ANKARA UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

COM 4062 PROJECT REPORT

Indoor Air Quality Prediction System for Automation

Kadir Gökdeniz

20290344

Gazi Erkan Bostancı

May, 2024

ABSTRACT

This thesis project aims to forecast how factors affecting indoor air quality in smart home systems will evolve in the future and to create an interface facilitating interaction between the user and the automation system. It is well known that factors such as temperature and humidity have a significant impact on human health. Predictions generated by the project aim to prevent possible illnesses and increase human well-being. Additionally, predictions of indoor air quality can facilitate the taking of necessary precautions before the occurrence of sudden disasters. Understanding how automation systems will function in the future within the context of smart home design can be important for system engineers to evaluate their performance during the testing phase. The creation of digital twins of data on smart home systems and its potential to serve as the basis for broader projects are significant aspects of this work. Data can be analyzed on a room-by-room basis and displayed separately. In Automation Pyramid, this type of visualization stage also known as SCADA. This study is the result of an analysis of time-series data using deep learning, specifically recurrent neural networks. Various neural network architectures such as GRU and LSTM have been experimented with on this dataset, and the models yielding the best results have been selected. The project can be executed on platforms such as Google Colab and Anaconda. It also encompasses a comprehensive literature review. The planned collaboration with my advisor, who is guiding me through this project, encourages me to pursue further research in this field in the future. Upon completion of this project, I intend to share the entirety of the project materials as open-source.

TABLE OF CONTENTS

1. INTRODUCTION

1.1Background and Subject Declarations

This project helped me understand the logic of automation systems used in smart home systems and enabled me to provide access to automation requests from users, as well as learn how to prepare data for RNN. It also expanded my knowledge of machine learning techniques and applications of deep learning. In short, it provided me with the opportunity to obtain desired outputs by using appropriate data and facilitated user interaction with automation systems, which is a significant milestone for users.

Automation systems are applications that are trained to perform desired functions by users, yet operate based on learned patterns without direct interaction from the user, thus providing time and energy savings.

For the design of automation systems, data researchers collect data, which is obtained from appropriate pools depending on the subject of automation, such as health systems, industrial applications, agriculture, and business process automation. The data is often not in the appropriate form and format, requiring some preprocessing steps to be performed in order for the data to be processed. Among these problems are outliers, cleaning and normalization of datasets for deep learning methods used for prediction (often such as RNNs), and sometimes feature engineering. These preprocessing steps are essential for the model to make more accurate and reliable predictions. Additionally, reducing the size of datasets and eliminating unnecessary information is an important step, as smaller and more focused datasets generally yield better results. Let's examine some word phrases:

Feature Engineering is an important step in the analysis of time data and the generation of predictions. In this step, the relational meanings between different attributes in the data are examined and compared. High or very low correlation relationships between data make it difficult to learn the necessary patterns from the data. Therefore, some attributes should not be included in the training set.

Additionally, sometimes it is possible to generate different attributes from the data, and meaningful patterns can be extracted from these newly generated attributes to better learn the data.

Deep learning methods, especially recurrent neural network architectures like GRU and LSTM in Python, enable predictions to be made based on time series data. These systems can work in conjunction with automation systems for important purposes such as making health and safety warnings in advance and improving quality of life. The automation system can gain a more reliable, flexible, and predictable structure as a result.

2. AIR QUALITY PREDICTION SYSTEM AND AUTOMATION

2.1. Definitions and Declarations about Automation

Increasingly, smart homes today come with automation systems alongside new technologies. Smart home systems not only offer automation systems but also provide comprehensive services in areas such as health, security, energy efficiency, well-being, and time savings. Developed automation systems focus on situations that are difficult for people to manage directly but have a negative impact on human life. Among these are topics such as detecting low air quality, fire and gas leakage, optimizing solar energy consumption (compared to grid usage), and system monitoring. Due to its extensive scope, this project cannot cover the development of systems that will design the entire home system. Therefore, it focuses on specific areas such as only detecting low air quality, monitoring the automation system, and ensuring interaction with the user.

With the rapid advancement of deep learning methods, there has been an increase in the number of projects developed in this field. On the other hand, automation systems are systems with minimal interaction with users and can adversely affect user well-being in cases of unwanted situations. Future time predictions with deep learning techniques can be transformed into a trace presented to users, allowing them to make change requests in automation systems. Such an approach is not included in traditional programming methods, and automation systems, processing real-time data, can affect users in unwanted situations. This project aims to prevent this situation with

future time predictions and to mitigate it to some extent through user interaction.

2.2. Tools and Platforms for Automation

2.2.1. Data Collection Tools

Data for automation and prediction systems can be prepared using different datasets.Sufficient access to high-quality and diverse data is crucial, especially for quality prediction capabilities. Automation systems can provide a more prosperous living environment and make saving easier with access to quality data. Access to many open-source datasets is possible:

- Climate Scale, Nature.com (scientific data), Joint Research Centre Data Catalogue - Indoor Air Quality, Mendeley Data examples.

2.2.2 Data Analysis Tools

Using time series data to make attributes meaningful and to design prediction systems is crucial. The use of auxiliary libraries on time series facilitates operations on time features, while deep learning libraries enable technical operations on data, such as backpropagation, to be applied more easily. These form the building blocks of prediction system design.

- Time series libraries include pandas, numpy, Matplotlib and Seaborn, Statsmodels, and Prophet.
- For deep learning frameworks, PyTorch and TensorFlow are used for time series.

2.2.3. Data Storage and Processing

The data used in this field is generally inaccessible due to data privacy concerns. Keeping a high volume of data is necessary, as mentioned in the previous paragraph. However, having an excessive amount of data also brings storage and processing issues. Time series data can be in unstructured, semi-structured, or structured formats. Some tools are commonly used to deal with these problems.

- For structured data, Apache Hadoop

- For semi-structured data, Apache Spark
- For unstructured data, Apache Kafka.
- Some cloud-based systems can also be used. AWS, Microsoft Azure, and Google Cloud Platform are examples of these.

2.2.4. Visualization Tools for Dataset and Results

Visualizing data is a commonly used practice among practitioners to examine how well the learned model fits the test data. Visualization is often used to detect overfitting and evaluate overall performance. While the matplotlib library is frequently used for time series data, Power BI and Plotly libraries are also quite useful.

2.2.5. Automation and Prediction Platforms

Automation systems designed for smart homes utilize real-time data to generate commands on how the system will operate. Test processes are applied to evaluate the possible outcomes of these commands. Raspberry Pi, Node-RED, OpenHab, MongoDB, InfluxDB, MQTT, and Ignition are currently used platforms for testing purposes [1]. Furthermore, in addition to deep learning methods for visualizing prediction data and user interaction, Streamlit and Gordio libraries are commonly used.

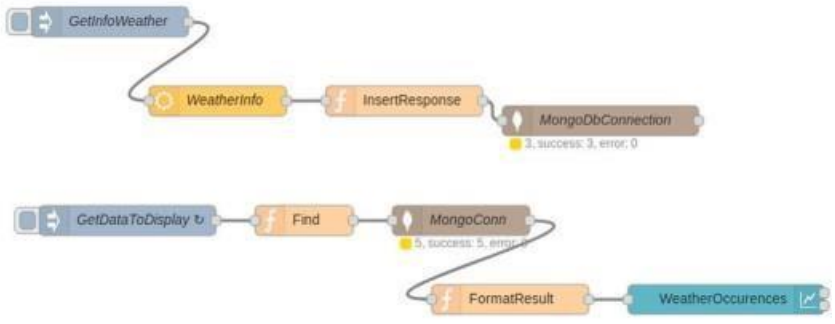


Figure 2.1. Automation Test Example with MQTT in smart office.

2.3. How are Automation&Prediction Systems Changing Our Lives?

With the technological leap in the last decade, just like every device nowadays, our homes have also started to become smarter. With the Covid-19 pandemic we have experienced in recent years, we started to spend more time at home and felt the impact

of the technologies in our homes more deeply.

Smart home systems aim to enhance life performance by taking a holistic approach to human life. By extracting the personal twin of each location and individual, use cases can be visualized, and systems can be optimized based on this information. In this task, automation systems, sensors, and high-speed fiber internet infrastructure play an important role. Transitioning to automation systems instead of user management can increase user well-being in important areas such as automatic lighting systems, heating and cooling, security and alarms, air quality and humidity.

However, due to potential error factors inherent in every software system; among these reasons, the automation system may be inadequate due to the inability to anticipate short-term changes in user demands (for example, let's consider a situation where the homeowner accepts a short-term guest and their demands are different), the inability to foresee abnormal situations (for example, a disaster occurring inside the room), or any reason causing sensors to produce incorrect information. An inclusive solution could be an alternative approach where automation systems and future prediction are used together.

2.3.1 How Automation & Prediction System is Run?

The collaboration of automation and prediction systems involves using future time prediction data to fill in the gaps where the automation system falls short. Real-time data is stored in a database and fed into the prediction system to train the model. Access to future data is provided through the trained model, and this data is transferred to a web application designed with the Streamlit interface. This interface allows homeowners to view relevant prediction data and obtain information about the potential operating patterns of the automation system. Additionally, homeowners can suspend the operation of the automation system or request it to operate through this interface. New data is recorded, and the automation system periodically checks for any external requests.

This process is evaluated cyclically. User feedback is collected, and based on

this information, the interaction between the automation system and prediction data is assessed. This cyclical approach enhances the harmony between the two different systems. Continuous improvement in the quality and acquisition of new data can lead to a continuous improvement in homeowners' quality of life.

2.3.2 Prediction & Automation: Obtaining Appropriate Answers

The rapidly increasing amount of data brings storage and processing challenges. Meanwhile, automation systems, coupled with deep learning techniques, can gain insights while improving human life. As our homes become smarter assistants, they provide homeowners with the necessary infrastructure for time and energy.

Indoor air quality prediction for automation is among the best examples that enhance human life quality. It is often preferred due to containing fewer outliers during the design phase of automation systems and carrying data for future times. In recent years, the prominent 'Smart Twin' application utilizes future time predictions for analyzing homeowners' behaviors and creating use cases. Prediction and automation systems can be used together for adapting to changing user demands, implementing early warning systems, and preventing possible sensor errors.

2.4 Indoor Air Quality Prediction System

2.4.1. Dataset

Examining datasets that encompass high-resolution, long-term, and publicly accessible electricity consumption data is crucial for smart home technologies. The dataset utilized in this project is the CN_OBEE dataset, covering data from May 31, 2021, to May 31, 2022. It includes minute-level data on occupant presence, window status (open/closed), thermal environment, and appliance usage from a single-family apartment in Beijing, China. Hourly meteorological data from the nearest weather station for the same period are also included, covering the same time intervals. The data were collected by an IoT-based data collection framework (IDCP) using sensors placed in the

rooms.

There are some advantages to choose this dataset:

Comprehensive Data Content: The dataset contains comprehensive data on electricity consumption, indoor environment, and occupant behavior in a typical urban household. This provides a rich source of data for various analyses and research purposes.

Universality: The dataset includes data from a typical urban household in Beijing, China, making it universally applicable on a global scale. Similar analyses and research can be conducted in households located in different geographical regions.

Thermal Comfort Analysis: Indoor temperature and humidity data can be used to evaluate the thermal comfort levels of the residence. Comfort analysis can guide improvements in indoor environmental conditions.

Data Collection Method: The Internet-of-Things (IoT)-based data collection method enables long-term and real-time data collection using small and cost-effective sensors. This can serve as a model for similar data collection projects. These points highlight the value and versatility of the dataset, making it suitable for various research and analysis purposes, particularly in the domains of energy management, building performance optimization, and indoor comfort analysis.

1	time,dry-bulb temperature (°C),relative humidity (%),air pressure (Pa),window state (0/1),east window state (0/1),occupancy presence (0/1)
2	2021-05-31 00:00:00,24.18,45.37,99900.0,1,1,0
3	2021-05-31 00:01:00,24.18,45.37,99900.0,1,1,0
4	2021-05-31 00:02:00,24.18,45.37,99900.0,1,1,0
5	2021-05-31 00:03:00,24.18,45.37,99900.0,1,1,0
6	2021-05-31 00:04:00,24.18,45.37,99900.0,1,1,0
7	2021-05-31 00:05:00,24.18,45.37,99900.0,1,1,0
8	2021-05-31 00:06:00,24.18,45.37,99900.0,1,1,0
9	2021-05-31 00:07:00,24.18,45.37,99900.0,1,1,0
10	2021-05-31 00:08:00,24.18,45.37,99900.0,1,1,0

Figure 2.2. A part of dataset for living room.

2.4.2. What is an Air-Smart Controller?

In the automation pyramid, SCADA (Supervisory Control and Data Acquisition) and MES systems, which are located at the supervisory level, can be integrated for remote management and visualization of data. SCADA systems are used to control industrial processes and collect, process, and visualize data related to these processes. These systems are utilized to monitor, control, and analyze devices and processes in factory environments. SCADA and MES systems at the supervisory level play a significant role in enhancing the performance of production facilities, increasing efficiency, and supporting operational decision-making.

Air-Smart Controller relies on the collaboration between an intelligent home automation system and a deep learning model. Instantaneous data collected from sensors is stored in a database. Subsequently, at specific intervals, this data is transferred to the prediction model. Utilizing deep learning techniques, the model generates future time data. Air-Smart Controller is a web application with a Streamlit interface that visualizes prediction data. Through this application, temperature, pressure, and humidity information for any room can be accessed at ten-minute intervals. The user can choose a day to view approximately 44 days' worth of prediction data. Alongside this data, the user can access information about the operation mode and schedule of the automation system. If desired, the user can suspend this operation mode or request a specific time range for operation.

2.4.3. Import Libraries

The project has been coded using Google Colab. Some of the key libraries I used in completing the project are Numpy, Pandas, Matplotlib, Seaborn, Sklearn, and TensorFlow. Since I work with time data, the datetime library is also quite useful.

```python
# Import libraries
from google.colab import drive
import zipfile
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import datetime
import seaborn as sns
from sklearn.preprocessing import LabelEncoder,MinMaxScaler
import keras
import tensorflow as tf
from keras.models import Sequential
from keras import layers,regularizers
from tensorflow.keras import optimizers
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
import json
import csv
from enum import auto
```

Figure 2.3. Importing Libraries.

2.4.4. Get Data Ready

The electricity consumption data and the data within the rooms are divided into intervals of 10 minutes each. On the other hand, the external weather information is obtained hourly from the nearest weather station. However, there is a discrepancy because I am examining the data at 10-minute intervals in the project. To address the time misalignment issue, I assume that the external weather conditions remain constant within 10-minute intervals within each hour.

```python
# Prepare outdoor for other datasets
outdoor_weather['time'] = pd.to_datetime(outdoor_weather['time'])

outdoor_weather.set_index('time', inplace=True)
outdoor_weather = outdoor_weather.resample('T').ffill()
outdoor_weather.reset_index(inplace=True)
```

Figure 2.4. Outdoor Weather Arrangement

After correcting all the data, it is important to check for any 'object' type data in the dataset. While conducting this check, I noticed that the 'wind direction' data is of 'object' type. Since the data needs to be numerical for modeling purposes, I am using the 'Label Encoder' method to convert this 'object' type data into numerical format.

Figure 2.5. Converting object type to numerical values.

After checking for the 'object' data type, it is necessary to look for the presence of None values within the data. A small percentage of None values can be tolerated. However, due to the high proportion of None values, as observed in the example of Room 5, also known as the 'Master Bedroom', evaluating this data is quite challenging. Therefore, the examination of the 'Master Bedroom' will be excluded. Data belonging to other rooms is filled with median values as they are more robust compared to mean and mode.

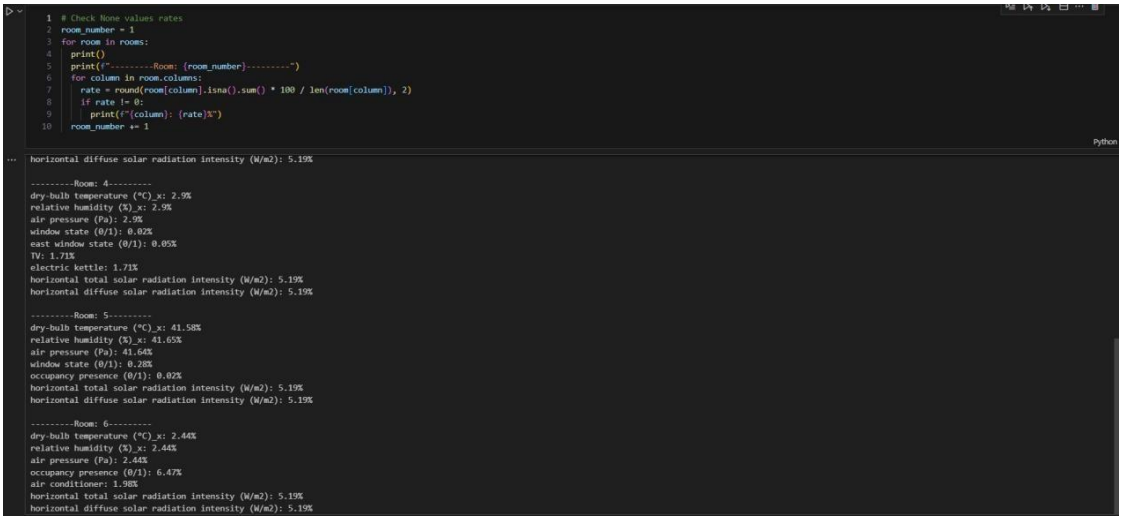


Figure 2.6. Handling with None Values

2.4.5. Data Visualization

After preparing the data, it is important to examine the characteristic features of the data as it is essential for understanding the data. Since we are working on a regression problem with time data, analyzing the presence of outlier behaviors and how frequently these outliers occur can facilitate this process. For this task, we have a function called 'plotMovingAverage' which utilizes the Exponential Moving Averages function to detect abnormal behavior. This function depicts the examination of temperature data in the living room over a specified time interval as shown below.

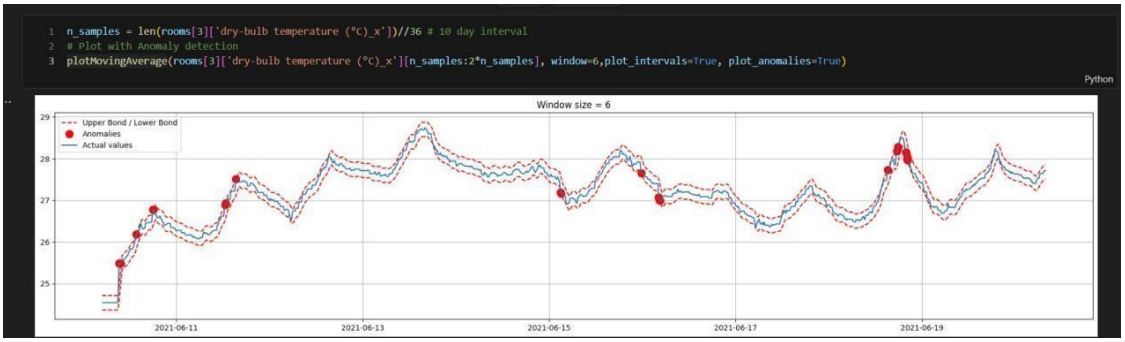


Figure 2.7. Abnormal value detection in living room for temperature.

2.4.6. Feature Engineering

The analyses we conduct on the dataset provide us with a good resource for training. In addition to these existing data, we can add new related data or exclude some data. Adding new data also involves generating new data. For example, information such as temperature, pressure, and humidity are closely related to seasons, but the dataset may not contain season information. In such cases, we can add season information to the data in a numerical form. On the other hand, there may be data with similar characteristics within the dataset. Using such data for training may increase the risk of over-fitting. We use a correlation matrix to examine the correlational schema.

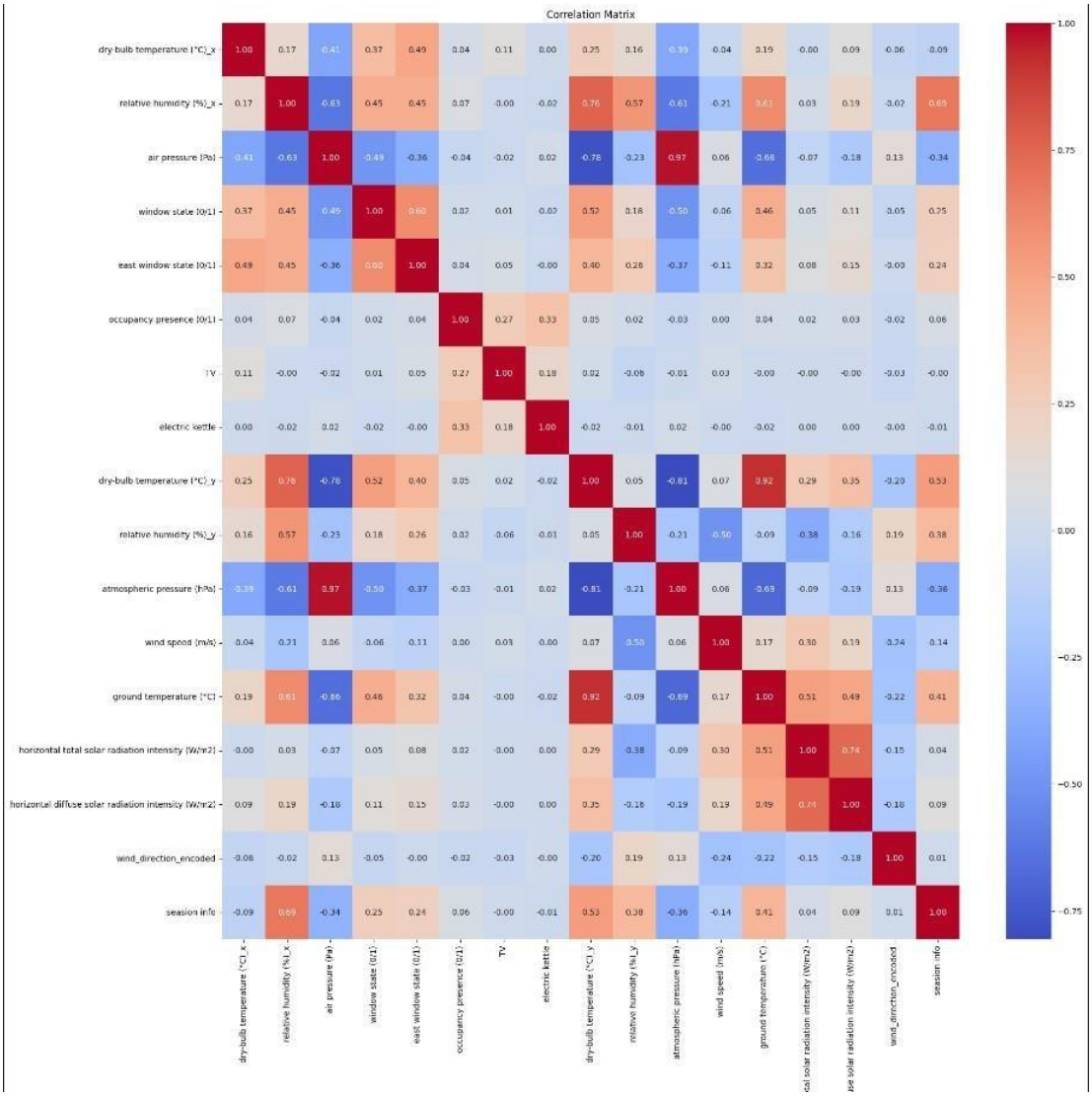


Figure 2.8. Correlation Matrix.

2.4.7.Data Preprocessing

As frequently utilized in other deep learning methods, normalizing data enhances prediction capability. One of the main reasons for this is its ability to reduce the impact of outliers.

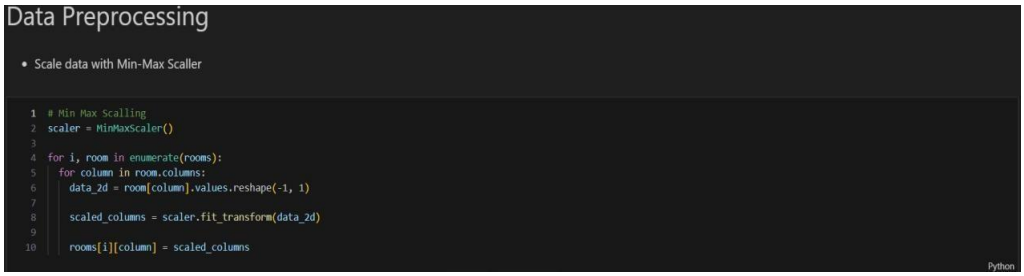


Figure 2.9. Min-Max Scaller

```python
def generator(data, lookback, delay, min_index, max_index,shuffle=False, batch_size=128, step=6, column_number=0):

  if max_index is None:
    max_index = len(data)- delay- 1

  i = min_index + lookback

  while 1:
    if shuffle:
      rows = np.random.randint(min_index + lookback, max_index, size=batch_size)

    else:
      if i + batch_size >= max_index:
        i = min_index + lookback
      rows = np.arange(i, min(i + batch_size, max_index))
      i += len(rows)

    samples = np.zeros((len(rows),
                        lookback // step,
                        data.shape[-1]))
    targets = np.zeros((len(rows),))

    for j, row in enumerate(rows):
      indices = range(rows[j]- lookback, rows[j], step)
      samples[j] = data[indices]
      targets[j] = data[rows[j] + delay][column_number]
    yield samples, targets
```

Figure 2.10. Data Generator for RNN.

This function creates a data generator for learning from time series data. This generator can be used to predict a future value by looking at previous data in a certain time window in the dataset.

Parameters:

- data: The time series data to be processed.
- lookback: The time window that the model will look back on previous data.
- delay: The time interval to be predicted.

- min_index, max_index: Used to slice the portion of the dataset.
- shuffle: A flag indicating whether the data should be shuffled.
- batch_size: The number of samples to be generated in each data loop.
- step: The time interval between samples.
- column_number: The column number of the target variable in the dataset.

The function continuously produces samples in a loop. These samples are taken from the given dataset, sliced according to the specified time intervals and steps, and matched with the target variables.

The lookback value is set to a width of 10 days. Step indicates intervals of 30 minutes. The delay is set to a length of 1 day. The batch size is determined as 128. For training, 70% of the data is allocated, 15% for validation, and 15% for testing.

2.4.8. Building Models

In this project, there are three RNN models that can be used. Initially, LSTM and GRU models have been tested. Due to the more complex structure of LSTM, its performance on the model was found to be lower compared to GRU. Based on the correlation matrix obtained from the feature engineering part, some data with similar correlation values have been excluded from training. Therefore, the breadth of hypothesis sets used for temperature, pressure, and humidity predictions will vary. The artificial neural network we construct may require an alternative architecture due to this variation, as preventing over-fitting is crucial.

Hence, a more complex model is initially preferred, but in case of early signs of over-fitting, an alternative simpler GRU model has been chosen.

Detailed information about the model is provided in the image below.

```python
# Model function
def build_model(model_name,shape, isAlternative=False):

  model = Sequential()

  if model_name =='LSTM':
    model.add(layers.LSTM(20,return_sequences=True, input_shape=(None, shape),kernel_regularizer=regularizers.l2(0.01)))
    model.add(layers.LSTM(10,return_sequences=False,kernel_regularizer=regularizers.l2(0.01)))

  elif model_name =='GRU' and not isAlternative:
    model.add(layers.GRU(25,return_sequences=True, input_shape=(None, shape),kernel_regularizer=regularizers.l2(0.01)))
    model.add(layers.GRU(13,return_sequences=False,kernel_regularizer=regularizers.l2(0.01)))

  elif model_name =='GRU' and isAlternative:
    model.add(layers.GRU(20,return_sequences=True, input_shape=(None, shape),kernel_regularizer=regularizers.l2(0.01)))
    model.add(layers.GRU(10,return_sequences=False,kernel_regularizer=regularizers.l2(0.01)))

  model.add(layers.Dense(1))

  return model
```

Figure 2.11. Build Model Function.

2.4.9. Other Helper Functions

Since using the 'build_function' function directly can be difficult in terms of readability, we use the 'assign_model' function. This function is called with the information about whether an alternative model is chosen or not.

```python
# Model Assigner
def assign_model(model_name,data,room_number,isAlternative=False):
  if isAlternative:
    model = build_model(model_name=model_name, shape=data[room_number].shape[-1],isAlternative=True)
  else:
    model = build_model(model_name=model_name, shape=data[room_number].shape[-1],isAlternative=False)
  return model
```

✓ 0.0s Python

Figure 2.12. Assign Model function.

Similarly, I use a helper function called 'compile_model' to compile the model. This function utilizes the Adam optimizer for training the model. The learning rate is set to 0.001. Additionally, a callback is employed to divide the learning rate by 10 after each epoch if the model over-fits. While the loss function used is 'mse' (Mean Squared Error), changes in the 'mae' (Mean Absolute Error) values can be also observed simultaneously.

```python
# Model Compiler
def compile_model(model):
  model.compile(optimizer=optimizers.Adam(learning_rate=0.001), loss='mse', metrics=['mae'])
```

✓ 0.0s

Figure 2.13. Compile Model function.

The 'train_model' function is used for model training. This function includes 10 epochs and monitors the performance on the validation dataset. As

mentioned in the previous paragraph, the model tracks over-fitting (based on the performance on the validation dataset) using a callback. If the model over-fits, the learning rate is reduced to slow down the rate of weight updates, thus preserving performance.

```
# Model Trainer
def train_model(model,room_number,attr):
  history = model.fit_generator(train_gen_list[room_number][attr],
                                steps_per_epoch=500,
                                epochs=10,
                                validation_data= val_gen_list[room_number][attr],
                                callbacks=callbacks_list,
                                validation_steps=val_steps)
  return history
```

✓ 0.0s

Figure 2.14. Train Model Function.

After training, the performance of the model is visualized using the 'visualize loss' and 'visualize_mae' functions.

```
# Visualize Loss
def visualize_loss(history,attr,room_number):
  room_dict={
      0: 'Cloakroom',
      1: 'Home Office',
      2: 'Kitchen',
      3: 'Living Room',
      4: 'Secondary Bedroom'
  }
  print()
  loss = history.history['loss']
  val_loss = history.history['val_loss']
  epochs = range(1, len(loss) + 1)
  plt.figure(figsize=(10,4))
  plt.plot(epochs, loss, 'b', label='Training Loss')
  plt.plot(epochs, val_loss, 'go', label='Validation Loss')
  plt.title(f'{room_dict[room_number]} {attr} Training and Validation Loss')
  plt.legend()
  plt.show()
```

Figure 2.15. Visualize Loss function.

```python
# Visualize MAE
def visualize_mae(history,attr,room_number):
    room_dict={
        0: 'Cloakroom',
        1: 'Home Office',
        2: 'Kitchen',
        3: 'Living Room',
        4: 'Secondary Bedroom'
    }
    print()
    mae = history.history['mae']
    val_mae = history.history['val_mae']
    epochs = range(1, len(mae) + 1)
    plt.figure(figsize=(10,4))
    plt.plot(epochs, mae, 'b', label='Training MAE')
    plt.plot(epochs, val_mae, 'go', label='Validation MAE')
    plt.title(f'{room_dict[room_number]} {attr} Training and validation MAE')
    plt.legend()
    plt.show()
```

✓ 0.0s

Figure 2.16. Visualize MAE function.

To understand how similar the predicted values are to the actual values, visualization is done using the 'plot_results' function.

```python
# Plot Results
def plot_results(original,predictions,model_name,room_number,attr):

    # Room names
    room_dict={
        0: 'Cloakroom',
        1: 'Home Office',
        2: 'Kitchen',
        3: 'Living Room',
        4: 'Secondary Bedroom'
    }
    print()
    plt.figure(figsize=(10,4))
    plt.plot(original[:-144:], color='blue', label='Original Value')
    plt.plot(predictions[144:,0][::] , color='green', label='Predicted Value')
    plt.title(f'{room_dict[room_number]} {attr} Prediction {model_name}')
    plt.xlabel('from 15 April 00:00 - 29 May  00:00 with 10 minutes interval')
    plt.ylabel(attr)
    plt.legend()
    plt.show()
```

✓ 0.0s

Figure 2.17. Plot Results function

To display metrics related to the results, the 'calculate_metrics' function is used.

```python
# Calculate Metrics
def calculate_metrics(original,predictions,room_number,attr):
    mae = mean_absolute_error(original, predictions)
    mse = mean_squared_error(original, predictions)
    r_squared = r2_score(original, predictions)
    rmse = np.sqrt(np.mean((original-predictions)**2))
    print()
    print(f'{attr}')
    print('MAE: %.3f' % mae)
    print('MSE: %.3f' % mse)
    print('R-squared: %.3f' % r_squared)
    print('RMSE: %.3f' % rmse)
```

Figure 2.18. Calculate Metrics

2.4.10. Finish Function

Throughout the application, all these helper functions are combined in a function called 'finish'. When the 'finish' function is called, models for each room are created separately, and then the helper functions are called sequentially. The 'finish' function completes the temperature, relative humidity, and pressure predictions for a room and presents the necessary measurement values. Since we have data for a total of 5 rooms, we obtain 15 different prediction data, and the data for each room is saved into separate CSV files.

2.4.11. Displaying Results

Through the helper functions, we can visualize the training results. Here are the results for any rooms: (All data for other rooms are available in the Air_Quality.ipynb file.)

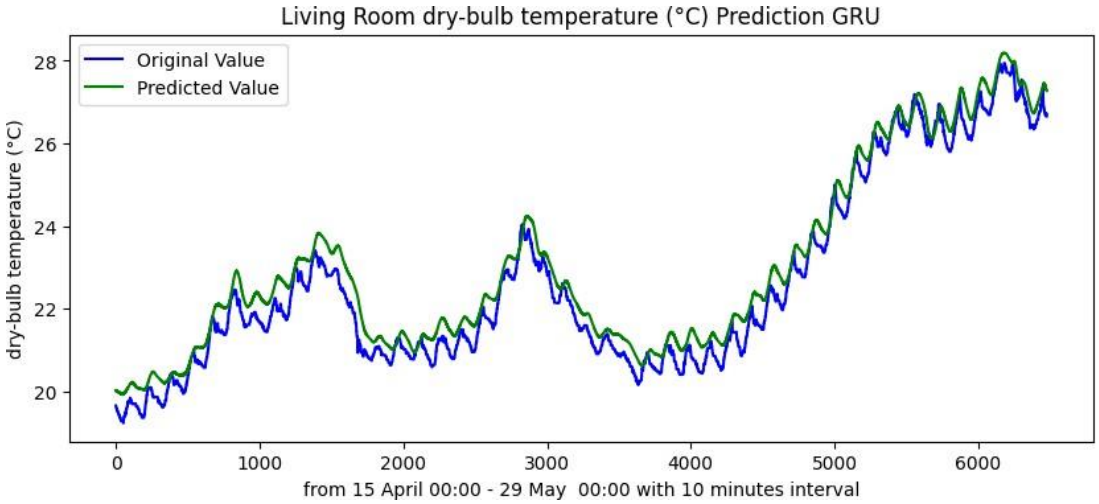


Figure 2.19. Living Room Temperature Prediction and Original Value

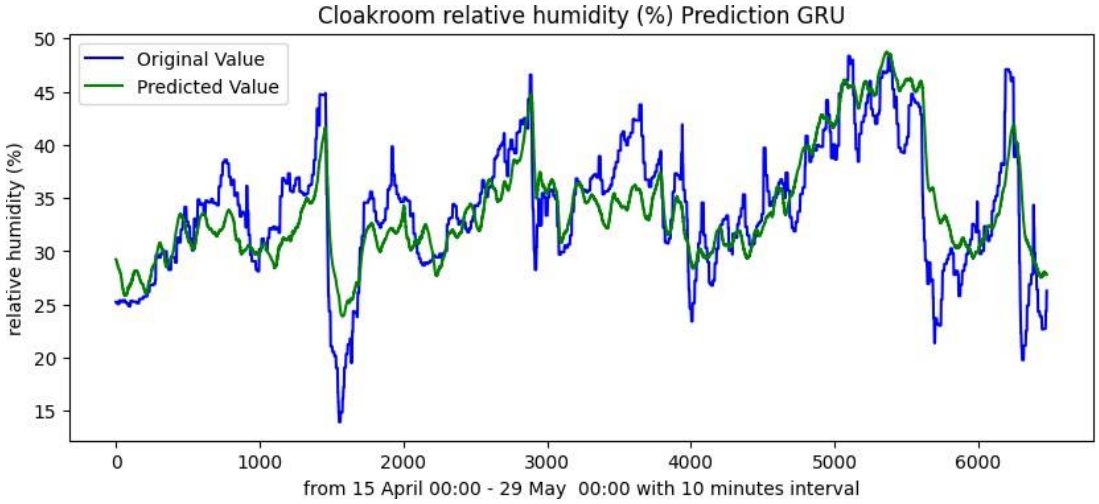


Figure 2.20. Cloakroom Relative Humidity Prediction and Original Values

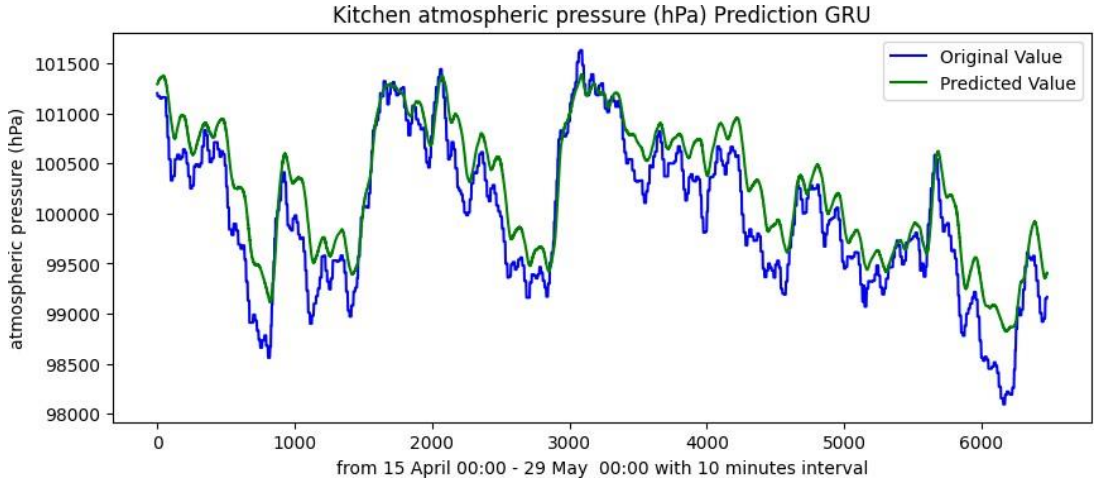


Figure 2.21. Kitchen Air Pressure Prediction and Original Values.

All predictions are very closer to original one. Generalization capability of the function is higher. Other details for further examination next section can help understand more.

3.RESULTS EXAMINATION

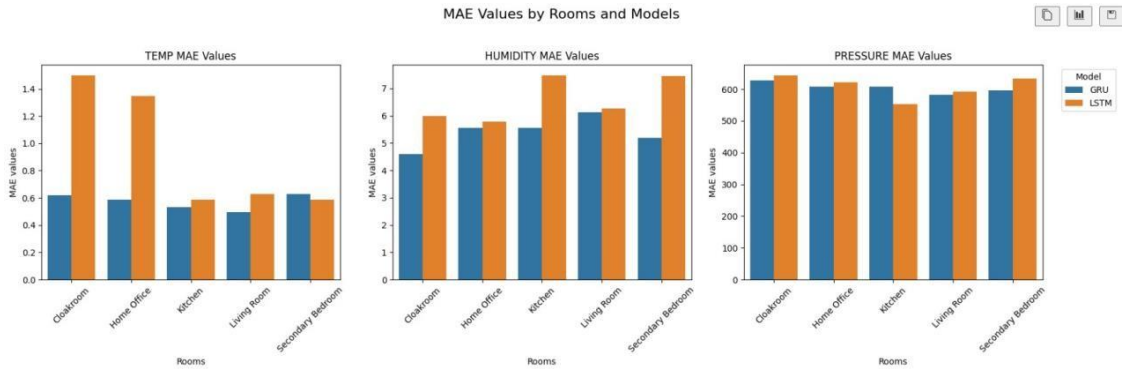


Figure 3.1. MAE Results

With these data, you can compare the performance of LSTM and GRU models based on the values of rooms and metrics (MAE, MSE, RMSE). MAE indicates how much the model's predicted values deviate from the actual values. The smaller the value, the better the model's performance. In temperature predictions, LSTM models generally have higher MAE values, indicating that GRU models perform better. A similar situation is observed in humidity predictions, where LSTM models typically have higher MAE values. In pressure predictions, LSTM models also show higher MAE values, while GRU models generally perform better.

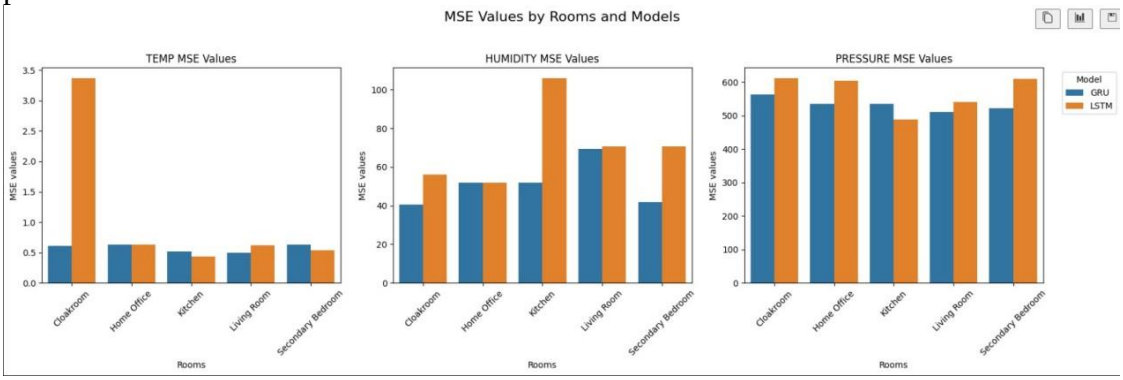


Figure 3.2. MSE values

MSE provides the average of the squared errors and shows how sensitive the model is to large errors. The smaller the value, the better the model's performance. In temperature predictions, GRU models generally have lower MSE values, indicating better performance. In humidity predictions, LSTM

26

models typically have higher MSE values, showing that GRU models perform better. Similarly, in pressure predictions, GRU models generally have lower MSE values, indicating better performance.

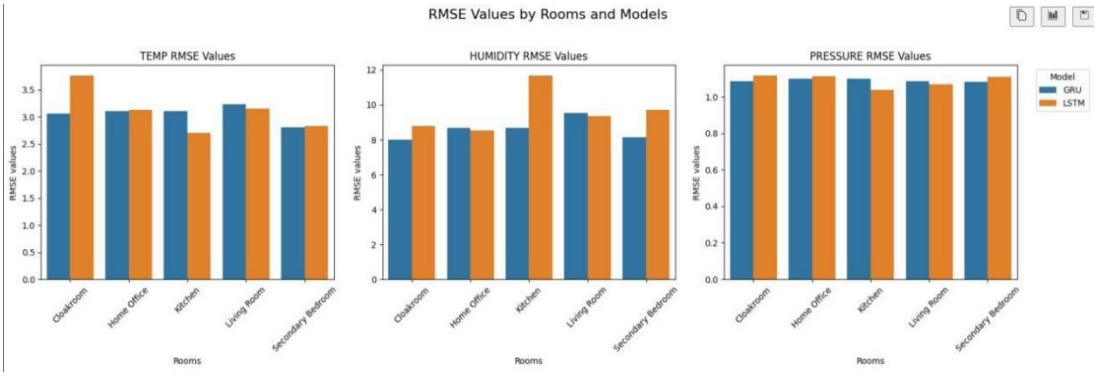


Figure 3.3. RMSE values

RMSE is obtained by taking the square root of MSE and, like MSE, is sensitive to large errors. The smaller the value, the better the model's performance. In temperature predictions, GRU models generally have lower RMSE values, indicating better performance. In humidity predictions, LSTM models typically have higher RMSE values, showing that GRU models perform better. In pressure predictions, GRU models generally have lower RMSE values, indicating better performance.

The data indicate that GRU models outperform LSTM models in predicting temperature, humidity, and pressure. GRU models consistently achieved lower values across all three metrics (MAE, MSE, RMSE), demonstrating superior performance on this dataset. However, it's important to also consider other factors such as training time and computational cost when choosing a model.

4.CONCLUSION

With the Industrial Revolution, the first automation devices profoundly changed human life. Industrial automation systems emerged during this period and became one of the pioneering systems with significant impact. Since the 2000s, automation systems have continued to transform our lives by being used in various fields. Automation systems have seen increased usage in areas such as automatic driving and transportation, energy management and smart grids, building automation and smart home technologies, telecommunications and network management, as well as health and medical technologies.

Meanwhile, we are currently experiencing a new technological revolution with the rapid spread of deep learning methods since 2017. Deep learning plays a significant role in many areas of life, including image recognition and classification, natural language processing, game and gaming strategies, speech recognition and synthesis, medical image processing and diagnosis, automatic driving and autonomous vehicles, as well as advertising and marketing.

Automation systems operate using real-time data. However, they are prone to errors due to reasons such as sensor failures, short-term changes in user demands, or lack of foresight regarding potential future hazards. Solutions to these kinds of problems can be found using deep learning or machine learning techniques to generate prediction data. Solutions can be achieved through interaction between the user and the automation system. These types of visualization and data processing tasks are performed by SCADA and MES devices at the "supervisory" level of the automation pyramid.

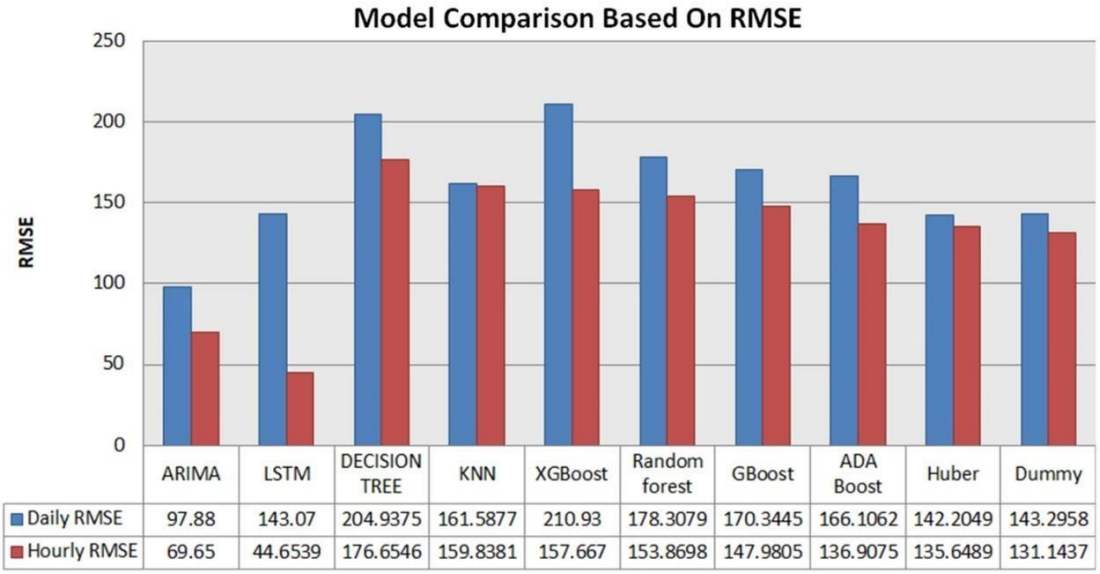
In this project, a web page interface was designed using the Streamlit library. This interface can store some parts of the automation system's modes or generate assignable data.

In my work, I trained deep artificial neural networks according to certain standards. These standards include using 70% of the data for training, 15% for validation, and 15% for testing. I selected appropriate hyper-parameters for each of them. For example, important hyper-parameter selections include the learning rate used for training, the number of epochs, batch size, the number of neurons in the layers, and the depth of the network.

The datasets used for training were different for relative humidity, temperature, and pressure. The reason for this is, as mentioned in the feature engineering section, to prevent over-fitting by excluding values with very close correlation links from training. This situation changes the capacity requirement of the artificial neural network to be used for prediction, and necessitates the creation of either a deeper or shallower network.

Since our work is a time series study, it has focused on GRU and LSTM. Training with a large dataset to predict 3 different variables at 10-minute intervals for each room is quite costly. Therefore, ARIMA, CNN (1D convolution), FCN, and some machine learning models were not attempted. In future studies, I plan to try other models with methods to shorten the training process such as increasing batch size, utilizing better GPUs (A100 GPU was used for this study), or early stopping. In literature, I found a deep learning comparison table as follows:

From: Comparative analysis of Air Quality Index prediction using deep learning algorithms



	ARIMA	LSTM	DECISION TREE	KNN	XGBoost	Random forest	GBoost	ADA Boost	Huber	Dummy
Daily RMSE	97.88	143.07	204.9375	161.5877	210.93	178.3079	170.3445	166.1062	142.2049	143.2958
Hourly RMSE	69.65	44.6539	176.6546	159.8381	157.667	153.8698	147.9805	136.9075	135.6489	131.1437

Figure 4.1. Model Comparison Table

The data indicate that GRU models outperform LSTM models in predicting temperature, humidity, and pressure. GRU models consistently achieved lower values across all three metrics (MAE, MSE, RMSE), demonstrating superior performance on this dataset. However, it's important to also consider other factors such as training time and computational cost when choosing a model.

Air-Smart Controller project has successfully implemented many of the concepts we had in mind. So far, most of the implemented projects have relied on RNNs with good prediction capabilities, hybrid models, and genetic algorithms, but there are very few open-source web interface works available. Upon examining these works, I observed that the structures facilitating the interaction between prediction models and automation devices are either discrete or entirely dependent on the automation system. In order to make people feel closer to the automation system, I aimed to create the Air-Smart Controller tool that can interact with the SCADA and MES model. This

project was particularly special for me because it was not feasible for me to examine open-source structures related to smart home systems, and access to data was limited due to the principle of privacy of personal data. Previous studies were focused on collecting data on temperature, relative humidity, and CO2 levels, but I struggled to find a common project where temperature, relative humidity, and pressure data were simultaneously examined. By completing this project, I not only presented a research where all three categories were examined together on RNNs, but also tried to demonstrate that automation-user interaction can be achieved by designing a user interface with Air-Smart Controller. Data can vary according to different geographies and human demands, and the automation system needs to be interactive. We showed that when we understand user demands, this interaction can be facilitated. At the end of the day, we managed to design an application that enables this interaction. As we closely examined models with better prediction capabilities in the literature, we believe that we can improve model performance and compete with current research.

In a very early study, MinJeong Kim and colleagues (2011) used machine learning to predict and visualize air quality inside the metro. Gaurav Priyadarshi and B. KiranNaik presented prediction studies on KNN in 2022. T. Akilan and K.M.

Baalamurugan conducted a study on early warning systems in agriculture using GRU in 2024. Abdelrazek Elnaggar and colleagues conducted a risk analysis study for a museum in Egypt in 2024, providing insights into the performance expectations of automation systems. All of these studies have laid important foundations for creating more prosperous environments in the future. This study evaluates the use of RNNs in smart home systems with current data and assists in preparing data for automation through the user interface. In our future work, we aim to complete the research with more advanced models to provide more satisfying results.

BIBLOGRAPHY

T. Akilan, K.M. Baalamurugan, “Automated weather forecasting and field monitoring using GRU-CNN model along with IoT to support precision agriculture”,Expert Systems with Applications,Volume 249, Part A,2024,123468,ISSN 0957-4174, doi: 10.1016/j.eswa.2024.123468.

Gaurav Priyadarshi, B. Kiran Naik,“Desiccant coated fin tube energy exchanger design optimization implementing KNN-ML tool and adsorption/desorption kinetics analysis using finite difference based transient model”, International Journal of Thermal Sciences,Volume 192, Part B, 2023,108422,ISSN 1290-0729, doi: 10.1016/j.ijthermalsci.2023.108422.

Catalin Teodosiu, Raluca Hohota, Gilles Rusaouën, Monika Woloszyn, “Numerical prediction of indoor air humidity and its effect on indoor environment, Building and Environment”,Volume 38, Issue 5,2003,Pages 655-664,ISSN 0360-1323, doi: 10.1016/S0360-1323(02)00211-1.

Sulzer, M., Christen, “A. Climate projections of human thermal comfort for indoor workplaces”. *Climatic Change* **177**, 28 (2024). doi: 10.1007/s10584-024-03685-7

Yang, S.; Mahecha, S.D.; Moreno, S.A.; Licina, D. “Integration of Indoor Air Quality Prediction into Healthy Building Design”. *Sustainability* **2022**, *14*, 7890. doi: 10.3390/su14137890

Praveen Kumar Sharma, Ananya Mondal, Shivam Jaiswal, Mousumi Saha, Subrata Nandi, Tanmay De, Sujoy Saha, IndoAirSense: “A framework for indoor air quality estimation and forecasting”, Atmospheric Pollution Research, Volume 12, Issue 1, 2021, Pages 10-22, ISSN 1309-1042, doi: 10.1016/j.apr.2020.07.027.

Challoner, A.; Pilla, F.; Gill, “L. Prediction of Indoor Air Exposure from Outdoor Air Quality Using an Artificial Neural Network Model for Inner City Commercial Buildings”. *Int. J. Environ. Res. Public Health* **2015**, *12*, 15233-15253. doi: 10.3390/ijerph121214975

Kim, M.H., Kim, Y.S., Lim, J. *et al.* “Data-driven prediction model of indoor air quality in an underground space”. *Korean J. Chem. Eng.* **27**, 1675–1680 (2010). doi: 10.1007/s11814-010-0313-5

H. Xie, F. Ma and Q. Bai, "Prediction of Indoor Air Quality Using Artificial Neural Networks," *2009 Fifth International Conference on Natural Computation*, Tianjian, China, 2009, pp. 414-418, doi: 10.1109/ICNC.2009.502.I. Desauziers, V., Bourdin, D., Mocho, P. *et al.* Innovative tools and modeling methodology for impact prediction and assessment of the contribution of materials on indoor air quality. *Herit Sci* **3**, 28

(2015). doi: 10.1186/s40494-015-0057-y

Saman Taheri, Ali Razban,“Learning-based CO2 concentration prediction: Application to indoor air quality control using demand-controlled ventilation, Building and Environment”,Volume 205,2021,108164, ISSN 0360 1323, doi: 10.1016/j.buildenv.2021.108164.

MinJeong Kim, B. SankaraRao, OnYu Kang, JeongTai Kim, ChangKyoo Yoo, “Monitoring and prediction of indoor air quality (IAQ) in subway or metro systems using season dependent models”,Energy and Buildings,Volume 46,2012,Pages 48-55, ISSN 0378-7788, doi: 10.1016/j.enbuild.2011.10.047.

Lu, L., Huang, X., Zhou, X. *et al.* “High-performance formaldehyde prediction for indoor air quality assessment using time series deep learning”. *Build. Simul.* **17**, 415–429 (2024). doi: 10.1007/s12273-023-1091-4

Mishra, A., Gupta, “Y. Comparative analysis of Air Quality Index prediction using deep learning algorithms”. *Spat. Inf. Res.* **32**, 63–72 (2024). doi: 10.1007/s41324-023-00541-1

Wang, C., Li, X., Sun, W. et al. Occupant behavior, thermal environment, and appliance electricity use of a single-family apartment in China. Sci Data.

Ioffe, S. amp; Szegedy, C.. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.Proceedings of the 32nd International Conference on Machine Learning,Proceedings of Machine Learning Research 37:448-456 Available from https://proceedings.mlr.press/v37/ioffe15.html.

D. Cort´es, J. Ram´ırez, L. Villag´omez, R. Batres, V. Vasquez-Lopez and A. Molina, ”Digital Pyramid: an approach to relate industrial automation and digital twin concepts,” 2020 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC), Cardiff, UK, 2020, pp. 1-7, doi: 10.1109/ICE/ITMC49519.2020.9198643.

Marc-Fabian K¨orner, Dennis Bauer, Robert Keller, Martin R¨osch, Andreas Schlereth, Peter Simon, Thomas Bauernhansl, Gilbert Fridgen, Gunther Reinhart, Extending the Automation Pyramid for Industrial Demand Response,Procedia CIRP, Volume 81, 2019,Pages 998-1003,ISSN 2212-8271,doi: 10.1016/j.procir.2019.03.241.

Mishra, A., Gupta, Y. Comparative analysis of Air Quality Index prediction 18using deep learning algorithms. Spat. Inf. Res. 32, 63–72 (2024). https://doi.org/10.1007/s41324-023-00541-1