COMP90024 Cluster and Cloud Computing

Analyzing the Impact of Environmental Factors on Public Health and Sentiment Using Cloud Computing Technologies

Team 48

Yifei ZHANG (1174267)

Yibo HUANG (1380231)

Hanzhang SUN (1379790)

Liyang CHEN (1135879)

Yueyang WU (1345511)

Github repo: https://github.com/fullstar5/CCC-Project-2-Group-48.git

May 21, 2024

Contents

Abstract

Using advanced cloud computing technologies and comprehensive data analysis methods, the project explores and understands the dynamic relationship between environmental factors and public health in Australia by integrating data from diverse sources. These data sources include social media platforms such as Twitter, the Environmental Protection Agency (EPA), the Bureau of Meteorology (BoM), and health data provided by the Urban Spatial Data Observatory (SUDO). The project analyses the public's emotional responses to environmental events on social media, explores how these emotions reflect the public's perception of health risks, and attempts to reveal the direct and indirect effects of environmental changes on health. By using Melbourne Research Cloud (MRC), our team has developed a scalable cloud infrastructure solution capable of efficiently processing and analysing large-scale data sets, ensuring robust data analysis and storage capabilities. Through a combination of multi-source data and advanced cloud computing platforms, the project demonstrates the potential of data-driven research in public health and environmental policy making, providing scientific basis and technical support for decision-making in related fields.

Keywords: Cloud Computing, Melbourne Research Cloud, Kubernetes, Openstack, Fission, ElasticSearch, Kibana, SUDO, Data Harvester, Restful API, Public Sentiment, Air Quality

1 Introduction

1.1 Project Overview

With the rapid development of digital technology, the ability to analyze society has never been greater. Integrating cloud computing, social media analytics, and data science offers new ways to generate insights and shape policy. Social media platforms like Twitter capture public emotions and behaviors in real-time, while data from the Environmental Protection Agency (EPA) and Bureau of Meteorology (BoM) provide valuable environmental insights.

This project leverages Twitter data, EPA and BoM data, and health data from the Spatial City Data Observation (SUDO) to explore the interactions between environmental factors and public health in Australia. Using the Melbourne Research Cloud (MRC), we will develop a cloud-based solution to collect, store, and analyze large-scale data. By integrating multiple data sources, this project aims to provide a comprehensive analysis to guide public health policy and environmental strategies. The insights will reveal the impact of environmental changes on public health and show how these changes affect public perceptions and emotions on social media. Overall, this project highlights the potential of integrating diverse data sources within a cloud computing framework to address complex social and environmental challenges, enabling informed decision-making and proactive policymaking.

1.2 Goals

The specific objectives of the project are multifaceted, focusing on data integration, in-depth analysis, and technological innovation. Firstly, the project aims to build a robust data processing platform capable of integrating and analysing diverse data sources, including social media data from Twitter, air quality data from the EPA, climate data from the BoM, and health data from SUDO. The platform will incorporate efficient data processing methods to handle large datasets in real-time, ensuring the support of complex data analysis requirements. Secondly, the project seeks to conduct comprehensive data analysis and extract actionable insights. By combining social media sentiment analysis with environmental and health data, the analysis will explore how environmental factors influence public sentiment and health outcomes. Machine learning and statistical models will be employed to uncover meaningful patterns and trends, providing a scientific foundation for policy development.

The project places great emphasis on technological innovation and dissemination. It will develop and optimise new data analysis tools and methodologies, particularly those utilising big data and cloud computing platforms, with the objective of enhancing the efficiency of data processing and analysis. Furthermore, the project aims to promote the application of cloud computing and big data technologies in environmental science and public health, demonstrating their potential and benefits in addressing real-world challenges.

Through these goals, the project not only deepens our understanding of the relationship between environmental factors and public health, but also promotes the application of science and technology in the field of public policy, providing new perspectives and tools for future research and policy formulation in the field of environment and health.

2 Platforms and Tools

2.1 Melbourne Research Cloud

2.1.1 MRC Introduction

The Melbourne Research Cloud (MRC), hosted by the University of Melbourne, provides on-demand computing resources for students, researchers, and business partners. Similar to Amazon Web Services (AWS) and Google Cloud Platform, MRC is an infrastructure-as-a-service (IaaS) provider that supports various tasks like data analytics and web hosting.

MRC offers approximately 20,000 virtual cores, advanced GPU capabilities, private networks, and load balancing, ensuring a high-quality user experience. Notable features include support for multiple operating system virtual machine images, such as common Linux distributions and Windows. Based on the OpenStack architecture, MRC provides compute, storage, network, and user management resources, with support for APIs and command-line clients.

Therefore, MRC is a powerful platform for researchers to perform complex data processing and analysis tasks

without hardware constraints.

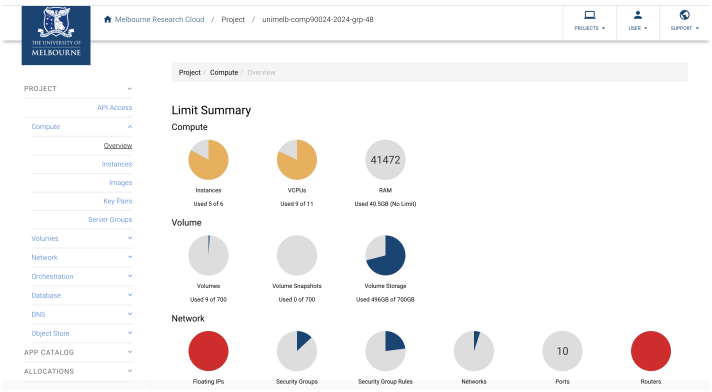

Figure 1: Melbourne Research Cloud COMP90024 2024 Group 48 Resources Overview

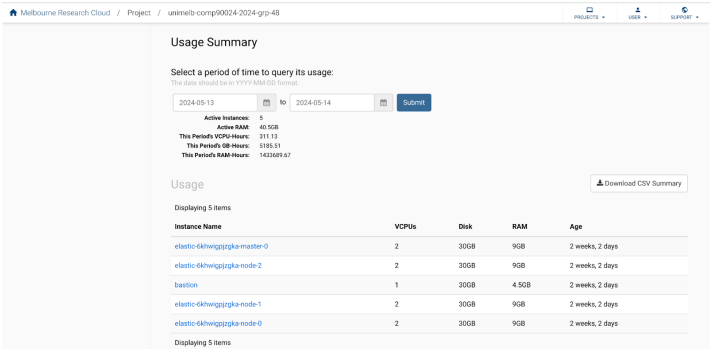

Figure 2: Melbourne Research Cloud COMP90024 2024 Group 48 Instances Configuration

2.1.2 Resource and Instances

The MRC usage report provides detailed resource usage statistics, including the number of active instances, virtual CPU (VCPU) hours, disk usage, and memory usage. This data helps users effectively monitor and manage resources. Based on the project specification, our team was given 11 virtual cpus, unlimited RAM and 700GB of storage. To develop the project, we strategically allocated all of the CPU and approximately 470GB of storage capacity to 5 instances. All resources can be utilized using automation scripts.

As shown in Figures 1 and 2, we allocated 9 CPUs and 496GB of storage capacity for the Elasticsearch cluster database. The primary database instance is allocated 2 CPUs, and each database slave node instance is allocated 2 CPUs, and they share the specified storage space. The fortress machine is allocated 1 CPU and 30GB of storage. This balanced allocation of resources helps maintain efficient performance across the different components of the project, while also ensuring that we have the flexibility to test for scalability.

2.1.3 Security Groups

The Melbourne Research Cloud uses security groups to manage and control inbound and outbound traffic for instances. Each security group acts as a virtual firewall for instances, specifying the protocols, ports, and source/destination IP ranges that are allowed.

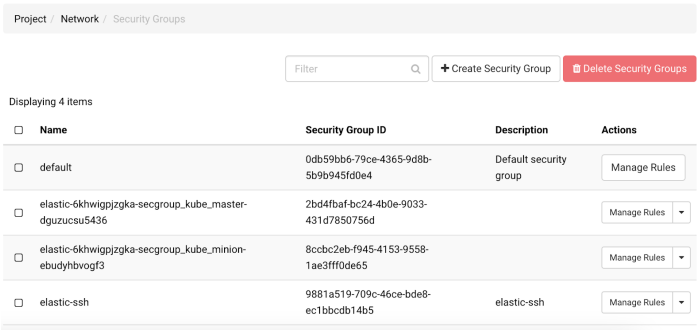

Figure 3: Melbourne Research Cloud COMP90024 2024 Group 48 Security Groups

The Figure 3 above shows the security groups configured in the Melbourne Research Cloud. Here are the details:

- default: This is the default security group provided by the MRC. It typically allows basic inbound and outbound traffic.
- elastic-6khwigpjzgka-secgroup_kube_master-dguzucsu5436: This security group is associated with the Kubernetes master node. It includes rules specific to managing and controlling traffic for the Kubernetes master.
- elastic-6khwigpjzgka-secgroup_kube_minion-ebudyhbvogf3: This security group is linked to the Kubernetes minion (worker) nodes. It has rules tailored to the needs of Kubernetes minions, allowing necessary communication between the master and worker nodes.
- elastic-ssh: This security group is configured to allow SSH access. It ensures secure remote access to instances for management and maintenance purposes.

2.1.4 Pros and Cons of Melbourne Research Cloud

The MRC offers a number of advantages and disadvantages to researchers. One of the primary benefits is its cost-effectiveness. As a publicly funded service and part of the Australian Research Cloud (Nectar), MRC provides affordable cloud computing resources with a transparent and stable pricing model. This reduces the risk of sudden price increases, allowing researchers to focus on their work without worrying about unpredictable costs. Furthermore, the MRC is replete with resources, comprising approximately 20,000 virtual cores, advanced GPU capabilities, private networking, and load balancing. These resources guarantee high-performance computing for large-scale data analysis and complex models, furnishing the requisite computing power and storage space.

Another significant advantage of MRC is its flexibility and customizability. Based on the OpenStack architecture, MRC supports virtual machine images of multiple operating systems, including common Linux distri-

butions and Windows. The instance snapshot feature allows users to save and deploy instance states quickly, increasing efficiency. Moreover, MRC's object storage and volume storage capabilities ensure data persistence and accessibility. In terms of security, MRC provides various features such as security groups and SSH protocols. Security groups permit users to define firewall rules, thereby ensuring instance security. SSH, on the other hand, enables encrypted remote logins for system management, thus enhancing overall security.

However, there are also some drawbacks to using MRC. One of the main challenges is the complexity of usage. MRC requires technical knowledge and experience to utilise effectively. Users unfamiliar with cloud computing and OpenStack architectures may need additional training to manage resources efficiently. Furthermore, the geographical location of the data centre can impact latency for applications requiring extensive data transfer or real-time processing, potentially leading to increased delays. Finally, MRC may lack certain advanced features and services found in commercial cloud platforms. Users requiring advanced machine learning services or complex network configurations may find MRC's capabilities less comprehensive.

Overall, while MRC provides numerous benefits in terms of cost, resources, flexibility, and security, it also presents challenges related to usage complexity, potential delays, and functional limitations.

2.2 Openstack

OpenStack is an open-source cloud computing platform that provides infrastructure as a service (IaaS), enabling organisations to build and manage both public and private clouds. It is designed to be highly scalable and flexible, allowing users to deploy and manage large pools of compute, storage, and networking resources through a web-based dashboard, command-line tools, or RESTful APIs. In this project, we installed few components as our key services and deployed them on MRC cluster:

- Nova: The computation service and responsible for provisioning and managing large networks of virtual machines.
- Neutron: The networking service, which provides flexible networking models, including management of networks, subnets, and routers.
- Cinder: The block storage service, it offers persistence storage service to running instances.
- Keystone: The identity service that provides authentication and high-level authorisation for all OpenStack services.
- Magnum: The container orchestration engine built on OpenStack. It allows users to deploy and manage containerised applications using popular container orchestration tools such as Kubernetes.
- Octavia: The scalable load balancing as a service (LBaaS) solution for OpenStack clouds. It provides advanced load balancing capabilities.

2.3 Kubernetes (K8s)

Kubernetes (K8s) is an open-sourced container orchestration platform that automates many of the manual processes involved in deploying, managing, and scaling containerized applications. Kubernetes is designed to run on various environments, from on-premises data centres to public clouds, ensuring portability and flexibility across different environments. Kubernetes aims to simplify the deployment, operation, and scaling of containerized applications. This often being done by automating routine tasks like deployment, scaling, and management, reduces the operational burden on development and operations teams. In this project, we use Kubernetes with Helm installed, to automatically deploy clusters with templates created by openstack, and easily manage and monitor each node's activities. Kubernetes is often compared with Docker, this is because they are both container management frameworks. However, Docker and Kubernetes serve different roles in the container ecosystem. Docker excels at containerization with its simplicity and lightweight nature, making it ideal for application development, shipping and running across various environments. On the other hand, Kubernetes offers advanced orchestration features such as automatic deployment, scaling, load balancing, and self-healing, making it an excellent choice for managing large containerized applications. While Kubernetes provides powerful management and scalability, it is more complex and resource-intensive, which may not be suitable for smaller deployments. Hence, Kubernetes focuses on orchestration and scaling, allowing efficient and large-scale deployment of applications.

2.4 Fission

Fission is an open-source, serverless functions framework designed for use in Kubernetes environments. It simplifies the process of developing, deploying, and managing serverless functions, allowing developers to focus on writing code without having to concern themselves with the underlying infrastructure. Fission leverages Kubernetes' powerful orchestration capabilities to automatically scale functions in response to incoming requests, ensuring optimal resource utilisation and cost efficiency. With support for multiple programming languages such as Python, it offers developers a flexible and cost-effective solution. Fission offers developers the flexibility to utilise their preferred language for the construction of serverless applications. Its event-driven architecture enables functions to be triggered by a range of events. In this project, our team uses fission to create harvester functions, work with a time-trigger to achieve auto data collection. Moreover, we created a fission REStful api with HTTP trigger for data extraction from ElasticSearch.

2.5 ElasticSearch (ES) and Kibana

Elasticsearch is an open source, distributed search and analytics engine designed for scalability and real-time data retrieval. It enables the efficient storage, search and analysis of large volumes of structured and unstructured data, making it ideal for use cases such as log and event data analysis, full-text search and business analytics. Kibana is an open source data visualisation and exploration tool that integrates seamlessly with Elasticsearch. It provides an intuitive web-based interface for creating dynamic dashboards, visualising data trends, and exploring Elasticsearch indexes. During the project, our team stores all the data we collected into ES and explores data by creating a data view table on Kibana dashboard.

2.6 Github

We use github mainly for version control, but we also will post some of the issues we encountered during the development, along with reasons that cause the problem and the solution to solve it. Our README file and github wiki will briefly introduce our projects and system requirements in terms of running our code.

2.7 Others

Besides tools and platforms mentioned above, we have several general tools used to manage our project development progress. These include: We use google doc for our project documentation, record meeting notes and tasks assignment. We use Draw.io for diagram stretch. We use overleaf for final report writing.

3 System Design and Architecture

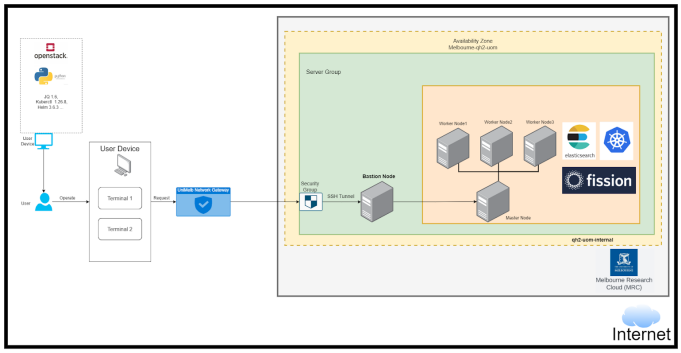

Figure 4: Project Architecture Diagram

The Figure 4 shows our project workflow deployed on the Melbourne Research Cloud (MRC) using OpenStack and Kubernetes, along with various integrated tools and services. The process begins with a user operating from their device, which has terminal access and potentially tools like OpenStack, JQ, Kubectl, Helm, and Python installed. The user device connects to the Melbourne Research Cloud through the UniMelb Network Gateway, which serves as a security and access control point. This ensures that only authorised requests reach the internal cloud infrastructure. Once the user's request passes through the UniMelb Network Gateway, it establishes an SSH tunnel to the Bastion Node. This Bastion Node acts as a secure intermediary server that provides controlled access to the internal network. This is managed by a security group that restricts network traffic to ensure legitimate access only. Access to the Kubernetes cluster within the Melbourne Research Cloud's availability zone (melbourne-qh2-uom) is gained through the Bastion Node. This Kubernetes cluster comprises a Master Node and multiple Worker Nodes.

The Kubernetes cluster comprises several services and applications, including Elasticsearch for search and data analytics, Kubernetes for container orchestration, and Fission for serverless functions, which enables function-as-a-service (FaaS) operations. The user interacts with and manages this Kubernetes-based application environment hosted on the Melbourne Research Cloud using a combination of network gateways, SSH tunnelling, and various Kubernetes tools and service.

4 Data Collection and Storage

4.1 Data Ingestion and Invocation

For the environmental data from EPA and BoM, we have implemented a Python script that can access the public API and automatically select the desired attribute. We then construct a fission function with the script we have programmed and add a time trigger to it. Whenever the public API information is updated, the fission function will be triggered and automatically harvest the newest information. For health data from SUDO and the Twitter data are loaded directly into Elasticsearch using its bulk API.,

We have two methods for obtaining data from Elasticsearch: direct access to the Elasticsearch API (port 9200) and the use of a RESTful API. The former is not secure or effective, and therefore a Python script was developed to query any attributes from any dataset in ES. This script was then converted into a fission function using a specific URL router and HTTP trigger. In this manner, we are able to invoke the data from the URL after connecting to the RESTful port, and pass the parameters to the URL in order to achieve a complex query. This approach is considerably more flexible, elegant, and secure than direct extraction of data from ES.

```
epa_url = 'http://127.0.0.1:9090/epa'

params = {
    'start': start,
    'end': end,
    'avg': avg,
    'time': time,
    'health_advice': health_advice,
    'city': city,
    'health_parameter': health_parameter,
    'size': size,
}

epa = requests.get(epa_url, params=params)
```

Figure 5: API for air quality data

For example, this endpoint retrieves air quality data stored in Elasticsearch for a specified date range. Users can filter the search results based on average value, time of day, health advice, city, and specific health parameters. The response is a JSON object containing hits, each representing an air quality data entry that matches the search criteria. If the system cannot connect to the Elasticsearch cluster, it responds with an error code 500.

4.2 SUDO - Health Data

4.2.1 Data Source

The Space Urban Data Observatory (SUDO) is an integrated data platform that collects and makes available a wide range of health-related data sets. In our project, SUDO has enabled us to analyse the relationship between environmental factors and public health outcomes in Australia by providing health data. The data sets provided by SUDO include information on premature death rates, hospital admissions, and disease incidence, which are

critical to understanding public health trends and their relevance to environmental conditions.

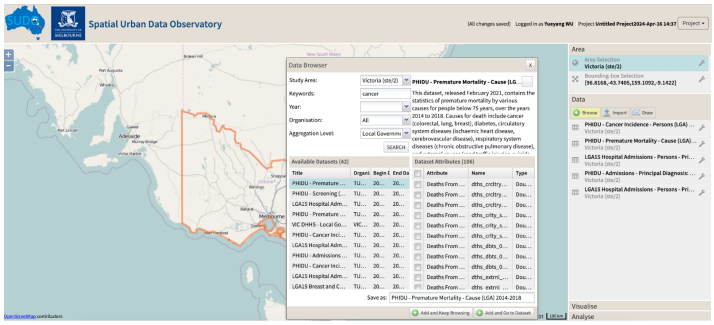

Figure 6: Spatial Urban Data Observatory (SUDO)

Table 1 below provides detailed information on the three main data sets of the Public Health Information Development Unit (PHIDU), including early mortality, primary hospital admission diagnosis, and cancer incidence. The data cover multiple time periods from 2003 to 2018 and include standardised mortality rates, annual adjusted mortality rates, hospital admission rates, and incidence of different cancers.

4.2.2 Data Preprocessing

Given the heterogeneity of the initial dataset in data format and field naming, we first standardized and integrated three health datasets for different time periods and disease categories. These datasets cover multiple disease-related indicators across all regions of Australia between 2003 and 2018.

To standardize field names, we began by reading the JSON file for each raw dataset to obtain the full list of field names for the first element. We then created field name mappings to replace lengthy and noncanonical field names with standard, short names according to predefined mapping rules. For example, we replaced `dths_cncr_0_74_yrs_2014_to_2018_avrge_annl_asr_pr_100000` with `Cancer_asr`. We iterated over each feature and modified field names based on the mapping relationship. The renamed field names conformed to the `DiseaseName_IndicatorType` format, where indicator types included NUM, SR, and ASR.

For data merging, we initialized multi-layer nested `defaultdict` data structures, using LGA codes, year ranges, disease names, and indicator types as keys. We then processed each data file along with its corresponding metadata file by reading the modified dataset file one by one. We iterated through the fields of each element, storing the corresponding values in the appropriate location of the nested dictionary.

The raw data is processed using the `restructure_data` function, which reorganizes the data according to the PHN and LGA levels and extracts relevant statistics. These statistics include ASR, SR, and NUM for each disease. The restructured data is then saved to a new JSON file using the `save_data` function.

Through this series of standardisation and integration steps, we succeeded in transforming multiple health datasets, which initially had fragmented formats and inconsistent field names, into one data set with a unified format specification. Table 2 below shows the format of the processed health data.

	Overview	Year	Attributes
Premature Mortality	The data includes details on deaths between the ages of 0 and 74, standardized mortality rates, and annual adjusted mortality rates.	2003-2007, 2008-2012, 2010-2014, 2011-2015, 2014-2018	Average Annual Standardized Mortality rate (ASR) Number of deaths Standardized Mortality rate (SDR)
Hospital Admissions Principal Diagnosis	The data covers admissions across all hospitals, detailing admission rates and diagnosis information for diseases of different systems.	2014-2015, 2012-2013, 2017-2018	Average Annual Standardized Mortality rate (ASR) Number of deaths Standardized Mortality rate (SDR)
Cancer Incidence	The data includes incidence details for diseases such as cancer, lymphoma, and melanoma of the skin.	2006-2010, 2010-2014	Age-Standardized Rate (ASR) Number of cases Standardized Rate (SR)

Table 1: Health Data Overview

4.3 Twitter Data

4.3.1 Data Source

In this project, the Twitter dataset is an important external data source for analyzing and understanding the discussion, sentiment, and trends on social media for a particular topic. The dataset comes from the SPARTAN platform and contains about 120GB of historical tweet data. This data is extracted and pre-processed into an Elasticsearch database for quick retrieval and analysis.

4.3.2 Data Preprocessing

To download Twitter data from the Spartan platform (file name: Twitter-100gb.json), use the `rsync` command. Given the data size of approximately 112GB, `rsync` is preferred over `scp` because it can resume from the breakpoint if the download is interrupted.

Then, extract the sentiment analysis data from the raw dataset by filtering tweets with emotional values. Use

Type	Name
keyword	PHN, LGA, Period, Disease
float	ASR
integer	SR, NUM

Table 2: Health Data (after Processing) in Elasticsearch

the Python library `langdetect` to detect the language of the tweet content, and extract the emotion score and the city name from the `full_name` field.

To improve processing efficiency and reduce storage requirements, compress the extracted data to retain only sentiment scores, city names, and languages. This reduces the final data size to about 2GB. The processed data is stored in Elasticsearch for efficient data query and analysis. The index structure of the data is as follows:

Type	Name
keyword	Full_name, language
float	sentiment

Table 3: Twitter Data (after Processing) in Elasticsearch

4.4 EPA

4.4.1 Data source

EPA's air quality monitoring data comes from monitoring stations around the world that regularly measure a variety of pollutants in the air, including PM2.5, PM10, sulfur dioxide (SO2) and carbon monoxide (CO). This data is managed through EPA's Air Quality System (AQS) database and accessed through a public API provided by EPA. Our project draws on air quality data sets provided by EPA through public apis, including air pollution indicators at various monitoring sites in Australia.

4.4.2 Data access and preprocessing

Our project involves obtaining air quality monitoring data from the Environmental Protection Agency (EPA), which is done automatically through the `EPA_harvester.py` script. The main function of the script is to extract air quality data from the API interface provided by the EPA, and then store this data into the Elasticsearch database for analysis.
First, the script handles initialization and configuration. It sets up the connection to Elasticsearch by configuring timeouts and basic user authentication. Additionally, it builds the HTTP request using the `requests` library to construct a GET request to the EPA API. This step includes setting appropriate header information, such as User-Agent and Cache-Control, and including API keys. The request parameter specifies `environmentalSegment` as air to accurately obtain air quality-related data.

Next, during data acquisition and parsing, the script handles the API response by parsing the JSON-formatted response content to extract key metrics about air quality, including health advisories for the site, average values, and monitoring times. The timestamp is separated from the response and formatted as date and hour for further

data indexing and analysis. The script then loops through each record, ignoring sites marked as "Camera" type, and extracts valid monitoring records' site name, coordinates, health advice, and average values.

For data indexing and storage, the script constructs each piece of extracted data as an Elasticsearch document. This document includes all relevant information parsed from the API response, such as site name, monitoring time, coordinates, and average values of air pollutants. Documents are assigned a unique ID based on the site name and timestamp, and index names are dynamically generated based on the date of the data. The script attempts to insert each document into Elasticsearch, logging any errors encountered and continuing to process other data.

Deployment and maintenance of the script are managed by executing the `build.sh` script, which installs all necessary dependencies and prepares the script's execution environment. The `requirements.txt` file details all required Python libraries, ensuring that the script can run in any environment with the corresponding dependencies. Through these steps, the `EPA_harvester.py` script efficiently automates the process of obtaining, parsing, and storing air quality data for further analysis.

The index structure of the data is as follows:

Type	Name
keyword	healthAdvice.keyword, healthParameter, hour.keyword, siteName.keyword
float	averageValue, coordinates
text	healthAdvice, healthParameter, hour, siteName

Table 4: EPA Data (after Processing) in Elasticsearch

4.5 BOM

The Australian Bureau of Meteorology (BoM) is an authority of the Australian government responsible for providing meteorological services, including weather forecasting, climate change monitoring and natural disaster alerts. BoM data is of great value to scientific research, policy making, disaster management and the daily life of the public.

4.5.1 Data source

The data collected and published by the BoM covers a wide range of meteorological information, including temperature, rainfall, humidity, wind speed and direction. The data comes from widely distributed weather stations across Australia that use high-precision instruments to monitor weather conditions in real time. In addition to on-site measurements, BoM uses satellites, radar and other high-tech equipment to monitor weather and climate conditions.

4.5.2 Data access and preprocessing

Our project involves obtaining real-time meteorological data from the Australian Bureau of Meteorology (BoM), which is done automatically through the `BoM_harvester.py` script.

The script configuration includes two main URLs pointing to weather data for Melbourne Olympic Park and Melbourne Airport. Using Python's `requests` library, the script requests data from these URLs and obtains JSON-formatted responses. Initially, the script analyzes the response content and extracts meteorological parameters such as temperature, humidity, and wind speed. It extracts fields such as site name, time, temperature, and wind speed from the JSON data and structures them into a document.

The parsed data is used to build documents, each including all necessary meteorological information. Each document is assigned a unique identifier based on the site name and timestamp, with a dynamically generated index name based on the date. Although the actual indexing operations are commented out in the script, each piece of data theoretically has a unique identifier, and a corresponding Elasticsearch index is created based on the date of the data, supporting fast queries and analysis.

Error handling logic is included in the script to manage potential issues during network request failures or data parsing. Deployment and execution of the script are managed by executing the `build.sh` script, which configures the required runtime environment and dependencies. The `requirements.txt` file lists all necessary Python libraries, ensuring consistent script execution across various environments.

After extracting key meteorological data from the public API interface provided by the BoM, the data is stored in the Elasticsearch database for subsequent data analysis and visualisation, and the stored data structure is shown in Table 5.

Type	Name
keyword	cloud.keyword, cloud_type.keyword, local_date_time.keyword, message.keyword, press_tend.keyword, site_name.keyword, vis_km.keyword, weather.keyword
float	air_temperature, apparent_temperature, coordinates, delta_temperature, dew_point, press
text	cloud, cloud_type, local_date_time, message, press_tend, site_name, vis_km, weather
date	timestamp
long	wind_spd_kmh

Table 5: BOM Data (after Processing) in Elasticsearch

5 Scenarios and Data Analysis

5.1 Map

Scenario 1: Analysis sourced from the Twitter data

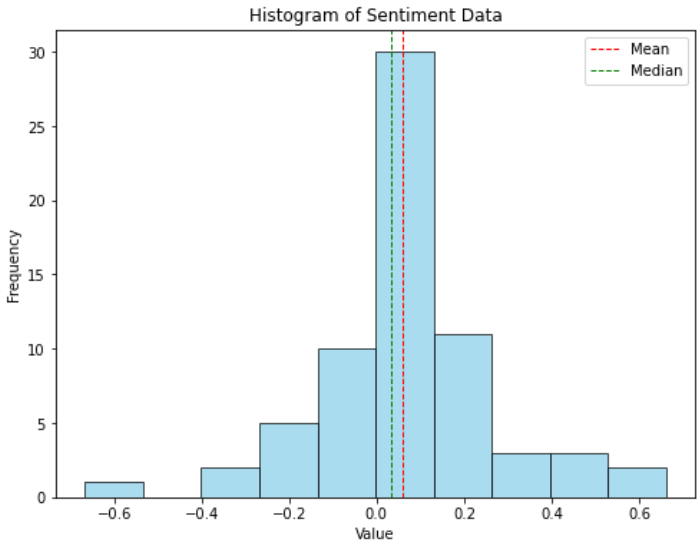

Figure 7: Distribution of the average sentiments

In terms of the sentiment values extracted from the processed Twitter data, initially, the sentiment values are averaged and mapped to cities in Victoria. According to Figure 7, the histogram provides an overview of the distribution of the average sentiments. To categorize the sentiment values:

- If the sentiment is less than -0.05, it is classified as negative.
- If the value lies between -0.05 and 0.1, it is identified as neutral.
- Sentiments greater than 0.1 are classified as positive.

The map (see Figure 8) offers an insight into the average sentiments across the cities. It can be observed that in Melbourne, the average sentiment is neutral, while the cities close to the coastline tend to have positive or neutral sentiments. Nevertheless, regarding negative sentiments, most of the red dots are away from the coastline, assembling in the interior of Victoria.

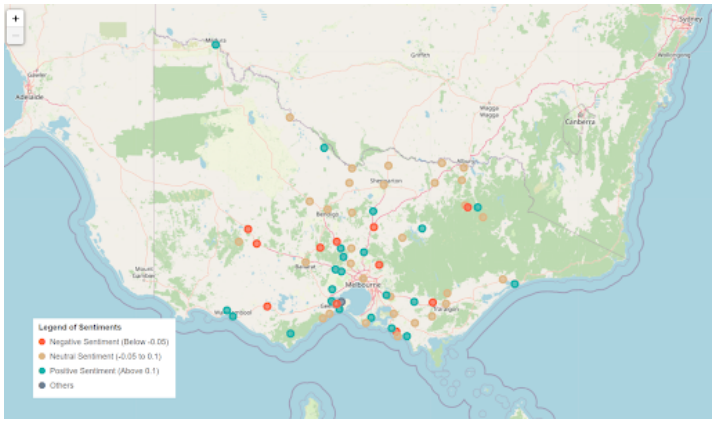

Figure 8: Sentiment Analysis from Twitter Data

Scenario 2: EPA data analysis

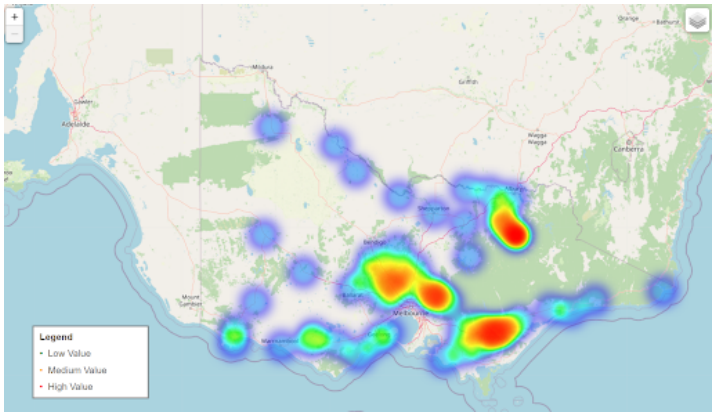

Figure 9: Heatmap of average particle values captured in Victoria

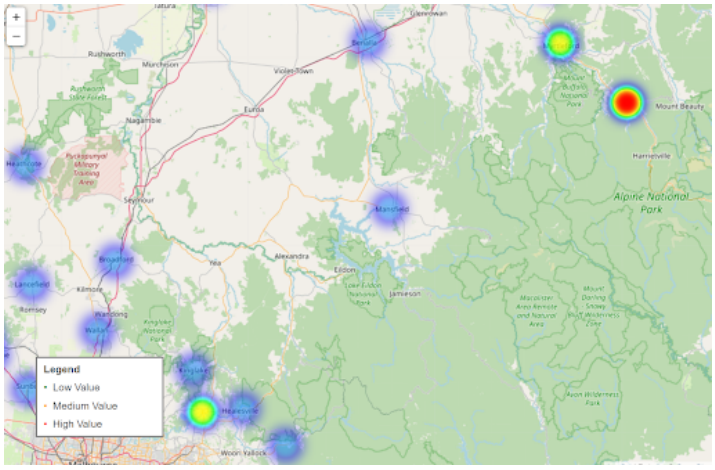

Figure 10: Areas with worse average particle values

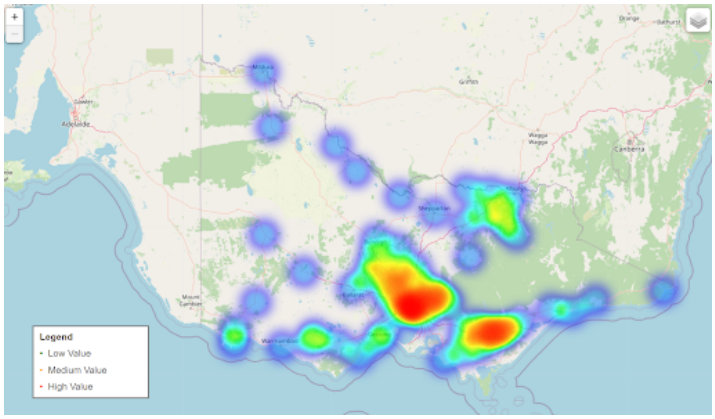

Figure 11: Heatmap of average PM2.5 values captured in Victoria

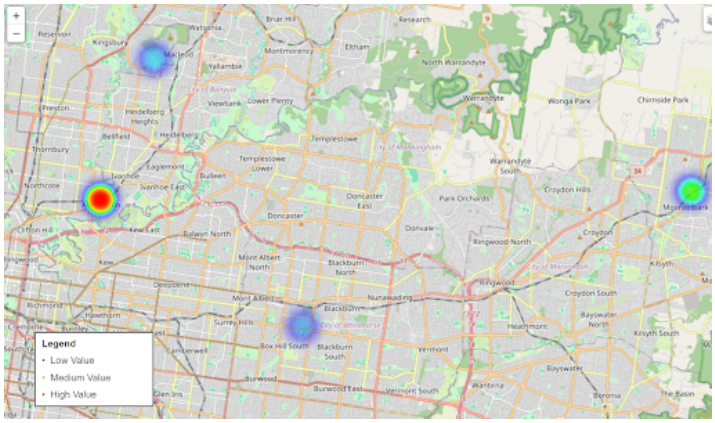

Figure 12: Areas with worse PM2.5 values

For the air quality analysis, we focus on PM2.5 and particles (excluding PM2.5, PM10, SO2, and CO) since they make up the majority of the EPA data. We use maps to visualize these two categories. Figures 9 and 11 are heatmaps displaying the average values of particles and PM2.5, respectively, based on the coordinates of air quality observations. Regarding these heatmaps, higher values not only indicate worse air quality but also reflect a higher density of the data.

In Figure 9, the particle records are concentrated around Melbourne, the northeast, and southeast of Victoria, while PM2.5 records are collected mainly in the southeast of Victoria and an area centered around Melbourne, extending to Bendigo (Figure 11).

When expanding the heatmap, most areas show good air quality. However, in Figure 10, the coordinates near Bright are marked in red, indicating high levels of particle pollution. Areas near Yarra Glen (lower left yellow and green circle) and Myrtleford (upper right yellow and green circle) show medium levels of particle pollution. In Figure 12, Alphington indicates very low air quality due to high PM2.5 levels, while the green circle near Mooroolbark suggests potential slight PM2.5 pollution.

Scenario 3: Health data analysis

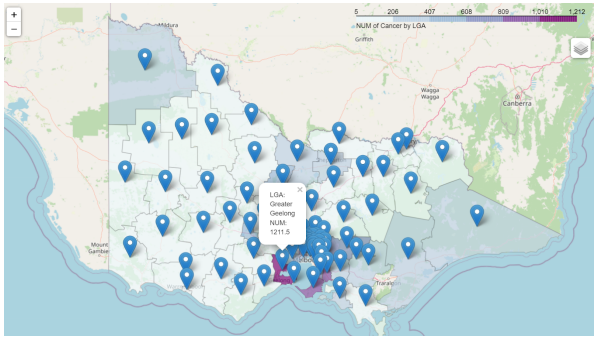

Figure 13: Choropleth map of the number of mortalities caused by cancer

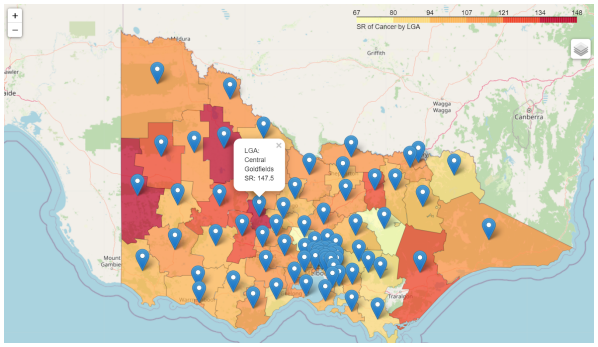

Figure 14: Choropleth map of the standardized mortality rate caused by cancer

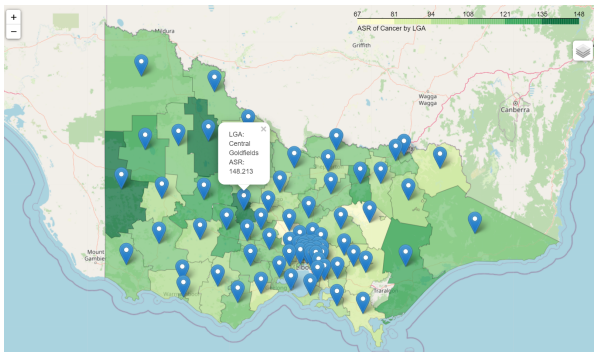

Figure 15: Choropleth map of the average annual standardized mortality rate caused by cancer

According to the health data, the average mortality rates for NUM, SR, and ASR across LGA cities are recorded for different health reasons. Since cancer is one of the most significant causes, we focus on cancer-related analysis. Figure 13 shows that the highest cancer mortality numbers are in the south of Melbourne, particularly in Greater Geelong, where over 1200 people have died from cancer. Meanwhile, the highest mean SR and ASR values are in the western part of Victoria. The city of Central Goldfields has the highest SR and ASR values, approximately 147.5 and 148.2, respectively (see Figures 14 and 15).

5.2 Chart

Chart 1: Twitter Data

Twitter is a global social software and is widely used in Victoria. We can analyze its data to get some inferences, so we analyzed the Twitter data from 4 aspects. Figure 16 demonstrates the average sentiment in different cities, which shows which city has happier people.

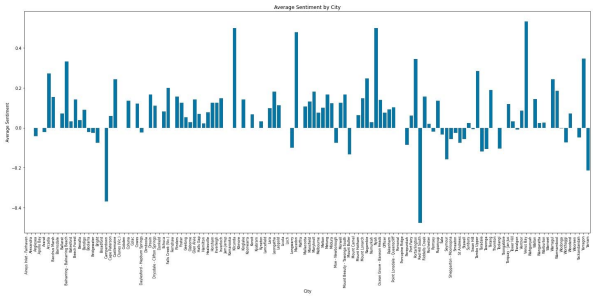

Figure 16: Average Sentiment in Different Cities

Figure 17 demonstrates the average sentiment of users of different languages, which shows which language users are happier.

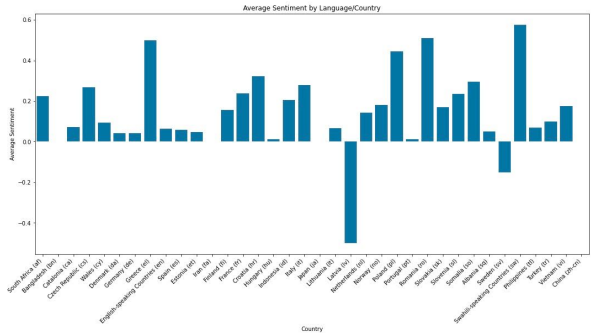

Figure 17: Average Sentiment by Language/Country

Figure 18 is the top ten most frequently used languages in a certain city (e.g., in Melbourne), excluding English.

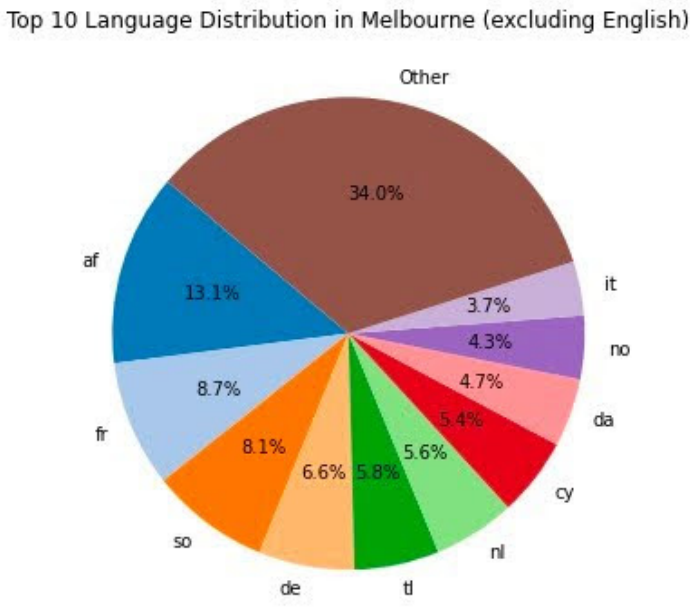

Figure 18: Top 10 Language Distribution (excluding English)

Figure 19 is which language is spoken by the most people in each city besides English.

	city	most frequent language (exclude en)
0	Ballarat	cy
1	Bendigo	de
2	Bridgewater	af
3	Bright	es
4	Cowes	sv
5	Daylesford – Hepburn Springs	et
6	Drysdale - Clifton Springs	de
7	Echuca	tl
8	Geelong	nl
9	Gisborne	da
10	Halls Gap	pt
11	Hamilton	sv
12	Lancefield	tl
13	Lara	fr
14	Leopold	sv
15	Mansfield	no
16	Melbourne	af
17	Melton	so
18	Mildura	fr
19	Moe - Newborough	de
20	Mount Hotham	fr
21	Ocean Grove – Barwon Heads	af
22	Point Lonsdale - Queenscliff	fr
23	Port Fairy	fr
24	Red Hill South	de
25	Shepparton - Mooroopna	pt
26	Stawell	no
27	Sunbury	af
28	Torquay – Jan Juc	af

Figure 19: Language Spoken by the Most People in Each City Besides English

Each language corresponds to a possible country (imported as JSON). The above analysis can also be regarded as an analysis between cities, sentiments, and people of different nationalities.

Chart 2: The Relationship Between PM2.5 and Certain Diseases

We analyzed the impact of PM2.5 concentration (air pollution) on some diseases (air pollution related: COPD, lung cancer, respiratory diseases) in a certain city (e.g., Melbourne) and output a scatter plot. The data points are spread across a range of PM2.5 levels from approximately 2 to 13. The ASR values also range widely, from about 2 to 14. There appears to be a positive correlation between PM2.5 levels and ASR. As the PM2.5 level increases, the ASR also tends to increase. But it is not a perfect linear relationship, possibly because of the influence of other potential factors (see Figure 20).

Figure 20: Relationship between PM2.5 Levels and ASR in Melbourne

Chart 3: The Relationship Between Air Temperature, Dew Point Temperature, Wind Speed, and Air Pollution (PM2.5) in Melbourne

We merged the data about Melbourne from the EPA and BOM by date and time, and then the heat map analyzed the relationship between air temperature, dew point temperature, wind speed, and air pollution (PM2.5). Red represents a positive correlation and blue represents a negative correlation (averageValue means PM2.5 concentration).

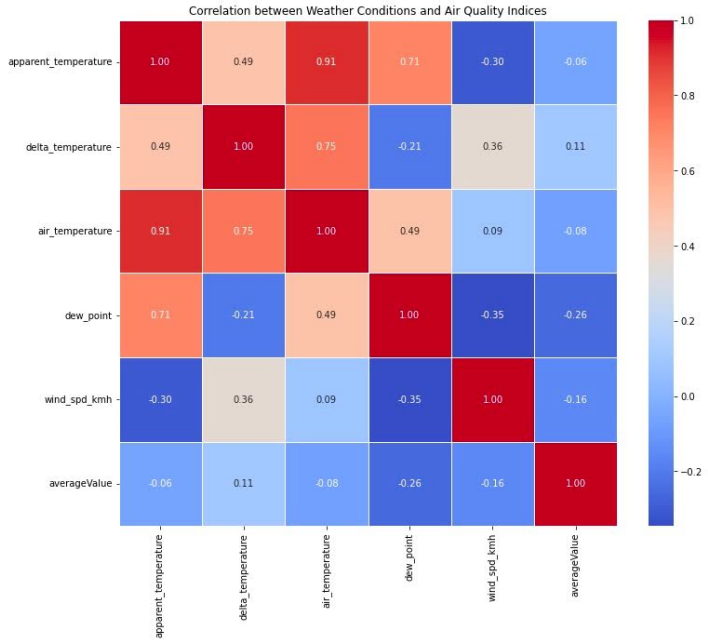

Figure 21: Correlation between Weather Conditions and Air Quality Indices

From figure 21, pm2.5 concentration (averageValue) shows only weak correlations with the weather variables (apparent temperature, delta temperature, air temperature, dew point, and wind speed), suggesting that these weather conditions have a minimal direct impact on PM2.5 levels in the dataset provided. Weather variables such as apparent temperature, air temperature, and dew point are strongly correlated with each other, indicating they are closely related in the dataset.

These observations imply that while there are interactions between different weather conditions, their direct effect on PM2.5 levels is relatively weak, this may be because the PM2.5 concentration in Victoria has always been at a low level and the air quality is generally better.

Chart 4: The Relationship Between Air Pollution (PM2.5) and Residents' Sentiment in a Certain City

We merged Twitter data and EPA data for the same city and used a dual-axis chart to show the changing trends of air pollution levels and resident sentiment on different dates.

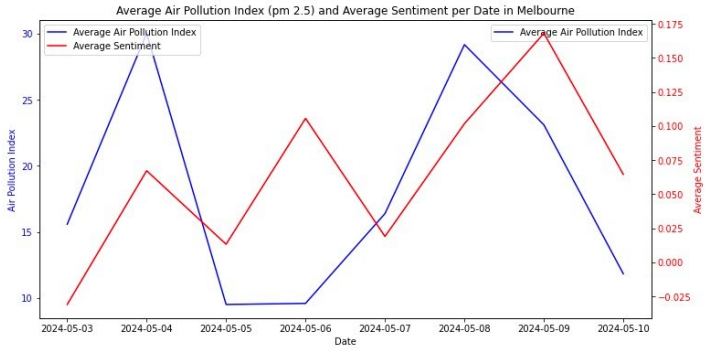

Figure 22: The Relationship Between Air Pollution and Residents' Sentiment in Melbourne

Figure 22 shows an inverse trend between air pollution and sentiment suggesting that higher air pollution negatively impacts residents' sentiment. This correlation is more pronounced on certain days.

For example, Air pollution increased sharply on May 3-4, while sentiment did not so much. Air pollution remained low from May 5-6, while sentiment began to rise. From May 6-7, air pollution increased while sentiment decreased. Moreover, both air pollution and sentiment rise, though sentiment rises slightly less from May 7-8.

There are exceptions, such as May 7-9, where both variables show a similar trend, indicating other factors may also influence sentiment.

Chart 5: The Relationship Between Temperature and Certain Diseases

We can check the relationship between the ASR of the disease (temperature-related: respiratory diseases, cerebrovascular diseases, circulatory system diseases) at different temperatures by setting the type of disease and the name of the city (a bar chart).

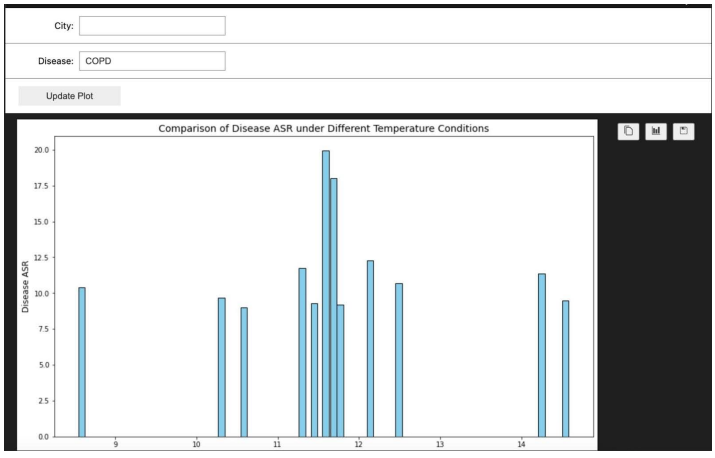

Figure 23: Comparison of Disease ASR under Different Temperature Conditions

Figure 23 shows that the Disease ASR shows significant variation across different temperatures. The highest ASR is observed at 11-12°C, indicating a potential correlation between this temperature and increased disease

incidence. The lowest ASR is observed at 9-10°C and 13-14°C, with lower disease incidence. This suggests a potential link between temperature and disease incidence, with the highest ASR observed at 11-12°C.

Chart 6: The Relationship Between Temperature and Certain Diseases in May

We use a calendar heatmap that displays both average temperature and ASR values for each day in May. The left side of the heatmap represents temperatures, with the intensity of the color indicating the temperature level—darker colors signify lower temperatures. On the right side, the ASR values are shown, with darker colors indicating higher values.

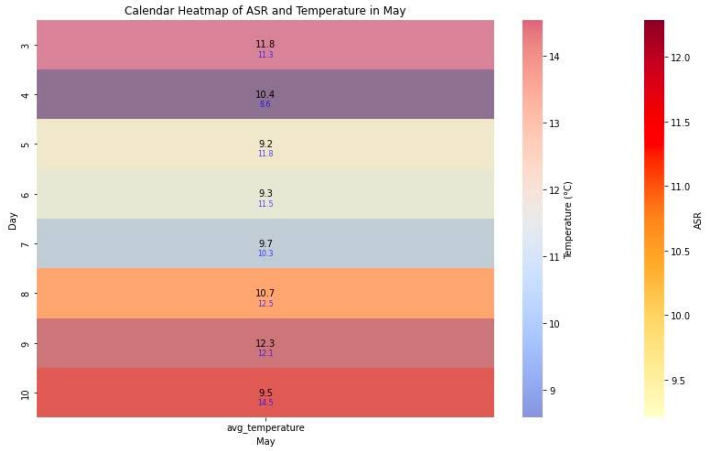

Figure 24: Calendar Heatmap of ASR and Temperature in May

The heatmap shows fluctuations in Disease ASR relative to daily average temperatures in early May. Figure 24 shows that on May 3rd and May 9th, indicating high disease incidence at moderate to high temperatures (11.3°C and 12.1°C respectively). And the lowest ASR is observed on May 5th at 9.2 with a higher temperature of 11.8°C, and on May 6th at 9.3 with a temperature of 11.5°C.

In conclusion, high ASR values are observed at moderate to high temperatures, while lower ASR values do not consistently correlate with specific temperature ranges, suggesting other factors may influence disease incidence.

6 Discussion and Error Handling

During the development of our project, we encountered several key challenges and proposed solutions accordingly.

6.1 Cluster Installation

Our cluster build with five nodes: one bastion node, one master node and three worker nodes. This was done to avoid the simultaneous use of all available resources. The use of redundant resources allows for the quick recreation of additional nodes to replace any damaged ones. For instance, if our bastion node be subjected to an

attack by a malicious individual, we are able to construct a replacement and seamlessly integrate it into the system. This process will result in the system being operational again in a relatively short period of time, therefore enhance flexibility of our system. Furthermore, we can utilise additional storage capacity to create additional volumes and mount them on any node that requires additional storage space. During the cluster installation, we encountered a TLS handshake timeout. When the `kubectl get nodes` command is executed, an error message indicating a TLS handshake timeout is returned. This is due to the network on the Bastion node being disconnected for unknown reasons. To resolve this issue, we removed the Bastion node, then recreated and deployed the required network.

6.2 Harvester Implementation

In the harvester implementation, we could only use the API in the Melbourne area. Specifically, we were only able to use APIs from the Melbourne area (such as Olympic Park and airport), while other APIs would return 403 forbidden access errors. This is because, since 2022, the BoM closed the website scraper, so if the IP address is not from the region, we cannot invoke too many APIs in a short period of time. This problem cannot be solved at present.

6.3 Data Collection

In terms of data collection, we did not have access to some of SUDO's datasets, so we ignored them. At the same time, due to the JSON file size of 100GB, we found that using `rsync` performed better than `scp`, because if the download is interrupted, `rsync` can resume the download from where it was interrupted, while `scp` can only re-download.

6.4 Data Processing

During data processing, we found that one LGA maps to multiple PHNs. To simplify the process, we map it to the PHN with the highest ratio.

6.5 Fission Usage

In the process of using Fission, we encountered many problems:

- **Package Build Failure**: This occurred when the harvester function was built using the `build.sh` file, due to permission being denied. We solved this problem by granting permissions using the `chmod +x` command.
- **Wrong Entrypoint Setting**: During function testing, the program failed to start due to the wrong entrypoint setting. We solved this problem by ensuring that the entrypoint is a Python file name without `.py`.
- **Insecure HTTP Requests and Connections**: Warnings about insecure HTTP requests and connections appeared during function testing. This was caused by an insecure and unauthorized network connection

being exposed to the server. We solved this problem by importing the `urllib3` package and deactivating warnings in the Elasticsearch client to suppress these warnings.

- **Function Response Error**: During function execution, Flask returned a 500 error code. This was because the function was executed but did not return an appropriate response code. The solution was to return a response code at some point in the function to indicate the execution result.

6.6 Elasticsearch Usage

We encountered two major problems with Elasticsearch:

- **Incomplete Data Insertion Before Timeout**: Only part of the data was inserted into the index before the connection timed out. We found that the timeout restriction was too short, so we extended the timeout in the Elasticsearch client.
- **Duplicated Data Insertion**: The same data was inserted repeatedly because the function was accidentally executed twice for some reason. We solved this problem by designing a unique document ID, such as using `date-hour`.

Through the above solutions, we effectively dealt with the various challenges encountered during the project development process and ensured the stability and reliability of the system.

7 Team Member Contribution

Team Member	Task
Yibo HUANG	Process Twitter data, design front-end data calling functions, design front-end (chart part), write README (Github) and write reports.
Liyang CHEN	Collect SUDO data, data ingestion, design front-end maps, report, basic data EDA.
Hanzhang SUN	Collect and process health data, develop ReSTful API, write README, GitHub Wiki, and report.
Yueyang WU	Process and upload health data, design scenarios, write report, and process reports with Overleaf.
YiFei ZHANG	Responsible for MRC environment deployment, ElasticSearch management, and project oversight. Responsible for EPA and BoM harvester programming and deployment on Fission with time-trigger. Responsible for 'Platform and Tools' and 'System Design and Architecture' writing in the report. In charge of development issues documenting on Google Doc and GitHub.

Table 6: Team Member Contributions

8 Future Developments and Conclusions

This project utilizes advanced cloud computing technologies and comprehensive data analysis methods to explore and understand the dynamic relationship between environmental factors and public health in Australia by integrating data from diverse sources. These data sources include social media platforms such as Twitter, health

data provided by the Environmental Protection Agency (EPA), the Bureau of Meteorology (BoM), and the Urban Spatial Data Observatory (SUDO). The project analyzes the public's emotional responses to environmental events on social media, explores how these emotions reflect the public's perception of health risks, and attempts to uncover the direct and indirect health effects of environmental changes.

Using Melbourne Research Cloud (MRC), our team has developed a scalable cloud infrastructure solution capable of efficiently processing and analyzing large-scale data sets, ensuring robust data analysis and storage capabilities. Through the combination of multi-source data and advanced cloud computing platforms, the project demonstrates the potential of data-driven research in public health and environmental policy making, providing a scientific basis and technical support for decision-making in related fields.

In the future, if more comprehensive and standardized data is available, our program can conduct a more comprehensive analysis of the data and can be subsequently developed to make predictions based on certain specific factors.

9 Limitations

Because the data is not large enough and not new enough, the accuracy and rationality of the results are open to debate. The data sets used were not extensive or up-to-date, which affected the reliability of the findings. However, in future studies, the accuracy of this study will be significantly improved if the data timeline can be aligned.

10 Appendix External Links

- Github: `https://github.com/fullstar5/CCC-Project-2-Group-48.git`
- Youtube Link: `https://youtu.be/5HZCUJ7JF1c`

References

- Griebel, L., Prokosch, H.-U., Köpcke, F., Toddenroth, D., Christoph, J., Leb, I., Engel, I., Sedlmayr, M. (2015). A scoping review of cloud computing in healthcare. BMC Medical Informatics and Decision Making, 15(1). https://doi.org/10.1186/s12911-015-0145-7
- Masumi Sugeno, Kawazu, E. C., Kim, H., Virasack Banouvong, Nazife Pehlivan, Gilfillan, D., Kim, H., Kim, Y. (2023). Association between environmental factors and dengue incidence in Lao People's Democratic Republic: a nationwide time-series study. BMC Public Health, 23(1). https://doi.org/10.1186/s12889-023-17277-0
- Melbourne Research Cloud Documentation. (2019). Unimelb.edu.au. https://docs.cloud.unimelb.edu.au/
- Salgado, M., Madureira, J., Mendes, A. S., Torres, A., Teixeira, J. P., Oliveira, M. D. (2020). Environmental determinants of population health in urban settings. A systematic review. BMC Public Health,

20(1). https://doi.org/10.1186/s12889-020-08905-0

- Asif, S., Ambreen, M., Muhammad, Z., Rahman, H. ur, Iqbal, S. Z. (2022). Cloud Computing in Healthcare - Investigation of Threats, Vulnerabilities, Future Challenges and Counter Measure. LC International Journal of STEM (ISSN: 2708-7123), 3(1), 63–74. https://doi.org/10.5281/zenodo.6547289
- Amanat, A., Rizwan, M., Maple, C., Zikria, Y. B., Almadhor, A. S., Kim, S. W. (2022). Blockchain and cloud computing-based secure electronic healthcare records storage and sharing. Frontiers in Public Health, 10. https://doi.org/10.3389/fpubh.2022.938707