COMP90024 2024 S1
ASSIGNMENT 2 REPORT

Group 11
Ziqiang Li, 1173898
Donghao Yang, 1514687
Rui Mao, 1469805
Xiaxuan Du, 1481272
Ruoyu Lu, 1466195

1. Deployment

1.1 Installation

Following the instructions in
https://gitlab.unimelb.edu.au/feit-comp90024/comp90024/-/tree/master/installation?ref_type=heads
Tips: Some downloads have version requirements, so the recommended approach is to follow the method in workshop 5-1 that uses asdf to download the specified version.

1.2 Run Jupyter Notebook

Just follow the instructions below:
1. Download and install Python from the official website
2. Open your terminal or command prompt.
3. Install Jupyter Notebook by running the command: pip install jupyter; and use the command to activate Jupyter Notebook: jupyter notebook.
4. Once installed, navigate to the directory where you want to store your Jupyter notebooks.
5. Type Jupyter Notebook and press Enter.
6. Your default web browser will open, displaying the Jupyter Notebook dashboard.
7. Navigate to the directory where your Jupyter Notebook (.ipynb) files are saved.
8. Find the desired .ipynb file.
9. Double-click on the .ipynb file to open it.
Tips: To use our system, the user needs to ssh connect to MRC and use port-forward in the router and elasticsearch-master.

2. System Architecture and Design

2.1 System Architecture Design Diagram

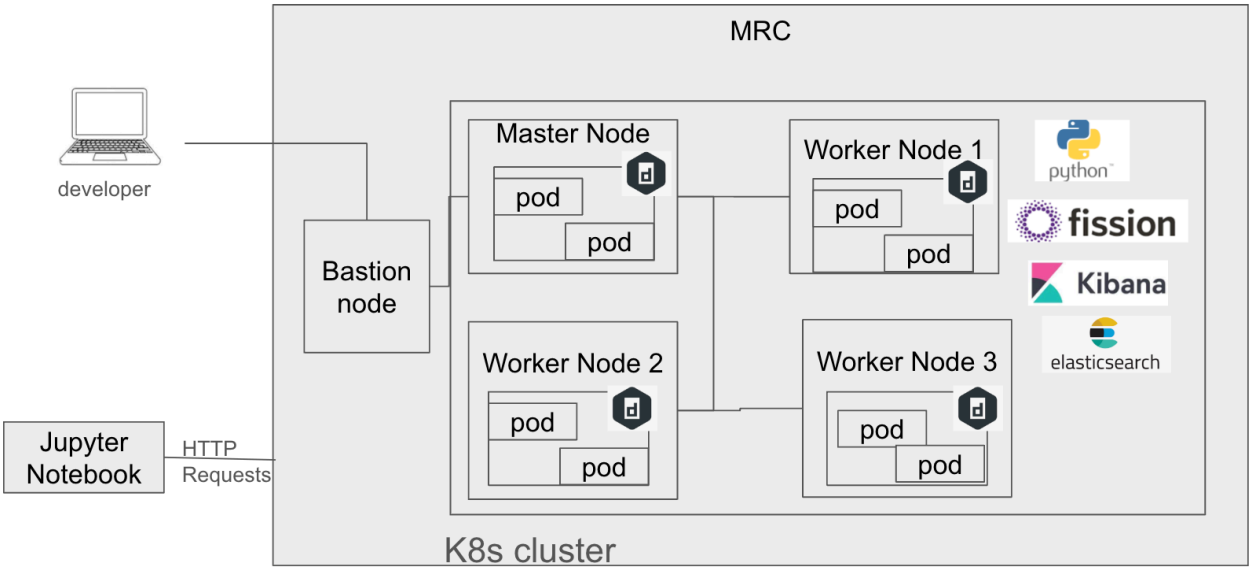

Figure 1. *System Architecture*

OpenStack —--(Provision VMs)----> Kubernetes Cluster (MRC) —(Database)----> Elasticsearch —->(Network and Storage)---> Elasticsearch —(DNS and Ingress)--->
DNS and Ingress

2.1.1 Reasons Behind the Stack

Using Kubernetes (k8s), Elasticsearch (ES), Fission, and Kibana offers a powerful, open-source solution for modern application development:
1. Scalability and Efficiency: Kubernetes orchestrates containerized applications, ensuring scalability and efficient resource utilisation.
2. Real-time Data Analysis: Elasticsearch handles large-scale data storage and real-time analytics, enabling quick insights into application performance and user behaviour.
3. Serverless Computing: Fission brings serverless computing to Kubernetes, simplifying development and automatically scaling applications based on demand.
4. Data Visualization: Kibana provides intuitive dashboards and visualisation tools for data management.
5. Open Source: All components are open-source, fostering flexibility, and easy to get.
Kubernetes (k8s), Elasticsearch (ES), Fission, and Kibana provide powerful open source solutions for our system.

2.1.2 OpenStack Control Layer

Nova (Compute)
Neutron (Networking)
Cinder (Storage)
Octavia (Load Balancing)
Glance (Image)

2.1.3 Kubernetes Layer

Master Nodes (including API Server, Controller Manager, Scheduler)
Worker Nodes
Running application containers
Using Flannel for Network Configuration
Using containers as container runtime

2.1.4 Database

Elasticsearch provides data storage and retrieval functionality

2.1.5 Networking and Storage

Flannel provides container networking
Octavia Ingress Controller provides external access
Cinder CSI Driver is responsible for integrating Kubernetes with Cinder for any block storage requirements, while Elasticsearch handles the storage aspect.

2.1.6 Application Layer

Fission runs serverless functions
Programming Languages: Python

2.1.7 DNS Service

CoreDNS provides cluster DNS resolution.

2.1.8 Monitoring and Visualization

Kibana used for data visualisation and monitoring

2.2 UniMelb Research Cloud

2.2.1 Introduction

Melbourne Research Cloud is a high-performance computing infrastructure that provides a scalable and flexible platform. It uses a private cloud deployment model and offers free on-demand computing resources to the researcher at the University of Melbourne. The MRC is often used for data analysis, web hosting and developing innovative research solutions.

2.2.2 Resource Allocation

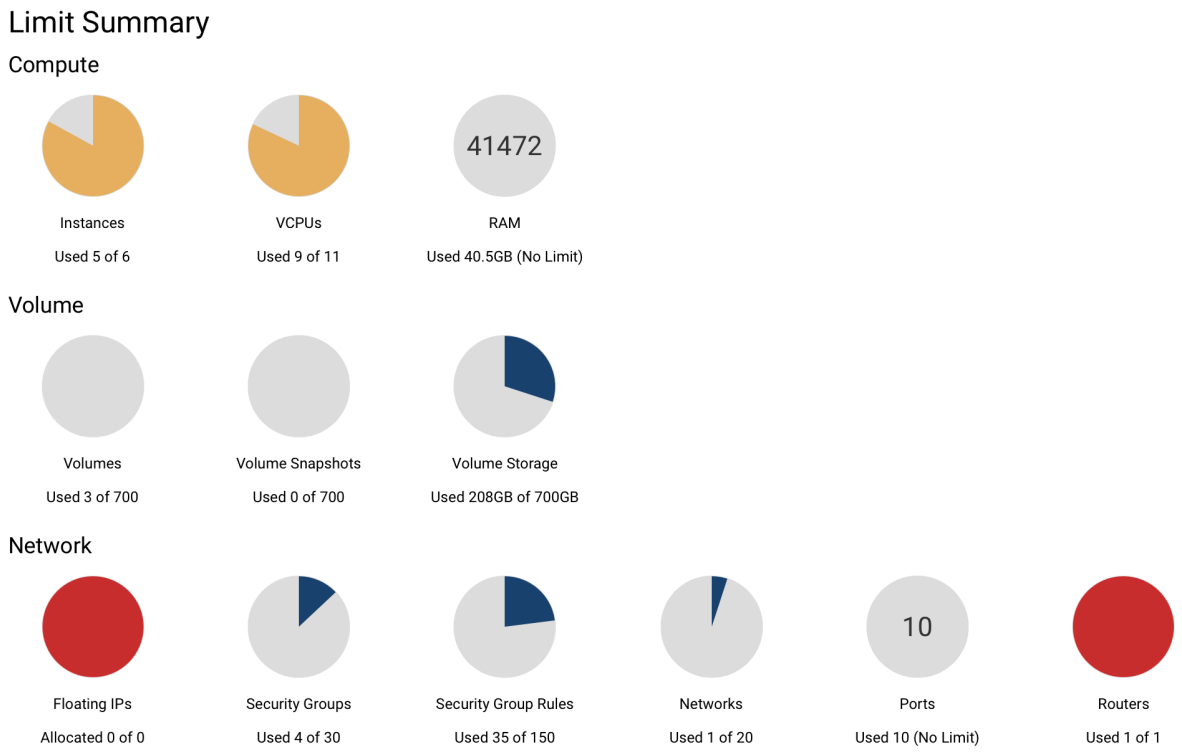

Figure 2. *Melbourne Research Cloud resource usage*

2.2.3 MRC - Pros and Cons

2.2.3.1 Pros

1) Controls

MRC is a private cloud that requires the user to use a VPN to connect to the network of the University of Melbourne, so it is easier to control. Specifically, it consolidates computational resources into a centralised infrastructure which makes it simple to maintain, manage as well and monitor to ensure the resources are used efficiently.

During the project, we found that MRC can allocate cloud resources to specific user groups, and control the access to the data. This feature improves the efficiency of cloud-based development.

2) Efficiency

MRC offers scalable resources allowing the users to efficiently adjust their computing power and storage needs based on our requirements of the project. The key advantage of the MRC is its efficiency as it has superior performance. As MRC is a private deployment model, the firewall of the network of the University of Melbourne ensures high efficiency and robust performance of the network. Therefore, the development team can benefit from the cloud with more computation power compared with other public cloud models. Moreover, as the server of the MRC is located in one location, this can substantially increase the efficiency of data transferring between different instances on the cloud and reduce latency. Therefore, the overall performance can be significantly improved.

3) Security

Security is also a major advantage of MRC, as the allocated cloud resources are exclusively allocated to one client within our group. This exclusivity ensures that the cloud infrastructure and system can be configured independently to meet specific requirements. Moreover, the encrypted key pair for login to the cloud instance also ensures that only authorised users can access the resources. We can also customise the security group within the MRC to the specific needs of each instance. We can design and control the use of ports by changing the security group setting based on the requirement. Lastly, only the people who have access to the network of the university can use MRC to reduce the risk of attack from outside.

2.2.3.2 Cons

1) Availability

The resources allocated to our users on MRC are limited. It does not support storing an extremely large amount of data due to its restriction. This restriction in resource allocation can hinder our project which may require big storage.

Moreover, we do not have physical access to the data stored on MRC meaning that it is impossible to transfer physical data between locations. It can be a significant drawback for those who need direct control over their data assets.

Lastly, accessing MRC, requires the use of a UniMelb VPN to access the network of the university, the VPN is not consistently stable and can introduce additional steps and potential connectivity issues. Therefore, it can be inconvenient for researchers or students working from home to access cloud resources.

2) Cost

As the MRC is a private cloud service it can have issues with staff or management overheads. People who need to use MRC need to build their own cloud infrastructure which introduces management costs. Though we are using the servers rented by the university, the allocation and management of these resources may still incur administrative problems. This includes the time and effort required to manage and allocate computing resources.

2.3 Front-end Component

2.3.1 Retrieving Data via HTTP Requests

```python
import requests
import json

def getdata(index, data):
    url = f"https://127.0.0.1:9200/{index}/_search"
    headers = {
        "Content-Type": "application/json"
    }
    auth = ("elastic", "elastic")
    response = requests.get(url, headers=headers, json=data, auth=auth, verify=False)
    return response.json()
```

Figure 3. *HTTP Requests Function (v1)*

We use HTTP requests as a means of obtaining data. This approach provides compatibility between different systems and enables flexible communication between clients and servers. By leveraging standardised HTTP methods such as GET, POST, PUT, and DELETE, we ensure efficient retrieval of data from the backend.

2.3.2 Backend Processing and Sending Filtered Data via RESTful API

```python
import requests

def getfromurl(url):

    response = requests.get(url)
```

Figure 4. *HTTP Requests Function (v2)*

Some of our data is processed on the backend and then dispatched to the frontend via a RESTful API. This approach offers several advantages, including enhanced security and control over data transfer. By adopting RESTful principles, we ensure that our API design is consistent, scalable, and maintainable. The stateless and resource-oriented nature of RESTful architecture simplifies the data retrieval and manipulation process and promotes efficient utilisation of server resources. Additionally, separating data processing from front-end operations allows for better organisation and encapsulation of business logic in the back end, resulting in a modular and maintainable code base

2.4 Backend Component

The server backend is deployed on the instance shown in Figure 2, written in Python and using the Flask framework. This setup supports providing services through the RESTful API interface. The main function is to handle API resource requests from the front end and return data for visualisation. The data is stored in the Elasticsearch database in JSON format.

2.4.1 Fission

Fission is an open-source framework that runs on Kubernetes infrastructure at fission.io. It leverages Docker containers to provide Function-as-a-Service (FaaS) functionality.
Fission FaaS simplifies application deployment by allowing developers to write functions and execute them in the cloud. Its benefits include serverless architecture, scalability, fast startup, flexible language support, event-driven execution, and resource efficiency.

We chose to use Fission to implement our system due to the many advantages Fission offers and our utilisation of Kubernetes.

2.4.2 RESTful API in Fission

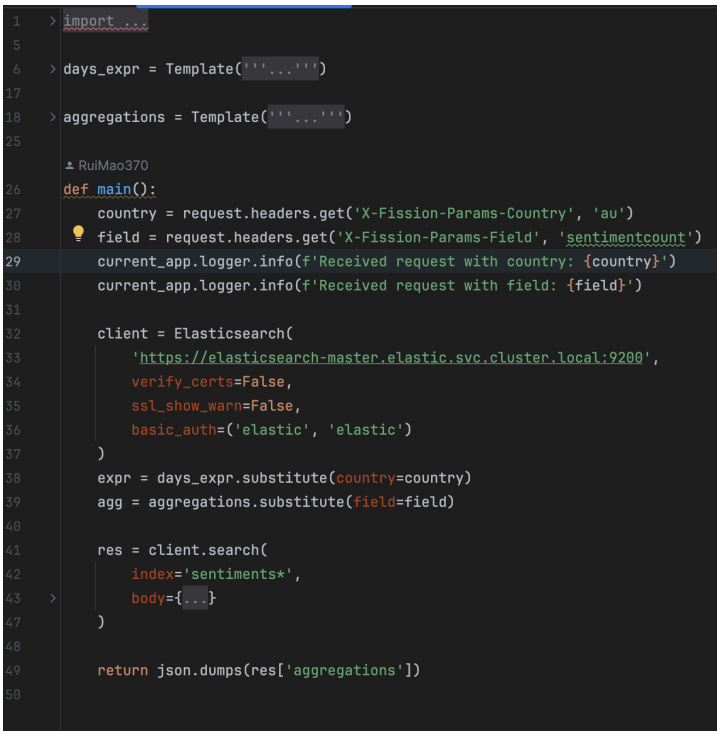

Figure 5. *RESTful API in Fission (sentiments)*

The request parameters and request body can be accessed using regular Flask objects, and the path parameters can be accessed via headers.
A RESTful API is a network service that follows specific communication principles for communication between clients and servers. It is based on standard HTTP methods such as GET, POST, PUT and DELETE and uses URLs to identify resources. Communication is stateless, meaning each request contains all necessary information, and responses are typically returned in JSON or XML format. This architecture makes it easier to build scalable and flexible web applications.

2.5 Database Component

2.5.1 Elasticsearch

Elasticsearch is an open-source distributed search and analytics engine designed for horizontal scalability and real-time data processing. It simplifies data indexing and search by allowing large amounts of data to be stored, searched, and analysed quickly and efficiently.

Due to the numerous advantages offered by Elasticsearch and the ability to handle large-scale, real-time data, we choose to implement our data indexing and search system using Elasticsearch.

2.5.2 Elasticsearch - Pros and Cons

2.5.2.1 Pros

Cluster setup in Elasticsearch allows the sharing of data and workload between nodes to ensure high availability. This allows easy scaling, boosted performance, and redundancy. In this way, it could ensure the availability of data even upon the loss of a single node.

Elasticsearch indexes data in a way that could accelerate the rate at which data could be ingested or retrieved, making this good for dynamic data. Indexing of form data is done in an optimised manner for speed and remaining life to queries of data search wisely. This is crucial for handling dynamic data like the weather information from BOM.

Elasticsearch also has a high integration and great ecosystem, which integrates well with other tools like Fission for serverless functions and Kubernetes for orchestration, facilitating a seamless workflow for our project.

2.5.2.2 Cons

While the Elasticsearch Query DSL is quite strong, it has a considerable learning curve compared to traditional SQL, and sometimes it can bring the need for extra training and adaptation of team members.

2.6 Test

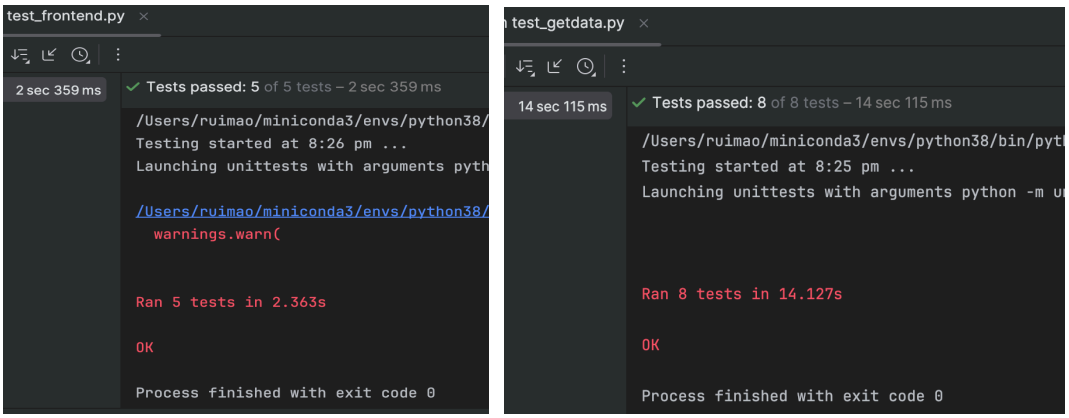

Figure 6. *Results of the test*

As shown in the pictures above, we test the function of getting data from ES and the basic function in the front end. We use automated test tools - Unittest.

Unittest in Python's unit testing framework. It provides a rich set of testing tools and methods, including test cases, test suites, assertions, test devices, etc. Unittest also supports functions such as test execution, test reporting, and test coverage. Using unit tests can help developers quickly discover and fix problems in the code during the development process, and improve the quality and stability of the code.

3. Data Delivery

In this section, we will describe how we transfer these data and how we process them.

3.1 Data Collection

3.1.1 SUDO

Figure 7. *Spatial Urban Data Observatory*

The data on SUDO is static and lacks an API for retrieval. After filtering out the data relevant to scenarios, we downloaded it to our local laptop and then utilised Node.js technology to upload the JSON files to Elasticsearch.

We obtained income data categorised by age from SUDO for the Melbourne area, as well as accident statistics from 2010 to 2020 for Tasmania.

3.1.2 Bureau of Meteorology

Figure 8: *Bureau of Meteorology*

We obtained real-time weather data for the city of Ballarat from BoM. As this data is dynamic and continuously updated, we configured a TimeTrigger to periodically fetch the latest data into Elasticsearch. This allows for real-time data analysis and presentation on the front end.

3.1.3 City of Ballarat: Victorian Government Open Data

Figure 9. *City of Ballarat*

We acquired the air quality data for the city of Ballarat from this website, which, like the weather, is dynamic and real-time. We've set up a TimeTrigger to periodically fetch the latest data into Elasticsearch for real-time data analysis and visualisation on the front end. Additionally, we're combining this data with weather information to analyse the relationship between weather conditions and air quality.

3.1.4 Australian Digital Observatory

Figure 10. *Australian Digital Observatory*

We acquired sentiment data of individuals in the Melbourne area from ADO. This data was derived from Twitter activity over a specific period and analysed using NLP techniques. It constitutes static data. Apart from analysing only this data, we also correlate this sentiment data with income levels, seeking to identify any relationship between income and sentiment. To facilitate joint analysis with income, both datasets are classified geographically according to the GREATER CAPITAL CITY STATISTICAL AREA (GCCSA) standard.

3.2 Data Storage in ES

We interact with the Elasticsearch using the curl command, we first define the setting and mappings of the index in the curl command including the number of shards and replicas, the scheme of the index is defined in mappings. Then, we run the curl command, the request is sent to the elastic search server localhost 9200. The index with a defined name will be created on Elasticsearch.

To view the index created, we use Kibana. We first do port forwarding which opens the port of Kibana on our local machine, localhost 5601. Then run it locally. To view data, we first create the index pattern in the “Management” section. We create the index pattern using the index we created in elastic search. Then we can view the data on the discovery page of Kibana.

4. System Functionality

In this part, the system functions include function implementation and description of three scenarios. The Scenarios section provides detailed information about the three scenarios, including explanations of their selections, as well as accompanying charts and comparison results.

4.1 Functionalities Achieved

1. Utilise Elasticsearch (ES) for data storage and analytics.
2. Build a cloud infrastructure using Kubernetes (MRC) to support analytics.
3. Implement automated data harvesting for data analytics.
4. Deploy Elasticsearch (ES) on Kubernetes (k8s) for data storage.
5. Add and get the collected data using Fission serverless functions.
6. Visualise the analysis results and display them using Jupyter Notebook.
7. Visualise the data using Kibana to check the functionality.

4.2 Cities to be Studied

In different scenarios, we chose different cities to study.

For scenario 1, we chose Ballarat as the target city as there are available real-time weather and air quality data sources that suffice the data stream requirement in the assignment specification. At the same time, these air quality data and weather data own attributes that are suitable for researching the correlation between them.

Figure 11. *Ballarat*

For scenario 2, we use the data from the Greater Capital City Statistical Areas (GCCSA), which is shown in Figure 12, as it includes the functional extents of the capital cities in each of Australia's eight states and territories. This comprehensive coverage allows us to represent Australia effectively. And it also provides detailed insights into the differences between the capital cities and other regions.

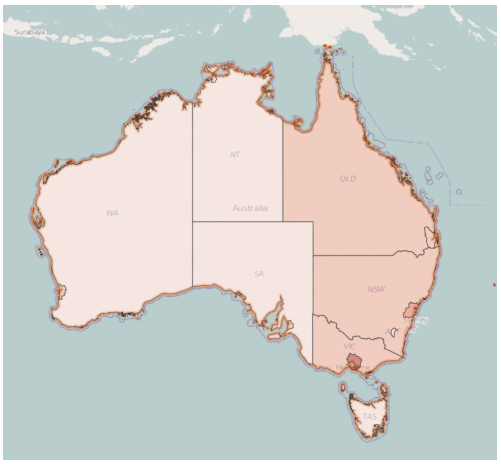

Figure 12. *GCCSA*

Figure 13. *Tasmania*

For scenario 3, we chose Tasmania crash data from 2010 and 2020 as we are interested in the relationship between light conditions and the occurrence of car accidents.

4.3 Collected Data Census

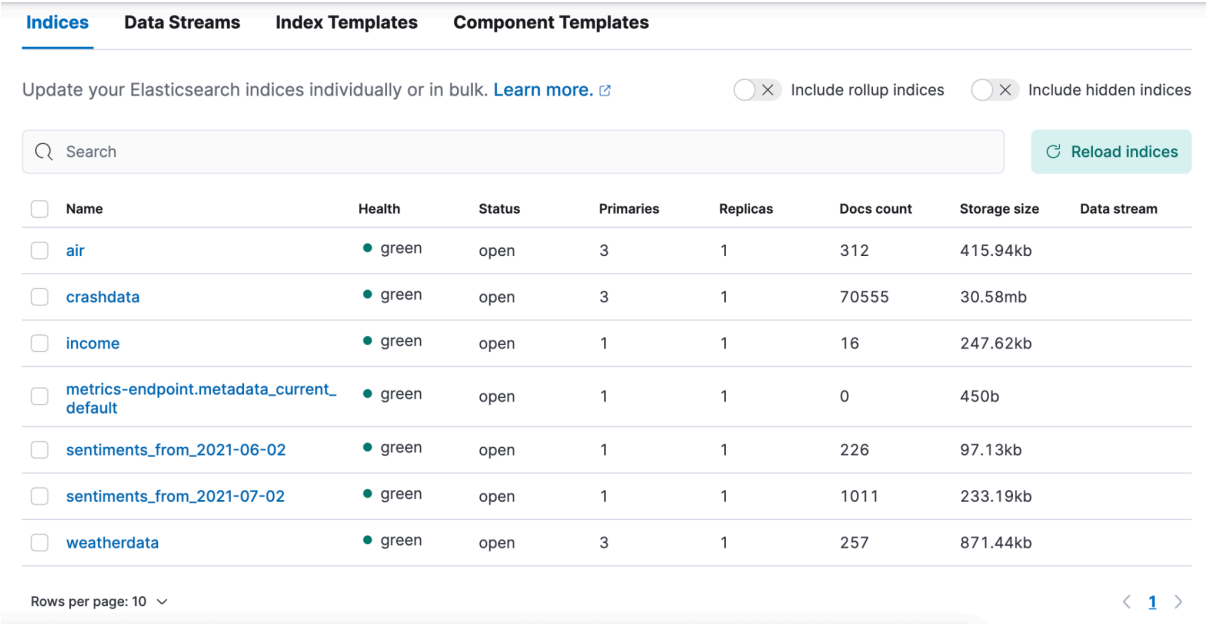

Figure 14. *collect data census for all scenarios*

4.4 Scenarios

4.4.1 Relation between Air Quality and Weather

4.4.1.1 Scenario Description

This scenario focuses on analysing the correlation between the daily air and weather in Ballarat. real-time data

4.4.1.2 The Reason Why We Choose This Scenario

Multiple factors contribute to the final weather conditions, including particulate matter measurements such as PM1, PM10, and PM2.5, as well as wind speed, and so on. These elements will influence various weather attributes, such as visibility and apparent temperature, affecting how we perceive and experience the weather. Therefore we analyse the relations between them.

4.4.1.3 Graphical Result

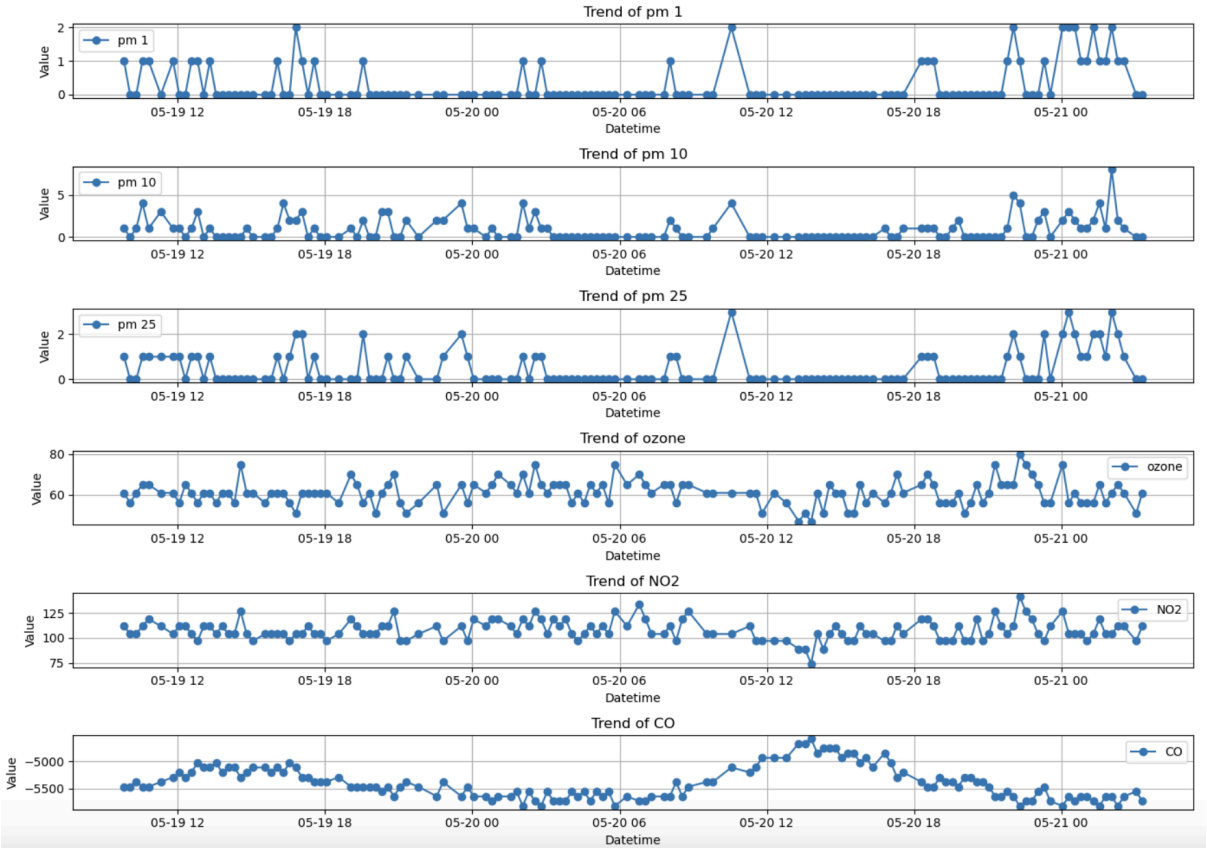

Figure 15. *Trends in Ballarat air quality level*

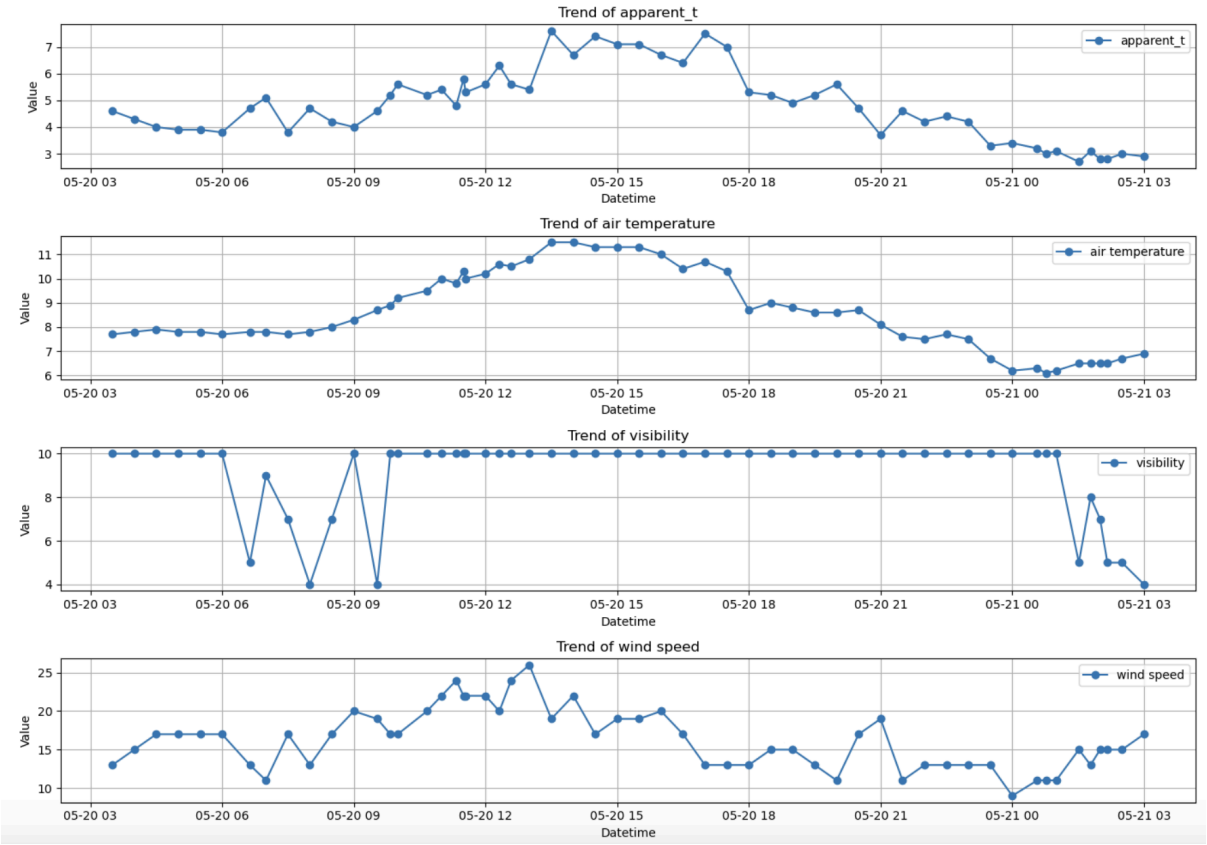

Figure 16. *Trends in Ballarat weather*

According to Figure 15 and Figure 16, apparent temperature is positively correlated with temperature. Visibility changes following the changes in wind speed. An increase in wind speed leads to the enhancement of visibility, and a decrease diminishes it. The trends for PM1, PM10, and PM25 are generally similar. As pollution levels rise, visibility decreases.

2024-05-20 00:00:00	Partly cloudy
2024-05-20 00:40:00	Partly cloudy
2024-05-20 01:00:00	Partly cloudy
2024-05-20 01:20:00	Partly cloudy
2024-05-20 01:30:00	Partly cloudy
2024-05-20 01:33:00	Partly cloudy
2024-05-20 02:00:00	Cloudy
2024-05-20 02:19:00	Partly cloudy
2024-05-20 02:35:00	Partly cloudy
2024-05-20 03:00:00	Partly cloudy
2024-05-20 03:30:00	Partly cloudy
2024-05-20 04:00:00	Partly cloudy
2024-05-20 04:30:00	Partly cloudy
2024-05-20 05:00:00	Cloudy
2024-05-20 05:30:00	Partly cloudy
2024-05-20 06:00:00	Partly cloudy
2024-05-20 06:30:00	Partly cloudy
2024-05-20 07:00:00	Partly cloudy

Figure 17a. *Ballarat cloud*

datetime	air quality
2024-05-20 01:33:11	Fair
2024-05-20 01:48:11	Fair
2024-05-20 02:18:11	Fair
2024-05-20 02:48:11	Fair
2024-05-20 03:18:10	Fair
2024-05-20 03:33:17	Fair
2024-05-20 03:48:17	Fair
2024-05-20 04:03:10	Fair
2024-05-20 04:18:10	Fair
2024-05-20 04:33:17	Fair
2024-05-20 04:48:10	Fair
2024-05-20 05:03:10	Fair
2024-05-20 05:18:16	Fair
2024-05-20 05:33:09	Fair
2024-05-20 05:48:09	Fair
2024-05-20 06:03:16	Fair
2024-05-20 06:18:16	Fair
2024-05-20 06:48:09	Fair
2024-05-20 07:03:09	Fair
2024-05-20 07:18:15	Fair

Figure 17b. *Ballarat weather*

These two figures show the weather and air quality in Ballarat, we can see in Figure 17a and Figure 17b that when it is partly cloudy, the visibility is not that good. According to Figure 17a and Figure 17b, the pollution in Ballarat is not so high and the air quality is fair.

4.4.2 Relation among Income, Age and Area

4.4.2.1 Scenario Description

This scenario focuses on analysing the correlation between income and age in 16 states and big cities in Australia.

4.4.2.2 The Reason Why We Choose This Scenario

Income is a crucial aspect of life, and it's clear that income levels vary significantly across different regions and age groups. In this context, we use data in the GCCSA format to study the income distribution among different age groups across Australia's 16 states and big cities. This research helps people gain a better understanding of the income situation in Australia, enhancing their knowledge of the country.

4.4.2.3 Graphical Result

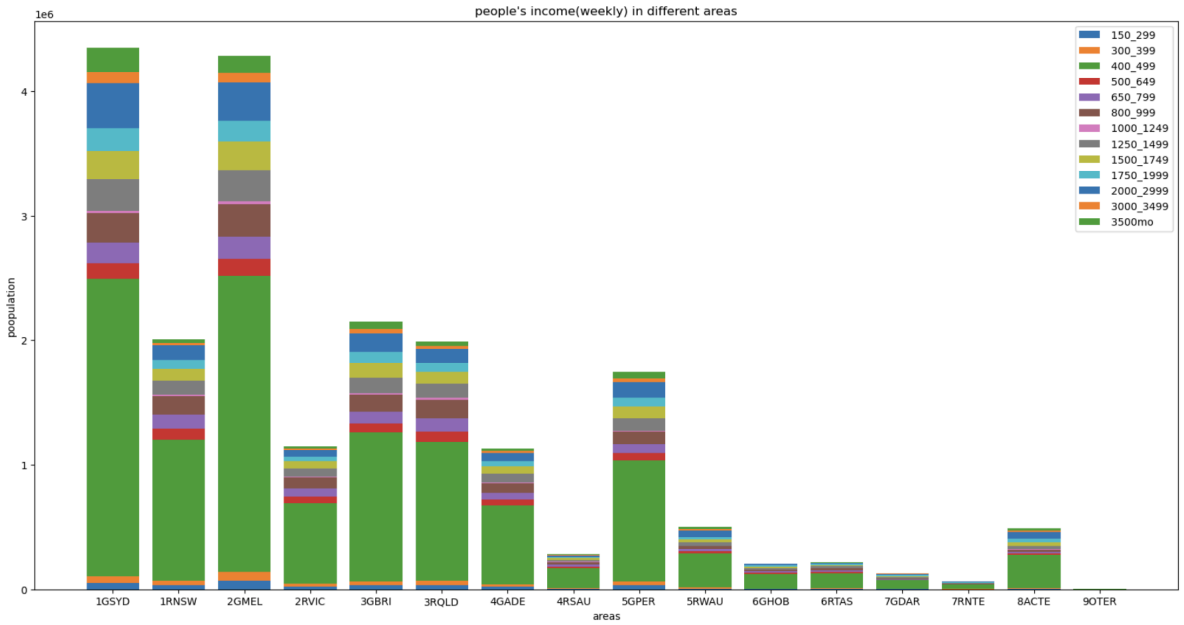

Figure 18. *Stacked Column Chart for income*

Figure 18 shows different incomes in various regions. According to the stacked column chart, the major cities such as Sydney, Melbourne, and Brisbane have a significantly larger population with income compared to other regions in their states. This is shown in the charts by the tallest bars, highlighting the economic significance of these major cities. What's more, it shows that these areas have a higher population density. Because they offer more job and lifestyle opportunities, which attract people seeking employment and a better quality of life.

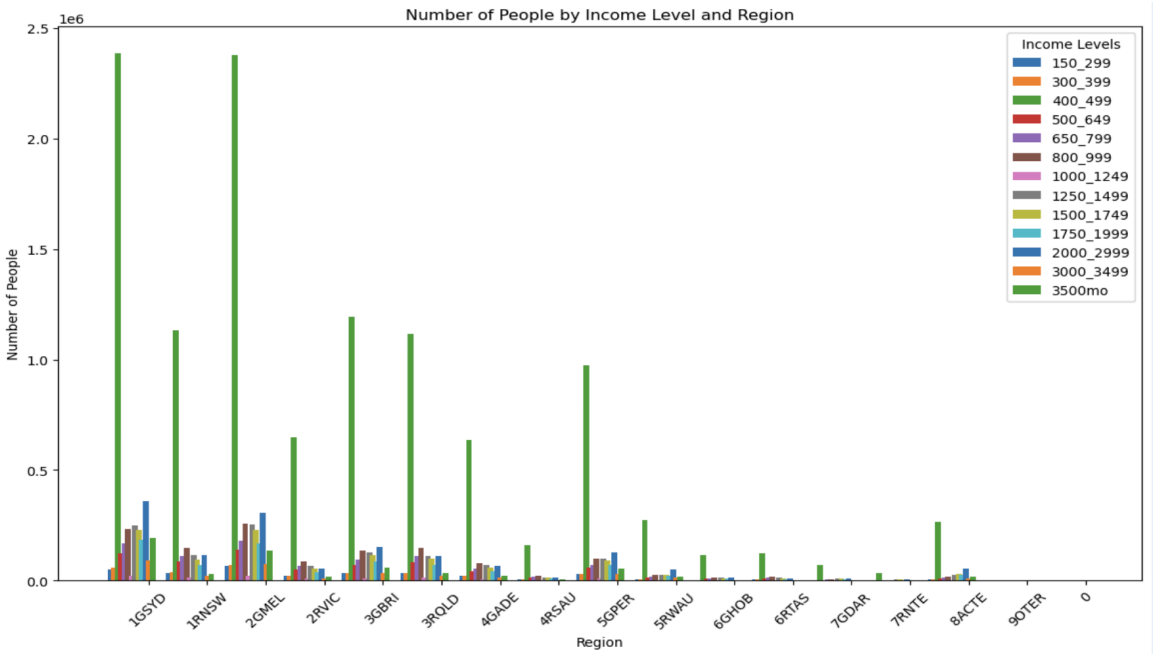

Figure 19. *group bar chart for income*

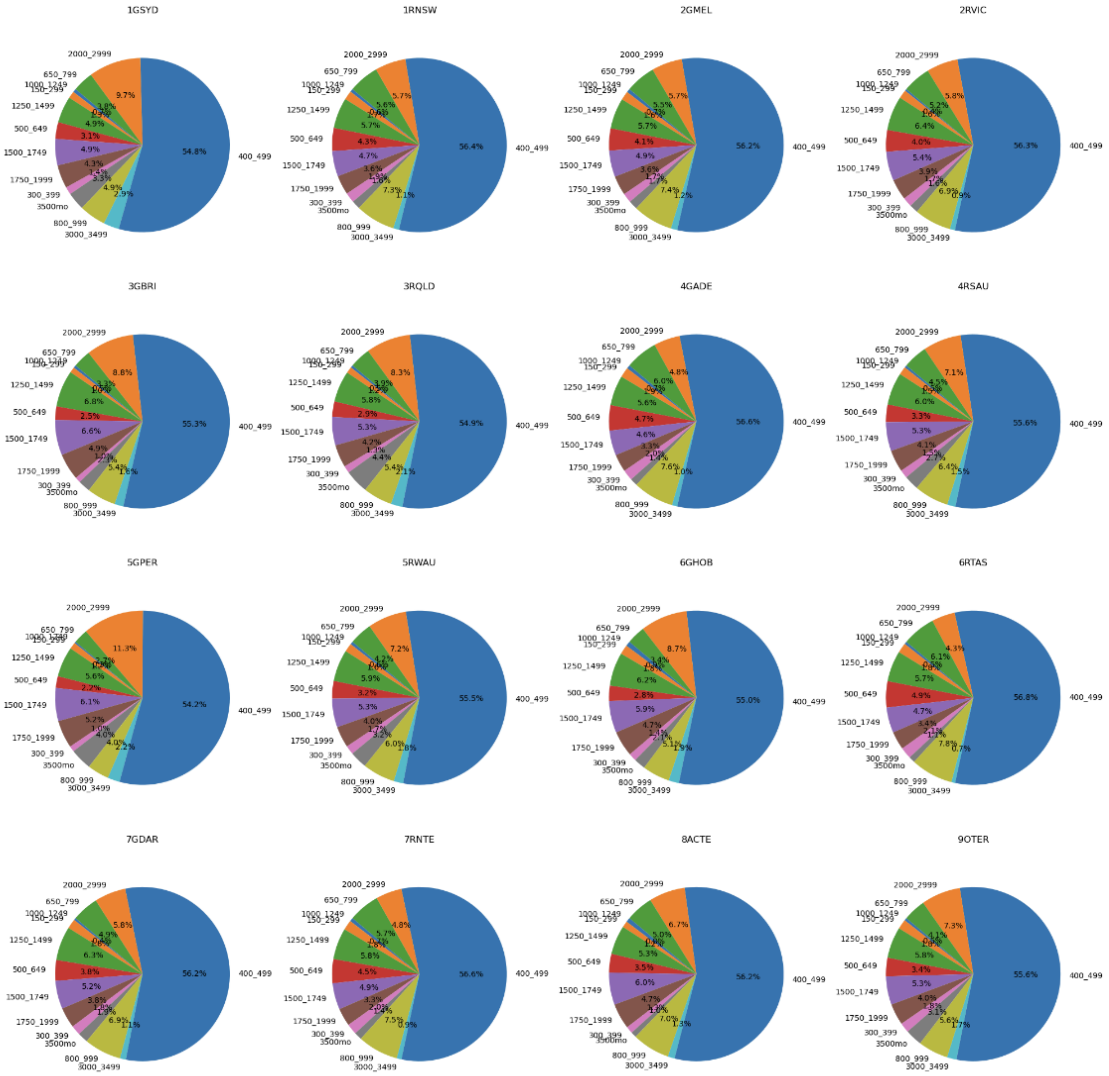

Figure 20. *pie chart for income in different areas*

Figure 19 and Figure 10 show the proportions of different income groups across various cities. According to the group bar chart and the pie chart, we can see that the majority of the population, around 54% to 57%, earns between 400 to 499 dollars per week which reflects the average income level in Australia. The income distribution across different regions is generally similar, with Queensland and Perth having a higher proportion of earning more than 3500 per week than other areas.

Figure 21. *heatmap for income and age in Melbourne*

Figure 21 is a heatmap that shows the relationship between income and age in Melbourne. Most people are between the ages of 25-64, which is the primary working age range. The highest income bracket, earning over 3500 per week, is mainly in people aged 25-34 and 35-44. These age groups have a wealth of experience and maturity, with ample energy to maintain high productivity levels. Among those aged 25-34, the most common income range is between 1000 and 1249.

4.4.3 Relation among Income, Sentiment and Area

4.4.3.1 Scenario Description

This scenario focuses on analysing the correlation between average income and average sentiments in 16 states and big cities in Australia.

4.4.3.2 The Reason Why We Choose This Scenario

People’s sentiment on tweets is an important part of our daily life. And people's moods can be influenced by various factors, with location and income being two significant ones. Therefore, we are exploring whether there is a correlation between people’s sentiments, their income levels, and their geographic locations.

4.4.3.3 Graphical Result

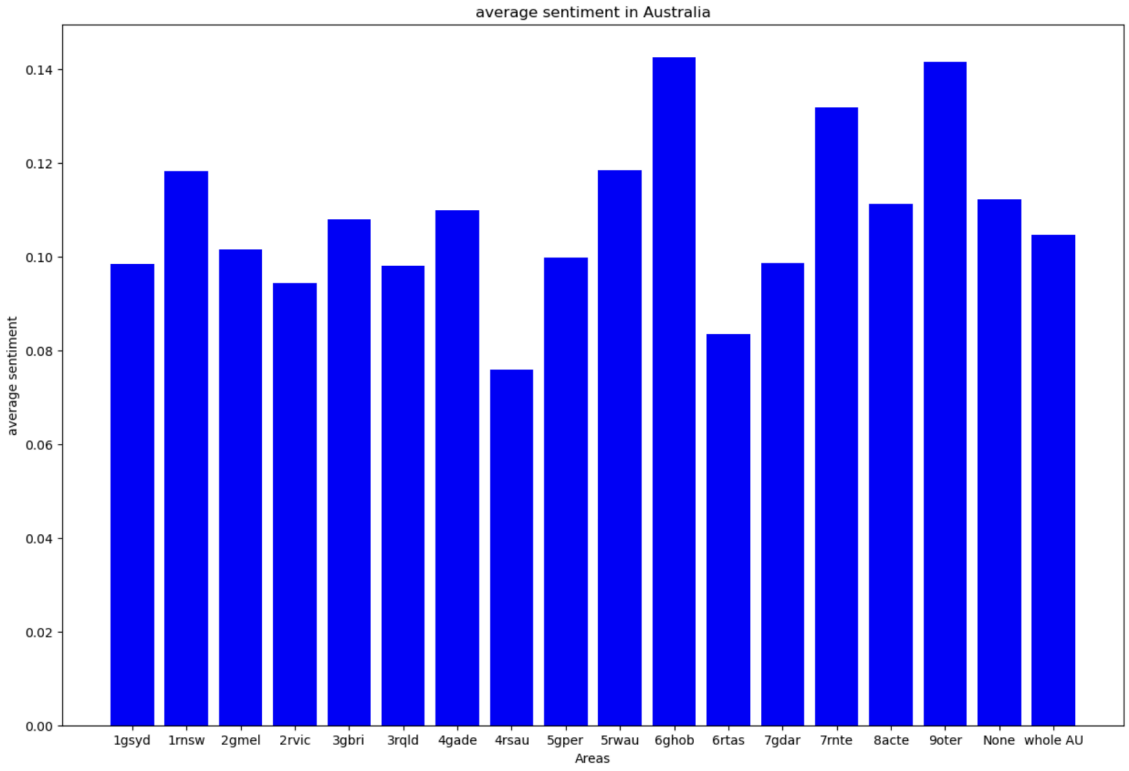

Figure 22. *Bar chart for average sentiment*

Figure 22 shows the average sentiment across various areas in Australia. Most areas have an average sentiment ranging between 0.08 and 0.12. The area "6GHOB" has the highest average sentiment, around 0.14. The area 4rsau has the lowest average sentiment, below 0.08.

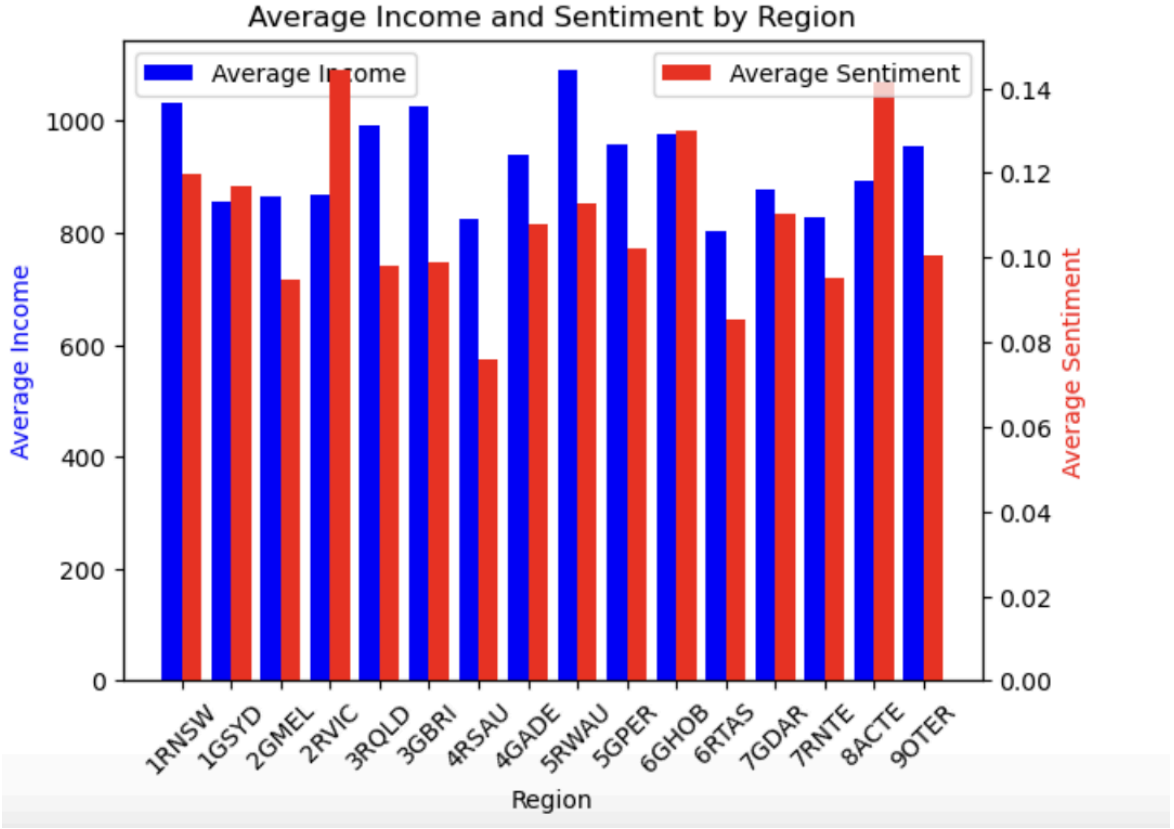

Figure 23. *bar chart for average sentiment and income*

Figure 23 shows the relationship between average income and average sentiment in various areas. From the chart, we can find that although higher income does not always lead to higher happiness, people with lower incomes tend to have a lower average sentiment. This indicates that while income does have some impact on people's mood, it is not the only or the most significant factor. Many other influencing factors also play a crucial role in determining people's sentiments.

4.4.4 Relation between the car crash and the light condition, speed respectively

4.4.4.1 Scenario Description

This scenario studies the surrounding conditions and the condition of the vehicles during car accidents.

4.4.4.2 The Reason Why We Choose This Scenario

There are various factors involved in the occurrence of car accidents, among which lighting and speed are very important. The severity of the accident is one of our primary concerns. Therefore, we studied the car accident data from Tasmania in the 10 years from 2010 to 2020 to analyse these three aspects.

4.4.4.3 Graphical Result

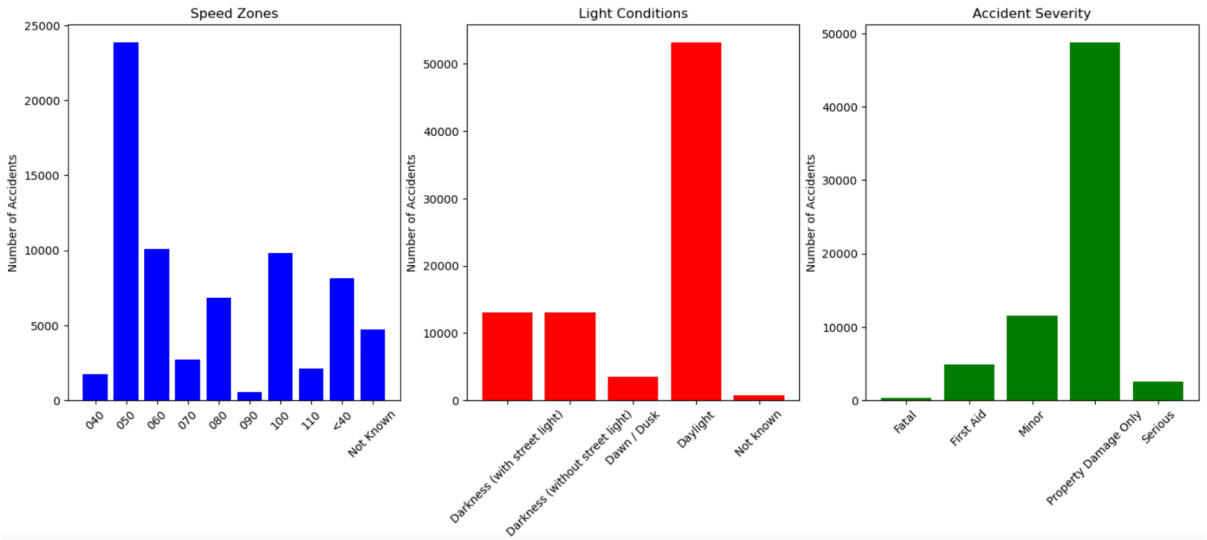

Figure 24. *bar chart for the crash index*

The three charts in Figure 24 show the speed of the vehicle, the severity of the car accident, and the lighting conditions at the time of the accident. From the charts, we can see that the probability of accidents is significantly higher when the vehicle speed is 50. This is likely because 50 is a common speed at which many vehicles travel.

The number of car accidents during the daytime is the highest, which is expected because people are most active during the day. Additionally, the number of accidents at night with and without street lighting is the same. This is because drivers tend to be more cautious in areas without street lights, and such areas generally have fewer pedestrians and vehicles. Most car accidents result in property damage. There is also a significant number of minor accidents, but the number of severe, first aid and fatal is still considerable.

5. Reflection

5.1 Issues and Challenges

5.1.1 Error Handling

5.1.1.1 Installation

Even after downloading, it still shows 'not found' (using the kubectl as an example):
execute the following command to obtain the installation path
asdf where kubectl
then manually add it to the $PATH.
source ~/.bash_profile

5.1.1.2 Backend Error

Timer
The weather data and air quality data are real-time data and they will generate a new piece of data every 10 minutes. This throws us an issue regarding how to upload the real-time data to Elasticsearch so that data won't be uploaded repeatedly. Our solution is to cut down the whole dataset obtained by the HTTPS GET method and only leave the latest one for uploading. In addition, we set the time intervals of timers so that they are the same as the time intervals of real-time data sources generating new data. This makes sure we only choose the latest piece of data every time timers call fission functions.

404 NOT FOUND
Check the logs and the process of creating the function

500 when using RESTful API
Not all Python objects can be transformed into JSON.

parse error: Invalid numeric literal when curl a function
Return data structure using jsonify or use 'curl "http://127.0.0.1:9090/temperature" ' (not use 'jq '.')'

Timeout of the functions
In the YAML file change the timeout of the functions (240) the only purpose of this it to give fission enough time to express what happens,

Function not work
General solution: add some senseful output; use logs to see what happened and locate the bugs.

5.1.1.3 Front-end Error

Time Zone Differences:
When using Fission to fetch real-time data from Elasticsearch, we noticed that the data timestamps were not up-to-date. Initially, we thought it was an issue with the API not pulling the latest data. However, we later discovered that while the timestamps in Kibana were aligned with the Melbourne timezone, they automatically converted to UTC standard time when fetched on the front end, resulting in a 10-hour difference from Melbourne time. Therefore, we adjusted the data timestamps to Melbourne time zone (UTC+10) on the front end.

5.1.2 Challenges Encountered

5.1.2.1 Data Challenges

1) Limited Data

We faced significant challenges due to the limited availability of relevant data. Many data in SUDO are not available to download, which means it will take plenty of time to decide on scenarios.

2) Data Quality

Ensuring the quality of the data was another major hurdle. The data collected from platforms often contained noise, inconsistencies, and missing values, which required extensive cleaning and preprocessing to make it suitable for analysis. Also, data from different sources have different formats, which makes it difficult to correlate them.

3) Massive Data Transformation

Transforming massive amounts of raw data into a structured and analysable format posed significant difficulties. This involved various steps including data extraction, transformation, and loading (ETL), which were time-consuming and computationally intensive.

Crash data as an example: It is a large dataset with over 70,000 items. Firstly we use fission to upload the JSON file to ES, which fails because of the limitation of the Fission function to upload data concurrently. More specifically, one HTTPS request cannot upload this huge amount of data. Our solution for this issue is to make use of the "Async" model to generate multiple HTTPS requests at the same time and every HTTPS request only needs to upload a limited number of data.

5.1.2.2 Technical Challenges

1) Database Management

Managing Elasticsearch databases in a scalable and efficient manner was crucial. This included ensuring high availability, fault tolerance, and optimal performance of the database clusters.

2) Fission

Using Fission for serverless computing introduced several challenges, primarily due to the limited availability of tutorials and documentation. This made it difficult to implement and troubleshoot serverless functions effectively.

3) RESTful API Integration

Integrating various components of the system through RESTful APIs required careful planning and implementation. Ensuring seamless communication between the data harvesting, processing, and visualisation components was essential for the overall functionality of the system.

5.2 Team Cooperation

5.2.1 Team Roles

Donghao Yang: Kubernetes cluster deployment, Elasticsearch index management, static data collection client deployment, real-time data collection client deployment, report writing, Gitlab repository management.

Ziqiang Li: Elasticsearch index management, static data collection client deployment, real-time data collection client deployment, report writing.

Rui Mao: Frontend data analysis, test, Elasticsearch index management, backend data processing, static data collection client deployment, external static data collection client deployment, real-time data collection client deployment, report writing, YouTube video recording.

Xiaxuan Du: Kubernetes cluster deployment, frontend data analysis, static data collection client deployment, external static data collection client deployment, report writing, YouTube video recording.

Ruoyu Lu: Kubernetes cluster deployment, Elasticsearch index management, real-time data collection client deployment, report writing, Gitlab repository management.

5.2.2 where the team worked well

The communication among team members is effective and efficient. Every team member makes a great contribution to the deliverable of this cloud system. The task allocation is reasonable as we assign every single task to more than one team member to complete collaboratively so that they can help each other out if there's any issue in the development.

5.2.3 where issues arose and how issues were addressed

All team members are struggling with the new technologies that were taught in this subject at the beginning of this project. Our solution to this issue is quite simple: just spend more time studying and researching the contents of this subject and assignment. Due to the procrastination of team members, we started the project relatively late and we didn't have enough time to complete some of the functionalities that we think are worthwhile to try. For example, integrate the Bert to analyse data from the Mastodon.

6. Discussion

Can your application and infrastructure dynamically scale out to meet demand?

The simple answer is yes! As our system is based on Kubernetes with features of horizontal pod autoscaler and cluster autoscaler, the orchestration capability of Kubernetes can scale the number of pod replicas in terms of CPU utilisation and scale up or down the number of nodes in the cluster should the storage capacity be insufficient or certain nodes are underutilised respectively. Our cloud system makes use of the serverless framework, namely Fission. Unlike other service frameworks, Fission functions will run when events come and will be "dead" and ready for being called by the next event. This feature makes Fission able to scale up and scale down in terms of system demand. Elasticsearch is the database system of our cloud service, which can add or remove nodes in terms of data loads. Once it adds or removes nodes, it will reallocate the data to ensure data integrity, availability and so on. However, the resources being allocated to our system by MRC(Melbourne Research Cloud) are limited. Thus, from this perspective, the scaling ability of our clouding service system is limited and cannot tackle more demands once beyond allocated resources.

7. Link

7.1 Video Link

https://youtu.be/qGX7yU5g2Lo

7.2 Gitlab link

https://gitlab.unimelb.edu.au/comp90024-group11/comp90024-a2

Reference

[1] SUDO(Spatial Urban Data Observatory): https://sudo.eresearch.unimelb.edu.au/
BoM: https://reg.bom.gov.au/vic/observations/

[2] “Air quality observations,” May 21, 2024. https://data.ballarat.vic.gov.au/explore/dataset/air-quality-observations/api/?disjunctive.location_description

[3] “Air Quality Observations - Victorian Government Data Directory.” https://discover.data.vic.gov.au/dataset/air-quality-observations

[4] Bureau of Meteorology, “Latest weather observations Ballarat.” https://reg.bom.gov.au/products/IDV60801/IDV60801.94852.shtml

[5] “API documentation – Australian Digital Observatory.” https://www.ado.eresearch.unimelb.edu.au/api-documentation/

[6] GCCSA: https://www.abs.gov.au/ausstats/abs@.nsf/Lookup/by%20Subject/1270.0.55.001~July%202016~Main%20Features~Greater%20Capital%20City%20Statistical%20Areas%20(GCCSA)~10003