THE UNIVERSITY OF

MELBOURNE

Cluster & Cloud Computing Assignment 2

Group: 48

Sergey Germogentov:	893900
Haaris Nazir Ahmad:	869969
Kriti Bhardwaj:	880873
Muhammad Umair:	863579

Acknowledgement

We would like to thank the professor and his team of excellent resources for assisting us in understanding the principles of cluster computing as well as exploring the various technologies to achieve data analytics. From the beginning, the encouraging conversations with a light-hearted undertone observed on LMS with the staff team was one of the contributing factors in keeping us focused and in a positive morale.

Project Resources:

This section lists all the resource links as per the requirements of the project.

1. **Bitbucket Source code link**: https://bitbucket.org/comp90024twitterresearch/twitterproject/src

2. **Youtube video upload links:**

   System Demonstration

   https://youtu.be/2XJQY2C8fMo

   Automation

   https://youtu.be/30Pd6GcRe_E

3. **Application Server**: NeCTAR - 115.146.93.96

4. **Database Server:** NeCTAR - 115.146.95.134

Table of Contents

1. Introduction 1
2. Architecture and Design 2
2.1. Architecture 2
2.1.1. Presentation Tier 2
2.1.2. Logic Tier 2
2.1.3. Data Tier 2
2.1.4. Nectar Cloud 2
2.2. Design 4
2.2.1. Scalability 4
2.2.2. Availability 4
2.2.3. Security 4
2.2.4. Manageability 6
2.3. Fault Tolerance 6
2.4. Single Point of Failure 6
2.5. Limitations 7
3. Data Extraction 7
3.1. Hypothesis – Most Liveable City (Variance in Suburbs) 7
3.2. Harvesting of tweet data 8
3.2.1. Harvesting using location filtering 8
3.2.2. Cleansing 8
3.2.3. Sentiment Analysis 9
3.2.4. Tagging 9
4. Data Analysis 10
4.1. General Overview of Twitter Activity 10
4.2. Visualization against AURIN data 12
4.2.1. Location versus Happiness 12
4.2.2. Marital Status Versus Happiness 14
4.2.3. Born Overseas percentage versus Happiness 16
4.2.4. Top 5 Happiest and Unhappiest Suburbs 18
4.3. Visualization of Lifestyle and Quality aspects 19
4.3.1. Crime Visualisation of Greater Melbourne 19
4.3.2. Drug and Alcohol Visualisation of Greater Melbourne 21
4.3.3. Sports Visualisation of Greater Melbourne 22
5. Cloud and Scalability 23
5.1. The Use of the Cloud 23
5.2. Elasticity 23
5.3. Scalability 23
6 Automation 24
6.1 Launching NeCTAR instances 24
6.2 Configuring the instances 25
7. Error Handling 27
8. Security 27
9. Availability 28
10. Team Dynamics 28
11. Conclusion 29
12. References 30

13. APPENDIX 31
13.1. Appendix A: Couch DB Architecture 31
13.2. Appendix B: AURIN Datasets 32
13.3. Appendix C: Couch DB Views 33

1. Introduction

In this age of technology, social media is the platform of choice for communication and keeping ourselves updated with the latest trends and news. However, it also provides the opportunity to extract data and knowledge about people and their opinions and lifestyles. Almost anyone can extract information and form patterns to a user's life. Some people believe that social platforms at times violate a user's privacy and it is fairly common to see such companies going under investigations by privacy groups. However, we believe that social platforms that allow us to extract knowledge from our social lives are a benefit to the society as they provide us with vital information whether related to a region, a city or even a place. A city like Melbourne attracts thousands of tourists every year and one such use case would be to understand the basic reason of why tourists come to Melbourne. Do tourists, students and temporary residents feel safe and happy in a city like Melbourne? Do residents feel safe living in a particular suburb of Melbourne? Do people feel satisfied living in a larger suburb as compared to smaller or coastal areas? Performing analysis on simple information extracted from various social platforms would provide us the answers for all these questions.

Twitter is one such platform where people share 'tweets' – simple messages which can be opinions, activities or even photos and videos. It has a large user base all over the world and an active user base of approximately 3 million accounts within Australia[1]. With its varied users, Twitter is one such place to find answers to all those questions. To extract user data, the platform has a public API which allows applications to access twitter data in the form of real-time streams or random searches based on location and keywords. Relying on the user data provided by twitter and creating custom scripts in Python and then utilising extensive data analytics tools like CouchDB and MapReduce to process that data, we can easily extract knowledge and find patterns to a user's behaviour. NeCTAR is used as the platform of choice for providing computational resources.

This study initially tries to find answers to the following questions:

> **Hypothesis 1**: Are people living in suburbs happier around the coast happier than inner city suburbs?
>
> **Hypothesis 2**: Are suburbs with higher percentage of married people more positive?
>
> **Hypothesis 3**: Does increase in percentage of people born overseas in a suburb have a impact on happiness level?

Furthermore, we also look at the Greater Melbourne suburbs from the perspective of other determinants of life quality and happiness including crime, prevalence of drug and alcohol usage and involvement in sports - a positive aspect of life.

> **Hypothesis 4**: Is it possible to identify crime hotspots in Greater Melbourne using Twitter?
>
> **Hypothesis 5**: Are inner city suburbs the hub of drug and alcohol related activity?
>
> **Hypothesis 6**: Are sports activities localised or equally spread over Greater Melbourne suburbs?

The report lists the technologies used to gather, process, and analyse datasets and then describes in detail the results and our findings. The results may assist us in understanding more about the user behaviours on a suburb level in Melbourne. We also analyse the sentiments of the tweets to extract the actual emotions from the messages which would assist us in the analysis.

2. Architecture and Design

The sections below elaborate in detail the working structure of each component, how they interact with each other and their specifications at module level.

2.1. Architecture

The architecture follows a 3 tier implementation comprising the following.

- Presentation Tier
- Logic Tier
- Data Tier

2.1.1. Presentation Tier

The user interface layer, being the top-most one, facilitates end user through a responsive web application accessed through any standard web browser. The technology used for front-end implementation is HTML/CSS/Java-script and a Bootstrap framework for responsiveness. A JS library 'Highmaps' and 'Highcharts' has been used for analytics at front-end (*see presentation layer Figure 1.0*).

2.1.2. Logic Tier

The logic layer serves as an intermediary between data source and front-end. It runs on the application server. An apache web server receives a request from client, it uses PHP to communicate with database via REST API and sends the response back to front-end. At this tier, the logic for transformation of data exist. These include but are not limited to calculations such as percentiles, total counts etc. (*see logic layer figure 1.0*).

2.1.3. Data Tier

Database: At this tier, data is stored and retrieved from database. The information here is passed to logic tier and then to the presentation tier eventually. A CouchDB's instance has been used for storage and retrieval purposes. CouchDB provides a RESTful API for data manipulation. These native APIs have been used to query the database and retrieve aggregated results via map-reduce views.

Data-source: This tier is also responsible for fetching data from twitter API, transform it, perform sentiment analysis, append flag to it based on keywords and store it in the database. Data from AURIN was retrieved and dumped in the database from application server as a one-off activity. Since AURIN data is used as a metric for analytical purposes, this was performed as a one-time activity.

2.1.4. Nectar Cloud

For implementing architecture, Nectar Cloud was used as a platform. The second and third tier reside here. Nectar provides access to virtual cores from an Ubuntu OpenStack environment. The system is developed with scalability and security in view.

Given below is an abstract architecture of the entire system. Starting from the top, data sources include Twitter and AURIN. Next comes the harvesting servers, which fetch data from twitter, process it and store it in the database server. These are scalable i.e. they can be 'n' in numbers, where n is the number of allowed/available nectar instances. Then comes the application server, which queries database server to respond to a web browser i.e. a responsive website (analytics dashboard).

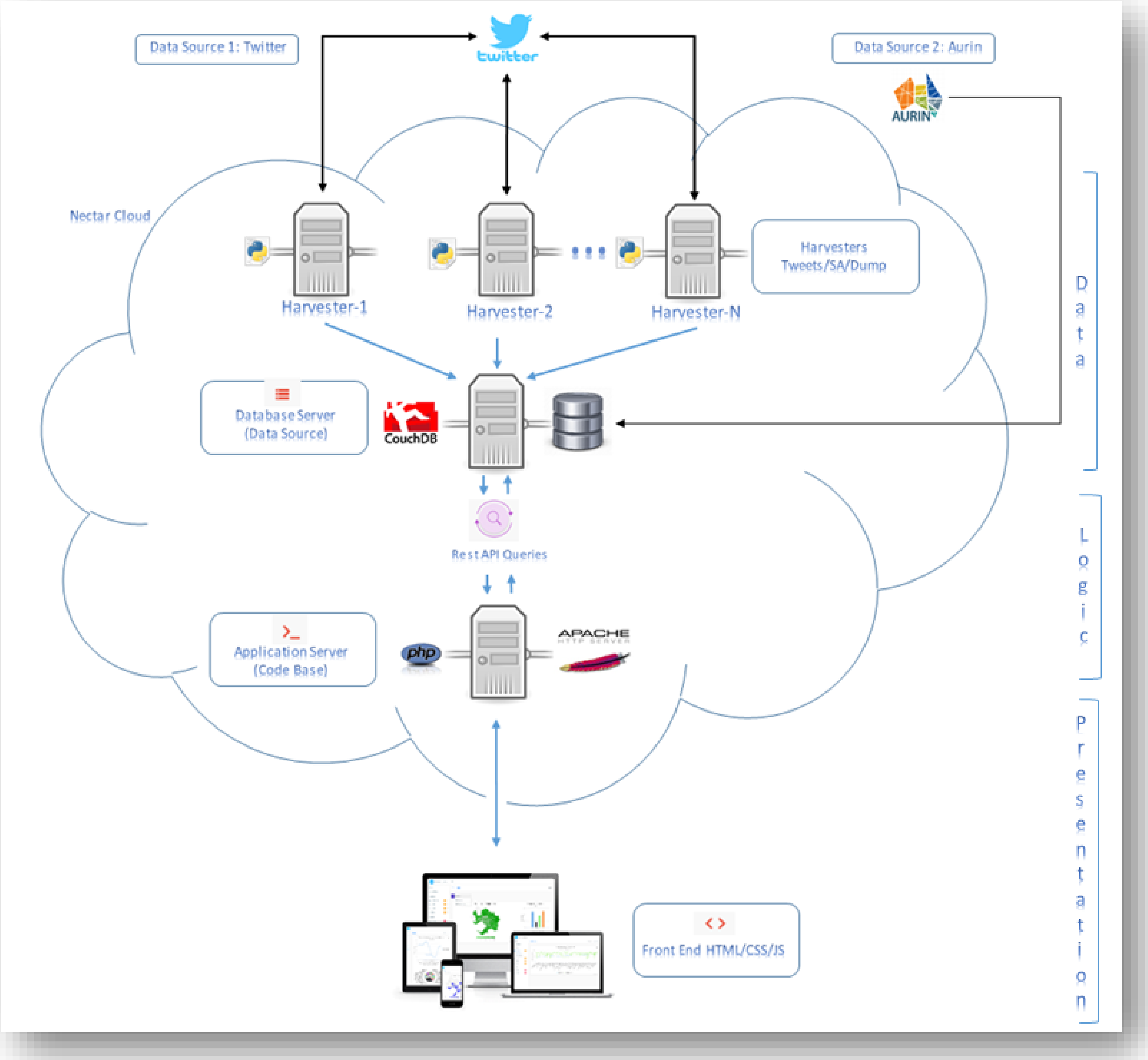


Figure 1.0 Architecture

2.2. Design

Cloud applications require a design which should fulfil four major aspects. These include but are not limited to scalability, availability, security and manageability. While choosing the design for this project, it was kept in view that the previously mentioned four aspects are covered. These lead to a design choice which are discussed in the following sections.

2.2.1. Scalability

The system encompasses an ability to scale horizontally. This is achieved by launching harvesting servers in an automated fashion. An automation script takes in the number of instances to launch and Melbourne's grid coordinate's (or any grid coordinate), south-west and north-east. It then divides the grid by number of instances, produces 'n' number of coordinates set as SW/NE boundaries and assigns it to each instance.

2.2.2. Availability

To make the system highly available a three-tier architecture was used. The database and application server, are separate from each other. A typical cloud environment such as this needs to have modules compartmentalized. Since we have multiple harvesters accessing and storing data in the database server, a high load is expected at this end. If the database and application servers were merged, it could have been the case that one of them becomes unavailable due to high hit ratio. For e.g. consider a scenario in which 100 harvesters are running, storing data in database server and at the same time a high number of users access the website concurrently. This could result in an unavailable server, either to the users or to the harvesters. To avoid this, the two have been separated out.

- **Volume:** The database server stores data on a 150GB nectar volume. In case if the server is down or it crashes, not only is the data safe but also the volume can be mounted to any other server and CouchDB can be reinstated to be available again.

2.2.3. Security

The security has been carefully deigned to make sure that access to the system is restricted from malicious sources. Keeping the database and application server separate, enables this to be achieved smoothly. For e.g. if even if an intruder does gain access to the application server, the data remains safe in a separate server, or if the application server crashes, data remains intact.

- **Key Pairs**: It is a secure authentication mechanism. Each of the new instances which are launched via automated script, are assigned a public key named 'cloudcc'. Only a private key holder can then access these servers. The database and application server are also assigned this key.

- **Security Groups**: For each of the server instances, specific security groups have been created so that servers can have limited access to prevent unauthorised intrusions. The following tables (Table 2.0 – 2.2) list all the security settings on the server types. It should be noted that for every harvester instance launched using the scripts, a security group 'harvesters' is automatically assigned to it with all the respective security settings.

Direction	Type	IP	Ports	Remote IP	Security Group
Egress	IPv6	Any	Any	::/0	-
Egress	IPv4	Any	Any	0.0.0.0/0	-
Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-
Ingress	IPv4	TCP	80 (HTTP)	0.0.0.0/0	-
Ingress	IPv4	TCP	5984	115.146.95.134/32	-

Table 2.0: Application Security Group assigned to Application server

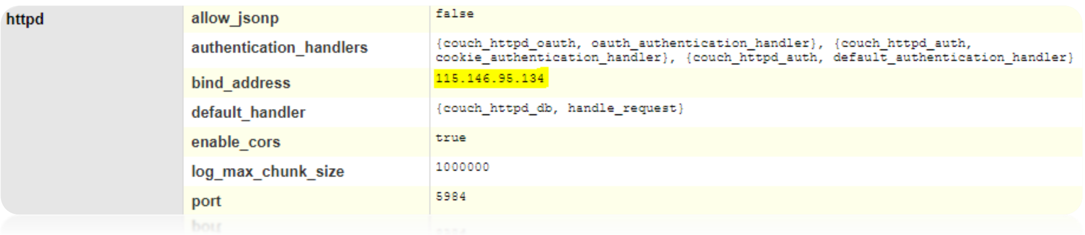
Direction	Type	IP	Ports	Remote IP	Security Group
Egress	IPv4	Any	Any	0.0.0.0/0	-
Egress	IPv6	Any	Any	::/0	-
Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-
Ingress	IPv4	TCP	5984	-	harvesters
Ingress	IPv4	TCP	5984	115.146.93.96/32	-

Table 2.1: Database Security Group assigned to database server

Direction	Type	IP	Ports	Remote IP	Security Group
Egress	IPv4	Any	Any	0.0.0.0/0	-
Egress	IPv6	Any	Any	::/0	-
Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-

Table 2.2: Harvester Security Group assigned to Harvester servers

- **CouchDB Access:** The CouchDB instance's http d bind address was changed to that of the database server's IP address so that harvesters can dump data on to this server using its IP and port 5984 (*Figure 2.3*)



httpd	allow_jsonp	false
	authentication_handlers	{couch_httpd_oauth, oauth_authentication_handler}, {couch_httpd_auth, cookie_authentication_handler}, {couch_httpd_auth, default_authentication_handler}
	bind_address	115.146.95.134
	default_handler	{couch_httpd_db, handle_request}
	enable_cors	true
	log_max_chunk_size	1000000
	port	5984

Figure 2.3: CouchDB Configuration

2.2.4. Manageability

This includes automated deployment and monitoring aspects. For automation, an Ansible script was written, which in turn triggers a Boto script to launch instances and assign configurations to each of the newly launched instance. It also installs the program and its respective configuration for each of the harvesting server, which then starts collecting twitter data and stores it in database after processing it. Detailed logs are generated by each of the launched harvesters, which determine the state of the program being run at each harvester instance. Keeping the application, database and harvesting server separate, also enable better performance due to load distribution in a modular fashion.

2.3. Fault Tolerance

The system exhibits fault tolerant behaviour i.e. if the system will continue to operate properly even if some of its components fail. These components include the tweet harvesters. The application developed for mining tweets, processing them and storing in the database, continues its operations, even if one of the instance fails. For e.g. if there are four harvesters running and one of them fails, three will continue to perform their designated operations, thereby providing a fault tolerant facility.

2.4. Single Point of Failure

A part of the system, which if it fails, the system will stop working is undesirable for a system which focuses on high availability and reliability. Even though the system has been designed to avoid such conditions, there, however, do exist scenarios in which SPF occurs. These are as following.

- **Application Server Fails:** If the application server fails, the web application for analytics will become unavailable.
  - **Resolution:** Since the application runs on Apache webserver, it will have to be restored on a new server. The size of web application is 78MB, which indicates that it is a light weight app i.e. expedited restoration.

- **Database Server Fails:** If the database server fails, the system will become unavailable. However, no data will be loss.
  - **Resolution:** This can be resolved by mounting volume to another server, getting CouchDB up and pointing towards the CouchDB file in the volume.

- **Volume Fails:** If the volume fails, the system will become unavailable.
  - **Resolution:** This can be overcome by maintaining a backup of the volume periodically or by maintaining replicated instances to store backup data on other instances as well. However, it is currently not implemented at this stage.

- **All Harvesters Fail:** If all the harvesting nodes fail, the system would no longer have updated data, but will continue to operate partially, showing outdated results at front end.

2.5. Limitations

The current limitations of the system, in addition to the ones mentioned in section previous section, '*Single Point of Failure'*, are as following.

- **Cluster vs Standalone Database versions:** A standalone version of CouchDB is currently used due to the limited scope of this project. In cases where a high volume of data storage and processing required, a clustered version would have been chosen instead. The current architecture is designed keeping in view of catering approximately 2-3 million tweets.
- **Replicated Instances:** Currently there are no replicated instances for application or database server. In a more available system, these instances would have been replicated and in case one or both go down, the replicated ones would be available.
- **Security:** Currently an http protocol is being used. This could be changed to https protocol, which is more secure and reliable. An SSL certificate of the https protocol ensures encrypted between client and the server.

3. Data Extraction

3.1. Hypothesis – Most Liveable City (Variance in Suburbs)

For seven consecutive years, Melbourne has been ranked among the top three most liveable cities globally. As a result, Melbourne, an already multi-cultural city has been further attracting people from around the globe as well as from other Australian cities in pursuit of a happier life. Given this continuous population surge from various cultural backgrounds and geographical locations, the suburb demographics are expected to show variance amongst themselves as people settle in different parts of Melbourne. The principal ambition of the study initiated with the proposition of exploring how '**Location'** and '**Demographic'** makeup of Greater Melbourne suburbs in terms of marital status as well as percentage of people born overseas effected the 'sentiment' of their tweets. The resultant hypotheses for the first approach were to evaluate how certain demographical aspects of suburbs disseminate in levels of happiness.

Hypothesis 1: Are people living in suburbs happier around the coast happier than inner city suburbs?
Hypothesis 2: Are suburbs with higher percentage of married people more positive?
Hypothesis 3: Does increase in percentage of people born overseas in a suburb have an impact on happiness level?

In the second approach of data analysis on the same twitter data, we wish to assess how the lifestyle and quality aspects show up in tweet patterns over Melbourne suburbs.

Hypothesis 4: Is it possible to identify crime hotspots in Greater Melbourne using Twitter?
Hypothesis 5: Are inner city suburbs the hub of drug and alcohol related activity?
Hypothesis 6: Are sports activities localised or equally spread over Greater Melbourne suburbs?

3.2. Harvesting of tweet data

The process for harvesting and processing the tweets is as follows. All the utilities were developed in Python. We used multiple sources for harvesting the tweets, twitter itself and The University of Melbourne tweet database.

3.2.1. Harvesting using location filtering

The tweets are harvested using Tweepy, a python library to stream tweets from twitter. The library required certain access credentials as parameters to access twitter data – access tokens and consumer keys. It was observed that twitter has a certain limitation for data extraction therefore we had to get access credentials from all the group members. While researching the methods for tweets extraction, we first implemented a twitter extractor using Tweepy's search function. However, we found out that it would only return a limited set of historical tweets till the last 7 days as per the twitter data policies. Therefore, we implemented the harvesting using an alternate method to stream and extract all tweets in real-time.

Different attributes can be added to the Tweepy streaming function including location filtering and keywords tracking. However, we are just filtering the tweets based on location within Melbourne. Therefore, we use the Tweepy function for tracking based on location coordinates for Melbourne. Furthermore, certain scripts were added to the harvesting program so that if we are getting a tweet from a user based in Melbourne, we also extract the recent most 1000 tweets from that user profile since it is highly likely that the same user has tweeted from the same location. As an additional check, we verify the location coordinates again to identify tweets strictly within the Melbourne grid and eliminate the possibility of getting random tweets from that user which do not fall within the location grid.

The initial location search implementation compared the location of each of tweets with a file listing all the coordinate points as well as the radius of each of the suburbs. After a couple of test analysis, we started getting many repetitive tweets and it was decided that this was an inaccurate approach since the suburb radius were calculated mathematically and it can be stated that suburb regions would not be circular to calculate their radius in the first place.

To identify suburbs, we then used a different approach where we extracted the suburb coordinate polygons from AURIN and used a python library knows as 'shapely' to identify and calculate the polygons and whether the tweet coordinate polygon falls within the respective suburb polygon. If it does, then we add the suburb name to that tweet. Therefore, using this way of comparing coordinate polygons, we minimized the tweet repetitions rather than relying on suburb radius which was highly inaccurate.

Finally, the extraction utility performs an additional functionality to ignore all streamed tweets having certain keywords like 'weather' and 'trending' – words which are frequently found among tweets.

3.2.2. Cleansing

After extracting a tweet within a location box, the next step involves cleansing the tweet. We use a python library 'Tweet-Preprocessor' to parse and cleanse a tweet. The library also offers extended functionalities including hashtag, URL links and smiley identification within a tweet. However, we are using the library just to cleanse the tweet to get rid of junk data. Also, we are performing an additional regex check just to get rid of additional symbols within the tweets. For our records, we are also saving the raw tweets data in the database as well.

3.2.3. Sentiment Analysis

Sentiment analysis is the process of determining the attitude or the emotion of a tweet. We used a python library 'TextBlob' to identify the sentiments and add those to the database for each of the tweets. TextBlob is built on the strong pillars of NLTK and PATTERN and harnesses much value from both these platforms. NLTK is one of the best known platforms for building Python programs that can function with human language data. This provides a straightforward interfaces to numerous lexical processes as well as facilitating a suite of text processing libraries for classification, tokenisation, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, supported by an active public chat forum. The PATTERN module encompasses an efficient part-of-speech tracker for English (identifies nouns, adjectives, verbs, etc. in a sentence), sentiment analysis, tools for English verb conjugation and noun singularization & pluralization, as well as a WordNet interface.

We utilise the Python library TextBlob to evaluate the tweet from the sentiment perspective by leveraging the inbuilt “sentiment()” function. TextBlob’s text parser file sentiments.py is essentially based on probability distribution. This function of TextBlob provides two values, the first one being polarity of sentiment and the other subjectivity which takes into account writer’s perspective. When the parser parses the tweet given to it, the subjectivity and polarity of every adjective is summated. Following this, a simple probably distribution is generated. Since the probability is constrained to lie between 0.0 to 1.0, the function returns the polarity of a tweet as a float value between -1 (Negative) to 1 (Positive). For the analysis, we are just extracting the polarity value to identify a tweet and saving it as a positive or a negative.

3.2.4. Tagging

To tag the tweets, we implemented multiple approaches including the identifying the presence of certain ‘keywords’ in those tweets as well as using ‘Soundex’ – identify words and matching them with keywords based on the phonetic spellings even if the actual spellings do not match. However, for the project we are just using the keyword identification as the soundex method gave pretty inaccurate results for words that do not even lie in the same category of keyword datasets. For this, we included three main keywords datasets – crime, alcohol (drinking) and sports as separate files to the program to check whether a tweet contains the keywords from each of the keyword datasets. Within each dataset, we have listed certain keywords which are commonly used under these word sets.

4. Data Analysis

4.1. General Overview of Twitter Activity

The following table (Table 4.0) shows the number of tweets harvested for Greater Melbourne that we have based our entire study on for tweets. The initial harvesting step fetches all tweets that belong to a rectangular are defined by latitude and longitude boundaries of Greater Melbourne which resulted in over half a million tweets. Upon mapping of tweets to pre-defined polygons, only 48865 tweets were found to not belong to a unique suburb and classified as N/A. As the study revolves around suburban analysis, these tweets were excluded, resulting in a very high data utilisation rate of the harvested data at 92%.

Description	Count
Total tweets harvested in Greater Melbourne Area	606985
Total tweets mapped to a Greater Melbourne Suburb	558120

Table 4.0 Count of Tweets

Greater Melbourne encompasses 309 suburbs and the aim of the study is to assess how the happiness levels and lifestyle activities vary in these suburbs. But prior to delving into the granularity of suburbs, we get an overview of the Greater Melbourne area.

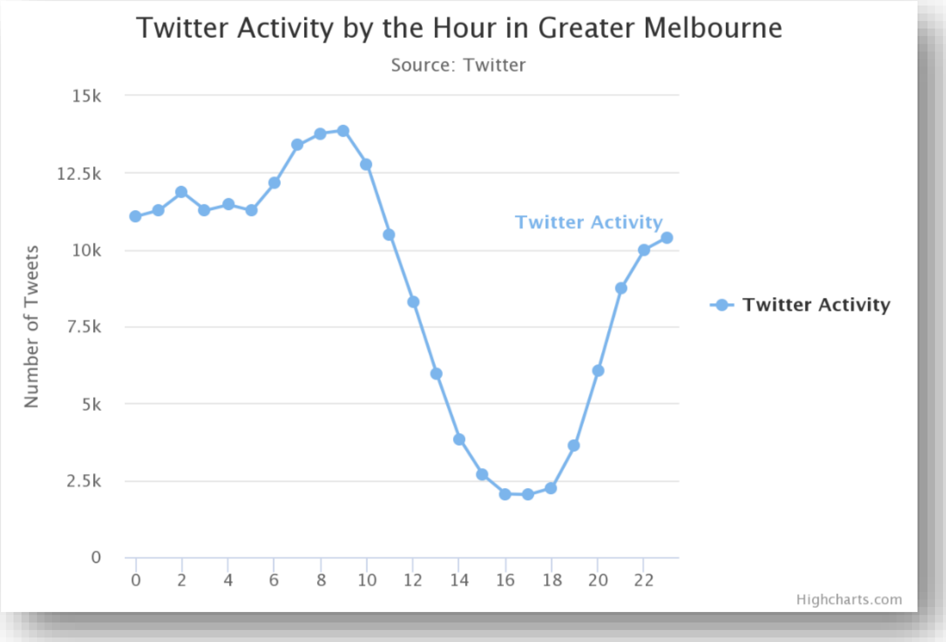


Figure 4.1 Variance in count of tweets over 24-hour period

Figure 4.1 shows that the twitter activity accelerates during the early hours of morning, reaching its peak just before the start of official business hour. Also, noteworthy from this image is the second surge starts to occur in the late afternoon after typical business hours of 5 PM continually increasing until 10 PM before plateauing.

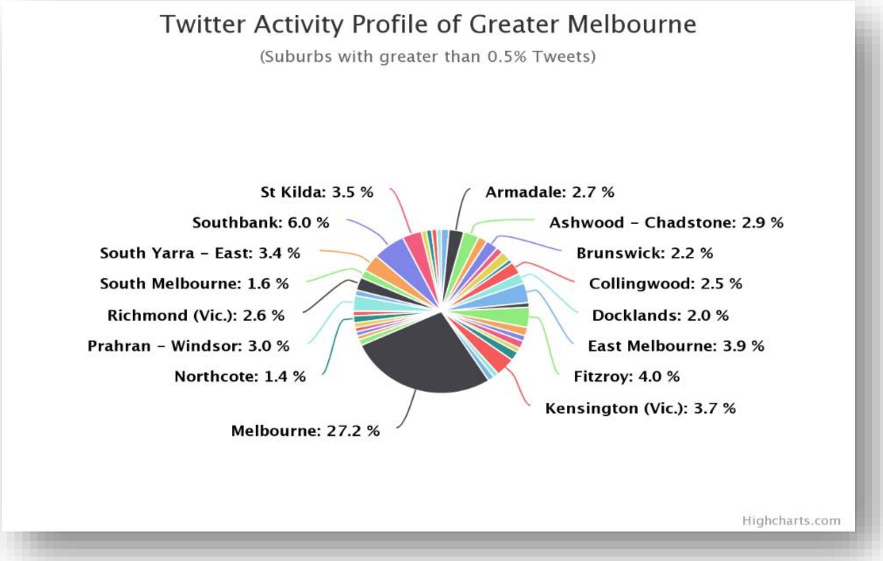


Figure 4.2 Distribution of tweets over Greater Melbourne suburbs

After assessing the activity against hour of the day, we then analyse the mapped tweets to identify what the most active tweeting suburbs are. Figure 4.2 reveals that Melbourne CBD suburb accounts for more than a quarter of the total tweets. This correlates well with the fact that Melbourne CBD is the centre of Greater Melbourne and hub of social, business and event activities.

4.2. Visualization against AURIN data

In this section, we will present the analysis performed on the processed tweets, along with visualisation of data from the webpage that was developed as part of this study. All the scenarios combine twitter analysis as well as at least one aspect of suburban profiling using information extracted from AURIN.

4.2.1. Location versus Happiness

The first suburbia level analysis was done using the mapped twitter data and the associated sentiment for the tweets. For each suburb, we determine the percentage of positive tweets which then presents itself as a relative happiness score for suburb comparison, such that this can be shown on a heat map as depicted in Figure 4.3. Darkening of the scale depicts increasing level of happiness of a suburb. The level of happiness is determined by computing a percentage of the tweets classified as 'positive' out of the total tweets mapped to a particular suburb.

At an initial glance, it is evident there are certain types of areas, for instance the outer regional suburbs (coastal as well as inland) with larger area in comparison to inner suburbs, appear to be generally darker than the smaller inner area suburbs.

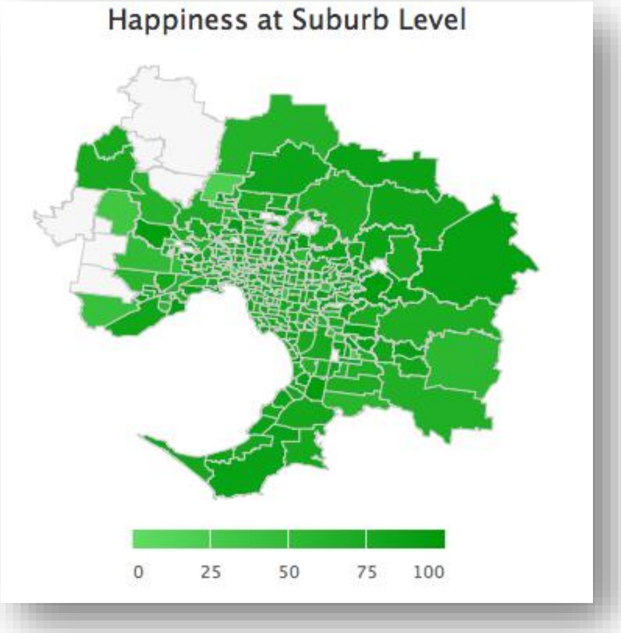


Figure 4.3: Happiness % over greater Melbourne suburbs

To evaluate **Hypothesis 1** of whether coastal area suburbs are happier than the inner-city suburbs, the happiness level of all the coastal suburbs have been presented in Table 4.4 and those of inner city suburbs have been listed in Table 4.5.

Coastal Suburb	Happiness Level (%)
Werribee	80.26
Point Cook - East	90.38
Altona	85.75
Williamstown	82.11
Newport	73.68

Coastal Suburb	Happiness Level (%)
Port Melbourne	80.43
Albert Park	81.47
St Kilda	77.25
Elwood	76.24
Brighton	79.58
Hampton	67.93
Sandringham	71.73
Beaumaris	80
Mentone	75
Mordialloc - Parkdale	80.87
Edithvale – Aspendale	79.31
Carrum – Paterson Lakes	94.74
Chelsea BonBeach	68.03
Seafood	72.48
Frankston	74.22
Frankston - South	77.78
Mount Eliza	84.85
Mornington	77.61
Mount Martha	86.07
Dromana	76.92
Rosebud McCrae	81.25
Point Napean	78.25
Flinders	86.42
Hastings – Somers	75.68
Somerville	78.95
Average	**79.2**

Table 4.4 Happiness level for coastal suburbs of Greater Melbourne

Inner City Suburb	Happiness Level (%)
Carlton	76.04
Carlton North	82.7
Docklands	79.65
East Melbourne	79.32
Flemington	75.89
Kensington	20.62
Melbourne	80.99
North Melbourne	23.42
Parkville	76.53
Port Melbourne	80.43
Southbank	86.02
South Yarra - East	81.96
South Yarra - West	76.94
West Melbourne	70.25
Average	**70.76**

Table 4.5: Happiness level for inner city suburbs of Greater Melbourne

From the above tables, a significant increase in level of happiness is noted. The average score of happiness increases by 11.9%, increasing from 70.76% to 79.2% from inner city to coastal, respectively.

4.2.2. Marital Status Versus Happiness

Presented in Figure 4.6 is a graph with a listing of suburbs depicted by the blue line, in an increasing order of percentage of married people at a suburbia level. The green line represents the happiness score (%) for each corresponding suburb. An overview of this graph does not show an apparent trend of happiness with increase in percentage of married people in a suburb.

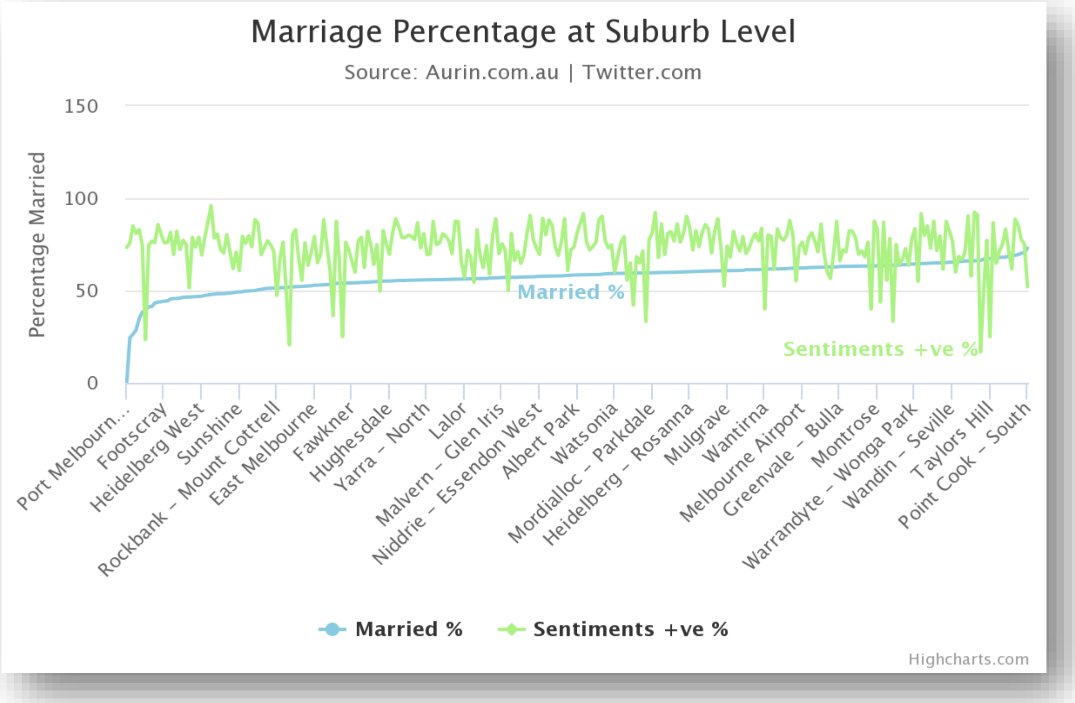


Figure 4.6: Marriage percentage versus Happiness for Greater Melbourne

As the happiness plot has spikes and troughs throughout, we drill down deeper on both extremes to see if there are any underlying correlations. In Table 4.7 an average of the 10 suburbs with the highest marriage percentage is computed and on the other hand Table 4.8 presents the average of the 10 suburbs with the least proportions of married people.

Suburb Name	Married (%)	Happiness Level (%)
Doreen	67.94	66.67
Gisborne	68.01	75
Mount Eliza	68.08	83.33
Truganina	68.66	71.43
Cranbourne East	68.97	61.76
Point Cook – East	69.05	90.38
Flinders	69.44	85.44
Wolbert	70.24	75

Suburb Name	Married (%)	Happiness Level (%)
Macedon	70.48	75
Point Cook – South	72.95	50.02
Average	**69.382**	**73.403**

Table 4.7: Top 10 Suburbs with highest marriage percentage

Suburb Name	Married (%)	Happiness Level (%)
Carlton	24.52	75.69
Braeside	26.32	85
Melbourne	28.55	80.91
Flemington Racecourse	35	83.06
Clayton	38.28	74.16
North Melbourne	39.37	23.43
Burwood	41.17	74.79
Parkville	41.29	76.76
Flemington	43.28	75.88
Collingwood	43.76	85.75
Average	**36.154**	**73.543**

Table 4.8: Top 10 Suburbs with lowest marriage percentage

The comparison of the two averages from Table 4.7 and Table 4.8 reveals that there is no significant difference in the two values.

4.2.3. Born Overseas percentage versus Happiness

Below in the Figure 4.9 the suburbs have been listed in an increasing order of percentage of people born overseas along with a corresponding happiness score (%) for each suburb. The graph maps suburbs with at least 50% overseas population and intends to provide a view if increasing percentage of overseas population has an impact on the happiness of a suburb.

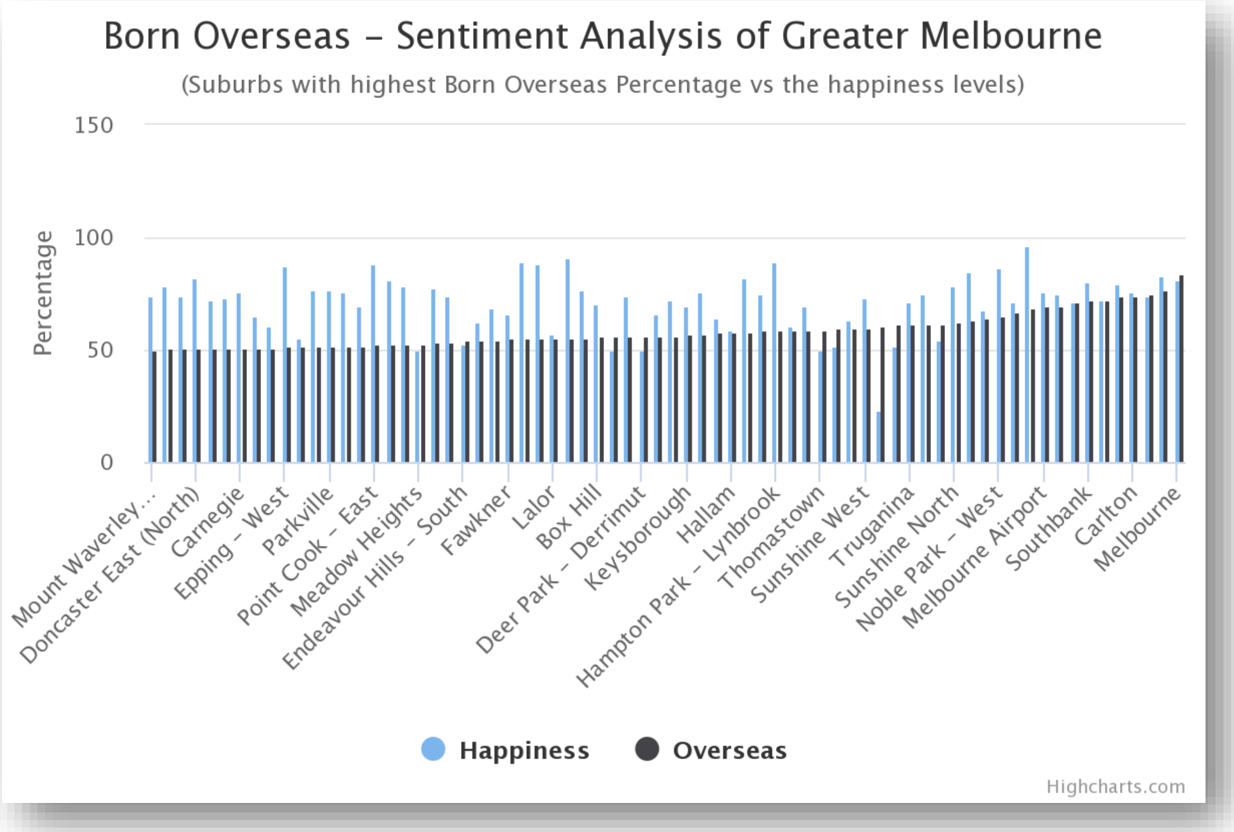


Figure 4.9: Marriage percentage versus Happiness for Greater Melbourne

It is evident from Figure 4.9 that as the percentage of overseas people in a suburb increases, for each corresponding suburb the score of happiness is either above or very close to the corresponding born overseas percentage except for one anomaly in the data for North Melbourne. So, as we proceed to look at the higher end of overseas people, the happiness stays equally high. A further zoomed view of this has been provided in Table 5.0 and Table 5.1.

Suburb Name	Born Overseas (%)	Happiness Level (%)
Melbourne	84.1	80.9
Flemington - Racecourse	83.1	77.2
Carlton	74.9	74.2
Docklands	74.1	79.6
Springvale	72.2	72.3
Southbank	72.0	80.5
Dandenong	71.5	72.6
Clayton - South	75.1	69.6

Suburb Name	Born Overseas (%)	Happiness Level (%)
Melbourne Airport	69.3	76.4
St Albans – South	68.6	96.0
Average	**74.49**	**77.93**

Table 5.0: Top 10 Suburbs with highest born overseas percentage

Suburb Name	Born Overseas (%)	Happiness Level (%)
Mount Waverly - North	50.4	73.7
Ormond – Glen Huntly	50.6	78.5
Mulgrave	50.7	74.2
Doncaster East (North)	50.7	82.4
Maribyrnong	50.8	72.2
Burnside Heights	50.8	72.9
Carnegie	50.9	76.2
Endeavour Hills	50.9	65.0
Glenroy	51.3	61.0
Epping - West	51.4	87.5
Average	**50.85**	**74.36**

Table 5.1: Top 10 Suburbs closest to 50% born overseas percentage

The average happiness level of suburbs with more overseas people is higher than the suburbs with fewer people that were born out of Australia. The happiness ranges from 69.6% to 96.0% for the overseas range of 68.6 to 84.1% in Table 5.0.

4.2.4. Top 5 Happiest and Unhappiest Suburbs

Further to evaluating happiness of suburbs against two sets of demographic information from AURIN, we have also included the following information for each suburb in our analysis sources.

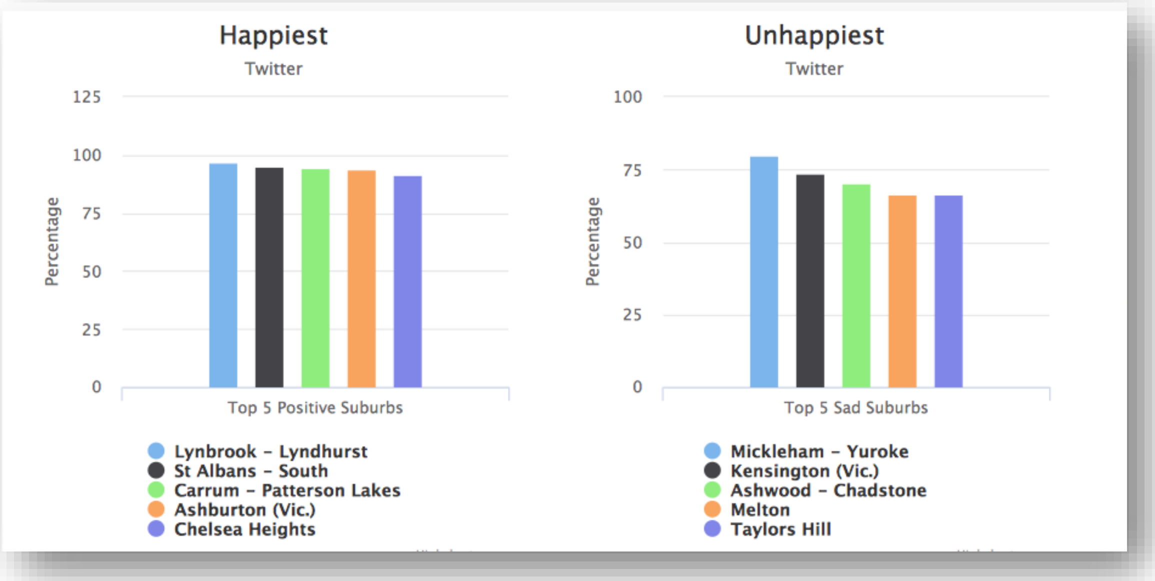


Figure 5.2 Top 5 Happiest and Unhappiest Suburbs

For each of the suburbs shown in Figure 5.2, we evaluate the happiness level against the four indexes derived from AURIN data sources. This has been presented in **Table 5.3** and **Table 5.4.**

Suburb Name	Happiness Level (%)	Married (%)	Born Overseas (%)	Education and Occupation Index (%)	Economic Resource Index (%)
Lynbrook - Lyndhurst	96.8	68	56.0	49	93
St Albans – South	95	47	68	2	6
Carrum – Patterson Lakes	94.7	58	30	64	66
Ashburton (Vic.)	93.8	61	30	92	84
Chelsea Heights	92.0	60	29	45	66

Table 5.3: Top 10 Suburbs closest to 50% born overseas percentage

Suburb Name	Happiness Level (%)	Marital (%)	Born Overseas (%)	Education and Occupation Index (%)	Economic Resource Index (%)
Michkleham – Yuroke	16.7	66	38	96	16
Kensington (Vic.)	20.8	51	39	93	19
North Melbourne	23.5	39	60	90	2
Taylors Hill	25	67	39	48	91
Melton	25	53	30	6	27

Table 5.4: Top 10 Suburbs closest to 50% born overseas percentage

From Table 5.3 it is notable that the marital proportion of happiest suburbs is relatively high at an average just below 60%. Also, as the percentage of overseas people increases an increase in happiness level is noted. From the wellness indexes in these tables no obvious trend could be noted as there are anomalies for both happy and unhappy suburbs when it comes to economic and education / occupation indexes.

4.3. Visualization of Lifestyle and Quality aspects

Moving on from the demographic indexes for evaluation of tweets, we now present a different evaluation set encompassing Crime, Drug and Alcohol and Sports, evaluating suburbs from safety and wellness. The harvested tweets were processed to identify if they belonged to one or more of these categories.

4.3.1. Crime Visualisation of Greater Melbourne

Crime is a noteworthy piece of each general public. Its expenses and impacts touch pretty much everybody to some degree. The sorts of expenses and impacts resulting vary from type to type of crime incident. What's more, a few expenses impacts are immediate while others endure forever. Different expenses to casualties can incorporate restorative costs, property misfortunes, and loss of wage and influence the overall liveability of a location. We evaluate the tweets of Melbourne for mention of crime related words and all the tweets revealed were mapped to their respective suburbs.

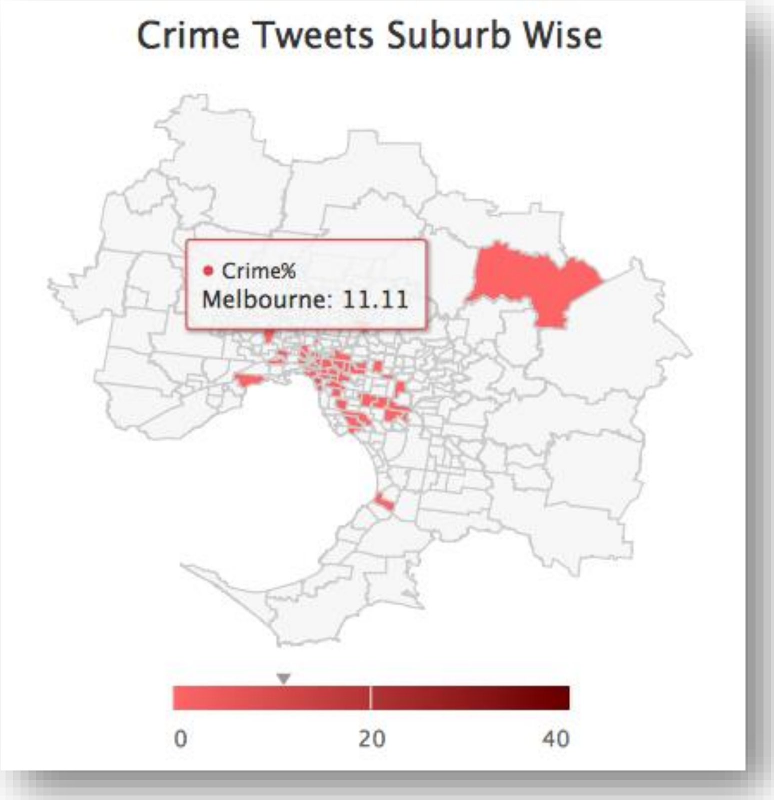
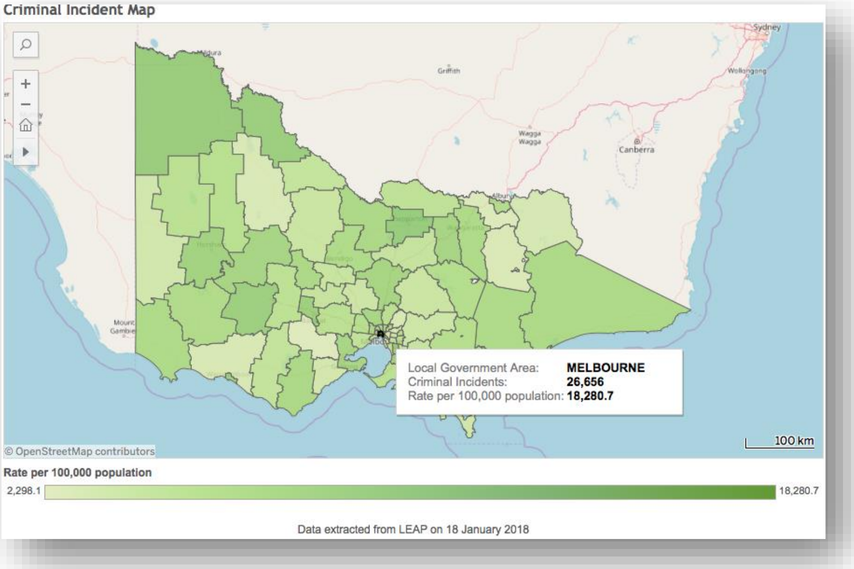


Figure 5.5: Heatmap of Crime related tweet

The heatmap in Figure 5.5 revealed that most of the crime tweets occur in the inner-city suburbs. Also, majority of the 309 suburbs specifically as we move away from the central Greater Melbourne, show insignificant crime related activity except for an outlier, Healsville. Further to that, the highest rate of crime, is noted in the Melbourne city suburb at 10% of the total crime tweets.



Local Government Area: MELBOURNE
Criminal Incidents: 26,656
Rate per 100,000 population: 18,280.7

Data extracted from LEAP on 18 January 2018

Figure 5.6: Heatmap of Crime activity from VIC government

Interestingly when we explored the data from Victorian government's official crime statistics, it also revealed Melbourne city as the suburb with the highest crime. This suggests that twitter is a potential tool for identifying crime hotspots. Figure 5.6 shows the crime statistics for the whole region.

4.3.2. Drug and Alcohol Visualisation of Greater Melbourne

Drug and Alcohol is perceived to have a negative impact on quality of life due to its ability to render people into states of vulnerability, such that they are more likely to come in harm's way or cause harm to others. However, alcohol consumption in moderation is part of our culture and twitter activity relative to keywords for alcohol is expected to bring tweets relating to celebratory occasions and holidays as well as getaways.

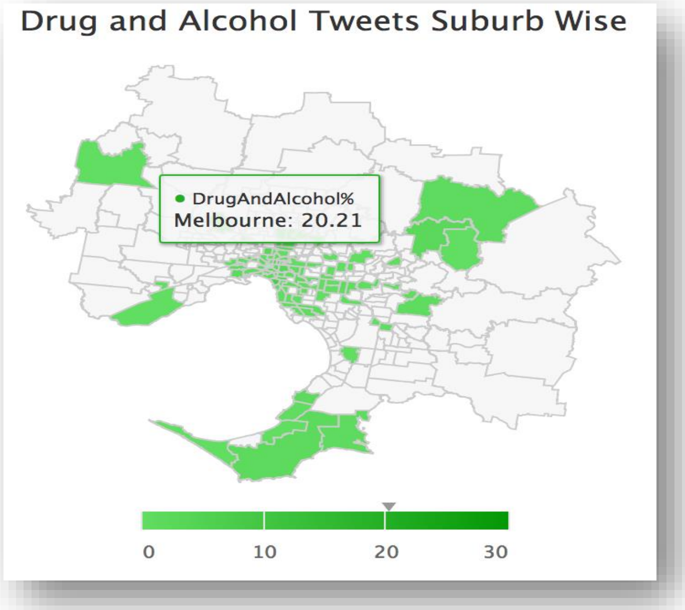


Figure 5.7: Heatmap of Drug/Alcohol tweets

From Figure 5.7 we identify a few different pockets of suburbs revealing tweets mapping to use of drug/ alcohol related keywords. The first pocket is the inner-city suburbs, with Melbourne city again being revealed as the suburb with most tweets from this classification. It may be deduced from the fact that this same suburb also the highest crime rate, that there is a correlation between crime rate and drug/ alcohol activity in the city central. On the other hand, for the other major localized set of suburbs revealing drug/ alcohol activity, located at the peninsula tip of Greater Melbourne revealed no crime tweet. However, what is common amongst these suburbs is that this cluster contains of tourist hot spots, like Mornington and Mount Martha. Hence, alcohol related tweets, most likely from positive category, can be expected to be seen here.

4.3.3. Sports Visualisation of Greater Melbourne

Melbourne is officially the sporting capital of the world for the last decade, having been awarded multiple World's Ultimate Sports City awards in last 10 years along with the SportBusiness Ultimate Sport City Award in the 2016. It is home to a wide range of sports from Australian Open Grand Slam, the Australian Formula 1 Grand Prix, the MotoGP and a long list of blockbusters including Cricket World Cup, Commonwealth Games. For this reason, we wanted to explore the twitter activity of Melburnians with respect to sports.

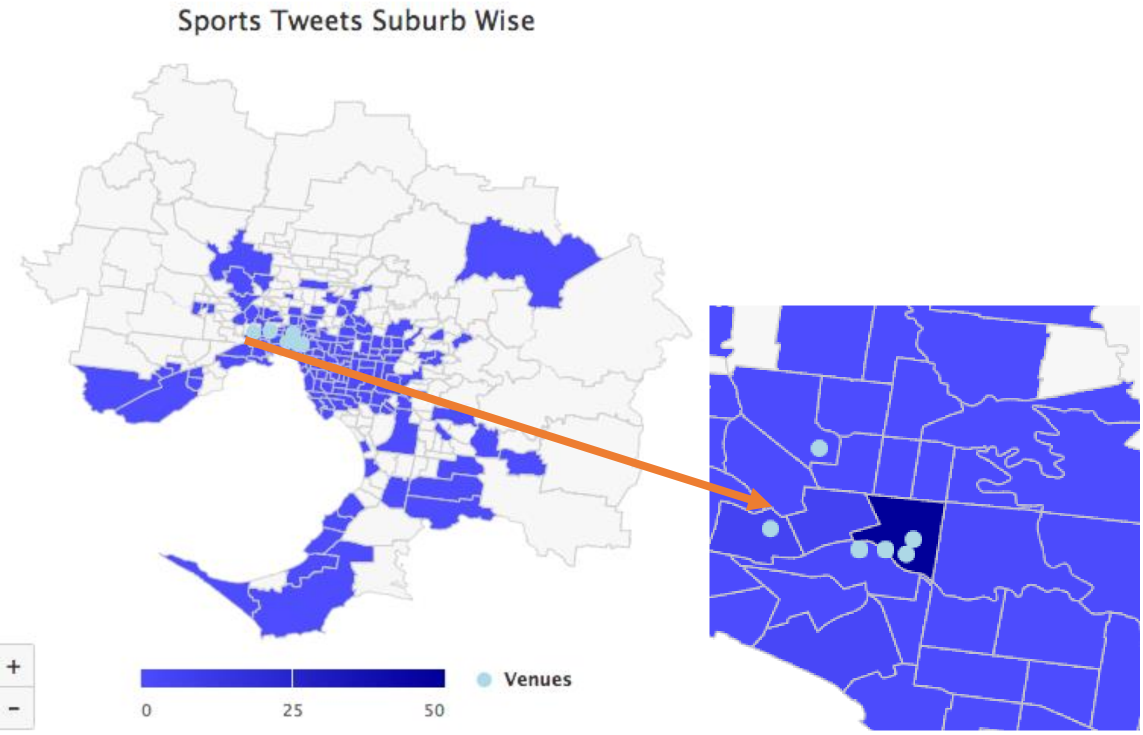


Figure 5.8 Heatmap of Sports related tweet

The tagged sports tweets have been shown on the heatmap in Figure 5.8. The sports activity is quite widely spread over not only the inner-city suburbs but also the coastal ones. On the map, we also have markers for the top ten sports venues of Greater Melbourne. Zooming in to identify the darkest (most sporting activity) suburb, reveals that 50% of the sports tweets come from here. This also happens to be the suburb where the majority of the top ten venues are located. Hence, revealing that twitter can help us identify most popular sporting venues.

5. Cloud and Scalability

5.1. The Use of the Cloud

For the purpose of our research we used NeCTAR Research Cloud, an initiative of the National Research Infrastructure for Australia. The aim of using this cloud facility was to provide the team with sufficient computational resources and enable fast and efficient collection and processing of the required data. During the implementation of our system in the cloud, several obstacles and limitation were encountered, nevertheless, various benefits were also achieved. This section will cover various aspects relating to the use of the cloud in general and the experience of using NeCTAR Research Cloud in particular. The major perspectives in consideration include elasticity, scalability and security.

5.2. Elasticity

Arguably, the most valuable feature of the cloud based system is its elasticity (or flexibility). Most cloud providers including NeCTAR provide "on-demand" amount of resources to facilitate the user's needs. In other words, should the user require the tenfold amount of resources he has now, for example, he can almost instantly gain access to them (given that he's entitled to this amount) and start utilising them straight-away. This feature makes the cloud a more attractive infrastructure compared to in-house resources as the users are able to satisfy their demands for resources straight-away and not be constrained by existing in-house capacities.

During the implementation of the TwitAur Analytics system, the research team observed the elasticity of NeCTAR Research Cloud which was reflected in the ability to launch multiple instances of virtual machines on demand, limited only by the pre-allocated quota (which could be extended for the purpose of future research).

5.3. Scalability

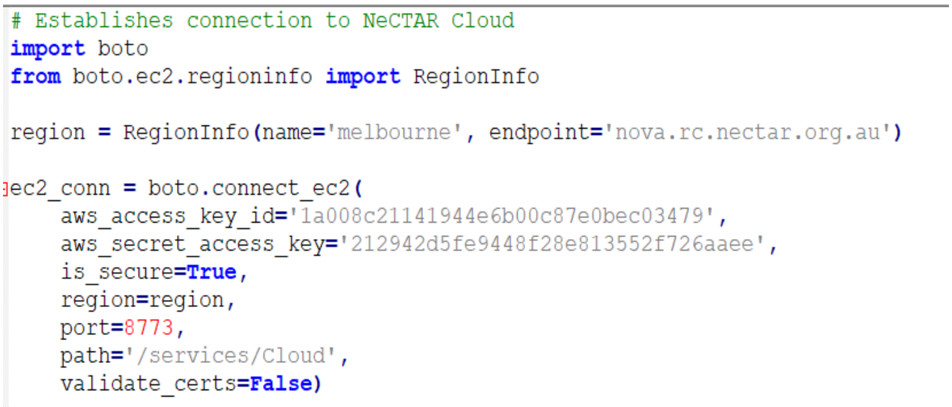
Scalability is another valuable feature of the cloud that opens up opportunities for researchers and businesses alike that in-house systems lack. It allows systems scale horizontally to increase (or decrease) its reach at a moment's notice. The principle of scalability is, therefore, built upon elasticity (the ability to have an instant access to sufficiently large amount resources) as well as the ability of the researcher to communicate with the cloud to configure newly added resources. The principle of automation plays a significant role in this process: even if one had an ability to instantly add a thousand additional virtual machines to their disposal, it would only achieve half of the purpose as those machines still would need to be configured for the researcher's or businesses' purpose. To achieve the necessary degree of automation, two stages of deployments are usually required: the setup of the additional cloud resources and their subsequent configuration. These two stages will now be covered in details in the context of configuring NeCTAR Research Cloud with Ansible (the automation tool), and the OpenStack API provided through the "boto" interface.

6 Automation

6.1 Launching NeCTAR instances

The NeCTAR visual interface allows users to create virtual machines, or instances, at a click of a button; there, it is possible to specify additional settings for those instances, including their geographical location, their size, security settings, their number and other settings. There are also additional menus to configure the network of those instances, attach and manage volumes and create snapshots to back-up the existing instances. However, creating and configuring a large amount of instances might prove tedious and countering the very benefit of speed and flexibility. Therefore, to scale the resources efficiently, the cloud can be approached programmatically through an API. In case of NeCTAR Research Cloud, we decided to use the OpenStack API exposed through the "boto" interface to configure the newly launched instances.

Launching instances with "boto" required writing Python scripts that would first establish connection with the cloud (as shown in Figure 6.0) and then launch the instances with the customised configuration (see Figure 6.1). During the configuration, various settings are specified, such as the amount of instances to be created, the image and size of the instance and the security group to assign it to. However, the benefit of such programmatic approach was not just incorporating this procedure in subsequent automation process but also in having power to specify additional logic for the instances which was not provided through the OpenStack API or the visual interface.

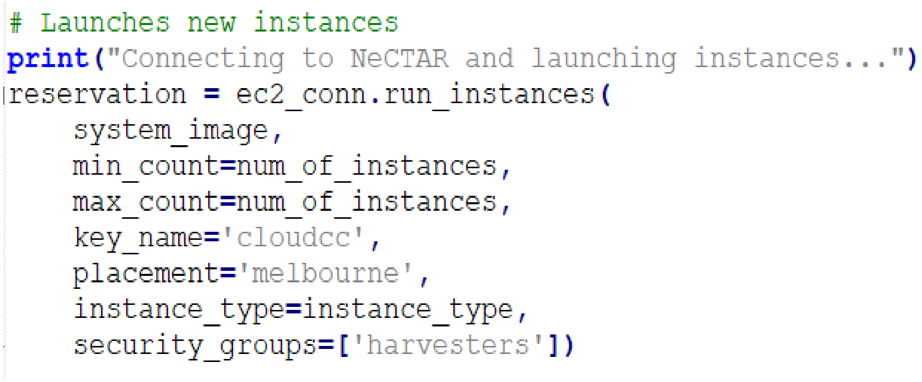


```python
# Establishes connection to NeCTAR Cloud
import boto
from boto.ec2.regioninfo import RegionInfo

region = RegionInfo(name='melbourne', endpoint='nova.rc.nectar.org.au')

ec2_conn = boto.connect_ec2(
    aws_access_key_id='1a008c21141944e6b00c87e0bec03479',
    aws_secret_access_key='212942d5fe9448f28e813552f726aaee',
    is_secure=True,
    region=region,
    port=8773,
    path='/services/Cloud',
    validate_certs=False)
```

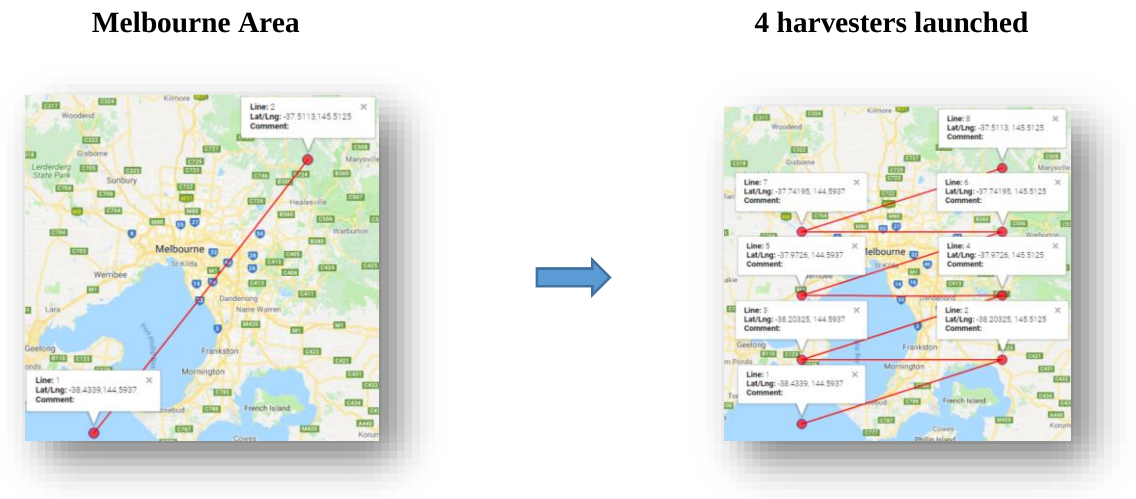
Figure 6.0: Establishing connection to NeCTAR



```python
# Launches new instances
print("Connecting to NeCTAR and launching instances...")
reservation = ec2_conn.run_instances(
    system_image,
    min_count=num_of_instances,
    max_count=num_of_instances,
    key_name='cloudcc',
    placement='melbourne',
    instance_type=instance_type,
    security_groups=['harvesters'])
```

Figure 6.1: Launching new instances.

For instance, our Twitter harvesting algorithm splits the target area in several quadrants, equal to the amount of running harvesters (illustrated in Figure 6.2), thus achieving parallelisation. This could only be achieved through assigning the harvesters different sections of the target area during the instance creation. These parameters are calculated and passed onto the newly created instances dynamically; otherwise, it would be the case of creating hundreds of instances performing exactly the same task. Through the programmatic access to the cloud, parallelisation could be introduced.


Melbourne Area

4 harvesters launched

Line: 2
Lat/Lng: -37.5113,145.5125
Comment:
Melbourne
Line: 1
Lat/Lng: -38.4339,144.5937
Comment:
Line: 8
Lat/Lng: -37.5113, 145.5125
Line: 7
Lat/Lng: -37.74195, 144.5937
Line: 6
Lat/Lng: -37.74195, 145.5125
Line: 5
Lat/Lng: -37.9726, 144.5937
Line: 4
Lat/Lng: -37.9726, 145.5125
Line: 3
Lat/Lng: -38.20325, 144.5937
Line: 2
Lat/Lng: -38.20325, 145.5125
Line: 1
Lat/Lng: -38.4339, 144.5937

Figure 6.2: Parallelisation of tweet harvesting

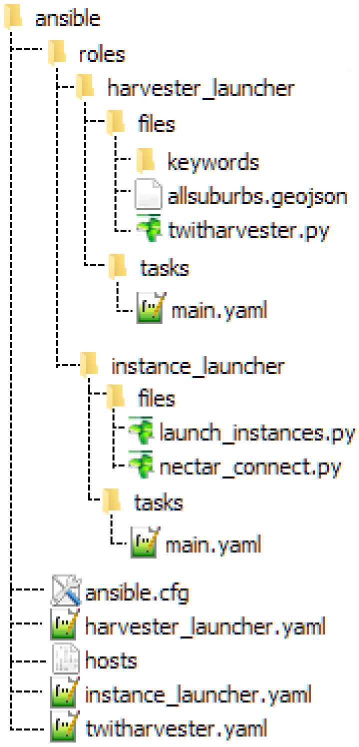
6.2 Configuring the instances

The second stage to achieve scalability was automating both the deployment and the configuration of the instances. For that purpose, Ansible was used. Ansible is an automation tool that allows idempotent configuration of multiple machines through the use of user-crafted “playbooks” with “tasks”. Idempotent in this context refers to the ability of Ansible to check the state of the machine and change it if it is considered to be misaligned with the provided directives, thus converging the state of all machines to the identical, user-customised configuration. We will now illustrate how automation and “single-click” deployment was achieved in NeCTAR Research Cloud through the use of Ansible.

To fully configure the system for the purpose of our research, several steps needed to be achieved:

1. Connect to NeCTAR Research Cloud
2. Launch a user-specified preconfigured (cloud-wise) amount of instances
3. Install packages required for running the programme
4. Transfer scripts required for running the programme
5. Run the programme which would connect to the database, start harvesting process and push the collected data to the database

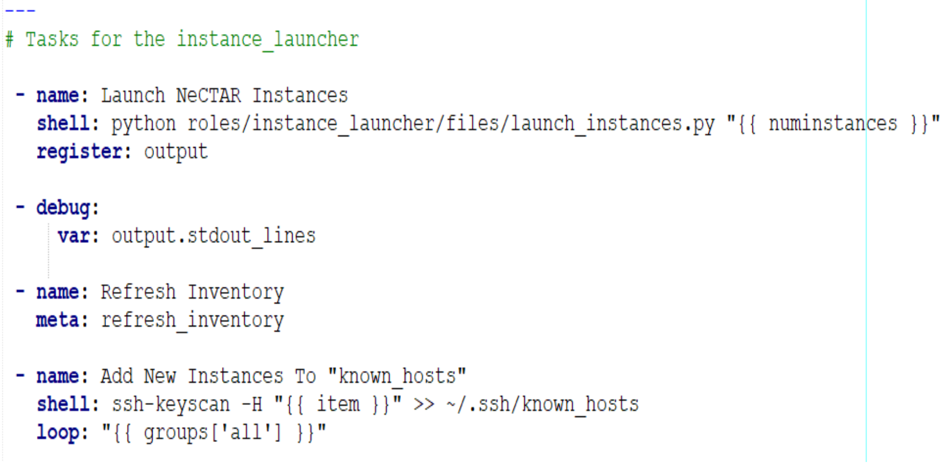
These steps were captured by using Ansible's attributes such as Inventory, Files and Tasks and were compiled into Playbooks which, upon being run, performed various tasks (such as installing Python packages, transferring required files and running the harvesters' scripts). The structure of our Ansible implementation is shown below (Figure 6.3).



```
ansible
├── roles
│   ├── harvester_launcher
│   │   ├── files
│   │   │   ├── keywords
│   │   │   ├── allsuburbs.geojson
│   │   │   └── twitharvester.py
│   │   └── tasks
│   │       └── main.yaml
│   └── instance_launcher
│       ├── files
│       │   ├── launch_instances.py
│       │   └── nectar_connect.py
│       └── tasks
│           └── main.yaml
├── ansible.cfg
├── harvester_launcher.yaml
├── hosts
├── instance_launcher.yaml
└── twitharvester.yaml
```

Figure 6.3: Ansible configuration

The first two steps were captured by tasks containing in the "instance_launcher" sub-Playbook. You can see the Tasks associated with launching the "boto" scripts and preparing the instances for subsequent configuration in the code snippet below (Figure 6.4).



```
---
# Tasks for the instance_launcher

 - name: Launch NeCTAR Instances
   shell: python roles/instance_launcher/files/launch_instances.py "{{ numinstances }}"
   register: output

 - debug:
     var: output.stdout_lines

 - name: Refresh Inventory
   meta: refresh_inventory

 - name: Add New Instances To "known_hosts"
   shell: ssh-keyscan -H "{{ item }}" >> ~/.ssh/known_hosts
   loop: "{{ groups['all'] }}"
```

Figure 6.4: main.yaml file for 'instance_launcher' playbook

The last three steps are captured in the “harvester_launcher” Playbook (see the Tasks file below). It first installs all the packages required for running the harvester script and then launches the actual script. It is important to note that the “twitharvester.py” script encapsulates such functionality as connecting to the CouchDB database, harvesting the tweets and pushing them to the respective database. Thus, full automation for the cloud deployment is achieved by using Ansible and boto functionalities.

7. Error Handling

Since the application relies on distributed systems, several techniques have been implemented to reduce errors that occur in such systems. The harvester utility has try-catch blocks to handle runtime as well as server connection errors. For the tweet harvesting, it was observed that twitter disconnects the connection after a certain number of tweet requests from the same server using the same credentials. To handle such a scenario, we implemented certain checks using multiple access credentials to identify and wait for a certain time period before attempting to reconnect. Furthermore, error management has been done at all instances where file handling is carried out. Another important feature is related to the database connection, where the application tries to connect with the database, otherwise it creates a new database with the credentials. All the server connections have proper error handling to ensure that errors are identified. The program also ensures that tweets are not repeated with different tweet IDs in the database by checking if that same tweet exists in the database after matching the actual id.

A proper logging mechanism has been added as well throughout the harvesting servers that outputs all the steps to a log file along with any errors that occur from the step the harvester instances are launched till the step where the machines are getting tweets. It also logs in the different tweets that are being ignored due to location or keywords.

8. Security

Security is another important consideration when it comes to cloud computing. It is generally considered to be a disadvantage of the cloud service due to the fact that your system and your data is stored remotely and the user largely relies on the security measures of the cloud provider to keep his data safe. In contrast, internal systems can be more protected as the user has more power over establishing its security and the data is generally less exposed to the outside world.

To compensate for this drawback, substantial measures were undertaken to provide additional security of the NeCTAR instances. This was achieved through creating a security group for the instances to communicate between each other which would restrict the access to these instances from the outside.

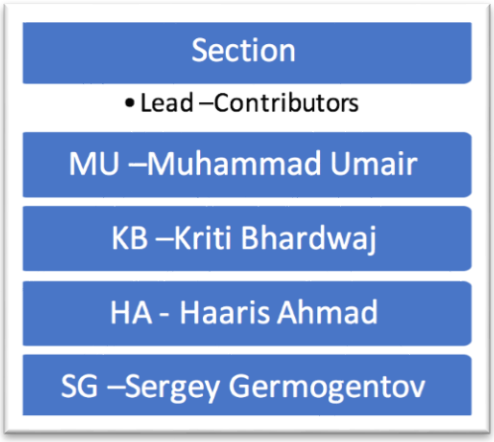
9. Availability

It is worth mentioning that one of the advantages of cloud consists of its persistent availability. This is mainly achieved by the cloud providers having the ability to provide an enormous amount of resources at a moment's notice; this allows researchers and businesses acquire necessary resources without the need to queue for them in case of increased demand. Furthermore, the cloud providers are able to carry out the service maintenance without affecting the cloud users, as they can transfer the users to another instances if necessary. On the contrast, if such service was provided internally, it would either require the users to queue up in case of massive demand or wait until the maintenance is done to regain access to the service.

However, it is necessary to mention that during the team's work with NeCTAR Research Cloud, several availability issues were encountered caused by what we believe are the technical issues of the cloud. Some of them included incorrect display of information about available instances and some would cause the instances to reject connection for a short time after their creation (to work around that bug, an arbitrary waiting time had to be imposed in one of the Ansible's Tasks). So, even though the cloud provides you with a greater degree of availability, you might become reliant on the cloud provider to maintain its services and addressing any technical issues in a prompt manner.

10. Team Dynamics

Our Team 48 which we internally named as "TwitAur", combining parts of the words Twitter and Australia, highlighting the underlying theme of the study being carried out as part of the project. We approached the project as a dynamic decentralised team, such that each major aspect of the project was being led by of the team members. Additionally, we had an overall project lead.



Section

- Lead –Contributors

MU –Muhammad Umair

KB –Kriti Bhardwaj

HA - Haaris Ahmad

SG –Sergey Germogentov

Figure 10.1: Team members

The team functioned well under the established structure with appropriate level of fluidity allowing team members to move around to different sections based on the order and load of work in a given section at any point of the project development. The project was largely broken down into four sections and the table below shows the lead for each and assistants. The names have been abbreviated (HA: Haaris, MU: Umair, KB: Kriti and SG: Sergei)

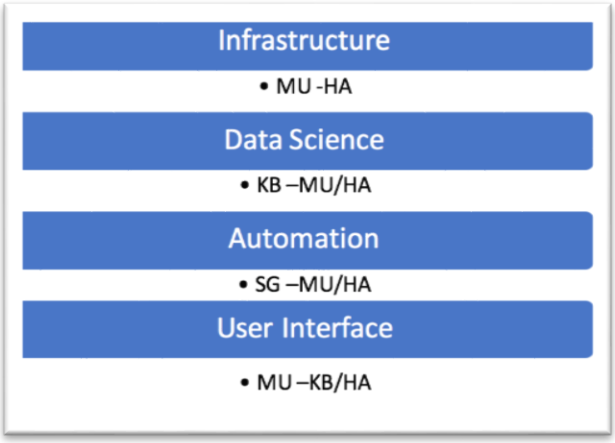


Figure 10.2: Team Roles

11. Conclusion

This project provided a great opportunity to evaluate the tweets of one of the world's most liveable cities, Melbourne. The data analysis that was facilitated via putting the learnings of Cluster and Cloud Computing course into practice, revealed interesting patterns over Greater Melbourne suburbs from various aspects. To provide a closure on all our hypotheses it is warranted that each is individually concluded.

From section 4.2.1 it was very clear that the average happiness level of the suburbs around the coast line of Greater Melbourne is significantly higher than inner city suburbs, where the former set was found to be close to 12% happier than the latter. The data discussed in section 4.2.2 reveals that there was no correlation in the proportion of married people in a suburb and its positivity, expressed in happiness score. The average happiness level of suburbs with more overseas people is higher than the suburbs with fewer people that were born out of Australia. The happiness ranges from 69.6% to 96.0% for the overseas range of 68.6 to 84.1% in Table 5.0. This suggests that multiculturalism brings more positivity to suburbs.

Comparison of the statistics obtained on crime rate from Twitter analysis with those collated by the Victorian government showed very strong correlation, both showing Melbourne city suburb to have the highest crime rate. This shows that twitter has the potential to be a tool to reveal crime hot spots. The mapping of drug and alcohol tweets revealed a couple of key clusters where the most drug/ alcohol tweets are found. One of these clusters is indeed made up of suburbs in and around the city central suburbs. However, 50% of the sports tweets come from here. This also happens to be the suburb where the majority of the top ten venues are located. Hence, revealing that twitter can help us identify most popular sporting venues.

Overall, we were able to validate all our hypotheses to reach to meaningful conclusions using the two key data sources, Twitter and AURIN.

12. References

Apache Software Foundation. (2018). *Apache CouchDB 2.1 Documentation*. Retrieved from CouchDB Relax: http://couchdb.apache.org/

AURIN. (2018). *SA2-P05_Registered_Marital_Status_by_Age_by_Sex_Census_2016*. Retrieved from AURIN: https://portal.aurin.org.au/AURIN Workbench/SA2

AURIN. (n.d.). *SA2_SEIFA_2016_-_The_Index_of_Economic_Resources__IER_-2*. Retrieved from AURIN: https://portal.aurin.org.au/AURIN Workbench/SA2 Select Data

AURIN. (n.d.). *SA2_SEIFA_2016_-_The_Index_of_Education_and_Occupation__IEO_*. Retrieved from AURIN: https://portal.aurin.org.au/AURIN Workbench/SA2 Select Data

AURIN. (n.d.). *SA2-G06_Social_Marital_Status_by_Age_by_Sex-Census_2016*. Retrieved from AURIN: https://portal.aurin.org.au/AURIN Workbench/SA2 Select Data

AURIN. (n.d.). *SA2-G53b_Industry_of_Employment_by_Occupation-Census_2016*. Retrieved from AURIN: https://portal.aurin.org.au/AURIN Workbench/SA2 Select Data

Boto. (2011). *An Introduction to boto’s EC2 interface.* Retrieved from Boto.cloudhackers: http://boto.cloudhackers.com/en/latest/ec2_tut.html

Cowling, D. (2018, February 1). *Social Media Statistics Australia – January 2018*. Retrieved from Social Media News: https://www.socialmedianews.com.au/social-media-statistics-australia-january-2018/

Highcharts. (2018). *Highcharts, Highstock and Highmaps documentation*. Retrieved from Highcharts: https://www.highcharts.com/docs

kramer, S. (2016, March 26). *These are the words you're most likely to tweet while drunk*. Retrieved from Business Insider: http://www.businessinsider.com/words-likely-tweet-while-drunk-2016-3/?r=AU&IR=T

Loria, S. (2018). *TextBlob: Simplified Text Processing*. Retrieved from TextBlob: http://textblob.readthedocs.io/en/dev/

Nectar. (2018). *Nectar Cloud*. Retrieved from Nectar: https://nectar.org.au/research-cloud/

Ozcan, S. (2016, February 2). *tweet-preprocessor*. Retrieved from Pypi: https://pypi.org/project/tweet-preprocessor/

Red Hat Inc. (2012, February 20). *Ansible Documentation.* Retrieved from Ansible 2.5: http://docs.ansible.com/ansible/latest/index.html

Roesslein, J. (2009). Retrieved from Tweepy: http://www.tweepy.org/

Toblerity. (n.d.). *Shapely*. Retrieved from Github: https://github.com/Toblerity/Shapely

Todd-Bennett, R. (2015, May). *Identifying Crime Hotspots using Twitter.* Retrieved from Computer Science and Informatics Resource: https://www.cs.cf.ac.uk/PATS2/@archive_file?c=&p=file&p=374&n=final&f=1-c1124856-CM3203_Final_Report.pdf

twitterdev. (2018). *Twitter Docs*. Retrieved from Twitter: https://developer.twitter.com/en/docs

13. APPENDIX

13.1. Appendix A: Couch DB Architecture

This section lists the main database fields and their description as saved in the CouchDB.

Database: Harvester	
Field	**Description**
_id	CouchDB ID generated from id_tweet
_rev	Auto-generated Revision ID
id_tweet	Tweet ID
lat	Tweet latitude coordinates
long	Tweet longitude coordinates
rawtweet	Raw tweet message
screen_name	Tweet account username
sentiment	Tweet Sentiment result
sports	Does the tweet contain the sports keywords?
alcohol	Does the tweet contain the alcohol keywords?
crime	Does the tweet contain the crime keywords?
suburb	Suburb Name
tweet	Cleansed tweet message
raw	Raw tweet data - includes tweet data

13.2. Appendix B: AURIN Datasets

This section lists all the datasets that were extracted from AURIN and saved to CouchDB.

13.2.1. AURIN Born Overseas data

aurin_born_overseas
_id
_rev
australia_p
sa2_name16
tot_p

13.2.2. AURIN Education, Economics and Occuption Resource Index data

aurin_edu_occ_index	aurin_eco_res_index
_id	_id
_rev	_rev
sa2_name16	sa2_name16
score	score
state_decile	state_percentile
state_percentile	state_rank

13.2.3. AURIN Marital Dataset

DB: aurin_marital
_id
_rev
f_tot_marrd_reg_marrge
f_tot_married_de_facto
f_tot_not_married
f_tot_total
m_tot_married_de_facto
m_tot_not_married
m_tot_total
p_tot_marrd_reg_marrge
p_tot_married_de_facto
p_tot_not_married
p_tot_total
sa2_main16
sa2_name16

13.3. Appendix C: Couch DB Views

This section lists all the views that were used for the respective databases

1. DB: harvester

i. **Title:** all

```
Map: function(doc) {emit([doc.suburb,doc.sentiment],1);
Reduce: function (key, values, rereduce){return sum(values)}
```

ii. **Title:** positive

```
Map: function(doc){
    if(doc.sentiment=="positive"){
    emit(doc.suburb,1)}}
Reduce: function (key, values, rereduce){return sum(values)}
```

iii. **Title:** negative

```
Map: function(doc) {
    if(doc.sentiment=="negative"){
    emit(doc.suburb,1)}}
Reduce: function (key, values, rereduce){return sum(values)}
```

iv. **Title:** time

```
Map: function(doc) {
    var dt = new Date(doc.raw.created_at);
    time = dt.toGMTString().replace(',',").replace('GMT',")
    .trim("").substring(4).replace(/ /g,"/");
    hour=dt.getUTCHours();
    emit(hour,1);
Reduce: function (key, values, rereduce){return sum(values)}
```

v. **Title:** neutral

```
Map: function(doc) {
    if(doc.sentiment=="neutral"){
    emit(doc.suburb,1)}}
Reduce: function (key, values, rereduce){return sum(values)}
```

vi. **Title:** crime

```
Map: function(doc) {
    if(doc.crime>1)
    emit(doc.suburb, 1)}
Reduce: function (key, values, rereduce){return sum(values)}
```

vii. **Title:** sports

```
Map: function(doc) {
    if(doc.sports >1)
    emit(doc.suburb, 1)}
Reduce: function (key, values, rereduce){return sum(values)}
```

viii. **Title:** alcohol

Map:
```
function(doc) {
    if(doc.alcohol>1)
    emit(doc.suburb, 1)}
```
Reduce: function (key, values, rereduce){return sum(values)}

2. **DB: aurin_eco_res_index**

i. **Title:** edu_occ_percentile

Map: function(doc) {emit(doc.sa2_name16, doc.state_percentile)}

3. **DB: aurin_marital**

i. **Title**: Details

Map:
```
function(doc) {
    if(doc.sa2_main16)
    emit(doc.sa2_name16,doc)}
```

4. **DB: aurin_born_overseas**

i. **Title**: percentage_overseas_aus

Map:
```
function(doc) {
    totalBorn = doc.australia_p;
    suburbName = doc.sa2_name16;
    totalpeople = doc.tot_p;
    totalOverseasPercentage = (totalpeople - totalBorn) * 100;
    totalOverseasPercentage = totalOverseasPercentage/totalpeople;
    totalOverseasPercentage = totalOverseasPercentage;
    totalAusBornPercentage = 100 - totalOverseasPercentage;
    emit(suburbName,[totalOverseasPercentage.toFixed(2),
    totalAusBornPercentage.toFixed(2)]}
```

5. **DB: aurin_edu_occ_index**

i. **Title**: edu_occ_percentile

Map: function(doc) {emit(doc.sa2_name16, doc.state_percentile)}