HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
School of Information and Communication Technology

AIR QUALITY PREDICTION USING WEATHER FORECAST INFORMATION

Introduction to Data Science

152473 - IT4142E
Semester 2024.1

Group 22	**Student ID**
Nguyễn Trọng Phương Bách	20225473
Vũ Minh Hiếu	20225494
Bùi Nguyên Khải	20225501
Trần Ngọc Quang	20225523
Nguyễn Việt Tiến	20225533

Supervisor

Assoc. Prof. Than Quang Khoat

Hanoi, December 2024

Contents

Abstract

Air pollution poses significant threats to public health, ecosystems, and climate, making accurate air quality prediction a critical area of study. This report explores the potential of using weather data to predict air quality across Vietnamese provinces. A comprehensive dataset combining hourly weather attributes (e.g., temperature, humidity, precipitation, wind speed) and air pollutants (e.g., PM2.5, PM10, CO, NO_2, O_3) from November 2020 to November 2024 was analyzed. The study employed rigorous time series analysis techniques, including STL decomposition and the Augmented Dickey–Fuller (ADF) test, to validate stationarity and uncover temporal dependencies. Autocorrelation (ACF) and partial autocorrelation (PACF) plots were utilized to determine optimal lag structures, while STL decomposition revealed key patterns and external influences, such as the Yagi storm. Multiple predictive models, including Random Forest, GRU, and ConvLSTM, were implemented to forecast air quality based on weather data. Results indicate that Random Forest achieved the best performance, with the lowest RMSE and MAE for most air quality indicators, while ConvLSTM effectively captured spatial-temporal dynamics. These findings confirm the strong predictive power of weather data for forecasting air pollutants. The integration of weather forecasts into air quality management systems offers significant potential for improving environmental health outcomes and informing policy decisions.

This study establishes a robust framework for using meteorological data to predict air quality, bridging statistical analysis and machine learning techniques. Future research can extend this work by incorporating external factors such as industrial emissions and urbanization trends to enhance predictive accuracy and scalability.

1 Introduction

Air quality has become a pressing concern in recent years due to its direct impact on public health, ecosystems, and climate. As industrialization and urbanization continue to rise, accurate air quality prediction has emerged as a critical area of study. Predicting air quality requires understanding the complex interplay between various meteorological factors and air pollutants, which necessitates the use of advanced data-driven methodologies.

In this study, we aim to predict air quality using weather forecast information. By leveraging hourly weather and air quality data collected from November 2020 to November 2024 across Vietnamese provinces, this report explores the interdependencies between atmospheric pollutants and meteorological conditions. The integration of air quality variables, such as particulate matter (PM2.5, PM10), CO, NO_2, SO_2, and O_3, with weather attributes, such as temperature, humidity, wind speed, and precipitation, provides a rich dataset for predictive modeling.

The primary objectives of this study are:

- To preprocess and analyze air quality and weather datasets for identifying trends, seasonality, and dependencies.
- To test the stationarity of the time series data using statistical techniques such as the Augmented Dickey–Fuller (ADF) test.
- To perform STL decomposition for understanding the seasonal, trend, and residual components of the data.
- To develop and evaluate predictive models for air quality using weather data, leveraging methods like Random Forest, GRU, and ConvLSTM.

By combining traditional time series analysis with modern machine learning techniques, this study seeks to advance the understanding and forecasting of air quality. The insights gained from this research are expected to enhance predictive accuracy, inform policy decisions, and contribute to mitigating the adverse effects of air pollution.

2 Data Preparation

2.1 Data Scrapping

We utilize the APIs of OpenWeatherMap and OpenMeteo to collect hourly weather and air quality data from November 2020 to November 2024 of Vietnamese provinces. Additionally, we use the Simplemaps API to ensure accurate naming of the 63 provinces. The air qualities include:

- The measurements of air pollutants' concentration such as CO (Carbon monoxide), NO_2 (Nitrogen dioxide), O_3 (Ozone), SO_2 (Sulfur dioxide), PM 2.5 (2.5 microns Particulate Matter), PM 10 (10 microns Particulate Matter) with measurement unit: $\mu g/m^3$
- AQI (Air quality index): We choose the AQI scale of Mainland China and it's measured using two steps:
  - The IAQI (Individual Air Quality Index) are explained using the following formula and table:

$$I = \frac{I_{high} - I_{low}}{C_{high} - C_{low}}(C - C_{low}) + I_{low}$$

where:

I = the Individual Air Quality Index,

C = the pollutant concentration,

C_{low} = the concentration break point that is $\leq C$,

C_{high} = the concentration break point that is $\geq C$,

I_{low} = the index breakpoint corresponding to C_{low},

I_{high} = the index breakpoint corresponding to C_{high},

IAQI	**SO_2, 24 hour**	**SO_2, 1 hour**	**NO_2, 24 hour**	**NO_2, 1 hour**	**PM_{10}, 24 hour**	**CO, 24 hour**	**CO, 1 hour**	**O_3, 1 hour**	**O_3, 8 hour**	**$PM_{2.5}$, 24 hour**
0	0	0	0	0	0	0	0	0	0	0
50	50	150	40	100	50	2	5	160	100	35
100	150	500	80	200	150	4	10	200	160	75
150	475	650	180	700	250	14	30	300	215	115
200	800	800	280	1200	350	24	40	400	265	150
300	1600	(2)	565	2340	420	36	60	800	800	250
400	2100	(2)	750	3090	500	48	90	1000	(3)	350
500	2620	(2)	940	3840	600	60	120	1200	(3)	500

Notes:
(1) SO_2, NO_2, CO 1 hour average concentrations are only for real time reporting. For daily reports use 24 hour average concentration.
(2) If the SO_2 concentration exceeds 800 $\mu g/m^3$ use the index from the 24 hour concentration instead.
(3) If the O_3 concentration exceeds 800 $\mu g/m^3$ use the index from the 1 hour concentration instead.

- The pollutant with the highest IAQI from the table is used to calculate AQI Index, with the scale ranging from 1 to 6.

AQI	Air pollution level	Air pollution category
0-50	Level 1	Excellent
51-100	Level 2	Good
101-150	Level 3	Lightly polluted
151-200	Level 4	Moderately polluted
201-300	Level 5	Heavily polluted
>300	Level 6	Severely polluted

Table 1: Air Quality Index (AQI) Levels and Categories

The weather qualities include:

- temperature_2m (Measured in Degrees Celsius): Temperature measured at 2 meters above land surface.
- relative_humidity_2m (Measured in percent): Relative humidity at 2 meters above ground.
- dew_point_2m (Measured in Degrees Celsius): Dew point temperature at 2 meters above ground.
- wind_speed_10m (Measured in km/h): Wind speed at 10 meters above ground.
- wind_direction_10m (Measured in degrees): Wind speed(km/h) and wind direction measured 10 meters above land surface.
- surface_pressure (Measured in hPa or hectopascal): the atmospheric pressure measured at mean sea level.
- cloud_cover (Measured in percent): The total cloud cover as an area fraction.
- precipitation (Measured in mm): Total precipitation (rain, showers, snow) sum of the preceding hour.

The integration of air quality variables and weather variables provides a comprehensive dataset to explore the interdependencies between atmospheric pollutants and meteorological conditions. The goal of this study is to utilize air quality variables as predictors to forecast weather variables, enabling a deeper understanding of their correlations and enhancing the accuracy of weather prediction models.

2.2 Data Cleaning

2.2.1 Data Overview

To ensure accurate and reliable results, it is critical to detect and handle null and negative values effectively. This process involves techniques such as imputation for missing data and filtering or correction for invalid entries. A robust data-cleaning pipeline establishes a strong foundation for training models, improving their ability to predict weather variables accurately using air quality data.

	Air quality data	**Weather data**
Null datapoints (Datapoints containing null values)	None	None
Negative datapoints (Datapoints with negative values)	252 (0.01% of total air quality dataset)	3532 (0.1% of total weather dataset)
Datapoints with missing provinces' data	8035 (0.3% of total air quality dataset without negative datapoints)	32984 (1.5% of total weather dataset without negative datapoints)
Number of timestamps with missing provinces' data	617 (1.7% of total timestamps in the cleaned air quality dataset)	580 (1.7% of total timestamps in the cleaned weather dataset)

Table 2: Comparison of Air Quality and Weather Data Quality Issues

The table provides an overview of null, negative, and missing province-related data in the air quality and weather datasets. Notably, there are no null datapoints in either dataset. However, the presence of negative values—252 in the air quality data (0.01% of the total) and 3,532 in the weather data (0.1%)—is a concern, as such values are physically invalid for variables like pollutant concentrations, temperature, and wind speed. These negative values will be removed to prevent inaccuracies in model training. The table also highlights missing province data, with 8,035 instances in the air quality dataset (0.3% of the total without negative values) and 32,984 in the weather dataset (1.5%). While these percentages seem small, they may create spatial inconsistencies in the data, especially if location-specific patterns are relevant to the prediction task. Additionally, when examining timestamps with missing province data, the table reveals that 617 timestamps in the air quality data and 580 timestamps in the weather data (both 1.7%) are incomplete even after filtering negative values. Handling such issues—by interpolation, or removal of affected rows—will be critical to maintaining data integrity and ensuring the accuracy of the model when predicting weather variables based on air quality data.

2.2.2 Methods of interpolation

Interpolation is a widely used technique in data preprocessing to estimate missing values within a dataset based on known values. It works by predicting the missing data points using surrounding observations, ensuring the continuity and integrity of the dataset. In the context of time-series data, such as air quality and weather variables, interpolation is particularly effective because it leverages the temporal sequence of data to make logical estimations.

2.2.2.1 Linear/Time Interpolation In our case that the data points are spaced evenly in time-series data, time-based interpolation and linear interpolation produce similar results because both methods rely on the assumption that changes between consecutive data points occur at a constant rate.

Suppose you have two known values, y_1 and y_2, corresponding to two known points, x_1 and x_2, where $x_1 \leq x \leq x_2$(i.e., you are estimating a value at x between these two points). The formula for linear/time interpolation is [1]:

$$y(x) = y_1 + \left(\frac{x - x_1}{x_2 - x_1}\right) \cdot (y_2 - y_1)$$

Where:

- x_1, x_2 are the known x-values (e.g., time points, positions, or indices),
- y_1, y_2 are the corresponding y-values (e.g., measurements or data points),
- x is the point where you want to interpolate the value of y,
- $y(x)$ is the interpolated value at x.

Steps in Linear Interpolation:

1. Determine the two known data points: Identify the values y_1 and y_2 that correspond to x_1 and x_2, and the point x where you want to estimate a value.
2. Calculate the slope: The rate of change between the two known points is calculated by the difference in the y-values divided by the difference in the x-values.
3. Apply the interpolation formula: Using the formula, estimate the value at x.

Time-based interpolation is particularly effective for time-series data as it explicitly considers the temporal relationship between data points. Its main strength lies in handling irregularly spaced timestamps by proportionally estimating missing values based on their distance in time relative to surrounding observations. This ensures that the continuity and trends of the data are preserved, making it ideal for smooth and gradual processes like temperature, wind speed, or pollutant concentrations. By maintaining the temporal structure, time interpolation provides a consistent and reliable dataset, which is essential for time-dependent models and forecasting tasks.

2.2.2.2 Spline interpolation Spline interpolation uses piecewise-defined functions (splines) to interpolate values. These functions are typically polynomials of low degree. A spline $S(x)$ is defined as a **piecewise polynomial** over subintervals of the data $[x_i, x_{i+1}]$.

$$S(x) = \begin{cases} S_0(x), & x \in [x_0, x_1], \\ S_1(x), & x \in [x_1, x_2], \\ \vdots \\ S_{n-1}(x), & x \in [x_{n-1}, x_n], \end{cases}$$

Each piece $S_i(x)$ is a polynomial of degree k on the interval $[x_i, x_{i+1}]$:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + \cdots + p_i(x - x_i)^k$$

Where:

- k is the order (degree) of the spline.
- $a_i, b_i, c_i, \ldots, p_i$ are the coefficients for the i^{th} interval, determined by constraints (*continuity, smoothness, and boundary conditions*).

Spline interpolation strengths include producing smooth, continuous curves that avoid sharp bends or oscillations, making it ideal for applications requiring smooth transitions between data points. However, its weaknesses include potential overfitting in cases of sparse or noisy data, and it can be computationally expensive for large datasets. Additionally, it may not perform well when there are significant outliers or irregularities in the data.

2.2.3 Accuracy Metric

We use three accuracy metric to evaluate interpolation result, which is

- RMSE: RMSE squares the prediction errors before averaging them. This means that larger errors will have significant impact on the RMSE value, making the metric sensitive to outliers. RMSE emphasizes the importance of accurate predictions for all instances, especially where large errors are unacceptable.

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

  where:

  - y_i is the actual value,
  - $\hat{y}_i$ is the predicted value,
  - n is the total number of data points.

- MAE: MAE calculates the average absolute difference between predicted and actual values, making it straightforward and easy to interpret. The error is in the same unit as the predicted values, which helps in understanding the performance directly. Since MAE treats all errors equally, it provides a more balanced view of model performance without disproportionately penalizing larger errors. Unlike RMSE, MAE is less sensitive to outliers because it does not square the error terms. This can be beneficial when the dataset contains outliers or when the impact of large errors should not dominate the evaluation.

  $$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2}$$

- MAPE: Mean Absolute Percentage Error is a popular metric for evaluating regression models, especially when percentage-based errors are more interpretable. It is defined as:

  $$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100 \tag{3}$$

  When the true value is close to zero, the denominator becomes very small, which can cause the percentage error to become extremely large. This leads to disproportionately large MAPE values that don't accurately reflect the model's performance, making MAPE a poor metric in these cases.

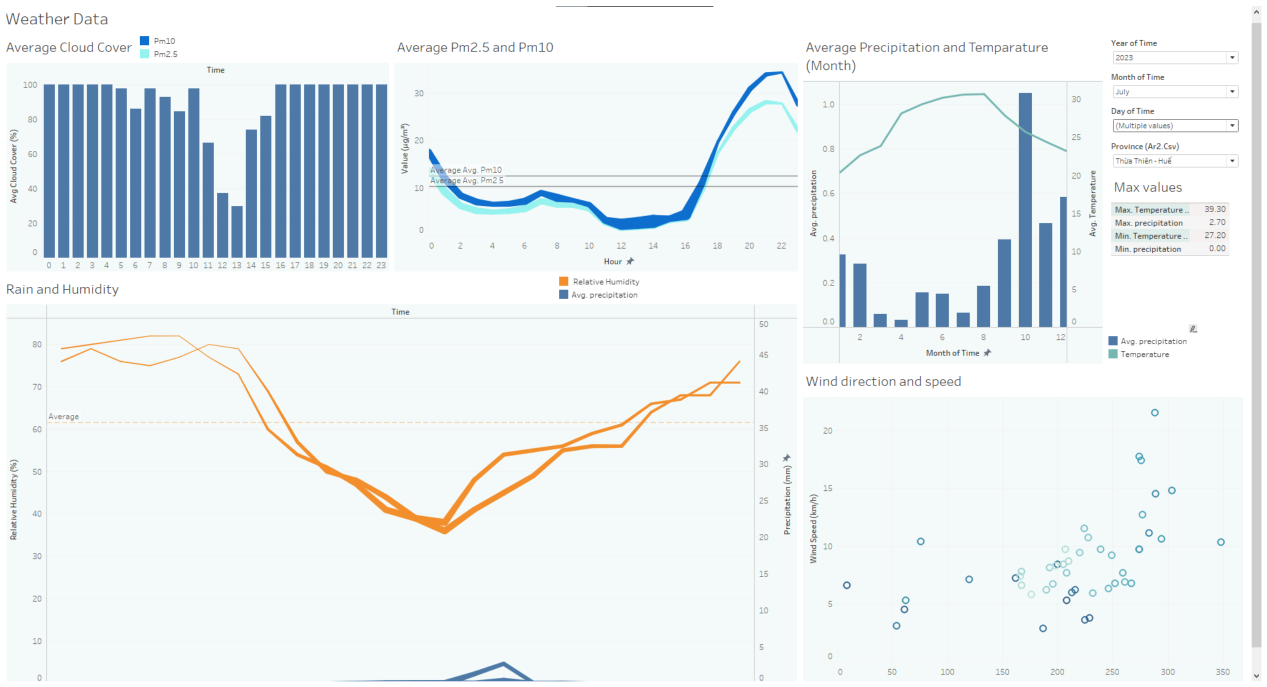
  After evaluating different interpolation techniques, time interpolation yielded the lowest errors compared to spline interpolation. Therefore, we will use time interpolation to interpolate our dataset.

2.3 Data Visualization

2.3.1 Visualization Tool

The data visualization tool selected for this project is Tableau. Tableau is a widely used data visualization platform that enables users to create interactive and dynamic dashboards. It can operate on various data sources, providing a unified framework for data exploration and visualization. Tableau is highly customizable, offering a wide range of built-in visualization options that allow flexibility in designing graphs and charts. These visualizations can be integrated into comprehensive dashboards, enabling users to quickly access, interpret, and derive key insights from the data.

2.3.2 Dashboard

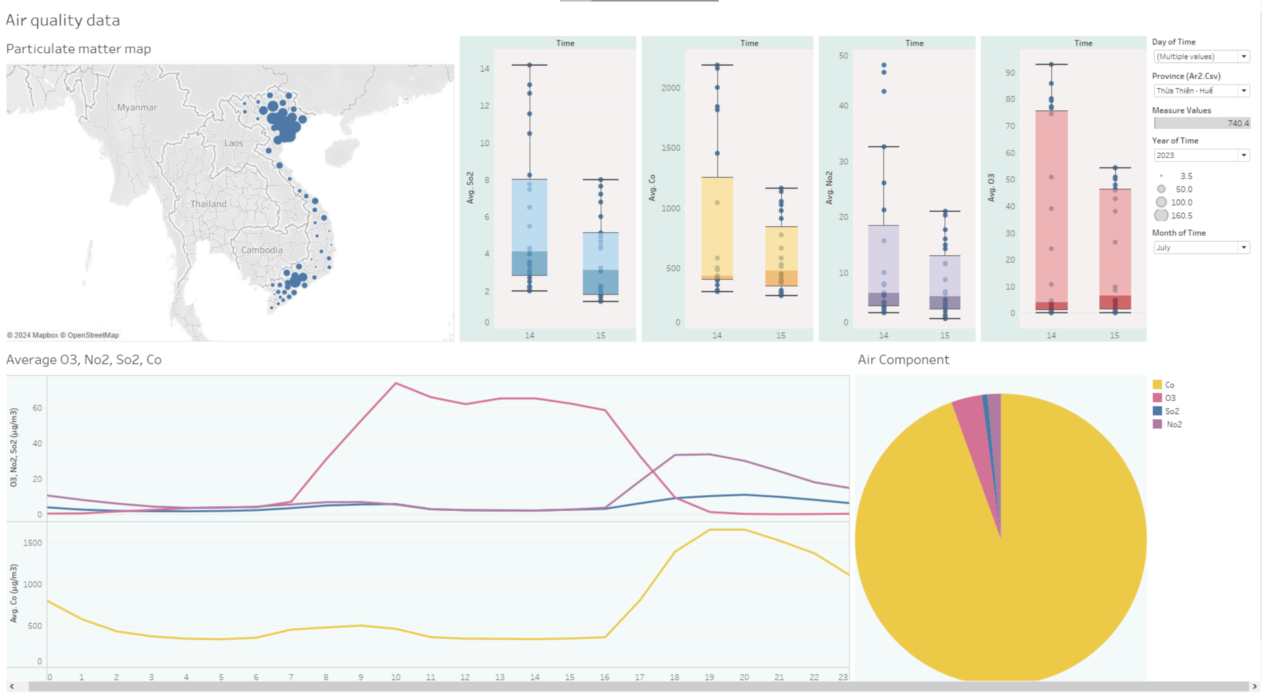


Year of Time: 2023
Month of Time: July
Day of Time: (Multiple values)
Province (Ar2.Csv): Thừa Thiên - Huế

Max values

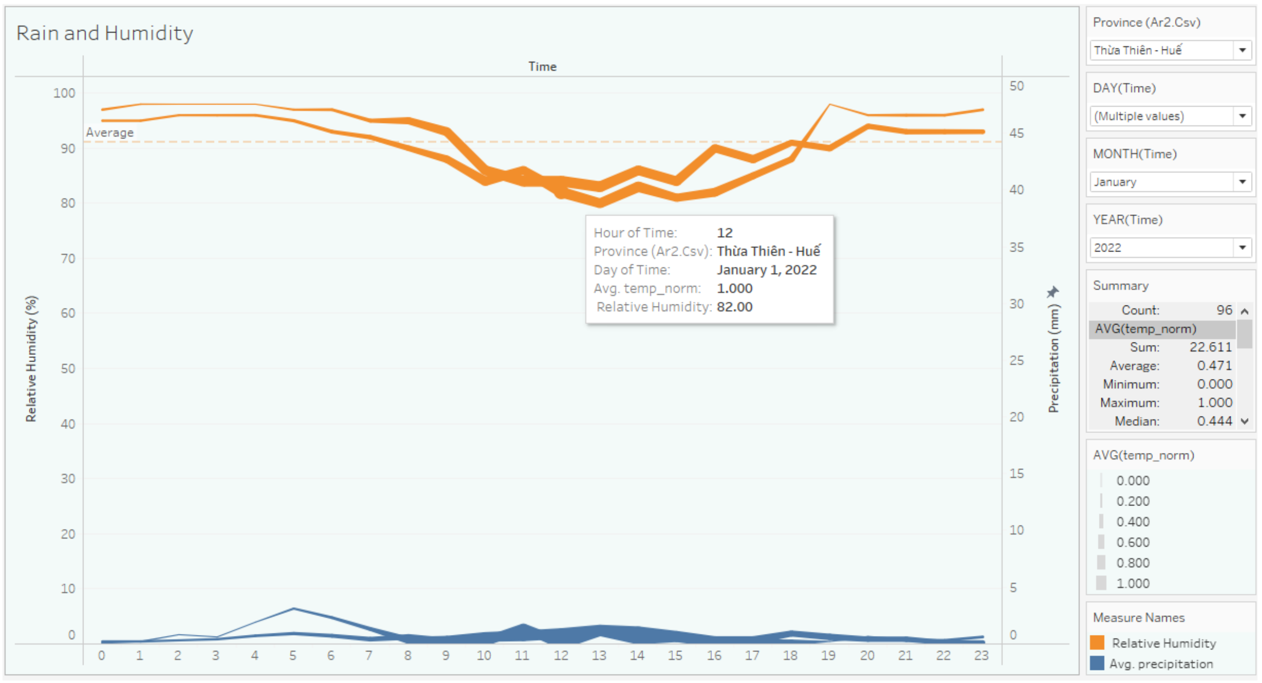
Max. Temperature ..	39.30
Max. precipitation	2.70
Min. Temperature ..	27.20
Min. precipitation	0.00

Figure 1: Weather Data dashboard



Day of Time: (Multiple values)
Province (Ar2.Csv): Thừa Thiên - Huế
Measure Values: 740.4
Year of Time: 2023
3.5
50.0
100.0
160.5
Month of Time: July

Figure 2: Air quality data dashboard



Hour of Time: 12
Province (Ar2.Csv): Thừa Thiên - Huế
Day of Time: January 1, 2022
Avg. temp_norm: 1.000
Relative Humidity: 82.00

Province (Ar2.Csv): Thừa Thiên - Huế
DAY(Time): (Multiple values)
MONTH(Time): January
YEAR(Time): 2022

Summary
Count: 96
AVG(temp_norm)
Sum: 22.611
Average: 0.471
Minimum: 0.000
Maximum: 1.000
Median: 0.444

AVG(temp_norm)
0.000
0.200
0.400
0.600
0.800
1.000

Measure Names
🟧 Relative Humidity
🟦 Avg. precipitation

Figure 3: Hourly Rain and Humidity

2.3.2.1 Rain and Humidity The rain and humidity values are measured and plotted using a line chart. The first Y-axis represents the relative humidity values, corresponding to the orange line, while the second Y-axis displays precipitation (rain amount), represented by the blue line. The line thickness is adjusted according to the normalized temperature with respect to the recording date:

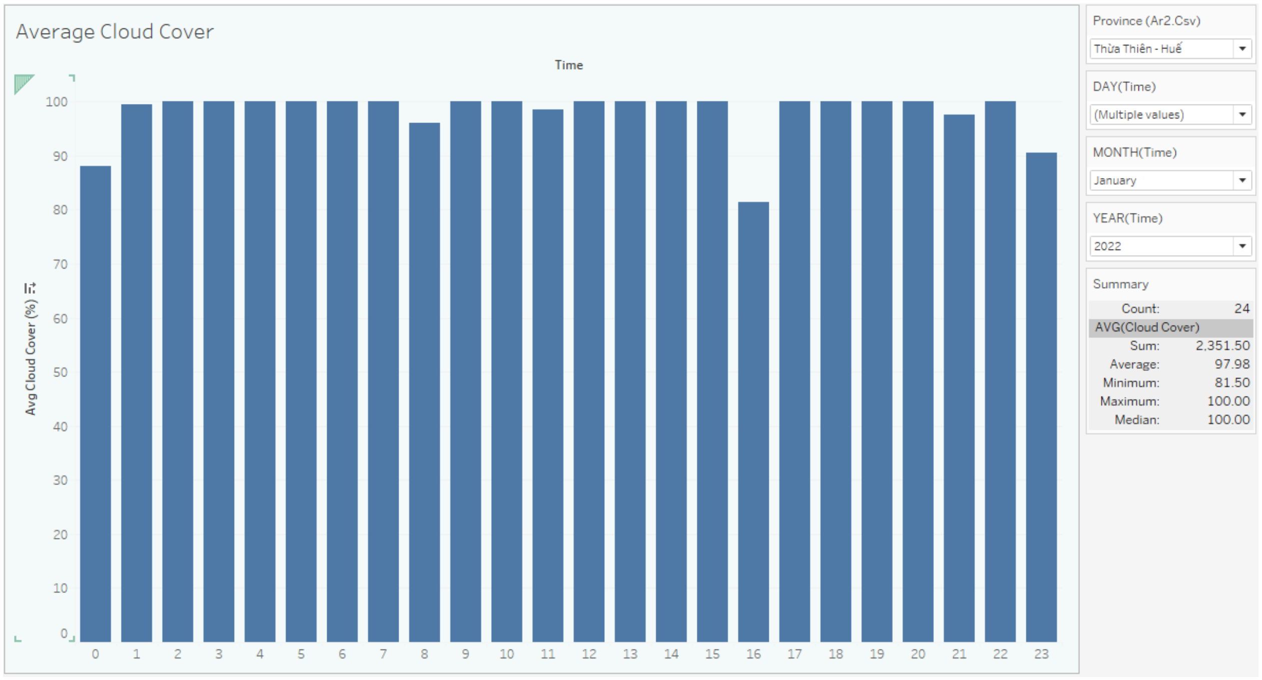
$$\text{Normalized temp}(i) = \frac{\text{Temp}(i,x) - \min(\text{Temp}(x))}{\max(\text{Temp}(x)) - \min(\text{Temp}(x))}, \quad i = \overline{0,23}$$

where i is the hour index and x is the date index.

This use of perceptual channels allows the visualization to encapsulate more information in a single chart, revealing valuable insights. For example, the lines thicken when humidity drops, suggesting an inverse proportional relationship between temperature and humidity.

A dropdown filter on the right enables the selection of days, months, years, and locations (provinces). If multiple records are selected, each is represented by corresponding lines illustrating humidity and precipitation data. Additionally, selecting any data point displays detailed values for the corresponding time stamp.

Statistical summaries of the data are provided below the filters.



Province (Ar2.Csv)
Thừa Thiên - Huế

DAY(Time)
(Multiple values)

MONTH(Time)
January

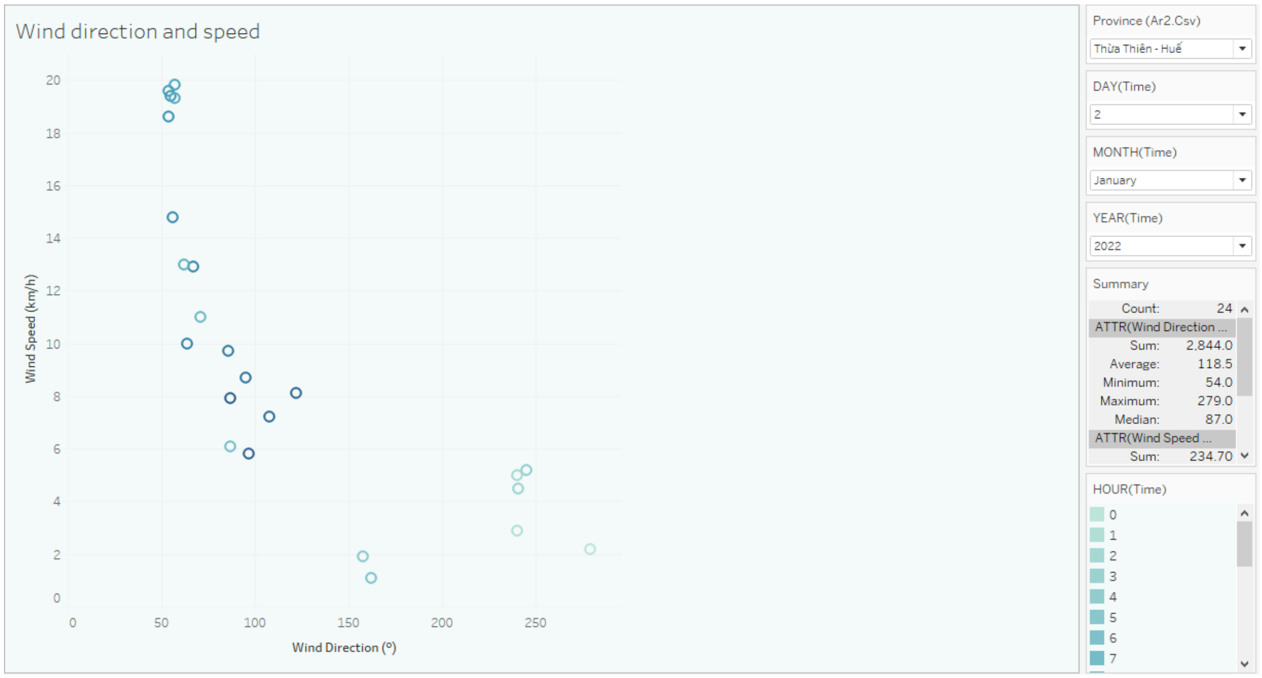
YEAR(Time)
2022

Summary

Count:	24
AVG(Cloud Cover)	
Sum:	2,351.50
Average:	97.98
Minimum:	81.50
Maximum:	100.00
Median:	100.00

Figure 4: Average Cloud Cover

2.3.2.2 Average Cloud Cover The average cloud cover, recorded in percentage, is plotted hourly using a vertical bar chart. Unlike the Rain and Humidity visualization, selecting multiple filter values calculates and displays the average cloud cover across the selected indices. This facilitates the visualization of monthly, yearly, or location-based data summaries.



Province (Ar2.Csv)
Thừa Thiên - Huế

DAY(Time)
2

MONTH(Time)
January

YEAR(Time)
2022

Summary

Count:	24
ATTR(Wind Direction ...	
Sum:	2,844.0
Average:	118.5
Minimum:	54.0
Maximum:	279.0
Median:	87.0
ATTR(Wind Speed ...	
Sum:	234.70

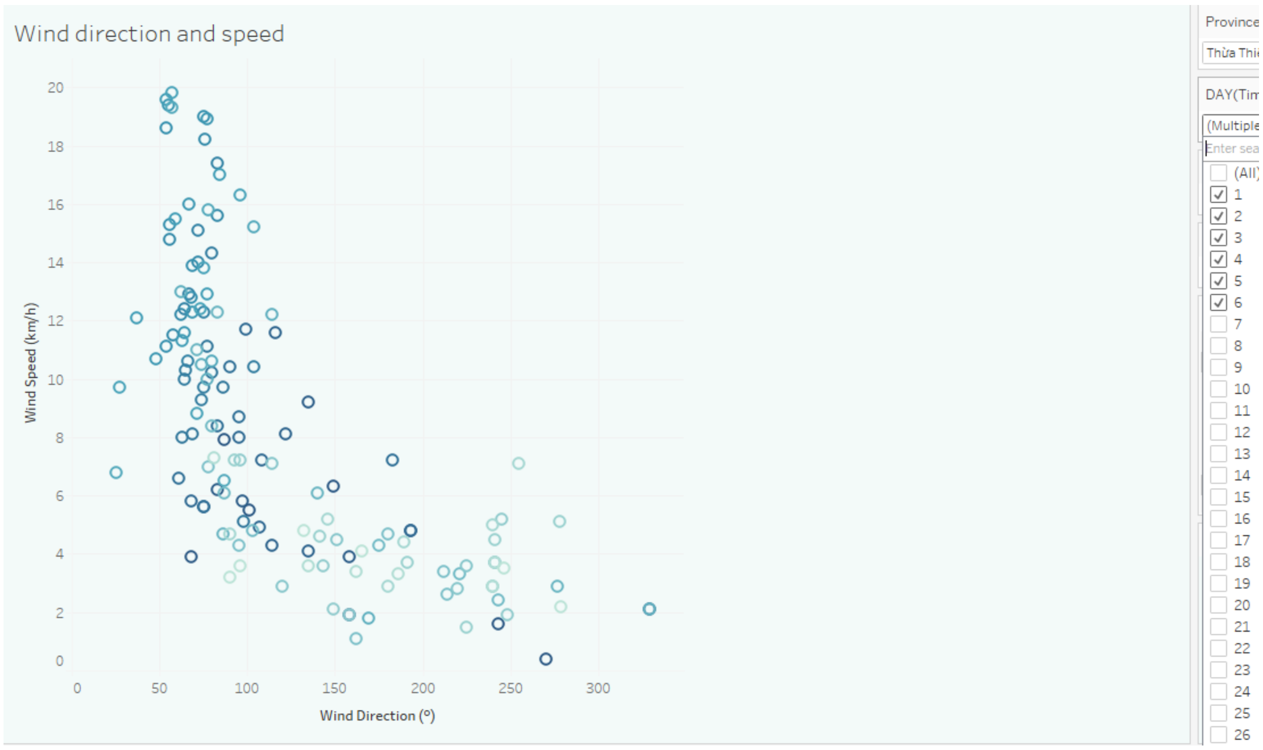
HOUR(Time)

🟦	0
🟦	1
🟦	2
🟦	3
🟦	4
🟦	5
🟦	6
🟦	7

Figure 5: Wind Direction and Speed

2.3.2.3 Wind Direction and Speed Wind direction and speed are visualized using a scatter plot, where the X-axis represents wind direction and the Y-axis represents wind speed. The color intensity of the scatter points encodes the time of day (0-23 hours), with stronger colors indicating later times.

Selecting multiple filter values produces more data points, enabling the identification of wind patterns in dense datasets. For instance, a curve pattern can emerge in dense data, as illustrated in the following example:

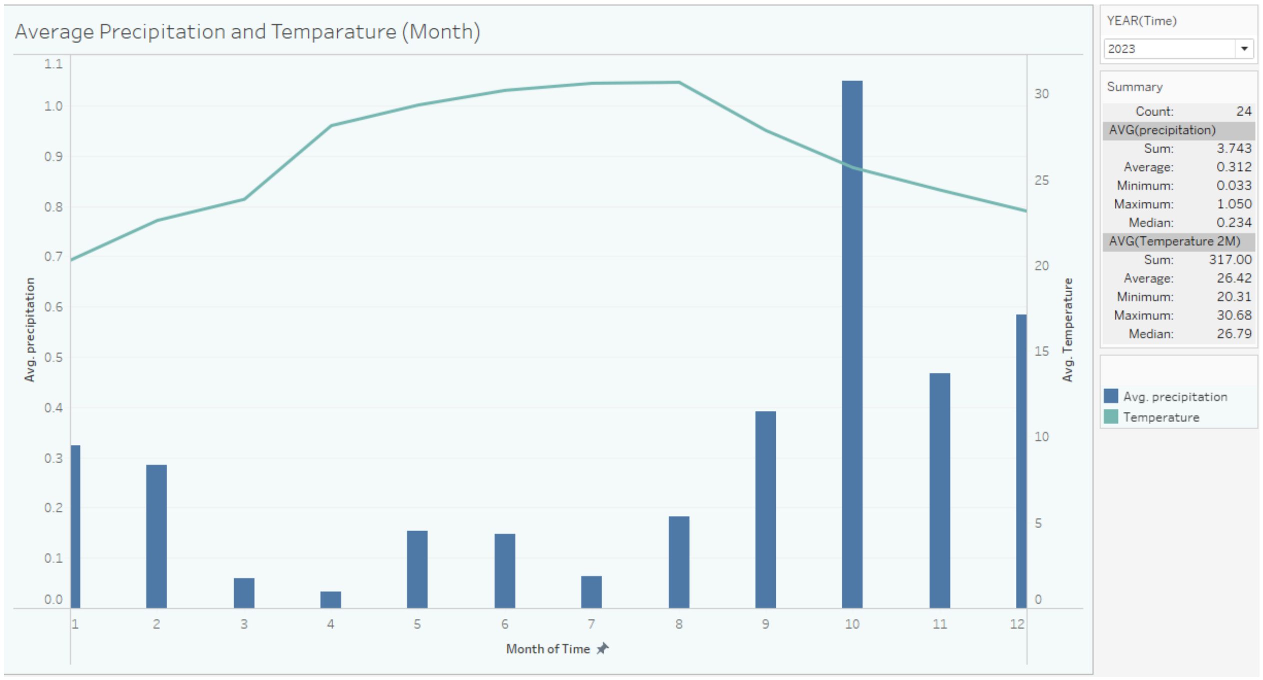


Province
Thừa Thi

DAY(Tim
(Multiple
Enter sea

- [ ] (All)
- [✓] 1
- [✓] 2
- [✓] 3
- [✓] 4
- [✓] 5
- [✓] 6
- [ ] 7
- [ ] 8
- [ ] 9
- [ ] 10
- [ ] 11
- [ ] 12
- [ ] 13
- [ ] 14
- [ ] 15
- [ ] 16
- [ ] 17
- [ ] 18
- [ ] 19
- [ ] 20
- [ ] 21
- [ ] 22
- [ ] 23
- [ ] 24
- [ ] 25
- [ ] 26

Figure 6: A Curve Pattern in Data from January 1 to 5



YEAR(Time)
2023

Summary

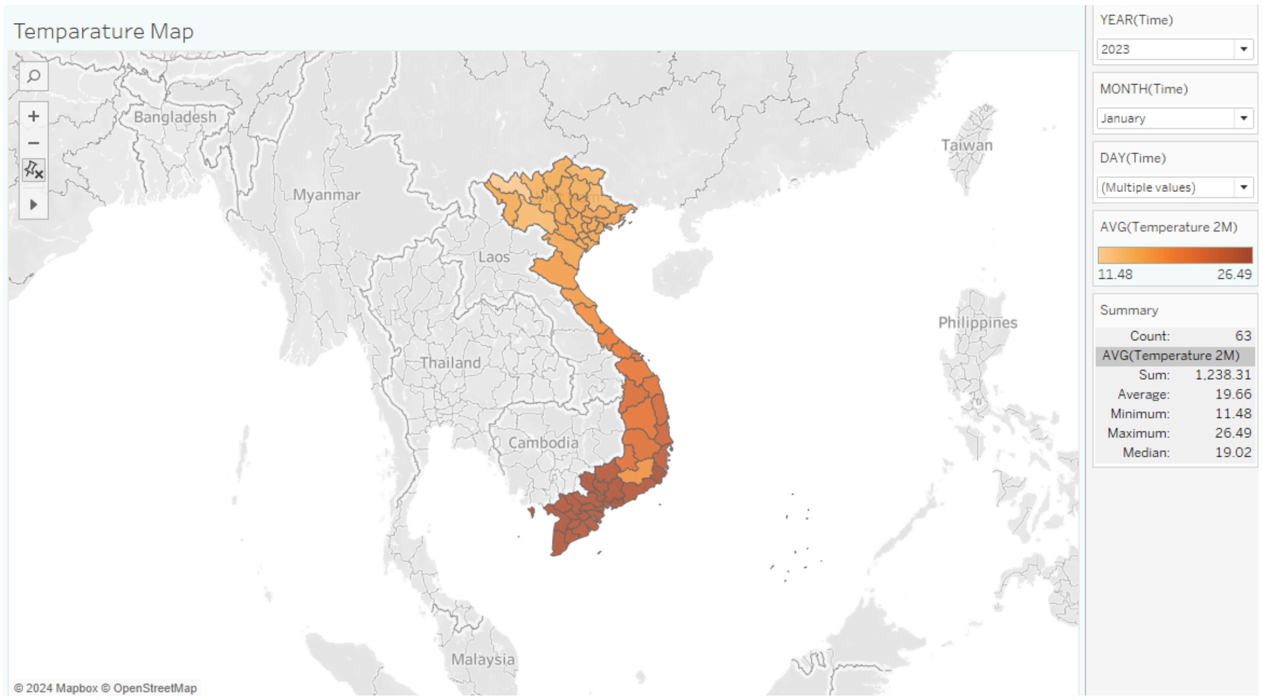
Count:	24
AVG(precipitation)	
Sum:	3.743
Average:	0.312
Minimum:	0.033
Maximum:	1.050
Median:	0.234
AVG(Temperature 2M)	
Sum:	317.00
Average:	26.42
Minimum:	20.31
Maximum:	30.68
Median:	26.79

🟦 Avg. precipitation
🟩 Temperature

Figure 7: Average Precipitation and Temperature

2.3.2.4 Average Precipitation and Temperature The average precipitation and temperature are summarized monthly and plotted using a combined bar and line chart. The first Y-axis represents average precipitation, corresponding to the bars, while the second Y-axis represents temperature, corresponding to the line. The X-axis marks the twelve months of the selected year.

If multiple years are selected, Tableau calculates and displays the combined average data on the graph.

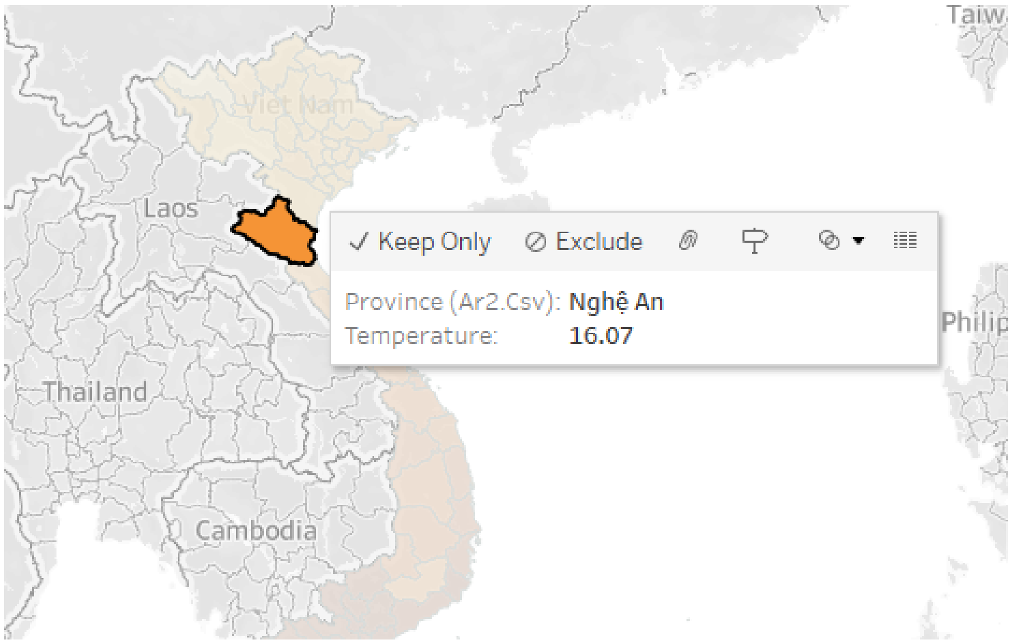


Summary

Count:	63
AVG(Temperature 2M)	
Sum:	1,238.31
Average:	19.66
Minimum:	11.48
Maximum:	26.49
Median:	19.02

Figure 8: Temperature Map

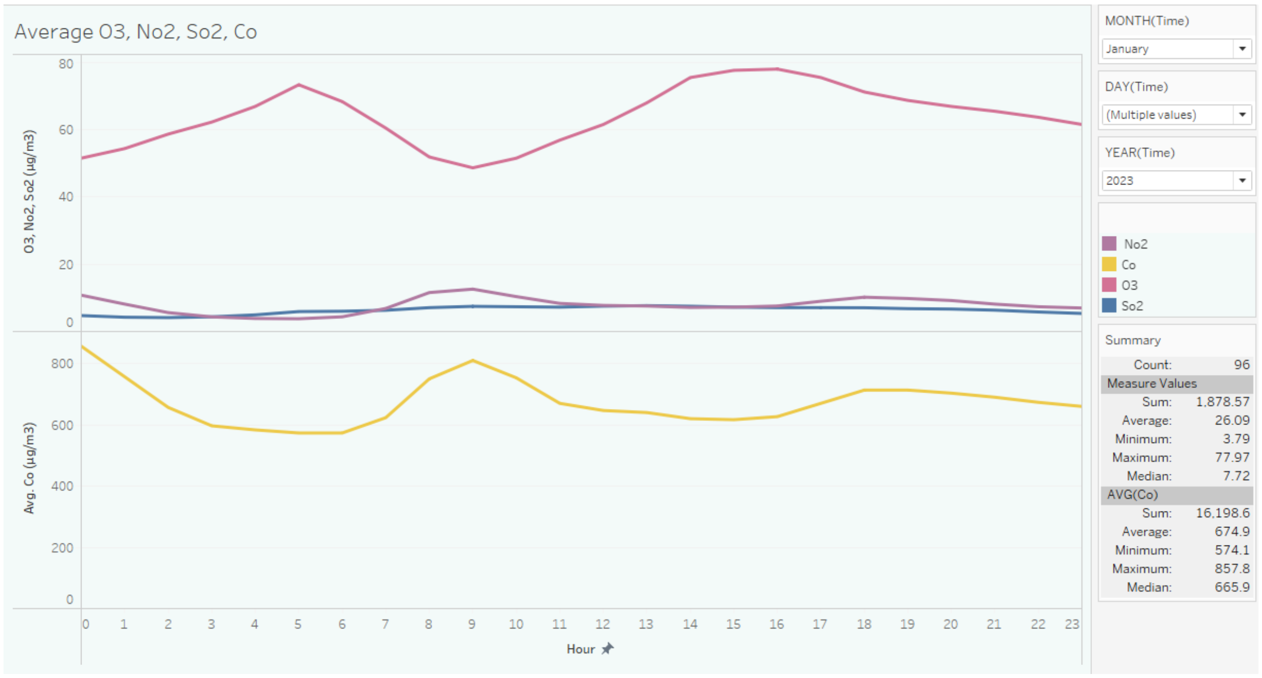
2.3.2.5 Temperature Map The temperature map shows the average daily temperature for each province, with color intensity encoding the temperature values. Users can interact with the map by selecting a province to display detailed temperature information.



Province (Ar2.Csv): Nghệ An
Temperature: 16.07

Figure 9: Detailed Province Temperature Display

Selecting multiple filter values displays combined averages, facilitating monthly or yearly temperature data analysis.



MONTH(Time)
January

DAY(Time)
(Multiple values)

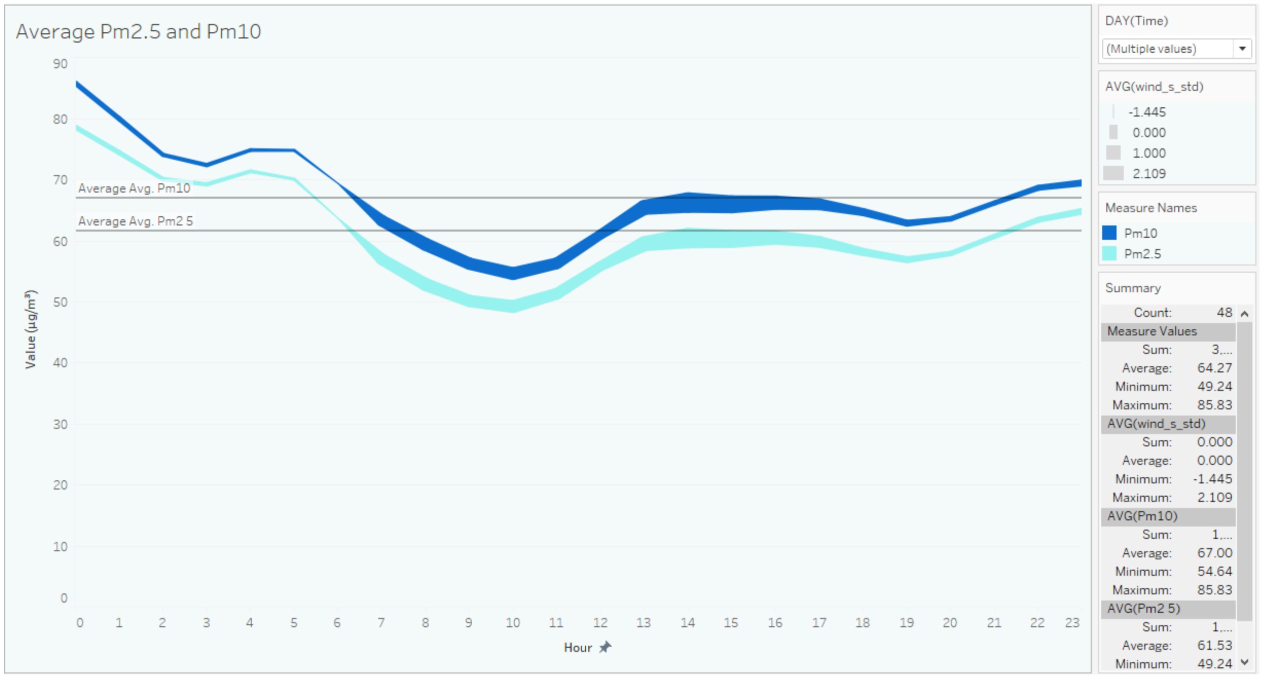
YEAR(Time)
2023

Summary

Count:	96
Measure Values	
Sum:	1,878.57
Average:	26.09
Minimum:	3.79
Maximum:	77.97
Median:	7.72
AVG(Co)	
Sum:	16,198.6
Average:	674.9
Minimum:	574.1
Maximum:	857.8
Median:	665.9

Figure 10: Average O_3, NO_2, SO_2, and CO

2.3.2.6 Average O_3, NO_2, SO_2, and CO The average concentrations of O_3, NO_2, SO_2, and CO are recorded hourly and plotted using two line charts due to the magnitude differences between CO and the other components. Consistent color encoding is applied across the dashboard to ensure unity and ease of interpretation.



DAY(Time)
(Multiple values)

AVG(wind_s_std)
-1.445
0.000
1.000
2.109

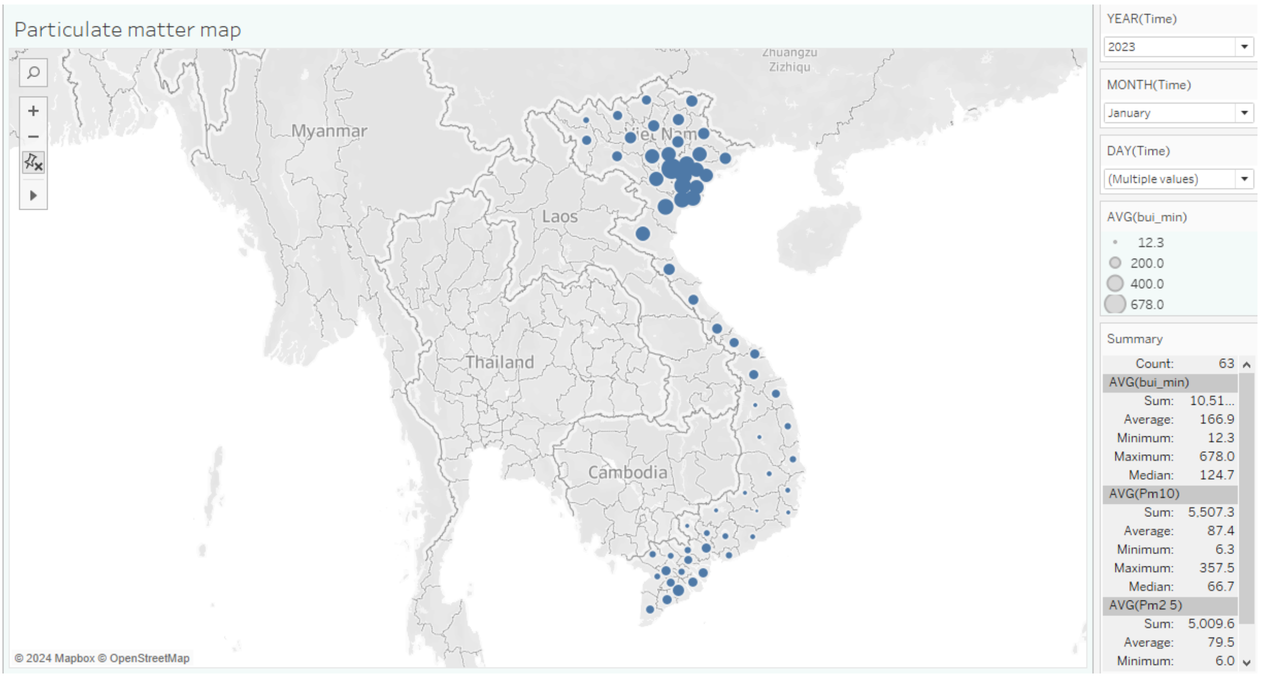
Measure Names

Summary

Count:	48
Measure Values	
Sum:	3,...
Average:	64.27
Minimum:	49.24
Maximum:	85.83
AVG(wind_s_std)	
Sum:	0.000
Average:	0.000
Minimum:	-1.445
Maximum:	2.109
AVG(Pm10)	
Sum:	1,...
Average:	67.00
Minimum:	54.64
Maximum:	85.83
AVG(Pm2 5)	
Sum:	1,...
Average:	61.53
Minimum:	49.24

Figure 11: Average $PM_{2.5}$ and PM_{10}

2.3.2.7 Average $PM_{2.5}$ and PM_{10} The average $PM_{2.5}$ and PM_{10} values are plotted using line charts with a common Y-axis. An average line is drawn for each attribute, with line thickness adjusted according to the normalized wind speed values. The visualization indicates that an initial increase in wind speed reduces air particulate matter, although levels increase again afterward.



YEAR(Time): 2023

MONTH(Time): January

DAY(Time): (Multiple values)

AVG(bui_min)

●	12.3
●	200.0
●	400.0
●	678.0

Summary

```
        Count:      63
AVG(bui_min)
          Sum:  10,51...
      Average:   166.9
      Minimum:    12.3
      Maximum:   678.0
       Median:   124.7
AVG(Pm10)
          Sum:  5,507.3
      Average:    87.4
      Minimum:     6.3
      Maximum:   357.5
       Median:    66.7
AVG(Pm2 5)
          Sum:  5,009.6
      Average:    79.5
      Minimum:     6.0
```

Figure 12: Particulate Matter Map

2.3.2.8 Particulate Matter Map The particulate matter map shows the sum of average PM_{10} and $PM_{2.5}$ values for each province, using size as the perceptual channel to represent the magnitude.

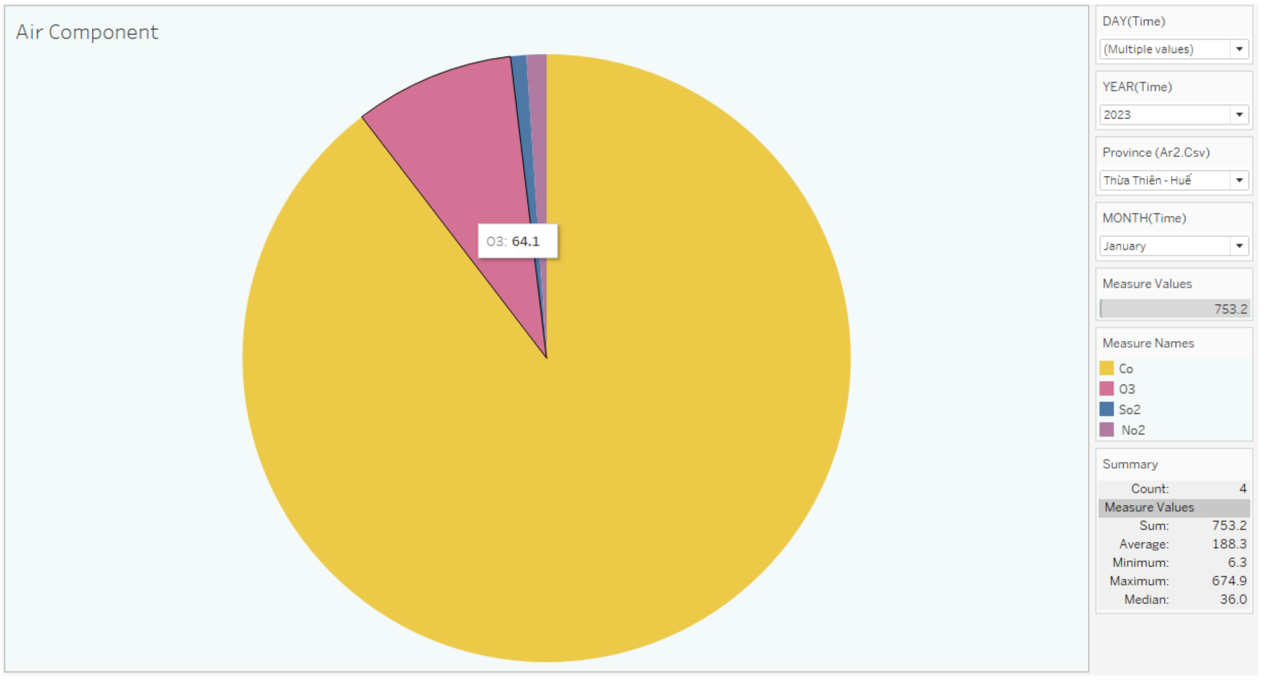


Figure 13: Air Component Pie Chart

2.3.2.9 Air Component The air component pie chart displays the average concentrations of CO, SO_2, O_3, and NO_2 in the air on a given day. Selecting a specific section of the chart reveals the exact values for that measure.

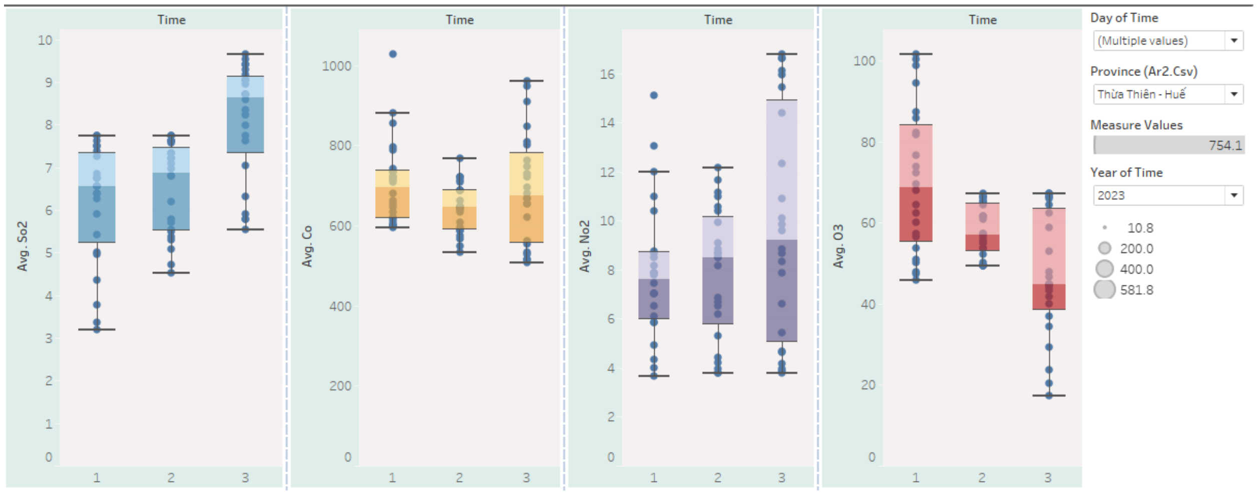


Figure 14: Daily Air Component Box Plots

2.3.2.10 Daily Air Component Box Plot The values of each air component are grouped by date and visualized using box plots. Selecting multiple days in the dropdown menu automatically generates additional box plots for each day. Similarly, selecting multiple provinces adds more data points to the plots. This allows for both day-to-day comparisons and location-based summarization of air components.

2.4 Data Exploration

2.4.1 Time Lags, ACF and PACF

2.4.1.1 Time Lags in Time Series In time series analysis, a **time lag** refers to the delay between a current observation and its past values. For a time series x_t, the lag-1 value corresponds to x_{t-1}, lag-2 to x_{t-2}, and so on. Time lags help capture temporal dependencies, which are crucial for identifying patterns, trends, and building forecasting models. For instance, if today's value influences tomorrow's value, there is a lag-1 dependence.

2.4.1.2 Autocorrelation Function (ACF) The **Autocorrelation Function (ACF)** measures the correlation between a time series and its lagged versions. It is defined as:

$$\text{ACF}(k) = \frac{\sum_{t=k+1}^{n}(x_t - \bar{x})(x_{t-k} - \bar{x})}{\sum_{t=1}^{n}(x_t - \bar{x})^2},$$

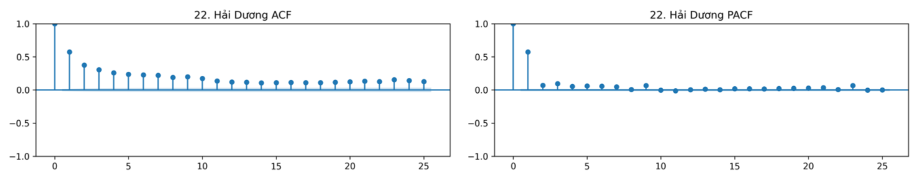
where k is the lag, x_t is the value at time t, and $\bar{x}$ is the mean of the series.

The ACF helps identify significant relationships between current and past values, revealing patterns such as trends, seasonality, or moving average (MA) components in the data. In the ACF plot:

- The x-axis represents the lag values.
- The y-axis represents the autocorrelation coefficients.
- Bars outside the confidence bands indicate statistically significant correlations.

2.4.1.3 Partial Autocorrelation Function (PACF) The **Partial Autocorrelation Function (PACF)** measures the direct correlation between the time series and its lagged values, after removing the influence of intermediate lags. It isolates the effect of a particular lag k. Unlike the ACF, which includes all intermediate lag effects, PACF removes those effects recursively. In the PACF plot:

- The x-axis represents the lag values.
- The y-axis represents the partial autocorrelation coefficients.
- Significant spikes outside the confidence bands indicate the lags that directly influence the series.



ACF and PACF of Precipitation in Hai Duong

2.4.1.4 Plotting ACF and PACF From the ACF plot (left), we observe a gradual decay in the autocorrelation values as the lag increases, which is characteristic of a process with strong autocorrelation. The significant spikes at early lags suggest the presence of persistent relationships over short lags.

On the other hand, the PACF plot (right) shows a significant spike at lag 1 followed by near-zero partial autocorrelations at subsequent lags. This behavior is an early indication of using lag-1, where the current value is mainly influenced by its immediate previous value, and additional lags do not contribute significantly once the lag-1 effect is accounted for.

Overall, after plotting the ACF and PACF plots for all weather and air variables,we concluded that the time series can be modeled effectively using time lagged weather variables.

2.4.2 Stationarity and the Augmented Dickey–Fuller Test

2.4.2.1 Stationarity in Time Series In time series analysis, a time series is said to be **stationary** if its statistical properties, such as the mean, variance, and the relationship (covariance) between observations depends only on the lag between them, not the specific time at which the observations are taken. Stationarity is a crucial assumption in many time series models, as it ensures that the relationships in the data do not change over time, allowing for reliable modeling and forecasting.

2.4.2.2 The Augmented Dickey–Fuller (ADF) Test The **Augmented Dickey–Fuller (ADF) test** is a statistical test used to determine whether a time series is stationary. Specifically, it tests for the presence of a **unit root**, which is a characteristic of non-stationary series.

2.4.2.3 Hypotheses of the ADF Test The ADF test considers the following hypotheses:

- **Null Hypothesis** (H_0): The time series has a unit root (i.e., it is non-stationary).
- **Alternative Hypothesis** (H_1): The time series does not have a unit root (i.e., it is stationary).

2.4.2.4 Test Equation The ADF test is based on the following regression equation:

$$\Delta x_t = \alpha + \beta t + \gamma x_{t-1} + \sum_{i=1}^{p} \delta_i \Delta x_{t-i} + \varepsilon_t$$

where:

- Δx_t is the first difference of the time series at time t, defined as $x_t - x_{t-1}$.
- α is a constant (drift term).
- βt is the trend component (optional).
- γx_{t-1} represents the coefficient for the lagged value of the series.
- p is the number of lags included to account for autocorrelation in the residuals.
- ε_t is the white noise error term.

The key parameter of interest is γ, which determines whether a unit root exists. If γ is significantly different from zero (negative), the null hypothesis is rejected, and the series is deemed stationary.

2.4.2.5 Interpreting the ADF Test Results The results of the ADF test include the following components:

- **Test Statistic**: This value is compared with critical values at 5% significance levels. If the test statistic is smaller (more negative) than the critical value, the null hypothesis is rejected.
- **p-value**: If the p-value is less than the chosen significance level, the null hypothesis is rejected, and the series is considered stationary.

2.4.2.6 Conclusion of the ADF Test We conducted the **Augmented Dickey–Fuller (ADF) test** for all variables in the both weather and air dataset to determine their stationarity. Based on the test results, we observed that the p-values for all variables were below the 0.05 significance level. Therefore, we **reject the null hypothesis** (H_0) of a unit root and conclude that the time series data for all variables is **stationary**. With 95% confidence, we confirm that the data meets the stationarity assumption required for further time series modeling and analysis.

2.4.3 STL Decomposition and Outlier Detection

STL (Seasonal-Trend Decomposition using Loess) is a statistical method for decomposing a time series into three main components:

- Seasonal Component (S_t): Represents repeating patterns or periodic fluctuations (e.g., monthly, weekly, or daily patterns).
- Trend Component (T_t): Represents the long-term direction or movement in the data over time.
- Residual Component (R_t): Represents the remaining variability after removing the seasonal and trend components, often referred to as noise or irregularities.

The STL decomposition is expressed as:

$$y_t = S_t + T_t + R_t$$

STL uses a locally weighted regression (Loess) method to iteratively estimate the components:

1. Estimate Seasonal Component (S_t): Remove seasonality by smoothing the data using a Loess filter.
2. Estimate Trend Component (T_t): Smooth the deseasonalized data ($y_t - S_t$) to identify the trend.
3. Estimate Residual Component (R_t): Compute the residual as:

$$R_t = y_t - S_t - T_t$$

After the residual component R_t is computed, it can be used to identify outliers in data. This process is illustrated in the following step:

1. Perform STL Decomposition: Decompose the time series into $y_t = S_t + T_t + R_t$. Residuals are given by:

$$R_t = y_t - (S_t + T_t)$$

2. Compute Residual Threshold: Use the standard deviation (σ) of R_t to set a threshold for identifying outliers:

$$\text{Threshold} = k \cdot \sigma$$

where k is a constant (e.g., $k = 2$ or $k = 3$ for 95% or 99.7% confidence intervals).

3. Identify Outliers: A point is an outlier if:

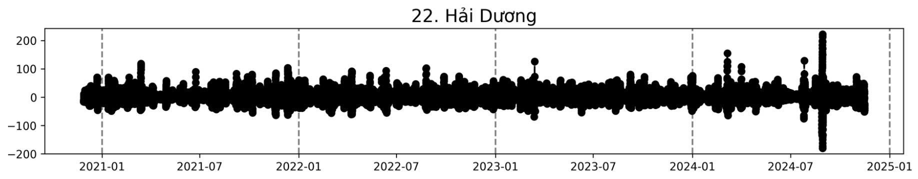
$$|R_t| > \text{Threshold}$$

Alternatively, compute the z-score:

$$Z_t = \frac{R_t - \mu}{\sigma}$$

where μ is the mean of R_t (usually zero in STL). Data points with $|Z_t| > k$ are flagged as outliers.

4. Visualize Residuals:



Residuals graph of PM_{25} in Hai Duong

Here's an example of a residuals graph of PM_{25} in Hai Duong. The residuals plot obtained from the STL decomposition shows a significant spike in September 2024, which aligns with the occurrence of the Yagi storm, a known external event. This spike is not indicative of typical outliers, as it reflects a real and explainable deviation caused by the storm's impact, which disrupted the underlying pattern of the time series. Apart from this event, the residuals remain largely centered around zero, with no other significant deviations, suggesting that the data does not contain unusual or unexplained outliers overall. In general, the spike in the residuals plots are results of natural events, and the data can generally be considered free from outliers.

2.4.4 VAR, Granger Causality and Optimal Lag

2.4.4.1 VAR The Vector Auto-Regressive (VAR) model introduced in [2] is a statistical method used to analyze the linear interdependencies among multiple time series variables. In a VAR model, each variable is expressed as a linear function of its own past values and the past values of other variables in the system. Mathematically, a VAR model of order p for k time series can be represented as:

$$Y_t = c + A_1 Y_{t-1} + A_2 Y_{t-2} + \cdots + A_p Y_{t-p} + \varepsilon_t$$

where:

$$
\begin{aligned}
Y_t &: k \times 1 \text{ vector of time series at time } t \\
c &: k \times 1 \text{ vector of constants} \\
A_i &: k \times k \text{ coefficient matrices for lag } i \\
p &: \text{lag order of the VAR model} \\
\varepsilon_t &: k \times 1 \text{ vector of white noise error terms}
\end{aligned}
$$

2.4.4.2 Granger Causality Granger causality tests whether the past values of one time series (X) help predict another time series (Y) beyond what is possible using the past values of Y alone. It is not true causality but a statistical measure of predictive ability. This test is commonly implemented using the VAR model.

VAR Model for Two Variables Consider two time series X_t and Y_t. The VAR(p) model for these series can be written as:

$$Y_t = c_1 + \sum_{i=1}^{p} a_i Y_{t-i} + \sum_{i=1}^{p} b_i X_{t-i} + \varepsilon_{1t}$$

$$X_t = c_2 + \sum_{i=1}^{p} d_i Y_{t-i} + \sum_{i=1}^{p} e_i X_{t-i} + \varepsilon_{2t}$$

Where:

- Y_t and X_t: The two time series.
- p: The lag order of the VAR model.
- a_i, b_i, d_i, e_i: Coefficients of the lagged variables.
- $\varepsilon_{1t}, \varepsilon_{2t}$: Error terms, assumed to be white noise.

Granger Causality Hypothesis

- **Null Hypothesis (H_0)**: X_t does not Granger-cause Y_t. This implies that the coefficients $b_i = 0$ for all i in the first equation.
- **Alternative Hypothesis (H_a)**: X_t Granger-causes Y_t. This means at least one $b_i \neq 0$.

Testing Procedure

1. **Fit the VAR model**: Estimate the full VAR model with both X_t and Y_t included.
2. **Restricted Model**: Fit a restricted VAR model excluding the lagged values of X_t from the Y_t equation.

3. **F-Test**: Use the F-test to compare the full and restricted models. The test statistic is calculated as:

$$F = \frac{(RSS_r - RSS_u)/m}{RSS_u/(T-k)}$$

   Where:

   - RSS_r: Residual sum of squares from the restricted model.
   - RSS_u: Residual sum of squares from the unrestricted model.
   - m: Number of lagged terms excluded in the restricted model.
   - T: Number of observations.
   - k: Number of parameters in the unrestricted model.

Interpretation

- If the F-test p-value is less than the chosen significance level (e.g., 0.05), reject H_0. This indicates X_t Granger-causes Y_t.
- Otherwise, fail to reject H_0, suggesting no Granger causality.

  After conducting the Granger causality test, we concluded that weather data Granger-causes air quality data with 95% confidence interval. This result indicates that past weather conditions provide statistically significant information to predict future air quality, suggesting a causal relationship between the two variables and reinforces the hypothesis that weather influences air pollution levels.

Optimal Lag We can also select the optimal lag given the VAR model with both X_t and Y_t included by using the Hannan-Quinn Information Criterion (HQIC), which is designed to find a balance between model complexity and the goodness of fit. The process involves the following steps:

1. **Fit VAR models with different lags:** Start by estimating VAR models for a range of lags. Typically, the range could be from 1 to a maximum value based on the dataset's length and the nature of the variables involved.
2. The **Hannan–Quinn Information Criterion (HQIC)** is given by:

$$\text{HQIC} = -2\ln(\mathscr{L}) + 2k\ln(\ln(n))$$

   where:

   - $\mathscr{L}$ is the likelihood of the estimated model.
   - k is the number of estimated parameters in the model.
   - n is the number of observations in the dataset.

   The first term, $-2\ln(\mathscr{L})$, penalizes the lack of fit in the model, while the second term, $2k\ln(\ln(n))$, accounts for model complexity.

3. **Identify the knee location of HQIC:** Plot the HQIC values against the lag order. The “knee" of the plot is the point where the HQIC stops decreasing sharply and starts to level off. This knee represents a balance between increasing model complexity (which reduces the HQIC) and the diminishing returns of adding more lags (which may lead to overfitting).

4. **Select the optimal lag:** The optimal lag is typically chosen where the HQIC reaches its minimum or the knee point, as this suggests a good fit without overcomplicating the model. Using this approach ensures that the model is sufficiently complex to capture the underlying dynamics, but not so complex that it overfits the data.

3 Methodology

In this study, we aim to predict air quality using weather data by employing a variety of machine learning and deep learning techniques. The methodology is built around three distinct models: Random Forest, GRU (Gated Recurrent Unit), and Convolutional LSTM (ConvLSTM). Random Forest, a powerful ensemble learning method, is used for its robustness in handling both linear and non-linear relationships in the data. GRU, a type of recurrent neural network, captures temporal dependencies in time-series data, making it well-suited for our sequential weather data. Lastly, ConvLSTM combines the strengths of convolutional networks and LSTM (Long Short-Term Memory) networks, allowing us to model both spatial and temporal patterns in the weather data. These models are evaluated and compared to determine the most effective approach for air quality prediction.

3.1 Sliding Window

Sliding window split is a common technique used in time series data to prepare the dataset for training machine learning models. It involves splitting the time series into overlapping windows (subsequences) of data points. This technique is particularly useful for time series forecasting tasks, where the model is trained on historical data to predict future values. To build the dataset for Random Forest and GRU Network, we iterated through our two data folder, “weather” and “air quality”. Every file in the folders contains weather information of a single province, and a series of sliding windows is generated from each pair of files. The process is described below:

- Check the time index of the file and iterate through it with a custom function. Every time n consecutive timestamp is detected, a small window containing all attributes from timestamp t - n + 1 to t is recorded. For example, if the window size is 5, window 1 will be from data points 0 to 4, window 2 will be from data points 1 to 5. Moreover, a valid window is a window in which, every timestamp from t-n+1 to t exist in both “weather” file and “air quality” file.

- If a missing timestamp is discovered (due to error when scraping, source data missing,. . .), the function will skip through that point and jump to the next timestamp, and the count is reset back to zero.

- The process continued until the end of the file, and repeated for every pair of files.

Sliding windows allow a model to learn temporal dependencies within the time series data, by considering a sequence of previous data points (or windows). Because of that, the model can better understand patterns and trends, which is crucial for forecasting values based on corresponding data (in our case, air quality data based on weather data).

3.2 Random Forest

3.2.1 Decision Tree

A decision tree is a tree-like structure where each node represents an attribute to test for a sample, each branch from a node corresponds to a possible value of the attribute associated with that node, and each leaf node represents a class or a final prediction. The decision tree's goal is to create a model to predict the value of the target variable by learning simple IF-THEN decision rules derived from the data attributes.

A learned decision tree predicts a sample by traversing the tree from the root node to a leaf node, where the class label or value associated with that leaf node is used for the prediction. Thus, a tree represents some classification or regression function. In the case of our problem, we use decision trees where the target variable takes on continuous values (usually real numbers) which are called regression trees. [**5**]

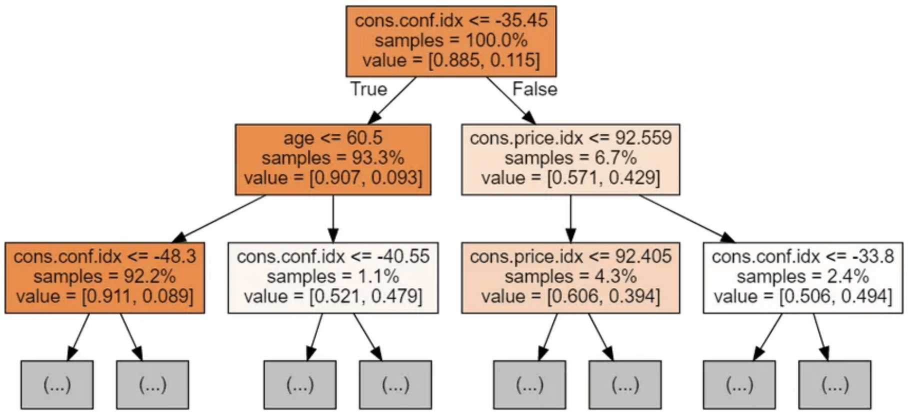


Fig 3.1 Decision tree diagram [3]

3.2.2 Random Forest

When constructing a decision tree, if the depth is left unrestricted, the tree will classify all the data in the training set correctly, which can lead to overfitting. The Random Forest algorithm consists of multiple decision trees, and each decision tree incorporates random elements:

- Randomly sampling data to construct the decision tree.
- Randomly selecting features to place at a node.

Since each decision tree in the Random Forest algorithm does not use all the training data, nor does it use all the features of the data to construct the tree, each tree may not perform well individually. As a result, each decision tree model is not overfitted but may be underfitted, or in other words, the model has high bias. However, the final result of the Random Forest algorithm is an ensemble of many decision trees, so the information from each tree complements the others, leading to a model with low bias and low variance, or in other words, a model with good predictive performance.

Random Forest Algorithm

Assume the training dataset D consists of n data samples, and each sample has d attributes. We learn K decision trees as follows:

1 Create K trees, with each tree generated as follows:

  a Build a subset D_i by randomly sampling (with replacement) from D.

  b Train the i^{th} tree from D_i.

  c At each node during the tree construction process:

    i Randomly select a subset of attributes.

    ii Split the tree based on this subset of attributes.

  d This tree will be grown to its maximum size without pruning.

2 Each subsequent prediction is obtained by averaging the predictions from all the trees.

To construct a subset D_i from D, we randomly sample n data points from the training dataset using Bootstrapping. This means that once a data sample is selected, it is not removed from the original dataset but remains available for further sampling until n samples have been selected. The Random Forest algorithm consists of multiple decision trees, each built using the Decision Tree algorithm on different subsets of data and different subsets of attributes. The final prediction of the Random Forest algorithm is then aggregated from the predictions of all the constructed decision trees.

3.3 LSTM

In theory, classic RNNs can keep track of arbitrary long-term dependencies in the input sequences. However when training a classic RNN using back-propagation, the long-term gradients which are back-propagated can vanish, causing the network to eventually stop learning. Long short-term memory (LSTM)[1] is a type of recurrent neural network (RNN) aimed at mitigating the vanishing gradient problem previously encountered by RNNs. LSTM is widely used for timeseries data, it provide short-term memories that can be stored and used latter for indefinite amount of time, thus the name Long short-term memory.

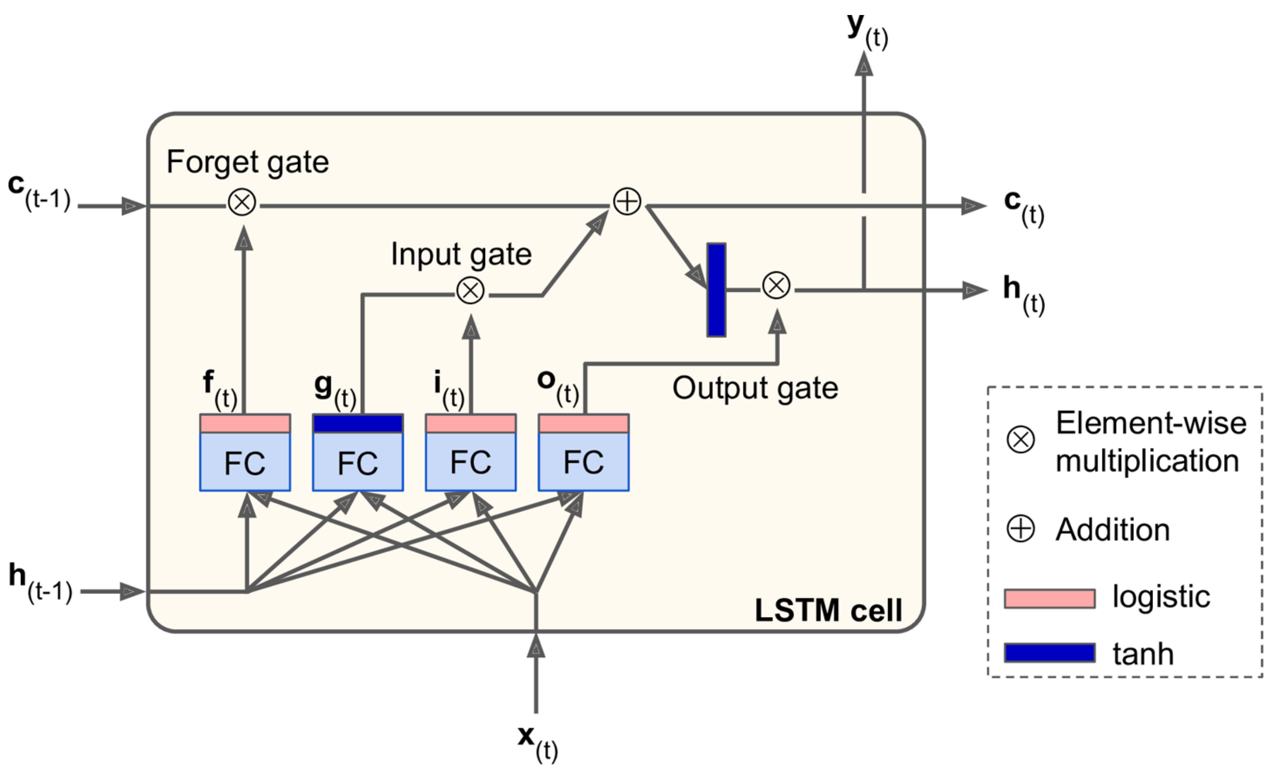


Figure 15: LSTM cell

The key idea is that the network can learn what to store in the long-term state, what to throw away, and what to read from it. First, the current input vector $\mathbf{x}(t)$ and the previous short-term state $\mathbf{h}(t-1)$ are fed to four different fully connected layers. They all serve a different purpose:

- The main layer is the one that outputs $\mathbf{g}(t)$. It has the role of analyzing the current inputs $\mathbf{x}(t)$ and the previous (short-term) state $\mathbf{h}(t-1)$. This layer's output is partially stored in the long-term state.
- The three other layers are gate controllers. Their outputs range from 0 to 1 and are fed to element-wise multiplication operations. Specifically:
  - The forget gate (controlled by $\mathbf{f}(t)$) controls which parts of the long-term state should be erased.
  - The input gate (controlled by $\mathbf{i}(t)$) controls which parts of $\mathbf{g}(t)$ should be added to the long-term state.
  - The output gate (controlled by $\mathbf{o}(t)$) controls which parts of the longterm state should be read and output at this time step (both to $\mathbf{h}(t)$) and $\mathbf{y}(t)$.

In short, an LSTM cell can learn to recognize an important input (that's the role of the input gate), store it in the long-term state, learn to preserve it for as long as it is needed (that's the role of the forget gate), and learn to extract it whenever it is needed. LSTM computations can be summarized as follow:

$$
\begin{aligned}
&\mathbf{i}_{(t)} = \sigma\left(\mathbf{W}_{xi}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hi}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_i\right) \\
&\mathbf{f}_{(t)} = \sigma\left(\mathbf{W}_{xf}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hf}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_f\right) \\
&\mathbf{o}_{(t)} = \sigma\left(\mathbf{W}_{xo}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{ho}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_o\right) \\
&\mathbf{g}_{(t)} = \tanh\left(\mathbf{W}_{xg}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_g\right) \mathbf{c}_{(t)} = \mathbf{f}_{(t)} \odot \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \odot \mathbf{g}_{(t)} \\
&\mathbf{y}_{(t)} = \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \odot \tanh\left(\mathbf{c}_{(t)}\right)
\end{aligned}
$$

- $\mathbf{W}_{xi}, \mathbf{W}_{xf}, \mathbf{W}_{xo}, \mathbf{W}_{xg}$ are the weight matrices of each of the four layers for their connection to the input vector $\mathbf{x}_{(t)}$.
- $\mathbf{W}_{hi}, \mathbf{W}_{hf}, \mathbf{W}_{ho}, \mathbf{W}_{hg}$ are the weight matrices of each of the four layers for their connection to the previous short-term state $\mathbf{h}_{(t-1)}$.
- $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o, \mathbf{b}_g$ are the bias terms for each of the four layers. Note that TensorFlow initializes $\mathbf{b}_f$ to a vector full of 1s instead of 0s. This prevents forgetting everything at the beginning of training.
- $\sigma(x)$ is the Logistic function - a function that takes an input and output an probability between 0 and 1:

$$\sigma(x) = \frac{e^x}{e^x + 1}$$

- $tanh(x)$ is the Hyperbolic tangent function (shifted and scaled version of the logistic function, above)

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

3.4 GRU

The Gated Recurrent Unit (GRU) cell was proposed by Kyunghyun[4] as an improvement over the traditional LSTM model. While LSTMs are effective at handling long-term dependencies in sequential data, they can be computationally expensive due to their complex structure, which includes multiple gates for managing memory. The GRU was designed to fix some of the weaknesses of LSTM by simplifying its architecture. Specifically, GRUs combine the forget and input gates into a single update gate, reducing the number of parameters and making the model more computationally efficient. This simplification allows GRUs to perform similarly to LSTMs, but with fewer resources and faster training times. Thus, GRUs are particularly useful for tasks like time-series prediction, where the goal is to capture temporal dependencies without incurring the computational overhead associated with LSTMs.

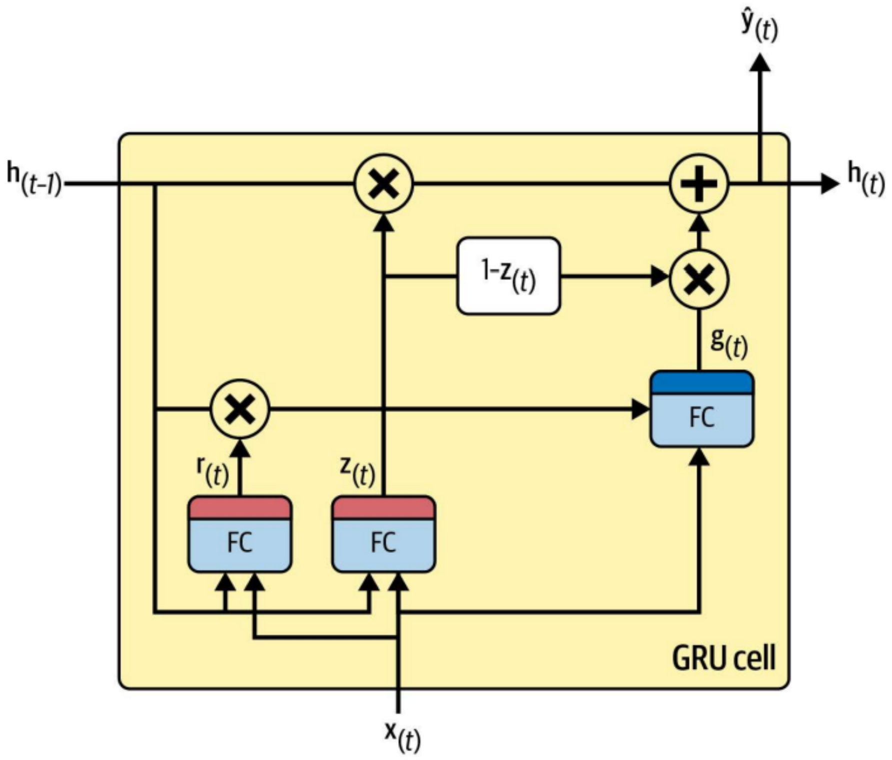


Figure 16: GRU cell

The GRU cell is a simplified version of the LSTM cell, the main simplifications are:

- Both state vectors are merged into a single vector $\mathbf{h}(t)$
- A single gate controller controls both the forget gate and the input gate.
- There is no output gate; the full state vector is output at every time step. However, there is a new gate controller that controls which part of the previous state will be shown to the main layer.

GRU computations can be summarized as follow:

$$\mathbf{z}_{(t)} = \sigma\left(\mathbf{W}_{xz}^{T}\cdot\mathbf{x}_{(t)} + \mathbf{W}_{hz}^{T}\cdot\mathbf{h}_{(t-1)}\right)$$
$$\mathbf{r}_{(t)} = \sigma\left(\mathbf{W}_{xr}^{T}\cdot\mathbf{x}_{(t)} + \mathbf{W}_{hr}^{T}\cdot\mathbf{h}_{(t-1)}\right)$$
$$\mathbf{g}_{(t)} = \tanh\left(\mathbf{W}_{xg}^{T}\cdot\mathbf{x}_{(t)} + \mathbf{W}_{hg}^{T}\cdot\left(\mathbf{r}_{(t)}\odot\mathbf{h}_{(t-1)}\right)\right)$$
$$\mathbf{h}_{(t)} = \left(1-\mathbf{z}_{(t)}\right)\odot\tanh\left(\mathbf{W}_{xg}^{T}\cdot\mathbf{h}_{(t-1)} + \mathbf{z}_{(t)}\odot\mathbf{g}_{(t)}\right)$$

3.5 ConvLSTM

When working with data that has a third dimension, such as the number of provinces in our case, traditional GRU and LSTM models, which are designed to handle sequential data with only two dimensions (time and features), are not directly applicable. To address this limitation, ConvLSTM (Convolutional LSTM)[5] offers a suitable solution by extending the traditional LSTM model to handle spatial-temporal data. ConvLSTM combines the capabilities of Convolutional Neural Networks (CNNs) and LSTMs to process both spatial and temporal dimensions effectively.

In ConvLSTM, the convolutional operations are applied to the input data before passing it through the LSTM cells. This allows the model to capture spatial patterns in the data (such as the geographic relationships between provinces) while simultaneously learning temporal dependencies (such as changes over time). By using ConvLSTM, we can effectively incorporate the third dimension, the number of provinces, into the model, enabling it to capture the spatial interactions between different locations and make more accurate predictions for air quality based on weather data from multiple provinces.

3.5.1 Convolutional Layer

The convolutional layer applies a series of filters (or kernels) to the input matrix. These filters slide over the matrix, performing element-wise multiplication and summation to produce feature maps. This process allows the network to learn spatial hierarchies in the data.

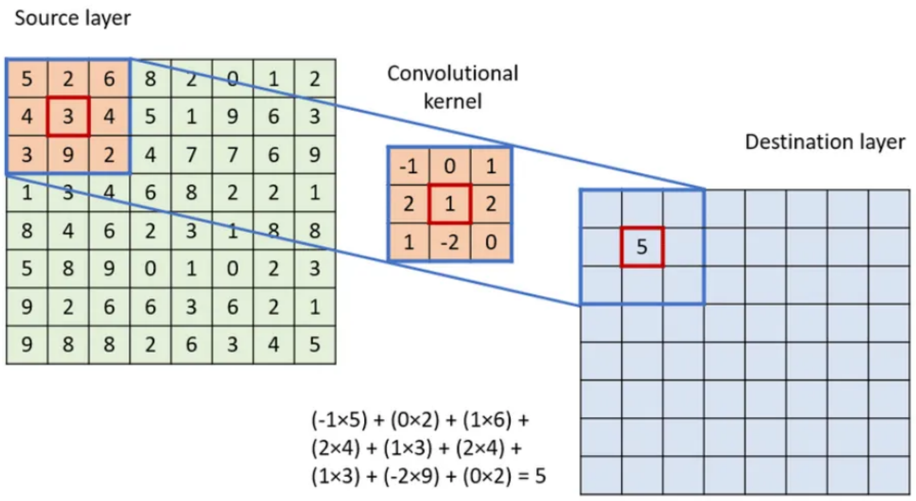


Figure 17: Convolutional operation.

3

$$f(x,y) * g(x,y) = \sum_m \sum_n f(m,n)\, g(x-m, y-n)$$

- $f(x,y)$: The input signal or feature map.
- $g(x,y)$: The convolution kernel (also known as the filter).
- m,n: Indices for summation, representing the positions in the input and kernel.

3.5.2 ConvLSTM model

In a ConvLSTM model, the input is passed through a convolution layer before added to the LSTM cell.

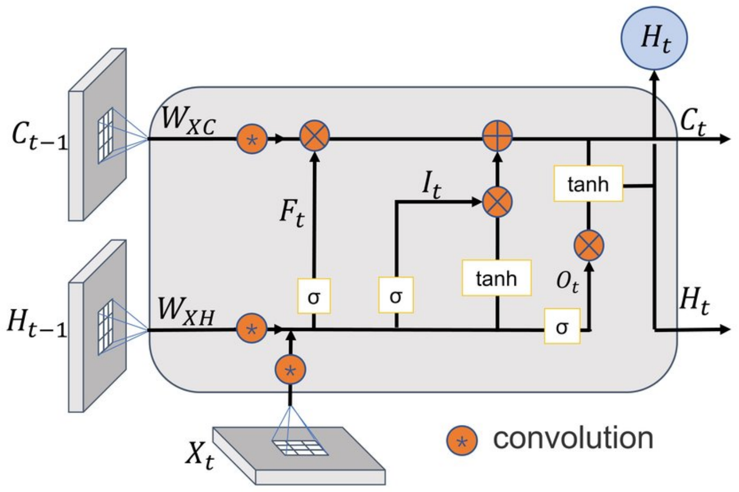


Figure 18: Inner structure of ConvLSTM

The ConvLSTM computations can be sumarize as follow:

$$
\begin{aligned}
\mathbf{i}_t &= \operatorname{Sigmoid}\left(\operatorname{Conv}\left(\mathbf{x}_t;\mathbf{w}_{xi}\right)+\operatorname{Conv}\left(\mathbf{h}_{t-1};\mathbf{w}_{hi}\right)+\mathbf{b}_i\right)\\
\mathbf{f}_t &= \operatorname{Sigmoid}\left(\operatorname{Conv}\left(\mathbf{x}_t;\mathbf{w}_{xf}\right)+\operatorname{Conv}\left(\mathbf{h}_{t-1};\mathbf{w}_{hf}\right)+\mathbf{b}_f\right)\\
\mathbf{o}_t &= \operatorname{Sigmoid}\left(\operatorname{Conv}\left(\mathbf{x}_t;\mathbf{w}_{xo}\right)+\operatorname{Conv}\left(\mathbf{h}_{t-1};\mathbf{w}_{ho}\right)+\mathbf{b}_o\right)\\
\mathbf{g}_t &= \operatorname{Tanh}\left(\operatorname{Conv}\left(\mathbf{x}_t;\mathbf{w}_{xg}\right)+\operatorname{Conv}\left(\mathbf{h}_{t-1};\mathbf{w}_{hg}\right)+\mathbf{b}_g\right),\\
\mathbf{c}_t &= \mathbf{f}_t\odot\mathbf{c}_{t-1}+\mathbf{i}_t\odot\mathbf{g}_t\\
\mathbf{h}_t &= \mathbf{o}_t\odot\operatorname{Tanh}\left(\mathbf{c}_t\right)
\end{aligned}
$$

The input to the ConvLSTM model consists of a three-dimensional array representing the spatial-temporal data, where the dimensions correspond to the time window, the number of provinces, and the number of features. Specifically, the time window represents the temporal sequence of weather data (hourly data), the number of provinces captures the spatial dimension, and the number of features corresponds to the different weather variables being measured. To effectively model the relationships between the features and between the provinces, we use convolution with a kernel size of (1, number of features). This kernel size is chosen to operate across the feature dimension, allowing the model to learn how the different features relate to each other for each province at a given time step. Additionally, the number of provinces is used as the number of input channels, allowing the model to capture the spatial dependencies and relationships between the provinces over time. This architecture enables the ConvLSTM to learn both the temporal dynamics of the weather data and the spatial interactions between provinces.

4 Experiments and Results

4.1 RMSE

Table 3: RMSE Comparison of Models for Different Air Quality Attributes

Model	**CO**	**NO_2**	**O_3**	**SO_2**	**$PM_{2.5}$**	**PM_{10}**
Random Forest	322.589	6.986	19.102	4.729	26.339	28.951
GRU Network	467.084	11.538	24.474	8.739	34.837	39.177
ConvLSTM	328.525	7.565	21.680	5.373	29.756	32.302

4.2 MAE

Table 4: MAE Comparison of Models for Different Air Quality Attributes

Model	**CO**	**NO_2**	**O_3**	**SO_2**	**$PM_{2.5}$**	**PM_{10}**
Random Forest	168.791	3.930	13.259	2.330	15.328	16.893
GRU Network	253.261	7.177	17.368	4.860	20.832	23.450
ConvLSTM	187.567	4.381	14.948	2.731	16.621	18.238

From Table 3 and Table 4, Random Forest achieves the lowest RMSE and MAE for most pollutants, showcasing its superior ability to model temporal dependencies in air quality data. The GRU Network records the highest RMSE and MAE across all attributes, indicating it struggles with prediction accuracy. Meanwhile, ConvLSTM demonstrates decent performance with errors close to those of Random Forest.

5 Discussion

5.1 Summary of Findings

In this study, we analyzed the time series data using STL decomposition, ACF/PACF analysis, and the Augmented Dickey–Fuller (ADF) test to evaluate stationarity. The STL decomposition effectively isolated the trend, seasonality, and residual components, providing valuable insights into the data structure. Notably, the residual analysis revealed a spike in September 2024, attributed to the Yagi storm rather than statistical outliers. Overall, no significant anomalies or outliers were detected in the residuals, suggesting the data's reliability for further modeling.

The ADF test results confirmed that all variables in the dataset are stationary with a 95% confidence level. This is a critical finding as it validates the assumptions required for time series modeling techniques.

5.2 Interpretation of Results

The ACF and PACF plots further support the stationarity of the time series. The ACF plot shows a gradual decay, while the PACF plot exhibits a sharp cut-off after the first lag, indicating that an autoregressive process of order one (**AR(1)**) is a suitable candidate for modeling the data. These patterns suggest strong short-term dependencies in the data, which can be effectively captured by LSTM-like models.

The absence of significant outliers in the residual analysis, aside from the storm-induced spike, indicates that the data is well-behaved and reflects real-world phenomena rather than noise or errors. This strengthens the robustness of the pre-processing steps and decomposition applied.

6 Conclusion

This study demonstrates the feasibility and effectiveness of using weather data to predict air quality across Vietnamese provinces. Through rigorous time series analysis, including stationarity testing, STL decomposition, and ACF/PACF analysis, we established a solid foundation for predictive modeling. The Augmented Dickey–Fuller (ADF) test confirmed that the data meets the stationarity assumptions required for advanced time series models, while the STL decomposition effectively separated trends, seasonality, and residuals, allowing for the identification of key patterns and external influences, such as the Yagi storm.

We evaluated multiple predictive models, including Random Forest, GRU, and ConvLSTM, to forecast air quality based on weather attributes such as temperature, precipitation, wind speed, and humidity. Among these models, the Random Forest algorithm achieved the lowest RMSE and MAE for most air quality indicators, showcasing its superior ability to capture temporal dependencies in the data. The ConvLSTM model also demonstrated strong performance, particularly in capturing spatial-temporal relationships.

The results highlight the strong interdependence between meteorological conditions and air quality. Weather variables provide critical predictive information for air pollutants like PM2.5, PM10, CO, and O_3, enabling accurate forecasting of air quality levels. These findings reinforce the potential of integrating weather forecasts into air quality management systems, offering actionable insights for policymakers to mitigate the adverse effects of air pollution.

In conclusion, this study establishes a robust framework for using weather data to predict air quality. By combining traditional statistical techniques with modern machine learning models, we not only improve forecasting accuracy but also contribute to a deeper understanding of the relationship between weather and air quality.

References

[1] BYJU'S. *Linear Interpolation Formula*. Accessed: 2024-04-27. 2024. URL: https://byjus.com/linear-interpolation-formula/#:~:text=Linear%20interpolation%20is%20a%20method,1%20)%20x%202%20%E2%88%92%20x%201.

[2] Helmut Lütkepohl. *New Introduction to Multiple Time Series Analysis*. Berlin, Heidelberg: Springer-Verlag, 2005. ISBN: 978-3-540-27752-1. DOI: 10.1007/978-3-540-27752-1.

[3] DataCamp Team. *Random Forest Classifier in Python*. Accessed: 2024-12-18. n.d. URL: https://www.datacamp.com/tutorial/random-forests-classifier-python.

[4] Kyunghyun Cho et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. arXiv: 1406.1078 [cs.CL]. URL: https://arxiv.org/abs/1406.1078.

[5] Xingjian Shi et al. "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting". In: *Advances in Neural Information Processing Systems* 28 (2015), pp. 802–810. URL: `https://arxiv.org/pdf/1506.04214v2`.