COMP90024 Cluster and Cloud Computing

Assignment2 Report

Jinuo Sun 1214174

Yulin Dong 1156680

Jin Wang 1266400

Zihan Xu 1174144

Gaoyuan Hao 1245712

1. Introduction

This project designs and implements a cloud data processing system based on Kubernetes and Fission framework on the OpenStack server platform. This system can automatically crawl and process raw real-time data through Fission Trigger, and store and manage it via Elasticsearch. At the same time, the lightweight Serverless deployment of Fission architecture makes it faster and easier to carry out analysis-intensive work. Function as a Service (FaaS) execution method and Kubernetes container management also make code maintenance more structured and orderly.

This report will discuss in detail the system architecture from design to deployment, focusing on the characteristics, operating mechanisms, advantages, and disadvantages of the Melbourne Research Cloud server platform and the Fission framework. This report will also show the data analysis tools developed based on Jupyter Notebook on the front end, as well as the entire data processing flow from data collection, and analysis to the final result displayed behind the system. Finally, this report will summarize the core issues and challenges faced in this design to provide more convenient full-process deployment and solution improvements in the future.

This report will explore several different scenarios that underscore the critical role of sustainable practices in urban environments. We aim to help people get to know more about the impact of the environment on daily life, find the potential linkage between humans and the environment, and ask people to focus more on climate protection.

Firstly, Scenario 1 examines how weather influences PM2.5 levels, serving as a foundational analysis that highlights the impact of weather on air quality. Based on this, Scenario 2 shifts the user's focus to the effect of air quality on people's sentiment in Australia and tries to illustrate the direct emotional impacts of pollution on residents around that place. Then turning to Scenario 3, it broadens the examination to include how general weather conditions can influence public sentiments. In Scenario 4, the discussion turns to travel mode preference across Australia, trying to find if people choose sustainable transportation options, and following the above scenario, we aim to emphasize the environmental benefits of choosing sustainable transportation options. Finally, Scenario 5 aims to highlight the relationship between air quality and the siting of child-care centers, promoting residential choices that prioritize environmental health, thereby ensuring safer environments for future generations. Each scenario is designed to build upon the last, create a collective and complex data analysis, and aim to champion eco-friendly policies and practices. The purpose of these scenarios is to demonstrate how the environment and pollution are relevant to our lives, what improvements we can make, and how to create a happier world for future generations.

2. System Architectures and Design

2.1 System Design Overview

This project designs an automated cloud computing system, realizing a complete cloud deployment system: from data crawling to preprocessing, then data collection into the database, followed by data analysis and model training in the cloud, storing the training/analysis results, and finally retrieving and visualizing them through the front-end Jupyter Notebook. This system's deployment relies on the Kubernetes platform and FaaS architecture, achieving automated script management and lightweight functional deployment. This architecture allows the complex and tedious data collection and analysis to be automated in the cloud, not only efficiently utilizing the cloud server's storage and computing capabilities but also avoiding the difficulties of local environment configuration and data transfer coordination. Thus, new data analysis tasks only need to write functions according to the FaaS structure requirements to utilize the cloud-deployed script architecture for automatic execution, and can also share existing data for further processing and analysis.

This section will start with the cloud configuration of the Melbourne Research Cloud, including the use of the front-end Jupyter Notebook application and the deployment of the back-end Fission framework, including Kubernetes container configuration. Additionally, it will provide an overview of Elasticsearch Database's data storage and query methods.

Appendix 3 provides a detailed visualization of the whole system architecture.

2.2 Melbourne Research Cloud

The cloud server used in this project is the Melbourne Research Cloud (MRC). MRC is a cloud computing server developed based on the OpenStack platform, providing a quick self-service portal to configure virtual machines. MRC offers automated resource management and scheduling, while also supporting rapid deployment. Its visual resource monitoring and management tools can greatly help with task allocation and debugging.

2.2.1 Resource Allocation

The resource configuration of this project is divided into the following aspects:

- Kubernetes：An elastic cluster was created using a 1 master and 3 nodes structure；
- CPU & RAM：Each node is configured with 2 virtual CPUs and 9GB of RAM memory (2c9g);
- Volume: Each node is allocated 30GB of volume by default.

It is worth noting that although the structure of 1 master and 3 worker nodes and their callable resources are predetermined at the time of creation, Kubernetes can still automatically scale Elasticsearch nodes up or down based on the specific load of tasks, optimizing resource usage.

2.2.2 Automation

The automated management of MRC mainly has two aspects: one is the automation of virtual machine creation and resource configuration provided by the OpenStack platform, and the other is the automatic allocation and rolling update mechanism brought by Kubernetes. At the initial creation of virtual machines, the OpenStack platform provides virtual machine templates and Ansible scripts. Users only need to follow the configuration guide provided by the portal step by step, and the OpenStack platform will automatically configure the network, storage, and computing resources as needed, greatly simplifying and accelerating the complex process of VM configuration. In addition, the OpenStack platform also offers dynamic expansion of storage and network bandwidth to ensure the stable and smooth operation of tasks.

Furthermore, Kubernetes provides mature automated management methods. Through the load balancing mechanism, it ensures the most balanced and efficient resource allocation and node performance utilization. The rolling update mechanism also guarantees seamless task execution for scenarios where container configurations change, and offers strong robustness by allowing rollback in case of potential errors.

2.2.3 Pros and Cons

2.2.3.1 Advantages

MRC is based on the OpenStack platform, thus having high availability and flexible scalability, which brings stable operation assurance and flexible resource scheduling for the project. The system scheduling is smooth, and the performance is efficient.

The multi-node configuration managed by Kubernetes not only improves security by preventing task stagnation or progress loss caused by single-node crashes but also allows isolating different applications within containers of a single node as needed. This increases overall operational stability and simplifies management, avoiding complex cross-deployment and updates.

As mentioned above, the combination of OpenStack + Kubernetes can achieve fully automated scheduling from the underlying construction to resource allocation. Once initialization is complete, it achieves a worry-free application effect with high stability and full automation. The visual portal provides direct and comprehensive resource monitoring and adjustment methods, which are clear and controllable compared to the Linux command line UI, reducing the difficulty of subsequent use to some extent.

2.2.3.2 Disadvantages

Although the user experience after construction is good, the initial configuration process involving complicated template installation, environment setup, management linkage, and resource allocation can be confusing and easily configured in ways not best suited for tasks. Additionally, due to the overall complexity of the configuration, subsequent daily maintenance, monitoring, and troubleshooting require relatively more effort to ensure the stable operation of the cluster regularly.

At the same time, virtualization and containerization technologies require additional computing and storage resources. Compared to simpler and more direct HPC, the infrastructure cost of servers increases, and the performance overhead of self-maintenance and the non-centralized nature of elastic scheduling can also become constraints for high-performance tasks.

2.3 Front-end

2.3.1 Jupyter Notebook Application

Our Jupyter Notebook acts as the front end of our system, it’s designed as a multifaceted tool for analyzing the impact of environmental and social factors across Australia. It features dynamically updated datasets and offers users the ability to access data through our self-designed RESTful API. Also, we provide the function of retrieving the data for a specific period, and the user can conduct targeted analysis based on the most relevant data in a specific period.

Then, the jupyter notebook supports analyzing several different scenarios using the dataset of weather, air quality, and mastodon. These three datasets are all lively updated.

For Air Quality vs Sentiment Analysis: It allows users to explore the relationship between air quality and the sentiment of the public living in Victoria. Helps to understand how environmental conditions influence people's moods and opinions.

For Weather vs Sentiment Analysis: Similar to the Air Quality, this function can help to analyze the correlation between different weather conditions and sentiment score, providing insights for users to know how weather impacts emotions and perceptions.

For Air Quality vs Weather: This tool helps the user to compare air quality with weather patterns. It can potentially be useful for studies focusing on climate control and environmental management.

For Language Distribution Analysis: To get to know more about people living in a different city and help the traveler or new resident to decide the place to stay, this tool examines the geographical distribution of different languages spoken around the different cities in Australia. It is crucial for social, cultural, and linguistic studies.

For Travel Methods Analysis: This tool explores the various traveling methods used by people living in different places in Australia, offering valuable data for transportation planning and policy-making.

We set a tool for checking daily weather and PM2.5 values by generating several diagrams based on today's dataset.

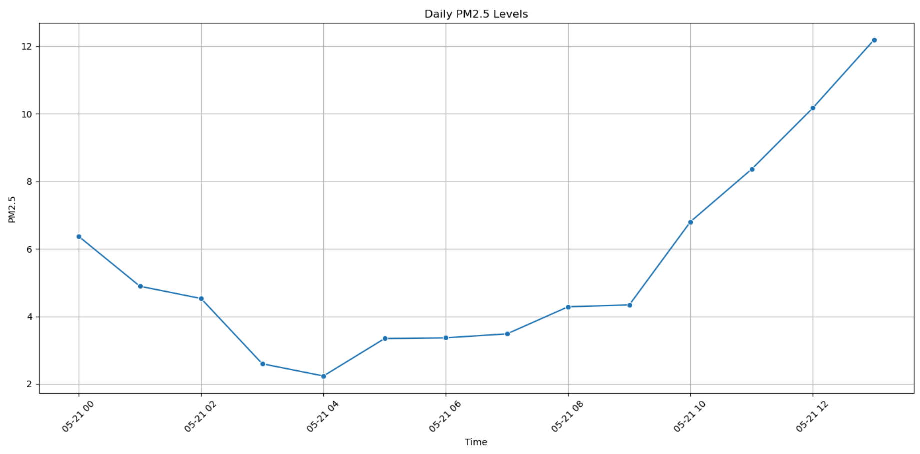


Figure 0. *PM2.5 showcase*

2.4 Back-end

2.4.1 Fission

In this project, the Fission framework is adopted as a lightweight deployment method. Fission relies on the container orchestration and resource scheduling functions provided by Kubernetes, packaging the dependency environment into independent containers. It deploys function-based servers in a lightweight manner via FaaS, with actions executed through trigger management.

Throughout the execution process of Fission, functions serve as behavior carriers, and deployment is event-driven. Fission allows developers to focus only on developing task execution logic without worrying about the tedious server deployment. The modular function calls not only enhance reusability and bring convenience to automation but also make the deployment process clear and controllable.

2.4.1.1 FaaS

Fission provides a Function as a Service (FaaS) deployment platform. As the name suggests, the FaaS platform calls functions in Python scripts to execute tasks instead of a server, eliminating the tedious server deployment logic. FaaS deploys function tasks in the most efficient way, with flexible scheduling to expand or shrink function instances according to task load. It also ensures that function instances are discarded after completing tasks, thus saving cloud computing resources effectively.

2.4.1.2 Docker/Container

The environment configuration required for the execution of function scripts called by Fission is provided and managed by containers created by Kubernetes. Kubernetes, as the orchestrator of containers, allows Fission to deploy, scale, and manage through its own provided interface. Fission's container management based on Kubernetes is divided into the following levels, from the underlying execution environment to specific function dependencies:

- Fission uses the env command to create environment containers through Kubernetes from pre-defined Docker Images during initial deployment, which is Python in this project. This container is the execution environment for all functions, representing the lowest level of Container;

  fission env create --name python --image fission/python-env --builder fission/python-builder

- Kubernetes automatically manages Pods, a minimal executable deployment unit, which contains Fission's functions and routes, automatically scaling up or down based on execution conditions;
- Fission deploys function code and dependencies to Kubernetes through compressed packages. Kubernetes creates separate containers for the function's environment to meet its runtime dependencies.

  fission package create --name <package> --sourcearchive <package>.zip --env python --buildcmd './build.sh'

At this point, Fission's container deployment on Kubernetes is complete, meeting the runtime conditions required by its Function Server. Containers for each function are independent and functionally isolated, not interfering with each other, ensuring high execution stability.

2.4.1.3 Route

Fission uses routes to define the mapping between HTTP paths and functions, calling function requests via HTTP. Routes act as intermediaries connecting Fission commands and the functions to be deployed in Kubernetes containers, achieving ultra-simplified custom commands in a concise deployment manner. After uploading the aforementioned package, the deployment method of the Router in the Fission Client is as follows:

- First, define the entrypoint in the Package, which will be the main dispatcher for the Router in the Package:

  fission function create --name <func> --env python --pkg <package> --entrypoint "<func>.<entry>"

- After defining the main dispatcher of the container, the Router uses --url /<func> to define the HTTP command method as the communication way between the inside and outside of the container. The dispatcher <func>, upon receiving the /<func> command, performs the corresponding function operations through the internal entrypoint <entry>, or verifies and forwards the command through an internally defined flask app.

  fission route create --name <func>Route --function <func> --url /<func> --method GET

At this point, Fission Client can conveniently and intuitively invoke and run function instances into the independently created containers in Kubernetes through ultra-simplified custom commands. Appendix 2 Fission README.md shows our detailed API design with regard to Fission Route.

2.4.1.4 Trigger and Automation

The automation of Fission deployment is achieved through Fission Triggers. The Route deployment mentioned in the previous paragraph is actually a manifestation of the HTTP Trigger and is one of the cornerstones of full automation. In addition, the following triggers can be added as needed (in this project, the main ones used are Timers with different intervals):

- Timer Trigger(where *--cron* is the expression to define trigger frequency):

  fission timer create --name <timer> --function <func> --cron "@every 1h"

- Kafka trigger (where *--topic* is the trigger topic, *--resptopic* is the corresponding topic):

  fission mqtrigger create --name <kafkatrigger> --function <func> --mqtype kafka --topic <topic> --resptopic <resptopic>

- Watch trigger (where *--type* is the type of resource being watched):

  fission watch create --name podwatch --function <func> --namespace default --type pod

Among the above methods, Timer provides a way to automatically trigger specified functions periodically; Kafka provides automation for monitoring and relaying; Watch provides automatic responses to Kubernetes pods. These methods offer simple automation scheduling solutions for tasks with multiple parallel and real-time data sources. They can also be combined with bash scripts and Routes defined by HTTP Trigger for complex behavior/responses automation. At this point, Fission's full capabilities are gradually revealed. Its highly customizable automated responses can fully deploy in complex and real-time updating scenarios, significantly reducing the deployment pressure on developers.

2.4.2 Pros and Cons

2.4.2.1 Advantages

The serverless architecture and function mechanism brought by FaaS allow developers to focus on code, with flexible scaling making resource utilization more efficient and centralized. The event-driven function trigger mechanism enables more flexible automated deployment, making it very suitable for real-time data processing and analysis in the cloud. For analysis-intensive tasks, the Fission framework greatly reduces the complexity of infrastructure management, enabling the rapid construction of analysis platforms. This allows analysts to focus on logical development, significantly increasing workflow efficiency.

2.4.2.2 Disadvantages

Although FaaS saves computing resources through the idle function reclamation mechanism, it also causes cold start delays due to re-instantiation every time a function is restarted. Additionally, the stateless mechanism of functions means that state information needs to be pre-configured and stored outside the container for the function within the container to access or update. Therefore, state management introduces new complexities, requiring developers to check the consistency of state storage and retrieval to ensure the function works as expected. Due to its serverless architecture, the hidden nature of errors makes monitoring more complex. If things do not work as expected, significant effort may be required for troubleshooting.

Furthermore, MRC imposes execution time and computing resource limits on FaaS-style function deployment. This means that the intensity of tasks needs to be manually controlled to ensure that tasks are not forcibly interrupted by the server, causing process loss.

2.5 Elasticsearch Database

The data storage structure used in this project is Elasticsearch Database. Elasticsearch is a distributed search and analysis engine that can search and process large amounts of real-time data and fundamentally supports large-scale data sharding and replication, thus having higher parallelism and fault tolerance. Elasticsearch database uses inverted indexing to achieve fast query, while compressing data storage to save storage resources, making it more suitable for cloud-based big data. Moreover, this database supports storing multiple data types simultaneously and complex query requests, thereby handling massive data more flexibly. The Elasticsearch framework also allows scalable nodes and shards to accommodate growing data volumes and more complex query needs.

The characteristics of the data in this project are real-time and growth, and there is a need for cloud deployment. Data management and analysis rely on the fission framework, and data behavior management is periodically automated deployment by fission trigger. Therefore, the efficient storage of real-time information, fast querying of complex queries, scalability, and modular deployment capabilities of Elasticsearch are precisely the ideal data management systems sought by this project.

In this project, Elasticsearch database designs independently Create, Read, Update, and Delete methods, including automated data validation, while packaging the mappings of corresponding data categories into library scripts, and ultimately calling through the HTTP router at any time by fission via each custom independent route of the Flask Application. Flask Application helps simplify fission deployment, creating a streamlined set of instructions through unified route management, and improving structure and reusability. In addition, flexible Query methods are defined to achieve customized data acquisition, including key-merge and on-demand filtering.

Appendix1 shows the four main databases supporting the system, namely air_quality, madstone, weather_condition, and child_care. Appendix1 also details their respective mappings, data types and structures. Appendix4 shows the dataset deployed inside Elasticsearch DBMS. Detailed data information would be discussed in the following section.

3. Data Collection and analysis

3.1 EPA Data

3.1.1 Collection & Preprocess

We collect real-time air quality data from this website, which is 'Environment Protection Authority Victoria'. This is a government agency responsible for monitoring and managing environmental issues, particularly air quality, in the state of Victoria, Australia. They enforce environmental regulations and provide information to the public about environmental issues. The raw data covers most areas in Victoria and includes many air quality parameters like PM2.5 and PM10. However, Not every area has the same air quality parameters, but most areas have PM2.5 data. So, we first removed other data reports except PM2.5. To match our study, we only kept data from Melbourne suburbs. We found that some areas, like Melbourne CBD, might have missing data sometimes. So, to ensure that the data we collect is not null, we decided to drop that null data and combine all suburb data to get an average PM2.5 value for each hour in Melbourne. We added the hourly average PM2.5, the time, and the area to our database. This method helps us get accurate air quality data for our analysis and provides a strong data base for our study.

3.1.2 Analysis

Figure 1 shows different periods in May. The vertical axis represents PM2.5 concentration levels, and the horizontal axis represents specific dates. The shaded area around the line indicates the variation in the data. According to this figure, it can be seen that the average PM2.5 has been decreasing over time until the 17th, indicating that the air quality in Melbourne improved during this period. However, starting from the 17th, the concentration of PM2.5 in the air gradually increases, which may be due to recent construction near Melbourne Central.

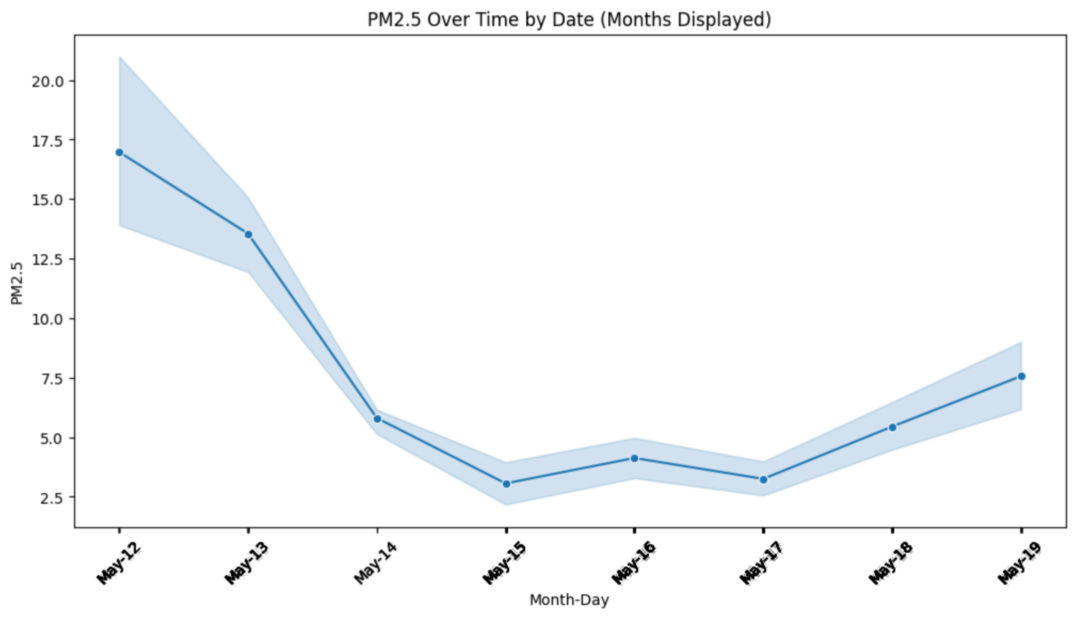


Figure 1. *PM2.5 Change over Time*

3.2 BOM Data

3.2.1 Collection & Preprocess

The Bureau of Meteorology (BoM) is Australia's national weather, climate, and water agency. The BoM website provides real-time weather updates, forecasts, warnings, and climate information to help individuals and organizations make informed decisions about weather-related events and conditions.

We collect hourly real time weather data from this website. However, we encounter many problems.The raw data has many useless attributes like 'wmo' and 'history_product', which are not relevant to weather data. Some attributes have large amounts of missing values and Some attributes have fixed values for all the time. Hence We removed these useless attributes and kept useful ones like name, full local date and time, latitude, longitude, apparent temperature, temperature difference, wind gust in km/h, wind gust in knots, air temperature, dew point, pressure, QNH pressure, mean sea level pressure, rain trace, relative humidity, visibility in km, wind direction, wind speed in km/h, and wind speed in knots. The remaining attributes are useful and suitable for data analysis.

After selecting the related attributes, we found that these weather data update every half hour. Each time we collect data from midnight to the current time. To match the air quality data, we collect data on the hour and only take the first two records of each hour. For example, we combine the 4:00 and 4:30 data. We average the numerical attributes and combine the categorical attributes into a list. This way, we can add the latest hourly weather data to our database. This ensures consistency and accuracy of the data, helping us compare air quality and weather data on the same time scale. It allows us to better understand and explain the impact of weather conditions on air quality. This improves our data analysis quality and provides strong data support for future scientific research and policy-making.

3.3 Mastodon Data

Mastodon is a decentralized social network platform where user can choose to communicate and interact on different servers. As it is an open-source platform, we can collect data from Mastodon using its API and analyze them to gain insights of user behavior, sentiment, daily habits, and trends. Here is the detail of collecting data from a Mastodon server.

The server we chose for this project is “aus. social” which is a Mastodon instance for people in Australia. However, before starting data collection, we get a valid Mastodon account, a private access token generated by my account from the Mastodon instance, and the Python libraries we installed is ‘requests’, ‘json’, ‘re’, ‘datetime’, ‘textblob’, and ‘beautifulsoup4’ for solving the HTML tags carried by the raw data. And the function was packaged inside the mastodon folder.

For the function we deployed on Fission, we set a trigger for controlling the function to collect data from Mastodon server every five minutes and all data between ten hours and five minutes to ten hours will be collected. What have to be aware of is the server only allow visitors to collect the data generated before 10 hours from the current time, which means the newest data we can get is 10 hours before now.

The data collected from the instance is full of html tags, contents, links, photos, etc… Therefore, we need to apply pre-processing methods to raw data. The first step is data cleaning to remove the invalid content and extract useful information. For example, we use ‘get_tokens’ function to process content to remove invalid characters and punctuation and keep the keywords for further analysis.

Then, we want to do some analysis about the sentiment, which is crucial to understand user emotions and opinions. We use “TextBlob” library to perform sentiment analysis on each toot and get the sentiment polarity scores. The score will be ranged from -1 to 1 and negative values means there is negative sentiment, positive value leads to positive sentiment, and 0 means neutral sentiment.

After that, cleaned and analyzed data will be stored in a JSON file for further analysis. For example, we can use word cloud to find the popular words appear in the dataset as shown below. And these keywords give us some idea about extracting some keywords, by combining with other data source.

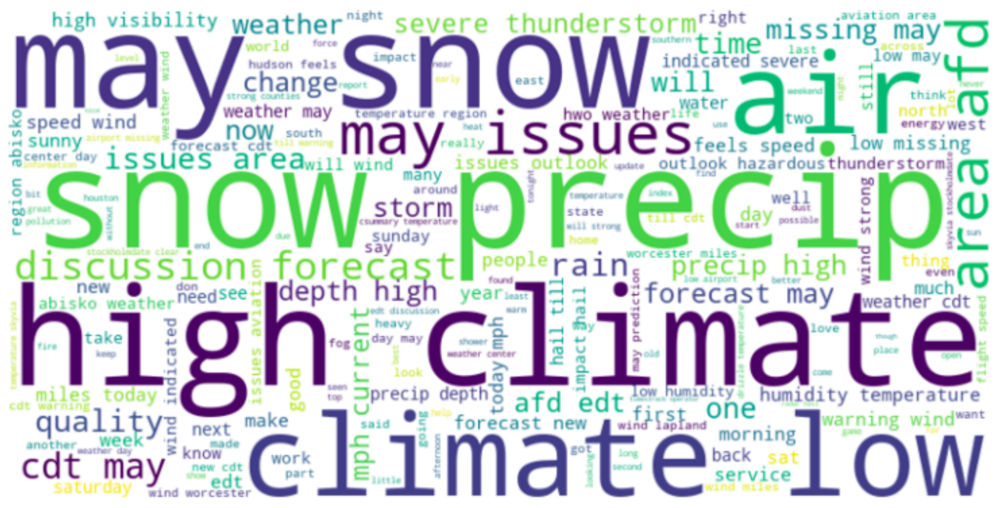


Figure 2. *description*

Also, we check the distribution of the sentiment score, and we found most of them are around neutral sentiment, and there are some strong positive and negative sentiments.

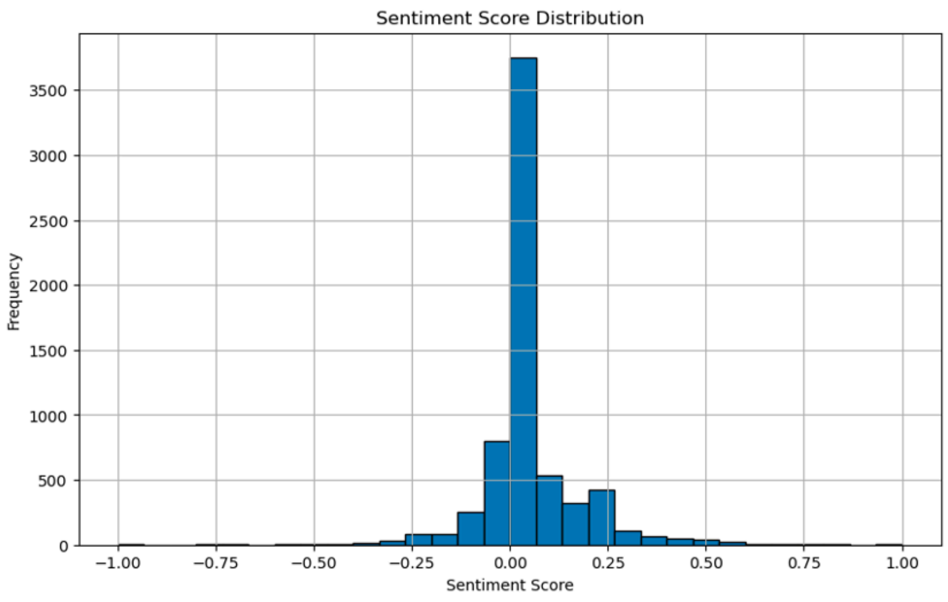


Figure 3. *distribution of sentiment*

Based on this observation, we decided to do some further analysis focusing on what factors could affect Australian sentiment and learn more about the living habits of Australian residents. And we can use this system to get to know more about people living in Australia.

4. Scenario result analysis

4.1 Select keyword & Demonstration of the web page

EPA: Once we sign in to the EPA website, click on the ‘All Air Monitoring Sites with Scientific Parameters’ selection, input the primary key we get when we subscribe to the API, and input the only parameter which is the siteID = air, interval = 1hr_av which is one hour. After that click response, we can have the air quality data.

BOM : To reach our target web page, which has real-time weather information from BOM, follow these keywords in order: Australia, Victoria, Victorian Observations, Latest Weather Observations for Victoria, and Melbourne.

4.2 Result Analysis

4.2.1 Scenario 1: How weather can affect PM2.5 levels in Melbourne.

Studying the relationship between air quality and weather is very important because poor air quality, especially with high levels of PM2.5 and other pollutants, can lead to respiratory and cardiovascular diseases. Some weather factors such as temperature, humidity, and wind may also affect the spread of pollutants like PM2.5, thereby impacting air quality. Therefore, analyzing and understanding this relationship helps improve public awareness of how weather changes affect air pollution, providing health advice for different weather conditions. For example, reminding people to reduce outdoor activities when the air quality is poor and encouraging outdoor activities in good weather.

The data for this scenario mainly comes from the monitoring times of weather (4.3) and air quality (4.1), combined for comprehensive analysis. We use the Pearson correlation coefficient to quantify the relationship between each factor and then visually depict it as a heatmap in Figure 4. These correlations help us understand how weather factors impact air quality and provide useful insights for air quality prediction and environmental policy-making. According to that figure, there is a correlation between PM2.5 and various weather factors. Specifically, PM2.5 has relatively positive correlation with apparent temperature (correlation coefficient 0.47), air temperature (0.39), and dew point (0.31). This means that as these factors increase, the PM2.5 concentration tends to increase. The possible reasons for the positive correlation are as follows: higher apparent air temperatures usually mean less wind and more stable air quality, leading to the accumulation of pollutants and thus increasing PM2.5 concentration. A higher dew point indicates more moisture in the air, which can cause pollutant particles to coalesce, leading to higher PM2.5 concentration. Therefore, as these weather factors increase, the PM2.5 concentration also tends to rise.

On the other hand, PM2.5 has relatively negative correlation with gust speed (km/h: -0.33, knots: -0.33), air pressure (-0.39), and wind speed (km/h: -0.29, knots: -0.28), indicating that as these factors increase, PM2.5 value tends to decrease. This suggests that higher wind speeds, gusts, and sustained winds may help disperse pollutants and reduce their concentration in the air. Generally, when the wind becomes stronger, it mixes the air more effectively, spreading pollutants to larger areas, thereby reducing PM2.5 levels. Additionally, Higher atmospheric pressure usually means more stable weather conditions, which typically correspond to clearer skies and less stagnant air. This stability can better disperse particulate matter in the air, thereby reducing the concentration of PM2.5. Moreover, higher wind speeds (in kilometers per hour and knots) may enhance the dilution effect of pollutants. This increased air flow prevents the accumulation of PM2.5 in local areas, thereby reducing the total concentration.

Finally, PM2.5 has a weaker correlation with temperature difference (0.28) and relative humidity (-0.21).

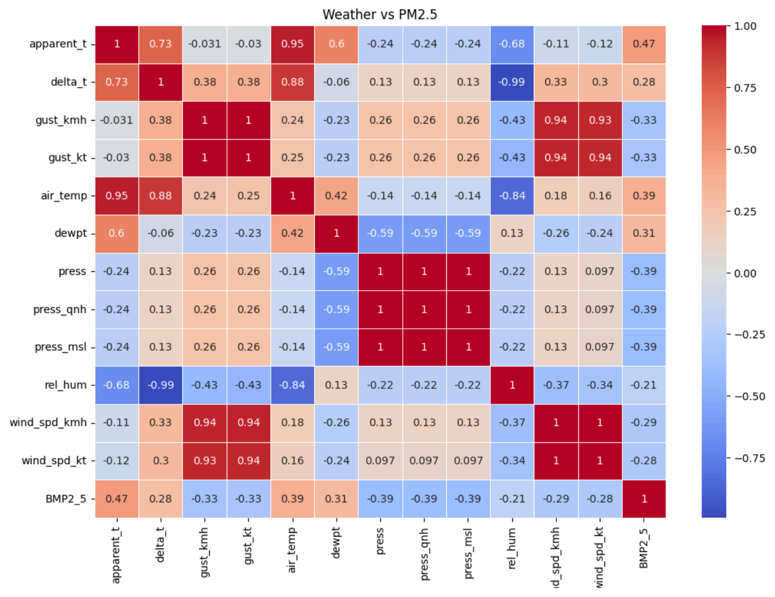


Figure 4. *correlation heatmap of weather data and air quality*

Model Name	MSE	R^2 Score
Linear Regression	16.741	0.423
Random Forest	7.834	0.730
Support Vector Machine	11.678	0.598

Table 1. Performance of each model

To better assist residents in Melbourne in understanding daily air quality, we decided to develop a model to predict PM2.5 levels based on the real time weather data.

Considering that PM2.5 is a numerical variable, we choose to train three models: linear regression, random forest, and support vector machine. The results of model training are shown in Table 1, indicating that the performance of the random forest model is better than the other two models, because it has the highest R-squared score and the lowest MSE.

The reason for this advantage may lie in the lack of a linear relationship between weather data and PM2.5 levels. The Random forest model can effectively capture complex nonlinear relationships between features and target variables by constructing multiple decision trees and combining their predictions, Therefore it has the best performance. Additionally, support vector machines (SVM) may outperform linear regression because they can capture complex data relationships by mapping the data to a higher dimensional space using kernel techniques.

In summary, the mean squared error (MSE) of our best random forest model is 7.834, indicating a mean square error of approximately 7.834 between the predicted PM2.5 concentration and the actual value. The R^2 score of 0.730 indicates that the model explains about 73.0% of the variation in PM2.5 concentration, showing that a significant portion of the variability in the target variable is captured.

Through studying the relationship between weather and PM2.5 levels (Scenario 1), we have clarified the crucial role of weather factors in influencing air quality, thereby providing the public with more comprehensive air pollution forecasting information. Based on this, in order to better serve the public, we need to further explore the impact of air quality on public sentiment (Scenario 2) to reveal its direct influence on residents' mental health.

4.2.2 Scenario 2: Evaluating the Impact of Air Quality on Public Sentiment in Melbourne

This research scenario explores how weather conditions affect PM2.5 levels in Melbourne. Understanding how different weather conditions, such as wind speed, rainfall, and temperature, affect the concentration of PM2.5 in the air can help predict and control air pollution events. This study is crucial for environmental scientists and policymakers as it provides scientific evidence to reduce air pollution and protect public health.

Firstly, based on the word cloud shown in *Figure 2*, we can see there are many conversations about today's weather and air quality. For example, the words 'snow', 'air', 'climate', etc… Therefore, we decided to make a scenario focused on finding the connection between air-quality and people's sentiment at that place.

The air quality data was from 'Environment Protection Authority Victoria' which is the most authoritative pm2.5 detection organization in Victoria, which means we will focus on people living in Victoria for this scenario. And they are mostly living in Melbourne. The data include the information of the time, location, and BMP2.5 (basically the measurement of pm2.5).

Before the analysis, we merge the data from EPA and Mastodon using their time and location. Since no labeled location exists in Mastodon data, we decided to categorize the location based on the tag of each content. For example, if the content is tagged "Melbourne", we define it as happening in Melbourne. And since the air quality data is collected hourly, we match air quality data with the Mastodon record created within 1 hour before or after the time of the air quality data. We categorized the air quality into several categories (good, moderate, etc) and then calculated the average sentiment score for each category.

Finally, we use bar charts to visualize the result, the graph is attached below. As some of the Mastodon data cannot be matched with the air quality, for example, there are Mastodon data from regions other than Victoria, or there is no location information shown in the content.

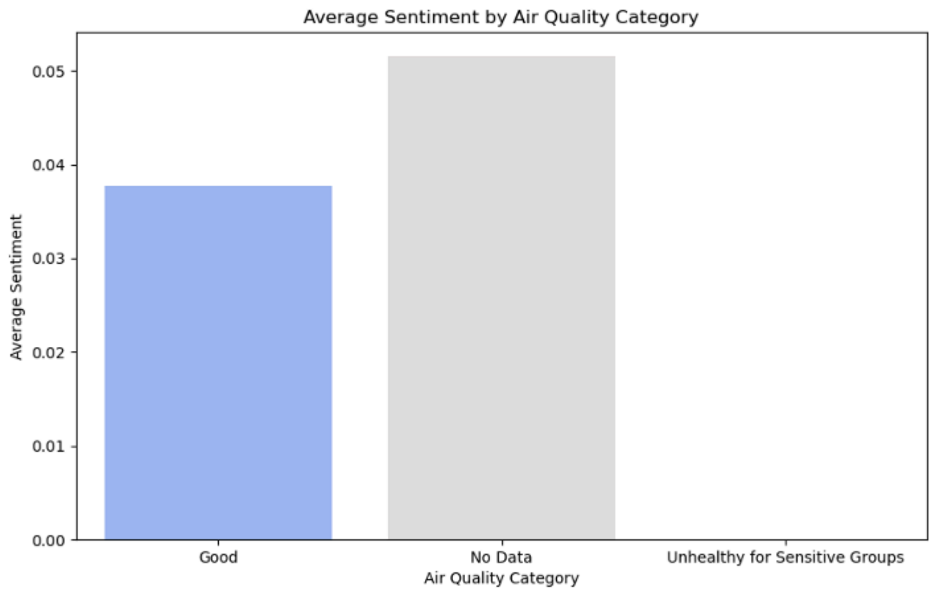


Figure 5: *Average value of sentiment per air quality category*

From Figure 5, we can see the average sentiment score under good air quality is around 0.4, which indicates a positive sentiment for people in Victoria. However, for the content generated under an unhealthy air quality, the average sentiment score is around 0, which indicates a neutral sentiment. Following this result, we can see air quality may have a significant impact on the sentiment of people in Victoria. Good air quality tends to uplift people's sentiments, and by continuously collecting and updating real time information, we can gain a better understanding of how people's sentiment is linked with the air quality.

After understanding that good air quality can bring a positive impact on people's sentiment, in order to have a more comprehensive understanding of the sentimen created, we plan to analyze how weather factors (such as temperature, humidity, etc.) affect residents' emotions, further deepening our understanding of the impact of environmental changes.

4.2.3 Scenario 3: Assessing the Impact of Weather on Daily Public Sentiment

This research scenario focuses on the impact of weather conditions (such as temperature, humidity, rainfall, etc.) on the daily emotions of the public. By analyzing weather and emotional data, it is possible to determine whether specific weather conditions can cause fluctuations in people's emotions. This is of great significance for urban planning and public health management, as it can help develop adaptive strategies to cope with the impact of extreme weather on public emotions.

We analyze how daily weather conditions affect the general public's sentiment by combining weather data (temperature, wind_gust, etc.) from the Bureau of Meteorology (BoM) with public sentiment data (collected from mastodon, 4.4) based on their time and location.

Due to the absence of marker positions in the Mastodon data, we decided to classify the positions based on the labels of each content, similar to the previous scenario, and then combine them with the weather data set.

We employed Pearson correlation to assess the relationship between each weather factor and public sentiment, and then visualized the results using a heatmap(Figure 6). From this, we observed that all weather factors exhibit a very poor negative correlation with sentiment. This suggests that while Melbourne residents' sentiment may slightly deteriorate in adverse weather conditions, overall, their sentiment is not significantly affected by the weather.

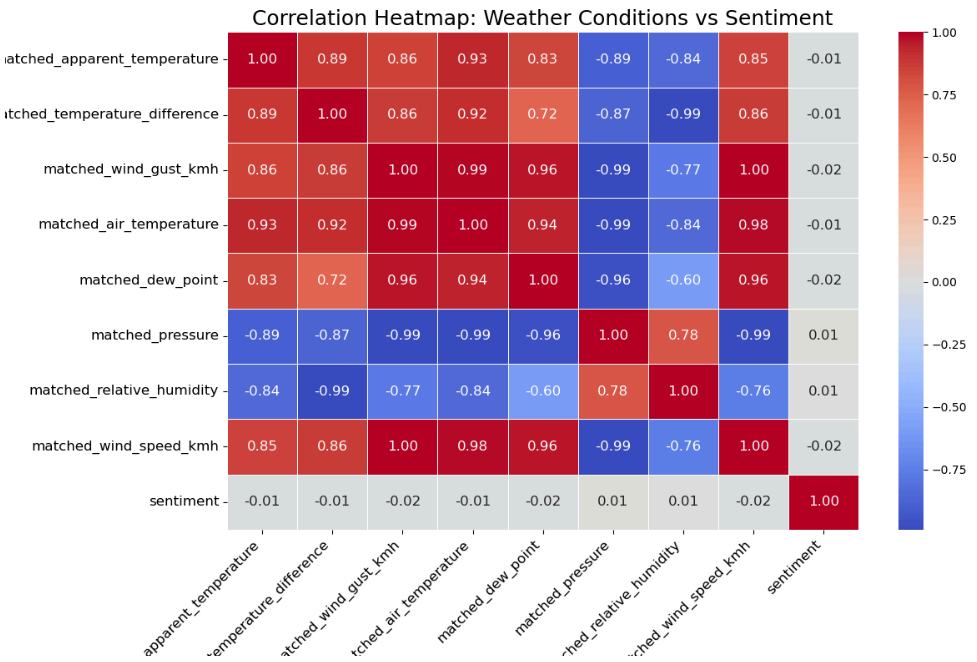


Figure 6: *correlation between weather and sentimen*

After discussing the impact of environmental factors on emotions, this scenario shifts to analyzing the preference of transportation. By comparing the transportation modes of different cities, we can gain a more comprehensive understanding of the lifestyle and environmental preferences of residents.

4.2.4 Scenario 4: Analyzing Travel Mode Preferences Across Australia

In this scenario, we only use the data collected from Mastodon. By searching for keywords like 'car', 'automobile', 'vehicle' can be categorized as travel by car, 'bus', 'coach' can be categorized as travel by bus, etc.

By using these processed data, and the location shown in the tags, we can find the commonly used travel vehicles people use in different cities and around Australia. For example, the bar chart below is the popular travel modes in regions all around Australia.

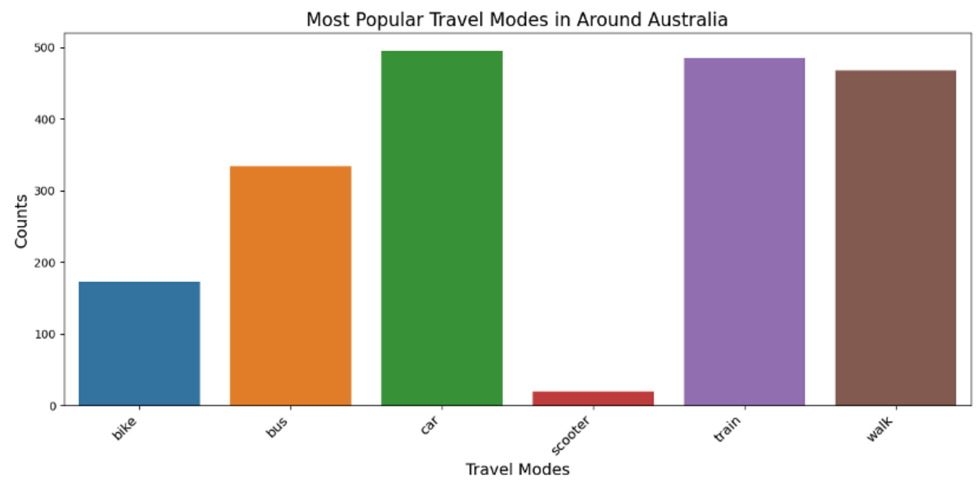


Figure 7: *Counts of each travel modes*

According to figure 7, we can see people like to travel by car and train instead of scooters and bikes. However since the data is only generated from Mastodon, and maybe not all data has been processed clearly. For example, maybe they just mentioned the vehicle, but didn't use it as the travel method in the end, in this case, we need to do some further analysis in the future. But based on the given information, we can see people nowadays is not inclined to choose low-carbon transportation options like bicycles and scooters. We can try to encourage these travel modes to decrease pollution.

After analyzing the transportation modes of residents in various cities, we will finally evaluate the environmental quality of households when choosing their place of residence, taking into account air quality and the geographical location of childcare centers. This not only summarizes the previous analysis but also provides a practical perspective for applications to help families make healthier lifestyle choices.

4.2.5 Scenario 5: Child-care Center Information and Air Quality Analysis

This scenario explores the relationship between air quality and childcare centers across Melbourne. By analyzing these datasets, we aim to identify areas with both high numbers of childcare facilities and favorable air quality, offering valuable insights for families seeking suitable residential neighborhoods.

We have collected air quality data from different regions over the past period of time from EPA, while nursery information, sourced from Sudo, provides details on licensed childcare providers. By merging the geographical coordinates of childcare centers with nearby air quality data, we created a comprehensive dataset for analysis.

The distribution of average air quality is depicted in Figure 8, illustrating the number of childcare centers within different air quality ranges.

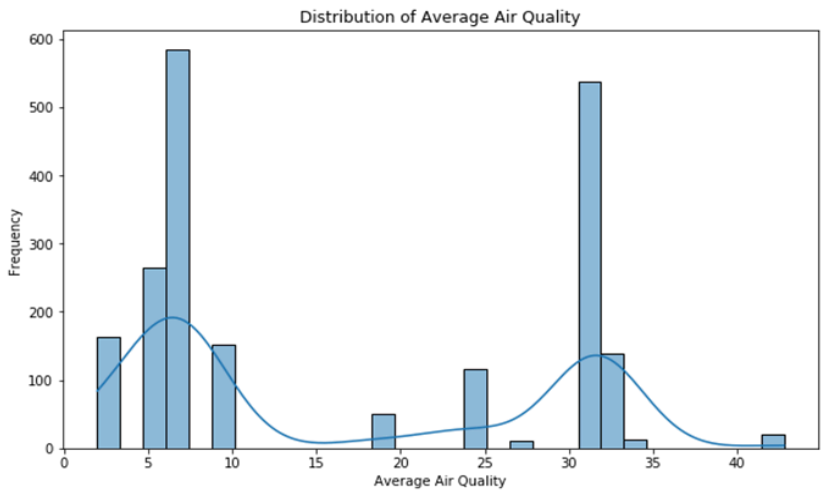


Figure 8: *distribution of Average Air Quality*

From Figure 9, we observe that the majority of childcare centers (over 800) are situated in areas with high average air quality scores between 5 and 10. Interestingly, the second-highest concentration of childcare centers exists in areas with relatively poorer air quality, with scores ranging from 30 to 35 on average.

The map in Figure 9 presents air quality variations across different geographic locations.

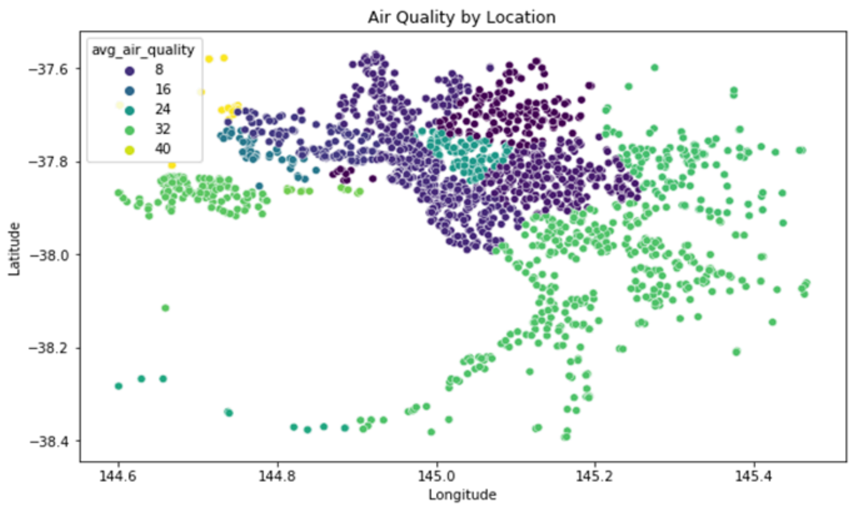


Figure 9: *Air Quality by Location*

Figure 10 further highlights the top 10 suburbs with the best air quality.

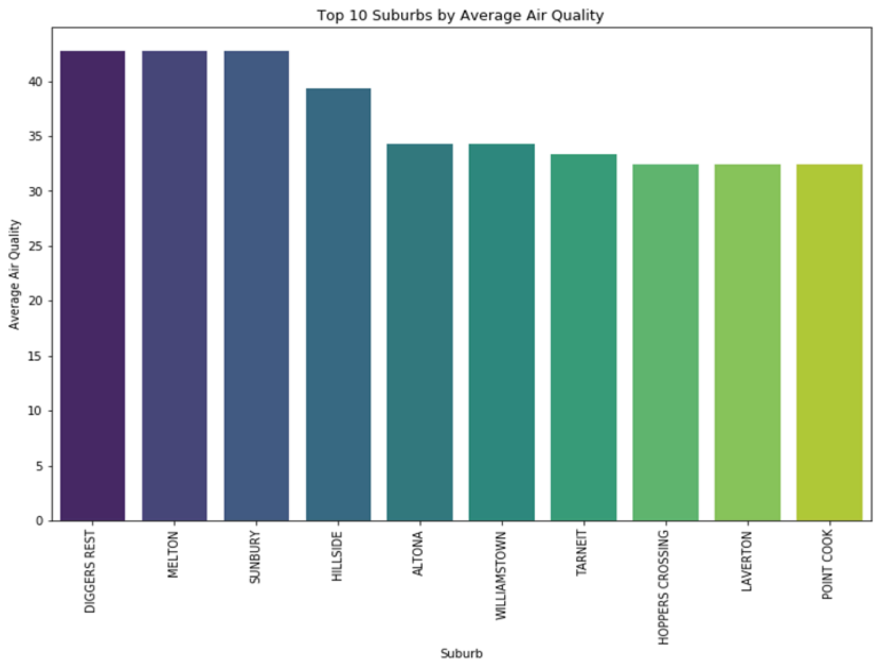


Figure 10: *Top 10 Suburbs by Average Air Quality*

Upon examining Figures 9 and 10, it becomes evident that Melbourne's central region has relatively superior air quality. The top 10 areas with the best air quality are mostly located in this region. However, towards the east and west sides of Melbourne, the average air quality is not as good, likely influenced by industrial activities, heavy traffic, and geographical factors.

Overall, the analysis highlights the importance of considering the number of childcare centres and air quality when choosing a suitable residential area for families with children. By focusing on areas with higher air quality, families can ensure that children live in a healthier environment, which is essential for their health and development. It is hoped that this analysis will help users make informed decisions about where to live and access childcare services, ultimately leading to better health outcomes and quality of life for Melbourne families.

5. Error & Challenge Handling

In the process of designing and implementing this system, many difficulties arose one by one continuously, deployment challenges were particularly significant among them. What the Fission framework initially brought was not its convenience, but its difficult initialization and various connection errors that kept appearing like a *nightmare*. To address the complex error situations, we designed several simple but important test scripts to assist or monitor the deployment of functions on the cloud through Fission:

- fissionRouteTest.sh: The working status of the Fission HTTP routes created by automated testing;
- put_database.py : Local tests can be conducted for manual data entry into the database, and the mappings returned by the database can be obtained;
- currentapp.logger: In the Fission built-in Flask runtime, the logger can be called to check the current running status of the function and return status codes. These status codes can help developers understand the current state of function execution;
- .yaml: Structured deployment with YAML files, where each package has a corresponding YAML file to reconfigure the deployment through YAML updates when the code is updated;
- compress_function.sh: Automated Package zip, scripted to package to the corresponding location, facilitating path configuration in YAML files.

These scripts would help the development be more controllable and visible, thus helping developers quicker to solve the problems.

6. Teamwork Collaboration

Backend developers:

- Gaoyuan Hao: Kubenetes and Fission Structure, Functionality Deployment, Testing
- Jinuo Sun: Elasticsearch Database, Report System Design

Frontend developers:

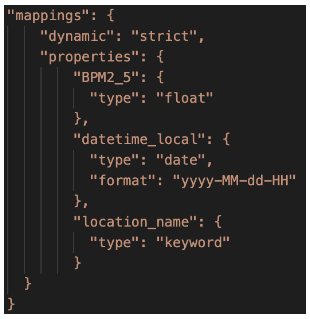
- Yulin Dong: Frontend Jupyter Notebook Application, Data Analysis, Report Data Analysis

Data Analyst:

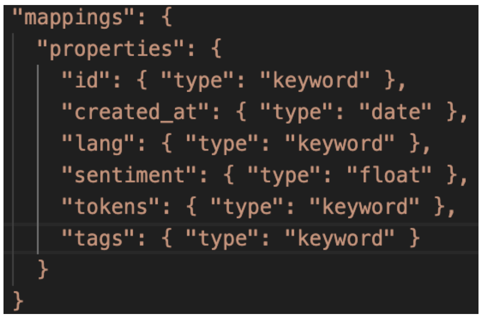
- Jin Wang: Data Collection and Analysis, Report Data Analysis
- Zihan Xu: Data Collection and Analysis, Report Data Analysis

Appendix

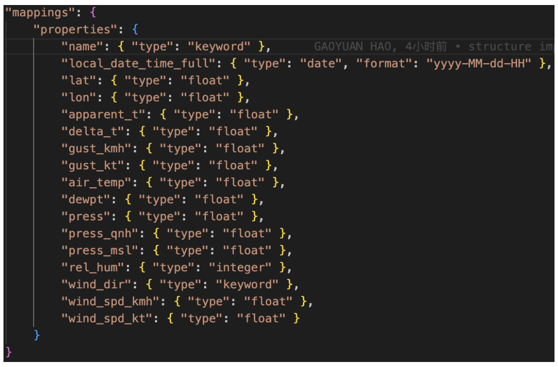
1. air_quality, madstone, weather_condition, child_care database mappings:



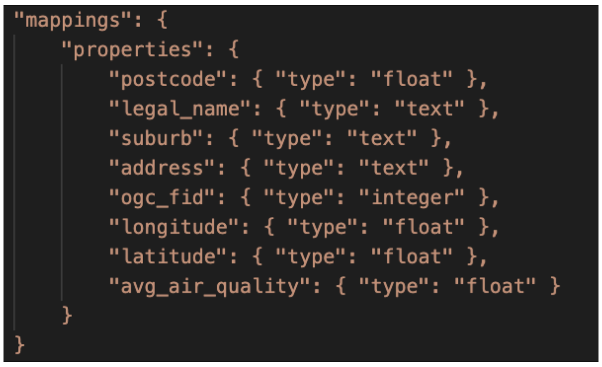
```
"mappings": {
    "dynamic": "strict",
    "properties": {
        "BPM2_5": {
            "type": "float"
        },
        "datetime_local": {
            "type": "date",
            "format": "yyyy-MM-dd-HH"
        },
        "location_name": {
            "type": "keyword"
        }
    }
}
```



```
"mappings": {
  "properties": {
    "id": { "type": "keyword" },
    "created_at": { "type": "date" },
    "lang": { "type": "keyword" },
    "sentiment": { "type": "float" },
    "tokens": { "type": "keyword" },
    "tags": { "type": "keyword" }
  }
}
```

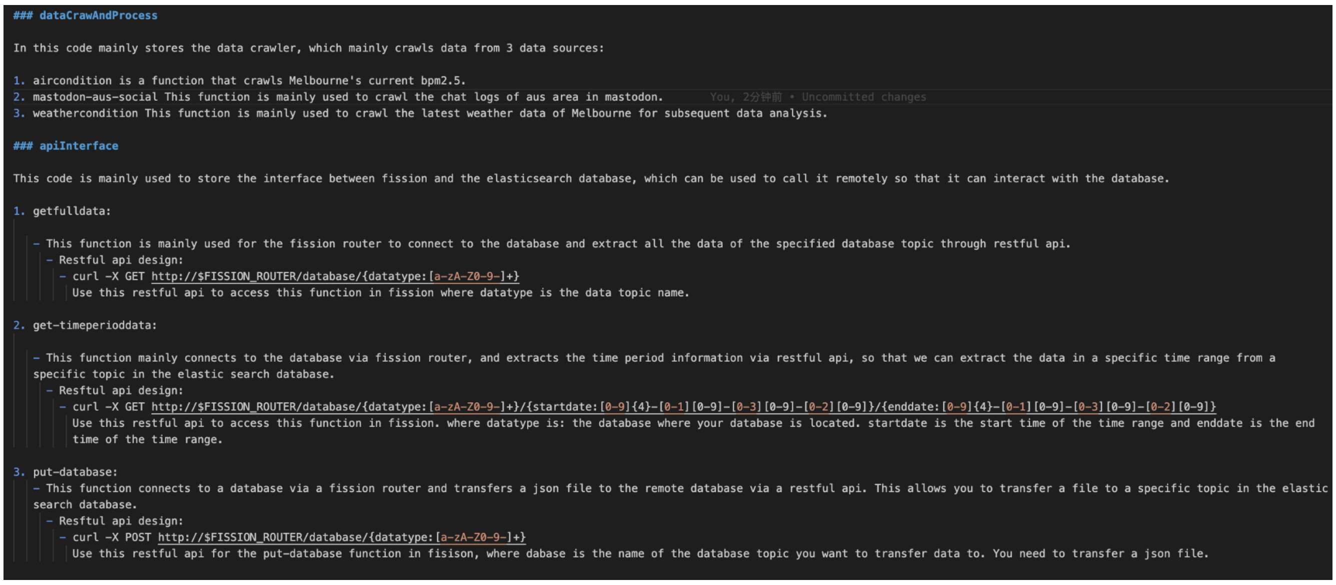


```
"mappings": {
    "properties": {
        "name": { "type": "keyword" },        GAOYUAN HAO, 4小时前 • structure im
        "local_date_time_full": { "type": "date", "format": "yyyy-MM-dd-HH" },
        "lat": { "type": "float" },
        "lon": { "type": "float" },
        "apparent_t": { "type": "float" },
        "delta_t": { "type": "float" },
        "gust_kmh": { "type": "float" },
        "gust_kt": { "type": "float" },
        "air_temp": { "type": "float" },
        "dewpt": { "type": "float" },
        "press": { "type": "float" },
        "press_qnh": { "type": "float" },
        "press_msl": { "type": "float" },
        "rel_hum": { "type": "integer" },
        "wind_dir": { "type": "keyword" },
        "wind_spd_kmh": { "type": "float" },
        "wind_spd_kt": { "type": "float" }
    }
}
```



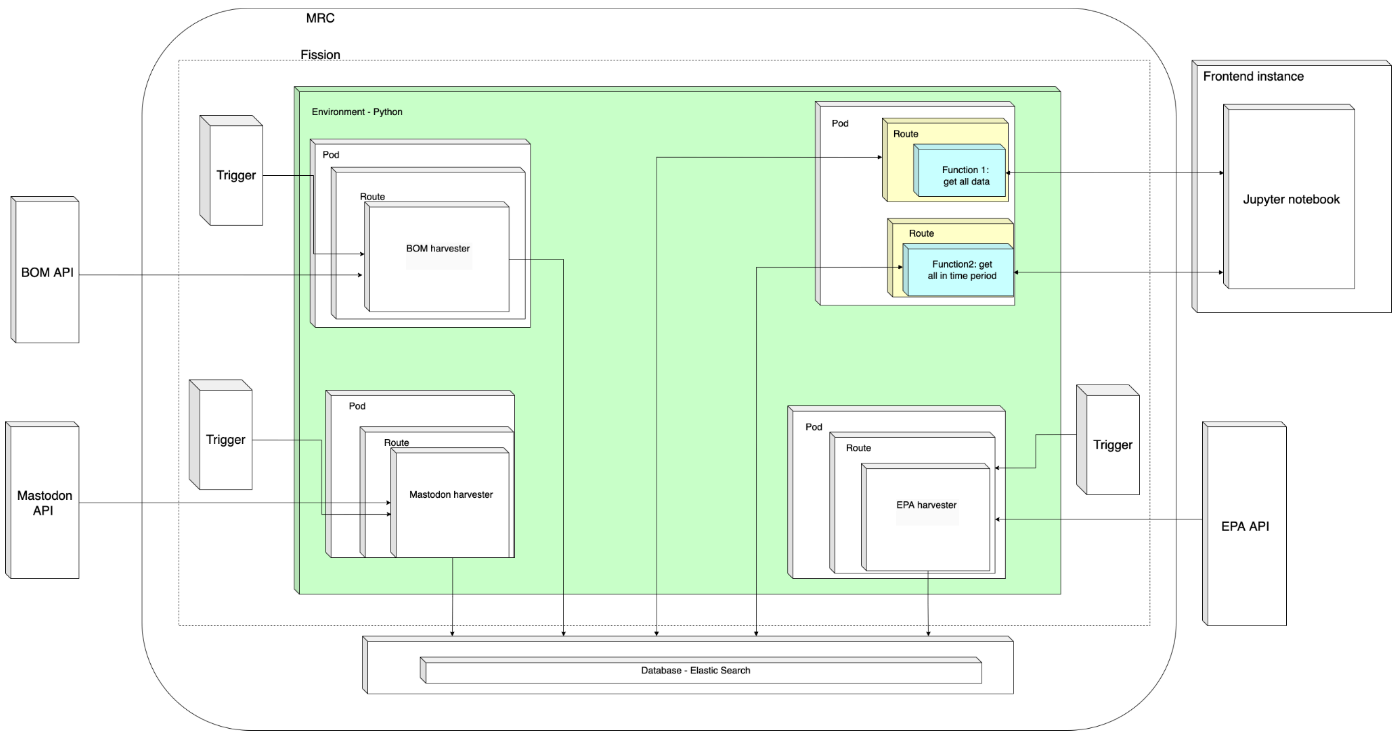
```
"mappings": {
    "properties": {
        "postcode": { "type": "float" },
        "legal_name": { "type": "text" },
        "suburb": { "type": "text" },
        "address": { "type": "text" },
        "ogc_fid": { "type": "integer" },
        "longitude": { "type": "float" },
        "latitude": { "type": "float" },
        "avg_air_quality": { "type": "float" }
    }
}
```

2. Fission README.md, with RESTful API design:

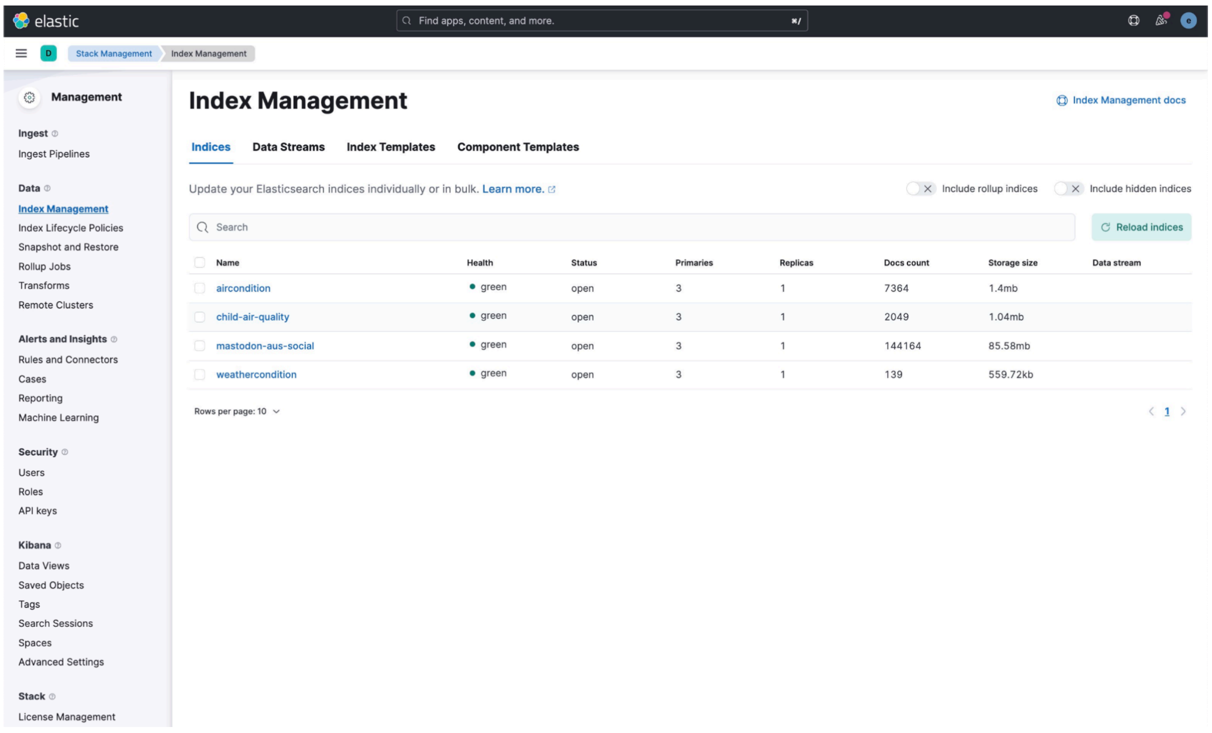


```
### dataCrawAndProcess

In this code mainly stores the data crawler, which mainly crawls data from 3 data sources:

1. aircondition is a function that crawls Melbourne's current bpm2.5.
2. mastodon-aus-social This function is mainly used to crawl the chat logs of aus area in mastodon.       You, 2分钟前 • Uncommitted changes
3. weathercondition This function is mainly used to crawl the latest weather data of Melbourne for subsequent data analysis.

### apiInterface

This code is mainly used to store the interface between fission and the elasticsearch database, which can be used to call it remotely so that it can interact with the database.

1. getfulldata:

   - This function is mainly used for the fission router to connect to the database and extract all the data of the specified database topic through restful api.
     - Restful api design:
       - curl -X GET http://$FISSION_ROUTER/database/{datatype:[a-zA-Z0-9-]+}
         Use this restful api to access this function in fission where datatype is the data topic name.

2. get-timeperioddata:

   - This function mainly connects to the database via fission router, and extracts the time period information via restful api, so that we can extract the data in a specific time range from a
   specific topic in the elastic search database.
     - Resftul api design:
       - curl -X GET http://$FISSION_ROUTER/database/{datatype:[a-zA-Z0-9-]+}/{startdate:[0-9]{4}-[0-1][0-9]-[0-3][0-9]-[0-2][0-9]}/{enddate:[0-9]{4}-[0-1][0-9]-[0-3][0-9]-[0-2][0-9]}
         Use this restful api to access this function in fission. where datatype is: the database where your database is located. startdate is the start time of the time range and enddate is the end
         time of the time range.

3. put-database:
   - This function connects to a database via a fission router and transfers a json file to the remote database via a restful api. This allows you to transfer a file to a specific topic in the elastic
   search database.
     - Resftul api design:
       - curl -X POST http://$FISSION_ROUTER/database/{datatype:[a-zA-Z0-9-]+}
         Use this restful api for the put-database function in fisison, where dabase is the name of the database topic you want to transfer data to. You need to transfer a json file.
```

3. System Architecture



4. Elasticsearch DBMS portal:



Index Management

Update your Elasticsearch indices individually or in bulk. Learn more.

Name	Health	Status	Primaries	Replicas	Docs count	Storage size	Data stream
aircondition	● green	open	3	1	7364	1.4mb	
child-air-quality	● green	open	3	1	2049	1.04mb	
mastodon-aus-social	● green	open	3	1	144164	85.58mb	
weathercondition	● green	open	3	1	139	559.72kb	

5. Github Code Repository:

https://github.com/BlueTriangleG/CloudComputing_HumanDensityVSweather.git

6. Youtube Video Demonstration:

https://youtu.be/ZOfMYno1ULs