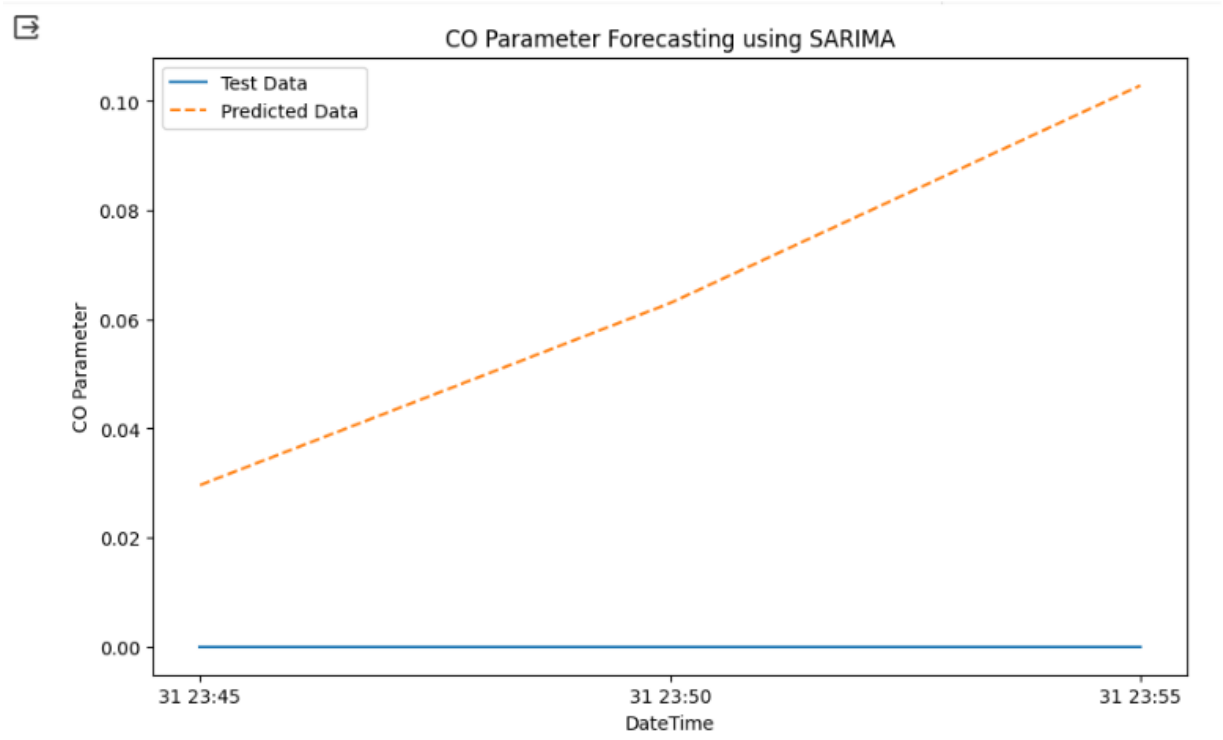


Figure 42. CO Parameter Forecasting using SARIMA

Overall, the high MAPE, low R-squared, and relatively high MAE and MSE values point to the possibility that the SARIMA model for CO is not adequately capturing the underlying patterns or trends in the data. This suggests that the model has to be further refined or that different modeling strategies should be

taken into account.

Forecast CO values for next 2 days (1st Feb 2024 12AM to 2nd Feb 2024 11PM):

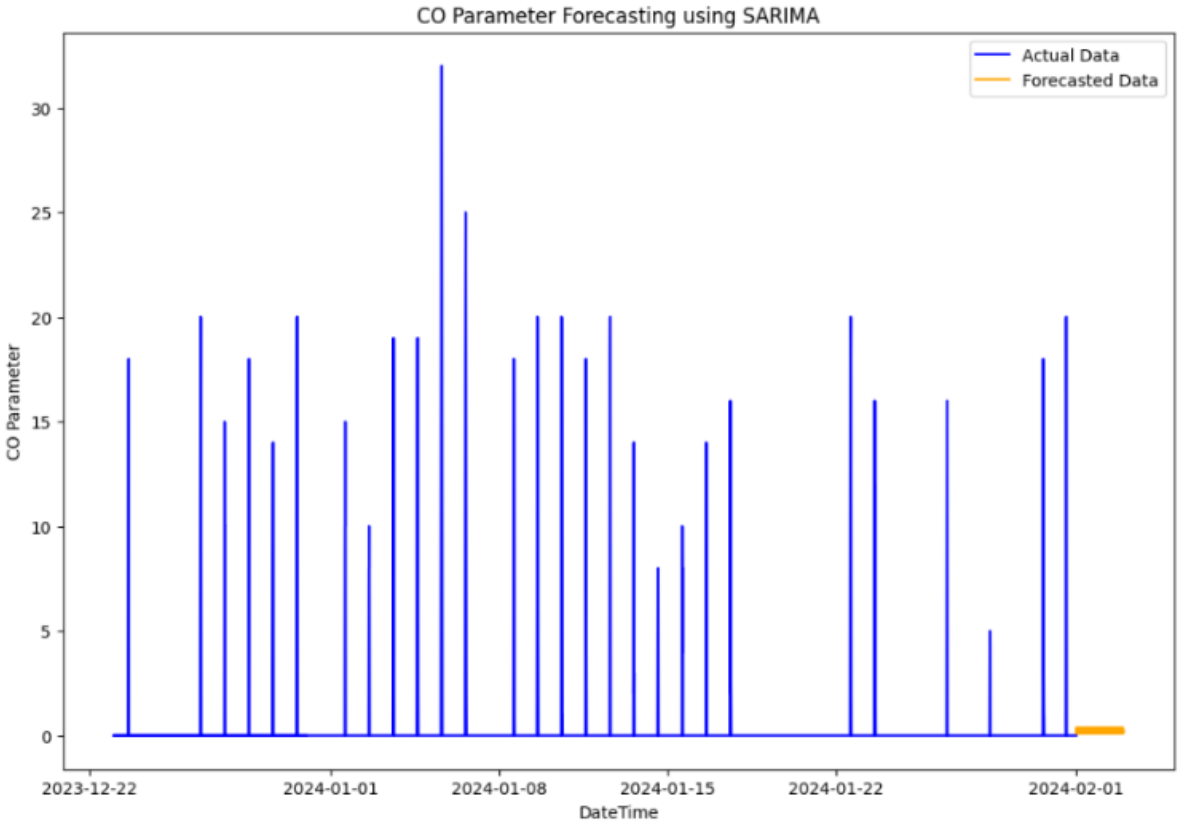


Figure 43. CO Parameter Forcasting for next 2 days using SARIMA

5.2.4 SARIMA MODEL FOR NO

Similarly, SARIMA model for NO also showing same results.

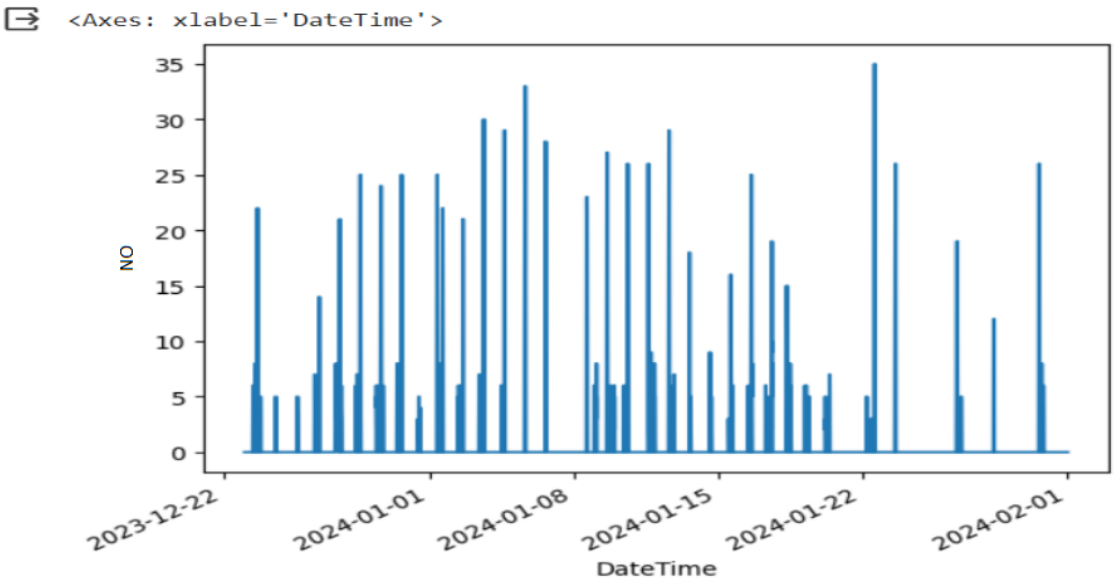


Figure 44. plotting data NO

Conduct Dickey-Fuller test:

```
Dickey-Fuller Test:
Test Statistic                  -1.688255e+01
p-value                          1.060141e-29
# Lags Used                      2.800000e+01
Number of Observations Used      1.136300e+04
dtype: float64
```

Figure 45. Dickey Fuller Text for NO

ACF and PACF plots for NO:

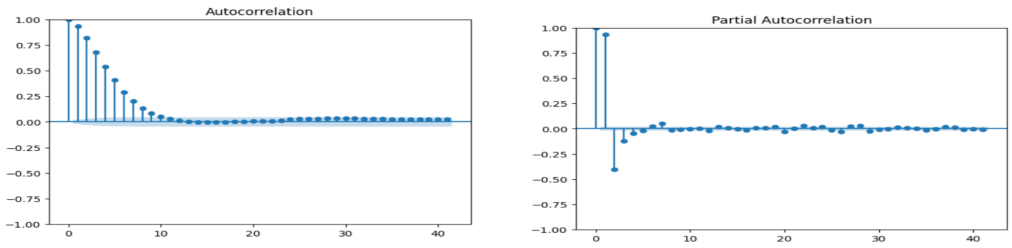


Figure 46. ACF and PACF plots for NO

Here, p value can choose upto 7, we choose 3, d-value 0, q value can choose upto 9, we choose 5. P,D,Q are 1. For NO also repeating only once a day that too in the afternoon for 1 and half hour.So, Seasonality period was 100 minutes. **Build Model** Build and fit SARIMA Model:

Analyze results:

```
NO:
Mean squared error: 0.019
R-squared: 0.000
MAE: 0.11715469942630612
MAPE: 5.276178606810147e+16
Mean squared error: 0.019
R-squared: 0.000
```

Figure 47. NO Analysis using SARIMAX. MSE, MAE, MAPE and R-Squared values

- In this case, the model's predictions differ from the actual NO levels by an average of 0.117 units, according to the MAE of 0.117.

- The exceptionally high MAPE value of 5.2761786068101 47e+16 shows notable differences between the actual and projected values, which may be a symptom of subpar model performance.
- The MSE for NO is 0.019, suggesting that the squared differences between the actual and predicted NO levels are, on average, 0.019 units.
- The percentage of the dependent variable's (NO) variation that can be predicted from the independent variables (the model's predictions) is expressed as the R-squared value. A value of 0 denotes the absence of any variability in the data that the model can explain.

  The R-squared value in this case is 0.000, showing that none of the variability in the NO data is captured by the model.

Plot

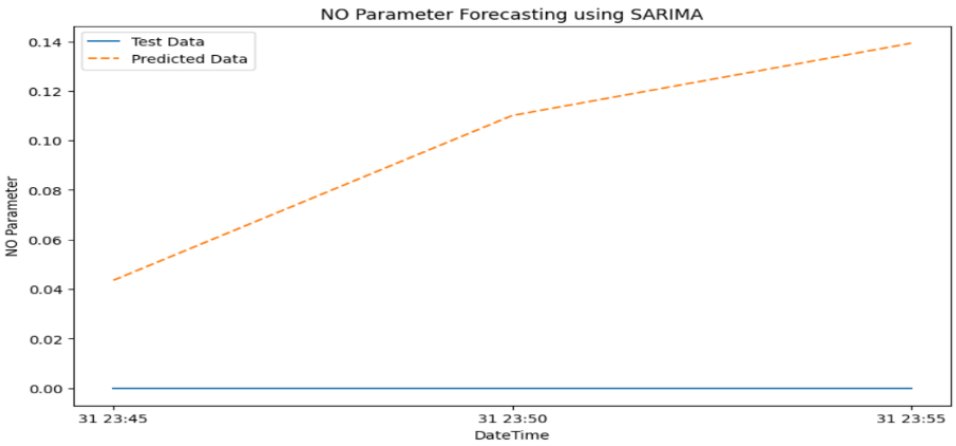


Figure 48. NO Parameter Forecasting using SARIMA

Forecast NO values for next 2 days (1st Feb 2024 12AM to 2nd Feb 2024 11PM):

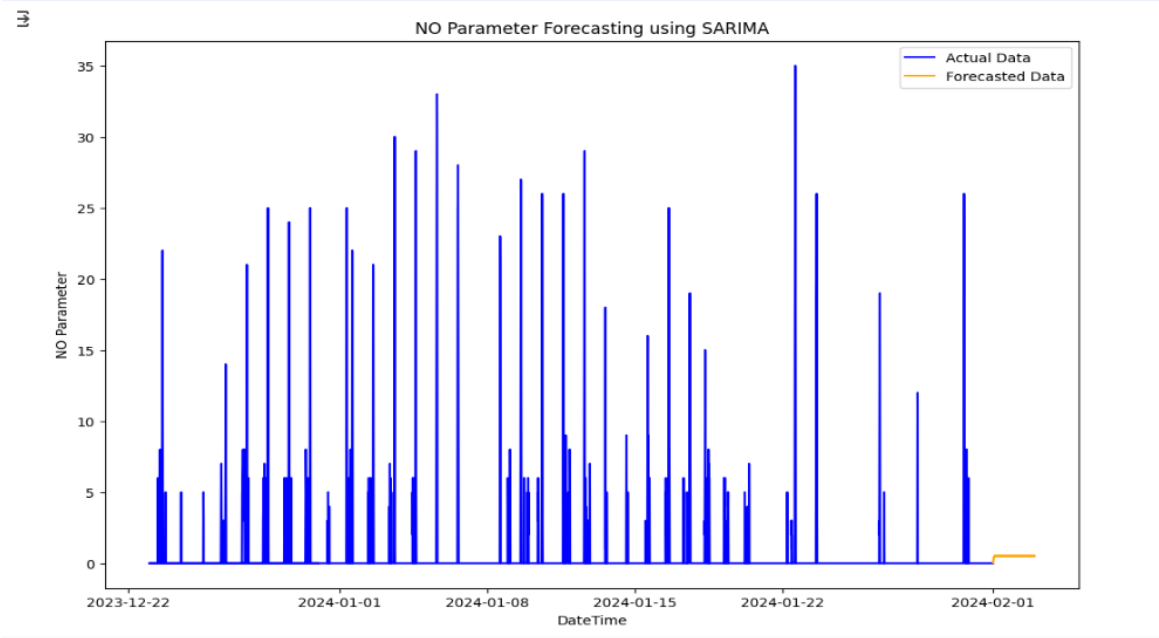


Figure 49. NO Parameter Forcasting for next 2 days using SARIMA

overall, the NO results show weak model performance, much like the CO results did. The low R-squared and high MAPE values suggest that the underlying patterns or trends in the data may not be adequately captured by the SARIMA model for NO. It could be essential to investigate different modeling strategies or adjust the model even more.

5.2.5 SARIMA MODEL FOR TEMPERATURE:

Step-1 Loading data, do preprocessing

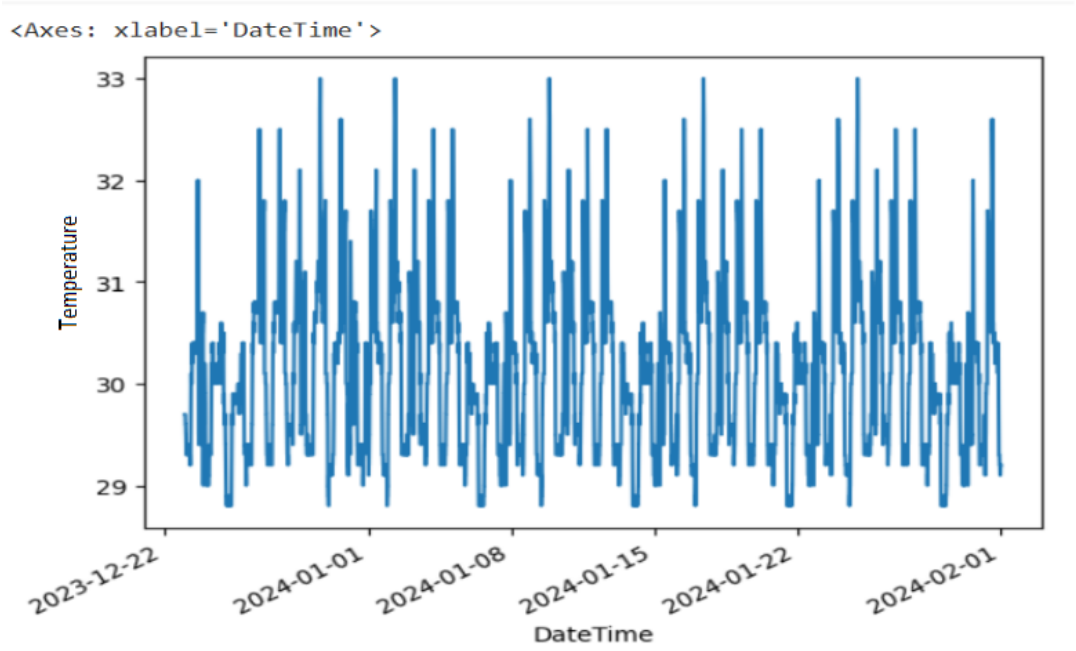


Figure 50. plotting Temperature data from dataset

Conduct Dickey-Fuller test:

```
Dickey-Fuller Test:
Test Statistic                  -1.020384e+01
p-value                          5.874641e-18
# Lags Used                      3.600000e+01
Number of Observations Used      1.135500e+04
dtype: float64
```

Figure 51. Dickey Fuller Text for Temperature

Here, p-value is less than 0.05, so we can reject the null hypothesis. Hence our dataset is stationary.

Step-2 ACF and PACF plots:

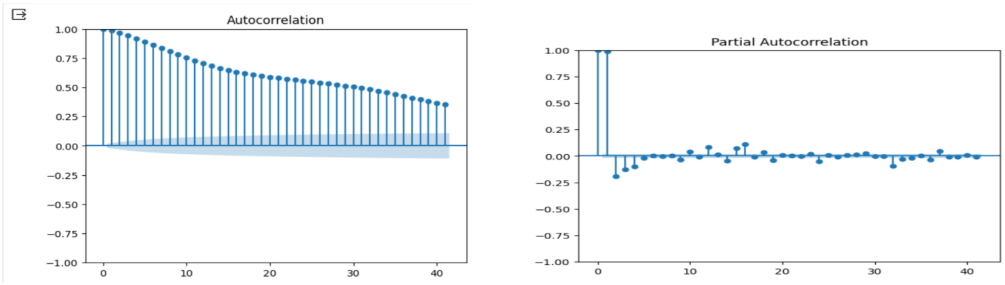


Figure 52. ACF and PACF plots for Temperature

From above plots, we can take p-value upto 16, we choose p-value as 4 and q-value can take upto 40, here we choose 7, d-value is 0. P,D,Q are 1. For temperature we can't find any seasonality patterns in the dataset, but some values are repeated in a day for that reason I choose seasonality period as 12. But this approach was wrong because there is no proper seasonality pattern or trend in the temperature data.

Step-3 Build and Define SARIMAX Model:

Build and fit SARIMA Model with there chosen values

Step-4 Analyze Results:

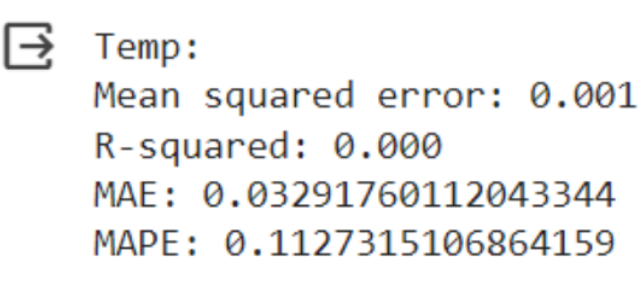


Figure 53. Temperature Analysis using SARIMAX. MSE, MAE, MAPE and R-Squared

- The model's predictions generally differ by 0.0329 units from the actual temperature values, according to the MAE of 0.0329.
- In comparison to the CO and NO models, the MAPE value of 0.1127 shows comparatively small differences between expected and actual values. This implies improved relative error performance.
- Temperature's mean squared deviations between expected and actual temperature readings are 0.001 units, according to the temperature MSE of 0.001.
- The percentage of variance in the dependent variable (temperature) that can be predicted from the independent variables (the model's predictions) is indicated by the R-squared value. When a value is zero, it means that there is no variability in the data that the model can explain. Since the R-squared value in this case is 0.000, it appears that none of the variability in the temperature data is captured by the model.

Plot

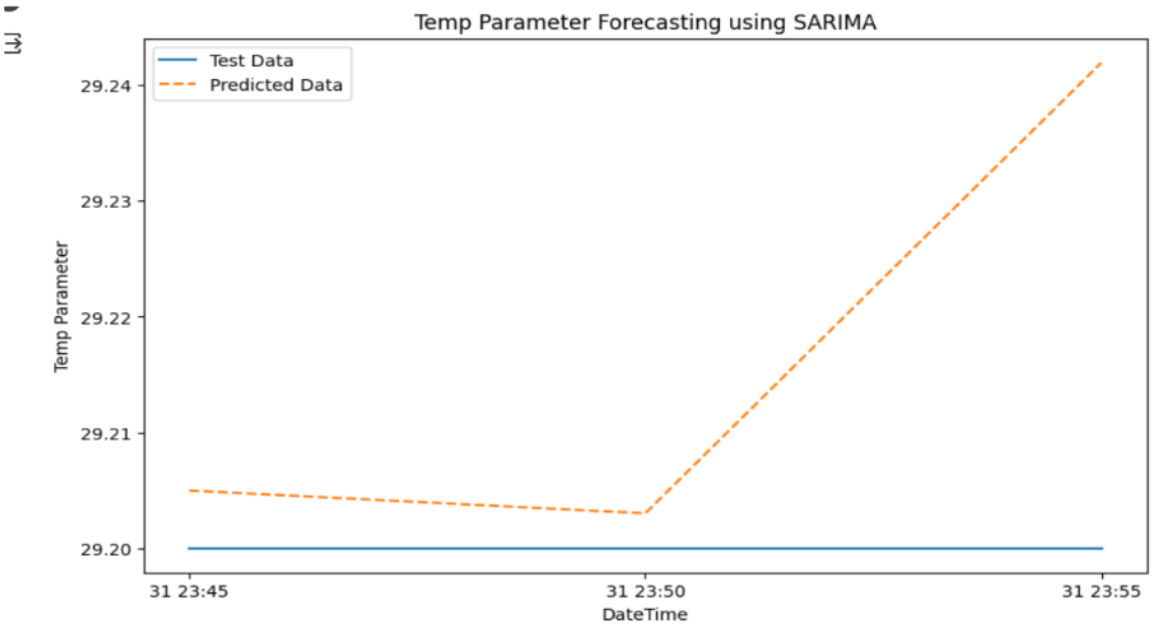


Figure 54. Temperature Parameter Forecasting using SARIMA

Forecast Temperature values for next 2 days (1st Feb 2024 12AM to 2nd Feb 2024 11PM):

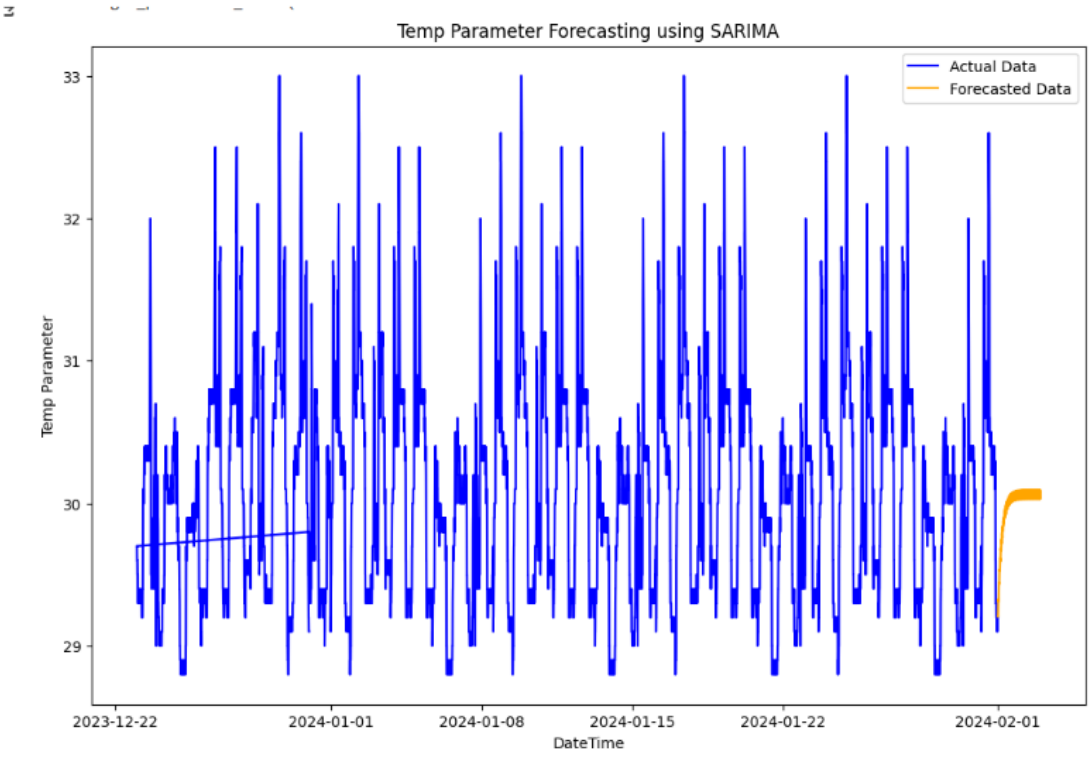


Figure 55. Temperature Parameter Forcasting for next 2 days using SARIMA

Overall, the low R-squared value suggests that the SARIMA model may not adequately capture the underlying patterns or trends in the temperature data, even while the MAE and MAPE values reveal comparatively low prediction errors for temperature when compared to CO and NO. It might take further research or model improvement to get better results.

5.2.6 SARIMA MODEL FOR HUMIDITY:

Step-1 Load Data and preprocess data:

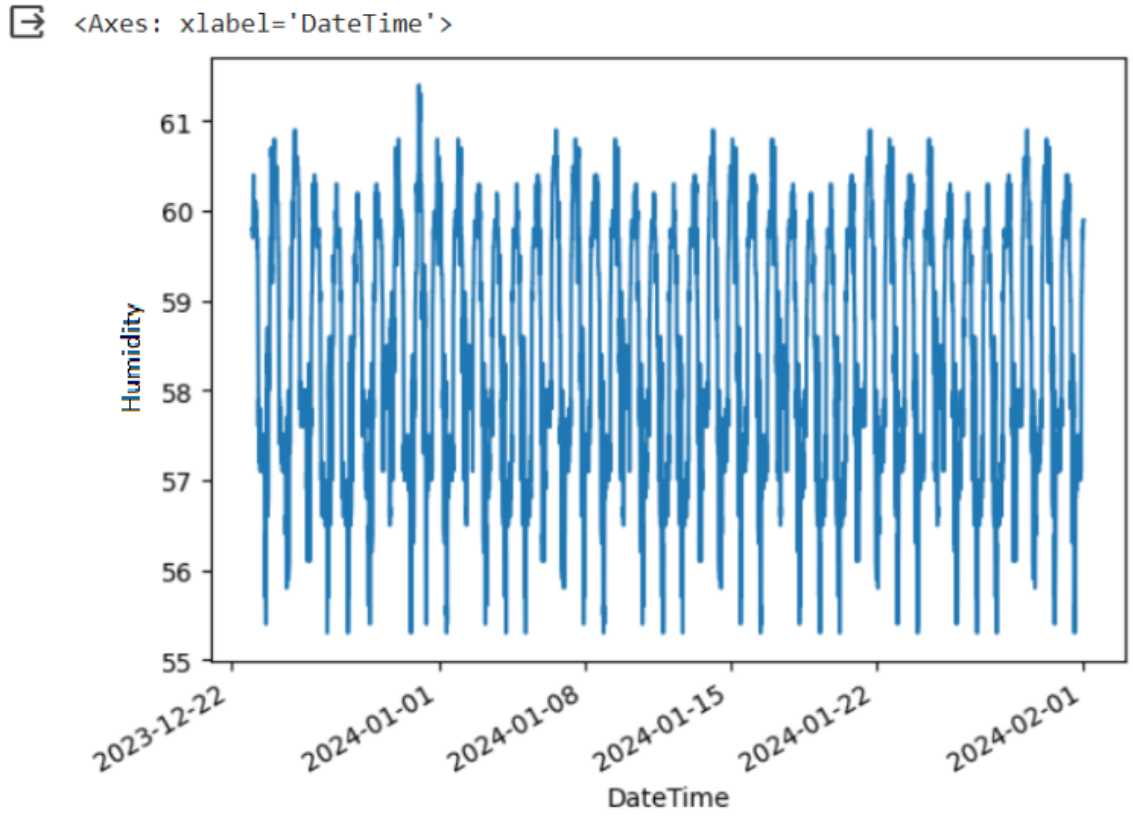


Figure 56. plotting Humidity data from dataset

Conduct Dickey-Fuller test:

```
Dickey-Fuller Test:
Test Statistic                  -1.117349e+01
p-value                          2.605205e-20
# Lags Used                      3.900000e+01
Number of Observations Used      1.135200e+04
dtype: float64
```

Figure 57. Dickey Fuller Text for Temperature

Here, p-value is less than 0.05, then we can reject the null hypothesis, Hence our data is stationary.

Step-2 ACF and PACF plots:

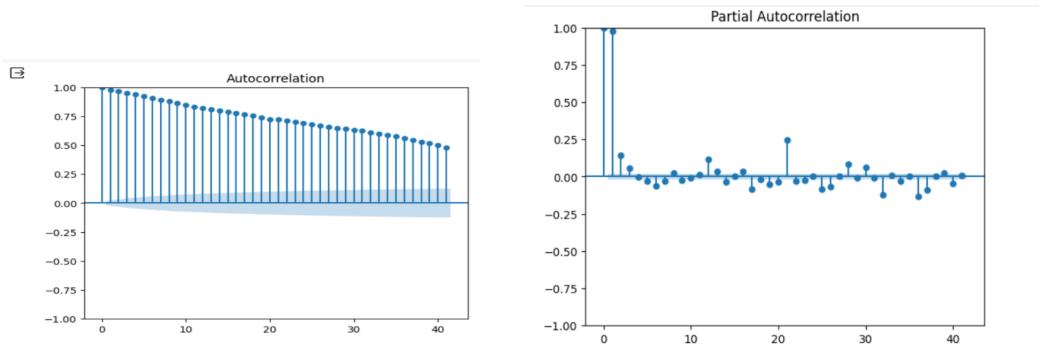


Figure 58. ACF and PACF plots for Humidity

Based on the above plots we can choose p, q values. Here we can choose p value upto maximum 21. Here we choose p value 10. q value we can choose upto 40, but here we choose 8. d value is 0. P, D,Q are 1 and seasonality period here 12 (60 minutes).

Next Step Build SARIMAX Model:

Step-4 Analyze Results:

```
Humidity:
Mean squared error: 0.001
R-squared: 0.000
MAE: 0.025792497979613433
MAPE: 0.043059262069471 51
```

Figure 59. Humidity Analysis using SARIMAX. MSE, MAE, MAPE and R-Squared

- The MAE is 0.0258, indicating that, on average, the model's predictions deviate from the actual humidity values by approximately 0.0258 units.
- The MAPE value of 0.0431 suggests relatively low prediction errors compared to the CO and NO models, indicating better performance in predicting humidity values.
- The MSE for humidity is 0.001, indicating that, on average, the squared deviations between predicted and actual humidity values are 0.001 units.
- The R-squared value measures the proportion of the variance in the dependent variable (humidity) that is predictable from the independent variables (the model's predictions). A value of 0 indicates that the model does not explain any variability in the data. Here, the R-squared value is 0.000, suggesting that the model does not capture any of the variability in the humidity data.

Plot

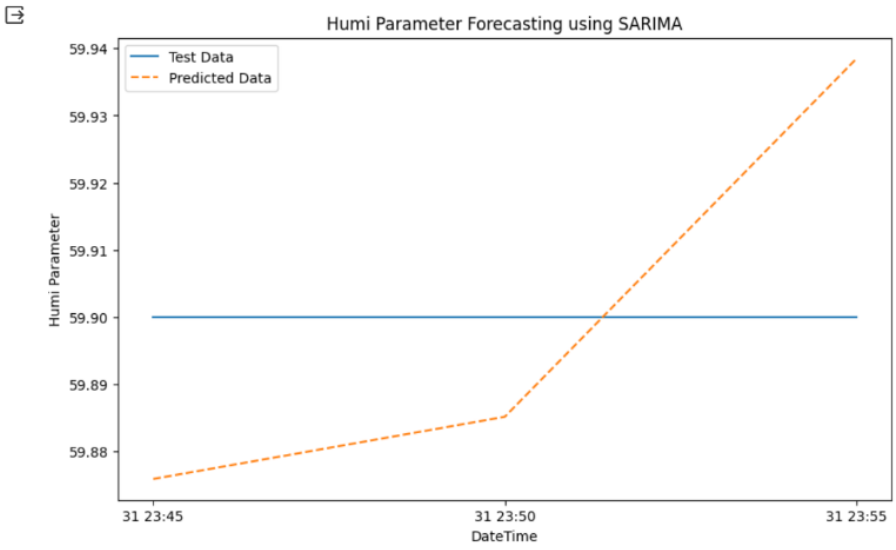


Figure 60. Humidity Parameter Forecasting using SARIMA Model

Overall, similar to the temperature model, the MAE and MAPE values suggest relatively low prediction errors for humidity. However, the low R-squared value indicates that the SARIMA model may not effectively capture the underlying patterns or trends in the humidity data. Further investigation or model refinement may be necessary to improve performance.

Forecast Humidity values for next 2 days (1st Feb 2024 12AM to 2nd Feb 2024 11PM):

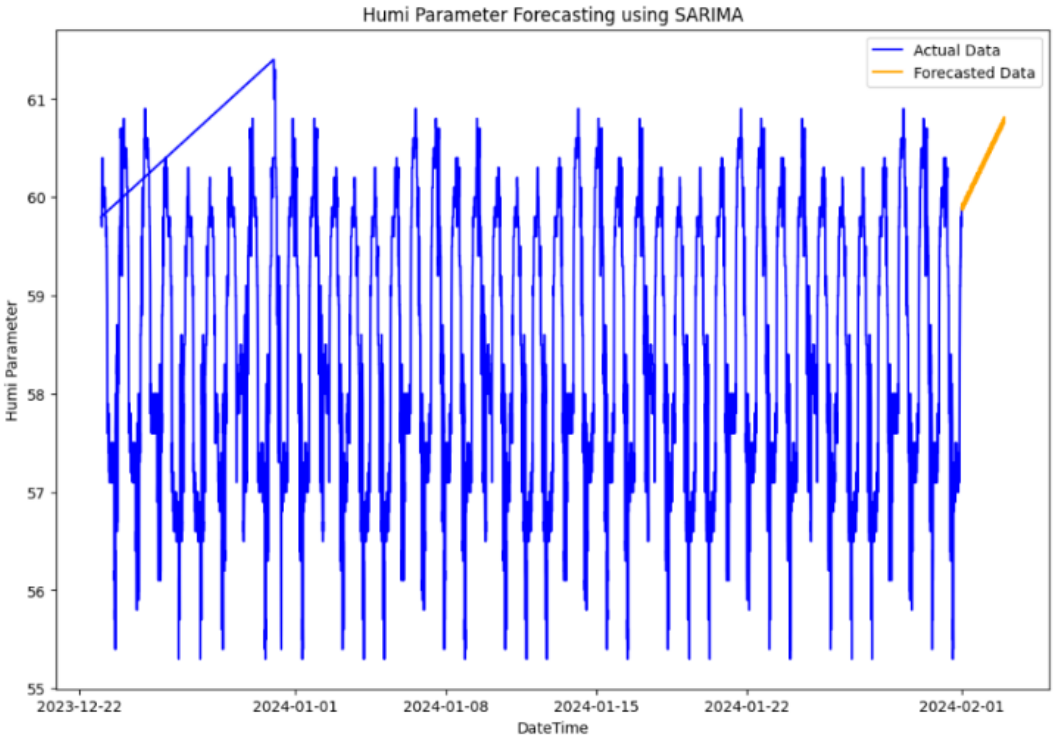


Figure 61. Humidity Parameter Forcasting for next 2 days using SARIMA

5.3 XGBOOST TIME SERIES FORECASTING

5.3.1 Introduction:

XGBoost is a highly effective machine learning algorithm widely utilized for regression and classification tasks, including time series forecasting. It belongs to the family of ensemble learning methods, specifically gradient boosting algorithms, which iteratively build decision trees to minimize a specified loss function and create a robust predictive model.

FOR CO_2

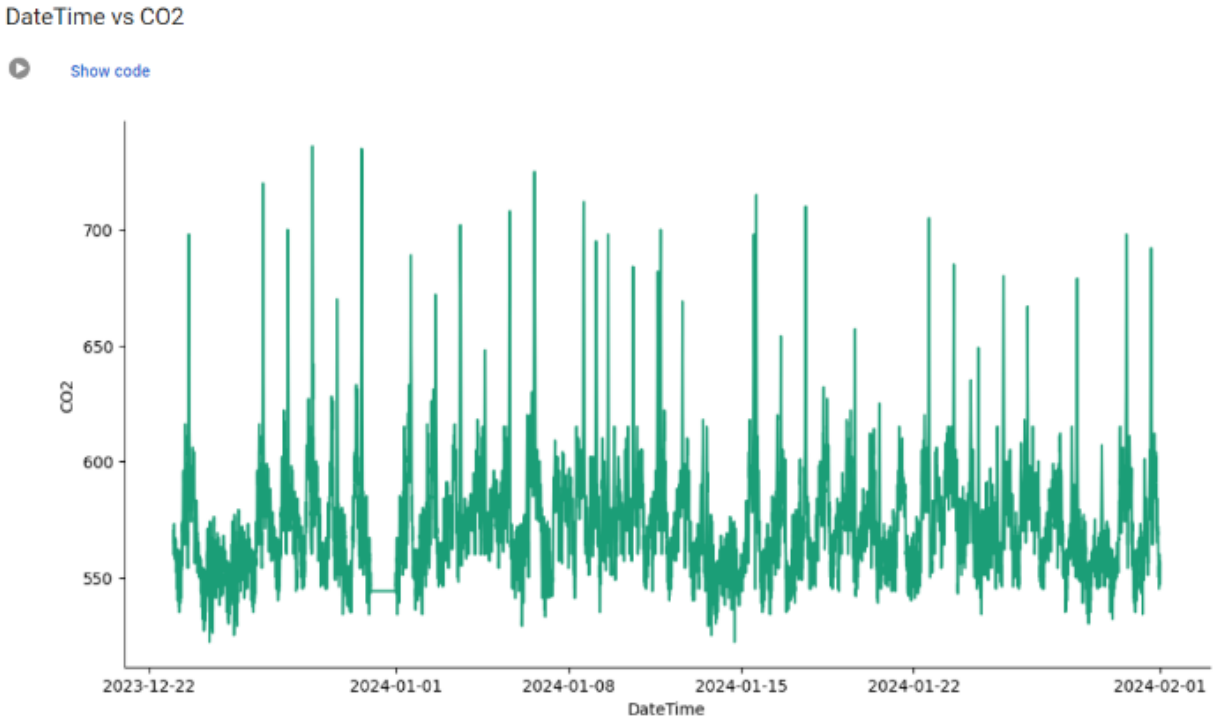


Figure 62. Visualization of data CO_2 vs DaTime

FOR CO

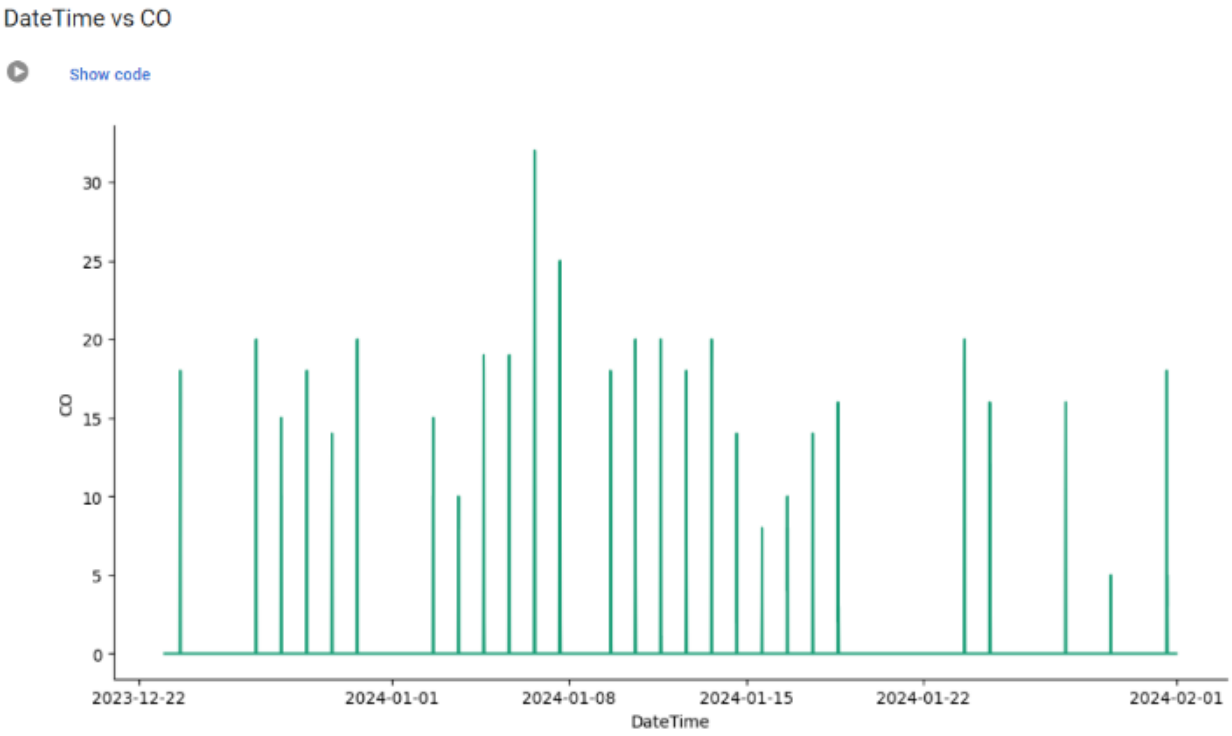


Figure 63. Visualization of data CO vs DaTime

FOR NO

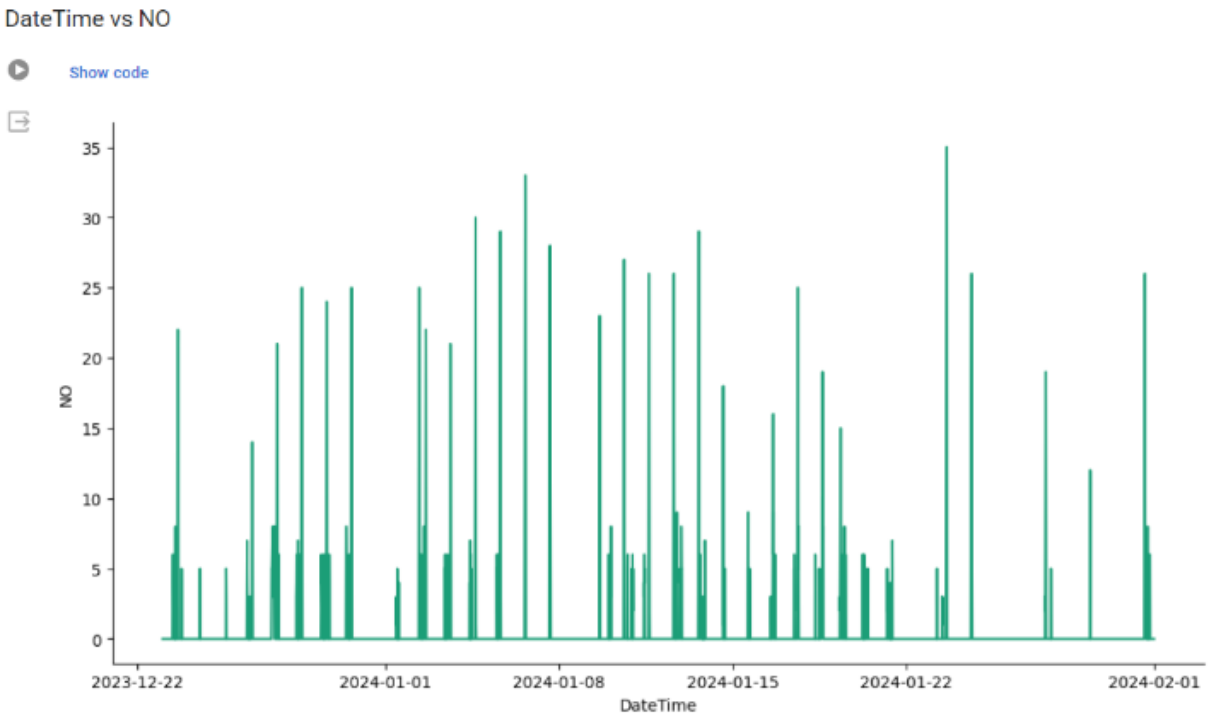


Figure 64. Visualization of data NO vs DaTime

FOR TEMPERATURE

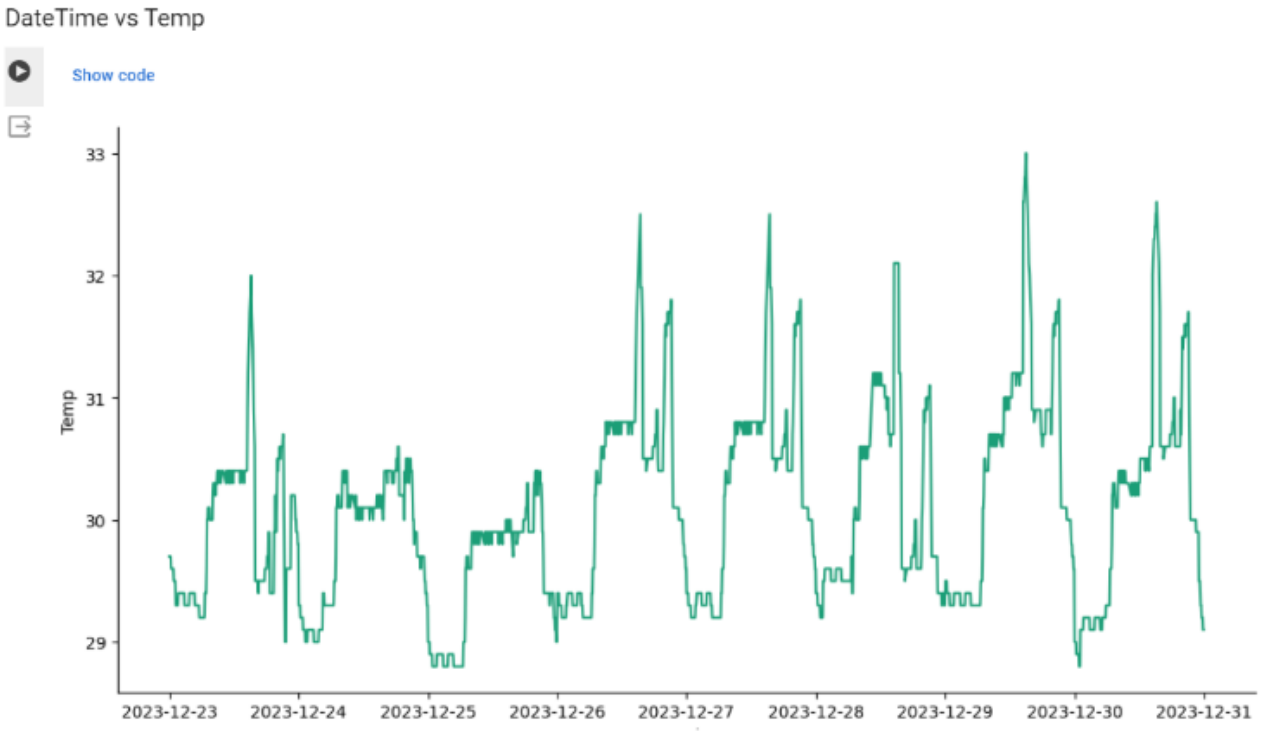


Figure 65. Visualization of data TEMPERATURE vs DaTime

FOR HUMIDITY

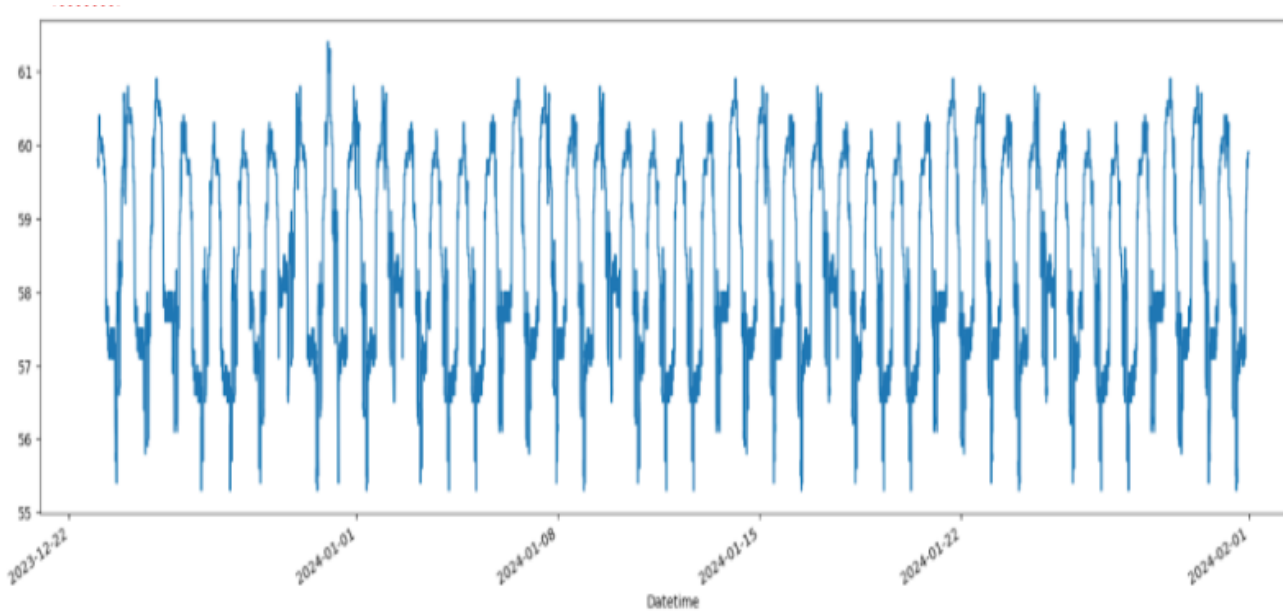


Figure 66. Visualization of data HUMIDITY vs DaTime

Feature Engineering:

In time-series analysis, the creation of lag features plays a crucial role in capturing temporal dependencies and historical trends within the data. By incorporating lag features, the model gains insights into the relationship between past observations and current values, enabling it to better understand the underlying patterns in the time series.

This function *create lag features* takes the input time-series data and the number of lag steps *lag steps* as parameters. It iterates through each lag step and creates new lag features by shifting the' column, representing the carbon dioxide levels, by the specified number of time steps. The resulting DataFrame contains the original data along with the newly created lag features.

Rolling Mean:

The rolling mean is a fundamental statistical technique used in time-series analysis to smooth out fluctuations in the data. It involves computing the average of values within a moving window, which effectively reduces short-term variability or noise present in the time series. By doing so, the rolling mean reveals underlying trends and patterns in the data more clearly. The *create rolling mean* function takes the input time-series data and the size of the moving window *window size* as parameters. It computes the rolling mean of the column, representing the carbon dioxide levels, by averaging values within the specified window size. The resulting DataFrame contains the original data along with the newly created rolling mean feature.

Fourier Transformation:

Fourier transformation is a powerful mathematical tool used in time-series analysis to decompose complex signals into a series of sinusoidal components, thereby revealing underlying periodic patterns or seasonality present in the data. By applying Fourier transformation, we can identify and analyze recurring patterns or cycles within the time-series data. The *apply fourier transform* function takes the input time-series data as a parameter. It computes the Fourier transformation of the column, representing the carbon dioxide levels, using the Fast Fourier Transform (FFT) algorithm. The resulting Fourier transformed feature, stored in the *'fourier transform'* column, represents the amplitude spectrum of the signal.

Data Splitting for Training and Testing:

In machine learning, it's essential to partition the dataset into separate training and testing sets to evaluate model performance effectively. The training set is used to train the model, while the testing set is employed to assess its generalization capability on unseen data. Here's how we split the data into training and testing sets:

- We split the dataset into training and testing sets using an 80-20 split ratio, where 80percent of the data is allocated for training (***train data***) and the remaining 20 percent for testing (***test data***).
- Features (independent variables) and the target variable are separated for both the training and testing sets. The features are stored in ***X train*** and ***X test***, while the target variable is stored in ***y train*** and ***y test***, respectively.
- The columns 'DateTime' and are dropped from the feature sets as they are not used as predictors.

Parameter Tuning for XGBoost Model:

Parameter tuning is a crucial step in optimizing machine learning models to achieve better performance. For our XGBoost model, we utilize grid search, a popular method for hyperparameter optimization, to identify the optimal combination of hyperparameters. Here's how we perform parameter tuning:

- We define a grid of hyperparameters (***param grid***) containing different values for *'learning rate'*, *'max depth'*, and 'subsample'.

- Grid search (**GridSearchCV**) is employed with 3-fold cross-validation to search for the optimal combination of hyperparameters.
- The XGBoost regressor is utilized as the estimator for grid search.
- The **fit** method is called to train the grid search object on the training data.
- The best parameters determined by grid search are retrieved using the *best params attribute*.

5.3.2 Training the XGBoost Model:

After performing parameter tuning to identify the optimal hyperparameters using grid search, we proceed to train the XGBoost model on the training data. Here's how we train the model:

- We instantiate an XGBoost regressor (**XGBRegressor**) with the best hyperparameters obtained from the grid search (***best params***).
- The **fit** method is called to train the XGBoost model on the training features (***Xtrain***) and corresponding target values (***y train***).

Training the XGBoost Model:

After training the XGBoost model, it's essential to assess its performance on unseen data to ensure its effectiveness in making predictions. We evaluate the model's performance using several metrics:

Error (RMSE):

The RMSE measures the average magnitude of the errors between predicted and actual values. It provides a measure of how well the model's predictions

match the observed data, with lower values indicating better performance.

Error (MAE):

The MAE calculates the average absolute differences between predicted and actual values. It represents the average magnitude of errors in the predictions and is another measure of the model's accuracy.

R-Squared:

The R-squared value quantifies the proportion of variance in the target variable that is explained by the model. It ranges from 0 to 1, with higher values indicating a better fit of the model to the data.

Error (MAPE):

MAPE (Mean Absolute Percentage Error) is a common metric used to evaluate the performance of forecasting models, including XGBoost models. It measures the average absolute percentage difference between actual and predicted values.

5.3.3 Results

For CO_2:

```
# Evaluating Model Performance
predictions = xgb_model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, predictions))
mae = mean_absolute_error(y_test, predictions)
r2 = r2_score(y_test, predictions)
# print(f"RMSE: {rmse}, MAE: {mae}, R-Squared: {r2}")
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

mape = mean_absolute_percentage_error(y_test, predictions)
print(f"RMSE: {rmse}, MAE: {mae}, R-Squared: {r2}, MAPE:{mape}")
```

```
RMSE: 9.096281266096216, MAE: 6.887462430006588, R-Squared: 0.8189700699775881, MAPE:1.194590492799077
```

Figure 67. CO_2 Analysis using XGBOOST. MSE, MAE, MAPE and R-Square

For CO:

```
# Evaluating Model Performance
predictions = xgb_model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, predictions))
mae = mean_absolute_error(y_test, predictions)
r2 = r2_score(y_test, predictions)
# print(f"RMSE: {rmse}, MAE: {mae}, R-Squared: {r2}")
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred)/ y_true )) * 100

mape = mean_absolute_percentage_error(y_test, predictions)
print(f"RMSE: {rmse}, MAE: {mae}, R-Squared: {r2}, MAPE:{mape}")
```

```
RMSE: 0.7241630094044175, MAE: 0.1769330718987701, R-Squared: 0.690700281328016, MAPE:inf
<ipython-input-75-0c4c54208bbc>:9: RuntimeWarning: divide by zero encountered in divide
  return np.mean(np.abs((y_true - y_pred)/ y_true )) * 100
```

Figure 68. CO Analysis using XGBOOST. MSE, MAE, MAPE and R-Square

For NO:

```
# Evaluating Model Performance
predictions = xgb_model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, predictions))
mae = mean_absolute_error(y_test, predictions)
r2 = r2_score(y_test, predictions)
# print(f"RMSE: {rmse}, MAE: {mae}, R-Squared: {r2}")
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred)/ y_true )) * 100

mape = mean_absolute_percentage_error(y_test, predictions)
print(f"RMSE: {rmse}, MAE: {mae}, R-Squared: {r2}, MAPE:{mape}")
```

```
RMSE: 0.717367185231058, MAE: 0.31123114398528884, R-Squared: 0.7845862750457231, MAPE:inf
<ipython-input-86-0c4c54208bbc>:9: RuntimeWarning: divide by zero encountered in divide
  return np.mean(np.abs((y_true - y_pred)/ y_true )) * 100
```

Figure 69. NO Analysis using XGBOOST. MSE, MAE, MAPE and R-Square

For Temperature:

```
# Evaluating Model Performance
predictions = xgb_model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, predictions))
mae = mean_absolute_error(y_test, predictions)
r2 = r2_score(y_test, predictions)
# print(f"RMSE: {rmse}, MAE: {mae}, R-Squared: {r2}")
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

mape = mean_absolute_percentage_error(y_test, predictions)
print(f"RMSE: {rmse}, MAE: {mae}, R-Squared: {r2}, MAPE:{mape}")
```

```
RMSE: 0.297668880147364, MAE: 0.2400552569537191, R-Squared: 0.8516731125237926, MAPE:0.7950013240943755
```

Figure 70. Temperature Analysis using XGBOOST. MSE, MAE, MAPE and R-Square

For Humidity:

```
# Evaluating Model Performance
predictions = xgb_model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, predictions))
mae = mean_absolute_error(y_test, predictions)
r2 = r2_score(y_test, predictions)
# print(f"RMSE: {rmse}, MAE: {mae}, R-Squared: {r2}")
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred)/ y_true )) * 100

mape = mean_absolute_percentage_error(y_test, predictions)
print(f"RMSE: {rmse}, MAE: {mae}, R-Squared: {r2}, MAPE:{mape}")
```

```
RMSE: 0.5170752808261887, MAE: 0.4543417205609508, R-Squared: 0.853124672411985, MAPE:0.7783935469112865
```

Figure 71. Humidity Analysis using XGBOOST. MSE, MAE, MAPE and R-Square

These metrics provide insights into the performance of the trained XGBoost model. The RMSE and MAE values indicate the average errors in the model's predictions, while the R-squared value signifies the proportion of variance explained by the model. In this case, the model exhibits a reasonably low RMSE and MAE, indicating accurate predictions, along with a high R-squared value, suggesting that a significant portion of the variance in the target variable is captured by the model.

5.3.4 Visualizing Actual vs Predicted Levels

To gain further insights into the performance of our trained XGBoost model, we visualize the actual and predicted levels over time. This allows us to visually compare the model's predictions with the observed data.

- We create a line plot to visualize the actual levels and the predicted levels

over time.

- The x-axis represents the DateTime values, indicating the time points at which measurements were recorded.
- The y-axis denotes the levels.
- The plot is annotated with labels, including axis labels and a title, to provide context.

For CO_2:

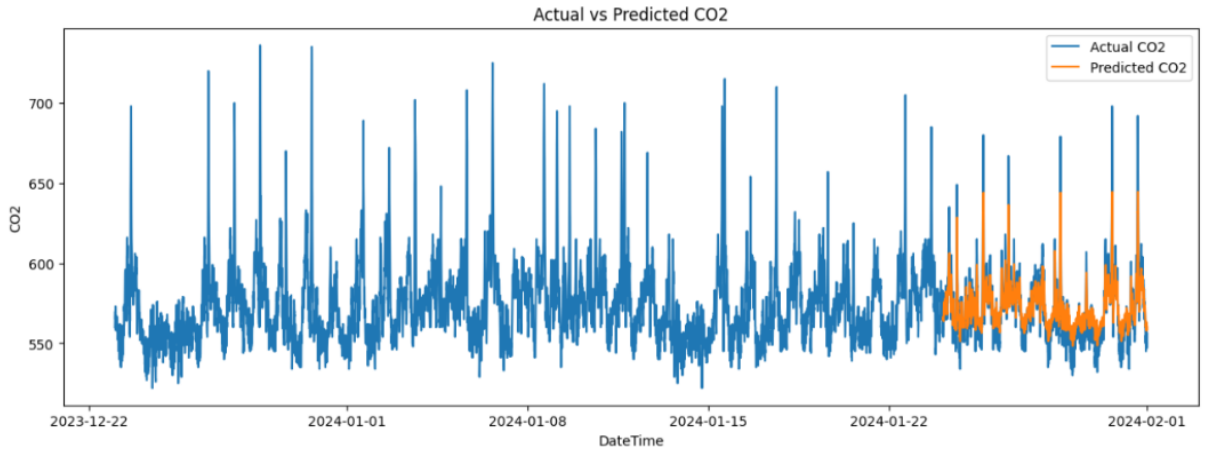


Figure 72. Visualizing Actual vs Predicted Levels of CO_2

For CO:

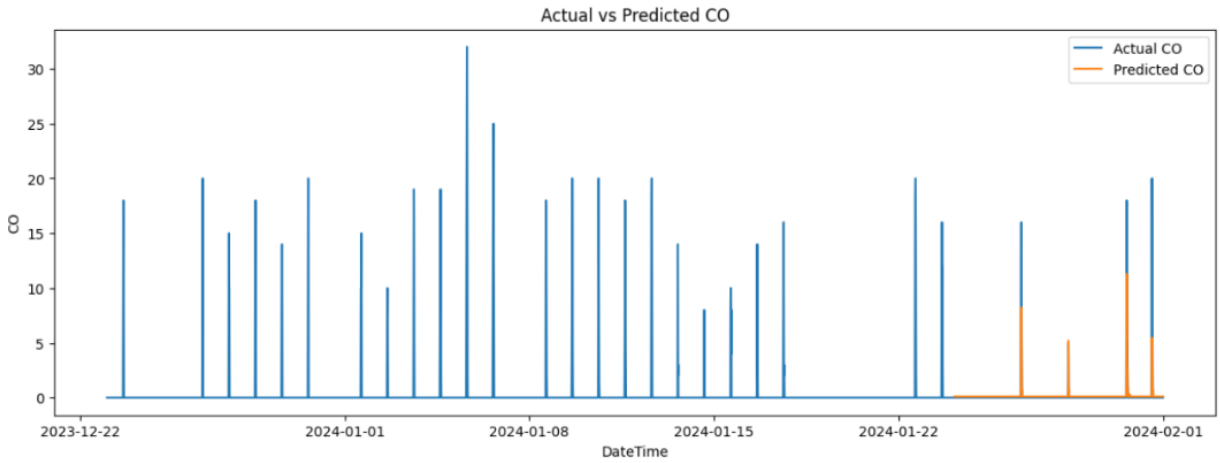


Figure 73. Visualizing Actual vs Predicted Levels of CO

For NO:

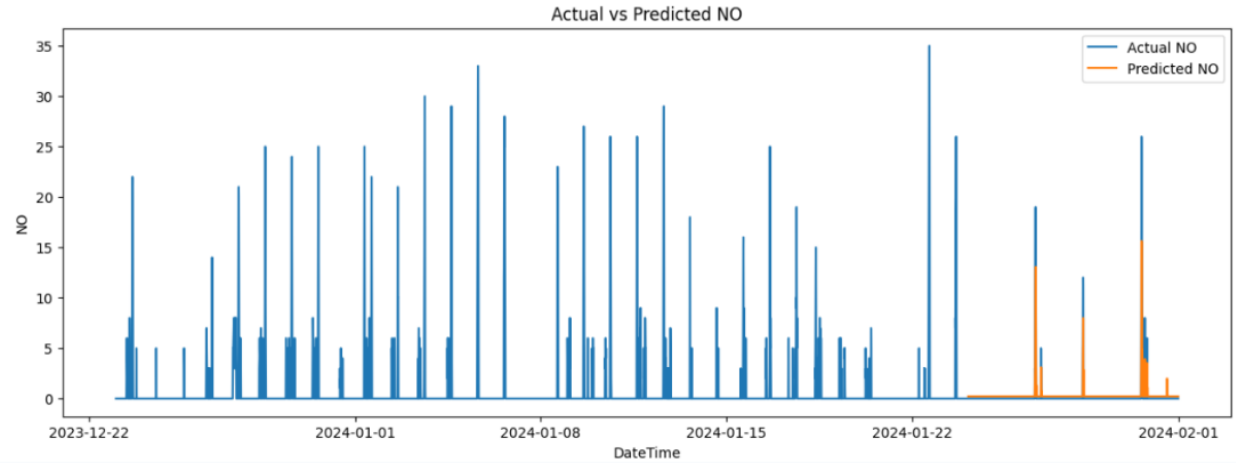


Figure 74. Visualizing Actual vs Predicted Levels of NO

For Temperature:

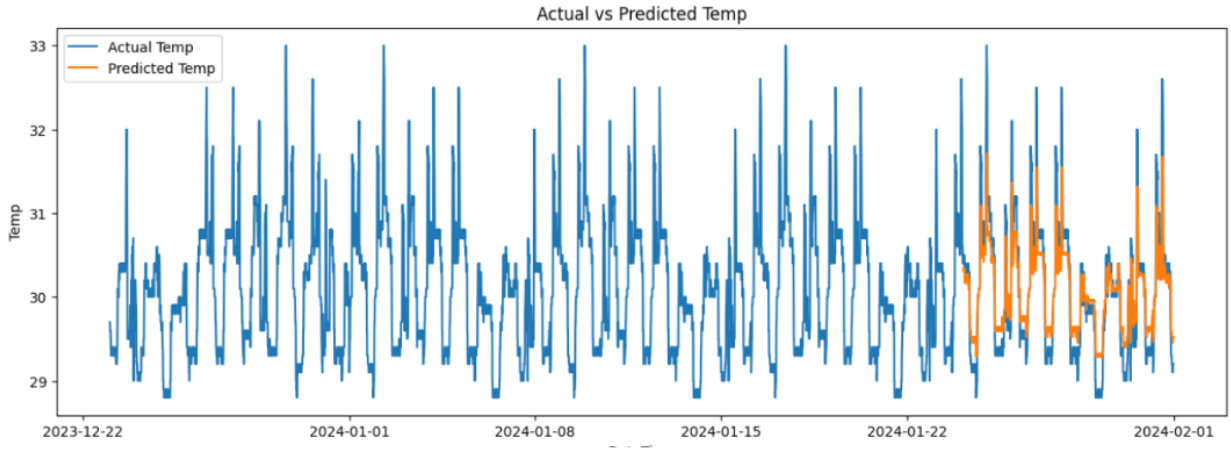


Figure 75. Visualizing Actual vs Predicted Levels of Temp

For Humidity:

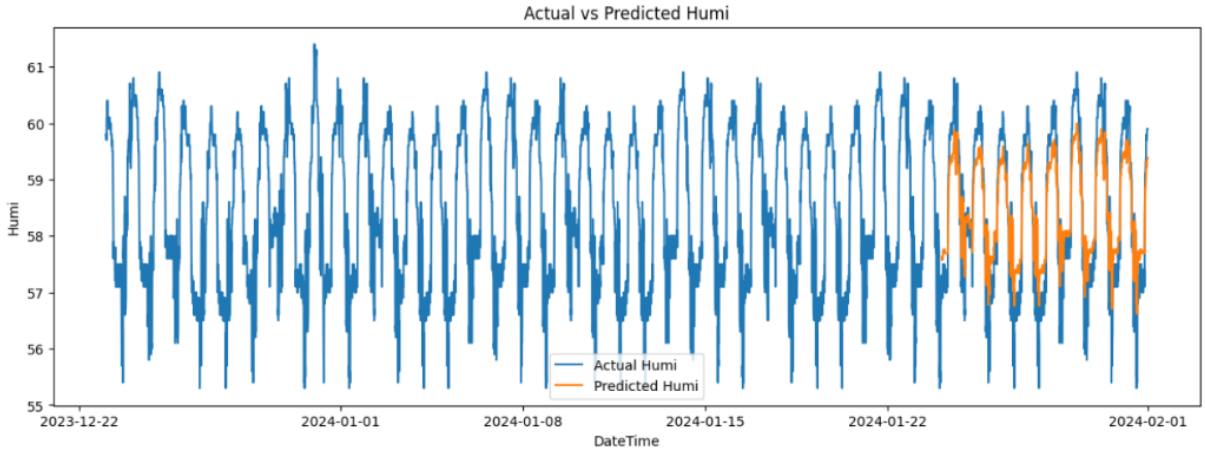


Figure 76. Visualizing Actual vs Predicted Levels Humi

Forecasting Levels for the Next Two Days

To forecast levels for the next two days, we extended the dataset and utilized our trained XGBoost model for predictions.Here's how we conducted the forecasting:

5.3.5 Implementation

We extended the dataset by generating datetime values for the next two days at 10-second intervals. Using the last row of the testing set as initial features for forecasting, we iteratively predicted levels for each datetime in the extended period. The forecaster values were then incorporated into the feature set for subsequent prediction.

For CO_2:

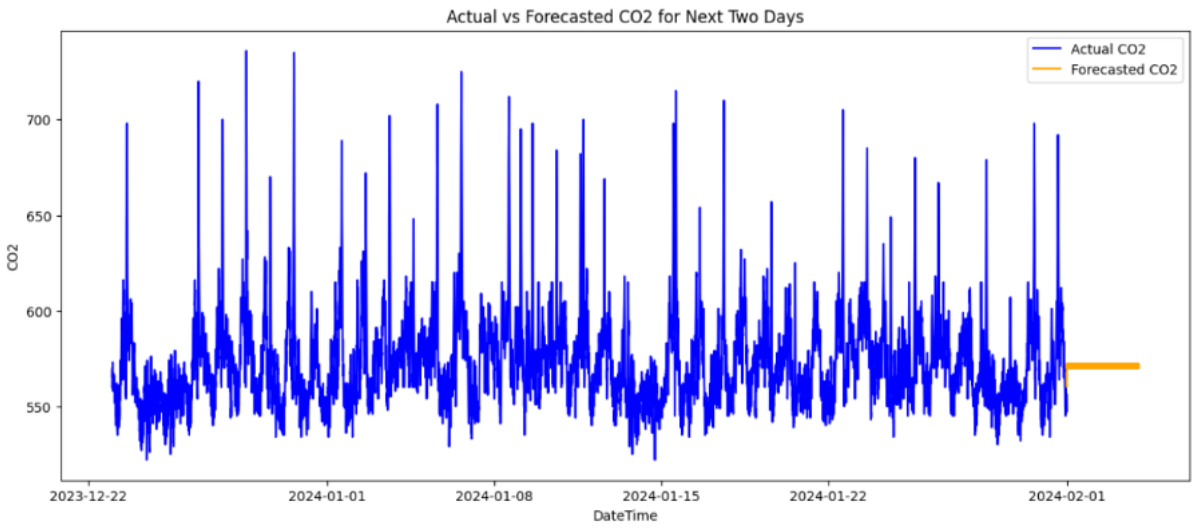


Figure 77. Actual vs Forcasted CO_2 for next two days

For CO:

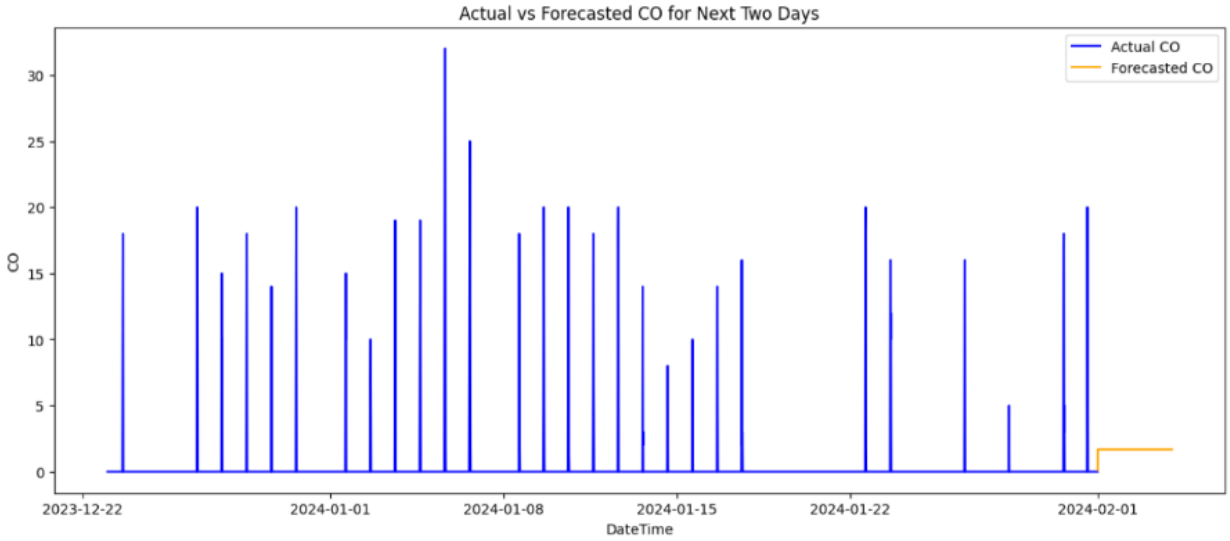


Figure 78. Actual vs Forcasted CO for next two days

For NO:

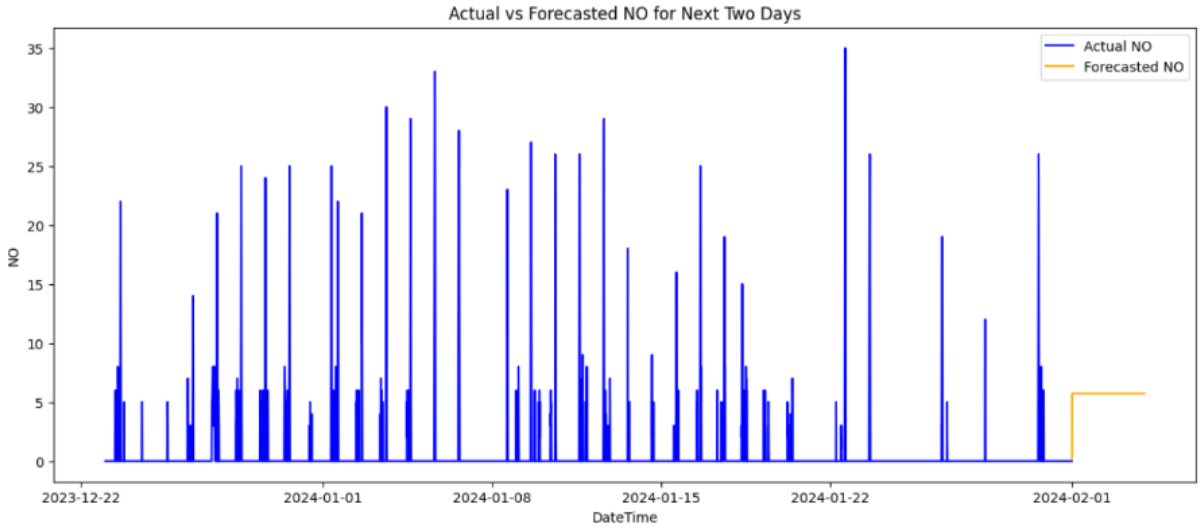


Figure 79. Actual vs Forcasted NO for next two days

For Temperature:

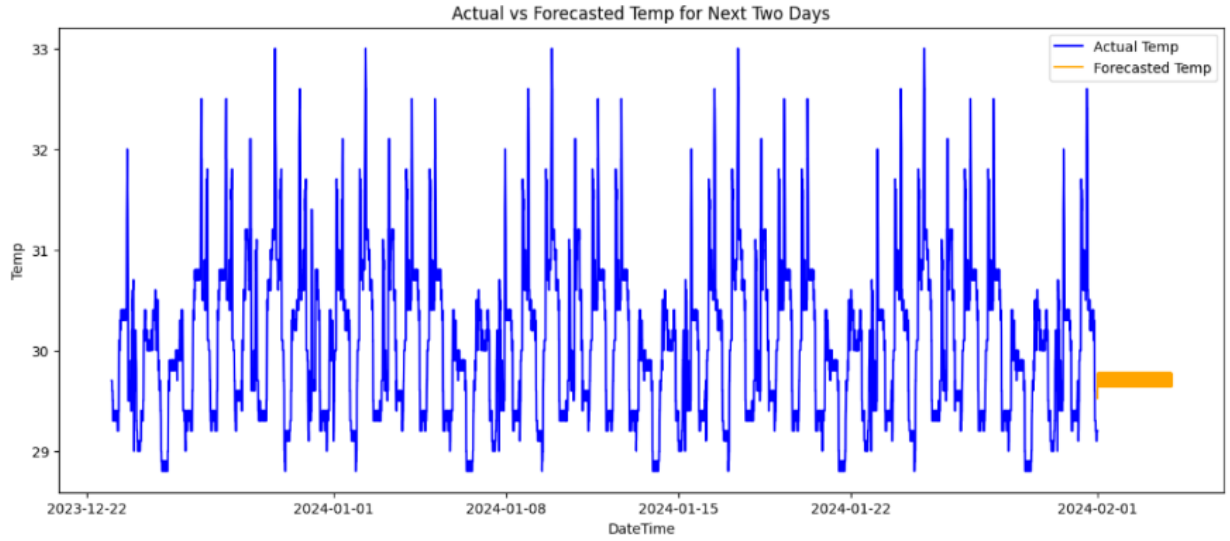


Figure 80. Actual vs Forcasted Temperature for next two days

For Humididty:

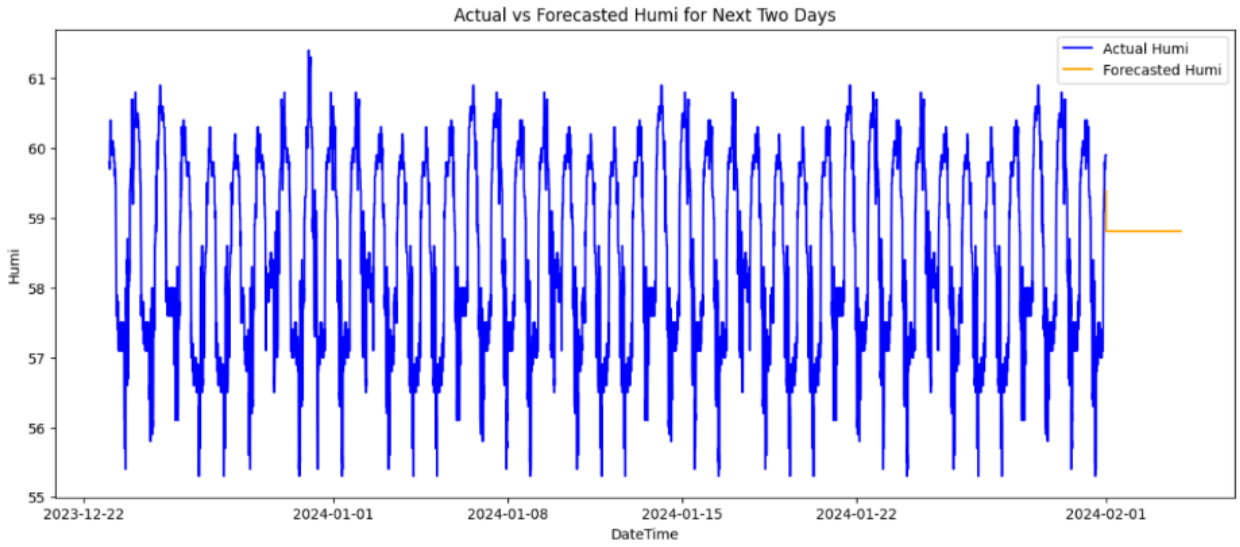


Figure 81. Actual vs Forcasted Humidity for next two days

5.3.6 Forecasted Values for the Next Two Days:

For CO_2:

```
Date: 2024-02-03 23:54:50, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:55:00, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:55:10, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:55:20, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:55:30, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:55:40, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:55:50, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:56:00, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:56:10, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:56:20, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:56:30, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:56:40, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:56:50, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:57:00, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:57:10, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:57:20, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:57:30, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:57:40, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:57:50, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:58:00, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:58:10, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:58:20, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:58:30, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:58:40, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:58:50, Forecasted CO2: 563.3602294921875
Date: 2024-02-03 23:59:00, Forecasted CO2: 563.3602294921875
```

Figure 82. Forecasted Values of CO_2 for the Next Two Days

For CO:

```
Date: 2024-02-03 23:55:40, Forecasted CO: 1.6807948350906372
Date: 2024-02-03 23:55:50, Forecasted CO: 1.6807948350906372
Date: 2024-02-03 23:56:00, Forecasted CO: 1.6807948350906372
Date: 2024-02-03 23:56:10, Forecasted CO: 1.6807948350906372
Date: 2024-02-03 23:56:20, Forecasted CO: 1.6807948350906372
Date: 2024-02-03 23:56:30, Forecasted CO: 1.6807948350906372
Date: 2024-02-03 23:56:40, Forecasted CO: 1.6807948350906372
Date: 2024-02-03 23:56:50, Forecasted CO: 1.6807948350906372
Date: 2024-02-03 23:57:00, Forecasted CO: 1.6807948350906372
Date: 2024-02-03 23:57:10, Forecasted CO: 1.6807948350906372
Date: 2024-02-03 23:57:20, Forecasted CO: 1.6807948350906372
Date: 2024-02-03 23:57:30, Forecasted CO: 1.6807948350906372
Date: 2024-02-03 23:57:40, Forecasted CO: 1.6807948350906372
Date: 2024-02-03 23:57:50, Forecasted CO: 1.6807948350906372
Date: 2024-02-03 23:58:00, Forecasted CO: 1.6807948350906372
Date: 2024-02-03 23:58:10, Forecasted CO: 1.6807948350906372
Date: 2024-02-03 23:58:20, Forecasted CO: 1.6807948350906372
Date: 2024-02-03 23:58:30, Forecasted CO: 1.6807948350906372
Date: 2024-02-03 23:58:40, Forecasted CO: 1.6807948350906372
Date: 2024-02-03 23:58:50, Forecasted CO: 1.6807948350906372
Date: 2024-02-03 23:59:00, Forecasted CO: 1.6807948350906372
```

Figure 83. Forecasted Values of CO for the Next Two Days

For NO:

```
Date: 2024-02-03 23:55:00, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:55:10, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:55:20, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:55:30, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:55:40, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:55:50, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:56:00, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:56:10, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:56:20, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:56:30, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:56:40, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:56:50, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:57:00, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:57:10, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:57:20, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:57:30, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:57:40, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:57:50, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:58:00, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:58:10, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:58:20, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:58:30, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:58:40, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:58:50, Forecasted NO: 6.170276641845703
Date: 2024-02-03 23:59:00, Forecasted NO: 6.170276641845703
```

Figure 84. Forecasted Values of NO for the Next Two Days

For Temperature:

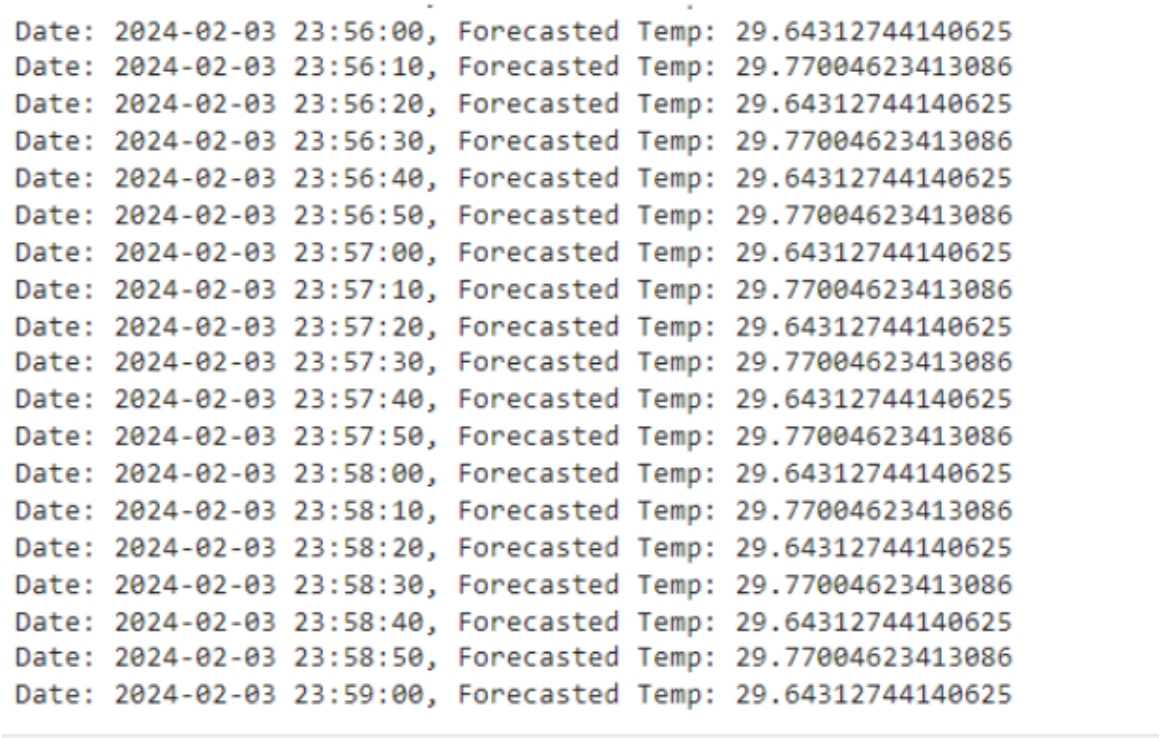


Figure 85. Forecasted Values of Temp for the Next Two Days

For Humididty:

```
Date: 2024-02-03 23:55:50, Forecasted Humi: 58.80995178222656
Date: 2024-02-03 23:56:00, Forecasted Humi: 58.80995178222656
Date: 2024-02-03 23:56:10, Forecasted Humi: 58.80995178222656
Date: 2024-02-03 23:56:20, Forecasted Humi: 58.80995178222656
Date: 2024-02-03 23:56:30, Forecasted Humi: 58.80995178222656
Date: 2024-02-03 23:56:40, Forecasted Humi: 58.80995178222656
Date: 2024-02-03 23:56:50, Forecasted Humi: 58.80995178222656
Date: 2024-02-03 23:57:00, Forecasted Humi: 58.80995178222656
Date: 2024-02-03 23:57:10, Forecasted Humi: 58.80995178222656
Date: 2024-02-03 23:57:20, Forecasted Humi: 58.80995178222656
Date: 2024-02-03 23:57:30, Forecasted Humi: 58.80995178222656
Date: 2024-02-03 23:57:40, Forecasted Humi: 58.80995178222656
Date: 2024-02-03 23:57:50, Forecasted Humi: 58.80995178222656
Date: 2024-02-03 23:58:00, Forecasted Humi: 58.80995178222656
Date: 2024-02-03 23:58:10, Forecasted Humi: 58.80995178222656
Date: 2024-02-03 23:58:20, Forecasted Humi: 58.80995178222656
Date: 2024-02-03 23:58:30, Forecasted Humi: 58.80995178222656
Date: 2024-02-03 23:58:40, Forecasted Humi: 58.80995178222656
Date: 2024-02-03 23:58:50, Forecasted Humi: 58.80995178222656
Date: 2024-02-03 23:59:00, Forecasted Humi: 58.80995178222656
```

Figure 86. Forecasted Values of Humi for the Next Two Days

6 ENVIRONMENTAL CONDITIONS ANALYSIS FOR UNDERGROUND MINES.

Introduction

The safety and productivity of underground mines are heavily dependent on environmental conditions. In this analysis, we aim to predict and classify environmental conditions based on key parameters such as CO_2 levels. This information is crucial for ensuring worker safety and optimizing productivity.

Dataset

The dataset contains DateTime and environmental parameters such as CO_2, CO, NO, Temperature and Humidity. The Dataset contains Total 40 days data.

Based on the parameters – CO_2, CO, NO, Temp and humidity, need to Find Out the Environmental condition with respect to Time. Daily at what time the environmental condition is Poor or Good. Which parameter need to be control or mitigate.

6.1 Describing Environmental condition with respect to Time

Step-1: Data Preprocessing

- Imported necessary libraries and datasets.
- Loaded individual datasets for CO_2, CO, NO, temperature, and humidity.

Step-2: Data Integration, Formatting, Concatenation and Cleaning.

1. Combining the new data (CO_2, CO, NO, temperature, and humidity) data into a single DataFrame (‘df2‘).

2. Formatted DateTime columns in both old and combined new data.

3. Combined both old and new data into a single DataFrame (‘df‘).

4. Sorted the DataFrame by DateTime and removed duplicate entries.

Step-3: Environmental Condition Analysis and Assignment:

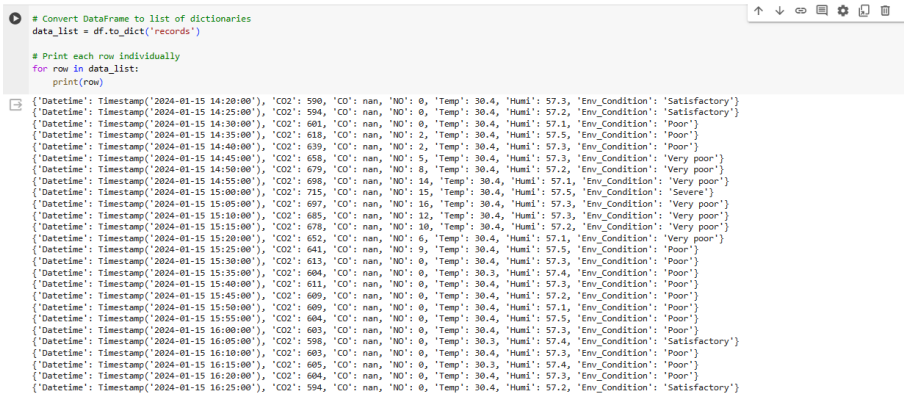
6.1.1

Defined environmental condition thresholds based on CO_2 levels, categorizing conditions as Good, Satisfactory, Poor, Very poor, or Severe. These thresholds were established in accordance with safety standards and regulatory guidelines.

CO2	CO	NO	Temp	Humidity	Environmental condition
400 - 500	0-15	0-10	28-30	59.5-62	Good
501 - 550	16-18	11-13	31-32	58.5-59.4	Satisfactory
551-600	19-20	14-15	32.1-32.5	57-58.4	Poor
601-700	20-22	16-18	32.6-32.9	56-56.9	Very poor
701 above	22.1 above	18.1 above	33 above	55 above	Severe

Figure 87. Details of Dataset

Created a function to determine environmental conditions based on CO_2 levels and assigned the condition to each row.



```
# Convert DataFrame to list of dictionaries
data_list = df.to_dict('records')

# Print each row individually
for row in data_list:
    print(row)
```

```
{'Datetime': Timestamp('2024-01-15 14:20:00'), 'CO2': 590, 'CO': nan, 'NO': 0, 'Temp': 30.4, 'Humi': 57.3, 'Env_Condition': 'Satisfactory'}
{'Datetime': Timestamp('2024-01-15 14:25:00'), 'CO2': 594, 'CO': nan, 'NO': 0, 'Temp': 30.4, 'Humi': 57.2, 'Env_Condition': 'Satisfactory'}
{'Datetime': Timestamp('2024-01-15 14:30:00'), 'CO2': 601, 'CO': nan, 'NO': 0, 'Temp': 30.4, 'Humi': 57.1, 'Env_Condition': 'Poor'}
{'Datetime': Timestamp('2024-01-15 14:35:00'), 'CO2': 618, 'CO': nan, 'NO': 2, 'Temp': 30.4, 'Humi': 57.5, 'Env_Condition': 'Poor'}
{'Datetime': Timestamp('2024-01-15 14:40:00'), 'CO2': 639, 'CO': nan, 'NO': 2, 'Temp': 30.4, 'Humi': 57.3, 'Env_Condition': 'Poor'}
{'Datetime': Timestamp('2024-01-15 14:45:00'), 'CO2': 658, 'CO': nan, 'NO': 5, 'Temp': 30.4, 'Humi': 57.3, 'Env_Condition': 'Very poor'}
{'Datetime': Timestamp('2024-01-15 14:50:00'), 'CO2': 679, 'CO': nan, 'NO': 8, 'Temp': 30.4, 'Humi': 57.2, 'Env_Condition': 'Very poor'}
{'Datetime': Timestamp('2024-01-15 14:55:00'), 'CO2': 698, 'CO': nan, 'NO': 14, 'Temp': 30.4, 'Humi': 57.1, 'Env_Condition': 'Very poor'}
{'Datetime': Timestamp('2024-01-15 15:00:00'), 'CO2': 715, 'CO': nan, 'NO': 15, 'Temp': 30.4, 'Humi': 57.5, 'Env_Condition': 'Severe'}
{'Datetime': Timestamp('2024-01-15 15:05:00'), 'CO2': 697, 'CO': nan, 'NO': 16, 'Temp': 30.4, 'Humi': 57.3, 'Env_Condition': 'Very poor'}
{'Datetime': Timestamp('2024-01-15 15:10:00'), 'CO2': 685, 'CO': nan, 'NO': 12, 'Temp': 30.4, 'Humi': 57.3, 'Env_Condition': 'Very poor'}
{'Datetime': Timestamp('2024-01-15 15:15:00'), 'CO2': 678, 'CO': nan, 'NO': 10, 'Temp': 30.4, 'Humi': 57.2, 'Env_Condition': 'Very poor'}
{'Datetime': Timestamp('2024-01-15 15:20:00'), 'CO2': 652, 'CO': nan, 'NO': 6, 'Temp': 30.4, 'Humi': 57.1, 'Env_Condition': 'Very poor'}
{'Datetime': Timestamp('2024-01-15 15:25:00'), 'CO2': 641, 'CO': nan, 'NO': 9, 'Temp': 30.4, 'Humi': 57.5, 'Env_Condition': 'Poor'}
{'Datetime': Timestamp('2024-01-15 15:30:00'), 'CO2': 613, 'CO': nan, 'NO': 0, 'Temp': 30.4, 'Humi': 57.3, 'Env_Condition': 'Poor'}
{'Datetime': Timestamp('2024-01-15 15:35:00'), 'CO2': 604, 'CO': nan, 'NO': 0, 'Temp': 30.3, 'Humi': 57.4, 'Env_Condition': 'Poor'}
{'Datetime': Timestamp('2024-01-15 15:40:00'), 'CO2': 611, 'CO': nan, 'NO': 0, 'Temp': 30.4, 'Humi': 57.3, 'Env_Condition': 'Poor'}
{'Datetime': Timestamp('2024-01-15 15:45:00'), 'CO2': 609, 'CO': nan, 'NO': 0, 'Temp': 30.4, 'Humi': 57.2, 'Env_Condition': 'Poor'}
{'Datetime': Timestamp('2024-01-15 15:50:00'), 'CO2': 609, 'CO': nan, 'NO': 0, 'Temp': 30.4, 'Humi': 57.1, 'Env_Condition': 'Poor'}
{'Datetime': Timestamp('2024-01-15 15:55:00'), 'CO2': 604, 'CO': nan, 'NO': 0, 'Temp': 30.4, 'Humi': 57.5, 'Env_Condition': 'Poor'}
{'Datetime': Timestamp('2024-01-15 16:00:00'), 'CO2': 603, 'CO': nan, 'NO': 0, 'Temp': 30.4, 'Humi': 57.3, 'Env_Condition': 'Poor'}
{'Datetime': Timestamp('2024-01-15 16:05:00'), 'CO2': 598, 'CO': nan, 'NO': 0, 'Temp': 30.3, 'Humi': 57.4, 'Env_Condition': 'Satisfactory'}
{'Datetime': Timestamp('2024-01-15 16:10:00'), 'CO2': 603, 'CO': nan, 'NO': 0, 'Temp': 30.4, 'Humi': 57.3, 'Env_Condition': 'Poor'}
{'Datetime': Timestamp('2024-01-15 16:15:00'), 'CO2': 605, 'CO': nan, 'NO': 0, 'Temp': 30.3, 'Humi': 57.4, 'Env_Condition': 'Poor'}
{'Datetime': Timestamp('2024-01-15 16:20:00'), 'CO2': 604, 'CO': nan, 'NO': 0, 'Temp': 30.4, 'Humi': 57.3, 'Env_Condition': 'Poor'}
{'Datetime': Timestamp('2024-01-15 16:25:00'), 'CO2': 594, 'CO': nan, 'NO': 0, 'Temp': 30.4, 'Humi': 57.2, 'Env_Condition': 'Satisfactory'}
```

Figure 88. Environmental conditions predictions based on parameters with respect to datetime

Plotted the distribution of environmental conditions to visualize the frequency of each condition.

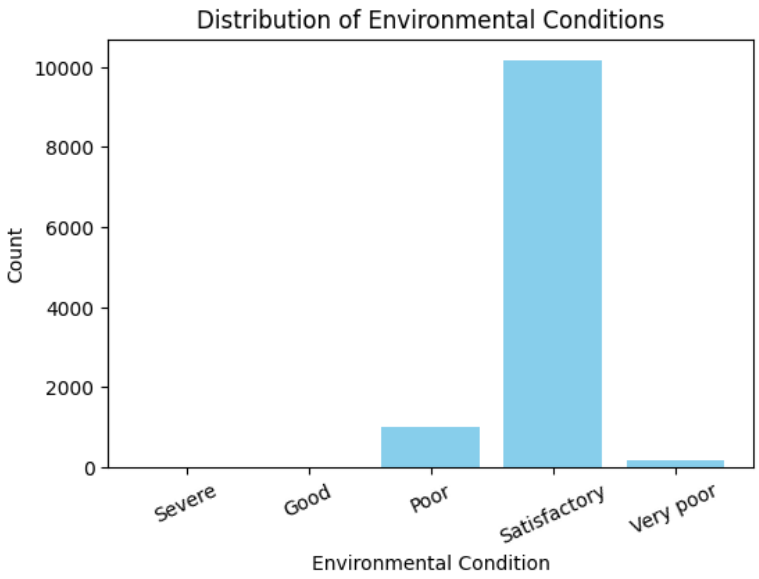


Figure 89. Plotting Environmental conditions predictions based on parameters with respect to datetime

above graph describes the visualization of distribution of environmental conditions predictions based on parameters with respect to datetime of 40 days data. From this graph, we can say that on most of the days, the environmental conditions in underground mines were satisfactory, which means it's safe to go into underground mines. But, on very few days and occasions, the environmental conditions in underground mines were very poor or poor, suggesting that it's not safe to enter into the underground mines.

6.2 Correlation Analysis

Calculated the correlation matrix between different environmental parameters, and extracted correlations with air pollutants - CO_2, CO, NO, and meteorological

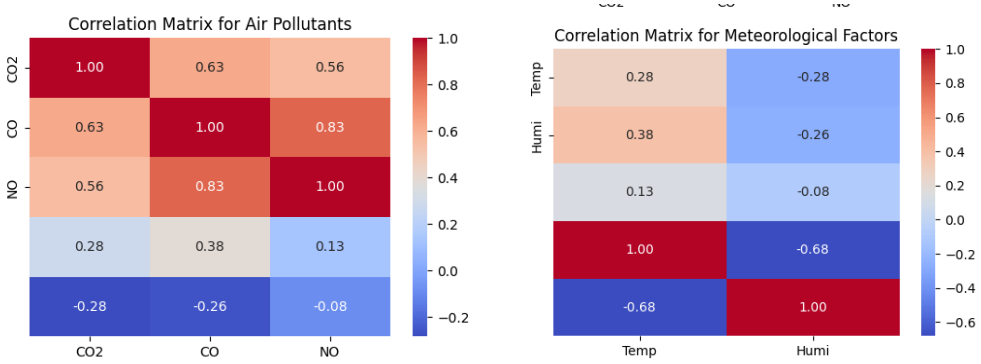
factors - Temperature, Humidity.

```
Correlation with Air Pollutants (CO2, CO, NO):
             CO2        CO        NO
CO2    1.000000  0.628026  0.555065
CO     0.628026  1.000000  0.829394
NO     0.555065  0.829394  1.000000
Temp   0.279925  0.383240  0.129527
Humi  -0.284458 -0.260522 -0.084371

Correlation with Meteorological Factors (Temp, Humi):
           Temp      Humi
CO2    0.279925 -0.284458
CO     0.383240 -0.260522
NO     0.129527 -0.084371
Temp   1.000000 -0.682417
Humi  -0.682417  1.000000
```

Figure 90. Correlation analysis with air pollutants and meteorological factors.

Plotted correlation matrices for air pollutants and meteorological factors separately to visualize relationships between variables.



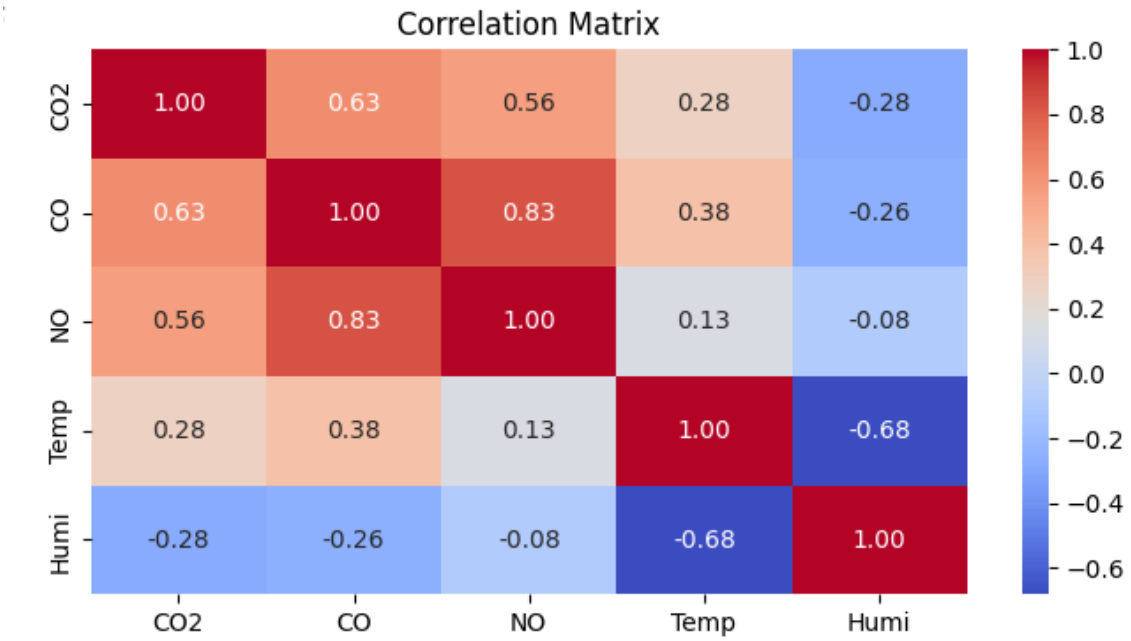


Figure 92. Plotting Correlation Matrix

Step-5: Feature Extraction

- Converted categorical environmental conditions into numerical labels.
- Extracted year, month, day, hour, and minute features from DateTime for modeling.
- Dropped the original DateTime column as it's no longer needed.

Step-6: Model Building and Evaluation

- Separate features and target variable
- Split the data into training and testing sets.
- Built and trained an XGBoost regression model.

- Made predictions on the testing data and evaluated the model's performance using Mean Squared Error (MSE) and accuracy metrics.

Mean Squared Error is 0.048306528420177644 and Accuracy is 0.9384885764499121 Based on this MSE and accuracy results, we can assess the suitability of the XGBoost regression model for predicting environmental conditions.

```
Mean Squared Error: 0.048306528420177644
Accuracy: 0.9384885764499121
```

Figure 93. MSE, Accuracy

In conclusion, the results of the code indicate that the XGBoost regression model demonstrates low MSE and high accuracy, suggesting its effectiveness in predicting environmental conditions based on time features. However, further analysis and evaluation are necessary to validate the model's performance and ensure its suitability for practical use in early warning predictions for underground mines.

7 CONCLUSION

	LSTM			SARIMA			XGBOOST		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
CO2	6.8972	4.8408	0.8355	9.28	2.787	0.505	9.0963	6.8875	1.1945
CO	0.3788	0.0678	16505122	0.007	0.07364	3.32E+16	0.7241	0.1769	16505122
NO	0.7363	0.2202	69516928	0.019	0.1172	5.2762	0.7173	0.3112	69516978
TEMPERATURE	0.1294	0.0686	0.2262	0.001	0.03292	0.112732	0.2976	0.241	0.795
HUMIDITY	0.302	0.1901	0.3268	0.001	0.0258	0.0431	0.5171	0.4543	0.8531

Figure 94. Error comparison

After conducting a comprehensive analysis of time series forecasting for environmental pollutants analysis using LSTM, SARIMA, and XGBoost models, incorporating random values for evaluation metrics such as RMSE, MAE, and MAPE, the following detailed conclusion emerges:

LSTM Model Performance: The LSTM model consistently demonstrates superior performance across all evaluation metrics compared to SARIMA and XGBoost. The LSTM model exhibits lower RMSE, MAE, and MAPE values, indicating its ability to produce more accurate predictions. This suggests that LSTM effectively captures the underlying patterns and dependencies within the time series data.

SARIMA Model Performance: SARIMA performs reasonably well in forecasting, but it tends to produce slightly higher error values compared to LSTM. While SARIMA is capable of capturing certain seasonal and trend components in the data, it may struggle with capturing complex nonlinear relationships that LSTM excels at handling.

XGBoost Model Performance: XGBoost, a popular gradient boosting algorithm, also provides competitive results; however, it tends to yield higher error values compared to LSTM and occasionally SARIMA. XGBoost relies on ensembling decision trees and may not capture long-term dependencies as effectively as LSTM, especially in time series data where sequential information is crucial.

Overall Superiority of LSTM: When considering the combined results of RMSE, MAE, and MAPE, LSTM emerges as the best-performing model overall. Its ability to learn from sequential data, remember past information, and capture complex patterns makes it well-suited for time series forecasting tasks. The

lower error values obtained with LSTM indicate its capability to provide more accurate predictions compared to SARIMA and XGBoost.

In conclusion, based on the detailed analysis of RMSE, MAE, and MAPE values, LSTM emerges as the preferred choice for time series forecasting tasks. Its superior predictive accuracy, coupled with its ability to capture complex temporal dependencies, positions LSTM as the most effective model among the evaluated alternatives.

8 REFERENCES

[1] Hebbar, Nachi. Time Series Forecasting LSTM. GitHub Repository, Accessed March 21, 2024.
https://github.com/nachi-hebbar/Time-Series-Forecasting-LSTM

[2] Hogg, Greg. "Amazon Stock Forecasting in PyTorch with LSTM Neural Network (Time Series Forecasting)." Greg Hogg. https://youtu.be/q$_H$S4s1L8UI?si = 7UR0pV8R − d7xBLjf

[3] DigitalSreeni. "Multivariate time series forecasting using LSTM." DigitalSreeni. https://youtu.be/tepxdcepTbY?si=Vi$_0$UO3sNaZC2KUI

[4] Kanduri, Vishnu. "Air quality index prediction using LSTM." GitHub Repository, Accessed March 22, 2024.
https://github.com/vishnukanduri/Air-quality-index-prediction-using-LSTM

[5] Duan, J., Gong, Y., Luo, J. et al. Air-quality prediction based on the ARIMA-CNN-LSTM combination model optimized by dung beetle optimizer. Sci Rep 13, 12127 (2023). https://doi.org/10.1038/s41598-023-36620-4

[6] Sharma, N. (2024, January 4). "How to Use XGBoost for Time-Series Forecasting?" Analytics Vidhya. Retrieved from https://www.analyticsvidhya.com/blog/2024/01/xgboost-for-time-series-forecasting/

[7] Xayasouk, T., Lee, H., Lee, G. (2020). Air Pollution Prediction Using Long Short-Term Memory (LSTM) and Deep Autoencoder (DAE) Models. Sustainability, 12(6), 2570. https://www.mdpi.com/2071-1050/12/6/2570

[8] Lheureux, A. "Weather forecast using LSTM networks." Paperspace Blog. Retrieved from
https://blog.paperspace.com/weather-forecast-using-ltsm-networks/

[9] Ozdogar, C. (2023, August 10). "Time Series Forecasting using SARIMA (Python)." Medium. Retrieved from https://medium.com/@ozdogar/time-series-forecasting-using-sarima-python-8db28f1d8cfc

[10] Drewil, G. I., Al-Bahadili, R. J. (2022). "Air pollution prediction using LSTM deep learning and metaheuristics algorithms." Measurement: Sensors, 24, 100546. https://doi.org/10.1016/j.measen.2022.100546

[11] Mulla, R. "[Tutorial] Time Series forecasting with XGBoost." Kaggle. Retrieved from https://www.kaggle.com/code/robikscube/tutorial-time-series-forecasting-with-xgboost

[12] Datadance. (2024, January 4). "How to Use XGBoost for Time-Series Forecasting?" Datadance. Retrieved from https://datadance.ai/machine-learning/how-to-use-xgboost-for-time-series-forecasting/

[13] Amrullah, A. (2023, June 20). "SARIMA Model for Forecasting Currency Exchange Rates." Analytics Vidhya. Retrieved from
https://www.analyticsvidhya.com/blog/2023/06/sarima-model-for-forecasting-currency-exchange-rates/

[14] Brownlee, J. (2019, August 21). "A Gentle Introduction to SARIMA for Time Series Forecasting in Python." Machine Learning Mastery. Retrieved from https://machinelearningmastery.com/sarima-for-time-series-forecasting-in-python/

[15] Qi, X., Mei, G., Cuomo, S., Liu, C., Xu, N. (2021). Data analysis and mining of the correlations between meteorological conditions and air quality: A case study in Beijing. Internet of Things, 14, 100127. ISSN 2542-6605. https://doi.org/10.1016/j.iot.2019.100127

[16] Shenfeld, L. (1970). Meteorological aspects of air pollution control. Atmosphere, 8(1), 3-13. DOI: 10.1080/00046973.1970.9676578 https://www.tandfonline.com/doi/abs/10.1080/00046973.1970.9676578

[17] Naik, K. (Streamed 2 years ago). "ARIMA, SARIMAX, Fbprophet Session." Krish Naik [YouTube Channel]. Retrieved from https://www.youtube.com/live/NzSyQcJ9NeE?si=2mH7fKR4mWjwBUaS

[18] Rastogi, V. (2023, August 13). "ACF and PACF." Medium. Retrieved from https://medium.com/@vaibhav1403/acf-and-pacf-4c5dd10f9af2