

The comprehensive study of big data analytics in human health and safety

Name: Can Wang, Kaisheng Su, Mingtao Yang, Zhihe Ping
Student number: 1176867, 1241049, 1527052, 1238760
Course: COMP90024 Cluster and Cloud Computing

May 21, 2024

Contents

1	Introduction	3
2	System Architecture	3
2.1	Cloud Cluster	3
2.1.1	Kubernetes	3
2.1.2	Fission functions	4
2.1.3	Elasticsearch	4
2.2	Front end	5
2.2.1	Folium	5
3	Detailed Implementation	5
3.1	Data collection	5
3.1.1	Data collected by using API	5
3.1.2	Data collected statically	7
3.2	Data Storage	7
3.3	Data processing	8
4	Data Analysis	9
4.1	The impact of air pollution on the mortality of respiratory system diseases and lung cancer	9
4.2	Air Pollution vs Respiratory System Disease Death Ratio	10
4.3	Air Pollution vs Lung Cancer Death Ratio	11
4.4	The impact of sentiment score on suicide rates	12
4.5	The impact of weather on traffic-injury	17
5	Error handling	17
6	Discussions	18
6.1	Melbourne research cloud	18
6.1.1	Pros	18
6.1.2	Cons	18
6.2	Elastic Search	18
6.2.1	Pros:	18
6.2.2	Cons	19
6.3	Kubernetes and Fission	19
6.3.1	Pros	19
6.3.2	Cons	20
7	Limitations and improvements	20
8	Team Collaboration	20
9	Conclusion	21

1 Introduction

In the modern development and delivery of cloud software systems, the use of Function as a Service and RESTful API has become increasingly significant. These technologies, along with Kubernetes, play a crucial role in modern software development due to their scalability, flexibility and efficiency.

In this assignment, our team leverages these advanced technologies to conduct a cloud analysis project.

The project utilised the use of diverse data sets from SUDO (Spatial Urban Data Observatory (SUDO)), and data from other external providers such as the Environment Protection Agency (EPA), and the Bureau of Meteorology (BOM). By using Kubernetes, Fission, and ElasticSearch, we aim to efficiently collect, store, and analyze these data sets.

In contemporary society, public health is always closely associated with the environment, weather conditions and emotional well-being. With the rapid development of industry and manufacturing, air pollution has long been recognized as a significant social public health issue. Pollutants such as fine particulate matter (PM2.5), nitrogen dioxide (NO2), and sulphur dioxide (SO2) are increasingly linked to the growth of mortality from respiratory diseases and lung cancer. The Earth's climate environment has also been tremendously affected by the greenhouse gases produced by human-being activities. The resulting extreme weather also greatly increases the risk of human beings when going out. Emotional well-being is also an important determinant of health. There is an increasing number of people suffering from psychological problems or mental illness, especially teenagers and the young. Suicide and self-injuries caused by various mental illnesses have become a social problem that cannot be ignored. This study aims to explore three specific correlations:

1. The impact of air pollution on the mortality of respiratory system diseases and lung cancer.
2. The impact of weather on the mortality of road traffic injuries.
3. The impact of sentiment value on the mortality of suicide and self-injuries.

2 System Architecture

The provider of Cloud Service is MRC (Melbourne Research Cloud), which offers free on-demand computing resources to the project.

Front End:

- Jupyter Notebooks sends requests to the Fission function via RESTful API.

Cluster:

- The fission-functions are triggered by the HTTP request.
- The Fission function can import data into Elasticsearch or retrieve data from it using RESTful API calls.
- Stores data received from the Fission function and provides data for queries sent by the Fission function.
- The fission functions process the data received by the elastic search

Front End:

- The result is processed to html via JupyterNoteBook.

2.1 Cloud Cluster

2.1.1 Kubernetes

Kubernetes is an open-source platform designed to automate the deployment, scaling and management of containerized applications. It provides a framework for orchestrating computing, networking, and storage infrastructure on behalf of user workloads. Kubernetes allows users to efficiently manage and scale their applications by running them in containers across a cluster of machines (nodes).

Pods are the smallest deployable units in Kubernetes. A Pod can contain one or more containers (usually Docker containers) that are tightly coupled and share resources.

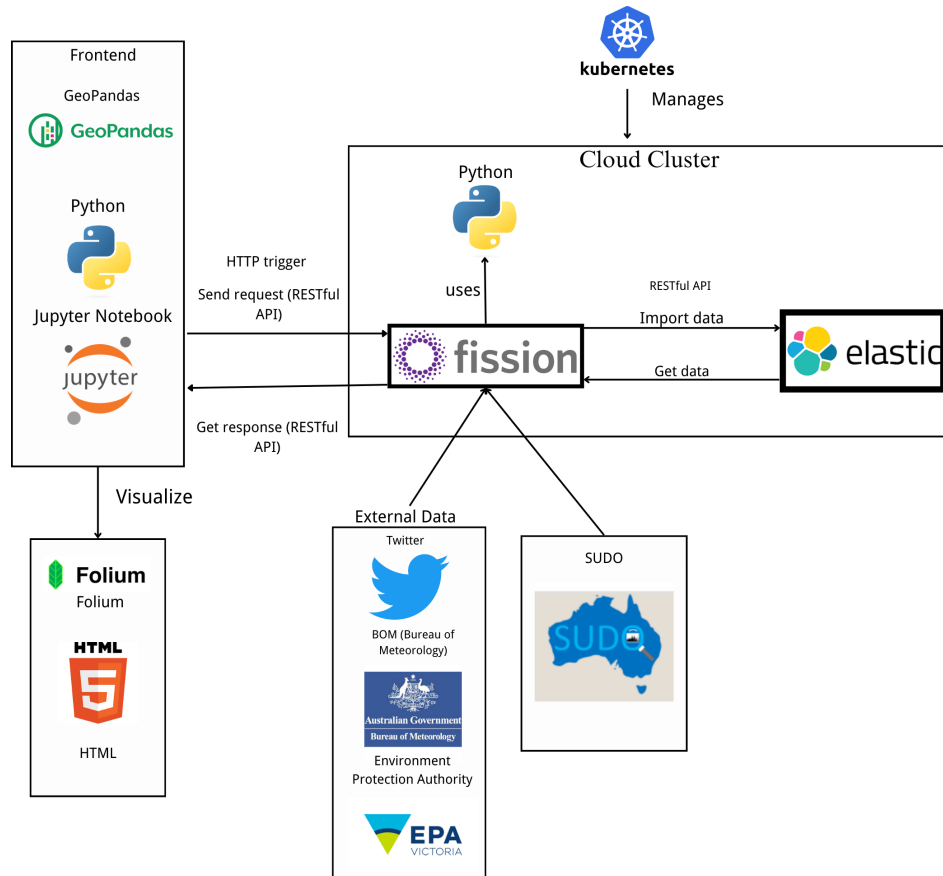


Figure 1: Technology architecture of the project

2.1.2 Fission functions

Fission is a serverless framework for Kubernetes that lets you run functions in response to events without managing servers. The fission functions are deployed by creating Python environments. Each fission function is allocated in a docker container with necessary runtime, and then deploy functions within their environment. When an event triggers a function, Fission's controller schedules a pod to run the function, scaling automatically based on demand.

The fission instance is capable for handling the HTTP request and parsing the content by leveraging Flask, which is a lightweight Python web framework used to build web applications and APIs. Using Flask to handle HTTP requests in a cloud-based Fission function is beneficial because it simplifies routing, request handling, and response generation, allowing for rapid development and easy deployment of microservices.

2.1.3 Elasticsearch

Elastic search is a distributed, RESTful search and analytic engine, which is capable of solving large-scale data sets. Elastic search uses a RESTful API, allowing users to interact with HTTP requests and standard HTTP methods (GET, POST, PUT, DELETE)

Port forwarding to direct get request from localhost:

By using: https://127.0.0.1:9200/weathers/_search

Here is the Kubernetes architecture:

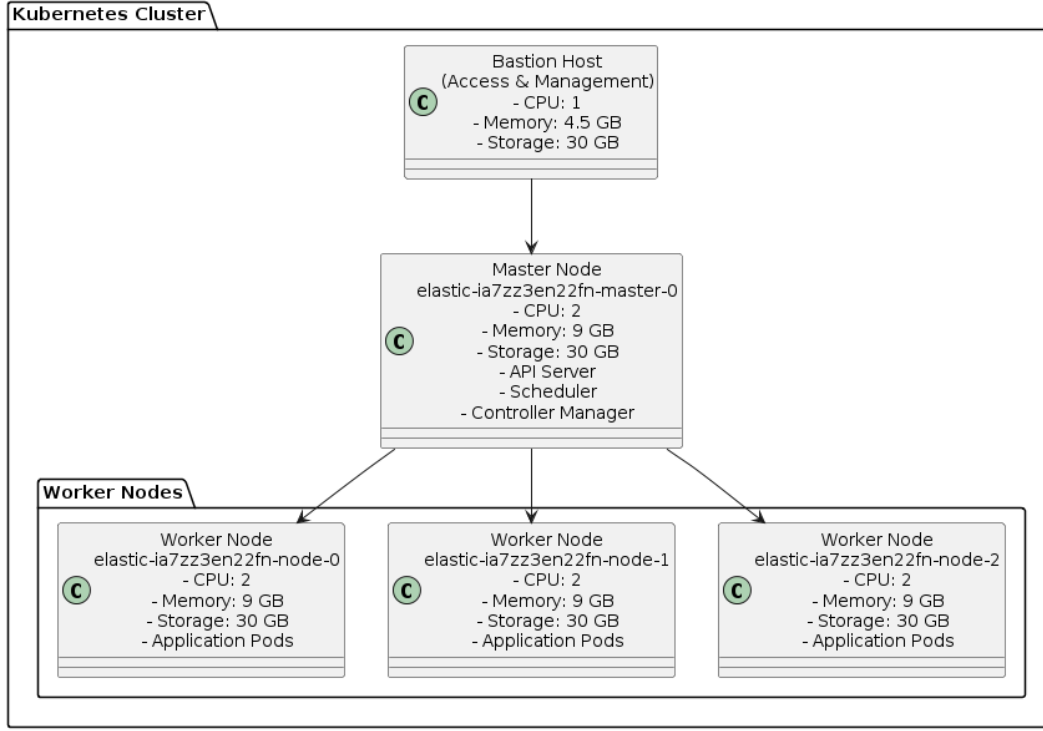


Figure 2: Kubernetes Node

2.2 Front end

2.2.1 Folium

For the Front End visualization, the Folium is utilized. Folium is a powerful Python library that is used to create interactive maps. It builds on the Leaflet.js library, which is a JavaScript library for user-friendly integrative maps. Folium allows for easy integration of data visualization and geographical plotting in Python. The Folium generate the HTML file from the Front end for the visualized map.

3 Detailed Implementation

3.1 Data collection

3.1.1 Data collected by using API

The data harvesters using API are all written in Python, and deployed as fission functions in Kubernetes.

EPA Air quality harvesters: The EPA Air quality harvesters implemented a process to retrieve air quality data from the Environment Protection Authority Victoria (EPA Victoria) and store it in Elasticsearch with RESTful API.

To avoid hardcoding the API key in the application and ensure the security of the harvester application, the API key of EPA Victoria is stored in a config map deployed into Fission. The harvester application can access the API key by reading the content at the path `/configs/default/shared-data/EPA_KEY` in Fission. Additionally, the HTTP response with status code 403 Forbidden, which indicates that the server already knows the user's identity but the user still doesn't have access, will be received if the HTTP request does not contain a User-agent in the header. Thus, the config map also contains a User-agent to be configured in the HTTP request header to prevent the 403 Forbidden response.

In the first step of collecting air quality data from EPA Victoria, an air monitoring site harvester is deployed in fission to collect the site information. The code defines the API endpoint: `https://gateway.api.epa.vic.gov.au/environmentMonitoring/v1/sites` and uses an HTTP request with a header containing the API key and User-agent and the parameter 'environmentalSegment'

set to be 'air' to retrieve all the air monitoring site information in Victoria, including 36-digit unique IDs used to identify particular sites and site names etc. The major information used in further steps is the IDs of the air monitoring sites. After obtaining the station IDs using the site harvester function in fission, an air quality harvester application iterates through each site ID and requests the air quality parameter data for that air monitoring site over the past 48 hours with 1 hour time interval from the API using the endpoint

`https://gateway.api.epa.vic.gov.au/environmentMonitoring/v1/sites/siteID/parameters`, with siteID replaced by the site ID iterated. For each site, the script parses the response data, extracting the station name, geographic location, and time series readings of relevant parameters. The extracted data is organised into bulk requests and stored in the Elasticsearch index using the Elasticsearch client. During the data storage process, the script also includes error handling and exception handling to ensure the reliability and continuity of the data harvesting and storage process. It is highly possible to receive 404 Not Found responses, which indicates that there has been no data on the specific air monitoring site over the past 48 hours or that the site is closed. Overall, this script automates the efficient migration of data from an external API to an internal database, ensuring data timeliness and completeness.

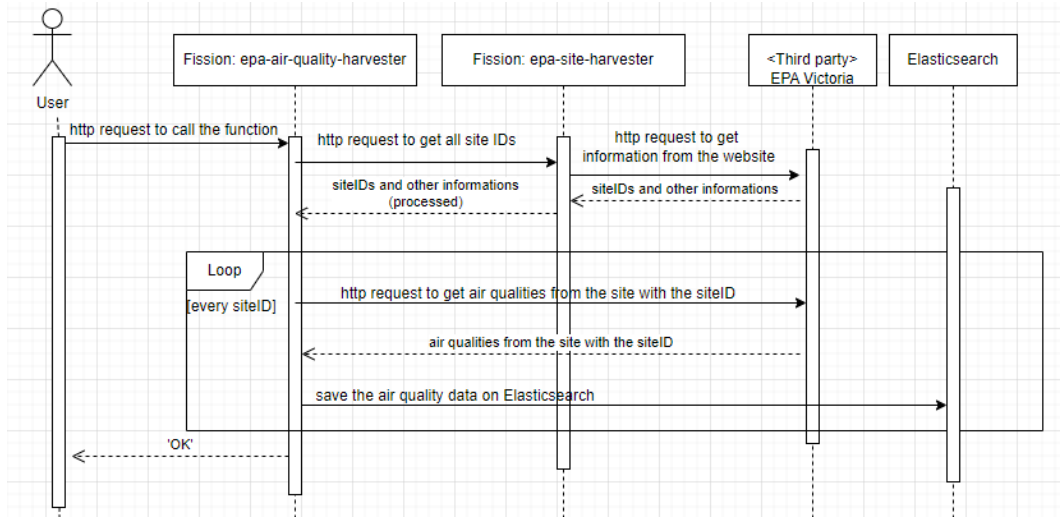


Figure 3: Sequence diagram of Air quality harvester

Weather harvesters:

The Victoria harvesters retrieve the real-time data from the BOM (Bureau of Meteorology) API for six observations in Victoria, which are: Melbourne Olympic Park, Geelong, Avalon, Melbourne Airport, Sk Kilda Harbour and Point Cook. Then, the harvesters will be sent to the fission inner router for further processing via the HTTP POST requests. The weather harvester module utilised a time trigger to periodically (every half an hour) send requests to the BOM API to obtain the latest weather data, which ensures the data is up-to-date.

The routers are connected to the data-processing module: add-weather using HTTP trigger. Each post message is designed as an event, once the specific router gets the post message, the data-processing module will be triggered and process the message automatically. The add-weather processor converts the extract json data and get the weather attributes like temperature, rain trace, wind direction and wind speed for further analysis.

Name	Method	Function	Authentication	URL Path
add-point-cook-weather	POST	add-vic-weather	false	/weathers/point-cook
add-sk-kilda-weather	POST	add-vic-weather	false	/weathers/sk-kilda-harbour
avalon	POST	add-vic-weather	false	/weathers/avalon
melb-airport	POST	add-vic-weather	false	/weathers/melb-airport

Table 1: Weather Endpoints

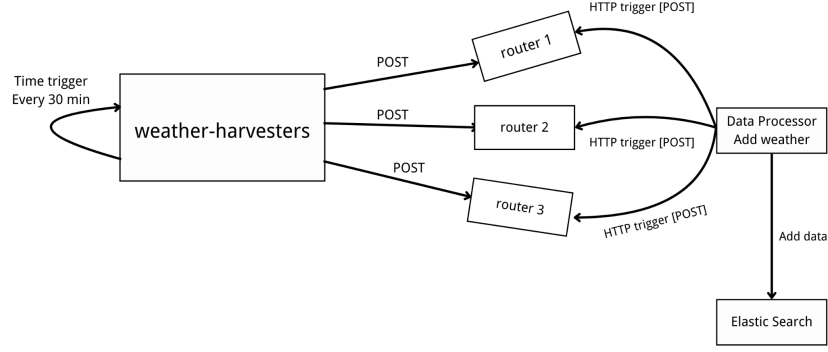


Figure 4: weather-harvesters diagram

3.1.2 Data collected statically

Part of used data is static data which are collected manually. For convenience, all manually collected data were uploaded onto the elastic search for further processing using the bulk API.

Twitter data:

The twitter data from Assignment 1 has been further processed to fit our topics needed for Assignment 2 using Spartan, and data uploaded onto elastic search using Bulk API. By using RESTful API, data can be easily obtained for further analysis and visualisation.

SUDO data:

The SUDO data from Spatial Urban Data Observatory has also been selected for specific discovery area and downloaded as shp file, which eventually uploaded onto elastic search as geojson file.

3.2 Data Storage

From harvester to elastic search The air quality data collected from EPA has the format of **siteID**, **siteName**, **geometry** and **parameters**, in which the **geometry** is a geo-point containing latitude and longitude and **parameters** have readings of different air quality indexes like PM2.5 and PM10 in 48-hour time series with 1 hour time interval. In order to reduce the complexity of data stored in Elasticsearch and improve the ease of querying data using KQL, the air quality data is imported to Elasticsearch in the index mapping with properties of **dataName**, **averageValue**, **siteID**, **siteName** since, until, **unit** and the **geometry**. **dataName** represents the air quality index including PM2.5, SO2 etc., and **averageValue** and **unit** indicates the exact measurement of the air quality index and its unit. **since** and **until** represents the start time and the end time of the measurement. **geometry** has the data type of geo-point containing latitude and longitude of the site.

The add-weather functions retrieve the data from the specific route of weather harvesters. These functions parse the JSON data for further processing. The Elasticsearch index contains the following fields: **stationid**, **timestamp**, and the geographic attributes **lat** (latitude coordinate) and **lon**(longitude coordinate). The **name** field represents the weather station, **air temp** stores the air temperature measurement, **rain trace** holds the rain trace measurement in millimeters (mm), **wind dir** indicates the wind direction, and **windspd kmh** represents the wind speed in kilometers per hour (km/h).

The premature mortality data utilised in this analysis is sourced from SUDO, covering the period from 2011 to 2015. This dataset includes fields such as **phn_code**, **phn_name**, **geometry**, **respiratory_system_disease_rate**, **respiratory_system_disease_deaths**, **lung_cancer_deaths**, **traffic_injury_rate**, **traffic_injury_deaths**, **suicide_rate**, and **suicide_deaths**, providing a comprehensive view of disease statistics across different regions. The data is stored in Elasticsearch, leveraging its powerful indexing and search capabilities. To store this data efficiently in Elasticsearch, a custom index mapping was defined. This mapping ensures that each field is appropriately typed and indexed for fast retrieval and analysis. For instance, fields such as **deth_suic**, **deth_lung**, **deth_res**, and **deth_rd** are defined as floats, while fields like **phn_code** and **phn_name** are stored as text with keyword subfields for efficient filtering and aggregation. The **geometry** field is stored as a **geo_shape** to support spatial queries. This setup allows for real-time querying and analysis, ensuring that large volumes of data can be processed efficiently. With Elasticsearch, we can perform complex queries and

aggregations, enabling in-depth analysis to uncover meaningful insights into death trends and patterns.

To store the twitter-data in Elasticsearch, we defined a mapping that ensures each field is appropriately typed for efficient storage and retrieval. When we ingest data, we specify the `bbox` field as a `geo_shape` to store and query geographical shapes. The `created_at` field is stored as a date to support time-based queries. Unique identifiers and language codes are stored in the `id` and `lang` fields as keyword types, which are optimized for exact matches. The location field is stored as text, making it suitable for full-text search. Sentiment scores are stored in the `sentiment` field as float, allowing for precise numerical analysis. By defining these field types in advance, we ensure that Elasticsearch can efficiently index and search the data, enabling fast and complex queries across large datasets.

3.3 Data processing

Aggregating different data together, from the backend (the elastic search to the front end) A function called `air-quality-death` is implemented to aggregate the datasets of the air qualities (`air-qualities`) and statistics for the deaths of people because of different causes (`premature_mortality_2011`). In the function, there are two KQL queries that search data from Elasticsearch. The first query is responsible for searching all the death ratios of respiratory system disease and lung cancer in different Public Health Network (PHN) areas from Elasticsearch, as well as the PHN code, and PHN name. The 'id' of each document storing the data in `premature_mortality_2011` is also collected in the first query for further use in the second query. Then, the ids of documents are iterated to cluster air quality data by checking whether geo-points of the air quality data are contained in the geo-shape of the PHN area. Finally, the clustered air quality data is aggregated with the death ratio data and the fission function returns a response in the JSON format containing a status code of the response and data with the PHN code, PHN name, two death ratios and lists of air quality index measurements ordered by time.

For static data of twitter data from assignment 1 and SUDO data, data is pre-processed before taking the action of upload. For instance, for twitter data used in assignment 2, students have further simplified the dataset, which reduced the pressure for data storage for elastic search. To achieve this, specific data are selected and irrelevant data are filtered out. Data without a valid geographical location are filtered. As an output data file, the `json` file consists of users' `id`, `created_at`, the language of the twitter, the sentiment value, and the name, the latitude and longitude of the geographical location. Once the file has been processed, although the size of the file has been reduced significantly, it is still oversized for bulk upload. To address this issue, the file has been split into multiple subfiles, each subfile consists of 75000 lines of raw data, and a file uploaded using python script.

For historical weather data, those data are collected manually as well, while in contrast weather data in real-time are harvested automatically. Those historical data are stored month-by-month unsorted, by concatenating multiple files and sorting the data by date, the date of collected data can hence be used as an identifiable id for upload. The file is uploaded similarly as the twitter file without split, using the bulk API which provides convenience.

The suicide-data function is deployed on Fission to connect to an Elasticsearch cluster, retrieve, and process data related to suicide and death. The function first configures logging to monitor its execution process and records the start time. Using the provided URL and credentials, it establishes a connection with the Elasticsearch cluster. A search query is defined to retrieve documents from the `prediction_mortality_2011` index, requesting fields such as `_id`, `phn_name`, `dth_suic96`, `dth_suic99`, and `geometry`, and matching all documents. The function executes the query, records the time spent, and stores the results in a variable. It then traverses the results, extracting relevant fields from each document. For documents with valid geometric data, it processes and structures the information into a feature list containing document ID, suicide count, suicide rate, public health network name, and geometric data. Finally, the function creates a response dictionary with the status code and processed data, logs the total execution time, and returns the response as a JSON string. If an error occurs, it logs the error message and returns a JSON string with the error message and a 500 status code.

The twitter-data function is deployed on Fission to connect to the Elasticsearch cluster, retrieve, and process geolocation-related Twitter data. It configures logging and records the start time. The function establishes a connection with the Elasticsearch cluster using the provided URL and credentials. A search query is defined to retrieve documents from the `geo_twitter_data` index, matching all documents. The function executes the query, logs the time taken, and stores the results in a variable. It traverses the results, extracting relevant fields, especially focusing on bounding box (`bbox`) data. If `bbox` data is

valid, it calculates the centroid and converts it into point geometry. The function extracts fields such as tweet_id, created_at, lang, location, and sentiment from the Twitter data, structuring them into a list. The function constructs GeoJSON data, converting each Twitter data item into GeoJSON features and gathering these into a GeoJSON feature collection. Finally, it creates a response dictionary with the status code and processed GeoJSON data, logs the total execution time, and returns the response as a JSON string. If an error occurs, it logs the error message and returns a JSON string with the error message and a 500 status code.

The traffic-inquiry function is deployed on Fission to connect to the Elasticsearch cluster, retrieve, and process data related to traffic injuries. It configures logging and records the start time. Using the provided URL and credentials, it establishes a connection with the Elasticsearch cluster. A search query is defined to retrieve documents from the prediction_mortality_2011 index, requesting fields such as _id, phn_name, dth_rd_t80, dth_rd_t83, and geometry, and matching all documents. The function executes the query, logs the time taken, and stores the results in a variable. It traverses the results, extracting relevant fields from each document. For documents with valid geometric data, it processes and structures the information into a feature list containing document ID, traffic injury count, traffic injury ratio, public health network name, and geometric data. Finally, the function creates a response dictionary with the status code and processed data, logs the total execution time, and returns the response as a JSON string. If an error occurs, it logs the error message and returns a JSON string with the error message and a 500 status code.

The weather-data we obtained from the BOM is pretty good, so we did not process the data on the fission and directly visualized it on the front-end.

4 Data Analysis

4.1 The impact of air pollution on the mortality of respiratory system diseases and lung cancer

phn_code	phn_name	Particles_mean	PM2.5_mean	PM10_mean	Visibility_mean	API_mean	O3_mean	NO2_mean	SO2_mean	CO_mean
PHN201	North Western Melbourne	8.508250	10.296638	20.256413	64.932862	0.821630	7.940000	12.043478	0.426739	0.503478
PHN202	Eastern Melbourne	8.253768	11.012500	14.584565	61.843696	0.788913	NaN	NaN	NaN	NaN
PHN203	South Eastern Melbourne	NaN	10.972283	14.195217	64.876739	0.708913	8.186522	NaN	NaN	NaN
PHN204	Gippsland	7.247003	3.952500	7.898087	66.426957	0.429638	15.774783	4.022065	0.419130	0.120000
PHN205	Murray	7.560605	4.347283	11.367935	NaN	NaN	NaN	NaN	NaN	NaN
PHN206	Western Victoria	9.863231	5.971739	16.650000	63.855000	0.518478	4.860435	7.891304	0.166304	0.143696

Figure 5: The mean values of air quality indexes in different PHN areas

Firstly, the mean values of air quality indexes in different PHN areas were calculated. It is obvious that there was severe data missing in the air quality dataset(see figure 5). Murray had the most serious data-missing situation, with only three air quality index measurements existing in the dataset. A possible explanation for this situation is that the time span of the dataset is extremely short. Since EPA Victoria only provides air monitoring data over the last 48 hours, there is a high probability that the harvested data will be missing due to temporary maintenance or shutdown of the air monitoring sites. The lack of data is likely to influence the result of the analysis on the impact of air pollution on mortality of respiratory system diseases and lung cancer, especially for the evaluation of air quality index measurements with the largest amount of missing data.

Figure 6 visualizes the measurements of Particle degree and API in the 6 PHN area in Victoria. Particle refers to particulate matter, which are tiny particles in the air including PM2.5 and PM10. API is the Air Pollution Index, indicating the overall air quality situation. Although missing data reduces the readability of line graphs, some general trends can still be obtained. Most of the PHN areas have tremendous growth in the degree of particles and API in the middle of the time span. Overall, Western Victoria had the highest degree of particles in the dataset and Gippsland had the best degree of particles among the 6 PHN areas. Additionally, Gippsland had the lowest API in the 48 hours, representing that Gippsland had the best air quality in Victoria according to the dataset. Although there was a peak value of API in South Eastern Melbourne, the API value of Eastern Melbourne was overall highest during the 48 hours.

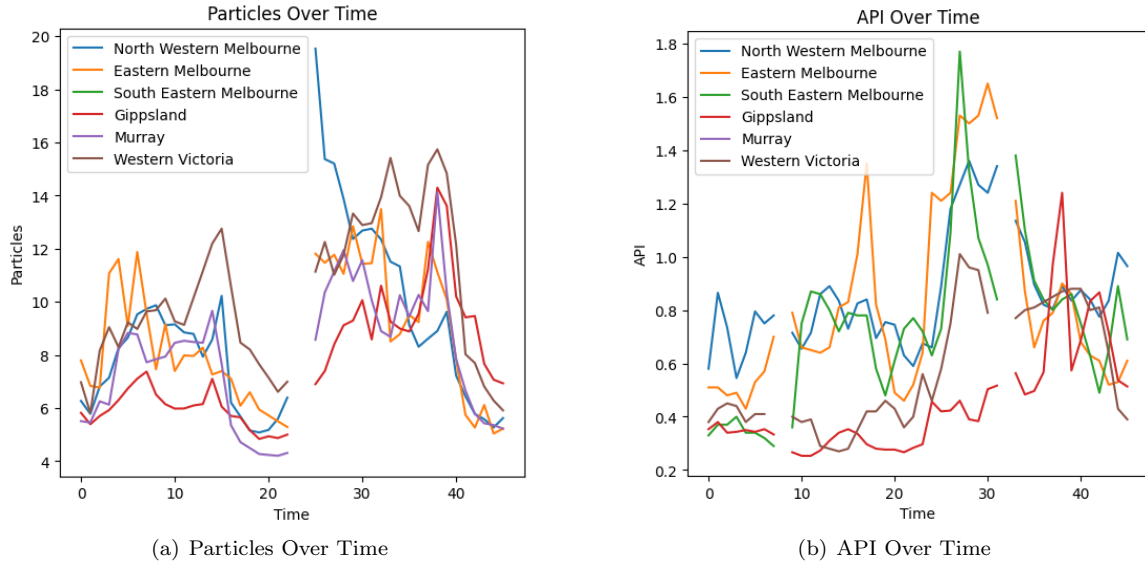


Figure 6: Figure of Particles and API Over Time

4.2 Air Pollution vs Respiratory System Disease Death Ratio

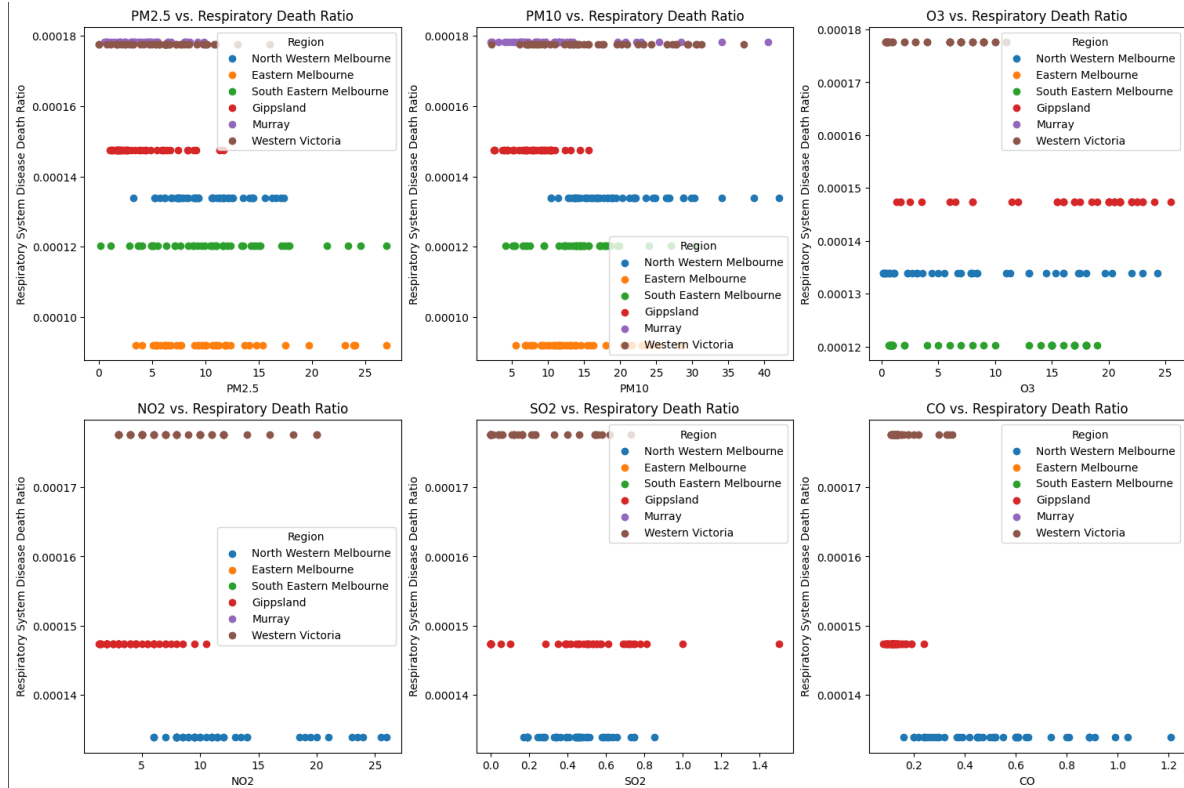


Figure 7: different air pollution versus respiratory system disease death ratio

Figure 7: Scatter plot of respiratory death ratio and degrees of different air pollutants According to Figure 7, CO is least associated with the respiratory disease death ratio among the 6 air pollutants because the degree of CO was low although the death ratio was high. The degrees of PM2.5, O3, NO2 and SO2 were in the median level when the death ratios were high, which indicates that these four air pollutants have some relationship with the death ratio of respiratory disease to some extent. PM10

has the largest correlation with the respiratory system disease death ratio based on the scatter plot. The measurements from Murray and Western Victoria illustrated the trend that the higher the degree of PM10, the higher the death ratio.

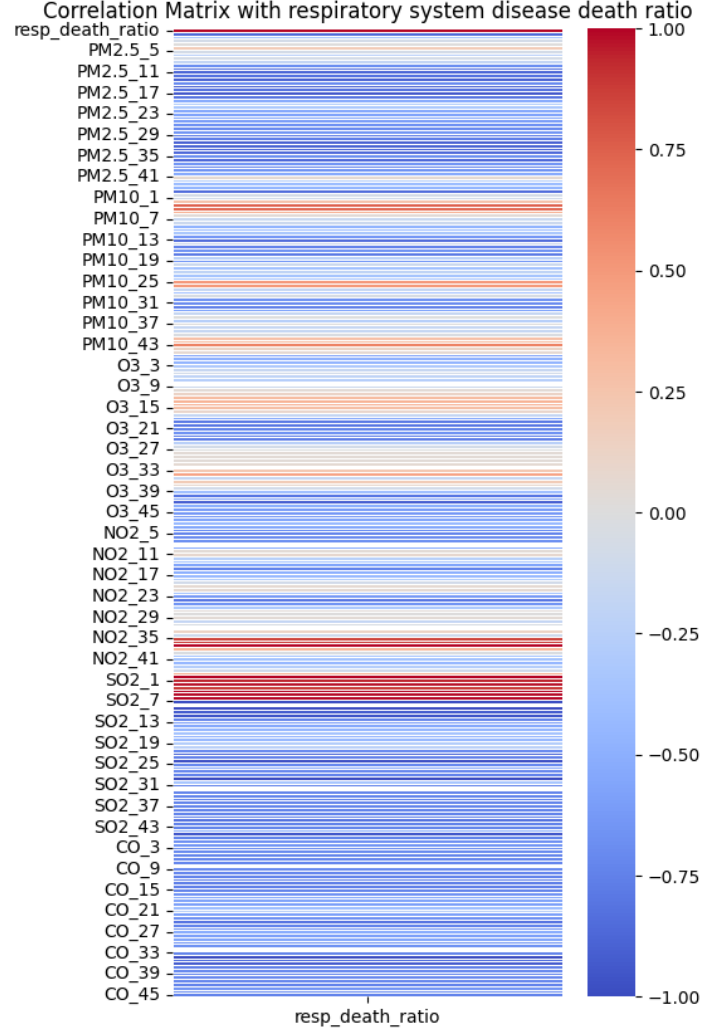


Figure 8: Correlation Matrix with respiratory system disease death ratio

This correlation matrix (see figure 8) demonstrates the degree of relationship between air pollutants and respiratory system disease death ratio, using Pearson correlation coefficient with the range from 1 to -1. According to the matrix, SO2 and NO2 have the highest positive correlations with the death ratio, followed by PM10. However, the result of correlation matrices diverges notably from that of scatter plot analysis. The most likely reason could be because the data missing in the measurements of SO2 and NO2 in the dataset, which leads to the biased result. Thus, it cannot be concluded that SO2 and NO2 are relative to the death ratio at the highest level. Since there are few missing data on PM10, PM10 can be considered to have a relatively large impact on the respiratory system disease death ratio.

4.3 Air Pollution vs Lung Cancer Death Ratio

Based on Figure 9, it is obvious that the measurement of O3 from Gippsland is relatively high and the proportion of lung cancer deaths is also high, which represents that there is a positive correlation between the degree of O3 and the lung cancer death ratio. Additionally, PM10 and SO2 are also positively related to the lung cancer death ratio. CO has the minimum influence on the lung cancer death ratio, followed by PM2.5 and NO2.

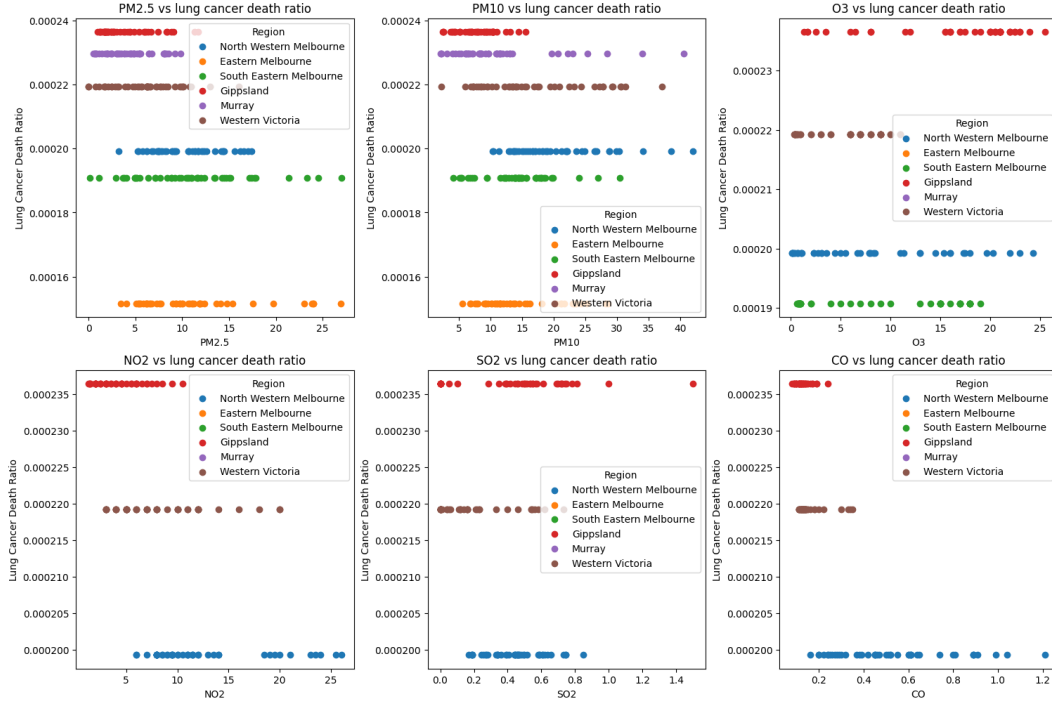


Figure 9: Different air pollution versus Lung Death

In this correlation matrix(see figure 10), O3 and SO2 have the highest correlation with lung cancer death ratio, with pearson correlation coefficient of around 0.75. PM10 is also relevant to the lung cancer death ratio in a positive degree because its pearson correlation coefficient is around 0.4. The conclusion obtain ed from this correlation matrix is relatively consistent with the conclusion obtained from the scatter plot. However, because of the data-missing situation in the measurements of SO2, the relationship between SO2 and the proportion of lung cancer deaths still needs further verification. It can be concluded that the degrees of O3 and PM10 are positive correlated with the lung cancer death ratio.

4.4 The impact of sentiment score on suicide rates

In this section, I will first analyze the sentiment score data, followed by an analysis of the suicide rate data. Then, I will combine both datasets to evaluate whether sentiment scores have a significant impact on suicide rates.

The figure 11 indicates that the distribution of data on the number of suicidal deaths is relatively concentrated, with the box ranging from approximately 300 to 570, indicating that the majority of the data is within this range and the median is around 330. There are a few extremely high values in the distribution of suicide mortality rate data, indicating that in certain regions, the suicide mortality rate is significantly higher than in other areas. Next, I will examine the distribution of the sentiment score and suicide rate data in more detail to better understand the underlying patterns and potential anomalies. By analyzing the density plots, we can gain further insights into the central tendencies, variability, and skewness of the data, which will be crucial for understanding the relationship between sentiment scores and suicide rates.

The Figure 12 indicates that the density plot reveals a bimodal distribution, suggesting that the data might be influenced by outliers, data grouping, or different underlying distributions. Given that the boxplots did not show significant outliers, it is plausible that the data come from two distinct groups or follow different patterns. To further investigate this, we will employ KMeans clustering analysis to identify potential clusters within the data, which can help us understand these patterns better. Through the figure 13, we can identify these clusters and better understand the different features in the data. This will provide information for our subsequent analysis of the relationship between sentiment scores and suicide rates, and provide more detailed insights into the potential impact of

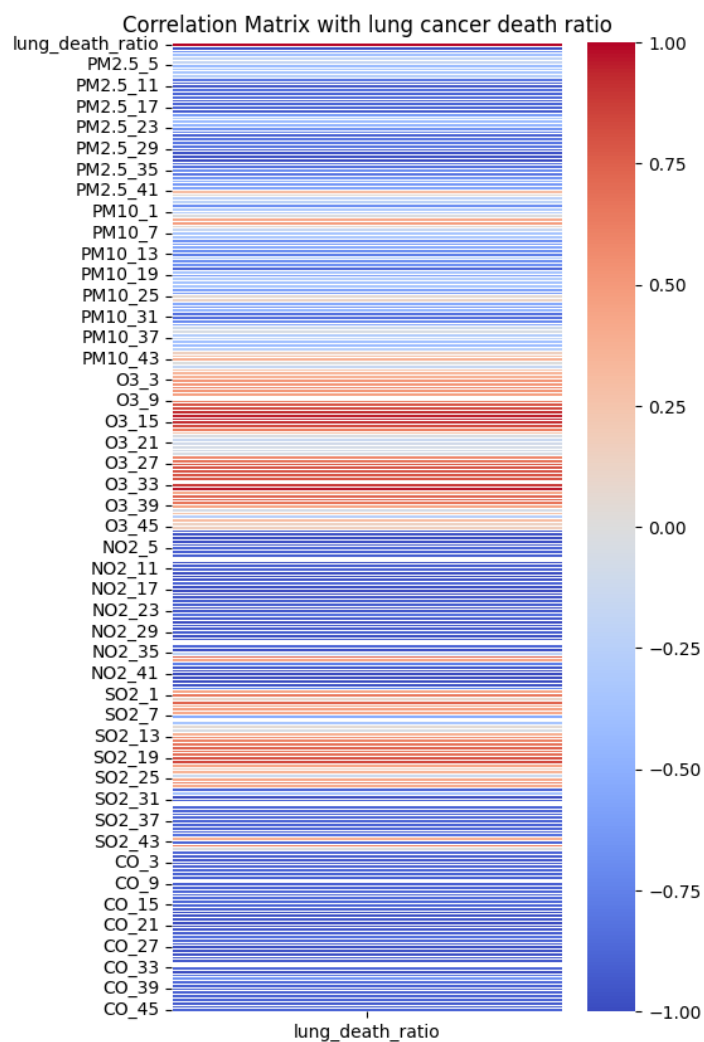


Figure 10: Correlation Matrix with Lung Death

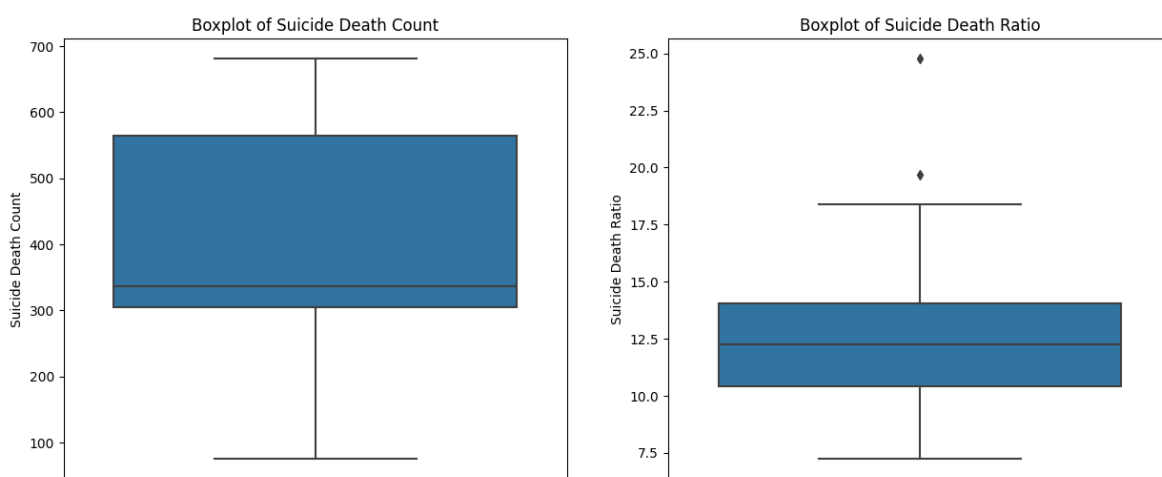


Figure 11: Box plot of Suicide Death Count and Suicide Ratio

sentiment states on suicide behavior.

The figure 13 shows that the eastern coast of Australia has higher suicide death counts, particularly

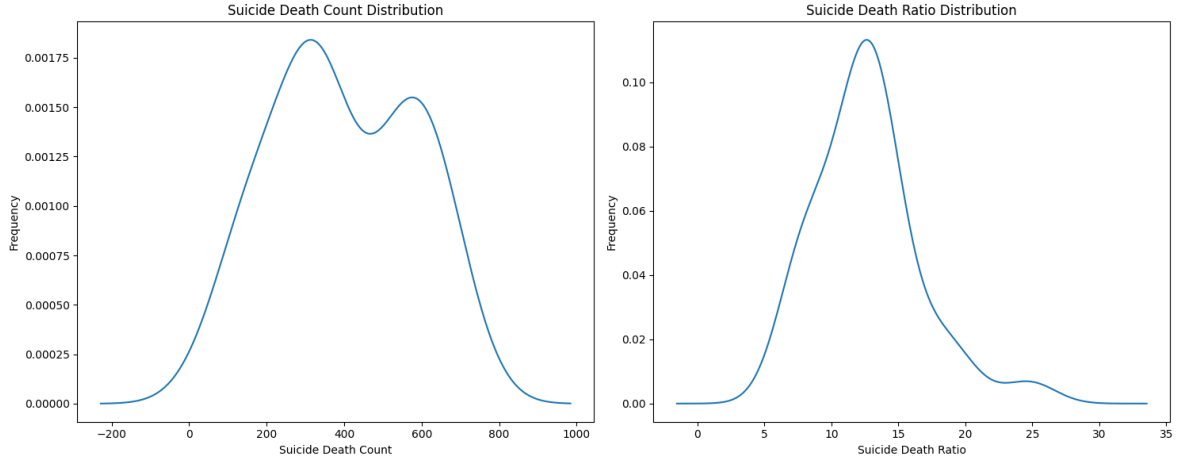


Figure 12: Suicide Death Count and Ration Distribution

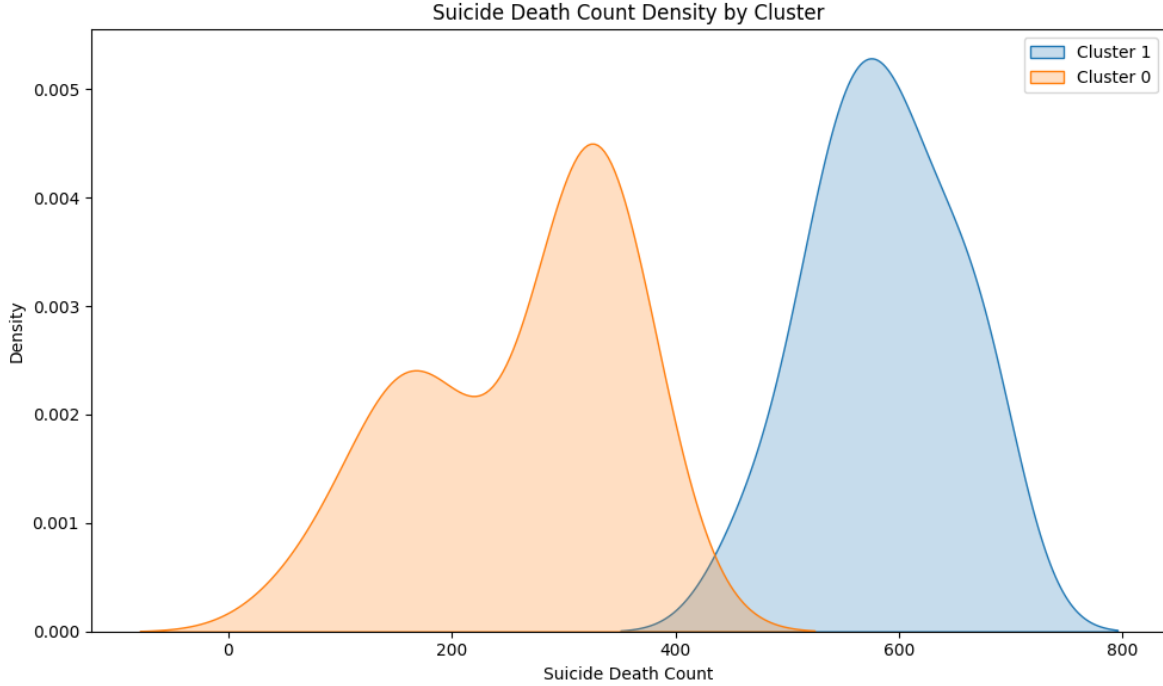


Figure 13: Suicide Death Count Density by Cluster

in parts of Queensland and New South Wales, where the darker colors indicate counts above 500. In contrast, the central region of Australia, including parts of South Australia and Western Australia, shows lower suicide death counts, indicated by lighter colors, with counts ranging from 100 to 300. The southern and western regions also have relatively high suicide death counts, shown by darker colors, although not as high as the eastern coast.

Having analyzed the geographical distribution of suicide death counts, we will now turn our attention to the sentiment score derived from Twitter. By examining the sentiment score of tweets, we can gain insights into the general sentiment of the population and investigate its potential correlation with suicide rates. The sentiment score data from Twitter will provide a complementary perspective to our understanding of the factors influencing suicide rates. The figure 15 shows that sentiment scores are mostly concentrated in neutral or near neutral positions, and there are some data points with extreme sentiments scores, but the number of these points is relatively small.

The figure 16 shows the geographical distribution of sentiment scores, with the majority of sentiment

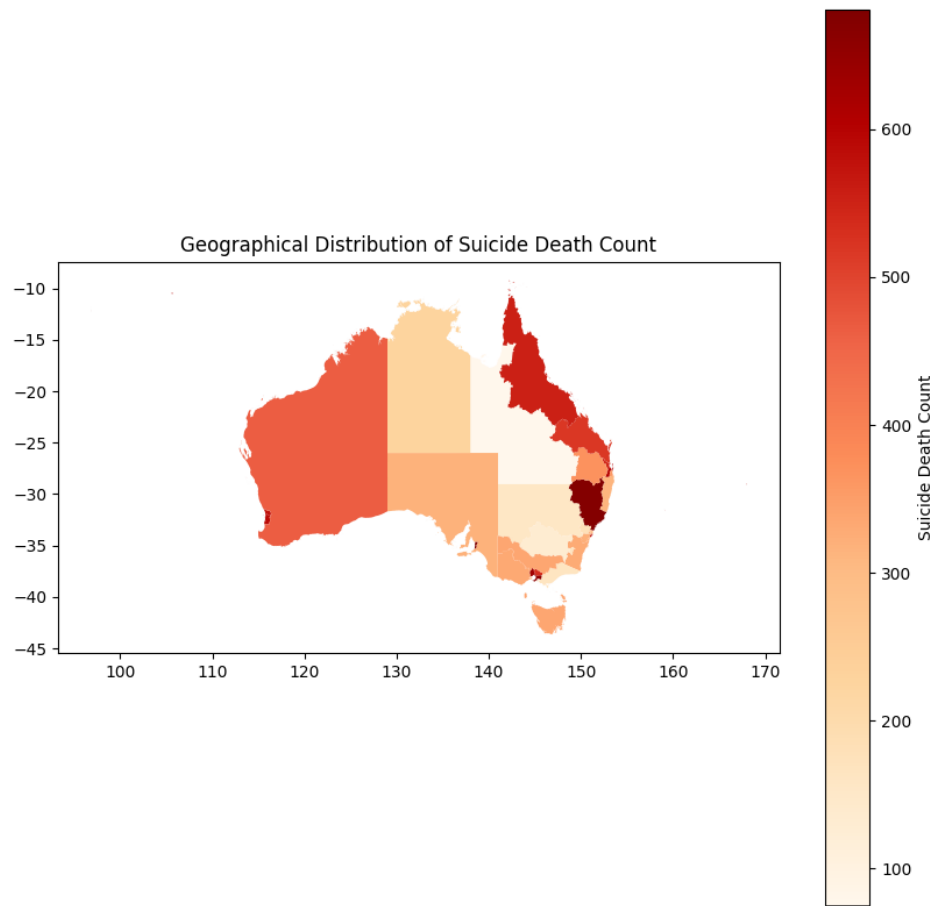


Figure 14: Geographical Distribution of Suicide Death Count

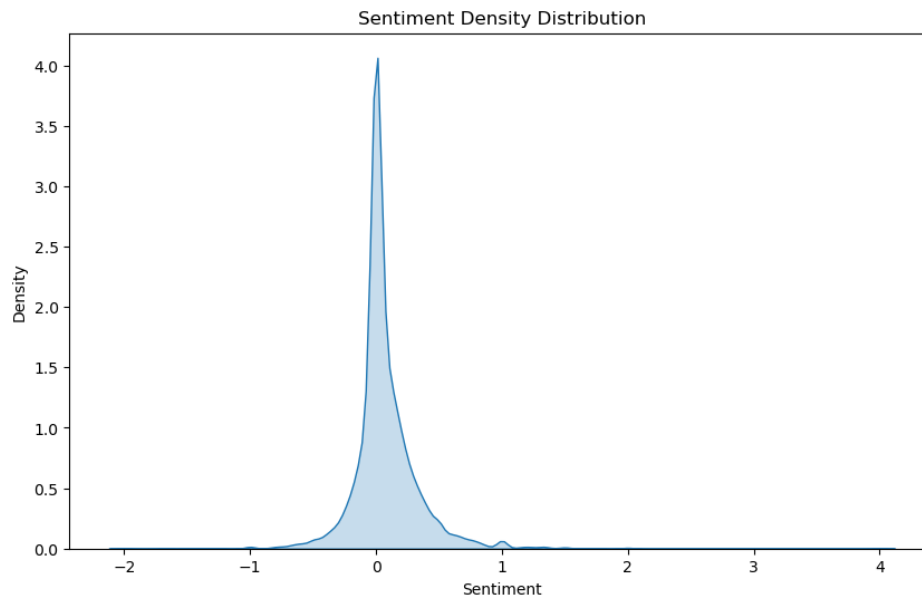


Figure 15: Sentiment Density Distribution

scores being neutral or close to neutral, mainly concentrated between 100 degrees east longitude and 180 degrees east longitude. There are relatively few data points for extreme sentiment scores, and

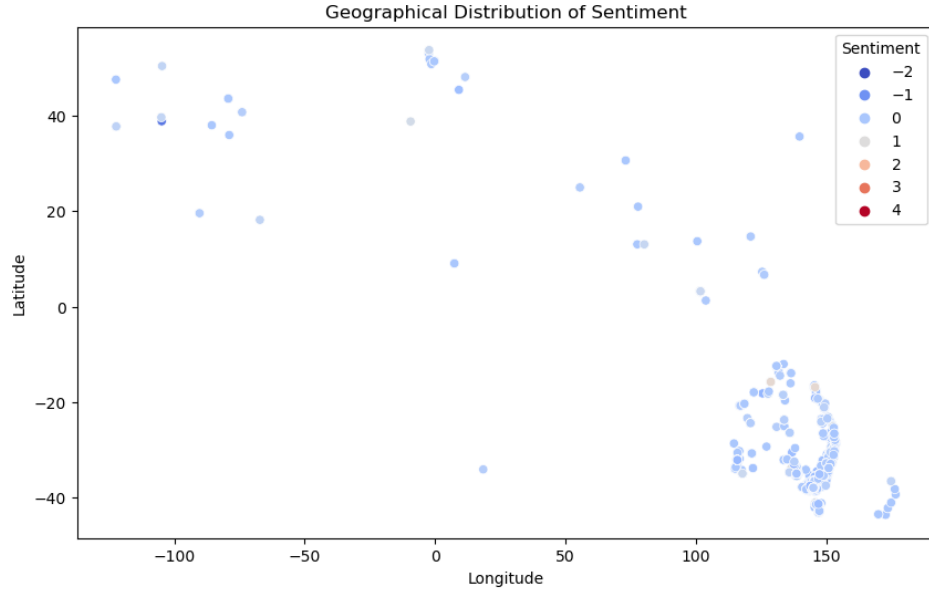


Figure 16: Geographical Distribution of Sentiment

their distribution is relatively scattered.

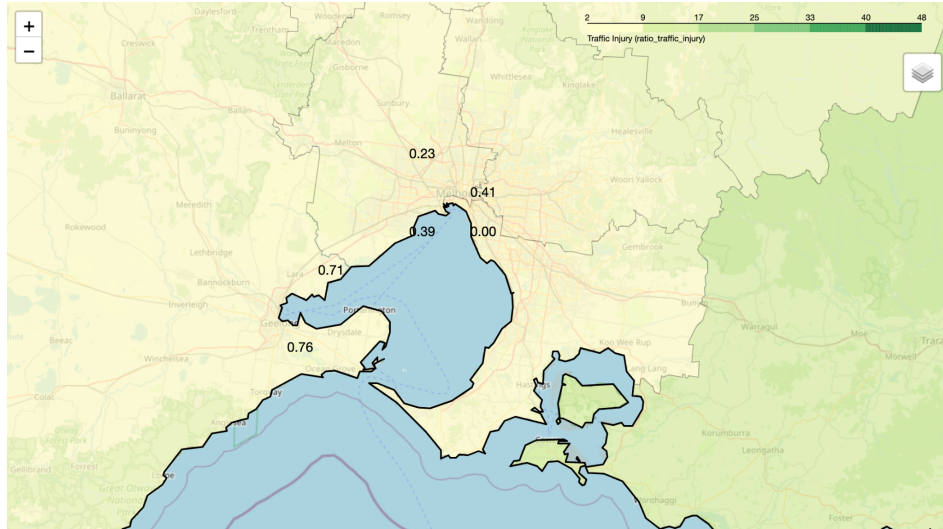


Figure 17: Injury Map

Having analyzed the geographical distribution of Sentiment, we will now turn our attention to combining sentiment score data with suicide rate data for a comprehensive analysis. By examining the sentiment score of social media posts, particularly from Twitter, we can gain insights into the general sentiment of the population. This sentiment data will then be correlated with suicide rates to explore any significant relationships between sentiment score and suicide behaviors. Through this integrated analysis, we aim to better understand the potential impact of sentiment scores on suicide rates. To investigate the linear relationship between sentiment scores and suicide rates, we calculate the Pearson correlation coefficient. The calculated value of -0.07 suggests a very weak linear relationship. The scatter plot with a fitted regression line confirms this finding, showing no clear linear trend between sentiment scores and suicide rates.

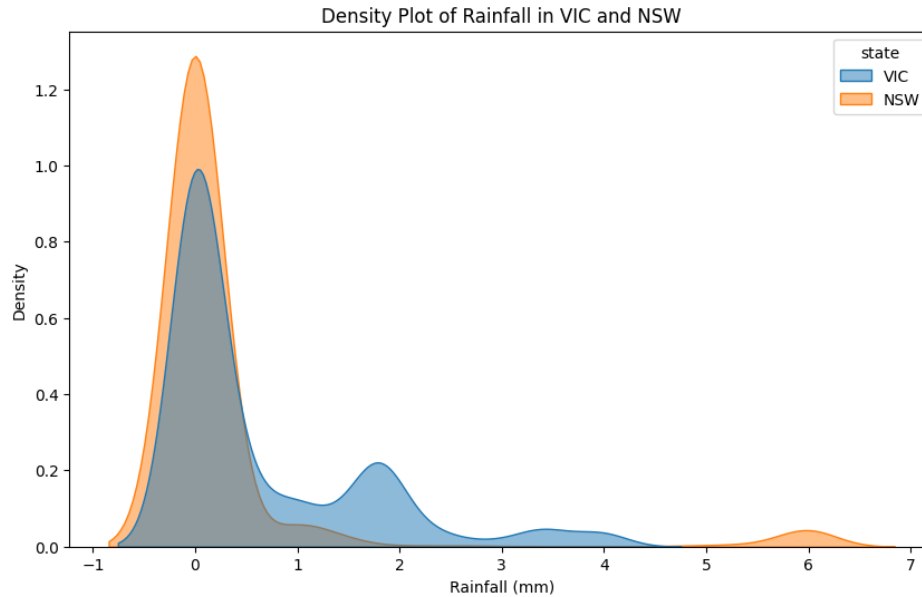


Figure 18: Rain density

4.5 The impact of weather on traffic-injury

For the trial for comparing and analysing the amount of rainfall and the traffic injury ratio (see figure 17), there's no significant relation between two events can be observed. This might be due to lack of observation stations for the weather, or the rough location of traffic injury ratio. As the weather is only collected from 9 different stations spreaded in Melbourne and Sydney, and traffic injury ratio are statistics statewide, and there's no direct time linkage between the dataset since the traffic injury dataset is a historical dataset. Therefore the analysis conducted is only for testing the hypothesis if there is a direct relation between the event of rainfall and the occurrence of traffic injury.

From the density plot (see figure 18), students can easily observe that the plot is skewness. Both Victoria state and new South Wales state are sharing a similar pattern of right or positive skewness. Where most of the data are concentrated in the lower rainfall range, while the case with higher rainfall are relatively rare. Both states have a peak at 0mm rainfall, which illustrates that most of the time the rainfall is quick and small, and no rainfall case is common. In terms of difference, the peak of NSW at 0mm is higher than the state of VIC, which means for the state of NSW, the number of days without rainfall is greater. Overall, both states of VIC and NSW share a majority of rainfall patterns similarly, with only small minor differences.

5 Error handling

When encountering various errors in the function, specific solutions can be implemented to resolve them effectively. For `ConnectionError`, ensure the network connection between the server and the Elasticsearch cluster is stable, verify the configuration details, and check the cluster's health status. For `TransportError`, check user permissions, validate the query syntax, and review the Elasticsearch logs for more details. For `NotFoundError`, confirm the index name is correct and ensure the index is created if it does not exist. For unexpected exceptions, add detailed logging to capture more context, use debugging tools to step through the code, and validate input data formats. By following these steps, we can effectively troubleshoot and resolve issues, ensuring the system's stability and reliability.

When using Kubernetes serverless functions, error handling is a crucial part. Serverless functions are often used to handle critical tasks or real-time data streams. Like the weather harvesters retrieves the real-time data. In the development phase, we handled the function timeouts. By default, Fission's timeout is set to 2 minutes. We have extended the timeout for all functions on Fission to more than 10 minutes.

To provide accurate error messages and facilitate precise error localization, we handle specific HTTP status code, such as error 400 or 500, and log appropriate log details for the error messages. The real-time code like the weather harvesters will retry three times after receiving an error and catch the Exception.

6 Discussions

6.1 Melbourne research cloud

6.1.1 Pros

Agile application delivery:

As the functions are serverless and isolated from each other, they can be delivered in real-time and not considering the current state of the system. They can be deployed and extract and process data immediately. Due to the cloud architecture, everyone is available to see the result.

On-Demand Computing Resources:

Scalability: MRC offers the flexibility to scale computing resources up or down based on project needs, ensuring that researchers have access to the necessary resources.

Elasticity: Resources can be allocated and reallocated dynamically.

Additionally, software developers using MRC no longer need to worry about performance configuration and hardware load balancing. MRC handles these aspects automatically, providing an optimized environment for applications to run smoothly. This allows developers to focus on their core research and development tasks without the need to manage underlying infrastructure complexities.

6.1.2 Cons

Network proxy and port connection: To connect the cluster in MRC, there are only two way:

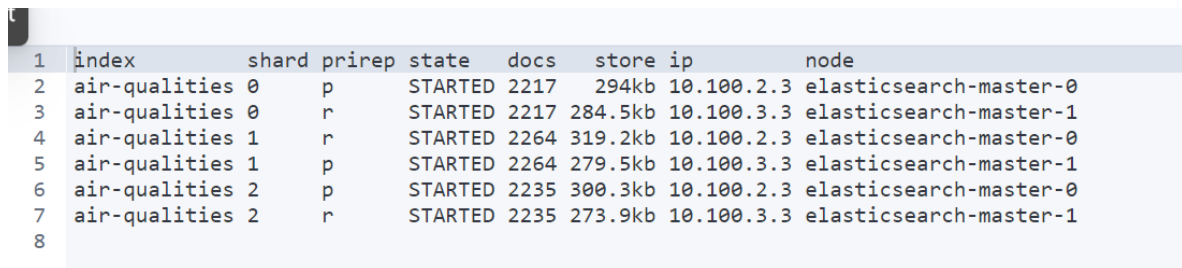
1. Connect to the school network
2. Use the Palo Alto Networks for Virtual Private Network.

The recommended operating system for Windows is the WSL (Windows Subsystem for Linux), WSL is isolated from the main Windows system and cannot access the proxy settings configured on Windows. The Linux version of the Palo Alto network requires a GUI interface for Linux. After attempting without success, our team decided to connect to the campus network at school for software development.

When testing locally and using Kibana, port forwarding is required. However, when multiple team members are testing together, it can lead to network anomalies.

6.2 Elastic Search

6.2.1 Pros:



	index	shard	prirep	state	docs	store	ip	node
1	air-qualities	0	p	STARTED	2217	294kb	10.100.2.3	elasticsearch-master-0
2	air-qualities	0	r	STARTED	2217	284.5kb	10.100.3.3	elasticsearch-master-1
3	air-qualities	1	r	STARTED	2264	319.2kb	10.100.2.3	elasticsearch-master-0
4	air-qualities	1	p	STARTED	2264	279.5kb	10.100.3.3	elasticsearch-master-1
5	air-qualities	2	p	STARTED	2235	300.3kb	10.100.2.3	elasticsearch-master-0
6	air-qualities	2	r	STARTED	2235	273.9kb	10.100.3.3	elasticsearch-master-1
7								
8								

Figure 19: Elasticsearch performance overhead

Advanced database management system When configuring the storage index for Elastic Search, each shard has a primary shard (p) and a replica (r) . This configuration ensures that if a node

fails, data will not be lost because the replica shard can take over the work of the primary shard. For example, shards 0, 1, and 2 of the air-qualities index each have their own primary and replica shards (see figure 19).

Shards ensure load balancing, while primary and replica shards store the same data, guaranteeing data redundancy and reliability.

RESTful API Elastic provides a RESTful API that allows performing CRUD (Create, Read, Update, Delete) operations on the database using HTTP requests. This greatly simplifies the interaction process with Fission.

Advantages of storing and processing geometry data Elasticsearch played an important role in storing the data collected, especially for geometry attributes representing the spatial information of the data. Elasticsearch's powerful querying capability was leveraged to aggregate data based on the geometry attributes. R tree and Quad tree spatial indexing are utilized in Elasticsearch, which ensures the high efficiency of data query and storage. There are two major spatial data types built in Elasticsearch: Geo-point and Geo-shape, which support a variety of geographic queries and aggregation functionalities.

GUI (Kibana) Through port forwarding, Kibana can be directly accessed on a local port and opened in a browser. Kibana's GUI is designed to be intuitive and user-friendly, allowing users to easily navigate and use the tool even without a professional computer background. Additionally, Kibana provides a RESTful API, enabling users to interact with Kibana programmatically, facilitating automation and integration. On the dev page, users can directly perform query processing.

6.2.2 Cons

Computational consuming

```
(ccc) charles@Charleskk:~/ccc/CCC-proj-2/CCC-assignment-2$ kubectl top nodes
```

NAME	CPU(cores)	CPU%	MEMORY(bytes)	MEMORY%
elastic-ia7zz3en22fn-master-0	194m	9%	6584Mi	73%
elastic-ia7zz3en22fn-node-0	217m	10%	7761Mi	86%
elastic-ia7zz3en22fn-node-1	95m	4%	7542Mi	84%
elastic-ia7zz3en22fn-node-2	127m	6%	7841Mi	87%

Figure 20: Nodes Resource usage

```
(ccc) charles@Charleskk:~/ccc/CCC-proj-2/CCC-assignment-2$ kubectl top pods --all-namespaces | sort -k4 -hr
```

kafka	my-cluster-kafka-0	8m	2213Mi	
elastic	elasticsearch-master-0	20m	1965Mi	
elastic	elasticsearch-master-1	42m	1887Mi	
kafka	my-cluster-entity-operator-79f7879b88-4vsb7	9m	667Mi	
elastic	kibana-kibana-74988905c-vc6st	18m	482Mi	
kafka	my-cluster-zookeeper-0	5m	257Mi	
kafka	strimzi-cluster-operator-66f59459b5-ktfp7	6m	243Mi	
fission	router-747967986f-4tkzq	4m	153Mi	
default	python-10418495-7874fcdd5-mpx72	1m	115Mi	

Figure 21: Pods resource usage

(See figure 20 and figure 21), even without performing search operations, Elasticsearch consumes a significant amount of memory. Elasticsearch performs many background tasks to maintain its high performance and high availability.

6.3 Kubernetes and Fission

6.3.1 Pros

Scalability and Flexibility

Data harvesters and query processors were deployed on Kubernetes using the Fission framework. By leveraging Kubernetes and Fission, scalability and flexibility were ensured in handling diverse data analytics tasks, with dynamic scaling of functions based on workload demands.

Rapid development

Fission leverages the Function as a Service framework, which enables developers to deploy functions to Kubernetes without managing the underlying infrastructure. There are multiple programming languages built into the Fission framework, which can adapt to the programming habits of most developers.

RESTful API

Fission functions can be called by RESTful API both inside and outside the cluster. Fission functions programmed can call each other by RESTful API, which allows functions compiled in different languages to trigger each other.

6.3.2 Cons

Cold Start Latency

Data harvesters and query processors were deployed on Kubernetes using the Fission framework. By leveraging Kubernetes and Fission, scalability and flexibility were ensured in handling diverse data analytics tasks, with dynamic scaling of functions based on workload demands.

Fission Real-time Log Latency and Loss

The real-time log of the fission function had severe log latency and loss phenomenon in the phase of development. This phenomenon results in the need to run the function several times during the debugging process to obtain a complete log to locate the bug.

Impact of Network Issue

The network issue tends to influence the rate of successfully using RESTful API to call Fission functions. The frequency of response "error to send request to function" received was relatively high in the development and data analysing stage. There is a positive correlation between the running time of Fission functions and the possibility of this issue occurring.

7 Limitations and improvements

This project utilized Kubernetes, Fission, Elasticsearch to perform big data analytics on the topics of the causes of deaths from respiratory system disease and lung cancer, the correlation between human sentiment and suicide death ratio, and the relationship between rainfall and the proportion of deaths from traffic injuries. Data harvesters and query processors were deployed on Kubernetes using the Fission framework. By leveraging Kubernetes and Fission, scalability and flexibility were ensured in handling diverse data analytics tasks, with dynamic scaling of functions based on workload demands. Processed data could be obtained via the RestFul API of the fission functions and detailed analysis were performed at the frontend.

Although this project demonstrated a detailed process of uploading and processing data on cloud services from MRC, there remains an opportunity for refinement in data processing. While the big data was processed and aggregated based on the geometry information of the data at the backend, the returned processed data could still be extremely large, which would consume a large amount of local resources for data analysis. Data analytics could be performed on the backend and deployed on the Kubernetes instead of on the frontend to reduce the workload of the local devices by offloading the computational workload to the servers of the cloud services.

There were detailed data analytics performed during the project and carefully judged conclusions were obtained in each scenario. However, these conclusions cannot make large contributions to the corresponding fields and can only be used as preliminary references. The major issue was the time difference between the datasets. Due to the disparity in the temporal alignment of the datasets, data processing and aggregation were conducted mainly based on geometric information of the dataset. Therefore, the conclusions obtained are limited in reflecting the actual situation.

8 Team Collaboration

Kaisheng Su: Responsible for developing EPA air quality harvester and analyse the relationship between air quality and the respiratory system disease and lung cancer death ratio.

Can Wang: Setting up the team cluster, Kubernetes, fission and Elastic Search. Further processed Twitter data from assignment 1 to simplify the dataset, and filtered to match with project 2's needs. Visualised sentiment data on maps and analysed correlation between traffic injury rate to the amount of rainfall.

Zhihe Ping: Responsible for BOM(Bureau of Meteorology) harvesters, analysis, improve the structure of Kubernates structures.

Mingtao Yang: Responsible for SUDO harvesters, analyse the relationship between suicide and sentiment score.

Jiankun Zheng: He was absent for all aspects of code development, report writing, demonstration, and presentation, expect for the two initial installation sessions.

In the aspect of communication, Wechat was leveraged during the development process. Github was utilized as version control technique. Multiple branches were established for isolating development work and parallel development. The github repo: <https://github.com/pingzhihe/CCC-assignment-2>
The youtube link: https://www.youtube.com/watch?v=DbgbRl_uhW8

9 Conclusion

In this project, students have built a powerful and efficient data analytics platform using modernised cloud computing technology, such as Kubernetes, Fission and Elasticsearch. By integrate and analyse data from multiple sources, including but not limited to SUDO, EPA, BOM and so on, students have delved into the impact of environment factors onto public health, such as impact of air pollution on mortality from respiratory disease and lung cancer, and the impact of weather on traffic injury rates, and the impact of sentiment scores on suicide rates.

Key findings and achievements

Impact of air pollution on mortality from respiratory disease and lung cancer Through the analysis, students have found that PM10, SO2, and O3 are positively correlated with mortality from respiratory diseases and lung cancer, even though there are some impacts of missing values to the results' accuracy.

Traffic injury rates and weather:

While comparing the amount of rainfall and the rate of traffic accidents, no significant correlations were found. This might be due to the limited amount of weather monitoring stations and rough geographical location of traffic accident data.

Sentiment value and the suicide rate:

By analysing data from social media platform , in this case twitter data, there's only weak correlations between the sentiment value and suicide rate. Despite the weak correlations, we have discovered that the number of suicide death are relatively high in the area of east coast, especially in the state of Queensland and New South Wales.

System advantages:

In the system, students have achieved high extensibility and flexibility, which can efficiently process and analyse datasets in large scale. By utilising RESTful API and FaaS, students have achieved dynamic expansion and flexible data processing, improved the efficiency and accuracy of data analysis.

Challenges and improvement:

During the processing procedure, there exists a cold start delay and network issue may also impact the possibility of a success function call. The limitation of dataset, to be specific the time difference between datasets, affects the comprehensiveness of the conclusion, and more precise time aligned data can be utilised for further improvements.

In terms of data collection, the lack of rollback functionalities may leads to data loss in some cases, and it is recommended to add this function in the future to improve the reliability of datasets.

Further improvements: It is recommended that perform data analytics at the backend instead of local machine, to reduce computational load of local machine. Data collection and processing procedure can be further improved, to ensure integrity and accuracy of data. More real-time data can be used, so the study of the impact of environmental factor to the public health can be further delved.

Through this study, students have not only demonstrated the powerful potential of cloud computing technology in the field of big data analytics, but also provided some insights in the field of public health.