

# Analyzing Major Trends in Victoria

COMP90024 - Assignment 2

Team 58 - University of Melbourne - Parkville

Nitika Malhotra (1037082)

Anupama Mampage (1102749)

Ribhav Shridhar (1037144)

Ronak Arvindkumar (1043591)

An extensive amount of social media data coupled with capabilities of distributed computing can provide insights and narrate interesting stories of the lives of people. This document details the architecture design and functionality of such a big data analytics system built using Twitter data, combined with official datasets from AURIN, making use of resources at UniMelb/NeCTAR research cloud and CouchDB data analytic capabilities.

## 1. Introduction

Social media is increasingly penetrating into the lives of people. As such, many aspects of the day to day lives of citizens in any geography are largely being reflected in their social media footprints.

This project undertakes the implementation of a distributed big data analytics application deployed on the Melbourne Research Cloud, making use of Twitter data gathered over a period of time along with relevant official statistics acquired from the AURIN database.

Stated below is a detailed narration of the system architecture, scenarios used for analytics, details of deployment for end user benefit along with the challenges faced in the overall development phase of the project.

## 2. System Design

### 2.1. Infrastructure set up

#### 2.1.1. System architecture and fault tolerance

The system architecture was designed to utilize the resources of the Melbourne Research Cloud to achieve maximum fault tolerance to the overall system while maintaining the network overhead of the system at a tolerable level. As such, our system consists of the following distributed components as follows:

1. Twitter harvesting applications to harvest real time and historical tweets
2. Front end web application
3. CouchDB cluster storing the raw twitter data, the views obtained after analysis of the gathered data and the related data from AURIN

Our overall system spans over four nodes on the Melbourne Research Cloud, configured suitably with required software dependencies. Figure 1 illustrates the implemented system architecture. As shown in the figure, three nodes are configured to be in a CouchDB cluster. The twitter harvesting applications developed using the twitter streaming and search APIs run on all three nodes configured within the CouchDB cluster. The harvesters on different nodes gather tweets from separate geological areas within Victoria and write the data on to the “tweets” database. Views are created from this data within the tweets database as part of the analysis.

Data from the AURIN database related to the selected scenarios are stored in separate databases populated over the clustered nodes.

The front end web application is deployed on the fourth node.

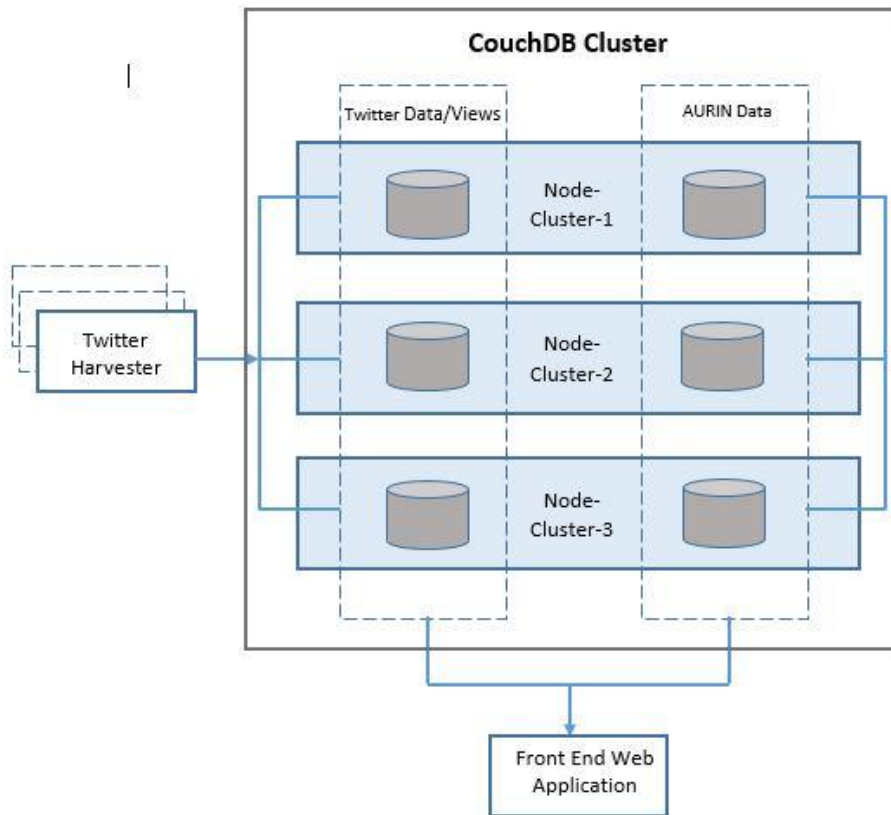


Figure 1:High level architecture diagram

With the distributed nature of the system, as explained above, we were able to achieve fault tolerance to the system since all the harvesters would be running independently in different nodes and without any dependency on any other component. In other words, the harvesters are able to use the twitter APIs and gather data onto the database even if any issue arises with the node containing the front end application. Similarly, the front end application would function independently as intended as long as the databases have sufficient data, irrespective of whether the harvesters are functioning properly.

Also, within the CouchDB cluster, we have used shards and replicas appropriately to achieve fault tolerance for data to a sufficient level while allowing the nodes to gather as much data as possible. This would be explained in detail under section 2.1.3 in the report.

Although the above mentioned mechanisms help create fault tolerance to a certain extent, a single point of failure does exist in our system, where the front end application is concerned. Since it is on a single node, any issue with that node could directly affect the functionality of the

system which is an issue to be addressed. Also, our system would not scale dynamically to meet the demand. These are areas to be explored in further improvement to the system.

### 2.1.2. Resource Allocation

For this project, our group was allocated with 250 GB of volume storage, 36 GB of RAM and an allowance of four NECTAR instances in total. With the limited resources thus provided, we decided to allocate resources to our system components as stated in Table 1. Figure 2 below gives a better illustration of the distributed component as per the system architecture described above.

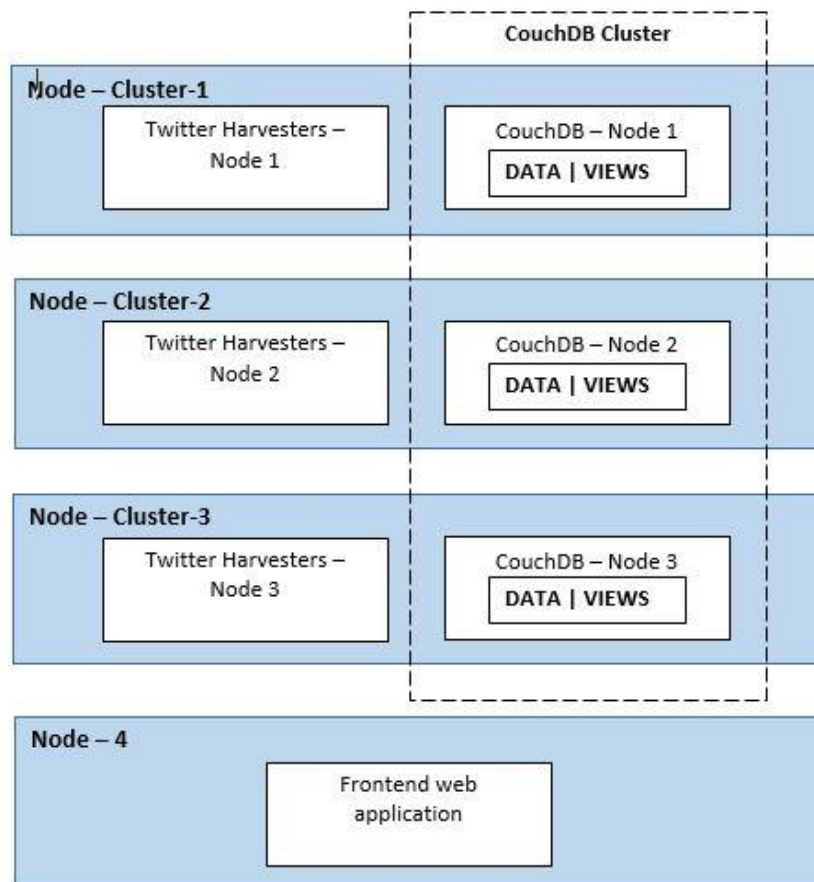


Figure 2: Distribution of system components on the NECTAR instances

Instance	RAM	VCPU	Disk	Volume
Node-Cluster-1	9 GB	2	30 GB	75 GB
Node-Cluster-2	9 GB	2	30 GB	75 GB
Node-Cluster-3	9 GB	2	30 GB	75 GB
Node-4	9 GB	2	30 GB	25 GB

*Table 1: Resource Allocation*

The three NECTAR instances comprising the CouchDB cluster were mounted with volumes of 75 GB each in order to provide more storage space for the twitter data to be harvested continuously so that the subsequent analysis would be more comprehensive.

### 2.1.3 CouchDB Cluster

The system architecture of the project is primarily based on an Apache CouchDB cluster deployed over three nodes.

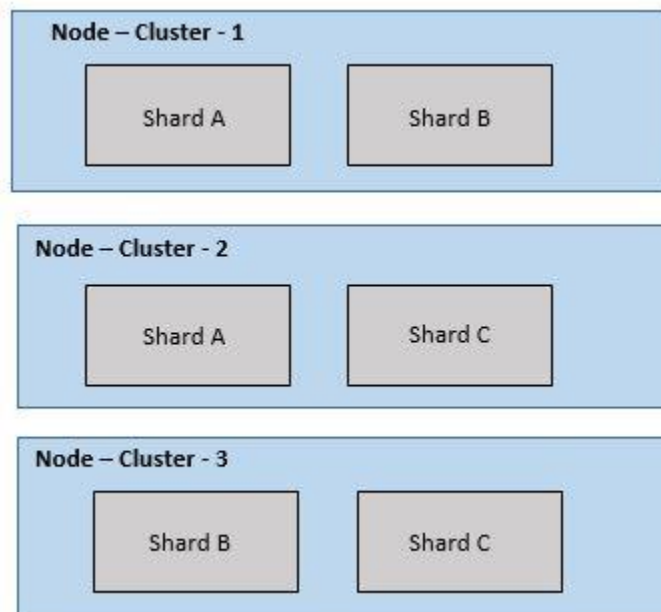
One main reason for incorporating CouchDB as the NoSQL database for our software implementation was its powerful ability to replicate data among a cluster of nodes. How replication works in CouchDB is over the concept of a “Shard” and “Replica” pair which we are free to choose as we desire in our database design. A shard is a horizontal partition of data in a database. A replica is a copy of each shard in the database cluster. Partitioning data into shards and distributing copies of each shard to different nodes in a cluster gives the data fault tolerance against node failure. How many shards and replicas we require could be decided at global level or at each database level. Even though depending on the size of the cluster and the shard replica pair defined, every node may not have access to every shard, every node knows where all the replicas of each shard can be found through CouchDB’s internal shard map.

CouchDB documentation states certain best practices with regard to specifying the number of shards and replicas in a database cluster. The shard replica pair has to be selected in a way so that shards are distributed evenly among the available cluster nodes. Since we have decided on a three node cluster, we have set the number of shards to be 3 and the number of replicas to be 2. This means each of our databases would have three horizontal partitions while each partition would have two copies.

Accordingly, one node in our cluster could fail without any impact to the system as a whole, resulting in high availability of our storage system.

Figure 3 provides a sample distribution of the 6 (shards\*replicas=3\*2) shard replicas in our database cluster.

A, B, and C represent the partition of each database into three equal segments.



*Figure 3: Distribution of shard/replicas across cluster nodes*

Advantages of using CouchDB for data storage:

1. Built in features of replication enabled with cluster mode of CouchDB which enables fault tolerance.
2. The twitter harvesting application extracts tweets in high volume and high velocity. CouchDB's clustered architecture can easily handle a high volume and high velocity of data.
3. Information available in tweets can vary for each tweet. For example some tweets do not have coordinates attached to them. Since CouchDB stores data as JSON files, veracity of data can be easily handled.
4. CouchDB's document based data structure enabled us to avoid duplicate tweets by enabling us to save tweets' Id parameter as the document Id, which refuses update of any document with the same Id.
5. CouchDB has built in support for Map reduce algorithms which makes tweet analysis and filtering easier.

6. CouchDB's support for REST API, makes the front end communication with the database much easier

## 2.2. Twitter Harvesting Applications

We have developed the Twitter harvesting application using Python programming language. The twitter API was accessed using the **Tweepy** python library. The harvesting application for tweets mainly consists of 3 **Streamers and Search API's** on each of the 3 nodes assigned to the cluster DB. The parameters related to Authentication and Location are given via the Configuration files.

Below are the essential elements of Twitter harvesting application that we have used in our project -:

### Authentication

For harvesting the tweets from Twitter, we created Twitter developer accounts and registered our application to generate authenticated credentials required for interacting with the Twitter API (Consumer Key and Access Token Key). Therefore, we authenticated our requests for harvesting the tweets using the oauth authentication scheme provided by Tweepy. The method used by our application for authentication is application-only authentication because using this approach we can harvest tweets at a faster rate.

### Overcoming rate limit restrictions

There is a rate limit imposed on the total number of requests that is made in a day to the Twitter APIs. Different authentication methods have variant rate limits as defined by Twitter for example- using User access based authentication only 180 requests are permitted in 15 minutes duration whereas in Application based authentication, we can make 450 requests in the same time which is significantly higher. Also, to fetch more tweets, we created 3 different harvesting applications, one on each node. Due to this, we adopted the Application based authentication approach allowing us to fetch maximum tweets overcoming the rate limit restrictions by deploying multiple harvesters.

### The Tweets Harvesting APIs

The Streaming API was used to harvest the real-time tweets of users belonging to a defined geographical area (given as a parameter by our config file) and the Search API to harvest tweets related to a particular keyword and geographical area (both keyword and geographical area given

as parameter). The format of geographical area given for both the APIs consists of different details like- for Streaming API the geographical area is defined by a bounding box consisting 2 latitude and longitude pairs to describe the coordinates of the box whereas for Search API the geographical area details consisted the latitude and longitude value for the centroid required along with its radius. The Search API returns the tweets that lie within the given radius of the centroid. We excluded all the retweets by using the parameter **-filter:retweets** because it prevents the reharvesting of tweets in the event of abrupt termination of the harvesters.

Mainly, our area of search was centered to top regions in Victoria and for each of these regions, our Search API was centered in a way to cover the maximum part of those regions by adjusting the radius of the centroid. Therefore, each harvester was assigned a big region based in Victoria. For harvesting twitter messages we divided our area of interest into 3 regions. To ensure that we do not miss out on the tweets in nearby areas, we gave the radius of the intended area to be sufficiently large enough. Through this approach, we are able to extract tweets related to the major areas and any overlap of tweets due to duplicates is taken care of by the logic of handling duplicate tweets (explained below).

### **Duplicate handling**

As we have multiple streamers and searchers running simultaneously, this can arise an issue of duplicate tweets. Each tweet consists of a Tweet ID which is a unique 64-bit unsigned integer, generated on the basis of the time at which the tweet was posted. The string version of these tweet IDs are also returned in the json for each tweet. While saving tweets as documents in the CouchDB cluster we used these unique tweet IDs as the document ID, as the Document ID(\_id) is automatically set to be unique. So if an attempt to save a duplicate tweet is made, i.e. a tweet having a Document ID which is already saved in the CouchDB, an error code of HTTP 409 (Document update conflict) is returned which is passed by the exception handling in the harvesting application. Thus, using Try- Except cases in our python code, we incorporated the same concept to our application and hence solving the duplicate tweets issue.

### **Data Storage**

To further analyse our harvested tweets, we needed to first store them in a uniform and standard format in the clustered CouchDB that existed over three nodes running in three different NECTAR instances in our application.

While having too many nodes in the cluster DB can make the system complex, degrading its performance but at the same time, having too few nodes can lead to less fault tolerances leading to more system downtimes. Therefore, it's essential to select the appropriate number of nodes in a cluster to ensure a good performance. Here we had a total of 4 nodes available to us with



CouchDB being the central driving unit, therefore, having 3 cluster nodes improved the performance of our application.

The CouchDB python library was used to handle the saving of tweets to the database and the tweets were saved in the JSON format as retrieved from the Twitter Stream and Search API. For streaming, we had to save each tweet as they are fetched, as the streamer returns one tweet at a time. For that, we overloaded the ***on\_data*** function of the ***TweetStreamHarvester*** class. Using the ***Server*** method of the Couchdb library, we initialised/opened connection to our Couchdb server. In the *Server* method we need to pass our username password along with the IP and port number as parameters. We created a Database in the Couchdb, and made an object for that particular Database in our python program. Using the *save* method of the Couchdb library we save the tweet json on to our database. Each tweet becomes a new json document in our database. For searching, as the tweepy *search* method returns a list of maximum of 100 tweets, we had to iterate through that list and save each tweet similarly as explained above. To keep the format of output for both Search and Stream similar, we use the ***JSONParser*** in the ***tweepy.API*** method.

## 2.3. Data Analytics

Twitter is one of the largest social media platforms that forms an integral part of where people express their opinions and views. This often reflects their different situations, lifestyle habits, related surroundings and experiences. Thus it makes it worthwhile to use twitter data for social media analytics to identify the major trends and gain insights, which further can be used to improve the associated processes by companies or even the government. With this mindset, we intensively conducted exhaustive harvesting of twitter data to carry out our analysis by leveraging various data analytics techniques using Python libraries and CouchDB MapReduce functionality.

The detailed explanation of the analysis process and components are described below.

### 2.3.1. Scenarios

We have picked up the most popular and essential topics that are widely discussed generally and conducted analysis on them to generate insights and validated them with AURIN data. The four scenarios/stories that we have worked upon are as follows:

- **Pollution and its effects on the Health of people in an area**

Pollution is one of the major concerns today and twitter is a platform where people share their concerns and opinions related to it. While Victoria is a relatively green region, people

living near to industrial areas, in heavy traffic zones or in nearby construction sites have expressed their concerns on the matter. We have conducted analysis on the tweets to extract and study all tweets related to pollution. Further, we verified and validated our analysis from the official AURIN data related to Carbon Monoxide emissions in various areas of Victoria. We have also related this to the health of people living in high polluted areas to analyze the correlation of pollution with its impact on the health of an individual.

Therefore in this story we are determining the areas where people are talking more about pollution, categorizing them into various zones and comparing it to the AURIN data to depict the relationship between pollution and health of people in those areas.

### **Twitter Data**

The tweets related to emissions, air quality, air pollution, smoke, dust, etc are filtered and their geocodes (Coordinates) are mapped to compare with the AURIN data reflecting total air emissions for all areas.

### **Health Analysis**

For this we have utilized the AURIN dataset having a total percentage of people who visit General Practitioners (Doctors) to observe the correlation of pollution level and health of people in different areas.

- **Heavy Traffic impacting total Accidents in an area**

Tweeting about busy roads, frustration in traffic jams, areas to avoid driving through due to choked roads, accident prone routes is quite commonly observed on twitter. We have filtered such tweets related to traffic to analyze the correlation of heavy traffic zones with the number of accidents/ crashes (both minor and fatal). For the purpose of validation of our analysis with AURIN data we have used two datasets from AURIN - the first dataset comprises mainly the total average volume of daily two way traffic for the various roads in Victoria, and the second one consists of the total number of car accidents in various regions in Victoria. This scenario will categorize the tweets related to traffic into various regions of Victoria and compare them with the AURIN data to depict the relationship between traffic and accidents.

- **Luxurious and Outgoing lifestyle in Wealthier areas**

It's a common trend for people to tweet whenever they visit a new place, restaurant or buy something new. Often people also share their opinion on different places, food, etc. Today social media is vastly by people to display and show the world their lavish and outgoing lifestyle. These tweets very much reflect the materialism, competition to be in

the highlights of their circle which directly indicates their income or is determined by their income level. This scenario categorizes such tweets into various areas in Victoria and analyses them to depict the relationship of people in such areas with the average income of people in those areas.

We have utilized the AURIN dataset for people in various areas of Victoria having weekly salaries greater than \$650 considering that this amount is the basic minimum for a person to afford this lifestyle.

- **More Healthy habits and Environment consciousness in Greener areas**

People often tweet related to their regular eating and exercising habits mentioning details like running in parks, meditation, hiking, healthy eating, cycling etc. Also discussing consciousness and efforts made to save the environment for example, solar energy, water conservation, etc. We filtered tweets consisting of such information, categorized them into areas and compared them with the AURIN data consisting of total green spaces in different areas of Victoria. This scenario analyses the relationship between regular healthy habits, environment consciousness of people and depicts their relationship with the type of area they live in.

### 2.3.2. Filtering Tweets

All tweets corresponding to the searched area are harvested and saved in the clustered database, as a preprocessing step of our analysis, we have filtered all the tweets that are related to any scenario and segregated them in different categories based on the scenarios they belong to. These tweets contain all the information like text, location, bounding box, #hashtags, etc, but primarily we worked on the tweet text and their location related attributes. We have defined a view for each scenario in the CouchDB consisting of a Map function to filter the tweets. A list of keywords related to the scenarios that are identified after tokenization and lemmatization of tweets text is given in the Map function as a parameter. Two separate keys, one keeping count of related tweets having keywords from the keyword list, and another for keeping count of relevant and geocode enabled tweets for all scenarios. The views generated consists of the mapped tweet id along with the tweet text and geocode (if enabled) .

Below is the screenshot of the cumulative keywords used for filtering the tweets (for all scenarios).

```
"searchKeywords" : ["pollution","cleantraffic","polluted","air emissions","ozone","particles","dust","smoke",
"smoking","fumes","ozone","layer","uv","litter","dirt","garbage","fire","rotten","trash",
"leakage","carbon","deforestation","harmful","industries","factory","quality","pollutant",
"cigarette","drink","party","booze","dance","wine","alcohol","dinner","party","stuff","lavish",
"happy hours","pizza","burger","weed","restaurant","pub","bar","eat","rich","luxury","holiday",
"vacation","extravagant","expensive","gift","shopping","pleasure","concert","theatre",
"celebration","grand","music","jewellery","tree","plant","garden","conserve","exercise","green",
"natural","fit","run","environment","health","fresh","workout","weather","climate","water",
"vegan","save","earth","solar","energy","peace","yoga","park","nature","clean","collision",
"accident","crash","overspeeding","speed","road","rash","driving","fine","infringement",
"traffic","ticket","license","suspend","demerit","points","fatal","drunk","driving","dui",
"vicroads","emergency","crossing","traffic","blind turn"]
```

Screenshot 1: Keywords list for filtering tweets of all scenarios

```
if(res27){
    emit("pollution",[doc.text, doc.geo])
}
if(res28){
    emit("pollution",[doc.text, doc.geo])
}
if(res29){
    emit("pollution",[doc.text, doc.geo])
}

if(res1 && doc.geo != null){
    emit("pollutionGeo",[doc.text, doc.geo])
}
if(res2 && doc.geo != null){
    emit("pollutionGeo",[doc.text, doc.geo])
}
if(res3 && doc.geo != null){
    emit("pollutionGeo",[doc.text, doc.geo])
}
```

Screenshot 2: Map function for filtering and mapping of pollution related tweets

### 2.3.3. Analysis

The analysis that we have conducted consists of 3 parts- first is the exploratory data analysis and second is the SA2 geographical location identification (using geopy python library) for the tweets and third is the validation with appropriate AURIN datasets.

- **Exploratory Data Analysis**

It usually consists of the critical process to perform low level investigations for discovering patterns in text and to carry the initial test of the hypothesis based on the assumptions using statistical and graphical representation. As a very basic step for any Data Analysis and Natural

Language Processing application, we extracted all the tokens related to our scenarios from the tweets text to form a list of relevant keywords. All the keywords were lemmetised to convert them in a standard form. We used the python library NLTK to perform these basic operations and the python JSON library to handle the JSON files.

- **Location Identification**

After obtaining the exhaustive keywords list from the above process, the resulting list was divided into sublists relevant to different views as well as augmenting more related words to the scenarios. These sublists were used to generate the mapping function for initial filtering of the tweets on the basis of the keywords.

However, it was observed that many tweets did not have specific geo location assigned to them because of the twitter privacy policy. They only had the bounding box assigned to them which covered the latitude and longitude range for the entire city which did not help us in deriving the hypothesis. For these tweets, we carried an in-text based filtering by searching for locations in hashtags or tweets text (example- many tweets comprised #MelbourneCBD and @HoppersCrossing in the tweets text). Such tweets were determined and they were assigned appropriate SA2 geographical location.

After the above step, all tweets having geocodes (i.e. geographical coordinates) were mapped to their SA2 geographical area. For this purpose, we used the Python package- **Geopy** which has functions to map coordinates to the exact addresses and reversely, addresses to coordinates as well.

We calculated the distribution count of tweets for each scenario belonging to different areas to get the most popular tweeting areas and further, validated our result by correlation with the top areas from the AURIN files related to that scenario. The AURIN files handling and validation has been explained in detail below.

- **Validation with AURIN Data**



There were six AURIN files to support and validate our analysis for all scenarios. These files were extracted as JSON from AURIN and their relevant attributes were saved to the clustered database. For standardization across the Database and better analysis, the individual records for all AURIN files were also mapped to the SA2 geographical area on basis of the **Geometry** attribute present in these files. The geometry attribute essentially consisted of the latitude and the longitude information.

The SA2 areas identified in the tweets data is compared and validated against the relevant AURIN data to derive the hypothesis as stated in the Scenarios section.

## 2.3.4. Results

### 2.3.4.1 Mapreduce based results

MapReduce capability provided in CouchDB is used for further analysis of generated results. MapReduce is a programming model that consists of a map procedure, which sorts data into key/value pairs and a reduce method that performs summary operations on mapped data. This concept is used in calculating the total number of tweets and geo enabled tweet count for each Scenario.

key	value
 pollution	5112
 pollutionGeo	101

*Screenshot 3: The result of running MapReduce function on the Pollution view*

Reduce function uses the in-built function `_count` and outputs two counts - The total tweets belonging to a Scenario and the total geo enabled tweet in the Scenario , so that the frontend can directly use the reduced data for visualizations.

### 2.3.4.2 Analysis results from AURIN Validation

The hypothesis is finally verified against the validated AURIN data to make a conclusion. For this purpose the identified SA2 geographical areas for tweets is mapped and compared to the SA2 geographical areas of the AURIN data. For example, the densely polluted areas are compared with highly polluted areas as per AURIN data, and the identified and verified highly polluted SA2 areas are then related to the total proportion of people spending money on General Practitioners in the various SA2 areas. As a result we observed that in high polluted areas people visit the doctors more often and are comparatively unhealthy. Similarly the same process is applied for all scenarios and their respective visualizations and conclusions are given in the Section 3 of our report.

## 2.4. Front End

JavaScript frameworks have risen to popularity in a past couple of years and since become a rewarding skill to possess. Owing to this and the fact that they are (mostly) open-source and well paired with Linux distributions, our team decided to implement the web app using two such frameworks, namely Node.js and Angular.

We wanted to incorporate best practices in our system i.e. restrict the front-end interacting directly with the CouchDB database and hence the inclusion of the Node.js back-end API.

## 2.5. Scripted deployment capabilities

The project is hosted on the Melbourne Research Cloud/Nectar, which is an open source Infrastructure as a Service (IaaS) platform, based on OpenStack. NeCTAR provides a centralized dashboard through which we can access the network, storage and processing resources.

Using the provided dashboard of Nectar, it is possible to create instances, volumes, security groups and configure them manually as required.

But for scaling purposes, using a configuration management tool for server configuration brings about many advantages where quick provisioning of new servers to handle increased data loads and quick recovery from critical events stand at the forefront.

As such, Ansible is one such open-source automation tool, or platform, used for configuration management, application deployment, intra service orchestration, and provisioning of resources. The Ansible scripts are called Playbooks and are written in YAML, that follows a structured approach, similar to a directory hierarchy.

In each Ansible playbook, the variables are set and defined first, which include constant file paths and folders, instance flavors and Image IDs, the security groups and rules, python versions to be used etc. This would be followed by the roles which are the sub-scripts defining various tasks in the playbook. The inventory or hosts on which the scripts are executed would be included in a file separately. Finally, the YAML playbooks are executed sequentially and this helps in identifying failure points and singling it out for debugging.

For our project, we have developed Ansible playbooks, which provide automation capabilities for the following tasks:

1. Creation of security groups, volumes, and four instances on the Melbourne Research Cloud which are added to the allNode group
2. Setting up required proxies, formatting and mounting volumes on each of the created instances
3. Installing CouchDB and its dependencies on three nodes and adding each node to the cluster
4. Deploying the harvester applications on each of the three clustered nodes

## 2.6. Source Code

Please find below the link to github repository.

**Link to the source code:** [Link](#)

## 3. Functionality of the System

The primary functionality of this system is to analyse the tweets and verify their relationship with various scenarios to give an insight about the lifestyle, daily-life routine, and habits of people living in Victoria. The twitter activity in Victoria is extremely high and people discuss varied topics therefore, enabling multiple researches on the tweets. We have picked some of the most fundamental yet crucial subjects to people including - Pollution, Traffic, Lavish trends, and Health to analyse impact of/on these subjects to daily life of people in Victoria.

**The video to see the functionality of the system:** [Link](#)

In total the tweets database consists of 215,704 tweets on which the analysis is conducted and below are all the AURIN files used for authentication and validation.

AURIN File Name	Associated Story Category
National Pollutant Inventory-Emissions(Point)2018	Pollution Story
AIHW-Patients Spending on Medicare-Out of POcket Cost per GP Attendants	Pollution Story
VicRoads - Traffic-Volume(Polyline)2017	Traffic Story
VicRoads - Crashes(Point)2012-2016	Traffic Story
SA2-P17c Total Personal Weekly Income by Age By Sex	Lavish Story
PSMA - Environment(Point)(August 2018)	Environment Story

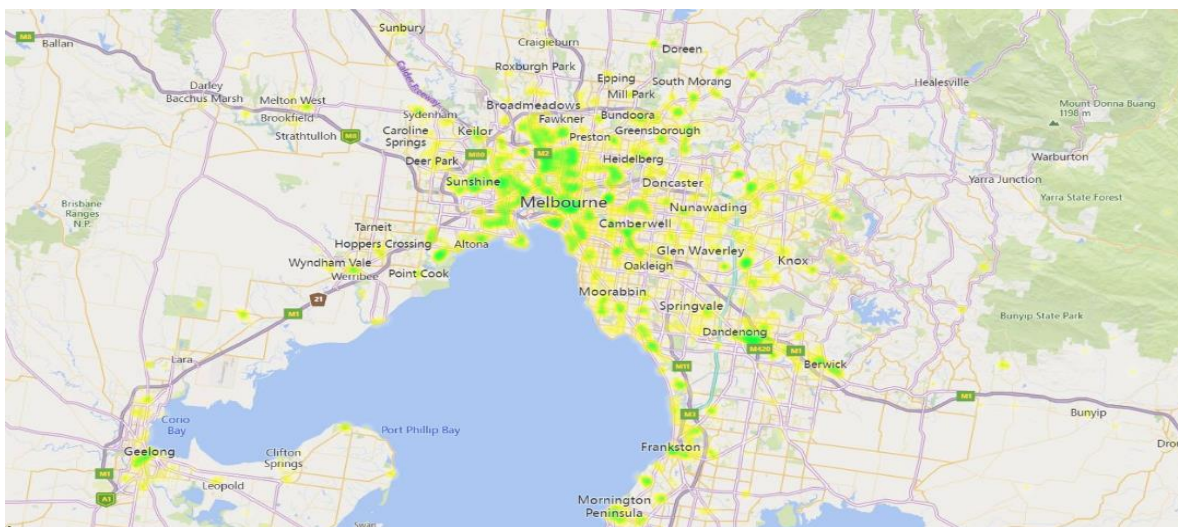
The in depth analysis result and visualization for each story has been explained below :



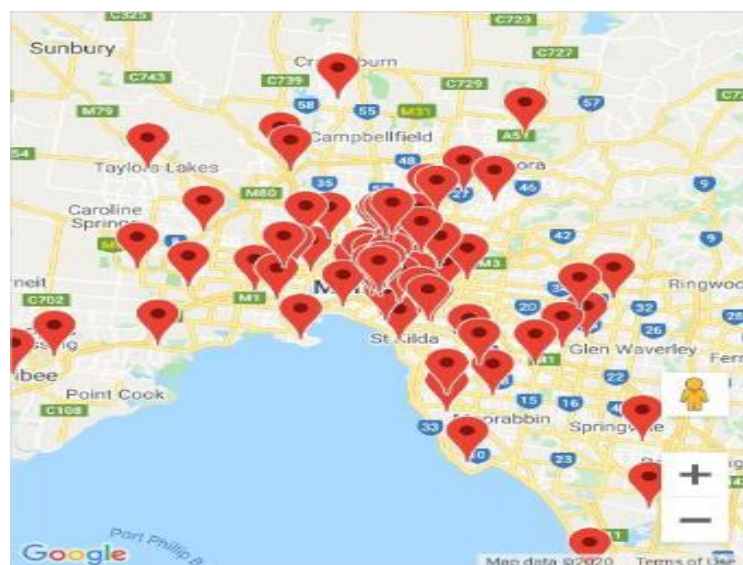
## Story 1 - More Healthy habits and Environment consciousness in Greener areas

In order to identify the relationship between people with healthy habits and environment consciousness the filtered twitter data set relates to words like water conservation, energy conservation, exercise, parks, solar energy, clean environment, good air etc and AURIN data set covers the number of green spaces in various areas. To analyse whether people in green areas have more consciousness and consideration for their health, we have plotted the below graphs-

- A heat map in tableau to represent the total number of greenspaces i.e. parks, gardens and reserves in various regions (on degree of their count) of Victoria as per the AURIN file. More heat represents that the area has more density of green spaces.



And the corresponding twitter dataset for this story is-

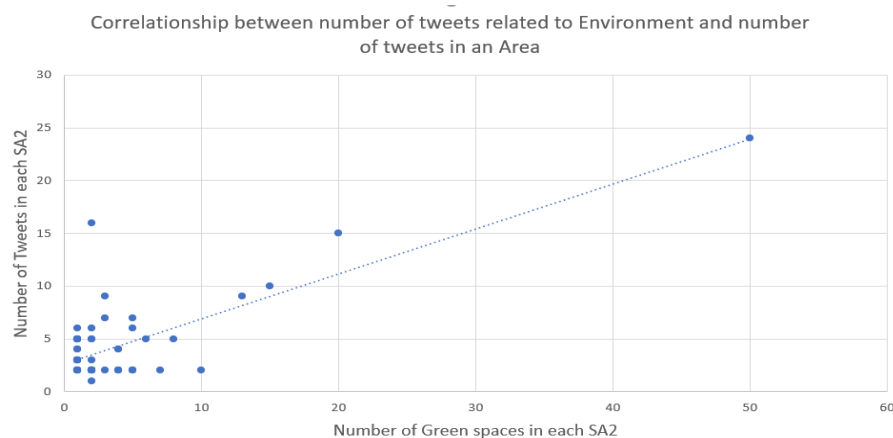


It can be clearly seen that the areas like- Werribee, Dandelong, Sunshine and Parkville have high density of both the tweets and the greenspaces. While as we go outwards, both the number of tweets and greenspaces decrease in density.

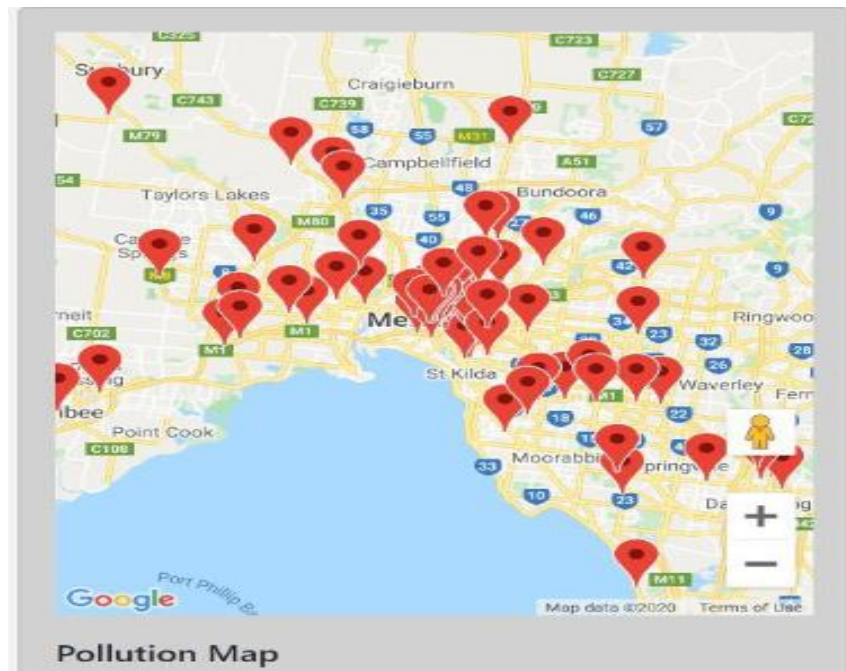
- A graph showing the plot of the top 5 areas in Victoria having the greatest number of greenspaces.



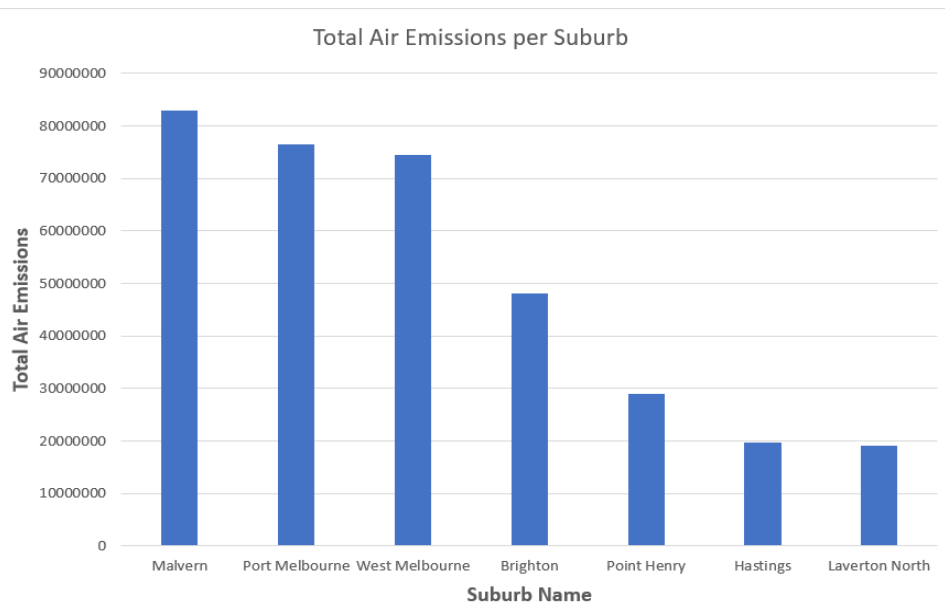
- For determining whether tweets related to environment and health are actually related to areas that have more greenspaces, we used the following scatter plot having Number of tweets and Total number of greenspaces in the several SA2 geographical areas.



- The location of tweets related to pollution.

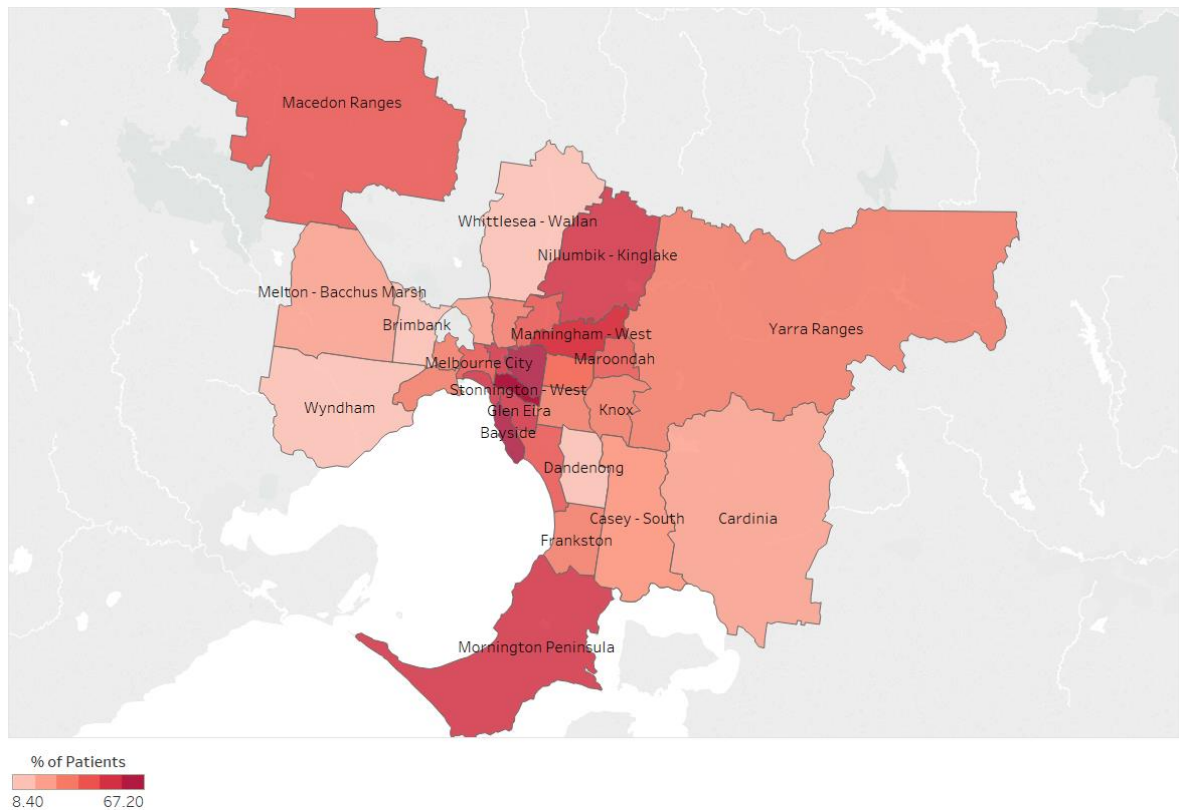


Top 7 polluted areas in Victoria as per AURIN



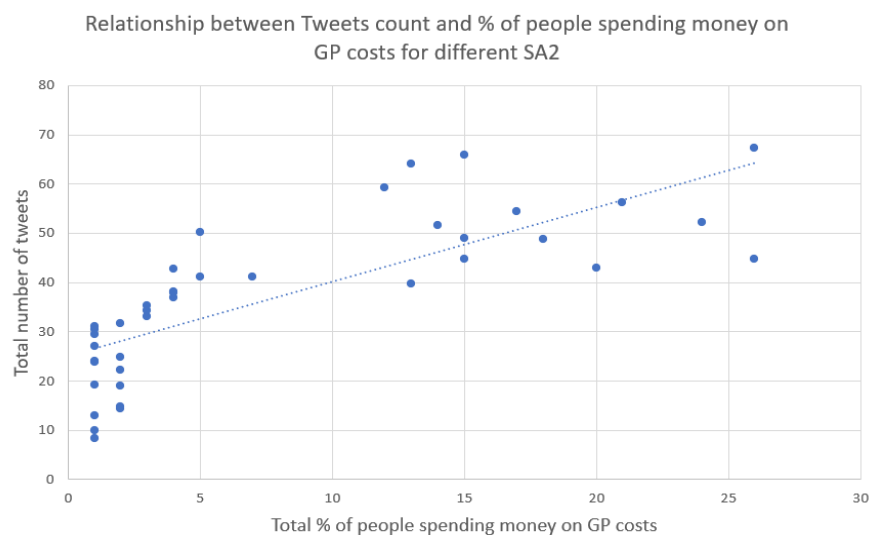
As per the Twitter google map, the density of tweets is more in the areas Malvern, Abbotsford, Port Melbourne, Brighton which is also reflected by the AURIN data(Large emissions of Carbon Monoxide) in these areas

- The health of people in an area is directly related to the number of people spending their money and visiting GPs. The filled map for this is made in Tableau and is given below



As it can be clearly seen that council areas like Mornington Peninsula, Stonnington, Nillumbik, Bayside, Melbourne City, and Carlton have a high percentage of patients spending money on GPs which also overlaps with the areas of high pollution.

- To further determine the relationship between pollution level and health of patients in an area, we have plotted the below graph showing the number of tweets related to pollution vs percentage of people spending money on GPs in each SA2.



From the above graph, it can be clearly seen that more pollution related tweets exist in areas where people spend more on GPs thus, there is a direct relationship between pollution and health of people.

Similarly, the analysis has been conducted for the other two stories.

## 4. User guide

Our project could be deployed either via running Ansible scripts or by creating and configuring the NECTAR instances manually. README files included in the repository, for each of the applications/Ansible scripts, provide guidelines on how to deploy the overall project components.

To run the Ansible scripts, the NeCTAR OpenStack API password and `./openrc.sh` file for our project are required. Once the instances are created, the private key associated with the instances is needed to start a SSH connection and log in to the nodes remotely. Since the IPs allocated to the instances are private, the user would need to be first connected to the Unimelb VPN network in order to start a SSH connection. (The OpenStack API password, `./openrc.sh` file for our project and the private key files for the created instances are provided in the submitted project folder)

Once the instances are created and volumes are mounted and formatted, a CouchDB cluster is created on three nodes. Then the twitter harvesters could be deployed on the clustered nodes to be run as background services, allowing them to retrieve and populate twitter data onto the database. Finally the front end application is to be deployed on the fourth node. The views set up on the tweets database are accessed by the front end web service that generates graphs and visualization along with the AURIN data resources, also present in the instances.

## 5. Using the Melbourne Research Cloud

### 5.1. Advantages

Melbourne Research Cloud provides cloud resources at hand, with a user friendly dashboard to create and configure instances. Since the infrastructure needed not be maintained on our own, we could focus more on the functionality of the project. The availability of a multi node infrastructure to deploy our project as a distributed system helped us gain fault tolerance in many aspects. In addition, having a common platform in cloud for all the group members to simultaneously access and work on created a good learning environment and a method to easily share each other's work. Additionally, the Research Cloud facilitates users to incorporate an authentication scheme of their choice, hence providing a mechanism of preventing unauthorized access to data and ensuring data safety.

## 5.2. Issues and Challenges

Although, having a common platform which could be accessed by all the members eases a lot of work, at the same time it could create complications as well. For example at times multiple members were working on the same nodes doing simultaneous changes creating conflicts and also resulting in loss of certain work. Also since we had limited resources on the cloud, using the nodes to collect twitter data and at the same time, using the same resources for testing infrastructure creation, i.e testing Ansible scripts, needed to be managed in parallel.

Since the Unimelb Research Cloud is a subset of the NeCTAR cloud service, the IP addresses of the instances created were not public and were restricted to only Unimelb IP addresses. When working remotely to access the created instances through VPN, network interruptions were seen at times.

Also complications arose when trying to create instances on the cloud which uses the OpenStack API, where it was not compatible with certain versions of python installed on the host machine.

## 6.Team

Our project team includes four members doing different courses/degrees at University of Melbourne.

Since many technologies required for the project such as CouchDB and its functionality, using the Twitter APIs and the python wrapper 'tweepy' and the Ansible configuration management tool, were new to all of us, we spent some to get an initial understanding on them. Afterwards we had many collective discussions on the architecture and the design of the overall project. Moving ahead with the work allocations, we equally divided the work as per the knowledge base of team members. The work allocations are presented in below table, which gave us the opportunity to learn the particular assigned area in depth and also to be responsible and accountable for the work.

Team Member	Contribution
Anupama Mampage	Handling the infrastructure set up -Learning CouchDB related theories, deciding a suitable cluster configuration, manually setting up and testing the cluster -Manually creating and configuring the NECTAR instances for testing and documenting all the followed steps -Automation of the deployment using Ansible scripts to create and configure instances, setup CouchDB cluster, and deploy harvesters on the cluster nodes -Documenting the report on the infrastructure/deployment



Ribhav Shridhar	<ul style="list-style-type: none"> <li>- Handling the Twitter Streaming API related part and deploying them on nodes</li> <li>-Creating views for Twitter data for analysis using MapReduce</li> <li>-Developing and Implementing the Location Mapper logic for twitter data</li> <li>-Developing and implementing the logic for exploratory data analysis (tokenization and lemmatization of tweets)</li> <li>-Documenting all the steps for the report</li> <li>-Creating visualisations for the application</li> </ul>
Nitika Malhotra	<ul style="list-style-type: none"> <li>-Developing the logic and code for Twitter Search API and its deployment on nodes</li> <li>-Working on the scenarios required to research and determining the associated AURIN files</li> <li>-Saving the AURIN files to database in suitable structure and AURIN data analysis including the location mapping for standardisation of all AURIN data, its relationship with the twitter data</li> <li>-Documenting the report</li> <li>-Creating visualisations for AURIN data</li> </ul>
Ronak Arvindkumar	<ul style="list-style-type: none"> <li>- Front End development in Angular 7 displaying the results of the analysis</li> <li>- Implementing various front end features</li> <li>- Back end development in NodeJS for fetching data from couchdb views</li> <li>- Setting up connection with the CouchDB views</li> <li>- Deploying the front end and back end on the instance</li> </ul>

Although work was divided among us, we had regular meet ups via zoom, twice a week from the start of the project work till the end to discuss any issues each of us were facing. Many complications arose from time to time with regard to setting up the infrastructure and also harvesting tweets. One main challenge that we faced was with regard to lack of data in the AURIN database which would relate to the sentiments expressed in the majority of the harvested tweets. Also the lack of location based data in most of the collected tweets made it impossible for them to be used for comparison purposes along with AURIN data. These resulted in us having to change the scenarios we initially thought of, from time to time.

At the beginning itself, we created a git repository where each of us could easily make our contributions to the project and work on our respective parts in parallel.

## 7.Conclusion

This project provided us with hands-on experience of using cloud infrastructure to design and deploy a real time big data analytics system. As such we got the opportunity to effectively put into practice, the theoretical concepts learnt in class. While engaging in this project we also experienced the practical use of cloud computing when multiple developers are simultaneously collaborating on the same application.

However, the system that we have implemented can be further improved by incorporating a mechanism for dynamically scaling the system up and down based on load and also by further improving fault tolerance.

Also, it was proven that the careful analysis of social media data could provide valuable insights into the lives of citizens of a particular geographical area.

Finally, we would like to extend our sincere gratitude to Prof. Richard Sinnott and the team for allowing us to acquire great hands on experience related to Cloud Computing through this project, and for assisting us constantly with the numerous queries raised throughout the project.



## References

- [1] Apache CouchDB® 3.1.0 Documentation. *Overview — Apache CouchDB® 3.1 Documentation*. [online] Available at: <https://docs.couchdb.org/en/stable/> [Accessed 2 May 2020].
- [2] Tweepy. *Tweepy Documentation — tweepy 3.8.0 documentation*. [online] Available at: <http://docs.tweepy.org/en/latest/> [Accessed 26 Apr. 2020].
- [3] Ansible.com. (2017). *How to build your inventory — Ansible Documentation*. [online] Available at: [https://docs.ansible.com/ansible/latest/user\\_guide/intro\\_inventory.html](https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html) [Accessed 7 May 2020].
- [4] AngularJS. (n.d.). *AngularJS*. [online] Available at: <https://docs.angularjs.org/api> [Accessed 18 May 2020].
- [5] Unimelb.edu.au. (2019). *Melbourne Research Cloud Documentation*. [online] Available at: <https://docs.cloud.unimelb.edu.au/> [Accessed 23 Apr. 2020].