Monitoring & Predicting Changes in Carbon Dioxide Levels in Enclosed Social Spaces

Harshini Arava, Corey Dearing, Nick Young

April 29, 2024

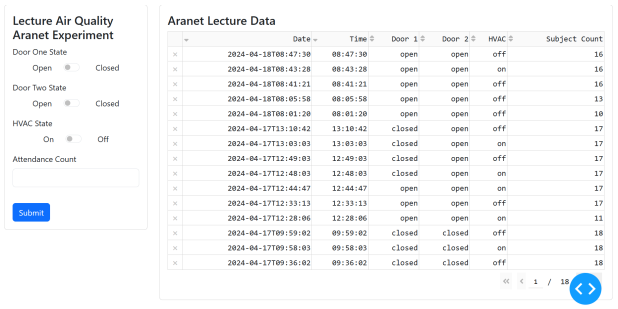
Abstract

Carbon dioxide (CO_2) is one of many gases often used as a standard for measuring how well ventilated an environment is. Studies have found (citation) that excess levels of CO_2 can lead to temporary declines in cognitive performance. Our study records numerous metrics and environment states by the minute across multiple lectures taken during three lecture windows given by host by two lecturers. We trained a Long Short-Term Memory neural network on a randomly sampled portion of the lectures and then use the unseen test samples to evaluate how well our model performs.

Introduction

Aranet4 Remote Sensor

Our initial motivation for this project was to determine how well we could forecast the remaining portion of a given lecture window based on the preceding minute-to-minute datetime sequence of metrics such as carbon dioxide (ppm), relative humidity (%), temperature(°F), and atmospheric pressure (hPa) recorded by an *Aranet4* remote sensor present in the room.



Lecture Air Quality Aranet Experiment

Door One State: Open / Closed

Door Two State: Open / Closed

HVAC State: On / Off

Attendance Count

Submit

Aranet Lecture Data

Date	Time	Door 1	Door 2	HVAC	Subject Count
2024-04-18T08:47:30	08:47:30	open	open	off	16
2024-04-18T08:43:28	08:43:28	open	open	on	16
2024-04-18T08:41:21	08:41:21	open	open	off	16
2024-04-18T08:05:58	08:05:58	open	open	off	13
2024-04-18T08:01:20	08:01:20	open	open	off	10
2024-04-17T13:10:42	13:10:42	closed	open	off	17
2024-04-17T13:03:03	13:03:03	closed	open	on	17
2024-04-17T12:49:03	12:49:03	closed	open	off	17
2024-04-17T12:48:03	12:48:03	closed	open	on	17
2024-04-17T12:44:47	12:44:47	open	open	on	17
2024-04-17T12:33:13	12:33:13	open	open	off	17
2024-04-17T12:28:06	12:28:06	open	open	on	11
2024-04-17T09:59:02	09:59:02	closed	closed	off	18
2024-04-17T09:58:03	09:58:03	closed	closed	on	18
2024-04-17T09:36:02	09:36:02	closed	closed	off	18

1 / 18

Figure 1: *Aranet4 Home* & Data Collection Dashboard

Experiment Data Collection Dashboard

We designed a simple web application with the *plotly dash* python API for an observer to utilize in order to record changes in state of the room such as datetime (lecturer), hvac (on/off), hallway door(1) (open/closed), study door(2) (open/closed), and number of occupants in the room to SQL database. Later we query and merge data from the *Aranet4* remote sensor to the experiment data (Figure 1).

In the methodology section we will discuss how we joined the sensor and lecture experiment data into lecture windows. For example, the April 17th, 2024 Professor Song lecture data gives a glimpses into one of the time-series windows that may be used training and testing our Long Short-Term Memory (LSTM) model for forecasting. The binary encoding representations are found in (Table 1).

Feature	**State**
Hallway (door1)	open = 1, closed = 0
Study (door2)	open = 1, closed = 0
HVAC	on = 1, off = 0
Lecturer	Song = 1, Chen = 0

Table 1: Description of binary state features.

In the April 17th lecture we see multiple changes in state occur all around the about the same point in time. We find the HVAC in the room was in an on state for roughly 5 minutes before turning off around the same moment the hallway door closes (and remains closed for the rest of the lecture). At the same time we find the sensor measures an increase in the CO_2 and rel. humidity while recording a decrease in the temperature until each reach either a max/min respectively. As we see temperature begin to rise the HVAC system again turns on again briefly at which point we see the environment possibly stabilizing for the remainder of the lecture. This sort of behavior inspired the design behind the steady-state monitoring dashboard as seen in (Figure 5). We explore these interactions a bit more in the PowerBI and CO_2 State Space sections.

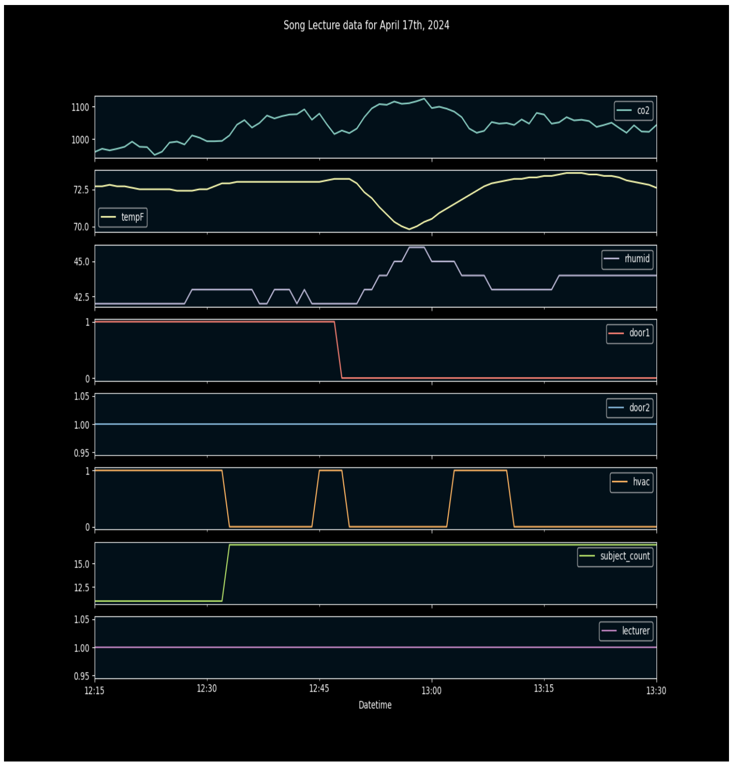


Figure 2: Lecture April 17th, 2024, Professor Song

PowerBI Dashboards

As part of our exploratory data analysis we built a Microsoft PowerBI Dashboard to explore the behavior of CO_2 throughout our lecture experiment with respect to other measured features. We find a story concerning the change in CO_2 across time both in terms of each lecture given the observed states and in terms the time of day. Professor Song's lectures occur at midday while both Professor Chen's lectures occur in the morning between 8 to 9:15 (Tuesday & Thursday) and 9 to 10:15am (Monday & Wednesday). We find through that the time of day leads to a considerable mean difference in CO_2 levels across morning to noon lecture times. We see this same difference in mean co2 distributions in (Figure 4).

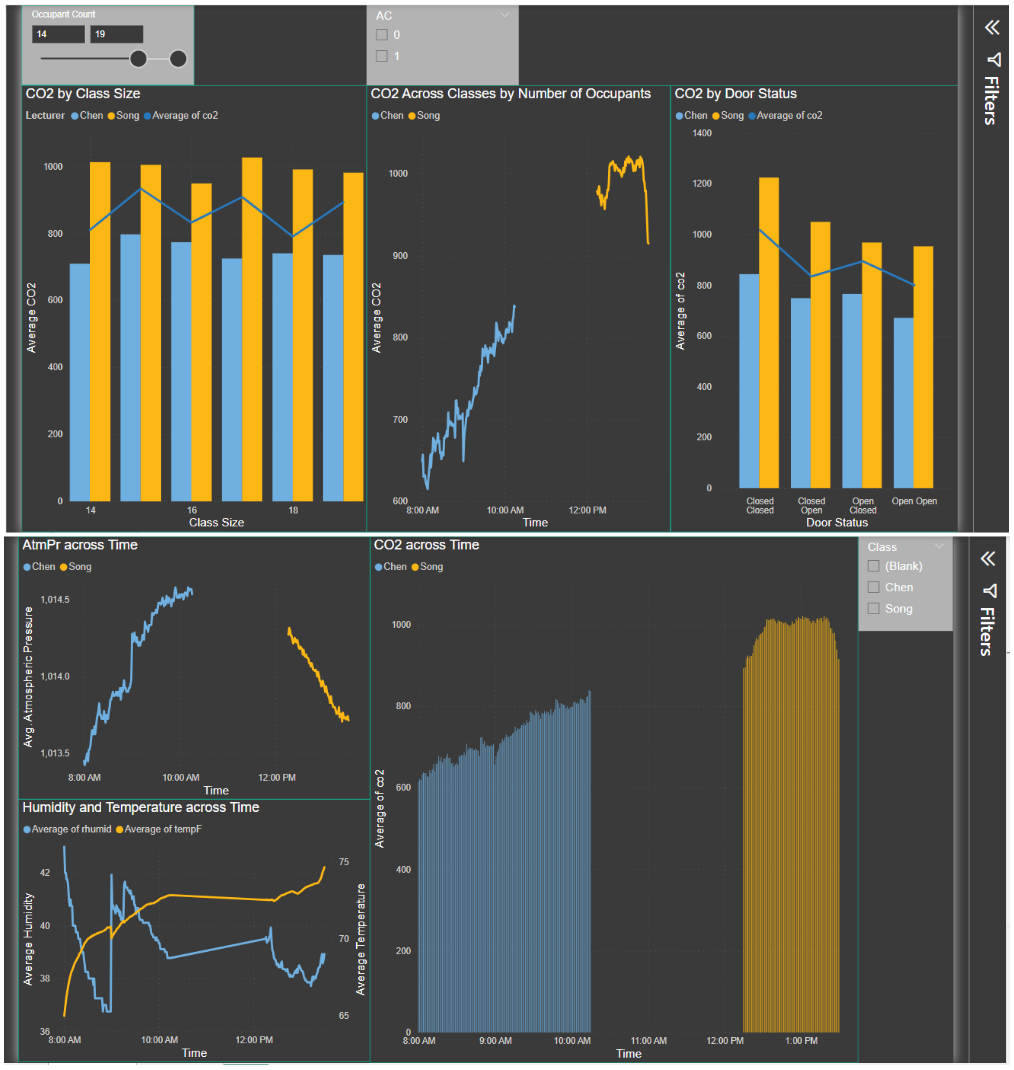


Figure 3: CO2, atmospheric pressure, humidity, and temperature measurements across different classroom settings.

We find both in that subject count really doesn't have a notable overall effect on CO_2 levels through a lecture. However, we find that the state of the hallway and study doors being open or closed does lead to a notable difference in CO_2 levels within each lecturers window of time.

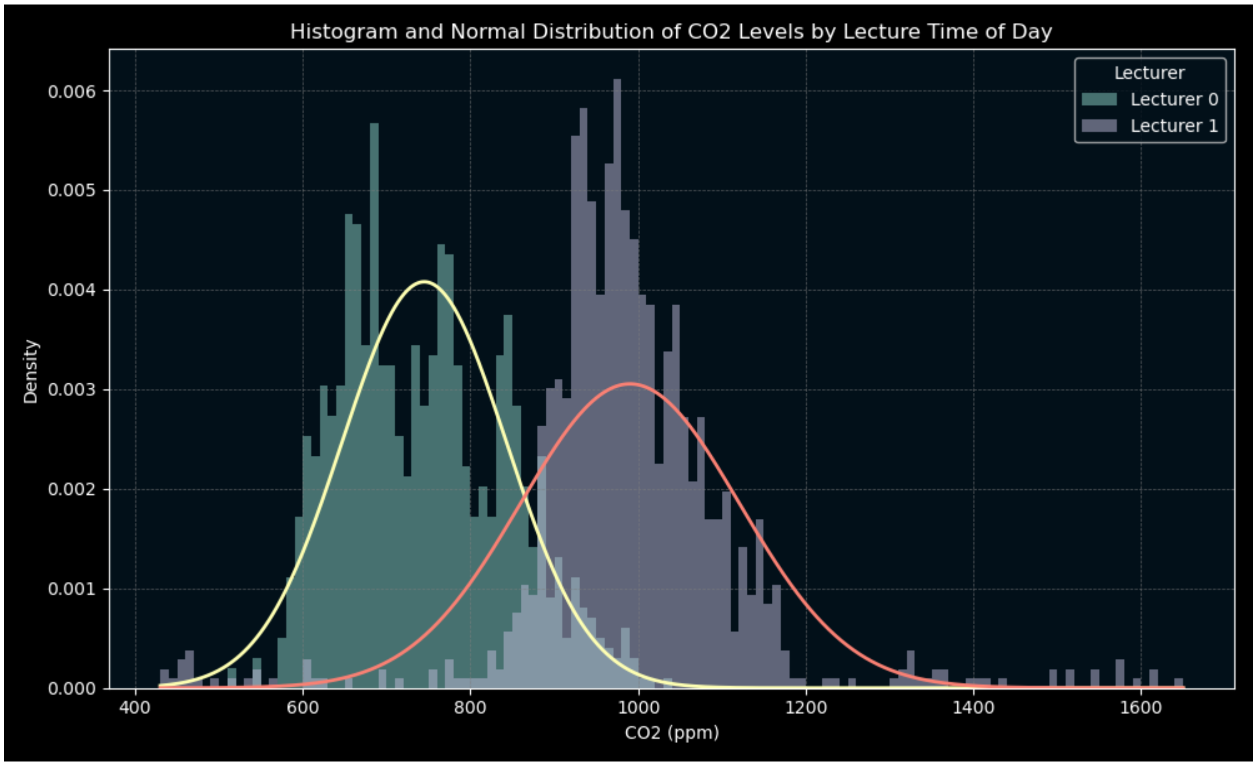


Figure 4: Distribution of CO_2 by Lecture

CO_2 State Space Dashboard

We designed the CO_2 state space monitoring dashboard with the intention of querying our sensor with a web application built to connect to the Aranet API where the dashboard tracks local changes in state for the environment. The standardized combined state space can be used to track changes in the locally specified steady state. Our dashboard is designed around our lecture window. In Figure 5 the cooler colors in the 2D and 3D standardized state space represents states earlier in time, while warmer pink colors represent more recent states. We see for the lecture sample from March 14th, 2024 for Professor Songs time period we find a disturbance in the stability of the state-space which was established during the middle portion of the lecture. On this day the lecture ended early and the observe took the sensor outside the building. We see the dashboard observes the state moving away from the stable point. This is reflected in the rapid sink of CO_2 and rise in temperature due to the outdoor environment as compared to the steady-state achieved over the first hour of lecture.

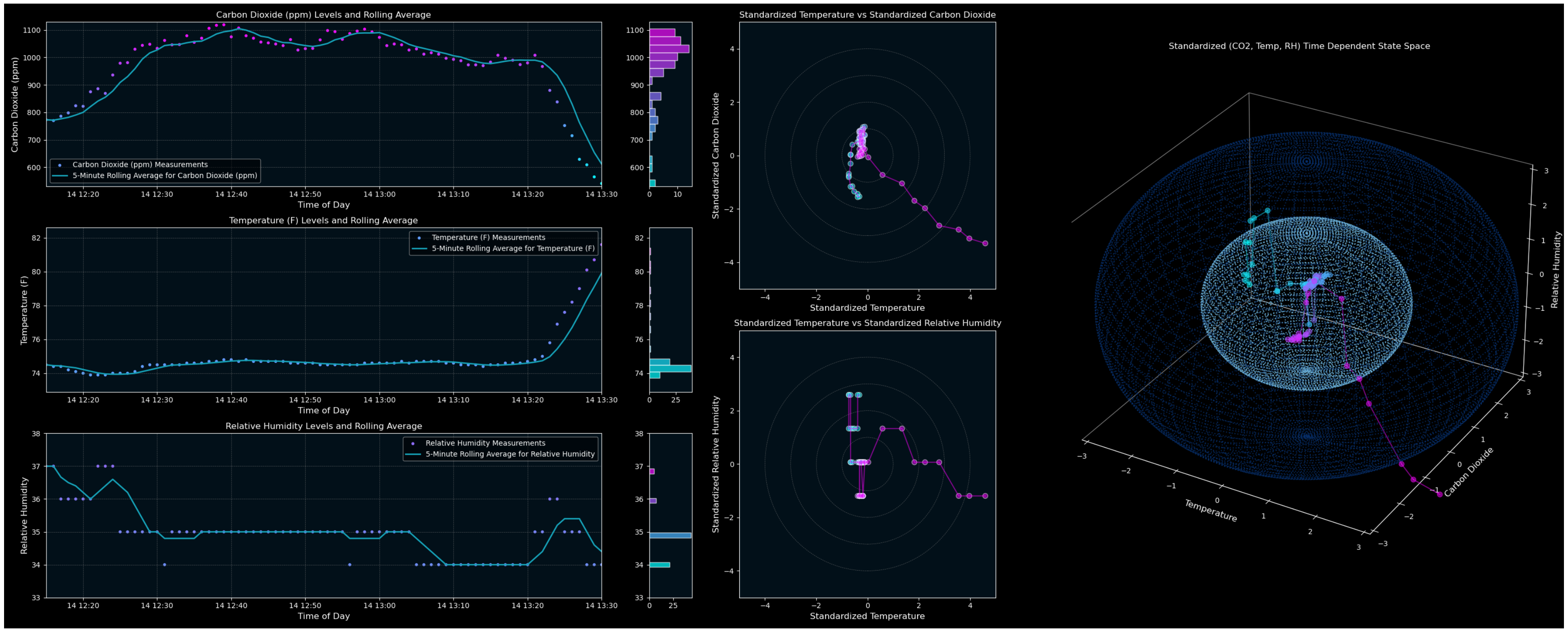


Figure 5: CO_2, Temperature, and Relative Humidity State Space Monitor System Prototype

Methodology

In this project, we dealt with two distinct datasets. The first was acquired from our Aranet sensor, which logged a comprehensive array of environmental measurements crucial for our analytical model. The second comprised data from our classroom monitoring system, detailing situational parameters such as the operational status of the air conditioning (HVAC), the position of doors (open or closed), and occupancy levels.

One significant challenge we encountered was the presence of NA values within the lecture data. Since changes in classroom state do not necessarily occur every minute, gaps appeared in the dataset upon attempting to merge it with the Aranet records, which maintain a minute-by-minute logging frequency.

Moreover, our data collection process for the classroom conditions sometimes yielded multiple entries for the same minute, resulting in duplicate records. This necessitated the implementation of a cleaning step to resolve instances of duplication prior to merging with the Aranet dataset.

The merging process was conducted on the shared Datetime Index, aligning data entries to the nearest minute. For intervals lacking state changes, we employed up-sampling techniques to forward-fill and backfill data, ensuring continuity until the next recorded change. Conversely, down-sampling methods were utilized to address the infrequent occurrences where sensor data was absent.

For the purpose of training Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) models, we segmented our time-series data into 'lecture windows,' each corresponding to the temporal bounds of a given lecture.

Subsequently, we divided these lecture windows into distinct training and test sets, a necessary step to prevent data leakage during the feature engineering phase.

In situations where 'datetime' records were duplicated, our approach was to retain the first entry and adjust the subsequent record to one minute later. This strategy was crucial in maintaining unique datetime indices for merging, while preserving the integrity of our lecture dataset.

Following this, we partitioned the data into uniformly sized lecture windows, each spanning 76 minutes. This approach, while robust, is not without its challenges. For instance, lectures do not always adhere to a standard duration, which presents the potential for outliers. An exploration of outlier detection and data stability techniques will be addressed in the subsequent sections.

TensorFlow Sequential API, RNN, & LSTMs

In forecasting the concentration of carbon dioxide (CO_2 in ppm) for forthcoming intervals of 4, 8, and 16 minutes, we anchored our predictions on data from preceding windows of 15, 30, and 60 minutes, respectively. The TensorFlow Sequential API was the framework of choice for this endeavor, offering a suite of tools well-suited for constructing the necessary neural network architectures.

The inherent structure and feature interdependencies within our dataset made Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks particularly appealing. These neural network variants excel in processing and generating predictions from sequential data, a characteristic that is central to the nature of time-series analysis.

Initially, we disregarded traditional time-series forecasting methods such as Exponential Smoothing State Space Model (ETS), Autoregressive Integrated Moving Average (ARIMA), and its variations, due to the discontinuous nature of our experimental data. However, these methodologies may be revisited and explored in the context of our future work.

Similarities

- **Sequential Processing:** Both RNNs and LSTMs are designed to handle sequential data by processing inputs in a sequential manner, allowing them to capture patterns over time.
- **Feedback Loop:** They both have a feedback loop in their architecture, where the output at each time step is fed back into the network as input for the next time step.
- **Variable Length Inputs:** They can handle input sequences of variable lengths, which is important for tasks where the length of the input sequence may vary.

Differences

- **Vanishing Gradient Problem:** RNNs suffer from the vanishing gradient problem, which occurs when the gradients become too small during training, making it difficult to learn long-range dependencies. LSTMs were specifically designed to address this issue by introducing a gating mechanism that allows the network to learn when to forget or update information.
- **Memory Cells:** LSTMs have memory cells that store information over long periods, allowing them to retain information for longer sequences compared to traditional RNNs, which only have a single tanh layer.
- **Gating Mechanism:** LSTMs have three gates (*input*, *forget*, and *output gates*) that control the flow of information, allowing them to selectively update and pass information to the next time step. RNNs do not have this gating mechanism.
- **Complexity:** LSTMs are more complex than traditional RNNs due to their additional components (gates and memory cells), which can make them slower to train and more computationally expensive.

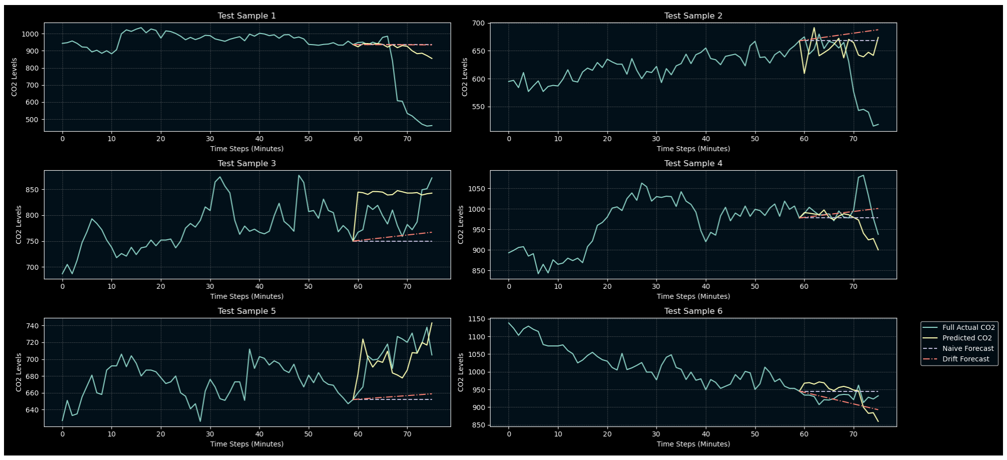


Figure 6: LSTM, naive, and drift forecast against their respective sample test data.

Analysis

In our analysis, we sought to evaluate the performance of various forecasting methods applied to time-series data. This comparative study encompassed a Long Short-Term Memory (LSTM) network, a naive forecasting approach, and a drift method. The effectiveness of each method was assessed using the Root Mean Square Error (RMSE), a widely accepted measure for evaluating the accuracy of predicted values against actual values.

The LSTM network, recognized for its proficiency in handling sequential data and its ability to remember long-term dependencies, was expected to perform robustly given the time-related nuances in our dataset. Conversely, the naive forecasting method was based on the assumption that future values are expected to be the same as the last observed value. While this method is simple, it often serves as an effective benchmark. The drift method was utilized to account for a consistent trend observed in the past data, linearly extrapolating this trend into the future.

Each test sample from our dataset underwent forecasting using these methods, generating multiple predictions per sample. The RMSE for each method was then computed by contrasting the forecasted values with the true values. This process was iterated across all test samples, accumulating the RMSEs to facilitate an average performance metric for each forecasting method.

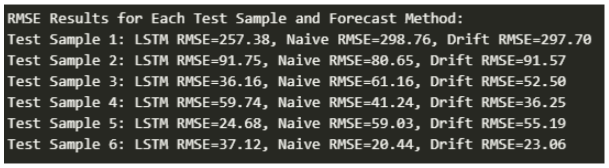
Results

The empirical results indicated a variance in performance among the three forecasting techniques. The LSTM network demonstrated lower RMSE values on several test samples, suggesting a high level of accuracy in its predictive capacity, particularly in instances where data exhibited complex temporal behaviors. In contrast, the naive approach yielded more favorable results in scenarios characterized by minimal change from the last observed value. The drift method's performance proved to be most effective when data displayed a consistent trend over time.

Interestingly, the performance of the naive method surpassed expectations, outperforming the LSTM in a subset of the test samples. This outcome highlights the importance of understanding the data's nature when selecting an appropriate forecasting model. Notably, the drift method's ability to capture and utilize a simple trend made it the superior choice in specific samples.

Two figures will be presented to illustrate the comparative performance of the forecasting methods across all test samples. These figures will encapsulate the RMSE values, thus providing a visual representation of each method's predictive accuracy.

The data visualization in Figure 6 will elucidate the relative performance differences between the LSTM, naive, and drift methods of Figure 7.

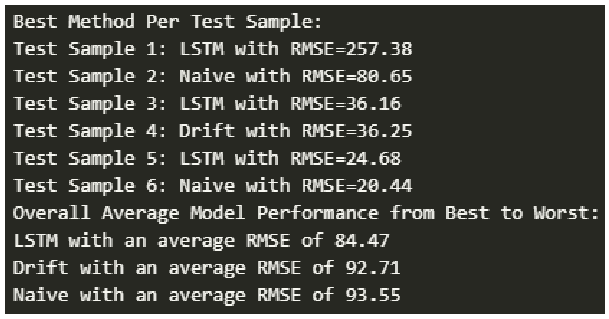


```
RMSE Results for Each Test Sample and Forecast Method:
Test Sample 1: LSTM RMSE=257.38, Naive RMSE=298.76, Drift RMSE=297.70
Test Sample 2: LSTM RMSE=91.75, Naive RMSE=80.65, Drift RMSE=91.57
Test Sample 3: LSTM RMSE=36.16, Naive RMSE=61.16, Drift RMSE=52.50
Test Sample 4: LSTM RMSE=59.74, Naive RMSE=41.24, Drift RMSE=36.25
Test Sample 5: LSTM RMSE=24.68, Naive RMSE=59.03, Drift RMSE=55.19
Test Sample 6: LSTM RMSE=37.12, Naive RMSE=20.44, Drift RMSE=23.06
```

Figure 7: Comparative RMSE Values of Forecasting Methods Across Test Samples

Figure 8 will offer a more detailed view, showcasing the distribution of RMSE values for each forecasting method across the dataset.

In summary, the analysis provided a comprehensive assessment of the forecasting methods. It reinforced the notion that the choice of model should be informed by the dataset's underlying characteristics and the specific analytical context.



```
Best Method Per Test Sample:
Test Sample 1: LSTM with RMSE=257.38
Test Sample 2: Naive with RMSE=80.65
Test Sample 3: LSTM with RMSE=36.16
Test Sample 4: Drift with RMSE=36.25
Test Sample 5: LSTM with RMSE=24.68
Test Sample 6: Naive with RMSE=20.44
Overall Average Model Performance from Best to Worst:
LSTM with an average RMSE of 84.47
Drift with an average RMSE of 92.71
Naive with an average RMSE of 93.55
```

Figure 8: Best Performing Model per Sample and Overall Model Performance by RMSE

Further Work

Expand the Sample Space

Our results show that the observed state of the doors, hvac, and occupant count may not impact our models ability to forecast for longer durations as much as including the lag history and moving averages of the target variable in question into the model. This was less true when using 15 minute windows to forecast the next 4 minutes where the number of lags and moving averages for each target variable becomes limited due to the risk of data leakage during the creation of said features.

However, if we drop the observed lecture experiment data, then we can expand our options for training windows from the entirety of the *Aranet4* sample space. The current *Aranet4* contains over 129000 minutes of continuous CO_2, r. humidity, temp., and atmospheric pressure records to slice windows from compared to the total 2052 total minutes of lecture window data. This would provide us with a space 62 times the size of the current on for a possible 1700 samples of 76 minute windows to train our model on. Recall that our project had 27 lectures of data to work with and we trained our model with 21 lectures so as to test on 6 unseen lectures. We saw in the outline of the state space dashboard to be built with the plotly dash API the impact that rapid changes in temperature and humidity can have on the stability of the CO_2 stability (Figure 5).

References