Cluster and Cloud Computing Assignment 2 - Weather's Influence on our Life in Melbourne

Group 64
Kejing Li, Xin Su, Xinhao Chen, Yueyang Li, Zheqi Shen
Student ID: 1240956, 1557128, 1166113, 1213643, 1254834

May 2024

Abstract

This report is about using cloud computing to gather, process and analyse the data from various data sources on the influence of weather on people's lives in Melbourne. The research is conducted on Melbourne Research Cloud with Fission and ElasticSearch. In the report, we discuss the architecture of the system and the analysis of the data with visualisations.

Contents

1 Introduction

1.1 Project Background & Objectives

in the connected world of today, the weather has a significant effect on many elements of our everyday life, affecting anything from social media activity and mood to traffic patterns and health. This research uses a Function as a Service (FaaS) application to investigate these complex interactions. Our application uses big data analytics to collect data from various sources, such as the Environmental Protection Agency (EPA), the Bureau of Meteorology (BoM), the Spatial Urban Data Observatory (SUDO), Victoria's open data platforms like DataVic and social media platforms like Mastodon. The primary objective is to analyse and understand how weather conditions affect different aspects of life in Melbourne.

Our project integrates several advanced technologies, such as the Melbourne Research Cloud, Kubernetes, Fission, and ElasticSearch. Our goal in using these tools is to build a scalable and robust system that can harvest, process, and analyse data in real time. This program targets real-world scenarios that emphasise the impact of weather patterns on social behaviour, public health, and transportation safety in addition to showcasing advanced technology.

1.2 Expected Goals & Outcomes

The main goal of our project is to provide valuable insights into the influence of weather on various aspects of life in Melbourne. Specifically, we aim to:

- **Analyse Social Media Activity:** Look at the relationship between the volume and mood of Mastodon postings and the weather. This entails figuring out whether specific weather patterns cause social media activity to increase in frequency or intensity.
- **Evaluate Traffic Accidents:** Examine the connection between inclement weather and traffic incidents. The objective is to pinpoint certain meteorological circumstances that lead to a rise in accidents and comprehend the underlying reasons.
- **Study Health Impacts:** Examine the relationship between public health and air quality, with a special emphasis on respiratory disorders. This investigation will contribute to our knowledge of how variations in weather-related air quality impact the prevalence of health problems.

By achieving these objectives, our study aims to offer practical insights that can raise public understanding of how the weather affects daily living, promote urban planning, and inform public policy. Comprehensive analytical reports, interactive visualisations, and prediction models are among the anticipated results, which taken together provide a thorough grasp of how weather affects Melbourne life. Our goal in doing this project is to show how big data analytics and cloud computing can be used to solve challenging real-world issues.

2 System Architecture & Design

2.1 System Overview

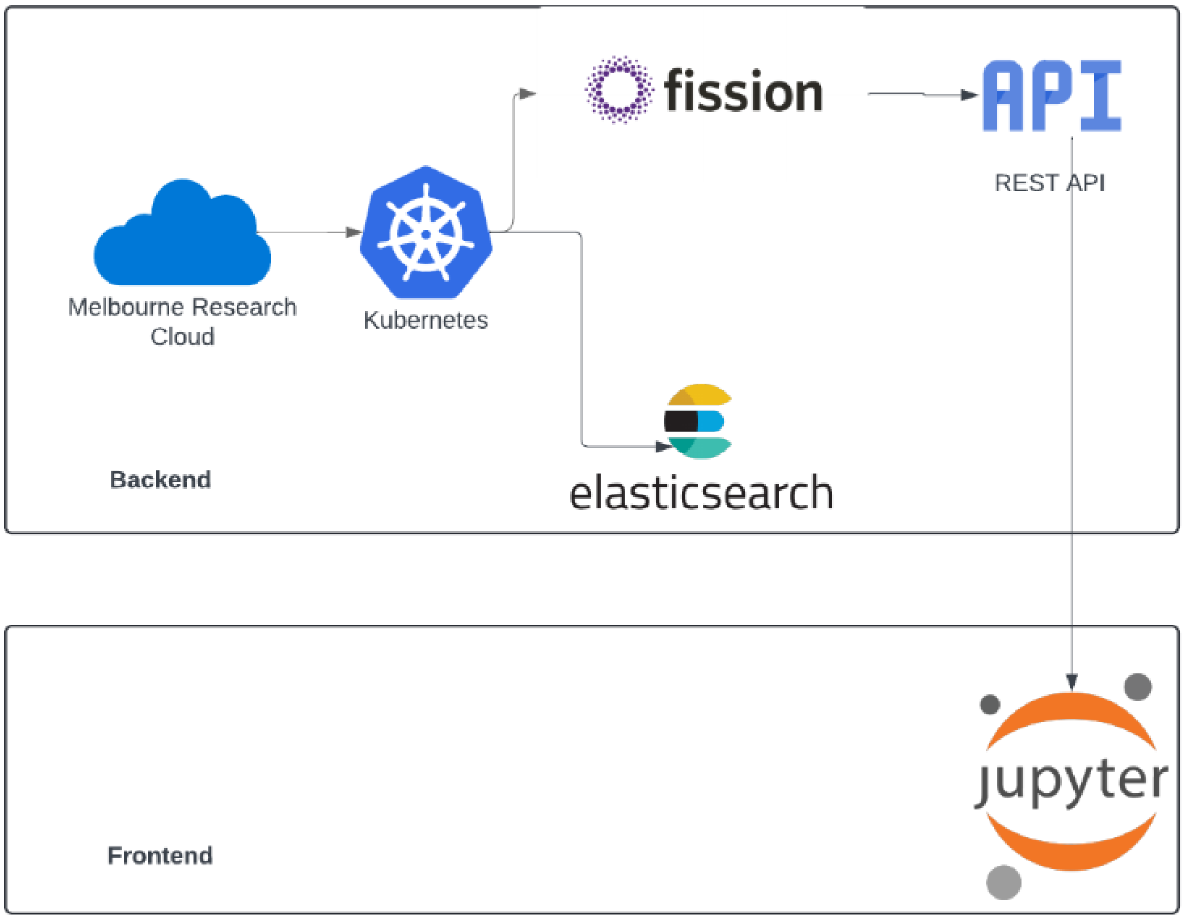


Figure 1: System Architecture

The physical system is deployed on Melbourne Research Cloud (MRC) which is based on OpenStack. It provides predefined resources for us to use as a project. All the other components are deployed on this resource.

To manage resources and services effectively, Kubernetes is used to provide automatic deploying, scaling and monitoring. It provides an abstraction layer on the physical machines that allow us to operate on them in a simple way.

To deploy popular complex applications on Kubernetes, Helm is adopted to help create the applications that we are using. Applications are abstracted as charts in Helm which allow deployment and configuration of a complex application to be done with a single command. We deploy Fission, ElasticSearch and Kibana with Helm. By using Helm, different components of the system can be created or deleted with simple commands.

At the architectural level, we decide to use a Function as a Service (FaaS) architecture. Thus Fission is chosen to run our functions written in Python because of its simplicity and performance. By utilising the various triggers like timer and HTTP routes, we are able to harvest real-time and expose data harvested as RESTful API.

For data persistence, we choose ElasticSearch to store the harvested data because of its flexibility to store arbitrary JSON data, the ability to search text data efficiently and the ability to scale in distributed systems. For this assignment, we deploy the ElasticSearch on 2 nodes.

To test queries and check indices of the ElasticSearch, we deploy the Kibana on a node in the cluster. It provides a dev console to test queries easily and an overview of the ElasticSearch cluster to monitor its status.

The language we use to create functions is Python. This is because of the need to access certain text analysis packages which are available in Python.

2.2 Data Security

To secure our service of ElasticSearch, it is important to create a secure enough password. Initially, we use a simple password, but later when changing it, we encounter bugs about ElasticSearch. It seems that the Helm chart of ElasticSearch does not allow change of password. In the end, we have to redeploy the whole ElasticSearch cluster. During redeployment, we create a randomly generated 32-digit long password.

To securely access the password, we use Kubernetes's secret resource. Specifically, we create a YAML file with all the confidential encoded by base64, and create the secrets on Kubernetes. In functions, we retrieve the secrets from Kubernetes and access the ElasticSearch database with basic authentication. This ensures that confidential information like passwords will not be present in the repository, thus improving data security. For index creation scripts, we store the username and password in a git ignored .env file. Inside

We found later in development that we can use access tokens generated with Kibana for better security. However, due to the time limit, we stick to basic authentication with username and password.

2.3 Testing

Testing is a crucial part of complex distributed systems. For this assignment, we are using pytest for integrated tests. For each endpoint of the REST API, pytest test functions are created to validate the structure of its successful response and its error message when an incorrect argument is passed to the API.

2.4 Fault Tolerance

The system ensures fault tolerance through Kubernetes. By utilising the tools Kubernetes provides like node monitoring and resource management, the system can recover when errors happen.

For ElasticSearch, two replicas are deployed to ensure fault tolerance. Even when one node of ElasticSearch fails, the other will be able to keep working. This greatly improves the reliability of the database.

The adoption of Fission ensures fault tolerance through its FaaS architecture. The components of the system are split into single-responsibility functions. Fission gets to decide where the function is going to be run, ensuring resilience to failures. Also functions are run in separate environments from other functions, thus fault in one function does not affect the others.

2.5 Scalability

The system allows easy scaling through OpenStack, Kubernetes, Helm and Fission. Through these tools, new nodes and new services can be added to improve performance and reliability. OpenStack can be used to request new machines, and Kubernetes can be used to manage them. Helm can be used to deploy services with more nodes. Functions are managed by Fission thus are unaware of physical resources, which allow them to scale automatically on new machines.

2.6 Tools Pros & Cons

2.6.1 Melbourne Research Cloud

Pros:

- MRC has a clear and feature-rich UI for common operations like key pair management and resources monitoring. It is based on OpenStack to provide a reliable cloud environment.
- OpenStack has a good cli tool. The cli tools are Python powered and can be easily installed with pip.

Cons:

- OpenStack cli tools have compatibility issues with Windows, thus requiring Windows users to use Windows Subsystem for Linux (WSL).
- MRC requires a University of Melbourne network to connect to. Thus to access it outside the university requires a VPN. However, using a VPN from WSL on Windows is a challenge, which requires additional configuration and experimental features of WSL.

2.6.2 Kubernetes

Pros:

- It generalises physical resources to the resource hierarchy of cluster, node, pod, namespace and services etc. By operating on such resources, the users won't need to worry about physical machines as Kubernetes manages them automatically for you.
- The scalability of Kubernetes is good. Just by specifying the configuration, Kubernetes automatically arranges all the physical resources for users. And the user can manage all the resources from his own local terminal with the help of Kubectl.

- Kubernetes allows easy monitoring of the whole application. By allocating applications into namespaces, users are able to monitor their health easily. From the local terminal, users can know which of the nodes and services are down and their resource usage.
- Easy discovery of services is backed up by Kubernetes DNS. Unique hostnames are given for services that allow other services to access it without hard-coding the IP address. This allows the system to easily scale when new machines are added or old machines removed, because Kubernetes manages the relationship between hostname and actual service, allowing seamless scalability.

Cons:

- Kubernetes introduces additional complexity compared to operating on actual machines. Since Kubernetes manages the physical machines, users do not have direct access to the machines. Thus the users have to learn many concepts about Kubernetes to understand their relationship to physical resources.
- Kubernetes brings overhead to operations as it is an additional layer over the physical machine. Due to the extra features that it provides, a decrease in performance is expected compared to direct access to machines.
- It is rather difficult to access the Kubernetes on MRC. A ssh port forwarding from remote to localhost is needed each time a user wants to access Kubernetes.

2.6.3 Helm

Pros:

- It makes it easy to install complex applications like ElasticSearch and Kibana.

Cons:

- The installation is a black box. When an installation or uninstallation fails, it leaves tons of resources behind which get in the way of reinstallation. There is no complete list of resources created for manual removal, and the only way to get a clean setup is to delete the whole namespace.
- Helm charts of ElasticSearch are deprecated for Elastic Cloud on Kubernetes (ECK). The installation lacks enough documentation to customise.

2.6.4 Fission

Pros:

- Fission is a FaaS serverless framework that allows users to run stateless functions on various triggers it provides. By creating single-responsibility functions, we are able to divide the large system into smaller, manageable parts.
- Fission allows access to Kubernetes resources like config maps and secrets, which increases the scalability and security.

- Fission spec provides a declarative way of specifying complex configuration. It makes the creation of environments, packages, functions and triggers easy and consistent. By writing function configurations as YAML files, it allows version management through tools like Git. Using spec significantly reduces the overhead of typing commands.

Cons:

- Using Python packages in Fission is cumbersome and buggy. Creating a Fission function with extra dependencies requires first a Python environment, then a package from a zip file with code, building a shell file and requirements file, and creating a function from the package. Creating a function usually takes at least 4 commands.
- Creating a package is slow and cumbersome. It takes much longer than a normal virtual environment on a local machine, and there is no watch mode, so the user has to continuously check the state of the package using fission package info.
- There is no check when a function is created. When the function creation fails, no output is given to the user and the user has no way to check if a function is created successfully with all its dependencies. Only a timeout error is given when errors like missing dependencies happen, and the user has no idea of the exact error he made. Users have no idea whether the wait is because of a slow network, busy calculation or the function itself.
- Fission's Python environment uses Flask internally which lacks support for common features like retrieving path parameters from URL. Fission provides a workaround through headers but is awkward and hard to use.

2.6.5 ElasticSearch

Pros:

- Index in ElasticSearch is tolerant to JSON data and can infer appropriate data types for columns. It also allows the explicit creation of indices to let users specify the data type they want.
- ElasticSearch is naturally scalable with concepts of shards and replicas. It fits well in a distributed environment like Kubernetes. With sensible defaults, it stores the data reliably in the distributed system. It also allows explicit configuration of an index's data distribution.
- Query with REST API is well-designed and handy for command-line testing. There is no need to specifically design clients to interact with ElasticSearch.
- It provides usable data types such as date for easy query over date and time.

Cons:

- Query DSL is just JSON which lacks syntax checking. Users cannot guarantee the correctness of complex queries, but have to test it manually. Also the query DSL has readability issues when getting longer. The naming of attributes and the tree structure of JSON makes it hard for users to understand complex queries.

- ElasticSearch is not suitable for join queries. It makes it hard to use when handling large amounts of data from different sources connected by some attributes.

2.6.6 Kibana

Pros:

- It integrates with ElasticSearch seamlessly. No configuration is needed to connect Kibana with ElasticSearch.
- As the official UI for ElasticSearch, it has a good dev console for query testing. The dev console can provide auto-completion for index names and column names.
- It provides easy monitoring of data status. The users can get the status of an index from the page provided, and the status is represented by colours.

Cons:

- Kibana has a rather steep learning curve, as it not only integrates with ElasticSearch but also other components like LogStash, which is not used in this project. Finding the functionality among all the integration it provides is hard.

3 Data Collection & Processing

3.1 Spatial Urban Data Observatory (SUDO)

Spatial Urban Data Observatory (SUDO) is a platform that aggregates and provides access to a wide array of urban data across more than 8500 datasets from over 150 organisations, including Geoscience Australia (GA) and the Australian Bureau of Statistics (ABS). It offers a federated access control system for secure data sharing, analytical tools across various domains such as health, demographics, and economic activities, and facilitates advanced data processing, analysis, and visualisation. SUDO supports collaborative research efforts through features like data export and social media integration, enabling researchers and policymakers to effectively analyse and visualise spatial data to inform urban planning and policy decisions.

1. **Data Collection:** We gathered datasets from SUDO titled ‘SA2-G21A Type of Long-Term Health Condition by Selected Person Characteristics’, which is provided by the ABS as part of the 2021 Census data and structured around Statistical Areas Level 2 (SA2) for easier geographical analysis. It details the counts for people born in Australia with a variety of chronic diseases.
2. **Data Cleaning:** According to the scenarios focused on in this project, we selected data on two health conditions that could be related to air quality, one is asthma, and the other is chronic obstructive pulmonary disease (COPD), and restricted the region to Greater Melbourne.

   This dataset also provides a unique code and name for each SA2 region, combining with the tools ‘Spatialised Aggregated Dataset’ in SUDO, the geometry information for a specific geographic location or region could be assigned to related data. The dataset with the geometry attribute could be downloaded to the local system in the Shapefile (SHP) format. Read the Shapefile with the package 'geopandas' in Python to convert it to JSON format.

3. **Data Storage:** The cleaned data is stored in a file named 'health_geo.json' and is indexed into ElasticSearch under the index name 'health_geo' for subsequent retrieval and analysis. Especially for the geometry data of each SA2 region, the data type set is 'geo_shape', which is different from regular data types.

3.2 Bureau of Meteorology (BOM)

Bureau of Meteorology (BOM) is the most important agency for weather observatory and forecast. Users could view detailed weather conditions for the past 72 hours, as well as a daily synopsis of the weather for the past year.

- **Past Year Data (April 2023 to May 2024)**

  1. **Data Collection:** Historical weather data for the Melbourne Olympic Park weather station was collected from the website of BOM. The downloaded CSV files were converted to JSON format to facilitate easier storage and querying in ElasticSearch. Each row in the CSV was mapped to a JSON document, with columns represented as key-value pairs.

  2. **Data Cleaning:** Columns that were not pertinent to the study were identified and removed from the dataset. This step streamlined the data and focused on the relevant weather parameters.

  3. **Data Storage** An ElasticSearch index 'bom_melbourne_weather_past' was created to store the cleaned data. The index was structured to accommodate various weather parameters, with fields corresponding to each data attribute. Appropriate mappings were defined to optimise storage and retrieval. Date fields were mapped as date types, and numeric fields were mapped as appropriate numeric types.

- **Real-time Data**

  1. **Data Collection:** The Bureau of Meteorology provides an API endpoint to access real-time weather data. The API is configured to query the Melbourne Olympic Park weather station, identified by its unique station ID. A data harvester function is executed to get the most recent weather record through the Bureau of Meteorology's API call. To achieve real-time data harvest, a Fission trigger is implemented to automate the data retrieval process. This trigger is scheduled to execute the data harvester function every 30 minutes, ensuring that the most recent weather data is consistently obtained.

  2. **Data Cleaning:** The response from the API is received in JSON format. This data includes various weather parameters such as temperature, humidity, and wind speed. Then, the JSON data is parsed, and the datetime, temperature, pressure, rainfall, relative humidity, wind speed and wind direction are extracted for further analysis.

  3. **Data Storage** An ElasticSearch index "bom_melbourne_weather" is created

to store the harvested weather data. The parsed and transformed weather data is prepared for storage in ElasticSearch. Each data point is structured as a document with appropriate fields representing the weather parameters. The data is indexed into an ElasticSearch cluster, enabling efficient querying and analysis. The indexing process ensures that each document is uniquely identified and stored in a way that supports real-time search capabilities.

3.3 Mastodon

Mastodon is a decentralised social networking platform where users interact through instances hosted on different servers. To collect and process data from Mastodon, we gather information from instances related to Australia, such as Aus.Social.

- **Past Year Data (May 2023 to May 2024)**

  1. **Data Collection:** We use the Mastodon API to extract posts (toots) containing the keyword ‘Melbourne’. This data spans the past year to ensure comprehensiveness and representativeness.

  2. **Data Cleaning:** Collected data is cleaned by removing duplicates, handling missing values, and eliminating URLs, HTML tags, and non-text symbols. We use the Python library TextBlob to perform sentiment analysis on the post content, calculating the sentiment score for each toot.

  3. **Data Storage:** The cleaned data is stored in a file named ‘mastodon_melbourne_past_data.json’ and is indexed into ElasticSearch under the index name ‘mastodon_melbourne_past’ for subsequent retrieval and analysis.

- **Real-time Data**

  In addition to historical data collection, we have implemented real-time data collection and processing every half hour to ensure the timeliness and dynamic updating of the data. The specific steps are as follows:

  1. **Data Collection:** Using the Mastodon API, we extract the latest posts containing the keyword ‘Melbourne’ every half hour to ensure data timeliness.

  2. **Data Processing:** The real-time collected posts are cleaned and sentiment analysed following the same steps as historical data processing.

  3. **Data Storage:** The processed real-time data is stored in another ElasticSearch index named ‘mastodon_melbourne’ to manage it separately from historical data.

  By following these steps, we can extract, clean, and systematically store a large amount of historical and real-time data related to ‘Melbourne’ from Mastodon. This allows us to continuously monitor and analyse toots related to ‘Melbourne’, providing a robust dataset for further analysis.

- **Integrating Mastodon and BOM Data**

To reduce the amount of code in the Jupyter Notebook and make the frontend cleaner, we further processed the data from the Mastodon and BOM indices and stored it in a new index named 'mastodon_bom_past'. The following steps outline this integration:

1. **Merging Mastodon Data:** We queried ElasticSearch to aggregate Mastodon data by date, calculating average, maximum, and minimum sentiment scores, extracting the number of toots per day, and identifying frequently occurring tags and content.

2. **Merging BOM Data:** We queried ElasticSearch to retrieve BOM weather data, including daily temperature (average, maximum, minimum), rainfall, and wind gust speed.

3. **Combining Data:** Using the date as the index ID, we combined the extracted data into a single entry. The combined data was then indexed into a new ElasticSearch index named 'mastodon_bom_past'. The 'get_bom_mastodon_by_date' harvester is used to accurately extract data into the notebook.

3.4 Environmental Protection Agency (EPA)

The Environment Protection Authority (EPA) provides information and resources on environmental protection and sustainability in Victoria, Australia. It encompasses subjects such as waste management, pollution control, environmental regulations, and monitoring of the quality of the air and water.

1. **Data Collection:** Data on air quality for 2021 was acquired by us from the Environmental Protection Authority (EPA). This dataset contains information on various Melbourne monitoring sites that track BPM2.5 levels. The specific data collected comprises the location ID, location name, BPM2.5 measurements, date, and time.

2. **Data Cleaning:** As EPA covers all air quality stations around Victoria, we selected the several stations in the range of Greater Melbourne and carried out a number of cleaning procedures to guarantee the accuracy and consistency of the air quality data. This involved making sure the date and time fields were structured correctly and consistently throughout all entries, confirming the integrity of the location IDs and names, and checking for and handling any missing BPM2.5 records or outliers.

3. **Data Storage:** For later retrieval and analysis, the cleaned data is saved in a folder called 'air_quality_json' and is indexed into ElasticSearch under the index name 'air_quality_all'.

3.5 Data VIC

DataVic is an initiative by the Victorian Government that provides public access to a wide range of government data to promote transparency and innovation. It provides data from a variety of industries, including the environment, health, and transportation, encouraging commercial and community solutions under a Creative Commons licence.

1. **Data Collection:** We obtained information on traffic accidents from Data VIC by Department of Transport and Planning from 2012 to September 2023, each traffic accident there is covered in detail.
2. **Data Cleaning:** To merge data reasonably with weather data, we filtered the time period to April 2023 to September 2023, and only kept the unique accident number (accident No.), accident date, accident location in latitude and longitude. We also double-checked the consistency and accuracy of the accident numbers, made sure the dates and times of the accidents were formatted correctly, and checked the latitude and longitude values. The integrity and dependability of the dataset were also preserved by identifying and addressing any missing or duplicate items.
3. **Data Storage:** The cleaned data is stored in a file named "traffic_accidents.json" and indexed into ElasticSearch with the index name "traffic" for further retrieval and analysis.

4 Results Analysis

4.1 Weather vs Mastodon (Number of Toots, Mood)

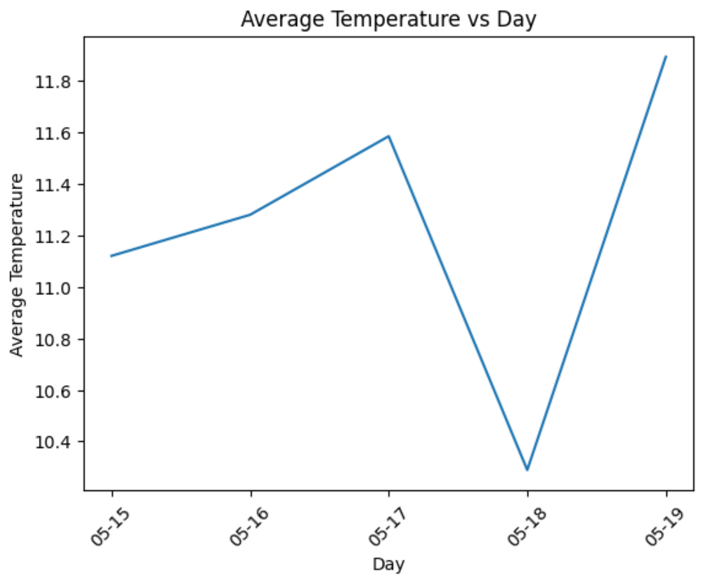


Figure 2: Average Temperature vs Day

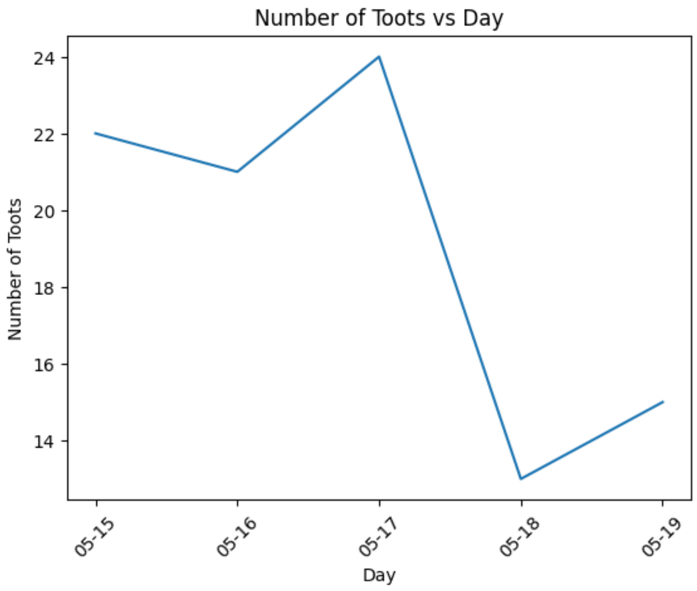


Figure 3: Number of Toots vs Day

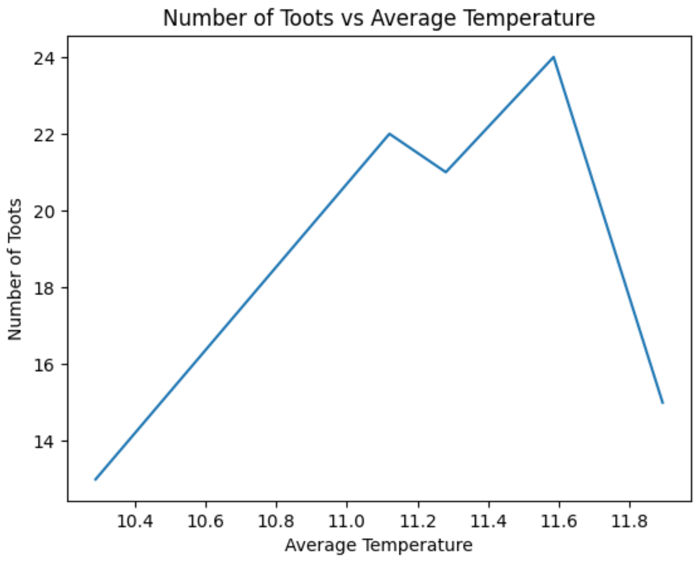


Figure 4: Number of Toots vs Average Temperature

Based on real-time data analysis, Figures 2 and 3 show the changes in daily average temperature and the total number of Mastodon toots from May 14 to May 18. Figure 4 illustrates the trend in the number of toots with varying average temperatures. Specifically, as the average temperature increases, the number of toots rises within a certain temperature range (approximately 11.5°C to 12.5°C), peaking at around 12.5°C, and then decreases as the temperature continues to rise.

From these three figures, we can see a correlation between the daily average temperature and the number of Mastodon toots. The number of toots increases with higher temperatures but decreases after reaching a certain threshold. This suggests that extreme heat may negatively impact social media activity, leading to reduced online engagement.

However, it cannot be ignored that we only fetch five days real-time data, which is not enough to make a conclusion about the relationship between the two variables, and the difference between the maximum and minimum values of the average temperature and the number of toots taken is not significant.

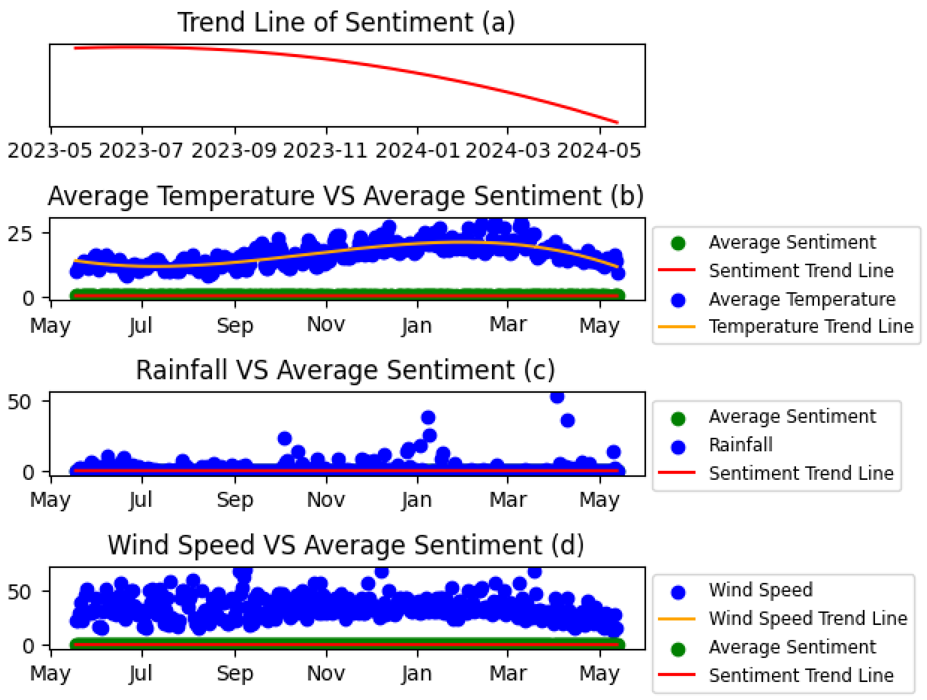


Figure 5: Relationship between Melbourne Weather Variables and Social Media Sentiment

Figure 5 shows the relationship between average sentiment and average temperature, rainfall, and wind speed in Melbourne over the past year. To illustrate changes in sentiment more clearly, Figure 5(a) provides a detailed view of the trend in Melbourne's sentiment over the year. According to Figure 5(b), Melbournians' sentiment rises as the temperature drops (from May 2023 to July 2023). However, as the temperature rises, the average sentiment quickly falls (from July 2023 to May 2024). Additionally, rainfall during this period increased rapidly, with one day reaching up to 50mm. Figure 5(c) shows that sentiment remains stable at low wind speeds but becomes more variable as speeds increase, with a slight decline at higher speeds.

The heatmap (Figure 6) further validates these observations. The correlation coefficient of -0.09 indicates a negative relationship between temperature and sentiment, suggesting that higher temperatures may slightly hurt sentiment. Although the correlation coefficient between rainfall and average sentiment is only 0.05, extreme rainfall can still significantly impact people's emotions.

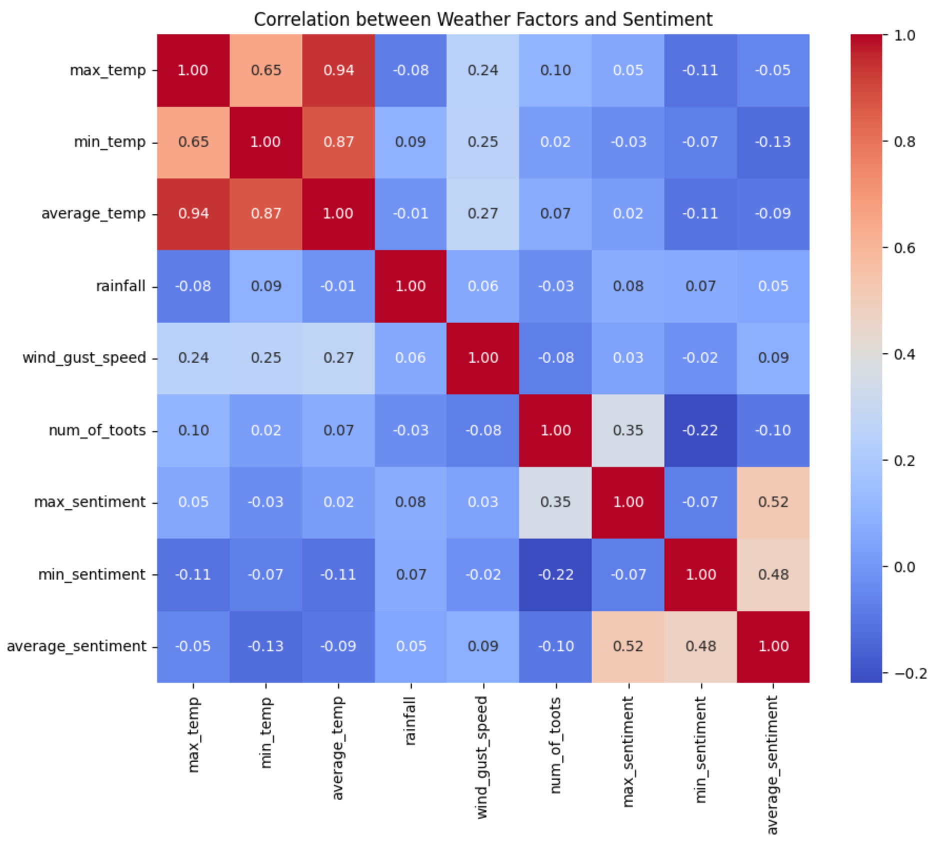


Figure 6: Correlation Matrix of Weather Factors and Social Media Sentiment

In summary, the impact of weather on residents is not uni-dimensional but a complex combination of factors. However, a cool (12.5°C), breezy, and rain-free day can indeed improve the mood of Melbourne residents.

4.2 Weather vs Traffic Accidents

In order to get the traffic accident data ready for analysis, we create a Fission function to count the number of daily accidents, especially including accidents that happened within a 15-kilometer radius of Melbourne Central.

Contrary to the common belief that higher rainfall leads to more accidents, Figure 7 doesn't show an obvious relationship between rainfall and traffic accidents. Most accidents occur when rainfall is between 0mm and 5mm, while accidents decrease significantly when rainfall exceeds 10mm.

There could be several reasons. First, there aren't that many days with heavy rainfall, resulting in a small sample size to show a possible association between the number of accidents and rainfall. Notice in the rightmost data point that there were nearly 15 accidents on a day with nearly 40 millimeters of rain, which is not a small number. Second, from a subjective point of view, Melbourne drivers are likely to drive more cautiously in poor weather conditions, and reducing the likelihood of accidents.

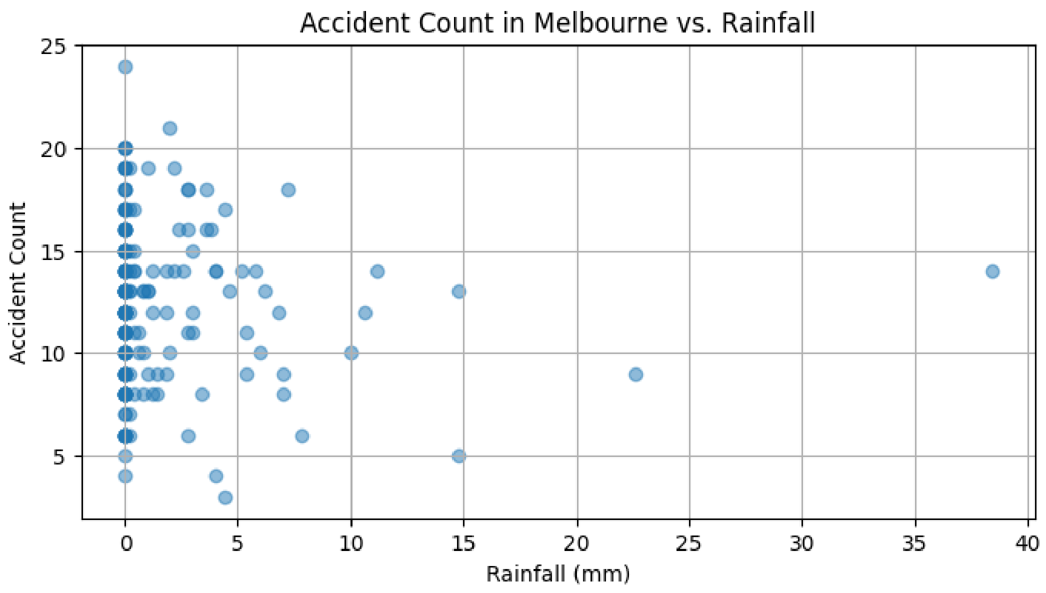


Figure 7: Number of Traffic Accidents VS Rainfall

4.3 Weather vs Air Quality

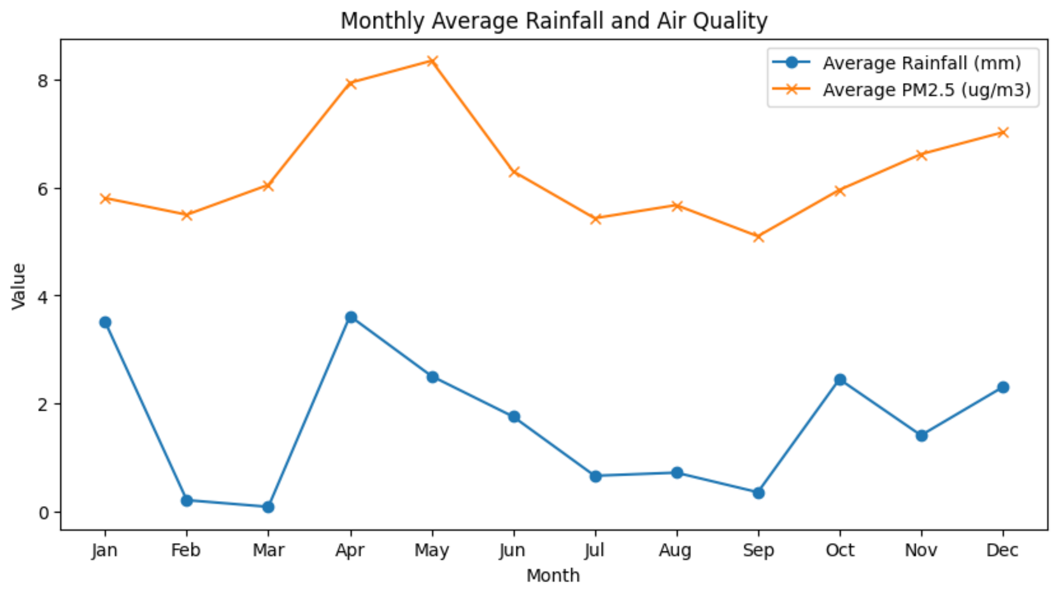


Figure 8: Monthly Trends of Average Rainfall and PM2.5 Level

Figure 8 compares the annual average monthly rainfall (mm) to the PM2.5 concentration in µg/m^3, which measures air quality. The average rainfall is shown by the blue line with dots, which fluctuates throughout the year, peaking in April and decreasing until September. The average PM2.5 concentration is shown by the orange line with crosses; it peaks in May and progressively declines until the end of the year, when it rises in the final few months. Similar trends are seen in these two variables, which might be due to increased rainfall in urban areas being accompanied by periods of heavy traffic, construction, or industrial activity, which might result in greater PM2.5 emissions.

4.4 Health vs Air Quality

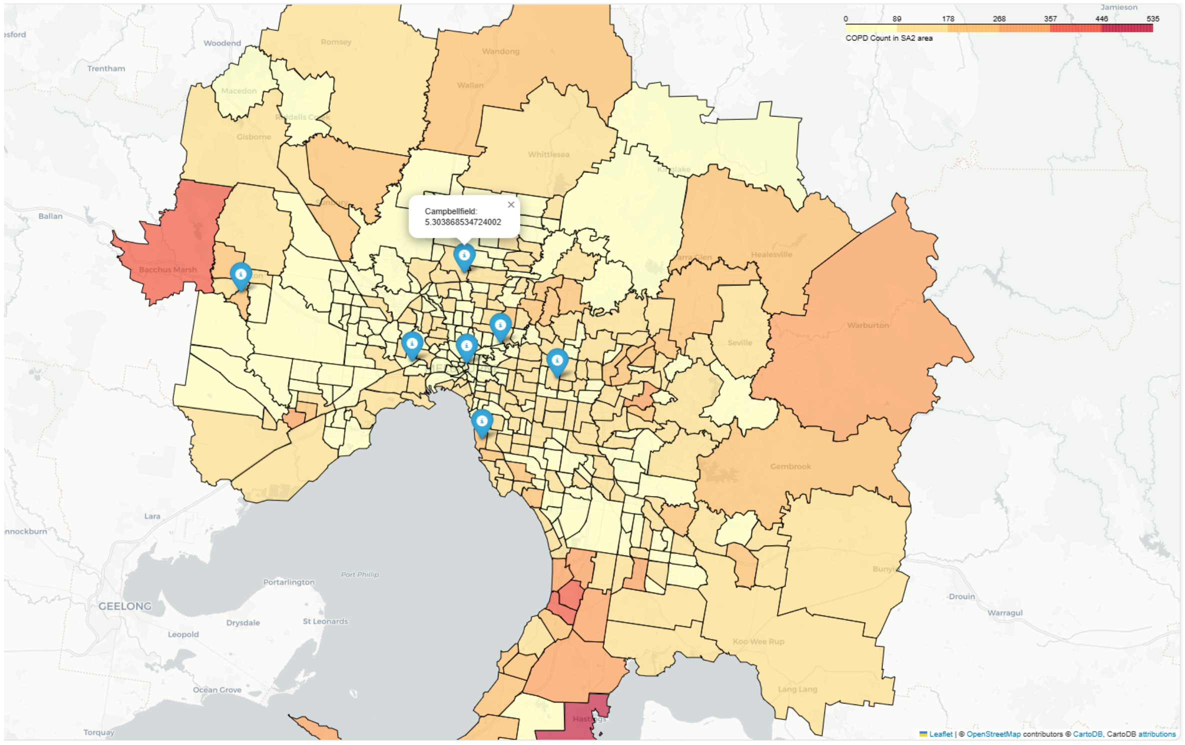


Figure 9: COPD Cases and Air Quality in Melbourne

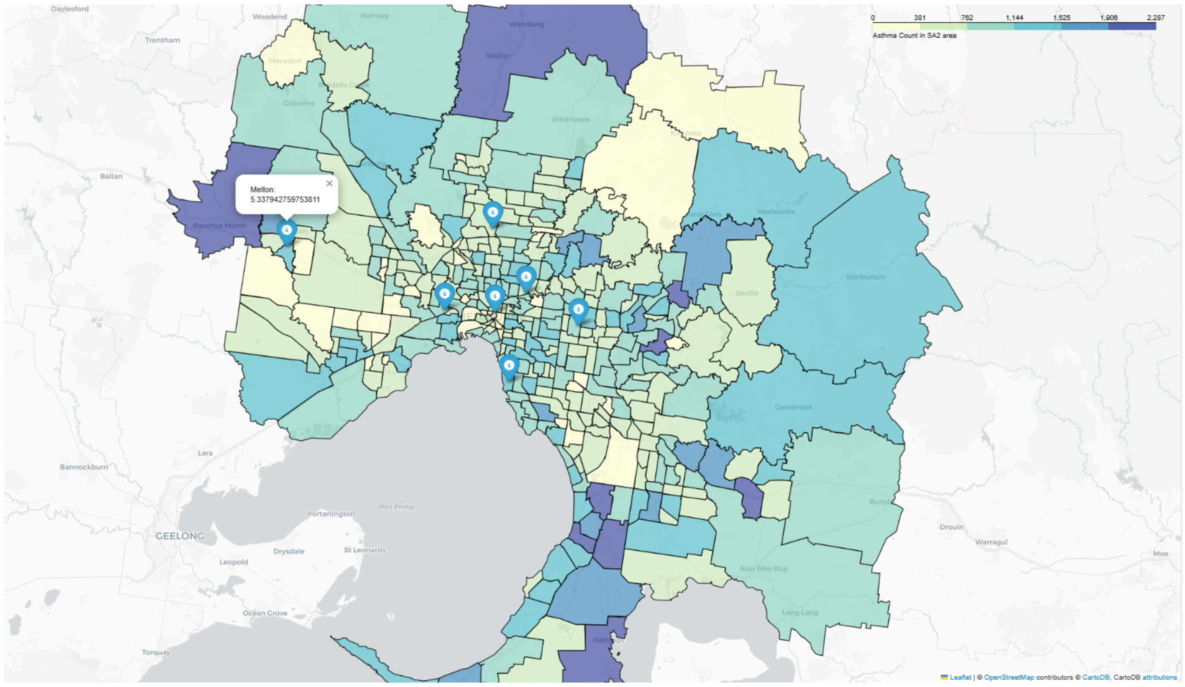


Figure 10: Asthma Cases and Air Quality in Melbourne

The geographic distribution of cases of COPD (chronic obstructive pulmonary disease) and asthma in the SA2 district of Melbourne can be observed in figures 9 and 10. Places with a higher number of patients are shown by deeper colours on the colour-coded maps that display the prevalence of COPD and asthma in different areas. Stations for measuring air quality are indicated by blue markers, users may view the BPM2.5 concentrations observed by clicking on the desired station. Based on the available data, it appears that regions with elevated PM2.5 concentrations do not always align with those with a greater prevalence of COPD or asthma. This implies that the primary cause of certain respiratory disorders may not be the concentration of PM2.5.

Apart from air pollution, there are several other risk factors that might be linked to health problem, including exposure to viruses, smoking, allergies, and occupational exposure. The incidence of COPD and asthma may also be influenced by the lifestyle choices and socioeconomic standing of the populace in various areas.

5 Error Handling

To handle errors properly, we utilise Python's exception mechanism. We have a try-except clause wrapping the function's body. When an exception happens, it will be captured by an except clause, and a detailed json response with a flag indicating whether the request is successful and an error message will be sent. For predictable exceptions, they are caught separately and handled accordingly.

Typical errors contain:

- URL parameter format error: the URL parameter is malformed or out of range.
- Arithmetic error: divide by zero or unexpected negative found.
- API request error: the API that the system harvests data from is temporarily down or the network is slow, request timeout happened.
- Database error: the ElasticSearch is down.

6 Problems Encountered and Solutions

1. **Mastodon rate limit:** Mastodon API limits rates to stop one user or app from sending too many requests in a short time, which could hurt the server and other users.

   Reasons for triggering rate limits:

   - Sending too many requests in a short period.
   - Using multiple instances to send requests simultaneously.
   - Failing to properly handle rate limit responses and continuing to send requests.

   Solution:

   - Pause for 3 seconds when no data is returned to avoid frequent requests.

- Pause for 5 seconds when an exception occurs to allow time for issues to resolve or recover.
- Pause for 1 second after successfully retrieving data to control request frequency.

2. **Error in getting average air quality:** When using the RESTful API to fetch the air quality data in ElasticSearch, the user is required to provide the name of the specific station, the start date, and the length of the time period. The execution often fails with the error message 'Response 400 - Bad Request', which indicates the client request has syntax errors or the request does not meet the requirements of the server.

   Reasons for bad request error:

   - Request API with incorrect format. For example, EPA stations with spaces in their names.
   - Missing necessary information.

   Solution:

   - Kibana supports non-whole keyword queries, which means users can type only 'Box' to fetch the information about the EPA station 'Box Hill' to avoid the space between words. This functionality also works in RESTful API.
   - Always convert the length of time period into String format when creating APIs in our client.

7 Limitation

Our project faces several notable limitations in data collection and analysis that must be considered when interpreting the results.

- **Data time Inconsistency:** The health data was gathered on SUDO and contains data only from 2021. As a result, we gathered data on air quality from 2021 in order to directly compare. However, we were forced to utilise weather data from 2023 in our search for connections between air quality and weather. We assume that each year has similar weather conditions, however, this assumption could not be accurate, which could compromise the validity of our results.
- **Limited Real-Time Data Sample Size:** Because of the relatively small sample size of our real-time data, it could be challenging to effectively describe trends and changes over time. This limitation might lead to less accurate correlation analysis and model predictions, especially when managing real-time data variations.
- **Monitoring Station Quantity and Distribution:** The BPM2.5 is the major focus of the air quality statistics, however, there are not many monitoring stations in SA2 region have this index, and most of them are centred in and around the city centre. Due to limited data support, this unequal distribution may lead to

an incomplete understanding of the general state of the air quality, especially for places outside the city centre.

- **Data Representativeness:** Our results might not accurately reflect the real circumstances of the whole research region due to the narrow scope of data collection and the unequal distribution of monitoring stations. Interpreting and using the results of the analysis will need careful consideration of this constraint.

Because of these limitations, attention must be paid when interpreting and implementing the results of the research to prevent making conclusions that are based on incomplete or inaccurate data. Expanding the coverage of monitoring stations and obtaining more temporally consistent real-time data are major objectives for future study to improve data representativeness and research accuracy.

8 Teamwork Management and Resources

8.1 Team Collaboration

To ensure effective collaboration and project management, our team utilises several tools and strategies:

- **Meeting Documentation with Notion:** All meeting information, including agendas, discussion topics, decisions taken, and action items, is recorded using Notion. Team members may remain informed and in synchronisation with project progress and duties within this unified repository. The date and subject organisation of meeting notes make it simple to look up previous conversations and follow the development of project tasks.

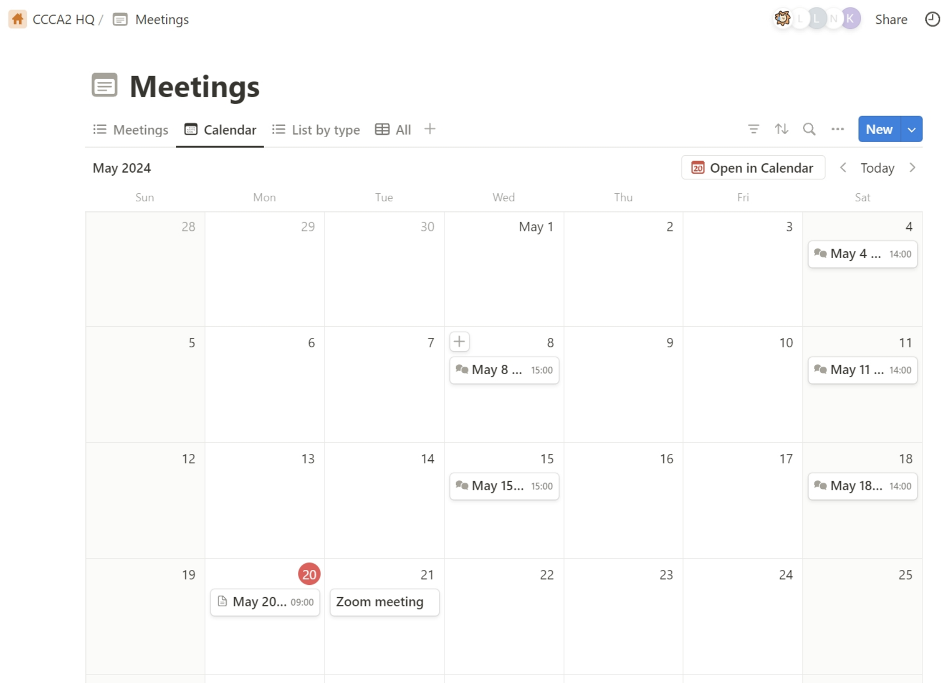
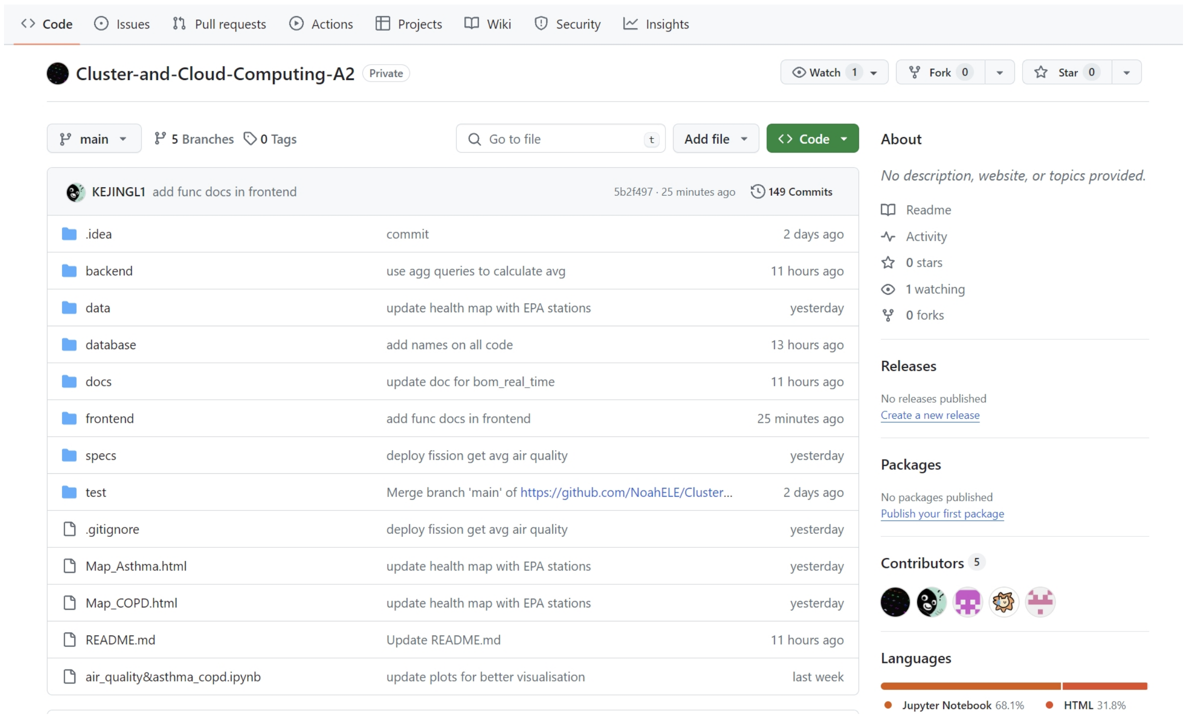


Figure 11: Notion Meeting Calendar

- **Regular Check-ins:** We have frequent team check-ins to go over progress, resolve issues, and make any required plan adjustments. In addition to ensuring that everyone is in agreement, these meetings promote open communication.
- **GitHub for Code Management:** GitHub is what we use for collaborative coding and version control. Because the repository is divided into branches for distinct features and tasks, all team members can work simultaneously without conflicts. Code reviews and pull requests are conducted to maintain code quality and facilitate knowledge sharing among the team.



Cluster-and-Cloud-Computing-A2 Private

KEJINGL1 add func docs in frontend 5b2f497 · 25 minutes ago 149 Commits

.idea	commit	2 days ago
backend	use agg queries to calculate avg	11 hours ago
data	update health map with EPA stations	yesterday
database	add names on all code	13 hours ago
docs	update doc for bom_real_time	11 hours ago
frontend	add func docs in frontend	25 minutes ago
specs	deploy fission get avg air quality	yesterday
test	Merge branch 'main' of https://github.com/NoahELE/Cluster...	2 days ago
.gitignore	deploy fission get avg air quality	yesterday
Map_Asthma.html	update health map with EPA stations	yesterday
Map_COPD.html	update health map with EPA stations	yesterday
README.md	Update README.md	11 hours ago
air_quality&asthma_copd.ipynb	update plots for better visualisation	last week

About

No description, website, or topics provided.

Readme
Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 5

Languages

● Jupyter Notebook 68.1% ● HTML 31.8%

Figure 12: GitHub Repository

Table 1 shows an overview of team responsibilities. In addition to these, all teammates contribute to data storage section, Fission deployments for data analysis, report writing, and demonstration video making.

Team Member	**Responsibilities**
Kejing Li	Collect data, Visualise data
Xin Su	Collect data, Visualise data
Xinhao Chen	Environment setup, Harvest data
Yueyang Li	Collect data, Visualise data
Zheqi Shen	Environment setup, Harvest data

Table 1: Overview of team responsibilities

8.2 Tools & Links

- Notion

- GitHub
- YouTube (System Demonstration)

9 Conclusion

9.1 Project Summary

This project utilises cloud computing resources from the Melbourne Research Cloud to analyse the influence of weather on various aspects of life in Melbourne. The project integrates diverse data sources such as the Bureau of Meteorology (BoM), Environmental Protection Agency (EPA), Spatial Urban Data Observatory (SUDO), DataVic, and the social media platform Mastodon. The system architecture which is built in Kubernetes, Fission, and ElasticSearch, provides us a scalable and robust platform for real-time and historical data analysis. By leveraging all these tools and datasets, we were able to gather, process, and comprehensively analyse how weather influences our daily lives.

9.2 Future Recommendations and Improvement Directions

While the project achieved its goals, we will continue researching the relative area and enhance our research method, including:

- **Enhanced Data Integration:** Incorporating additional data sources and variables could provide a more comprehensive understanding of weather impacts. For example, integrating data on economic activities or educational disruptions could further enrich the analysis.
- **Advanced Machine Learning Models:** Implementing machine learning algorithms to discover deeper connections between weather and other datasets, which allows for more accurate forecasting of weather impacts on traffic accidents, social media sentiment, and health outcomes.
- **User Interface and Visualisation:** Developing a user-friendly interface with interactive visualisations could make the insights more intuitive and accessible to the audience. Tools like dashboards can be integrated to better display the research result.