COMP90024 Cluster and Cloud Computing | Assignment 2 | Big Data Analytics on the Cloud

Team 53

Niket Singla

1288512

nsingla@student.unimelb.edu.au

Patipan Rochanapon

1117537

prochanapon@student.unimelb.edu.au

Parsa Babadi Noroozi

1271605

pbabadinoroo@student.unimelb.edu.au

Liam Brennan

1269948

brennanlm@student.unimelb.edu.au

Jason Phan

1180106

phanjp@student.unimelb.edu.au

Keywords Cloud computing · Serverless · Kubernetes (K8s) · Fission functions · Elasticsearch · Kibana · Jupyter notebook

Contents

1 Introduction

The immense scale of data available online and the ease of internet connectivity in today's world enables millions of users to access applications, databases and websites at every given second. This calls for a complex and scalable infrastructure that can handle the multitude of users that are using them. Therefore, this is the reason behind cluster and cloud computing being fundamental to the operation of most large-scale software systems. Hence, understanding the end-to-end operations of cluster and cloud computing systems has become vital for developing any large-scale applications.

The purpose of this report is to describe the solution developed by our group for Assignment 2 of the COMP90024 Cluster and Cloud Computing subject at the University of Melbourne. In this report, we highlight the various aspects of our cloud-based solution, ranging from its system architecture and infrastructure to the datasets and tools used to analyse the chosen scenarios.

We explored and analysed the following three scenarios

- Crashes and alcohol consumption
- Air Quality, temperature and fertility and mortality rates
- Twitter counts, sentiments and average age, income, population and education data

In one scenario we investigate potential relationships between the number of crashes involving injuries or fatalities and common health risk factors (such as alcohol consumption rates) in Victoria's SA2 districts. The health risks dataset is collected from the Spatial Urban Data Observatory (SUDO) and the Victorian road crash dataset is provided by the Department of Transport and Planning, 2024 website. Another scenario explores the relationship between temperature levels, fertility and mortality data for each suburb. For this scenario, temperature data is collected from the Environment Protection Authority (Environment Protection Authority Victoria, EPA 2024) API, and weather data is collected from Bureau of Meteorology, BOM 2024 along with fertility and mortality data from SUDO. Finally, in the third scenario, we investigate potential correlations between sentiments from a Twitter dataset and the average age and income of SA3 districts. This is accompanied by an analysis of the total number of tweets against the population density of the corresponding SA3 districts. The dataset for population attributes such as age, income and density is supplied by SUDO.

In the following sections, we outline the tools we used and their respective roles in our cloud-based solution. This includes Kubernetes for managing our cloud infrastructure and Elasticsearch with Kibana for managing our data. Fission enables the development of our serverless functions and finally, Jupyter Notebook is utilised for our data analysis and visualisations.

2 System Architecture & Design

2.1 System overview

In this section, we provide an overview of the infrastructure and functional aspects of the implementation. Figure 1 represents the high-level overview of our system architecture, illustrating the various components of our infrastructure, their interactions and their dependencies. We will be discussing our implementation, such as hardware, libraries, installed applications and their interactions in detail in later sections.

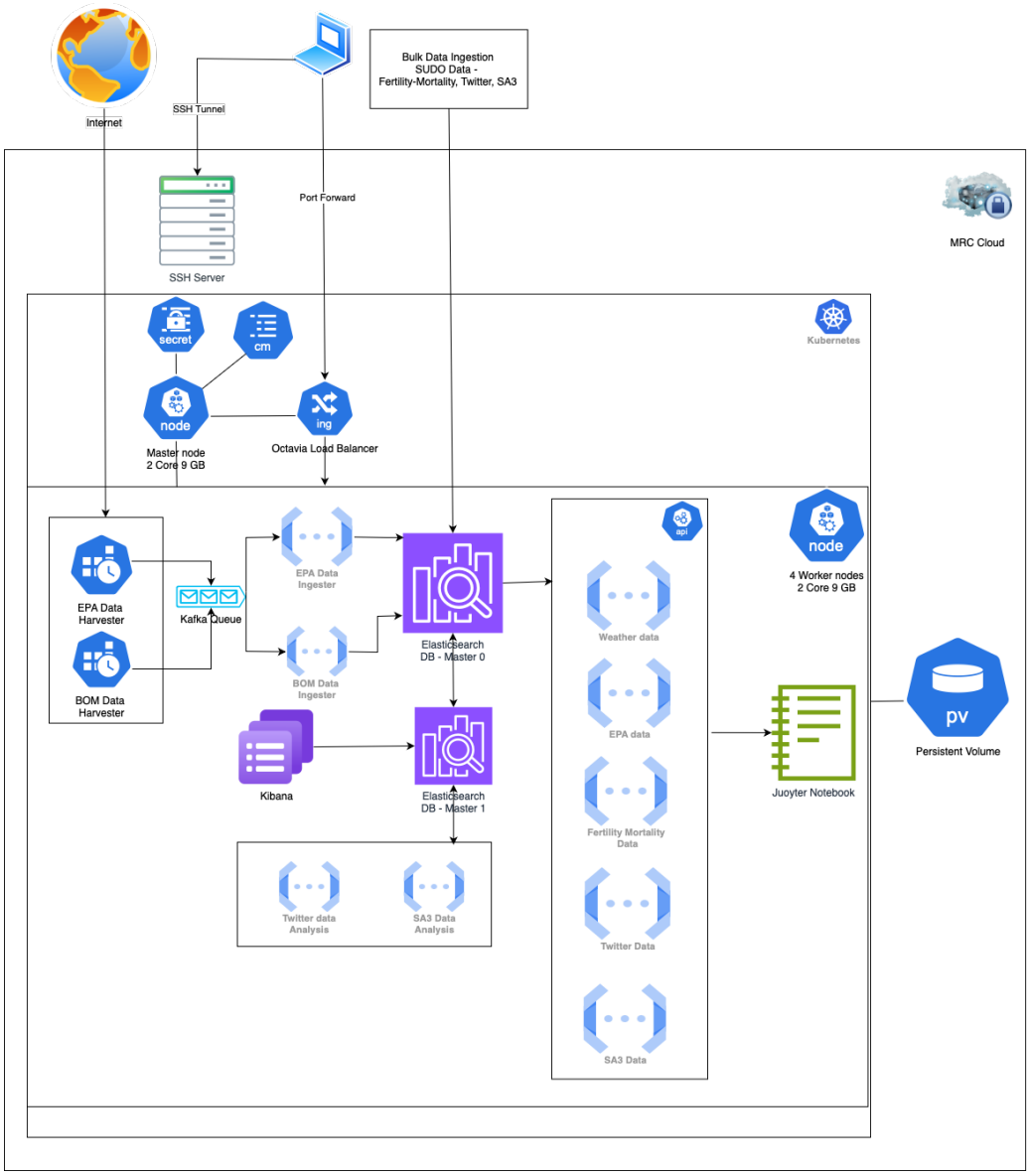


Figure 1: System Architecture

Following are the major components of our projects

- Cloud infrastructure, including necessary RAM, storage and processing capacity
- Kubernetes (K8s)
- Kafka
- Fission
- Elasticsearch & Kibana
- Jupyter notebook

The project's objective was to extract data from external sources, store it in cloud-based persistent storage, provide a RESTful API for data retrieval, and utilize Jupyter Notebook for visualization and interaction. To achieve this, we set up a Kubernetes (K8) cluster to run multiple pods for the harvester functions, database, API hosting, and Jupyter Notebook.

We developed serverless functions using Fission to harvest data from external providers such as the Bureau of Meteorology and the Environment Protection Authority, Victoria. This harvested data was then added to Kafka, which was integrated to establish an event-driven architecture and ensure data security and loss prevention. Subsequently, additional serverless functions subscribed to the Kafka queue and stored the data in Elasticsearch. Since the data providers such as SUDO, Department of Transport and Planning and Twitter do not provide an API for data collection, these datasets were added manually. However, we have additional serverless functions to pre-process the data retrieved from these sources. To retrieve the data from the server via stateless API, we created Fission functions that act as an API and this data was retrieved by the Jupyter Notebook installed on the server.

2.2 Architectural Components

2.2.1 Cloud Infrastructure

Cloud infrastructure allows on-demand provisioning of resources to cater to the requests of internet users. Overall cloud infrastructure capacity can then be further subdivided, which allocates the necessary virtual computing power and storage to the hosted resources. Infrastructure resources provided are to be allocated across all workloads as necessary, thus implementing a fault-tolerant, efficient and performant system. For this project, we were allocated 6 compute instances, 11 VCPUs, Unlimited RAM, and 700GB volume storage and below are the details of how they were used.

- 6 compute instances
  - 4 out of the 5 compute instances hosts 3 nodes for the Kubernetes (k8) cluster
  - 1 master node for the k8 cluster
  - 1 bastion server to facilitate connectivity to the cloud.
- 11 VCPUs

  - Each of 5 nodes in the k8 cluster utilises 2 cores each.
  - Bastion server is deployed with pre-defined 1 core flavor.
- ≈ 50 GB RAM
  - Each k8 node is deployed using the 2 cores 9 GB RAM flavour
  - Bastion node is deployed with 1 core 4GB RAM
- 300 GB Storage

2.2.2 Kubernetes

Kubernetes, also known as K8s, is an open-source system for automating containerised application deployment, scaling, and management. Fig 1 shows how our system is a Kubernetes-managed containerised cluster. Our cluster has one master node and four worker nodes each with the same configuration. A master node in Kubernetes is responsible for managing the entire cluster. It keeps track of the resource utilisation within each node so that the pods created within the nodes have sufficient resources to carry out their task. Each node in the cluster is running multiple containers responsible for hosting the Elaticsearch, Jupyter hub, and Fission functions as shown in Fig 1. Our K8 cluster faces the risk of a single point of failure due to operating with only one master node. Assessing the likelihood of failure as low and given the performance requirements, we have decided to keep a single node.

2.2.3 Fission

Fission is a framework for serverless functions on Kubernetes. Fission allows developers to focus on code without worrying about the underlying infrastructure. They provide a fast cold start and can auto-scale with incoming requests and server load. In our project, we have utilised the Fission function to write the data harvester functions to retrieve and parse data from external data providers such as the Bureau of Meteorology (BOM) and Environment Protection Authority (EPA) Victoria. Fission functions are also used to create the timer triggers to run on schedule for data harvesting and analysing the existing data. Once the data is available, Fission functions are deployed as a RESTful API to serve data to our Jupyter Notebook.

2.2.4 Kafka

Apache Kafka is an open-source distributed event streaming platform. In our project, we have used the Kafka queue to stream incoming data from external data providers. Kafka allowed us to limit the functions that change system state (e.g., performing CRUD operations on a database). It also allowed us to make our system more fault-tolerant by storing events as long as necessary, should the ingestion function fail to ingest the data.

2.2.5 Elasticsearch

Elasticsearch is a distributed, RESTful search and analytics engine that centrally stores data for fast search, fine-tuned relevancy, and powerful analytics that scales with ease. We have used Elasticsearch version 8 to store our data as documents. Further details about the Elasticsearch and Kibana are discussed in the section 4

2.3 Scaling

Scaling is the process of adding more resources to the application to fulfil the user request promptly and without issues. In our application, we are utilising the Kubernetes cluster as the container management system and Fission to create serverless functions - both of which provide scaling capabilities. Scaling can be achieved in a cloud-native application with two different approaches

- Autoscaling
- Manual Scaling

Autoscaling - Both K8s and Fission support autoscaling by using the Horizontal Pod Autoscaler service. However, this requires the installation of the metric service to be able to monitor the pods' CPU and memory usage. This is then used by the Autoscaler to decide when to add new pods to handle the load spike. In our application, we don't expect a sudden increase in the load and hence this option was not explored any further.

Manual Scaling - We created the K8s cluster using the OpenStack template which provides a *node_count* attribute to control the number of nodes in the cluster. We initially created the cluster using the provided template to have 3 worker nodes and 1 master node but, based on the volume of data ingested and large application load, we added one more node to the cluster using the below command.

```
openstack coe cluster update elastic replace node_count=4
```

Fission functions come with built-in auto-scaling capabilities, allowing them to adjust the number of pods on demand. By default, Fission sets the pod pool size to 3 but, in our configuration, we've increased the pool size to 4 to match the number of nodes, ensuring high availability.

```
fission env update --name python --poolsize=4
```

By default, Fission utilises the pool manager to manage specialised pods that execute functions effectively. This setup also facilitates warm starts, optimising response times. If a pod remains idle for a defined period, it's automatically reclaimed and respawned upon receiving a new request. In our application, data

harvesters run at 30 and 60-minute intervals, processing roughly 90 data points into the database each time. Analysis of pod usage, as depicted in Figure 2, indicates low utilisation levels. Therefore, further customisation for auto-scaling may not be necessary at this time.

```
niketsingla@ravpn-266-2-student-10-8-64-64 COMP90024-Assignment2 % kubectl top pod
NAME                                          CPU(cores)   MEMORY(bytes)
nodejs-386341-6fcd5ff65b-7c9kp                1m           14Mi
poolmgr-nodejs-default-386341-86cbff87d5-btk9q   1m           36Mi
poolmgr-nodejs-default-386341-86cbff87d5-j7kth   1m           36Mi
poolmgr-nodejs-default-386341-86cbff87d5-pqx55   1m           36Mi
poolmgr-python-default-23807-598d8749c8-42r8l    6m           46Mi
poolmgr-python-default-23807-598d8749c8-4xff2    1m           35Mi
poolmgr-python-default-23807-598d8749c8-5mqg2    1m           35Mi
poolmgr-python-default-23807-598d8749c8-6xjbz    1m           36Mi
poolmgr-python-default-23807-598d8749c8-7j4kw    1m           35Mi
poolmgr-python-default-23807-598d8749c8-7qn9k    1m           35Mi
poolmgr-python-default-23807-598d8749c8-8m2lg    2m           36Mi
poolmgr-python-default-23807-598d8749c8-9265q    1m           35Mi
poolmgr-python-default-23807-598d8749c8-bkww6    1m           35Mi
poolmgr-python-default-23807-598d8749c8-bspd4    1m           35Mi
poolmgr-python-default-23807-598d8749c8-c6p4x    1m           35Mi
poolmgr-python-default-23807-598d8749c8-c6zj9    1m           36Mi
poolmgr-python-default-23807-598d8749c8-cm2rl    1m           36Mi
poolmgr-python-default-23807-598d8749c8-czc5z    1m           35Mi
poolmgr-python-default-23807-598d8749c8-dcnm9    1m           56Mi
poolmgr-python-default-23807-598d8749c8-gkqhx    1m           35Mi
poolmgr-python-default-23807-598d8749c8-gwhwp    1m           35Mi
```

Figure 2: Fission pod usage

2.4 Security

The University of Melbourne's MRC is a private cloud based on OpenStack architecture. As recommended, we have used the *qh2-uom-internal* network. However, using this network, our VMs are only accessible from within the University network or over Unimelb VPN. This offers greater security over the default public IP option because the instances are inaccessible elsewhere.

Additionally, we have created a security group that only allows ingress on port 22 to access the Kubernetes cluster deployed in the cloud via SSH securing the access via a public-private key pair. We have created a public and private key pair to authenticate the users accessing the cloud instances. All users must register their public key with the cloud and provide their private key to authenticate and connect over the Secure shell protocol (SSH). Furthermore, all the Kubernetes nodes are added to the same security group to restrict their access from the outside world and can only be accessed once the user is connected to the MRC cloud over SSH.

Besides, each request to the MRC passes through a series of firewalls to restrict unauthenticated and unauthorised access, and each authenticated request is logged and monitored to enhance the security of our cloud instances.

2.5 Fault tolerance

Fault tolerance is the ability of the system to continually operate even in case of multiple failure events. We will discuss what challenges it presents in our application and how it was addressed.

- **K8s worker node failure** - Kubernetes is a self-healing system. The master node in the Kubernetes cluster monitors each worker node and if any of the nodes fails, it will create a new node and add to the existing cluster. The likelihood of experiencing an all-node failure is extremely low and hence no further strategies were explored.
- **K8s master node failure** - A Kubernetes master node failure can have unwanted results. The cluster will continue to serve the requests if the master node fails, however, all the nodes will act as individual servers. If other nodes fails, they will not be recreated as the master node is unavailable. Further failure of the K8s API will result in the unavailability of the *kubectl* commands.
  To recover from a master node failure, we would need to recreate the master node and attach all other nodes to the master node to achieve a cluster. Even though the master node failure can have an undesirable impact, we have decided to keep a single master node to allow more worker nodes in the system. In the production systems, it is not recommended to have a single master node.
- **Data loss** - We have deployed *Elasticsearch* as the database for our application on the Kubernetes cluster. In our installation, we have specified 2 replicas for Elasticsearch i.e. there will be two instances of the same database that will be deployed and the data will be replicated across. Since Elasticsearch is deployed in the Kubernetes cluster, the system is intelligent enough to spread the replicas across different nodes to ensure one of the instances is always available. In addition, the new instances can be reattached to the same persistent volume without experiencing any data loss.

2.6 Data Error Handling

2.6.1 Twitter Error Handling

When conducting map visualizations to analyze bounding boxes (bbox) in the Twitter dataset, we discovered that the provided bbox might inaccurately represent the area when viewed on a map, hindering joins with SA3 data. This discrepancy arose due to the bbox area varying widely in size across different regions, which presents challenges for accurately joining the data, especially when very large bboxes overlap with each other. To address this issue, we refine the process by calculating the centre coordinates (latitude, longitude) of each bbox for subsequent analysis. By choosing a larger geospatial area statistic like Statistical Area Level 3 (SA3)[1], which is suitable for accommodating the varied bbox sizes, we can

[1]SA3s create a standard framework for the analysis of ABS data at the regional level through clustering groups of SA2s that have similar regional characteristics

enhance the accuracy and effectiveness of our mapping efforts. To handle the large volume of queries, despite reducing the unique coordinates to 30,354 by using parent-child relations, pagination is essential due to Elasticsearch's query limit. Additionally, the challenges we encountered in ingesting the large files are discussed further in section 5.4.

2.6.2 API Error Handling

The API endpoints exposed for the crashes and health risks datasets have been constructed to allow for a large degree of flexibility in the kinds of analysis they make possible. Although this has been achieved with a relatively lightweight implementation, such a customisable system demands some attention to the management of errors resulting from bad query requests. For instance, when requesting a random sample of crash documents, limits imposed by Elasticsearch require that at most 10,000 documents can be returned at once. To increase the usability of the endpoint, exceeding this limit forwards an appropriate error to the client with the correct HTTP error code and an explanation of why the query failed. We also ensure that understandable error messages are returned from the joined analysis endpoint for situations such as when an unknown grouping field is supplied or when an aggregation method provided is not in the allowed list. These checks are also essential to sanitise user-controlled input to prevent possible injection scenarios, whether deliberate or not.

3 Cloud Infrastructure

Cloud computing is the on-demand availability of computing resources such as storage, compute from a shared pool of resources. It enables the consumers to rapidly provision the resources as needed without having to acquire or manage the underlying hardware and pay as per their usage.

3.1 Melbourne Research Cloud (MRC)

The Melbourne Research Cloud is the University of Melbourne's own private cloud environment which offers free cloud resources much like the commercial cloud providers such as AWS, and Google to students and other affiliated institutes. The MRC offers a range of virtual machine sizes to its users with a total capacity of almost 20,000 virtual cores. It also includes specialist resources such as GPGPUs, private networking, load balancing, and DNS.

The MRC is based on the *OpenStack* which offers free and open-source software platform for cloud computing for (mostly) IaaS. It is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a data centre, all managed and provisioned through APIs with common authentication mechanisms. OpenStack also offers a dashboard for the cloud administrator for easy provision and management of the resources via the web interface. OpenStack is broken up into services to allow the users to plug and play components depending on their needs. In this project, we used the following components to provision the services.

Component	**Services**
OpenStack-helm	Lifecycle management and deployment
Keystone	Identity
Octavia	Load Balancing
Cinder	Volume Storage
Magnum	Workload provisioning

Table 1: Open Stack components and Services

3.1.1 OpenStack Template and MRC Offering

To provision the resources in the cloud, we defined a cluster template using the OpenStack format and available attributes to customise the resources. Following are some of the key attributes we used in the template and how they are connected with the MRC.

- **Flavor** - The flavor attribute in the OpenStack template is used to define compute, storage and memory of Nova computing instances. University has a predefined list of flavors to choose from and as a part of this project, we used the *uom.mse.2c9g*. This flavor offers 2VCPUs, 9 GB memory and 30 GB of root disk storage. By default, MRC offers up to 30GB of root disk storage. Additional storage can be mounted to the instances. We have been provided with 700GB of storage for this project.
- **Image** - An image is a file used to populate a blank instance with an operating system; depending on the image, additional software applications may be included. MRC provides a variety of official images covering the most common Linux distributions, as well as Microsoft Windows. In this project, we used the *fedora-coreos-37*.
- **COE** - Specifies the Container Orchestration Engine to use. Supported COE is *kubernetes*.
- **Keypair** - Secure Shell (SSH) is a protocol primarily used to permit secure remote command line logins to servers. MRC requires the public-private keypair to be used to access the cloud resources. Keypair adds an extra authentication factor, and it permits logging in without a password. Keypair can be generated from a local OpenSSH library on the laptop or from the MRC dashboard.

3.1.2 Strengths and Weaknesses

The Melbourne Research Cloud (MRC) primarily operates as an Infrastructure-as-a-Service (IaaS) platform, offering fundamental IT infrastructure components such as computing resources, networking facilities, and storage solutions. This IaaS model holds significant value for research endeavours. Conventional computer procurement processes entail considerable time investment, and the acquired hardware typically possesses a lifespan of three to four years, potentially extending beyond the duration of a specific research project. In contrast, the Melbourne Research Cloud offers swift provisioning of computing resources, enabling users to deploy virtual machines in minutes. Moreover, upon project completion or when resources are no longer required, users can promptly release these resources without managing physical hardware disposal arrangements. Following are the key advantages of the MRC

- **Choice of hardware and operating system** - MRC offers a range of compute engines suitable for small to large-scale projects. These resources can be provisioned and released on demand. MRC also offers all types of operating system images to their users.
- **Operating cost** - MRC offers its resources to the students, researchers and affiliated institutes for free of cost. Researchers can have the required resources allocated to them for the timespan of the project to conduct their research without incurring any additional cost while the same resources from a public cloud provider such as AWS, or Google could cost millions of dollars.
- **Efficiency** - MRC empowers the users by allowing them to spin up new resources in a matter of minutes. This allowed complex applications to be built with very low startup costs keeping the research cost low as well.

- **Digital obsolescence** - It is the risk of data loss due to the incompatible format as the hardware and software are continually being replaced by newer versions. Researchers can build their system using the MRC without carrying this risk as the digital assets are continually upgraded by the MRC team to mitigate this risk

While the MRC is enormously advantageous to the researchers, it does have some limitations as well

- **Performance** - The MRC is based on the OpenStack architecture. While OpenStack offers convenience and high availability it requires complex configuration and can experience performance issues at times. OpenStack is not as advanced as other industry cloud providers such as AWS, Google or Azure but the additional sophistication comes at a cost.
- **Limited PaaS and Saas Services** - The MRC is largely IaaS and offers very limited PaaS and SaaS services which results in the users performing the system administration themselves. Performing all the tasks of the system administration adds an overhead to the users. The users who can outsource the program installation and system administration and are willing to have their job wait in the queue can consider using HPC, Spartan.
- **Single Availability Zone** - The MRC data centre is physically located in Melbourne and has a single availability zone. While the likelihood of the data centre being unavailable is low, it still does not safeguard the customers from an unfortunate situation such as power outages or extreme weather events causing the services to go down. In such circumstances, it is not suitable for production load or time-critical workload.

Differences from HPC - While both the MRC and HPC are suitable for performing large-scale data analysis, there can be instances where the users must prefer HPC over the MRC.

- HPC is a Linux-based system, If the user does not have a preferred choice of Operating system to run their program.
- The program does not require a user interface to run.
- While the MRC has a large number of resources it may still struggle to allocate the resources at time. If the user requires large compute and memory resources then HPC is more suitable.
- Users are willing to delegate the program installation and system administration.
- The program is not time-critical and ok to wait in a queue to have necessary resources allocated.

4 Elasticsearch & Kibana

Elasticsearch functions as a decentralised document repository tailored for high-performance search and analytics. Rather than organising data like a traditional SQL database in rows and columns, Elasticsearch uses a schema-flexible, JSON-based document model for data storage. An Elasticsearch cluster consists of multiple nodes that work together to distribute and store these documents, ensuring high availability and fault tolerance. This architecture enables efficient data retrieval and powerful, large-scale analysis capabilities which were essential requirements for the scale of data storage and manipulation used in this project.

4.1 Indexing

In Elasticsearch, an index serves as an optimised aggregation of documents, with each document comprising multiple fields representing key-value pairs that encapsulate the data. By default, when data is stored in Elasticsearch, documents undergo the indexing process and become fully searchable in near real-time, typically within 1 second.

4.2 Index Mapping

Elasticsearch uses what are called 'mappings' to define how each field of the documents within a database's indexes are stored and indexed. Mappings can either be specified explicitly, where fields are manually given a data type when the index is created, or dynamically, where Elasticsearch automatically determines which data types are best suited for each field based on its contents. This can impact the space efficiency of the stored field data as well as determining which kinds of querying operations are possible. For instance, a document with a string field could be mapped as a `text` type for full-text search or as a `keyword` type to support sorting and aggregation queries. We have explored both the options in our project for the mapping of data types. We have made sure the geo spatial data have the *geo_shape* or the *geo_point* based on the data to allow for the complex geo spatial operations such as *nearest_distance*.

4.3 Ingest Pipelines

As part of the ingestion process for the Victoria Road Crash Data from the Department of Transport, an Elasticsearch ingest pipeline was created to transform the raw temporal fields of each crash instance into a date-time format that is indexable by Elasticsearch. The pipeline is configured to run a simple Painless processor script on each uploaded document before its addition to the crash data index. This architecture was chosen to take advantage of the flexibility of the Elasticsearch ingest pipeline system which can be horizontally scaled as throughput requirements change and avoids the need for external

dataset transformation steps prior to ingestion. By keeping these transformation steps within Elasticsearch, the understandability and repeatability of our data preprocessing is greatly enhanced.

4.4 Sharding & Replication

Sharding is one of the key performance enchantments provided by Elasticsearch. Each index can be divided into smaller segments, known as shards, which are then spread across multiple nodes enabling distributed, parallel computation of query results. When a client makes a query to Elasticsearch, it is broadcast by a coordinating node to every other node in the cluster containing a shard of the specified index. There, the query is executed locally on each subset of the data and the results are collected and sent back to the coordinating node to be returned to the client. This parallelisation greatly reduces the query response time as each shard need only process a portion of the data enabling Elasticsearch to handle larger volumes of data and queries more efficiently.

Further, an index can be configured to have two kinds of shards: primaries and replicas. Primary shards serve as the main data holders for an index while replica shards are redundant copies of the data to protect against failures and increase capacity to serve read requests. This architecture is highly fault-tolerant and resilient to hardware issues where even the loss of a node containing a primary shard can be recovered from using a failover to a currently in-sync replica as a new primary. Replica shards also play a crucial role in performance as read queries can be served by both primary and replica shards. This can distribute the load more evenly across a cluster, preventing any single node from becoming a bottleneck. Additional flexibility is possible by reconfiguring the number of replica shards even after index creation, which is not possible for the number of primary shards.

4.5 Querying & Analysis

Elasticsearch provides a powerful and flexible querying system that enables searching and aggregation to be performed on vast quantities of data in a resource and time-efficient manner. Search types such as term, match and range queries can all be combined to find documents that meet specific requirements through the provided JSON-based query domain-specific language. Especially relevant to our project, Elasticsearch also supports many varieties of geospatial operations including bounding-box, intersection and distance queries between different geospatial datatypes such as points and polygons.

Aggregations are an important part of big-data analysis as important insights often can only be extracted by considering the context of an entire dataset. Elasticsearch enables this with the support of various summaries and statistics, including metrics such as averages, sums, and counts, as well as more complex operations like histograms and nested aggregations. An important application of these abilities for our project was for joining geospatial datasets by intersection. We used this in scenarios where a dataset of points needed to be linked to a dataset of regions overlapping with those points. Using geospatial

aggregation queries, it was possible to efficiently perform joins between datasets with the results being written to a new index.

Following these joining operations, the crashes and Twitter datasets contain such a large number of documents that aggregations become the only way to perform further analysis effectively. Commonly, we would perform grouping aggregations by region (either SA2 or SA3 district) to maintain some sense of geographic context while allowing comparison against other region-associated variables. Additional metric aggregations, such as averaging the number of serious injuries in each crash, were also used to gain a further understanding of dataset trends and relationships. Combining and nesting these aggregations was often a powerful tool for extracting insights.

4.6 Kibana

Kibana is a powerful data visualization and exploration platform for Elasticsearch. It allows users to create interactive visualizations and dashboards, and perform searches to analyze the data stored in Elasticsearch indices.

4.6.1 Kibana Maps

Kibana Maps is a feature in Kibana that lets users visualize and analyze geospatial data stored in Elasticsearch, allowing users to explore location-based data easily. Moreover, it allows direct upload of GeoJSON and shape files which simplifies the process of incorporating external geographic data without the need for scripting or complex ingestion procedures. Figure 3 illustrates a basic intermediate data analysis, revealing SA3 regions where the weekly average income surpasses $1000. It also showcases the tool's capability to directly upload shape and GeoJSON files.

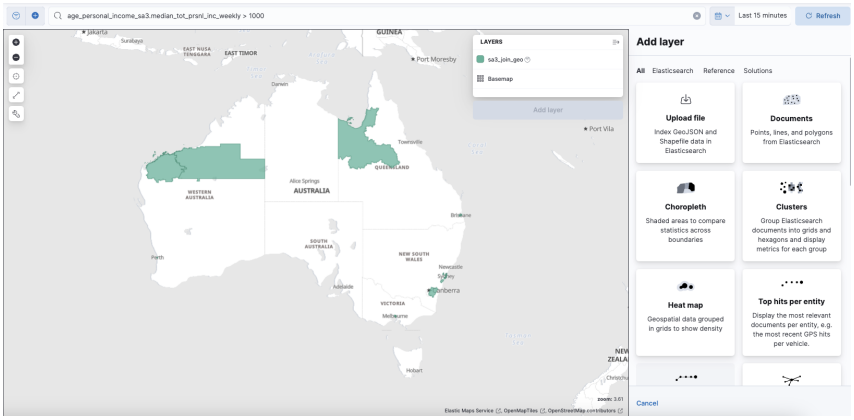


Figure 3: Exploring Geospatial Data with Kibana Maps

4.6.2 Kibana Index Management

Kibana Index Management simplifies the administration of Elasticsearch indices. It allows users to inspect mappings, check shard and replica configurations, and delete indices conveniently from the user interface.

4.6.3 Kibana Dev Tool

During the development phase, instead of executing curl commands in the terminal, Kibana Dev Tools offers a more intuitive approach for testing Elasticsearch queries. With its user-friendly interface, developers can seamlessly craft, execute, and debug queries without the need to manually construct curl commands. This convenience simplifies the process of exploring and experimenting with Elasticsearch's Query DSL. Moreover, the ability to reuse Query DSL tested in Kibana Dev Tools makes it incredibly convenient to transfer these queries to Python scripts. This seamless transition enhances the workflow and ensures consistency between development environments, allowing developers to leverage their tested queries effectively in Python code.

5 Data Collection

The scope of this project was to design a serverless application that can collect data from multiple sources of our choice, perform data analysis and let users interact with the application through plots and charts. We have employed various methods to ingest suitable data into the Elasticsearch such as live data ingestion via API, bulk ingestion API, streaming file ingestion, Kibana GUI. In this section, we shall discuss the data sources from where the data is collected and the methods used to ingest the data.

5.1 Data Collection via API

5.1.1 Environment Protection Authority Victoria (EPA)

As the population in Victoria increases, so does air pollution. We wanted to assess the impact of air quality on life in Environment Protection Authority Victoria, EPA 2024. EPA provides a convenient set of APIs to extract the environment monitoring data from their monitoring sites. EPA currently does not support water quality data extraction, so the scope was limited to air quality monitoring. Since there are roughly 90 sites in Victoria, we extracted all scientific parameters from all the monitoring sites to ensure the data availability during analysis.

The EPA website provides complete information on the required and optional header attributes to send the request payload. However, we observed that our request was still failing. The root cause of the issue was the missing *user-agent* header attribute, which was not specified in the document but was necessary. Once we added the header, we could successfully extract the required data. While the EPA API is protected by *Rate-limiting*, we did not encounter this restriction or had to come up with alternate ways to avoid this restriction since the data is refreshed once an hour.

5.1.2 Bureau of Meteorology (BOM)

While the Bureau of Meteorology does not provide a set of free APIs to extract the data, they upload the latest weather observations every 30 min as a JSON file on their website Bureau of Meteorology, BOM 2024. We have developed a harvester function that downloads the JSON data from the BOM website based on the static list of site IDs provided. BOM data provides the weather attributes such as air temperature, humidity, and rain for each of their site. We decided to ingest all the data to ensure its completeness during the analysis.

Since the BOM does not provide an API to extract the data, we had to manually look for all the sites, collate a list of all site IDs in Victoria, and store it statically in our program. The shortcoming of this approach is that should a new site be added in the future, we will need to update the program manually; otherwise, we will not be able to extract the data for the new site.

5.1.3 Data Ingestion

We used the Kafka queues to ingest the data from the API into Elasticsearch. We introduced an event-driven architecture to limit the number of functions changing the application's state. The functions interacting with external data sources will publish the data to the relevant Kafka queue, and *data-ingestion* function will subscribe to the event of interest. Publishing the data to the Kafka queue generates an event that triggers the data ingestion function to read the message and ingest the published data into Elasticsearch. We did not employ a batching strategy to limit the number of messages published to Kafka as the amount of data published is predictably small and at a constant interval.

5.2 SUDO Datasets

The Spatial Urban Data Observatory (SUDO) offers a robust data environment, building upon the Melbourne eResearch Group team's established open-source datasets and tools from the past decade. It will also integrate new datasets and capabilities, such as national social media data from the ARDC-funded Australian Data Observatory project (ADO), national bushfire data commons datasets from the ARDC bushfires initiative, and national air quality datasets from official agencies and citizen science endeavours Conference, eResearch Australasia, n.d. This presentation will provide an overview of SUDO and its support for the broader research community.

5.2.1 SA3 Population, Highest Education, and Average Age & Income

The SA3 Population dataset provides demographic information such as total population counts, age distribution, and gender distribution within Statistical Area Level 3 (SA3) regions. Complementing this, the SA3 Highest Education dataset offers insights into educational attainment levels. Additionally, the SA3 Average Age & Income dataset furnishes data on median age and income within SA3 regions. Collectively, these datasets provide a comprehensive understanding of demographic, educational, and socio-economic profiles across SA3 regions which will be used in our scenario.

5.2.2 Health Risk Factors

This dataset contains the modelled estimates of health risk factors such as alcohol consumption, weight and smoking by SA2 districts. This dataset includes counts and rates for each of the mentioned risk factors, of which, we use alcohol consumption and SA2 codes for our analysis with the crash dataset.

5.2.3 Combined SLA11 Premature Mortality & Fertility

This dataset combines the two datasets, i.e. 2008-2012 premature mortality data and 2006-2011 fertility data [2]. The Torrens University Australia - Public Health Information Development Unit originally collected the data later extracted from the Torrens University Australia Public Health Information Development, SUDO 2011. We had initially searched for the premature mortality data based on the Statistical Local Area (SLA) 2011 [3]. Since we wanted to understand the effect of the weather and air quality on life in Australia, the premature mortality data was later combined with fertility data based on the SLA to get complete information. The combined dataset provides the count of different causes of death in each SLA, and it also provides information about the count of mortality for males vs females along with the total births in the area.

5.3 Other External Data Providers

5.3.1 Crash Dataset

This dataset was obtained from the (Department of Transport and Planning, 2024) and consists of detailed records of car crash incidents within Victoria consolidated from police and hospital reports. Collection began in January 2012 and updates are released with a 7-month lag, making data up until September 2023 available to us. Each record includes information regarding the vehicles involved, the location of the crash, limited demographic information about the people involved, and the number of injuries and fatalities, among other attributes. The dataset was ingested using the streaming file ingestion which is discussed further in section 5.3.4.

5.3.2 SA2 & SA3 GeoJson Dataset

SA2 and SA3 data which are classifications by the Australian Bureau of Statistics, offer mid-sized regions for statistical analysis Australian Bureau of Statistics, 2016. It provides insights into social and economic interactions. Mapping Twitter data with SA3 is effective due to the varied bbox sizes, with SA3 offering a standardized framework for analysis and visualization. Similarly, the crash data and the health risks data are also used with SA2 data to allow for geographical visualisations of the health risks data. While the SA3 and Twitter datasets were ingested by leveraging Kibana's web interface, the crash dataset, health risks and SA3 datasets were ingested using Python scripts.

[2] Project focuses mainly on the cloud implementation rather than accurate scientific analysis hence the non-overlapping data was not considered as an issue

[3] The SLA is the smallest unit defined in the 2011 edition of the ASGC

5.3.3 Twitter Dataset

After downloading the Twitter dataset from Spartan (120GB), preprocessing becomes crucial to reduce file size. This involves filtering for valid sentiment and necessary fields like timestamp, site_name, bbox, text, and sentiment. During preprocessing, we noticed that entries with the same site_name share identical bbox values, suggesting identical center coordinates. Utilizing a parent-child relationship optimizes this, minimizing redundancy across the Twitter dataset.

Once preprocessing is completed and the file size has been reduced to approximately 2GB, to avoid memory issues and large file ingestion due to its large size, we save the file in smaller chunks. Then, we create two Python scripts: the first one to ingest parent documents followed by the second one to ingest child documents. This sequential approach is necessary as child documents must be associated with their respective parent documents in Elasticsearch.

Additionally, to ensure accurate data type assignment and prevent Elasticsearch from generating mappings automatically, it's crucial to create a mapping prior to ingestion. For instance, specifying the appropriate data types is crucial for efficient data handling. For example, when grouping data, using the "keyword" datatype ensures accurate grouping, and for visualizing coordinates, defining fields as "geo_point" allows for map visualization in Kibana. Additionally, utilizing "geo_distance" in queries enhances location-based searches and analyses.

5.3.4 Streaming File Ingestion

For medium-sized files, we utilised the python package (Rodrigo Tobar, 2023) in order to use a generator to stream values from the JSON file without loading the whole file into memory, but only chunks of it. However, to do so we needed to have our JSON file be a list of all the records which was easily done by slightly modifying the original JSON file and extracting the list of records. These chunks would then be passed through the streaming_bulk API and ingested into Elasticsearch using our specified mappings.

5.4 Limitation for Large File Ingestion

While handling large file ingestion, several challenges may arise, including VPN disconnections and cluster connection timeouts. When testing the streaming_bulk and bulk API with large files, two specific issues emerged. Firstly, datasets like SA3 pose ingestion difficulties due to the substantial size of each entry, despite having a relatively small number of entries. These entries contain extensive Polygon or Multipolygon data, making ingestion challenging even with streaming_bulk, as connection timeouts are common. Similarly, the Twitter dataset, containing 5 million entries, presents its own set of challenges despite the relatively small size of each entry. The sheer volume of data increases the likelihood of connection timeouts during the ingestion process. Even with smaller individual entries, these timeouts can disrupt ingestion operations.

To mitigate this problem, breaking down files into smaller chunks or using Kibana for GeoJSON uploads proved advantageous. Secondly, streaming_bulk lacks atomicity; if the program halts midway, ingested data doesn't revert. To tackle this, we implemented a solution by tracking and printing the last ingested entries, enabling re-ingestion from the last successful entry upon errors. Although the bulk API seemingly offers atomicity, it falters with exceedingly large files, failing to ingest any data due to file size. This presents a significant challenge, necessitating file segmentation or alternative approaches like streaming methods or Kibana utilization for improved data ingestion efficiency.

6 Backend

6.1 Twitter and SA3

6.1.1 Twitter SA3 Mapping

Considering the extensive size of the Twitter dataset (roughly 5 million entries), conducting preliminary analysis is essential. This involves establishing join relations and creating a new index to efficiently manage latency of API responses.

Mapping the Twitter index coordinates with the SA3 index (339 documents) polygon area involves processing 30,354 parent locations from the Twitter index, which may include locations outside Australia. Both the Twitter and SA3 maps are illustrated in Figure 4.

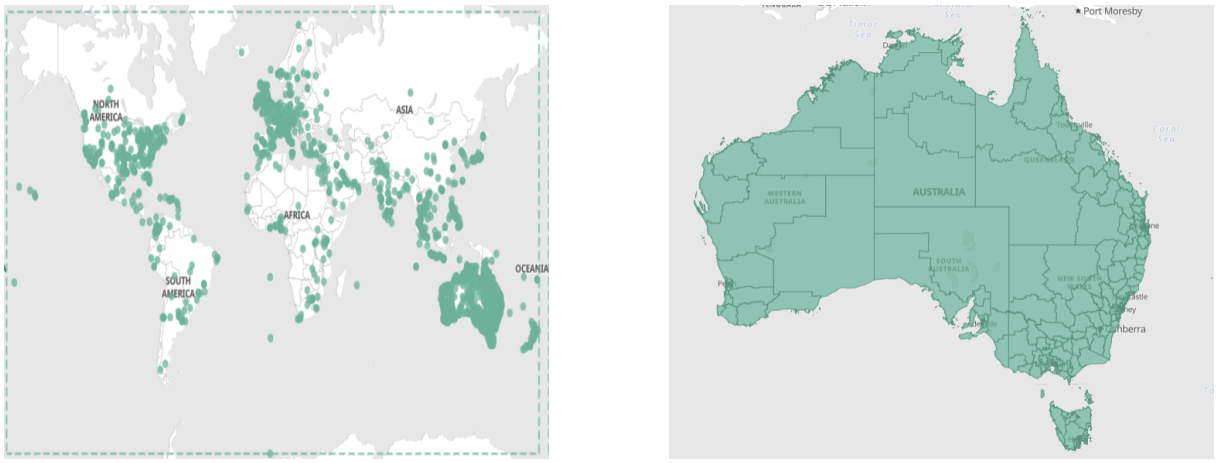


Figure 4: Twitter VS SA3 Map on Kibana

By using an Elasticsearch query with the "geo_distance" filter, specifically configured with a 1km distance and a size limit of 1, the objective is to pinpoint the nearest SA3 polygon document for each coordinate within the Twitter dataset. Subsequently, the Twitter index is updated by incorporating the "sa3_code_2021" field, derived from the SA3 polygon mapping. By implementing this approach, we ensure that only relevant Twitter coordinates within Australia are associated with SA3 regions, maintaining the accuracy and relevance of spatial analyses involving SA3 regions within the Australian context.

6.1.2 Twitter & SA3 SUDO data Joined

For a scenario involving the Twitter dataset, we are interested in comparing average sentiment and Twitter count with information from SA3 SUDO data. Hence, it's crucial to group by sa3_code_2021, aggregate sentiment in Twitter by sa3_code_2021, and then create a fission function to query and join SUDO data with Twitter data that shares the same sa3_code_2021. This joined data is served via the Fission

APIs. Additionally, data (median income, median age, education and population age/gender) regarding individual SA3 areas can be retrieved via an additional set of API endpoints. Documentation for any Twitter and SA3 data APIs are provided in the Appendix section 10.2.

6.2 Road Crashes and SA2 Health Risks

To perform a comparative analysis between the road crashes and SA2 health risk datasets, a spatial join was required. This operation was implemented with a fission function which iteratively pulls data from the crashes, health risks and SA2 indexes and then uploads the joined results to a new index. Iterating over each SA2 document, the instance's geometry is used as a pre-indexed shape in spatial intersection query against the geo-locations in the crashes index which returns every crash document located inside. These crash documents are then extended with the corresponding health risk and SA2 information before being uploaded into the joined index. To process such a large number of documents, the Elasticsearch scroll API is used to paginate the crash intersection query response and the ElasticSearch bulk API for managing uploads.

The analysis made available for this joined dataset has been exposed with two API endpoints: one to take a subsample and one to perform customisable grouping and aggregation. The former simply takes a desired sample size and returns a randomly sampled subset of the documents. The latter is more complicated as it allows for nested aggregation operations. Firstly, the dataset is grouped by a user-specified parameter. Commonly, this would be set to group by SA2 regions. Secondly, and optionally, the data can be further aggregated by specifying an aggregation operation and the field it will apply to. This flexibility allows for a wide range of possible analyses.

An additional endpoint was exposed for downloading every SA2 region's geometry together in a format easily convertible to GeoJSON. This was included to aid with spatial visualisation.

6.3 EPA, BOM and SUDO Mortality Fertility data

SUDO mortality and fertility data was combined using the SUDO tool based on the area code present in the dataset. We have also used SUDO interface to add the geometry into the dataset before extracting the dataset. We used the *ogr2ogr* (GDAL., 2018) python utility to ingest the shape file directly into the Elasticsearch which automatically generates the index and mapping as decided by Elasticsearch.

We exposed the RESTFul API endpoints to retrieve the data from the Elasticsearch to present it to the users. For the weather and air quality data, we introduced 3 endpoints for each dataset: to get the entire data, data based on a start date and data between specific dates. For the mortality and fertility data, we made 3 endpoints available: Get the entire dataset for mortality and fertility data, mortality and fertility data for a specific geo location and top 7 mortality and fertility sites based on a specific cause. The details of the API endpoints and their usage is available in Appendix section 10.2.

7 Front-end

7.1 JupyterHub

JupyterHub is a multi-user notebook version designed to increase collaboration and scale (Kluyver et al., 2016). It allows the user to focus on development tasks without worrying about the installation and maintenance tasks. The user of JupyterHub can collaborate on the shared workspace or work on their workspaces to run their experiments. In this project, we have installed JupyterHub on the Kubernetes environment, allowing maximum collaboration between the team and keeping track of our work. Following are the key features of our implementation.

- **Scalable** - We have deployed JupyterHub on the Kubernetes cluster, allowing it to be scaled and maintained efficiently for large numbers of users using Zero to JupyterHub. Zero to JupyterHub is a Helm Chart for deploying JupyterHub quickly.
- **Authentication** - While Jupyterhub is flexible and supports various user authentication methods, we have used the default authentication method, a local authentication with a shared username and password. Given the overall security of the MRC and the inaccessibility of JupyterHub from the Internet, we have not delved into authentication any further.

The JupyterHub deployed in the Kubernetes cluster can be accessed locally in the browser by establishing the SSH connection to the MRC and port forwarding for the jupyter lab.

```
kubectl --namespace=jupyter port-forward service/proxy-public 8080:http
```

7.2 IPyWidgets

IPyWidgets, developed by Jupyter widgets community, 2015, are the interactive HTML widgets for Jupyter notebooks and the IPython kernel. They allow the user to interact with the jupyter notebook and dynamically render the output based on the user selection. IPyWidgets not only provide a myriad of fundamental widgets known as core interactive library but also provide a framework to add custom widgets. Following are the set of core interactive widgets we have added to our project

- Button
- Dropdown
- Display areas (VBox, HBox) - To group the similar widgets together.

7.3 Folium

Folium is the powerful python-visualization, 2020 library based on the Leaflet.js. It enables both the binding of data to a map for choropleth visualisations and the passing of rich HTML visualisations as markers on the map. In this project, we have used the Folium library to draw the choropleth maps within the Jupyter Notebook. Folium allowed us to pass on the custom HTML content and JavaScript to customise the user's interaction with the map.

```
# Create a map centered around Australia
m = folium.Map(location=[-37.8136, 144.9631], zoom_start=7.5)
```

Weather, air quality, and mortality-fertility data analysis pass the custom HTML and JavaScript to allow users to interact with the maps beyond the fundamental map interaction functions. We also pass the plotly.js to plot the scatter plot and bar chart upon user interaction within the Choropleth map to enhance the user experience.

7.3.1 Plotly.js

Plotly.js (Plotly Technologies Inc., 2015) is an open-source JavaScript library that allows users to create various plots, such as bar charts and scatter plots. The code snippet below shows how to load the minified Plotly library and custom JavaScript into the folium map.

```
m.get_root().html.add_child(folium.JavascriptLink(
    'https://cdn.plot.ly/plotly-2.32.0.min.js'))

m.get_root().script.add_child(Element(js))
```

7.4 Matplotlib

Matplotlib (Hunter, 2007) is a versatile library for creating static, interactive, and animated visualizations in Python. It offers a wide range of plotting functions to create various types of plots, such as line plots, scatter plots, bar plots, histograms, and more. In this project, we utilized Matplotlib to generate static visualizations within the Jupyter Notebook, customizing options to tailor the appearance and layout of our plots to meet specific requirements.

8 Data Analysis

8.1 Weather, Air quality and Fertility-Mortality data analysis

In this section, we shall discuss the analysis performed on the three datasets i.e. Weather data, Air quality data and the combined fertility and mortality data.

8.1.1 Individual Dataset analysis

The charts presented in this section aim to provide a foundational understanding of weather and air quality data. We utilised Fission functions and Flask-based APIs to retrieve data from Elasticsearch. These API functions employ Elasticsearch aggregation queries to fetch the average daily particle count and average daily temperature for the SLA in Victoria. Users can interact with these plots using the interactive IPyWidgets within the Jupyter Notebook. For more details on the Jupyter Notebook and IPyWidgets, refer to section 7.

To analyse the daily variations in weather and air quality data, we initially plot box plots to observe the variance in daily averages for each dataset. A box plot showing the temperature variation is shown in Figure 5 and another box plot showing the variation in air quality is shown in Figure 6.

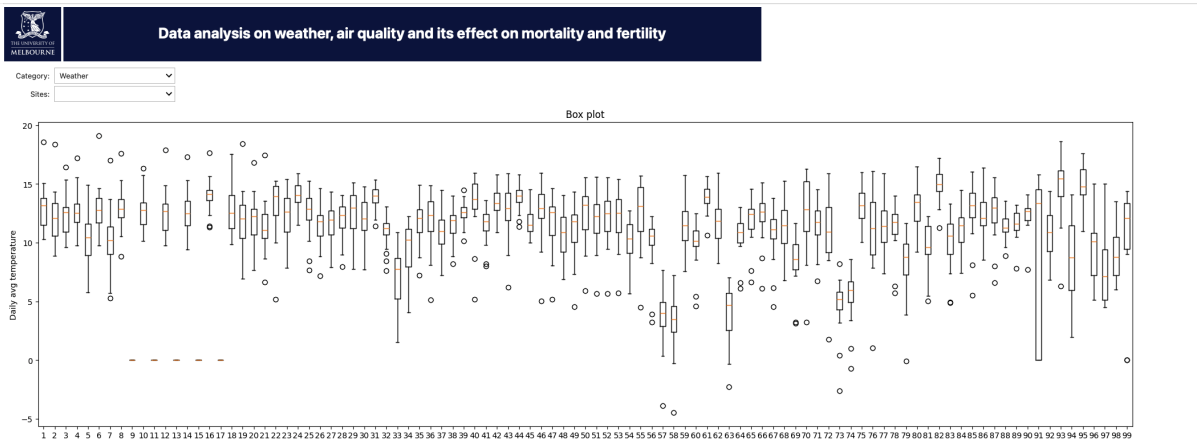


Figure 5: Box plot showing the temperature variation for each site

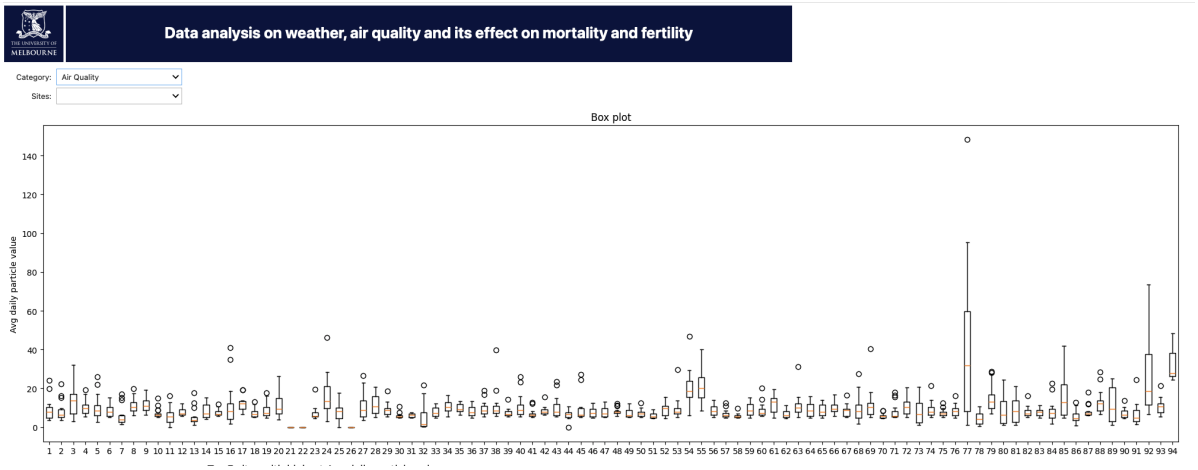
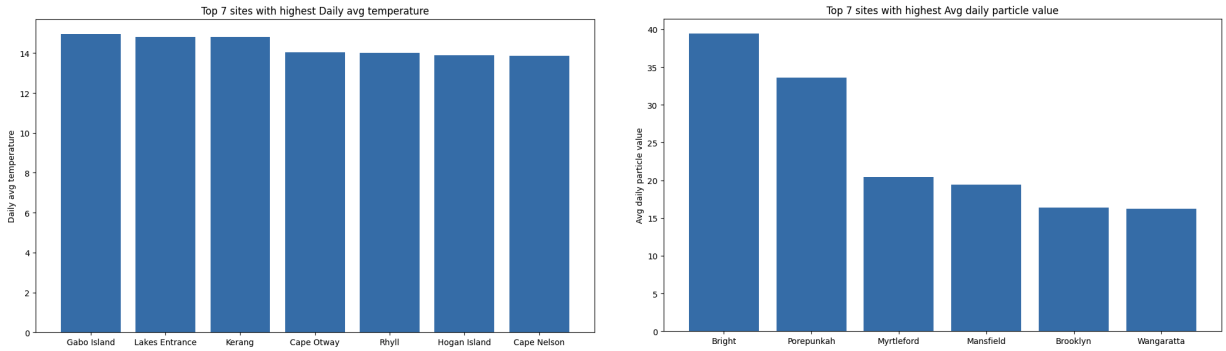


Figure 6: Box plot showing the air quality data variation for each site

Since data is available for approximately 90 sites, we have also plotted the top 7 sites with the highest average values for daily particle counts or temperatures as shown in Figure 7. Users can further interact with the map by selecting specific sites and examining the box plots for daily average variations at those sites.



(a) Top 7 sites with highest average temperatures

(b) Top 7 sites with highest particle value

Figure 7: Analysis for the top 7 sites

8.1.2 Is there a relationship between air quality and daily temperatures?

We incorporated a correlation matrix (shown in Figure 8) into our analysis to determine whether there is a relationship between daily particle values and average daily temperatures. While the geo point location available in each dataset does not match exactly, we spatially joined the two datasets with a tolerance of 1 KM distance between the two locations. The correlation matrix revealed a negative correlation between the two variables. Specifically, this indicates that areas with higher particle counts tend to experience lower daily temperatures.

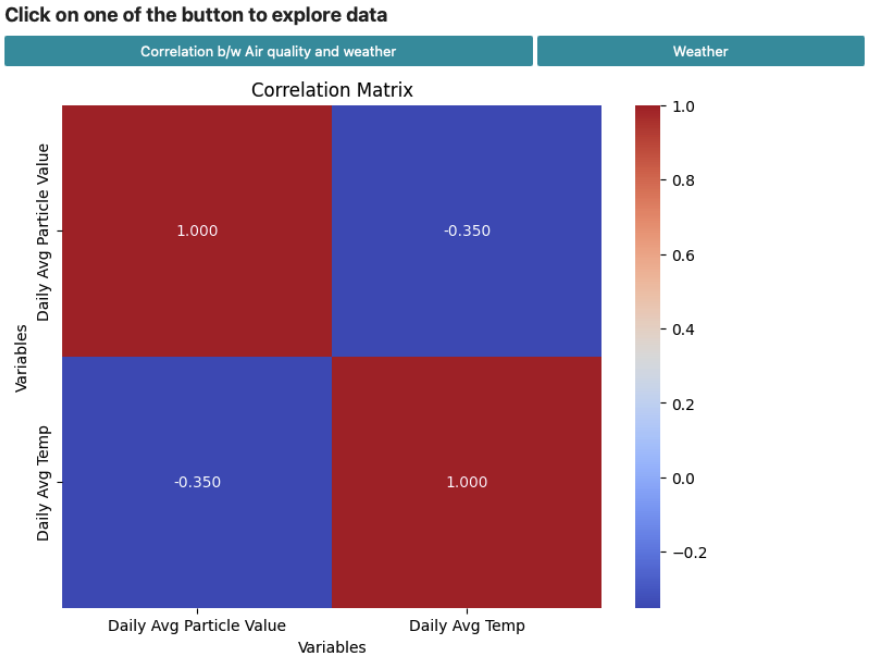


Figure 8: Correlation matrix for the weather and epa data

8.1.3 Is there a relationship between weather, air quality and fertility and mortality

To investigate the potential relationship between fertility, mortality, temperature and air quality data, we plotted a choropleth map to visualise the locations of each weather and air quality monitoring site in Victoria as shown in Figure 10 and Figure 11. We defined specific ranges for average particle count and temperature to differentiate the sites and illustrate their variations. Users can hover over each data point to view the average values for the entire collection period.

To examine the correlation between fertility and mortality data and air quality or weather data, users can click on the desired point on the map to explore that area's fertility and mortality figures. Our analysis indicates that the top seven sites identified earlier for the highest particle counts do not match the sites for the the highest number of premature mortalities as shown in Figure 9. This behaviour is expected as the weather data and the air quality data is current but the mortality and fertility data is from 2011.

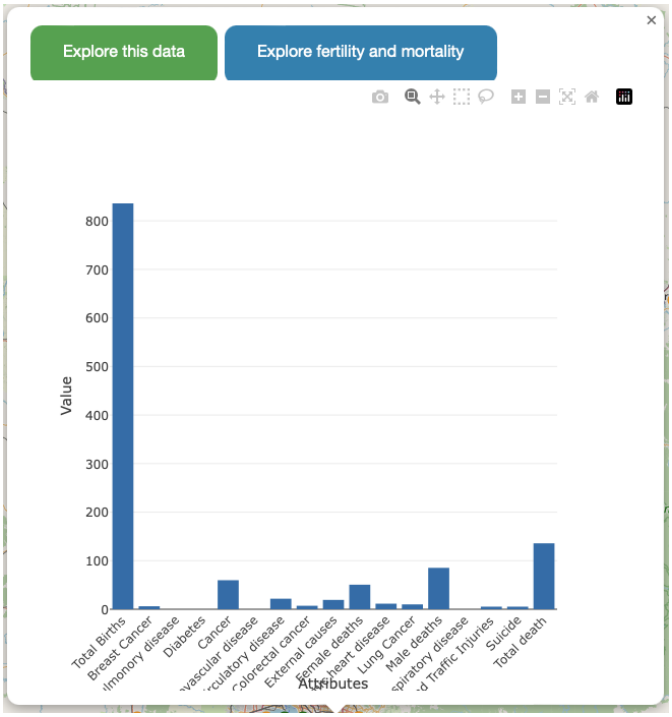


Figure 9: Mortality and Fertility data for a site

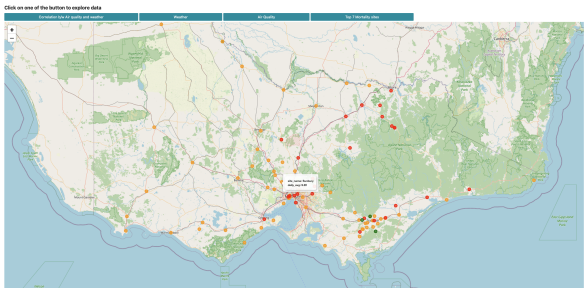


Figure 10: Choropleth Map for the Weather data depicting all sites

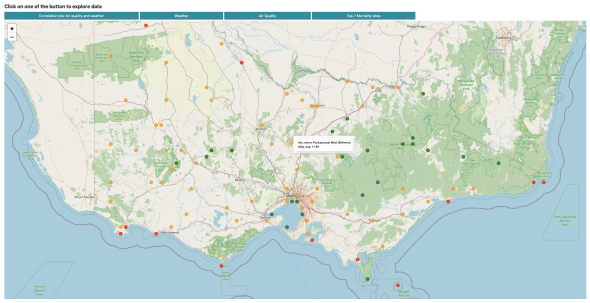


Figure 11: Choropleth Map for the EPA data depicting all sites

8.2 Road Crashes and SA2 Health Risks

8.2.1 Road Crash Distribution

The first two maps included in the road crashes section of the notebook are designed to summarise the distribution of crash locations and their frequency as they relate to SA2 regions within Victoria. The heatmap (Figure 13) comprises a 5000-point random subsample of crash locations which visually combine together to show the relative hotspots of road crash instances. This map highlights how more densely populated regions are more likely to have crashes, as too are features such as major roads and freeways.

To link this data with the SA2 districts, we have also created a choropleth view (Figure 12) of the crash frequency in which the crashes are grouped by the SA2 region they are within. This count of crashes is then normalised against the size of the SA2 region and transformed into a log scale. Similar conclusions can be drawn from this map; most towns and urban areas have higher crash frequencies with the innermost suburbs of Melbourne having significantly more crashes than anywhere else. The CBD itself has by far the most crashes for its area, as might be expected. Each SA2 region can be hovered over to see its exact value.

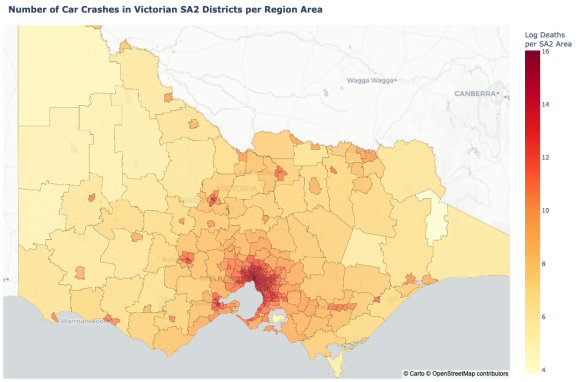


Figure 12: Crashes by SA2 regions

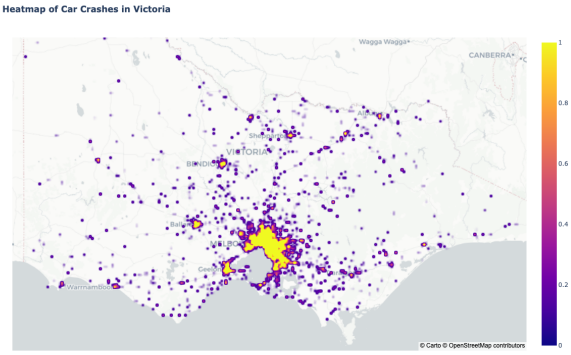


Figure 13: Heatmap of crashes

8.2.2 Alcohol Consumption vs Crash Severity

Crash severity is another angle we can explore to understand how these risk factors influence road crash characteristics. To dig into this relationship, we included a scatter chart (Figure 14) of SA2 districts plotted by alcohol consumption rate and the average number of crash fatalities. Despite the somewhat noisy data, the linear trendline shows some degree of positive correlation between an SA2 district's average alcohol consumption rate and the likelihood of road crashes having more fatalities. Another lens from which we can look at this question is by instead comparing the number of serious injuries resulting from each crash. The bar chart (Figure 15) visualisation groups each individual crash by its number of serious injuries, for each of which the average alcohol consumption rate of the crash SA2's are plotted. From this figure, we can observe another clear trend where, as the number of serious injuries increases, so too does the average alcohol consumption rate.

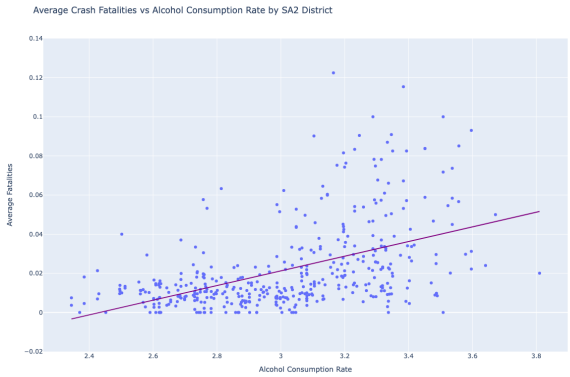


Figure 14: Fatalities vs. Alcohol

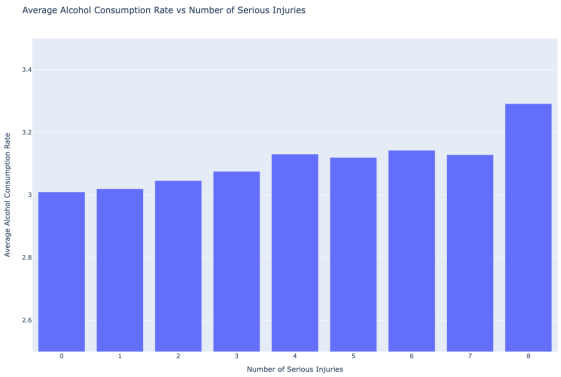


Figure 15: Average Alcohol per Number of Injuries

8.2.3 Alcohol Consumption & Crash Densities

Finally, we highlight the densities of both alcohol consumption and car crashes across Victoria's SA2 districts. To visualise these two densities together we utilised a choropleth map (Figure 16) that highlights the distribution of alcohol consumption. We then used a marker cluster to visualise the density of car crashes across Victoria. The grouping of the markers allowed for easier interpretation of the density of crashes. Both the choropleth and the markers exhibit high values in the inner suburbs and generally decrease the further you move away from the CBD. This indicates an association between the two variables but the correlation with distance away from CBD suggests an association with the concentration of population in SA2 districts which must also be considered. It is also worth noting that the SA2 district marked as blue did not have a corresponding alcohol consumption value.

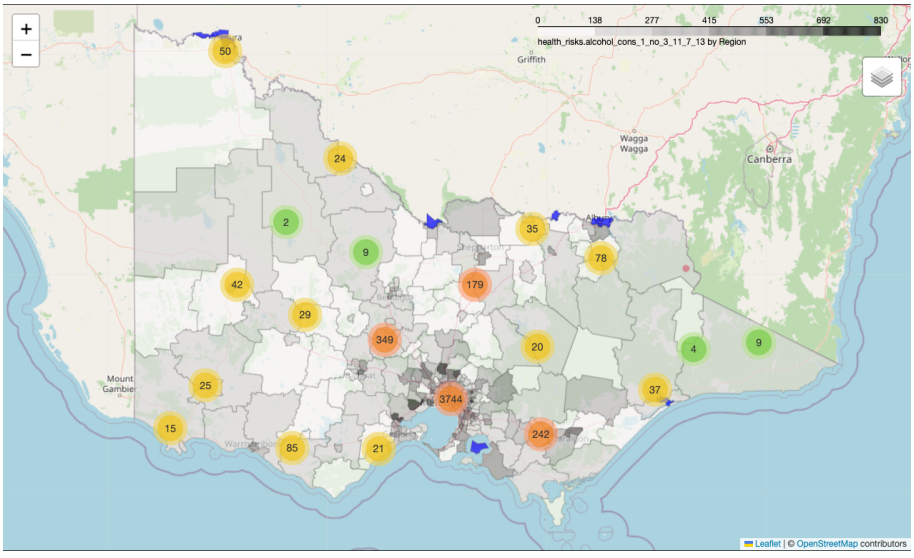


Figure 16: Crashes by SA2 Districts

8.3 Twitter Sentiment Analysis

8.3.1 Twitter Use and Population Counts

For this analysis, we examined Twitter data from June 2021 to June 2022 and compared it with Australian population data from 2021. We focused on correlating the overall tweet counts for each area as defined by the "Statistical Areas Level 3 (SA3)". We see there is an important correlation between Twitter usage and population count.

The data presented in the top right chart of Figure 17 indicates that the ratio of men to women is consistent across the regions which suggests that gender does not significantly impact Twitter usage patterns. Therefore, this implies that people are equally likely to use Twitter, irrespective of region. On the other hand, we observe that population size may have an exponential correlation with Twitter usage, as measured by the ratio between the number of tweets and the total population, as seen in the bottom right chart in Figure 17. This means regions with larger populations tend to have a disproportionately higher number of tweets than smaller populations. This upward trend suggests that Twitter engagement tends to scale alongside population numbers.

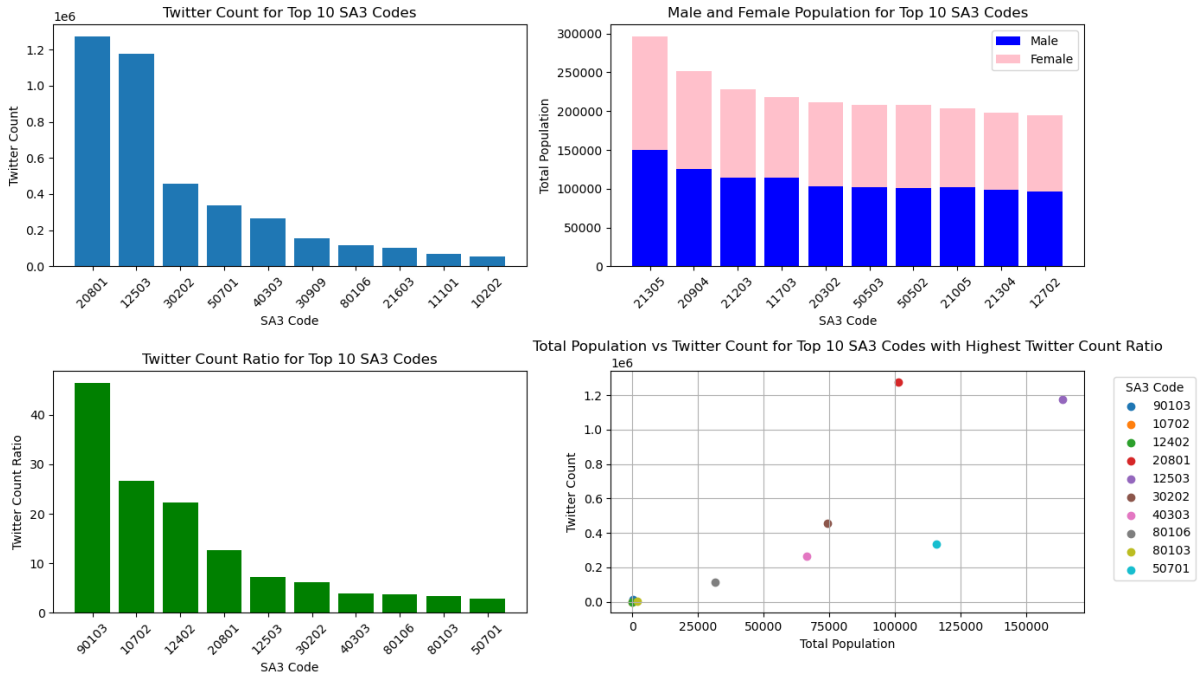


Figure 17: Twitter Use vs Population data

8.3.2 Is there a relationship between happiness and age?

We have generated a layered choropleth map of Australia showing happiness, as measured by Twitter sentiment, and median age as seen in Figure 18. This map highlights areas where Australians are on average happier or older. We can see that overall, the average age of Australians is quite evenly distributed across the SA3 areas despite the notable variation in sentiments.

For example, areas such as East Pilbara, have high sentiment but no substantial difference in age to the general population with a median age of 33. Areas such as the Yorke Peninsula, have an older population with a median age of 55 but an average sentiment of only 0.061 which is middling, as seen in Figure 19.

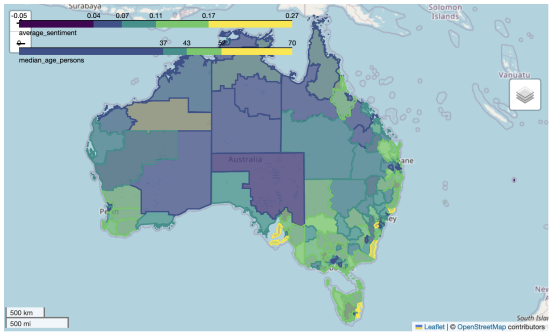


Figure 18: Choropleth map showing the relationship between median age and sentiment

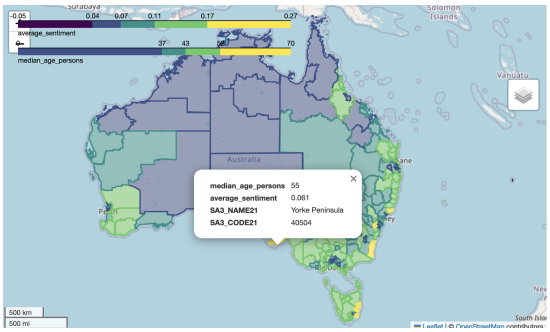


Figure 19: Yorke Peninsula data from Sentiment v. Age map

Therefore, the data demonstrates little correlation between age and happiness across different regions, challenging the common belief that happiness is predominantly found in youth. We suspect there may be underlying socio-economic factors influencing general sentiment or happiness that are not evident through age data alone. Furthermore, we explore potential variables that might affect overall feelings of satisfaction or joy among the population, such as education and income levels.

8.3.3 Is there a relationship between happiness and education?

To determine if there is a correlation between happiness and education, we used data from the ABS to show the population distribution across various education levels. We define being educated as individuals who have completed Year 12. Consequently, education ratio is the proportion of individuals that have completed Year 12. By creating a choropleth map, we can observe that Australia has a relatively uniform level of education, with very few exceptions, as illustrated in Figure 20.

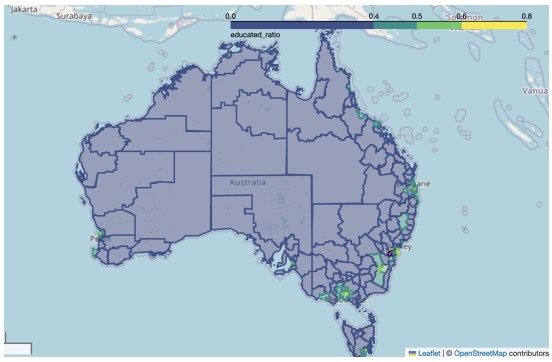


Figure 20: Choropleth map for Education Ratio

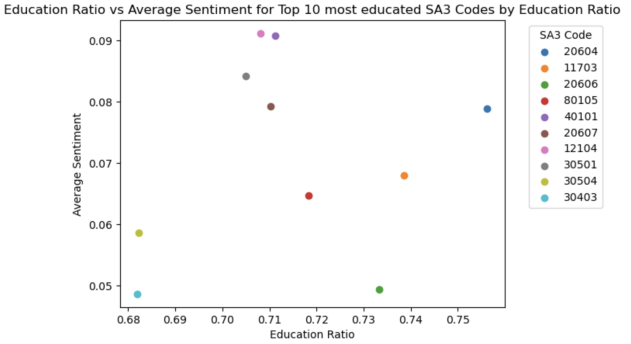


Figure 21: Education vs Average sentiment for top 10 most educated SA3 Areas

Areas with higher sentiment generally do not seem to have higher or lower education levels than lower sentiment areas, as shown in the distribution in Figure 21.

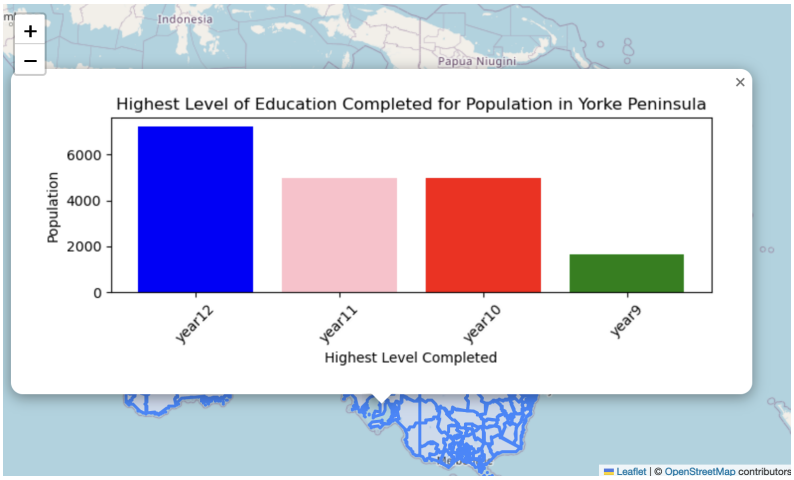


Figure 22: Education Distribution of Yorke Peninsula

Furthermore, we observe that the population, as a whole, is generally well educated to a Year 12 standard and does not deviate much for areas such as the York Peninsula as seen in Figure 22.

Overall, despite the initial hypothesis that there may be an observable relationship between happiness and education, we can not currently see any real influence that may support this case.

8.3.4 Is there a relationship between happiness and income levels?

Similar to the distribution of happiness by education level, Figure 23 indicates that personal weekly income does not have a significant impact on happiness.

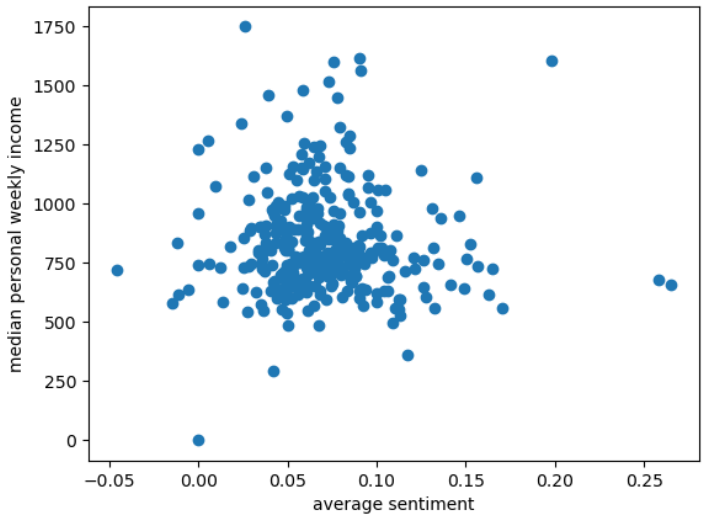


Figure 23: Sentiment and income distribution across all SA3 areas

8.3.5 Final words for Twitter analysis

Overall, our basic analysis of happiness, as measured by Twitter sentiment, has offered some interesting initial insights into the factors that may influence public happiness. However, a comprehensive understanding of happiness likely requires a more complex investigation beyond the factors we have explored. Simple analysis alone cannot provide meaningful results, and an in-depth exploration of various variables is necessary for a thorough study of the nuances of population happiness.

9 Conclusions

In this report, we have discussed our use of the Melbourne Research Cloud (MRC) to facilitate big data analytics from various sources, including the Spatial Urban Data Observatory (SUDO), Twitter, the Environment Protection Authority (EPA), the Bureau of Meteorology (BoM), and DataVic. By leveraging the MRC in conjunction with Kubernetes, Fission, ElasticSearch, Kibana, Kafka, and Jupyter Notebooks, we have developed a robust system for harvesting, storing, and analyzing data.

By employing Kubernetes on the MRC, we were able to efficiently handle the deployment and resource management of nodes and containers, enabling the creation of performant cloud solutions. Data harvesting and ingestion into ElasticSearch were automated using serverless Fission functions, with further processing and API endpoint serving also facilitated by Fission.

Our front-end application, built with Jupyter Notebooks and Python libraries such as Matplotlib, IPYWidgets, Folium, and Plotly.js, enabled us to perform comprehensive data analysis and visualization. Using the EPA and BoM data, we identify whether there is a correlation between air quality and daily temperature through the creation of correlation matrices and regression plots. Furthermore, we link this to fertility and mortality data obtained from SUDO to chart areas that may have higher mortality rates while serving accompanying weather statistics. Additionally, we visualise the geographic distribution of road crash data obtained from DataVic with heatmaps and region-grouped choropleth maps. Further charts plotting aggregations made on the joined crash and health risk dataset complement this to provide comparative analyses between crash severity and factors such as alcohol consumption. Similarly, we use choropleth maps, scatterplots and bar charts to conceptualise various features that may influence happiness expressed on social media. By using Twitter sentiment as a measure of happiness across certain SA3 areas, we create comparisons with age, income and education, to understand how these factors impact the happiness of the population.

Throughout the project, we handled certain difficulties and errors. For example, file ingestion was a task that had certain size limitations due to the size of our data.

For example, many of our datasets needed additional care during the ingestion process due to their size. This was addressed by modifying our processes to effectively chunk and ingest the information. Additionally, some Twitter data was

This was addressed by modifying our processes to effectively chunk and ingest the information. Additionally, some Twitter data was inconsistent and varied in geometric specificity (where some bounding boxes encased entire states while others were a single location). This was addressed by ignoring data with missing geometry and remapping tweets that contained geometries to a corresponding SA3 area. Furthermore, we ensure strong API serviceability by generating comprehensible error codes to notify the client of any issues.

In conclusion, the MRC provided a suitable cloud infrastructure for our big data analytics needs, enabling us to harvest, store, and analyze large-scale data effectively. This project demonstrates the potential of integrated cloud solutions and modern data analytics tools in addressing different urban and environmental challenges.

References

Australian Bureau of Statistics. (2016). *SA3 Data*. https://www.abs.gov.au/ausstats/abs@.nsf/Lookup/by%20Subject/1270.0.55.001~July%202016~Main%20Features~Statistical%20Area%20Level%203%20(SA3)~10015

Bureau of Meteorology. (BOM 2024). Latest Weather Observations for Victoria. https://reg.bom.gov.au/vic/observations/vicall.shtml

Conference, eResearch Australasia. (n.d.). *Introducing the Spatial Urban Data Observatory*. https://conference.eresearch.edu.au/introducing-the-spatial-urban-data-observatory/

Department of Transport and Planning. (2024). *Victoria road crash data*.

Environment Protection Authority Victoria. (EPA 2024). Environment Monitoring - Air Quality. https://www.epa.vic.gov.au/for-community/airwatch

GDAL. (2018). *Gdal/ogr*. https://gdal.org/programs/ogr2ogr.html

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Jupyter widgets community. (2015). *Ipywidgets, a github repository* [Retrieved from https://github.com/jupyter-widgets/ipywidgets].

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., & Willing, C. Jupyter notebooks – a publishing format for reproducible computational workflows (F. Loizides & B. Schmidt, Eds.). In: *Positioning and power in academic publishing: Players, agents and agendas* (F. Loizides & B. Schmidt, Eds.). Ed. by Loizides, F., & Schmidt, B. IOS Press. 2016, 87–90.

Plotly Technologies Inc. (2015). *Collaborative data science*. https://plotly.com/javascript/

python-visualization. (2020, December 28). *Folium* (Version 0.11.0). https://python-visualization.github.io/folium/

Rodrigo Tobar, I. S. (2023). Ijson 3.2.3. https://pypi.org/project/ijson/#description

Torrens University Australia Public Health Information Development. (SUDO 2011). SLA 11 Combined Mortalitity Fertility Data. https://sudo.eresearch.unimelb.edu.au

10 Appendix

Youtube link - TO DO

Github Link - https://github.com/nikets4/COMP90024-Assignment2

10.1 Individual Contribution

- **Niket Singla** - Cluster setup, Jupyter hub setup on K8 cluster, Kafka setup, EPA & BOM data ingestion, Sudo data ingestion for fertility and mortality, Elasticsearch, Data analysis API, Jupyter frontend, Unit tests for all API and ingestion functions.
- **Liam Brennan** - Crash and health risk processing backend, Joined crash and SA2 analysis API , Crash and health risk frontend visualisation, Unit and end-to-end API endpoint testing
- **Parsa Babadi Noroozi** - Crash and health risk data collection, Crash and health risk elasticsearch ingestion, Crash and health risk front end visualisation, Crash and health risk analysis
- **Patipan Rochanapon** - Twitter, SA3, SUDO (SA3) Ingestion, Twitter and SA3 coordinates mapping, Join Twitter with SUDO (SA3), Twitter and SA3 Data Analysis
- **Jason Phan** - Twitter API routes, SA3 data API routes, Twitter and SA3 Data Analysis, Twitter sentiment and age/income/education data visualisations, Elasticsearch queries, End to End testing for API endpoints

10.2 API Docs

10.2.1 `/highest-level-of-schooling/sa3/[sa3_code]`

- **Method:** GET
- **Description:** Retrieves the highest level of schooling data from the elastic search index.
- **Parameters:**
  - `sa3_code` (required): The SA3 code
- **Example Request:**

```
GET /highest-level-of-schooling/sa3/12101
```

- **Example Response:**

```
{
    "f_hghst_yr_schl_ns_tot": 2113,
    "avg_education":
        {
            "male": 11.715261140392908,
```

```
            "female": 11.634709082566804,
            "person": 11.807369637002205
        },
    ...
}
```

10.2.2 /age-by-sex/sa3/[sa3_code]

- **Method:** GET
- **Description:** Retrieves age by sex data from the elastic search index.
- **Parameters:**
  - `sa3_code` (required): The SA3 code
- **Example Request:**

```
GET /age-by-sex/sa3/12101
```

- **Example Response:**

```
{
    "age_75_84_c21_p": 5474,
    "age_35_44_c21_p": 20266,
    "age_grp_85over_c21_m": 962,
    ...
}
```

10.2.3 /median/age/sa3/[sa3_code]

- **Method:** GET
- **Description:** Retrieves median age data from the elastic search index.
- **Parameters:**
  - `sa3_code` (required): The SA3 code
- **Example Request:**

```
GET /median/age/sa3/12101
```

- **Example Response:**

```
{
    "median_age_persons": 38
}
```

10.2.4 /median/income/sa3/[sa3_code]

- **Method:** GET
- **Description:** Retrieves median income data from the elastic search index.
- **Parameters:**
  - `sa3_code` (required): The SA3 code
- **Example Request:**

```
GET /median/income/sa3/12101
```

- **Example Response:**

```
{
    "median_tot_prsnl_inc_weekly": 1255
}
```

10.2.5 /get-sa3-geojson

- **Method:** GET
- **Description:** Retrieves all SA3 GeoJSON data from the elastic search index.
- **Parameters:**
  - `N/A`
- **Example Request:**

```
GET /get-sa3-geojson
```

- **Example Response:**

```
[
    {
        "id": "0",
        "type": "Feature",
        "properties": {
            "SA3_CODE21": "10303",
            "SA3_NAME21": "Lithgow - Mudgee"
        },
        "geometry": {
            "type": "Polygon",
            "coordinates": [
                ...
            ]
        }
```

```
    },
    {
        "id": "1",
        ...
    },
    ...
]
```

10.2.6 /sa3-joined/all

- **Method:** GET
- **Description:** Retrieves all SA3 GeoJSON data from the elastic search index.
- **Parameters:**
  - N/A
- **Example Request:**

```
GET /sa3-joined/all
```

- **Example Response:**

```
{
    {
        "sa3_code_2021": "40403",
        "average_sentiment": 0.08262499777430837,
        "twitter_count": 253,
        "school_sa3": {
            ...
        },
        "person_age_sex_sa3": {
            ...
        },
        "age_personal_income_sa3": {
            ...
        }
    },
    {
        "sa3_code_2021": "12101",
        ...
    },
    ...
```

```
}
```

10.2.7 /crashes/sample/[size]

- **Method:** GET
- **Description:** Retrieves a random subsample from the joined crashes index.
- **Parameters:**
  - `size` (required): The number of documents
- **Example Request:**

```
GET /crashes/sample/100
```

- **Example Response:**

```
[
    {
        "type": "Feature",
        "geometry": {
            "coordinates": [143.153247, -37.573295],
            "type": "Point"
        },
        "properties": {
            "NO_OF_VEHICLES": 2,
            "INJ_OR_FATAL": 2,
            "DRIVER": 2,
            "FEMALES": 2,
            "PASSENGERVEHICLE": 2,
            ...
        }
    },
    ...
]
```

10.2.8 /crashes/by/[by-field]/[aggregation]/[aggregation-field]

- **Method:** GET
- **Description:** Executes customisable grouping and aggregation on the joined crashes dataset
- **Parameters:**
  - `by-field` (required): The field to be grouped by
  - `aggregation` (optional): The aggregation method

  - `aggregation-field` (optional): The field to which the aggregation is applied

- **Example Request:**

```
GET /crashes/by/sa2/avg/health_risks.alcohol_proportion
```

- **Example Response:**

```
[
    {
        "health_risks.alcohol_cons_1_no_3_11_7_13": 643.75640,
        "count": 2528,
        "sa2.properties.sa2_main11.keyword": "212041311"
    },
    ...
]
```

10.2.9 `/sa2/geometry`

- **Method:** GET
- **Description:** Retrieves every SA2 region's geometry in a format easily convertible to GeoJSON
- **Parameters:**
  - `N/A`
- **Example Request:**

```
GET /sa2/geometry
```

- **Example Response:**

```
[
    {
        "type": "Feature",
        "id": "sa2_2011_aust.586",
        "geometry": {
            "type": "MultiPolygon",
            "coordinates": [...]
        },
        ...
    },
    ...
]
```

10.2.10 `/getepadata`

- **Method:** GET
- **Description:** Retrieves the entire EPA data
- **Parameters:**
  - None
- **Example Request:**

```
GET /getepadata
```

- **Example Response:**


```
{
   {
"siteID": "c69ed768-34d2-4d72-86f3-088c250758a8",
"siteName": "Alphington",
"siteType": "Standard",
"geometry":
{
    "type": "Point",
    "coordinates":
    [
        -37.7784081,
        145.0306
    ]
},
"siteHealthAdvices":
[
    {
        "since": "2024-04-17T15:00:00Z",
        "until": "2024-04-17T16:00:00Z",
        "healthParameter": "PM2.5",
        "averageValue": 4.36,
        "unit": "&amp;micro;g/m&amp;sup3;",
        "healthAdvice": "Good",
        "healthAdviceColor": "#42A93C",
        "healthCode": "1021"
    }
}
```

10.2.11 `/getepadata/startdate/Startdate:[0-9-]*`

- **Method:** GET
- **Description:** Retrieves the EPA data from a specific start date
- **Parameters:**
  - `Startdate` (optional): StartDate
- **Example Request:**

```
GET /getepadata/startdate/2024-05-01
```

- **Example Response:**


```
{
    {
"siteID": "c69ed768-34d2-4d72-86f3-088c250758a8",
"siteName": "Alphington",
"siteType": "Standard",
"geometry":
{
    "type": "Point",
    "coordinates":
    [
        -37.7784081,
        145.0306
    ]
},
"siteHealthAdvices":
[
    {
        "since": "2024-04-17T15:00:00Z",
        "until": "2024-04-17T16:00:00Z",
        "healthParameter": "PM2.5",
        "averageValue": 4.36,
        "unit": "&amp;micro;g/m&amp;sup3;",
        "healthAdvice": "Good",
        "healthAdviceColor": "#42A93C",
        "healthCode": "1021"
    }
}
```

10.2.12 /getepadata/startdate/Startdate:[0-9-]*/enddate/Enddate:[0-9-]*

- **Method:** GET
- **Description:** Retrieves the EPA data between a specific start and end date
- **Parameters:**
  - `Startdate` (optional): Start Date
  - `Enddate` (optional): End Date
- **Example Request:**

  GET /getepadata/startdate/2024-05-01/enddate/2024-05-05

- **Example Response:**


```
{
    {
"siteID": "c69ed768-34d2-4d72-86f3-088c250758a8",
"siteName": "Alphington",
"siteType": "Standard",
"geometry":
{
    "type": "Point",
    "coordinates":
    [
        -37.7784081,
        145.0306
    ]
},
"siteHealthAdvices":
[
    {
        "since": "2024-04-17T15:00:00Z",
        "until": "2024-04-17T16:00:00Z",
        "healthParameter": "PM2.5",
        "averageValue": 4.36,
        "unit": "&amp;micro;g/m&amp;sup3;",
        "healthAdvice": "Good",
        "healthAdviceColor": "#42A93C",
        "healthCode": "1021"
    }
```

```
}
```

10.2.13 /getweatherdata

- **Method:** GET
- **Description:** Retrieves the entire BOM (Weather) data
- **Parameters:**
  - None
- **Example Request:**

```
GET /getweatherdata
```

- **Example Response:**


```
{
"hits": {
  "total": {
    "value": 8119,
    "relation": "eq"
  },
  "max_score": 1,
  "hits": [
    {
      "_index": "weather",
      "_id": "95872-2024-04-21T20:30:00Z",
      "_score": 1,
      "_ignored": [
        "airtemp.float"
      ],
      "_source": {
        "siteid": 95872,
        "sitename": "Fawkner Beacon",
        "geo": "-37.9,144.9",
        "timestamp": "2024-04-21T20:30:00Z",
        "airtemp": "13"
      }
}]}}
```


10.2.14 /getweatherdata/startdate/Startdate:[0-9-]*

- **Method:** GET

- **Description:** Retrieves the BOM (Weather) data from a specific start date
- **Parameters:**
  - `Startdate` (optional): StartDate
- **Example Request:**

  GET /getweatherdata/startdate/2024-05-01

- **Example Response:**


```
{
"hits": {
  "total": {
    "value": 8119,
    "relation": "eq"
  },
  "max_score": 1,
  "hits": [
    {
      "_index": "weather",
      "_id": "95872-2024-04-21T20:30:00Z",
      "_score": 1,
      "_ignored": [
        "airtemp.float"
      ],
      "_source": {
        "siteid": 95872,
        "sitename": "Fawkner Beacon",
        "geo": "-37.9,144.9",
        "timestamp": "2024-04-21T20:30:00Z",
        "airtemp": "13"
      }
}]}}
```


10.2.15 /getweatherdata/startdate/Startdate:[0-9-]*/enddate/Enddate:[0-9-]*

- **Method:** GET
- **Description:** Retrieves the BOM (Weather) data between a specific start and end date
- **Parameters:**
  - `Startdate` (optional): Start Date

  - `Enddate` (optional): End Date

- **Example Request:**

  GET /getweatherdata/startdate/2024-05-01/enddate/2024-05-05

- **Example Response:**


```
{
"hits": {
  "total": {
    "value": 8119,
    "relation": "eq"
  },
  "max_score": 1,
  "hits": [
    {
      "_index": "weather",
      "_id": "95872-2024-04-21T20:30:00Z",
      "_score": 1,
      "_ignored": [
        "airtemp.float"
      ],
      "_source": {
        "siteid": 95872,
        "sitename": "Fawkner Beacon",
        "geo": "-37.9,144.9",
        "timestamp": "2024-04-21T20:30:00Z",
        "airtemp": "13"
      }
}]}}
```


10.2.16 `/mortalityfertilitydata`

- **Method:** GET
- **Description:** Retrieves the entire SUDO mortality and fertility data
- **Parameters:**
  - None
- **Example Request:**

  GET /mortalityfertilitydata

- **Example Response:**


```
  {
"_index": "mortality-fertility-data",
"_id": "vespFI8BDS0m6zmxniPN",
"_version": 1,
"_score": 0,
"_source": {
  "ogc\_fid": 1,
  "geometry": {
    "type": "polygon",
    "coordinates": [],
    }
  "area_code": 205054601,
  "area_name": "Melbourne (C) - Inner",
  "dths_can17": 24.3050617700087,
  "dths_can18": 73.4563071202514
  }}
```


10.2.17 /mortalityfertilitydata/lat/lat:[0-9.-]*/long/long:[0-9.-]*

- **Method:** GET
- **Description:** Retrieves the SUDO mortality and fertility data for a specific lat long (geo location)
- **Parameters:**
  - `lat` (Mandatory): Latitude
  - `long` (Mandatory): Longitude
- **Example Request:**

```
GET /mortalityfertilitydata/lat/25.2744/long/133.7751
```

- **Example Response:**


```
  {
"_index": "mortality-fertility-data",
"_id": "vespFI8BDS0m6zmxniPN",
"_version": 1,
"_score": 0,
"_source": {
  "ogc_fid": 1,
  "geometry": {
```

```
    "type": "polygon",
    "coordinates": [],
    }
  "area_code": 205054601,
  "area_name": "Melbourne (C) - Inner",
  "dths_can17": 24.3050617700087,
  "dths_can18": 73.4563071202514
  }}
```


10.2.18 /sortmortalityfertilitydata/attribute/attribute:[a-zA-Z.-_]*

- **Method:** GET
- **Description:** Retrieves the top 7 results from SUDO mortality and fertility data for a specific category
- **Parameters:**
  - `attribute` (Mandatory): Mortality or Fertility attribute
- **Example Request:**

  GET /sortmortalityfertilitydata/attribute/dths_res60

- **Example Response:**


```
    {
  "_index": "mortality-fertility-data",
  "_id": "vespFI8BDS0m6zmxniPN",
  "_version": 1,
  "_score": 0,
  "_source": {
    "ogc_fid": 1,
    "geometry": {
      "type": "polygon",
      "coordinates": [
      ]
    },
    "area_code": 205054601,
    "area_name": "Melbourne (C) - Inner",
    "dths_can17": 24.3050617700087,
    "dths_can18": 73.4563071202514
    }}
```

10.3 Table of Abbreviations

Abbreviation	**Full Form**
BOM	Bureau of Meteorology
EPA	Environment Protection Authority
SUDO	Spatial Urban Data Observatory
MRC	Melbourne Research Cloud
SA2	Statistical Area 2
SA3	Statistical Area 3
BBox	Bounding Box
ABS	Australian Bureau of Statistics
K8s	Kubernetes

Table 2: Table of Abbreviations