Cluster and Cloud Computing
Assignment 2 - Big Data Analytics on the Cloud

Group 50

Ruilong Wang
1074694
ruilong.wang@student.unimelb.edu.au

Congcong Zhao
1107156
congcong.zhao@student.unimelb.edu.au

Qichi Liang
1392005
qichil@student.unimelb.edu.au

Yesheng Yao
1108951
yesheng.yao@student.unimelb.edu.au

Chuqiao Wu
1386417
chuqiao.wu@student.unimelb.edu.au

May 2024

1. Introduction

People in Australia have become increasingly concerned about the crime rate in recent years. People's views on crime and the influence of sentiment on crime are interesting aspects to analyse. The impact of population growth on the crime rate has also become a topic of discussion.

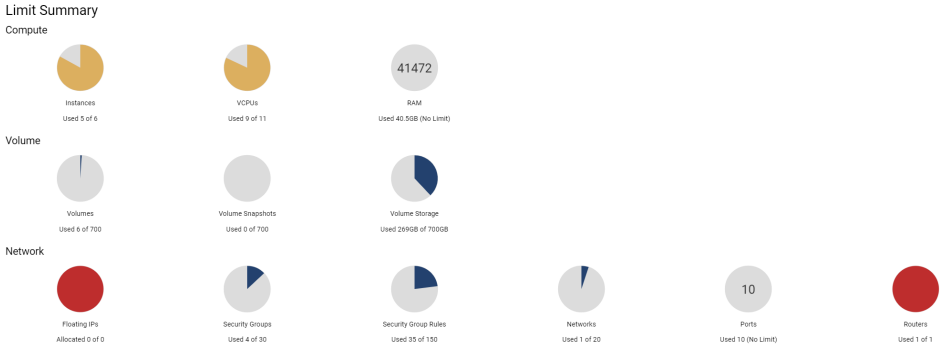
Our project uses a cloud system for collaboration, employing Elasticsearch to store and query data, and finally displays the data in various formats.

In recent years, concerns about crime have been increasing in Australia, prompting a need for more sophisticated tools to analyse and understand these trends. Our project addresses this need by leveraging the Melbourne Research Cloud (MRC) to deploy an advanced system integrating multiple technologies, including Kubernetes, Elasticsearch, and Fission. By focusing on crime perception and social sentiment, particularly through social media platforms like Twitter and Mastodon. We aim to provide insights into the relationship between public sentiment, crime reports, and population statistics. These data were obtained from the Spatial Urban Data Observatory (SUDO) [1], as well as external sources like Australian Bureau of Statistics (ABS) [2], Crime Statistics Agency (CSA) [3] and Data SA [4]. This report details the system architecture, design, implementation, and analysis scenarios that demonstrate the capabilities and potential applications of our system in addressing pressing social issues related to crime.

2. System Architecture and Design

2.1 MRC

The Melbourne Research Cloud (MRC) operates as a private cloud, offering Infrastructure-as-a-Service (IaaS) to researchers at the University of Melbourne. It provides them with on-demand access to a powerful suite of virtualized computing resources, including servers and storage. This service enables researchers to swiftly obtain scalable computational power as their research needs expand, eliminating the need to invest significant time and money in establishing their computing infrastructure.



(Figure 2.1)

As shown from the graph (Figure 2.1), We allocated 5 servers (instances) with 9 virtual CPUs (40.5GB memory total) on the Melbourne Research Cloud. What's more, we are allowed to create a maximum of up to 700 volumes for the instances and the total storage space for these volumes is 700GB. We will use these given cloud resources to build our system for the scenarios as shown in the figure below:

The whole project is deployed by leveraging the Unimelb MRC platform. And we will discuss the pros and cons of the Unimelb MRC.

2.1.1 Advantages of MRC

2.1.1.1 Costs

Cost is a major advantage of using the UniMelb Research Cloud. Firstly, there are no hardware costs, which saves us the high initial investment required for an on-premise infrastructure. Secondly, it reduces our administrative burden since we don't have to pay for electricity or daily maintenance. Additionally, the cloud services provided by MRC are free, eliminating service fees for our project. This makes it a more cost-effective option compared to expensive services from popular cloud service providers like AWS and Azure

2.1.1.2 Security

Security is a major advantage of using the UniMelb Research Cloud. Firstly, high-level security is achieved because the cloud resources are dedicated to our group. We can create a bastion node for building the SSH tunnel to meet our security so that external users can only access the MRC with encrypted key pairs through the Uni VPN, and it can be used to call remote service by using the localhost port number. Secondly, we can customise security group settings. This includes designing port usage for each instance by adjusting security group rules, which enhances the security of our cloud instances, although we only use the default ports in the installation guide.

2.1.1.3 Convenience

MRC Dashboard which allows the person with no background knowledge to create a cluster without any problems. Firstly, it provides some prepared flavours, images, volumes and so on, which is user-friendly to create the cluster with zero problems. Secondly, it supports the OpenStack CLI client and is also convenient to create the cluster through the command line by following the installation guide, which saves lots of time for cluster creation and deployment.

2.1.2 Disadvantage of MRC

2.1.2.1 Availability

Some availability issues can be one major disadvantage of using the UniMelb Research Cloud. Firstly, the MRC is 24 hours and 7 days per week accessible within and outside The University of Melbourne, or anywhere in the world. It still has the possibility that it cannot log in to the MRC. As you can see in the Ed discussion, some groups may meet MRC problems due to cinder issues, which can affect their user experience of the MRC. Secondly, we need to use a VPN to connect from outside of the university network which can cause network issues.

2.1.3 Summary of the MRC

Overall, although MRC itself has potential availability issues and the need for a VPN to access the cloud from outside the university network, MRC still offers significant advantages such as cost savings, enhanced security with dedicated resources, customizable security settings, and user-friendly tools for cluster creation.

2.2 Kubernetes

Kubernetes, often abbreviated as K8s, is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. Kubernetes helps manage clusters of containers, ensuring that they run efficiently and are highly available.[5] OpenStack CLI client provides a convenient way to create a k8s cluster through the command line so that it saves us lots of time for cluster creation. We created an elastic search cluster with one master node and three working nodes. Of these three working nodes, 2 of them for Elasticsearch and 1 for Kibana.

Apart from that, we create a shared data.yaml ConfigMap file and apply this file to the Kubernetes Config folder so that the function uses OpenStack CLI to create the config files and change the IP address to localhost 127.0.0.1. The reason for changing the IP address is that we need to use the local IP address to send requests to the bastions in the fission (Backend) that can read the ElasticSearch credentials to securely log in to the ElasticSearch Client.

2.2.1 Security

Kubernetes provides robust security features, including role-based access control (RBAC). Our cluster admin created several certificates for each group member so that only the people who added the certificate to the cluster can access the cluster, which improves the security of the cluster.

According to the installation guide. The cluster admin n node first, which will then access the remote cluster services. After each group member puts these files under the /.kube/config, they have access to the Kubernetes cluster and check all the nodes in the cluster by using the command *kubectl get nodes.*

2.2.2 Self-Healing

Kubernetes constantly monitors the health of nodes and pods. It replaces or restarts containers that fail, reschedules containers when nodes die, and kills containers that don't respond to user-defined health checks, ensuring applications remain in a healthy state without manual intervention.
Once the nodes and all the necessary services have been successfully deployed, the pods within these nodes will start utilising the node resources such as CPU and memory to complete their tasks. Sometimes, when we log into Kibana to transmit data, we might encounter errors like "500 Internal Server Error", but after a short time, it goes back to normal. Our initial assumption was that this might be due to excessive resource usage on a particular node, which sometimes causes the node to crash. The node's status will show as "not ready," as shown in the following image (Figure 2.2.1).

At this point, we need to attempt to inspect the status of the pods within the failed node, and we will find that many of the pods show a status of "terminating." (Figure 2.2.2)

Due to the inherent fault-recovery capabilities of Kubernetes, combined with the fact that we previously created three worker nodes during deployment (two nodes for Elasticsearch and one node for Kibana), Kubernetes will automatically restart these pods after a short wait. It will then reschedule these pods to available nodes, restart the failed node, and rebalance the pods. After waiting for a short while and logging back into the cluster, all services will be running normally, demonstrating Kubernetes' ability to self-heal and ensure service continuity.

2.3 ElasticSearch

Elasticsearch is an inverted index database renowned for its high-speed data retrieval capabilities. It excels in handling structured, unstructured, and semi-structured data efficiently. By deploying Elasticsearch on Kubernetes (K8s), we can leverage its powerful querying capabilities to access and

manage our data. Whether through full-text search, complex queries, or aggregations, Elasticsearch offers a robust for real-time data processing and analysis.

2.3.1 Upload test file in ElasticSearch

We can upload the file through Kibana. Our topic is the relationship between crime, sentiment and population size. However, Kibana has a limited upload size requirement. Since the Twitter file that we utilised to analyse sentiment is over 100MB, We met some problems in loading them into the dataset. To address these problems and since it is the first we upload them into the elastic search database, we decided to create a test file containing a few lines of code from the original file as a test set.

After operating to overwrite the time format, we successfully uploaded the data set and named its index Twitter. We found that after we uploaded the test data, the index was automatically created and had the corresponding settings and content as follows. (Figure 2.3.1)

We found that each Twitter has a row with a corresponding ID, rather than the uploaded files sharing one ID, so even if the file is divided into 4 parts and uploaded separately, as long as the content is in the same format, it can be put into one index. However, since we did not give each column a name and did not use a headline when uploading it as a test machine, we need to delete it first and upload it again

2.3.2 Delete index

Since we want to resubmit the data, we need to delete the index (which includes the context of this index). We used the curl code to delete it in the command line. -u represents identity authentication and then enters our account password. Our password was a 36-bit garbled code that improved security. However, we encountered an SSL certificate error during the first attempt.

We found that adding -k to ignore ssl authentication can solve the problem. (Figure 2.3.2)

Since our document size is 500+MB and the upload limit of Kibana is 100MB, we use a Python file to divide the CSV document into six equal parts. (Figure 2.3.3)

2.3.3 Reupload the files

Now we start to re-upload 6 files. This time we write column names for each of their columns and upload them to the Twitter index. Our data has the first column representing time, the second column representing sentiment, and the third column representing text, so the column names are set to the corresponding names

Next, we want to upload the second data in the same way, but we find that the same upload file intersection can only create a new index to upload, which does not meet my original intention.(Figure 2.3.4, Figure 2.3.5)

So we looked for a solution to use a Python file to get the address and then upload it. But first, we wanted to try if we could use Python to connect to the Elasticsearch client. In this way, we wrote a file to get the health, but it kept reporting an error. No matter how we modified it, it couldn't connect.(Figure 2.3.6, Figure 2.3.7)

After some investigation, we found it might be because Python was run through cmd, not the wsl subsystem, so it had not been sourced and therefore could not establish a connection. So we ran this Python file on Ubuntu and successfully got the result(Figure 2.3.8)

Next, we wrote the Python file for uploading. Our code is written like this, but it quickly gave me an error that it could not read the file that existed on the Windows system. So we decided to transfer the file to the subsystem and modify my Python code.(Figure 2.3.9, Figure 2.3.10)

We used the copy command to move it to home/yesheng yao and checked to confirm that it was in it. At this time, we modified our read path and ran it again.(Figure 2.3.11, Figure 2,3,12)

The file was successfully run, but after running for a long time, an error was reported. The error result is as follows. (Figure 2.3.13)

After analysis, we interpret it might be because the size of the uploaded file is too large, so it times out. Therefore, we changed the bulk. We cut it into 100 pieces and uploaded them. This time we added logging while running the cod and ran the results synchronously with this code

The operation was successful. Now we use the command to check the number and we find that the count has increased from 734796 to 882896. (Figure 2.3.14)

We found that the number he uploaded seemed wrong. It should be around 70,000. This might be a mistake, but he only uploaded 150,000. We need to re-import the data. Before that, we need to modify our code first so that there will be no error in the next import. This time we increased the chunk size to 1000. (Figure 2.3.15, Figure 2.3.16)

We re-ran it and found that it stopped after running for 10 seconds. After checking, the count only increased by 1000. We can see from the log that it was interrupted due to connection timeout. This time we increased the read time in the server settings. (Figure 2.3.17, Figure 2.3.18)

This time, after we modified it, we found that the data had increased by more than 70,000 items, indicating success. From then on, we can delete the index first, recreate it, and re-upload it to obtain all the data without duplication and error.(Figure 2.3.19, Figure 2.3.20)

When we finally upload all the files, we can see that there are more than 420,000 files. We uploaded our other datasets in the same way. This completes the upload of the data set.

2.3.4 Calling the database

First, we want to use sentiment and date as icons on the front end, so we wrote a Python method that can call the sentiment and date in the database. The code is as follows: After connecting to the server, request data. (Figure 2.3.21)

We found that when running it in wsl, it took a long time to run without any results. This was because it had to return a lot of value. Therefore, after discussing with my team members, we changed direction and decided to search in the elastic client first and then return the search results to the front end.

Hence, we created a view about the average sentiment of each day in the same way as before and used a JSON file to get the view. This is part of it. It was successfully displayed, but we found a problem. When we returned to the homepage and clicked the view file again, its timestamp would be reset. So although we wrote a Python file to get the sentiment through the view, we finally decided to download the CSV first and then re-upload it to make the index. Not only that, we added some data after communicating with my former teammates. (Figure 2.3.22)

We classified sentiment greater than 0 and less than 0 and made three tables showing the average sentiment greater than 0, the average sentiment less than 0 and the average sentiment equal to 0. We downloaded it to CSV and uploaded it to a new index average_sentiment so that we don't need to call the data and recalculate it when the front end is running. We can directly calculate the existing data. (Figure 2.3.23)

At this time, we just need to call the index of the new index again to return the JSON file to the front-end analysis. Therefore, we made a Python file to call the data on the index. Note that get returns 10 data by default, so we need to use the scroll function instead. Finally, it runs successfully and gets a JSON file like this. We need to pack them by fission later, and we still download the JSON file first, because it could help teammates to do their part first.

We continue to process other data using this method and then upload all the required data we need, create corresponding Python files to download the JSON files for every topic we want to analyse, and finally end the elastic search part. (Figure 2.3.24)

2.3.5 Pros and cons of ElasticSearch

Pros:

1. Elasticsearch uses inverted indexes which support efficient full-text search capabilities. This feature is particularly obvious when we analyse Twitter data. We could get the average sentiment during a certain period in a second through kibana.
2. At the same time, elasticsearch supports fuzzy search and absolute search. We can search for the text to include some crime words or search for the sentiment which is absolutely 0

Cons:

1. The problem with Elasticsearch is that it doesn't support big data upload.
2. When we return big data to the previous period, it will take a long time.
3. If the system fails and the node crashes, the fault repair time will be very long.

2.4 Fission

Fission is a Kubernetes-native serverless framework designed to enable the deployment and management of functions as services (FaaS) without managing the underlying server infrastructure. It allows developers to write short-lived functions that can be triggered by events, HTTP requests, or other triggers, and it abstracts the complexity of server management.

2.4.1 Backend Architecture

According to the picture below, we use fission as listed in the installation guide for the application backend. We used YAML specifications to deploy functions. We created timers for fetching the real-time data from the external source, which in this case is only Mastodon. We created routes for each of the functions so that these routes can be used for front-end http requests (Jupyter Notebook) to start running event-driven functions. These functions include codes with Elastic search query and return the search result from the Elasticsearch. As we mentioned above, except for Mastodon, we finally decided to upload other external sources of data to Elasticsearch by using a bulk API to upload relatively big data sets to the Elasticsearch and using Kibana UI for uploading relatively small data sets. The reason for that is that most of our data is directly obtained from the website, which is supposed to be static data and it is easy to upload after we create the index in Elasticsearch. We cut the dataset by filtering some useful information after we downloaded the data. Then we did some data processing locally and split data into several batches before we put them into the Elasticsearch (Figure 2.4.1).

(Original picture can be found here)

2.4.2 Restful API

We create several routes that can be used for front-end HTTP requests. Due to the scope of our project, we only use the get method. However, our routes still follow the style of Restful API that uses a uniform interface, leveraging standard HTTP methods for operations, making the API simple and intuitive to use. Because RESTful APIs are stateless and each request can be handled independently, the system can be more easily horizontally scaled. When the project scope increases, we can add more standard HTTP methods (such as PUT, POST, and DELETE) to handle requests as needed, improving system performance and availability.

(You can find the route list in the under the Doc package “system architecture.md” file for detailed information)

2.4.3 Functionalities

Our functions include codes with Elastic search query and return the search result from the Elasticsearch. However, as we mentioned above, our project scope is not complicated, therefore most of the functions only return the whole data under the ElasticSearch without any search filtering and we process most of the data at the front end.

(You can find the function list under the Doc package “system architecture.md” file for detailed information)

2.4.4 Future improvement

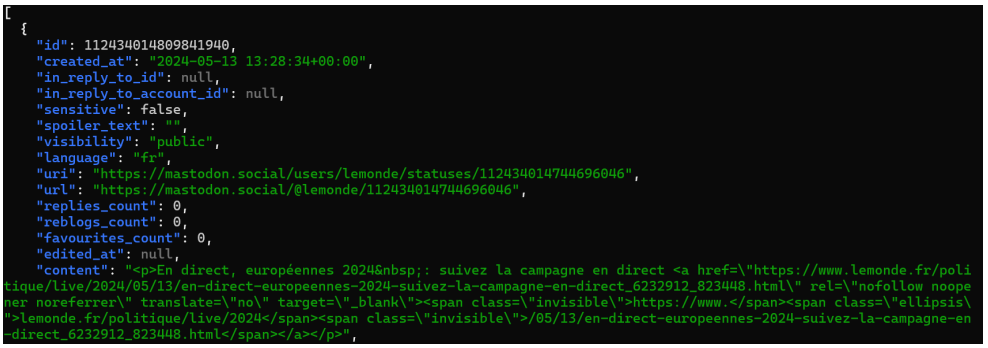
Kafka can handle large volumes of data with low latency, making it suitable for applications that require real-time data processing. A Message Queue is a form of asynchronous service-to-service communication used in serverless and microservices architectures. It allows messages (data, requests, or events) to be sent between services without requiring the services to be simultaneously available.

Based on our data source and project scope, we did not apply the Kafka cluster and message queue in our project. However, as the project scope expands, if we can find more comprehensive and more real-time data sources, we will use the combination of these two to build a more complicated application.

3. System Functionality and Scenarios

3.1 Mastodon

We want to get real-time data from the Mastodon. We modified the example codes changed the password to the elastic search set the sleep time to 30s to get the data and then printed the result in JSON format and found that the data as listed below:



```
[
  {
    "id": 112434014809841940,
    "created_at": "2024-05-13 13:28:34+00:00",
    "in_reply_to_id": null,
    "in_reply_to_account_id": null,
    "sensitive": false,
    "spoiler_text": "",
    "visibility": "public",
    "language": "fr",
    "uri": "https://mastodon.social/users/lemonde/statuses/112434014744696046",
    "url": "https://mastodon.social/@lemonde/112434014744696046",
    "replies_count": 0,
    "reblogs_count": 0,
    "favourites_count": 0,
    "edited_at": null,
    "content": "<p>En direct, européennes 2024 : suivez la campagne en direct <a href=\"https://www.lemonde.fr/poli
tique/live/2024/05/13/en-direct-europeennes-2024-suivez-la-campagne-en-direct_6232912_823448.html\" rel=\"nofollow noope
ner noreferrer\" translate=\"no\" target=\"_blank\"><span class=\"invisible\">https://www.</span><span class=\"ellipsis\
">lemonde.fr/politique/live/2024</span><span class=\"invisible\">/05/13/en-direct-europeennes-2024-suivez-la-campagne-en
-direct_6232912_823448.html</span></a></p>",
```


Figure 3.1.1

From the picture, it is obvious that it consists of different attributes such as the toot created time and content. What we want is just to make a scenario that when we ask for an HTTP request, the data would be fetched from the elastic search. However, there are several problems:

The first one is what kind of data we should use. We thought one idea was to only fetch the toot-created time, username and toot content. Since this is related to our crime topic, therefore we can show real-time information about the crime in Australia.

We found that the date from the “created_at” attribute is a UTC which does not align well with the local time. So, we change to local time by using the Pytz library. Another problem is that the data in the content attribute consists of lots of HTML tags. Therefore, we use BeautifulSoup to make the pure context with only the p tag. After simple format cleaning, we then put the data into the elasticsearch.

The second problem is the number of toots published for each period is unpredictable so that we must decide how to keep a certain amount of data and keep the whole toot set to be relatively the latest. The easiest way is to set a timer, by creating a timer with fission. We tested several combinations of the fetching time with the mastodon last id mechanism and timer which is a switch to periodically run the function. And we finally decided to set a timer to run the function every 5 minutes. Then, we created another function for deleting the old data in the elastics search to maintain the health of the elastic search nodes. Then we set a timer with another function to delete the old data each day at midnight so that when asking for HTTP requests, it would always return the latest data in the day. However, when

we checked the elasticsearch data and found that there is so much data that has been deleted, even though the time is not midnight. Then, we realised that the cron format may align with the Linux time, not the local time. Therefore, we then set the Linux time to the local time. And after that, this problem has been solved.

The third problem is that we may request a small piece of the data from the elastic search, otherwise requesting the whole data is probably time-consuming in this case. Therefore, we create a route that receives a size parameter which can be used to request a certain amount of data, controlling the data input. When we successfully implemented this functionality, we found the data is not the latest now. The reason behind this is that we did not sort data in descending order. Then we add the conditions to sort the data in descending order based on the created_at field in the data. If the value of this attribute is the same, then we used secondary sorting to compare their IDs to avoid this case.

We obtain real-time data from Mastodon and then retrieve the content to check for any criminal-related content and extract and display it.

3.1.1 Keyword Search

At first, we decided to use a keyword search. We made a list of keywords related to crime. However many of the posts filtered out by this method are not related to crime. Then we updated the search method to ensure that punctuation is removed from the content and that keywords are matched as whole words. For instance, the keyword "crime" will match "crime" but not "crimes" or "criminal". The optimised method works better to filter the posts related to crime by displaying better outputs.

3.1.2 Scikit Learn

Meanwhile, we'd like to know if there is any way to automatically classify based on content. We found a way to deploy a scikit-learn machine learning model. We first manually indicated the classification of 5 posts for training and then provided 3 unrelated posts for prediction.

The performance of the scikit-learn model is not very ideal. It can only classify text that has appeared in training and cannot classify text that has not appeared in training. This scikit-learn model requires us to manually classify the content. Due to the uncertainty of training data size, it may consume a lot of time and some results that we don't expect to have.

3.1.3 Zero-Shot Classification

We tried another pre-trained model to select the contents related to crime. The zero-shot classification evaluates each content and each content is scored based on relevancy. A good threshold must be set to ensure this classification performs well.

3.1.4 Pre-trained Scikit Learn

Later, we tried the scikit learn model again by loading pre-trained TF-IDF vectorizer and classifier models. At first, not a single piece of content was filtered out of 10000 obtained posts. We expanded the pre-trained dataset and keywords used, but the result is still not ideal, not a single piece of content has been filtered out. As the keyword search results are very good, we did not choose machine learning classification methods.

3.1.5 Pre-trained BERT

However, we found some machine-learning models that can classify emotions. Firstly, we tried a pre-trained BERT model, there are pre-trained databases for BRET. The BERT model can T perceive emotions based on contents and classify them into predetermined levels. However, the BERT model also requires fine-tuning of model parameters to achieve ideal results.

3.1.6 Pre-trained Transformers

We chose a pre-trained transformers model. This model can divide content into positive and negative content while outputting a confidence level. Because we also have research on sentiments, the model's dichotomy of content also corresponds to the positive and negative sentiments in Twitter data. So we choose this transformer model to test the emotion for each post on mastodon.

3.1.7 Final Implementation and Improvements

In the end, we modified and integrated the code so that it can obtain, retrieve, and classify real-time data of a given size. The output result is also ideal.

If we have enough time and effort to manually classify a large size of content to provide enough training data for the scikit-learn model, we believe the scikit-learn model would perform better than keyword research. Due to the exaggerated techniques used by some posts, many contents containing crime-related keywords may be related to crime. Meanwhile, more computer resources are also being occupied, and there is not much room for improvement.

3.2 Twitter

Overview

This scenario focuses on analysing Twitter sentiment data to understand public sentiment trends and their relationship with tweet volume and crime perception. The analysis spans various scenarios and uses pre-processed datasets to ensure efficient storage and transmission.

Data Preparation

The tweet data was retrieved from SPARTAN as desired. Given the size of the full dataset, a filter was applied using the 'tag' element that contains "Australia-based" or "Australia-user." After extracting these lines, the fields 'created_at', 'sentiment', and 'text' were gathered and exported to a CSV file. This task used two scripts: "filter_data.py" and "filter. slurm," and took approximately 30 seconds to complete using 2 nodes and 16 cores on SPARTAN. As a result, the file 'tweets_100Gb_filtered.csv' was 920.3 MB.

After downloading the output file from SPARTAN, we further reduced its size by removing rows with zero sentiment and all emojis from the text. The resulting file, "tweets_cleaned.csv," is 580 MB and contains about 4.3 million rows of tweets. However, the reduced file size was still considered an issue for transmission on "Elastic."

We then designed the showcase topics for the Twitter scenario. By implementing the sub-scenarios locally, we developed the front-end dataset needed to display the showcase. Finally, the dataset focused on Australia-based tweets created from 2021-06-21 to 2021-07-31. These datasets were exported to JSON files, ranging in size from 2 KB to 783 KB, which were ideal for storing on “Elastic" and transmitting through “ElasticSearch”.

Analysis Scenarios

Scenario 1: Daily Sentiment and Tweet Volume Analysis

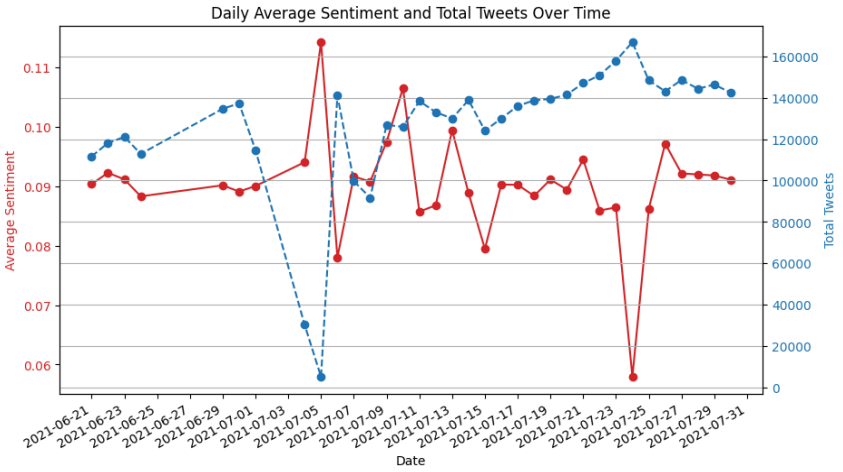


Fig. 3.2.1

This analysis presents the relationship between daily tweet volumes and average sentiment scores over a period from June 21, 2021, to July 31, 2021. We observed that peaks in tweet volume often coincide with drops in sentiment, suggesting that events triggering increased tweeting evoke strong emotional responses. Additionally, sharp changes in the graph may align with specific events, indicating public reaction intensity and engagement levels. We found that the busiest day corresponds

with the lowest average sentiment, and the quietest day corresponds with the highest average sentiment. Therefore, we selected these two days for further investigation.

Scenario 2: Sentiment Distribution on The Busiest and Quietest Days

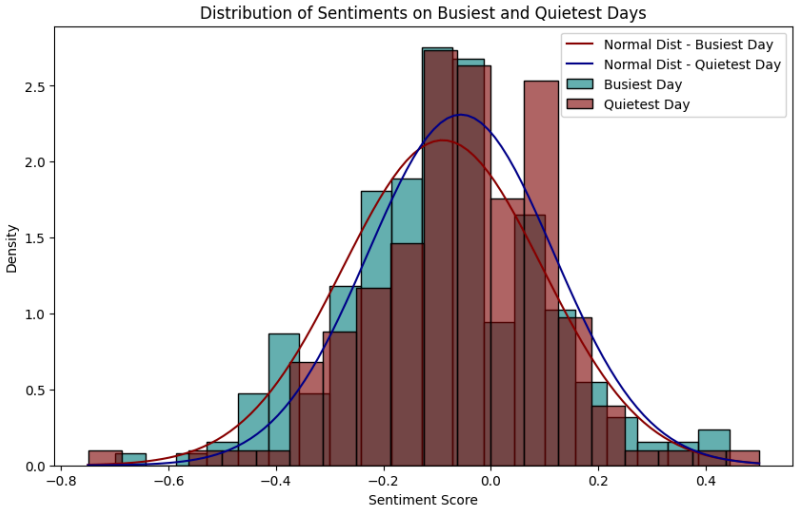


Fig. 3.2.2

This analysis of *Fig.3.2.2* compares the distribution of sentiment scores on the busiest and quietest days, visualised through histograms and normal distribution fits. Both days exhibit roughly normal distributions, but the quietest day shows a slightly more positive skew compared to the busiest day. The busiest day tends to have a broader spread of sentiment scores, suggesting more varied reactions among tweets compared to the quietest day. This visual comparison highlights how public sentiment varies between days with high and low Twitter activity, potentially reflecting different public moods or events triggering these tweets.

Scenario 3: Correlation between Crime Reports and Sentiment

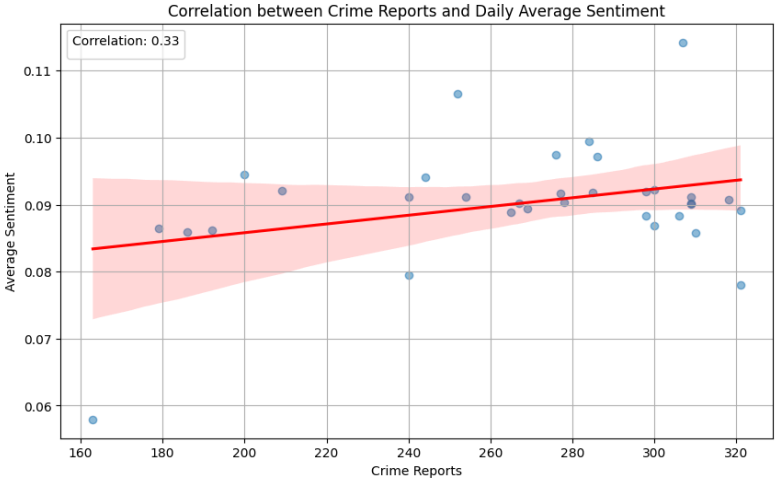


Fig.3.2.3

This analysis of Fig.3.2.3 explores the correlation between daily crime reports and average sentiment scores from tweets. The red line represents the regression line, illustrating the trend in the data. Combined with a Pearson correlation coefficient of 0.33, it suggests that higher crime reports might be associated with days of slightly more positive sentiment, possibly reflecting increased public awareness and engagement. The purpose of this scenario is to understand how public sentiment on social media corresponds with crime dynamics, potentially aiding in community-focused strategies and public safety measures.

Scenario 4: Sampling Crime-Related Tweets

As shown in Fig. 3.2.4, this scenario demonstrates the functionality of dynamically sampling and displaying five tweets related to crime from the busiest and quietest days. Every time the functions 'sample_crime_busiest_tweets()' and 'sample_crime_quietest_tweets()' are called, they randomly select five crime-related tweets with sentiments from the two days, respectively. These functions are interesting because they can help in understanding how public reactions vary with the intensity of social media activity. By executing these functions, users can quickly gather examples of public sentiment on crime-related issues from specific days, which is invaluable for research, reporting, or further analysis in social science and digital humanities.

Testing

- Test Case 1: Verify that the "get_url_data" function correctly fetches data from the API. We provide an invalid port or endpoint to the function and expect the function to handle errors by raising an informative exception.

Challenge and Improvement

- Scenario 1: This scenario used two datasets, one containing two columns{"Date", "Sentiment"}, the other containing two columns{"Date", "Number of Tweets"}. We initially demonstrated two plots of 'Daily Average Sentiment Over Time' and 'Daily Total Tweets Over Time', respectively. Since these two plots share the same x-axis, the Date, we combined them into one plot (Figure 3.2.1).
- Scenario 2: This scenario used two datasets, both containing two columns{"Sentiment", "Text"} for the busiest day and the quietest day, respectively. As the two datasets of the two days have distinct numbers of rows, we applied random sampling on the busiest day for alignment with the quietest day. The original size of the busiest day data containing about 170,000 rows was considered time-consuming for transmission through 'ElasticSearch', so we decided to make a sample size of it.

- Scenario 3: We assumed that an increase in crime reports might decrease the sentiment. Although the result shows a positive relationship between the number of crime reports and the average sentiment over time, the correlation between these two variables is still low. We also conducted an analysis that went over the "tweets_cleaned.csv" file and filtered out tweets with crime-related text, using keyword indicators from pandas. The average sentiment over time was calculated, with the bag of keywords suggested by ChatGPT. Similar correlation and linear regression analyses were applied to the crime-related data. The result also showed a weak relationship between crime reports and the crime-related average sentiment, with a p-value above 0.05, suggesting that the relationship is not statistically significant (Figure 3.2.5). The R-squared value is 0.094, indicating that only about 9.4% of the variability in sentiment can be explained by the number of crime reports. This drew our interest to conduct some qualitative analysis in Scenario 4.
- Scenario 4: When we tried to directly print the five sample tweets, the long text would be shortened with an apostrophe. To fully display the full picture of the text, we used a package called IPython.display that allows us to display the HTML with custom CSS for auto-adjusting row heights. It looks pretty good (Figure 3.2.4).

3.3 Crime vs Population (Victoria and South Australia)

Overview

This scenario focuses on the relation between crime count and population. There are three sub-scenarios in this scenario. The first sub-scenario is to find the relationship between Victoria state crime count and population discussed among different crime Subdivisions

Data Preparation

The data for the following three scenarios comprises crime or population statistics from either Victoria State or South Australia State. Crime data for both states is sourced from their respective official government websites, while population data for Victoria State is obtained from SUDO. Unfortunately, population data for South Australia is unavailable on SUDO. However, we've leveraged this limitation to develop a method for estimating the population using crime counts for the South Australia scenario.

For Victoria State, we've restricted the crime data to the years 2019 to 2023 to ensure greater validity for predictions, especially considering the impact of the recent pandemic. Additionally, we have two datasets stored on ElasticSearch categorised by crime subdivision and Local Government Area (LGA) for different sub-scenarios. The population data includes estimates for years beyond 2023, facilitating crime count predictions in subsequent scenarios, and it also includes LGA information for mapping.

In the South Australia dataset, we've eliminated rows with null values. Since population data is unavailable, we've utilised the timestamp and crime count for analysis.

	Vic Crime by LGA	Vic Crime by Crime Subdivision	Vic Population	South Australia Crime over Time
Sub Scenario 1		√	√	
Sub Scenario 2	√		√	
Sub Scenario 3		√	√	√

Sub Scenario Data Table

Analysis Scenario

Sub Scenario 1: The Relationship Between Victoria Crime Count and Population Among Different Crime Subdivisions

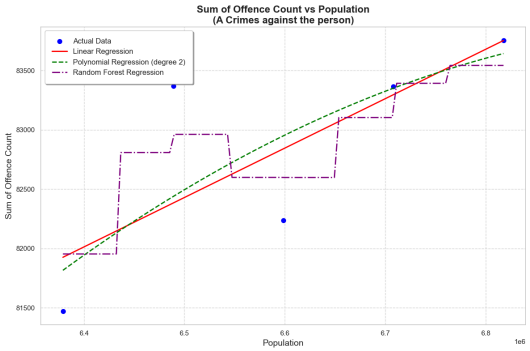
This sub-scenario presents an analysis of the relationship between crime offence counts and population in Victoria, Australia. The data spans from 2019 to 2023, focusing on specific crime subdivisions. The function includes data processing, merging, and visualisation using various regression models to understand the trends and correlations between population and crime offences. A user input function allows the selection of a specific crime subdivision, including Crimes against the person, Property and deception offences, Drug offences, Public order and security offences, and Justice procedures offences. The user selects the desired subdivision, and the analysis is filtered accordingly.

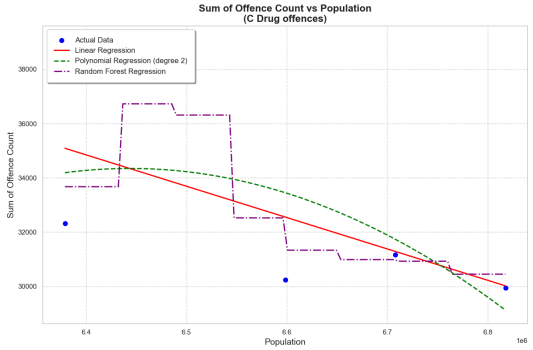
The relationship between offence counts and population was analysed using three regression models: Linear Regression, Polynomial Regression (degree 2), and Random Forest Regression. The Linear Regression model is a basic linear approach to understanding the direct linear relationship between population and offence counts. The Polynomial Regression model, with a degree of 2, captures potential non-linear relationships, while the Random Forest Regression model is a more complex, non-linear approach that can capture intricate patterns in the data.

Initially, we preprocess the Victoria state population and crime subdivision count data and merge them into a combined dataset. This combined dataset is then filtered based on the selected crime subdivision. Each regression model is trained on the filtered data, and predictions are generated using these trained models. The results are visualised with a scatter plot of the actual data and line plots for

each regression model's predictions. In the plot, the X-axis represents the population, and the Y-axis represents the sum of offence counts. The scatter plot shows the actual data of offence counts against the population, while the regression lines represent predictions from the linear, polynomial, and Random Forest models. This visual comparison helps to understand how each model fits the data, offering insights into the potential relationship between population growth and changes in crime offence counts.

Two example graphs generated using this function are presented. The graph on the left shows information for crimes against the person, while the graph on the right shows data for drug offences. From these examples, it is evident that polynomial regression performs the best due to its flexibility and ability to handle a small amount of data effectively.





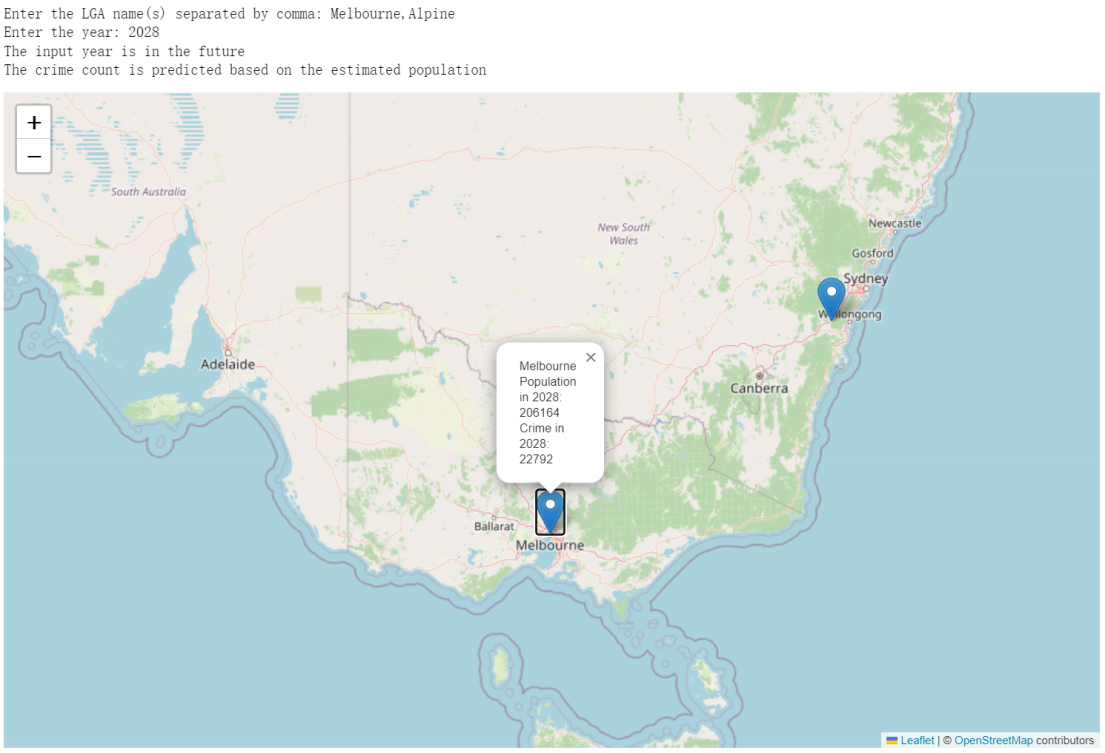
Sub Scenario 2: Population and Crime Data Visualization and Prediction for Victorian Local Government Areas

This scenario presents a function designed to fetch, process, visualise, and predict population and crime data for local government areas (LGAs) in the state of Victoria, Australia. The function provides an interactive map displaying the selected LGAs, their population figures, and crime counts for a specified year. When the specified year is in the future, between 2024 and 2031, it will present the predicted population and crime counts. The predicted population is provided by Victoria's projections, while the predicted crime counts are estimated using a polynomial regression model trained on data from 2019 to 2024 for each selected LGA.

There are two main reasons for choosing the polynomial regression model. Firstly, the training data is limited since the scope narrows down to specific LGAs, and a polynomial regression model of degree 2 is simple enough to avoid overfitting the data. Secondly, as demonstrated in sub-scenario 1, the polynomial regression model performs well in predicting crime counts for various crime subdivisions.

Two important packages are used for building the map: folium and Nominatim from geopy.geocoders. The folium package helps build the interactive user interface, providing clickable tabs with popup text displayed on OpenStreetMap. The Nominatim package assists in finding the geographic locations for each selected LGA, enabling the accurate placement of tabs on the map.

The figure below illustrates an example of using this function. The user first entered "Melbourne" and "Alpine" as the LGA names and then selected 2028 as the specified year. By clicking the tab for Melbourne, the map displays an estimated population of 206,164 and an estimated crime count of 22,792.

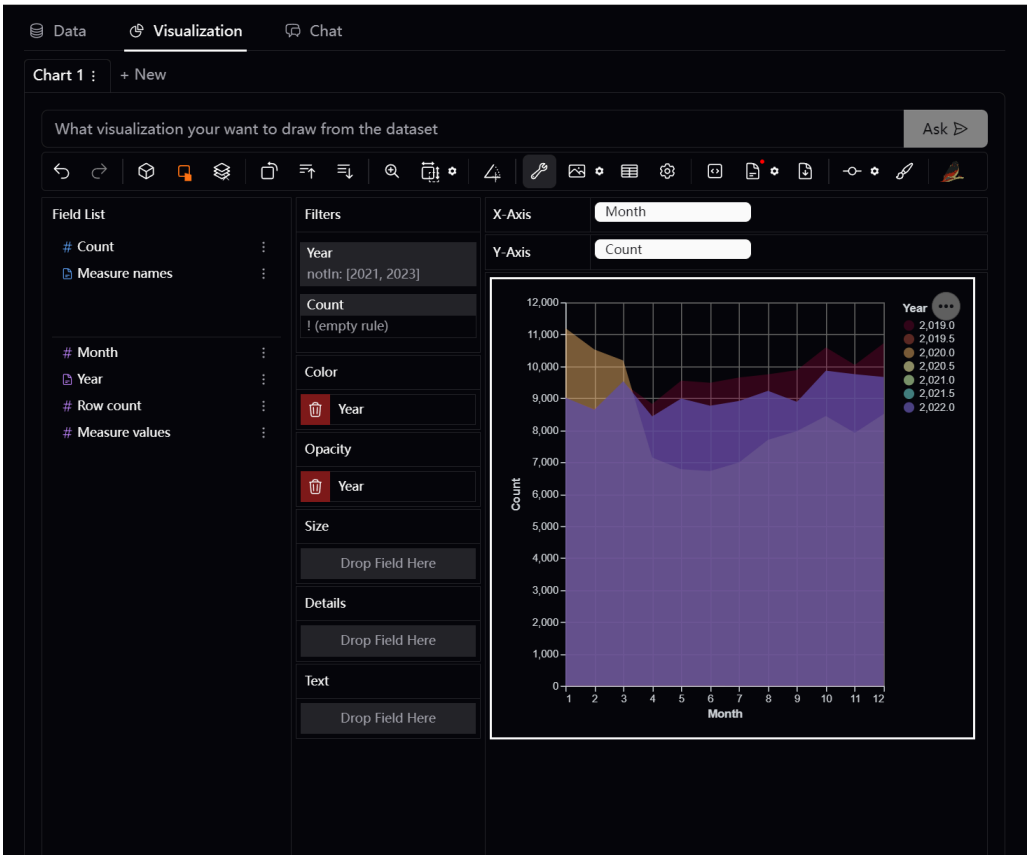


Sub Scenario 3: **The Visualisation of South Australia Crime Count Against Month and the Prediction of Population-based on Crime**

The primary objective of this sub-scenario is to visualise the historical crime data of South Australia and predict its population based on this data using various regression models, including Polynomial Regression, Linear Regression, and Random Forest Regression, trained on the population and crime data of Victoria state.

There are two stages in this sub-scenario. The first stage involves data visualisation conducted using PyGWalker to generate a visual representation of the South Australia crime data over months from the years 2019 to 2023. The functional purpose of this visualisation is to showcase the robust interactive

user interface provided by the PyGWalker package, which allows users to freely drag and choose what data to include in the graph and add filters to compare different data groups. For demonstration, we set the x-axis to be the month, the y-axis to be the crime count, and the filter to be the year to observe how crime counts vary at different times of the year for each year. The optimal view of this graph includes data for up to three years, so we present the years to include 2019, 2020, and 2022:



The next phase involves model building and training. Crime count and population data for Victoria are combined, and offence counts are summed by year. This grouped data creates the feature (X) and target (y) variables for model training. Three models are trained: a Polynomial Regression model using a pipeline with polynomial features of degree 2, a Random Forest Regression model with 100 estimators, and a simple Linear Regression model. Using these trained models, predictions are made based on yearly crime counts from the South Australia dataset in (Figure 3.3.4.1) and (Figure 3.3.4.2).

After comparing the different regression models, we observe that the Polynomial Regression model produces negative predictions and the Random Forest model yields unchanged predictions for each year. Therefore, we focus on visualising the predictions of the Linear Regression model as the above right figure shows. The final plot illustrates the predicted population over the years, providing insights into the potential influence of crime rates on population estimates.

Testing

Test Case Number	Test Case Function	Test Case Port	Router	Successful /not
1	get_vicpop_data	9030	search-vic-population	yes
2	get_subcrimecount_data	9030	search-vic-crime-by-offence-count	yes
3	get_viccrimegov_data	9030	search-vic-crimegov	yes
4	get_southaus_data	9030	search-south-crimegov	yes

Challenges

- In sub-scenario 2, we wanted to be able to show all possible locations of LGA recorded in the data and this requires us to have all the coordinate points or geometry columns of the data. However, we don't have them in the data, so we turned out to use the Nominatim to transform the name of LGA to coordinate points instead.
- During sub-scenario 3, we had a problem fixing the PygWalker graph axis with the variables that we wanted to show that users are all on their own, left with the blank interface. At the same time, we also had problems managing the style of the graph presented. We then found out the function for copying the code of fixed graph generation.
- In both sub-scenarios 2 and 3, when training the models, we faced the problem of training data too small. We tried to solve the problem by adjusting the best model to deal with each identical training set.

Improvements

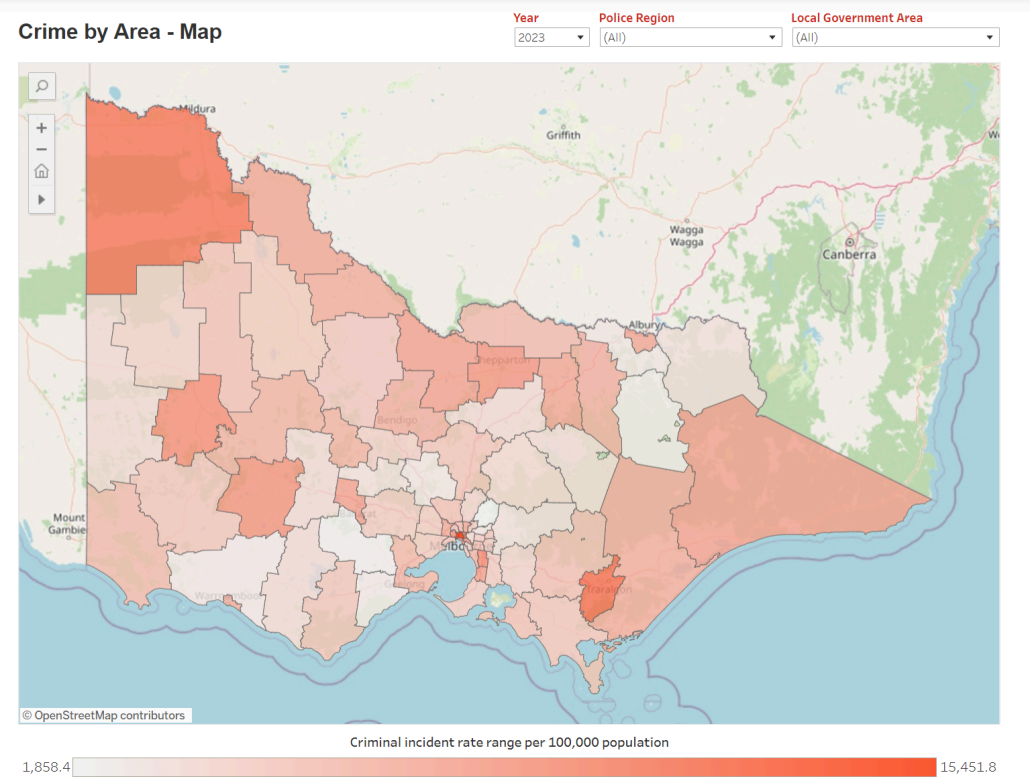
There are three parts we can improve for this scenario.

- The prediction for crime subdivision. In sub-scenario 1, if we have more time, we can also make predictions and visualisation of the future crime counts for each crime subdivision.
- The styling and loading of PyGWalker. For the colour, opacity, and legend of the PyGWalker, there are still some problems occurring if we change the filter like the colours and opacity might change to an ugly combination and the legend might happen to be in a weird format. For the loading of PyGwalker, sometimes it only shows a fixed graph not able to interact. Then the user has to rerun the function.
- The OpenStreetMap. In the map of scenario 2, sometimes the user needs to open a website nominatim.openstreetmap.org in their browser to be able to access the map. However, for

some Mac users, even if they open the website in the browser, there is still a chance to fail to access the map.

- The size of the training set. Both training sets in sub-sections 2 and 3 are too small. If we increase the training set, the precision of the predicted data will be largely increased.

3.4 Vic Crime Heatmap



The heat map for crimes in the subs of Victoria is produced by Tableau. Because the visualisation effect of Tableau is better than that of Python code. We plan to upload the data to ElasticSearch and then call the data to visualise it in Tableau. In the beginning, the data could not be uploaded to ElasticSearch normally, and Elastic Search showed that there were many garbled characters in the data. However, when viewing the CSV file locally, these garbled characters did not exist. We chose to use UTF-16 encoding to read the file and clean up the invisible characters inside. From the cleaned file content, it can be seen that the CSV file still has issues and the file content has not been properly parsed. The current file may be a string merged in one column instead of being split in the expected multi-column format. We split the content of each row to ensure that each field is processed correctly. From the sample line content, the file uses a tab (\t) as a delimiter instead of a comma. We can use tabs as delimiters to reread the file content and parse the data. The file content has been successfully parsed and correctly displayed as multiple rows and columns.

There is a simple way to use a tableau connector for elastic search published by elastic search. (https://www.elastic.co/cn/downloads/tableau-connector) However the JDBC driver requires a platinum elastic search subscription, so we had to find another way. We chose to use an elastic search web data connector and configured relevant HTML and Javascript files. After the connection is successfully established, we can directly obtain data for tableau. Meanwhile, since Tableau only needs to upload data once, the data related to this heat map in elastic search will become the cold index. After visualising the content on Tableau, we embed the chart using HTML methods.

3.4.1 Improvements

As mentioned, there is a simple way to use a tableau connector for elastic search published by elastic search if we subscribe to a platinum elastic search subscription. There is another way, to connect elastic search data to the tableau server. But Tableau server is only used by companies or organisations, we do not have the access to create a Tableau server. If we have one of the above permissions, this part can be greatly simplified.

4. Conclusion

The implementation of our system on the Melbourne Research Cloud has demonstrated the significant advantages of using cloud-based infrastructure for data analytics projects. Despite some challenges with availability and VPN access, the MRC provided a cost-effective, secure, and convenient platform for deploying our project. Utilising Kubernetes for container orchestration, Elasticsearch for data indexing and retrieval, and Fission for serverless function management, we achieved a scalable and efficient system capable of processing large datasets in real time.

Our analysis scenarios explored the relationship between public sentiment and crime perception through Twitter and Mastodon data, as well as the correlation between crime rates and population statistics. We obtained real-time data from Mastodon, fitter posts and analysed the sentiment of posts related to crime to show the public's emotions about crime.

5. Reference

[1]“AAF Discovery Service,” *ds.aaf.edu.au*. https://sudo.eresearch.unimelb.edu.au/

[2]Australian Bureau Of Statistics, “Australian Bureau of Statistics, Australian Government,” *Abs.gov.au*, 2024. https://www.abs.gov.au/

[3]V. State, “Homepage | Crime Statistics Agency Victoria,” *Vic.gov.au*, Dec. 31, 2017. https://www.crimestatistics.vic.gov.au/

[4]S. A. Police, “Crime statistics,” *data.sa.gov.au*, Available: https://data.sa.gov.au/data/dataset/crime-statistics

[5] Kumar, M., & Santhosh, K. (2022). "Serverless Computing on Kubernetes: An Overview of Fission." *Journal of Cloud Computing: Advances, Systems and Applications*.

6. Video Link

- Front end functionality (Link)
- System Architecture (Link)

7. Tools and Software

- Jupyter NoteBook
- Tableau
- MRC
- ChatGPT
- Fission
- Kubernetes
- PyGWalker
- Folium
- Github for code storage and CI pipeline for Backend (Link here and can check Actions)
- Nominatim (OpenStreetMap)

8. Appendix

2.2

Figure 2.2.1

```
congcong@LAPTOP-CUIF0VS6:/mnt/c/Users/congc/ccc-project$ kubectl get nodes -n fission
NAME                              STATUS     ROLES    AGE   VERSION
elastic-prbqujuorcuh-master-0     Ready      master   12d   v1.26.8
elastic-prbqujuorcuh-node-0       Ready      <none>   11d   v1.26.8
elastic-prbqujuorcuh-node-1       Ready      <none>   12d   v1.26.8
elastic-prbqujuorcuh-node-2       NotReady   <none>   11d   v1.26.8
```

Figure 2.2.2

```
congcong@LAPTOP-CUIF0VS6:/mnt/c/Users/congc/ccc-project$ kubectl get pods --all-namespaces -o wide --field-selector spec
.nodeName=elastic-prbqujuorcuh-node-2
NAMESPACE     NAME                                                READY   STATUS        RESTARTS        AGE    IP
       NODE                          NOMINATED NODE   READINESS GATES
elastic       elasticsearch-master-0                              0/1     Terminating   9 (60m ago)     10d    10.100.2.3
       elastic-prbqujuorcuh-node-2   <none>           <none>
fission       buildermgr-b9d6b56f9-9qzts                          1/1     Terminating   0               138m   10.100.2.2
09     elastic-prbqujuorcuh-node-2   <none>           <none>
fission       kubewatcher-5dbb4c466-v78vz                         1/1     Terminating   0               138m   10.100.2.2
08     elastic-prbqujuorcuh-node-2   <none>           <none>
fission       router-747967986f-g5t8s                             1/1     Terminating   2 (178m ago)    10d    10.100.2.6
       elastic-prbqujuorcuh-node-2   <none>           <none>
fission       timer-66c797b5c6-qx755                              1/1     Terminating   0               10d    10.100.2.7
       elastic-prbqujuorcuh-node-2   <none>           <none>
fission       webhook-7bb6d4875b-r2h2d                            1/1     Terminating   0               10d    10.100.2.4
       elastic-prbqujuorcuh-node-2   <none>           <none>
kafka         my-cluster-entity-operator-75f7879b88-slgg2         3/3     Terminating   14 (103m ago)   138m   10.100.2.2
10     elastic-prbqujuorcuh-node-2   <none>           <none>
kafka         strimzi-cluster-operator-66f59459b5-d47rz           1/1     Terminating   34 (125m ago)   3d7h   10.100.2.1
84     elastic-prbqujuorcuh-node-2   <none>           <none>
keda          keda-operator-metrics-apiserver-769cddf687-5jx9d    1/1     Terminating   3 (174m ago)    3d7h   10.100.2.1
83     elastic-prbqujuorcuh-node-2   <none>           <none>
kube-system   csi-cinder-nodeplugin-z8wgl                         3/3     Running       1 (178m ago)    12d    192.168.10
.100   elastic-prbqujuorcuh-node-2   <none>           <none>
kube-system   kube-flannel-ds-4957s                               1/1     Running       0               12d    192.168.10
.100   elastic-prbqujuorcuh-node-2   <none>           <none>
kube-system   npd-wlsqm                                           1/1     Running       0               12d    10.100.2.2
       elastic-prbqujuorcuh-node-2   <none>           <none>
```

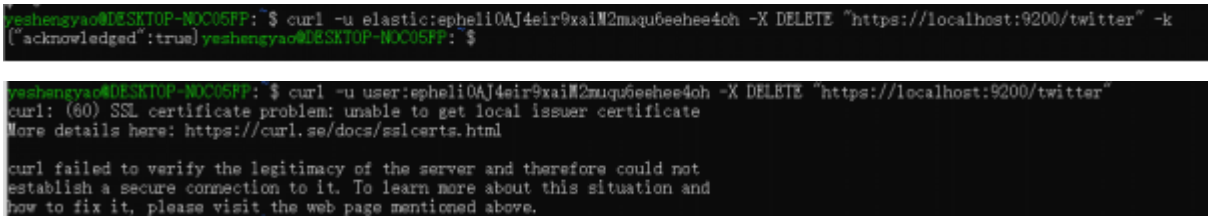
2.3

Figure 2.3.1

```
{
  "took": 0,
  "timed_out": false,
  "_shards": {
    "total": 1,
    "successful": 1,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 32,
      "relation": "eq"
    },
    "max_score": 1,
    "hits": [
      {
        "_index": "twitter",
        "_id": "Hq4xe48BDMJHZS09JjiK",
        "_score": 1,
        "_source": {
          "column1": "2021-06-21T03:18:59.000Z",
          "@timestamp": "2021-06-21T03:18:59.000Z",
          "column3": "Such a cute pupper ",
          "column2": 0.714285714285714
        }
      },
      {
        "_index": "twitter",
        "_id": "Hb4xe48BDMJHZS09JjiK",
        "_score": 1,
        "_source": {
          "column1": "2021-06-21T03:21:05.000Z",
          "@timestamp": "2021-06-21T03:21:05.000Z",
          "column3": "Nawww thanks!  *Nuzzles* right back at ya pup! ",
```

```
{
  "twitter": {
    "aliases": {

    },
    "mappings": {
      "_meta": {
        "created_by": "file-data-visualizer"
      },
      "properties": {
        "@timestamp": {
          "type": "date"
        },
        "column1": {
          "type": "date",
          "format": "iso8601"
        },
        "column2": {
          "type": "double"
        },
        "column3": {
          "type": "text"
        }
      }
    },
    "settings": {
      "index": {
        "routing": {
          "allocation": {
            "include": {
              "_tier_preference": "data_content"
            }
          }
```

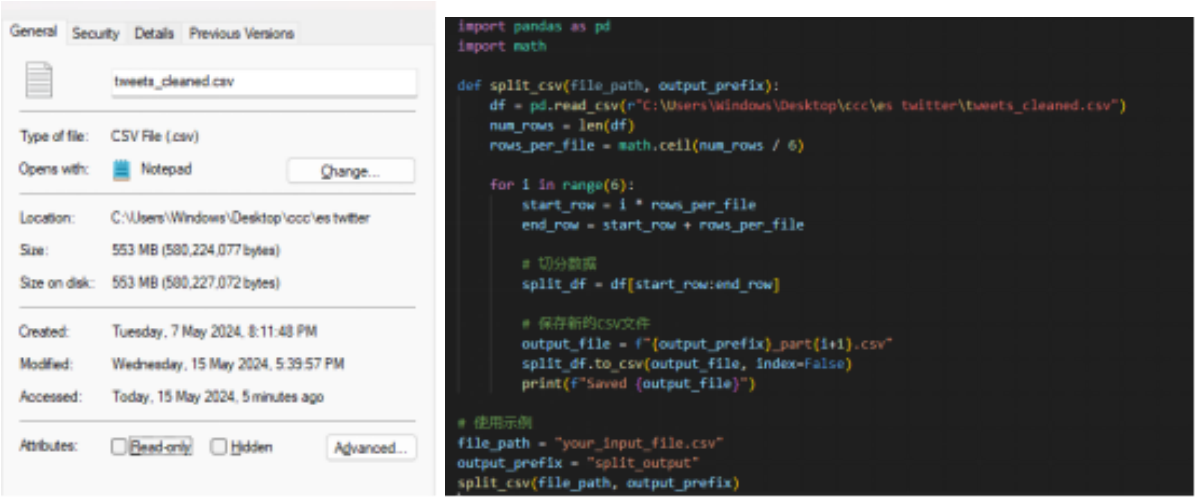

Figure 2.3.2



```
yeshengyao@DESKTOP-NOC05FP: ~ $ curl -u elastic:epheli0AJ4eir9xaiM2muqu6eehee4oh -X DELETE "https://localhost:9200/twitter" -k
{"acknowledged":true}yeshengyao@DESKTOP-NOC05FP: ~ $
```

```
yeshengyao@DESKTOP-NOC05FP: ~ $ curl -u user:epheli0AJ4eir9xaiM2muqu6eehee4oh -X DELETE "https://localhost:9200/twitter"
curl: (60) SSL certificate problem: unable to get local issuer certificate
More details here: https://curl.se/docs/sslcerts.html

curl failed to verify the legitimacy of the server and therefore could not
establish a secure connection to it. To learn more about this situation and
how to fix it, please visit the web page mentioned above.
```


Figure 2.3.3




```python
import pandas as pd
import math

def split_csv(file_path, output_prefix):
    df = pd.read_csv(r"C:\Users\Windows\Desktop\ccc\es twitter\tweets_cleaned.csv")
    num_rows = len(df)
    rows_per_file = math.ceil(num_rows / 6)

    for i in range(6):
        start_row = i * rows_per_file
        end_row = start_row + rows_per_file

        # 切分数据
        split_df = df[start_row:end_row]

        # 保存新的CSV文件
        output_file = f"{output_prefix}_part{i+1}.csv"
        split_df.to_csv(output_file, index=False)
        print(f"Saved {output_file}")

# 使用示例
file_path = "your_input_file.csv"
output_prefix = "split_output"
split_csv(file_path, output_prefix)
```

Figure 2.3.4

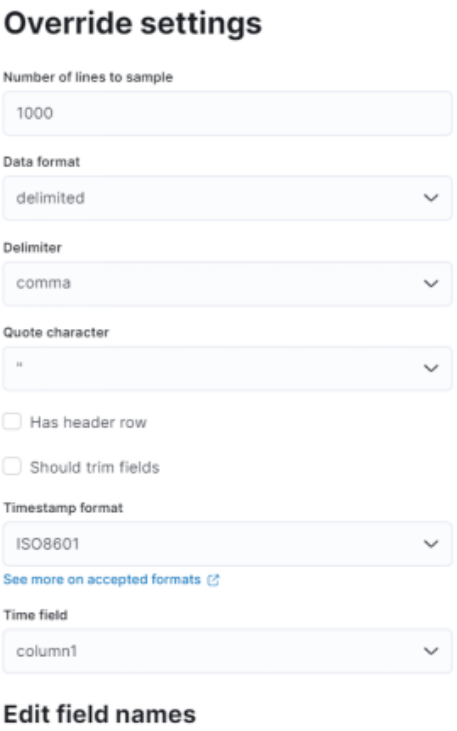


Figure 2.3.5

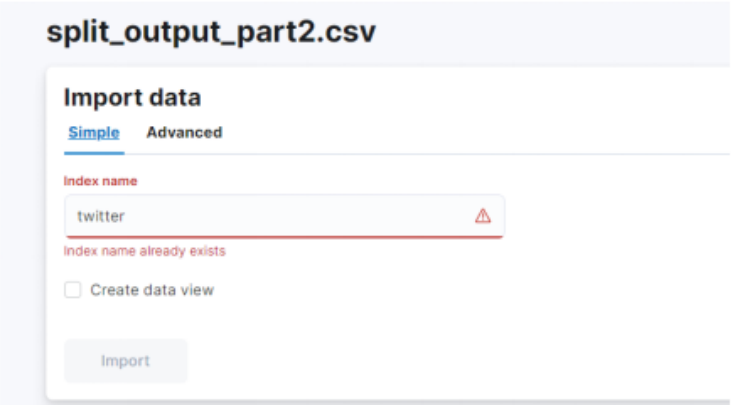


Figure 2.3.6

Figure 2.3.7

```python
from elasticsearch import Elasticsearch

es = Elasticsearch(
    ['https://localhost:9200'],
    verify_certs=False,
    ssl_show_warn= False,
    basic_auth=('elastic', 'epheli0AJ4eir9xaiM2muqu6eehee4oh')
)

# Example to check cluster health
try:
    health = es.cluster.health()
    print("Cluster health:", health)
except Exception as e:
    print("Error connecting to Elasticsearch:", e)
```

Figure 2.3.8

```
yeshengyao@DESKTOP-MOC05FP:~$ python3 /mnt/c/Users/Windows/Desktop/ccc/es\ twitter/upload_rest_data.py
Cluster health: {'cluster_name': 'elasticsearch', 'status': 'green', 'timed_out': False, 'number_of_nodes': 2, 'number_of_data_nodes': 2, 'active_primary_shards': 20, 'active_shards': 40, 'relocating_shards': 0, 'initializing_shards': 0, 'unassigned_shards': 0, 'delayed_unassigned_shards': 0, 'number_of_pending_tasks': 0, 'number_of_in_flight_fetch': 0, 'task_max_waiting_in_queue_millis': 0, 'active_shards_percent_as_number': 100.0}
```

Figure 2.3.9

```python
from elasticsearch import Elasticsearch
import pandas as pd
from elasticsearch.helpers import bulk

es = Elasticsearch(
    ['https://localhost:9200'],
    verify_certs=False,
    ssl_show_warn= False,
    basic_auth=('elastic', 'epheli0AJ4eir9xaiM2muqu6eehee4oh')
)


df = pd.read_csv("C:\Users\Windows\Desktop\ccc\es twitter\split_output_part2.csv")
df.fillna("", inplace=True)
records = df.to_dict(orient='records')

def gen_data():
    for record in records:
        yield {
            "_index": "twitter",
            "_source": record,
        }

bulk(es, gen_data())
```

Figure 2.3.10

```
yeshengyao@DESKTOP-MOC05FP:~$ python3 /mnt/c/Users/Windows/Desktop/ccc/es\ twitter/upload_rest_data.py
Traceback (most recent call last):
  File "/mnt/c/Users/Windows/Desktop/ccc/es twitter/upload_rest_data.py", line 13, in <module>
    df = pd.read_csv(r"C:\Users\Windows\Desktop\ccc\es twitter\split_output_part2.csv")
  File "/home/yeshengyao/.local/lib/python3.10/site-packages/pandas/io/parsers/readers.py", line 1026, in read_csv
    return _read(filepath_or_buffer, kwds)
  File "/home/yeshengyao/.local/lib/python3.10/site-packages/pandas/io/parsers/readers.py", line 620, in _read
    parser = TextFileReader(filepath_or_buffer, **kwds)
  File "/home/yeshengyao/.local/lib/python3.10/site-packages/pandas/io/parsers/readers.py", line 1620, in __init__
    self._engine = self._make_engine(f, self.engine)
  File "/home/yeshengyao/.local/lib/python3.10/site-packages/pandas/io/parsers/readers.py", line 1880, in _make_engine
    self.handles = get_handle(
  File "/home/yeshengyao/.local/lib/python3.10/site-packages/pandas/io/common.py", line 873, in get_handle
    handle = open(
FileNotFoundError: [Errno 2] No such file or directory: 'C:\\Users\\Windows\\Desktop\\ccc\\es twitter\\split_output_part2.csv'
```

Figure 2.3.11

Figure 2.3.12

Figure 2.3.13

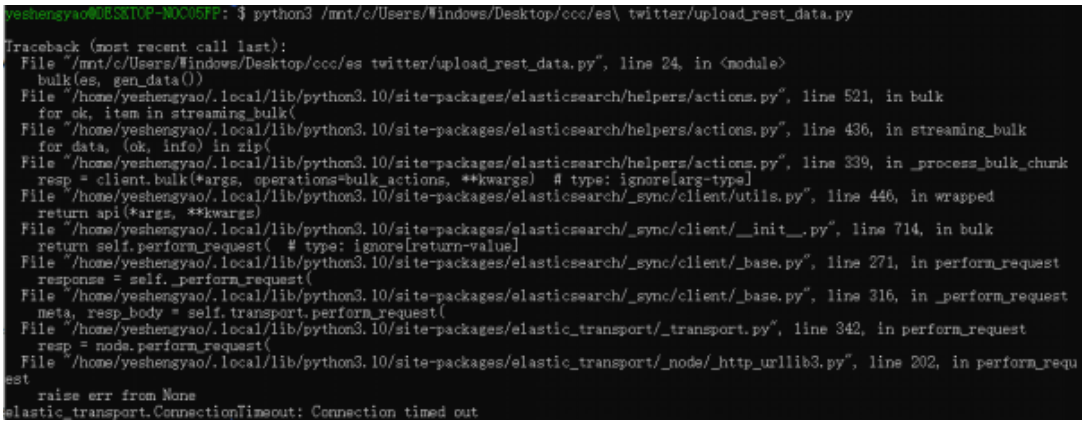


Figure 2.3.14

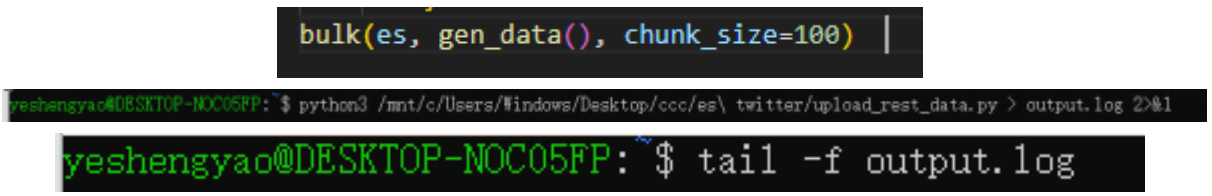


Figure 2.3.15

Figure 2.3.16

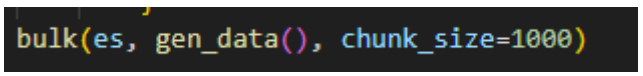


Figure 2.3.17

Figure 2.3.18

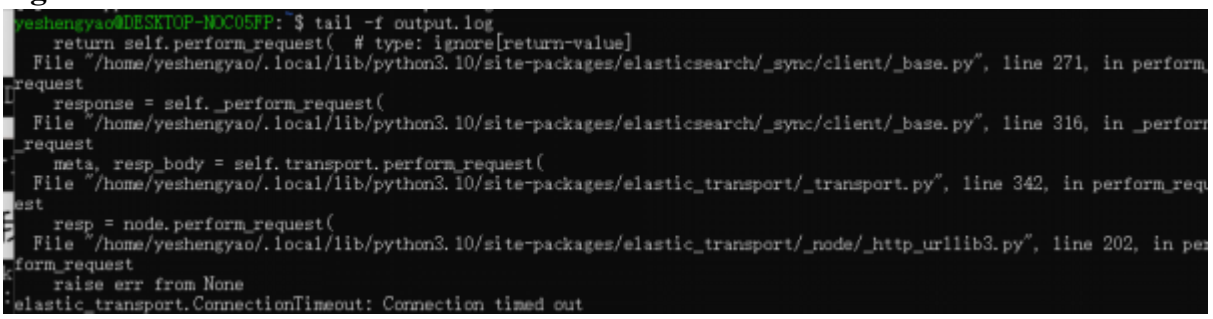


Figure 2.3.19

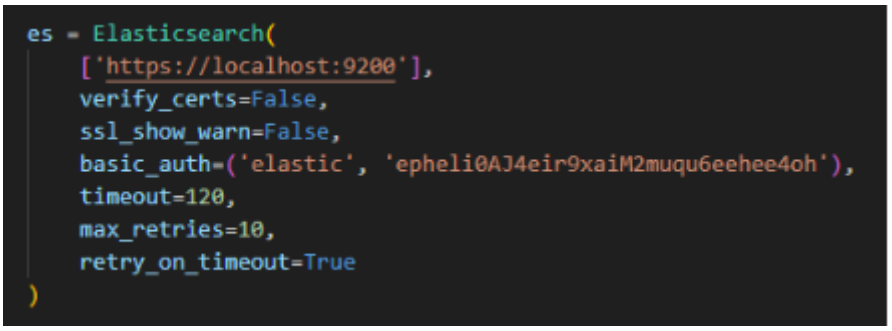


Figure 2.3.20

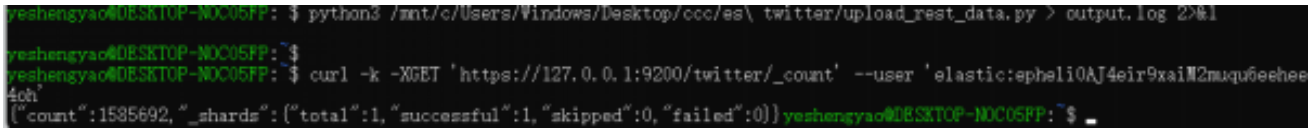


Figure 2.3.21

```python
app = Flask(__name__)

def search_twitter():
    client = Elasticsearch(
        'https://elasticsearch-master.elastic.svc.cluster.local:9200',
        verify_certs=False,
        ssl_show_warn=False,
        basic_auth=('elastic', 'epheli0AJ4eir9xaiM2muqu6eehee4oh')
    )

    query = {
        "_source": ["sentiment", "time"],
        "query": {
            "match_all": {}
        }
    }

    response = client.search(index="twitter", body=query)

    results = []
    for doc in response['hits']['hits']:
        twitter_list = doc['_source'].get('twitter', [])
        for item in twitter_list:
            created_at = item['Created_At']
            results.append(created_at)
            app.logger.info(f'Found Created_At: {created_at}')

    return json.dumps(results)

@app.route('/search')
def search():
    return search_twitter()

if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO)
    app.run(debug=True)
```

Figure 2.3.22

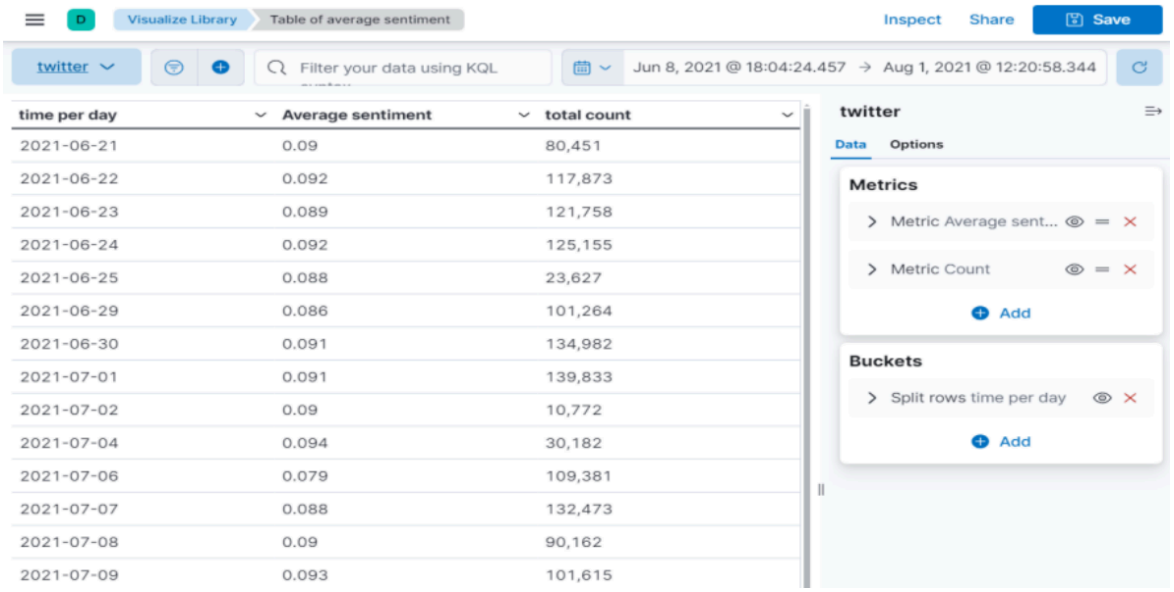
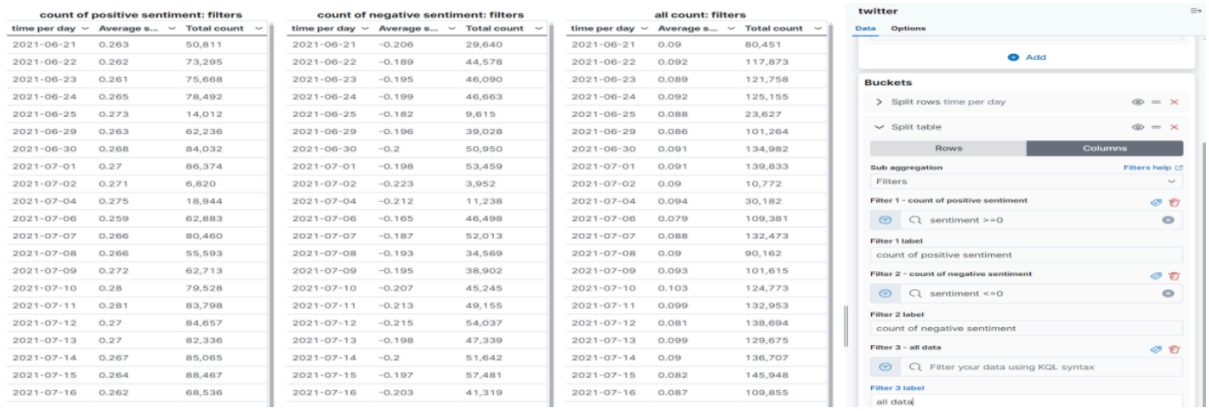


Figure 2.3.23



count of positive sentiment: filters

time per day	Average s...	Total count
2021-06-21	0.263	50,811
2021-06-22	0.262	73,295
2021-06-23	0.261	75,668
2021-06-24	0.265	78,492
2021-06-25	0.273	14,012
2021-06-29	0.263	62,236
2021-06-30	0.268	84,032
2021-07-01	0.27	86,374
2021-07-02	0.271	6,820
2021-07-04	0.275	18,944
2021-07-06	0.259	62,883
2021-07-07	0.266	80,460
2021-07-08	0.266	55,593
2021-07-09	0.272	62,713
2021-07-10	0.28	79,528
2021-07-11	0.281	83,798
2021-07-12	0.27	84,657
2021-07-13	0.27	82,336
2021-07-14	0.267	85,065
2021-07-15	0.264	88,467
2021-07-16	0.262	68,536

count of negative sentiment: filters

time per day	Average s...	Total count
2021-06-21	-0.206	29,640
2021-06-22	-0.189	44,578
2021-06-23	-0.195	46,090
2021-06-24	-0.199	46,663
2021-06-25	-0.182	9,615
2021-06-29	-0.196	39,028
2021-06-30	-0.2	50,950
2021-07-01	-0.198	53,459
2021-07-02	-0.223	3,952
2021-07-04	-0.212	11,238
2021-07-06	-0.165	46,498
2021-07-07	-0.187	52,013
2021-07-08	-0.193	34,569
2021-07-09	-0.195	38,902
2021-07-10	-0.207	45,245
2021-07-11	-0.213	49,155
2021-07-12	-0.215	54,037
2021-07-13	-0.198	47,339
2021-07-14	-0.2	51,642
2021-07-15	-0.197	57,481
2021-07-16	-0.203	41,319

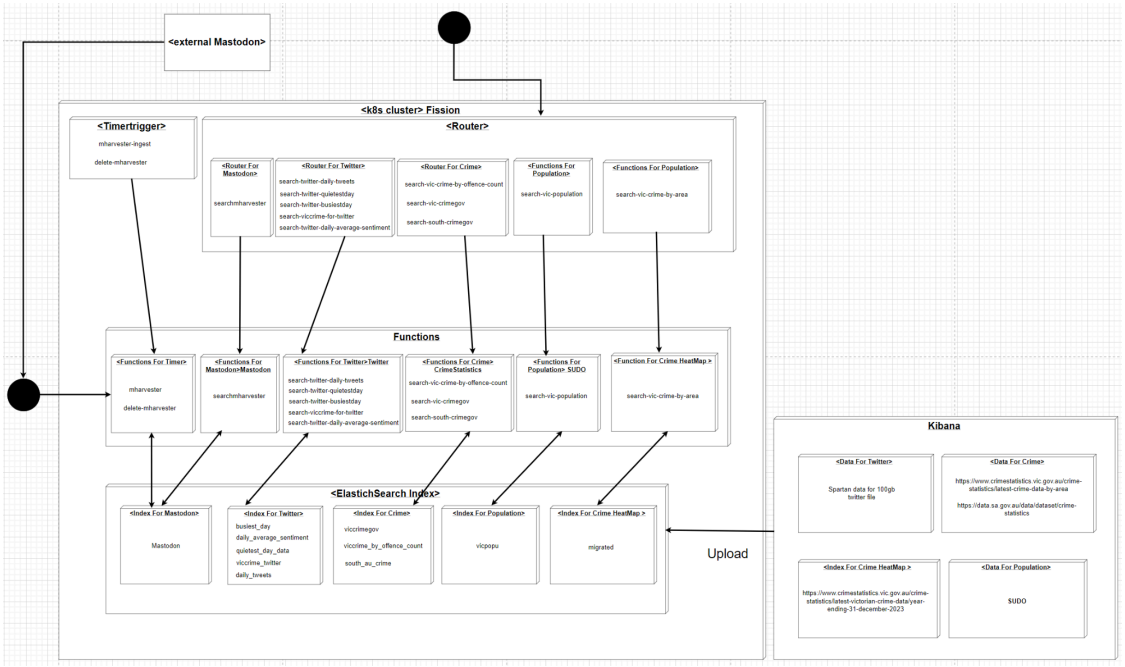
all count: filters

time per day	Average s...	Total count
2021-06-21	0.09	80,451
2021-06-22	0.092	117,873
2021-06-23	0.089	121,758
2021-06-24	0.092	125,155
2021-06-25	0.088	23,627
2021-06-29	0.086	101,264
2021-06-30	0.091	134,982
2021-07-01	0.091	139,833
2021-07-02	0.09	10,772
2021-07-04	0.094	30,182
2021-07-06	0.079	109,381
2021-07-07	0.088	132,473
2021-07-08	0.09	90,162
2021-07-09	0.093	101,615
2021-07-10	0.103	124,773
2021-07-11	0.099	132,953
2021-07-12	0.081	138,694
2021-07-13	0.099	129,675
2021-07-14	0.09	136,707
2021-07-15	0.082	145,948
2021-07-16	0.087	109,855

Figure 2.3.24

```python
index_name = 'average_sentiment'
scroll_size = 1000
scroll_time = '2m'
response = es.search(
    index=index_name,
    body={
        "query": {"match_all": {}},
        "size": scroll_size
    },
    scroll=scroll_time
)
scroll_id = response['_scroll_id']
data = response['hits']['hits']
while True:
    response = es.scroll(
        scroll_id=scroll_id,
        scroll=scroll_time
    )
    if len(response['hits']['hits']) == 0:
        break
    data.extend(response['hits']['hits'])
    scroll_id = response['_scroll_id']
all_docs = [doc['_source'] for doc in data]
save_path = "/mnt/c/Users/Windows/Documents/GitHub/ccc-project/es twitter/average_sentiment_data.json"
os.makedirs(os.path.dirname(save_path), exist_ok=True)
with open(save_path, 'w') as json_file:
    json.dump(all_docs, json_file, indent=2)
```

Figure 2.4.1



3.1

Figure 3.1.1 Keyword Search

```
def check_crime_related(content, keywords):
    stemmer = PorterStemmer()

    keyword_stems = {stemmer.stem(keyword) for keyword in keywords}

    content_lower = content.lower()
    content_clean = content_lower.translate(str.maketrans('', '', string.punctuation))

    words = word_tokenize(content_clean)
    word_stems = {stemmer.stem(word) for word in words}

    return any(keyword in word_stems for keyword in keyword_stems)
```

```
37 of 10000 may be related to crime

Time: 2024-05-19 17:58:0
Content: Corrupt government models greedy theft to corporations who rob U.K. blind. As all could see our bills are being hiked with no oppositio
n. Expect teethless Keith or the greens to do anything about it? Dream on. UK profiteering as study finds margins rose 30% post-pandemic.https://
www.theguardian.com/business/article/2024/may/15/uk-firms-accused-of-profiteering-study-finds-margins-rose-30-percent-post-pandemic
Time: 2024-05-19 17:55:1
Content: ICYMI: #AMUpdate #News Lawmakers, advocates renew calls to pass missing and murdered Black women and girls task force bill https://wausa
upilotandreview.com/2024/05/17/lawmakers-advocates-renew-calls-to-pass-missing-and-murdered-black-women-and-girls-task-force-bill/?utm_source=dlv
r.it&utm_medium=mastodon
Time: 2024-05-19 17:48:4
Content: Reasons to ban llm stuff, per Gentoo Council member Michał Górny:“1. Copyright concerns.  At this point, the copyright situation around
generated content is still unclear.  What's pretty clear is that pretty much all LLMs are trained on huge corpora of copyrighted material, and al
l fancy "AI" companies don't give shit about copyright violations. In particular, there's a good risk that these tools would yield stuff we can't
legally use.2. Quality concerns.  LLMs are really great at generating plausibly looking bullshit.  I suppose they can provide good assistance if
you are careful enough, but we can't really rely on all our contributors being aware of the risks.3. Ethical concerns.  As pointed out above, the
"AI" corporations don't give shit about copyright, and don't give shit about people.  The AI bubble is causing huge energy waste.  It is giving a
great excuse for layoffs and increasing exploitation of IT workers.  It is driving enshittification of the Internet, it is empowering all kinds o
f spamand scam.”
Time: 2024-05-19 17:45:0
Content: Pep Guardiola warns of West Ham threat and expects repeat of final-day drama against Aston Villa in 2022https://www.skysports.com/footba
ll/news/11095/13137819/pep-guardiola-warns-of-west-ham-threat-and-expects-repeat-of-final-day-drama-against-aston-villa-in-2022?utm_source=flipbo
ard&utm_medium=activitypub Posted into Top Stories in Sport @top-stories-in-sport-FlipboardUK
```

```
Content: Oscar Wilde #OnThisDayOn 19th of May 1897 Oscar Wilde was released from prison. His crime 'gross indecency' , accused and convicted of h
omosexuality, and sentenced to two years hard labour. The 1885 Criminal Law Amendment Act made all homosexual acts of 'gross indecency' illegal.
The bill was primarily concerned with the protection of women and girls by increasing the age of consent and yet this small section in the Act wa
s a pivotal change in homosexual legislation.https://theorkneynews.scot/2024/05/19/oscar-wilde-onthisday/
Time: 2024-05-19 16:24:5
Content: "Why is it so hard for western politicians to grasp the broader, existential nature of the Russian threat? Recurring spying rows, sabota
ge, assassinations, arson and cyber-hacks show Moscow "is waging war on European countries", Russia expert Edward Lucas warned. "How is it that R
ussia, a country with an Italy-sized economy, is able to attack the entire west with impunity? The answer is that Russia does not take us serious
ly."#NATO #defense #US #EU #Ukraine #war https://www.theguardian.com/commentisfree/article/2024/may/18/natos-failure-to-save-ukraine-raises-an-ex
istential-question-what-on-earth-is-it-for
```

Figure 3.1.2 Scikit Learn

```python
data = [
    {'content': 'https://www.europesays.com/1221829/ Gratulerer med dagen!🇳🇴🇳🇴 #norway', 'category': 'celebration'},
    {'content': 'Body Shop stores have closed, but we can't let its products disappear...', 'category': 'news'},
    {'content': 'Thelma the Unicorn review – sunny Netflix cartoon offers simple pleasures...', 'category': 'review'},
    {'content': 'Brighton's Katie Robinson: 'I want to be playing with and against the best'...', 'category': 'sports'},
    {'content': 'Week in wildlife – in pictures: amorous frogs, battling stallions...', 'category': 'nature'}
]

texts = [item['content'] for item in data]
labels = [item['category'] for item in data]

X_train, X_test, y_train, y_test = train_test_split(texts, labels, test_size=0.2, random_state=42)

model = make_pipeline(TfidfVectorizer(), MultinomialNB())

model.fit(X_train, y_train)

predicted_labels = model.predict(X_test)

accuracy = metrics.accuracy_score(y_test, predicted_labels)
print(f"Accuracy: {accuracy}")
print(metrics.classification_report(y_test, predicted_labels))

new_contents = [
    'Breaking news: Major earthquake hits the city',
    'Happy birthday to you! Wishing you a wonderful day!',
    'Match review: The game was thrilling with both teams giving their best'
]

predicted_new_labels = model.predict(new_contents)
for content, label in zip(new_contents, predicted_new_labels):
    print(f"Content: {content}\nPredicted Category: {label}\n")
```

```
Accuracy: 0.0
              precision    recall  f1-score   support

 celebration       0.00      0.00      0.00       0.0
        news       0.00      0.00      0.00       1.0

    accuracy                           0.00       1.0
   macro avg       0.00      0.00      0.00       1.0
weighted avg       0.00      0.00      0.00       1.0

Content: Breaking news: Major earthquake hits the city
Predicted Category: review

Content: Happy birthday to you! Wishing you a wonderful day!
Predicted Category: sports

Content: Match review: The game was thrilling with both teams giving their best
Predicted Category: sports
```

Figure 3.1.3 Zero-Shot Classification

```python
classifier = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")
```

```
crime_related_records = []
for item in filtered_data:
    content = item['content']
    result = classifier(content, candidate_labels=crime_labels, multi_label=True)
    if any(score > 0.5 for score in result['scores']):
        crime_related_records.append(content)
```

```
0
Ich würde mich ja nicht allein auf den Wahlomat verlassen, aber ich bin ganz froh, dass ich am wenigsten mit der AfD übereinstimme.
non-crime

Loneliness Is a Problem That A.I. Won't SolveHuman touch matters. #presshttps://www.nytimes.com/2024/05/18/opinion/artificial-intelligence-loneliness.html?utm_source=press.coop
non-crime

Disneyland character and parade performers in California vote to join labor unionDisneyland performers who help bring Mickey Mouse, Cinderella and other beloved characters to life have chosen to unionize following a three-day vote #presshttps://www.independent.co.uk/news/disneyland-ap-california-anaheim-cinderella-b2547455.html?utm_source=press.coop
non-crime

Artist: notafurrytho https://www.furaffinity.net/view/56674451/ #furry #yiff #furryartwork
non-crime


non-crime

https://www.europesays.com/1226027/ MP @ Szentendre #hungary
non-crime

sunbeams flicker offbirds flying into the clouds -fleeting vision ends#haiku #smallpoems Kigo: 鳥雲に入る (Tori kumo ni iru) - Birds flying into the clouds
non-crime

🌄                    Dobro jutro 🍵🌿❤️.#photography #flowers
non-crime

#Aerotelegraph - Ethiopian Airlines eröffnet neues Inlandsterminal in Addis Abeba - https://www.aerotelegraph.com/ethiopian-airlines-eroeffnet-neues-inlandsterminal-in-addis-abeba #Ticker
non-crime
```

Figure 3.1.4 Pre-trained Scikit Learn

```
categories = ['rec.sport.hockey', 'sci.crypt', 'talk.politics.mideast', 'sci.space']
data = fetch_20newsgroups(subset='train', categories=categories)
X_train = data.data
```

```
y_train = ['crime' if any(keyword in text.lower() for keyword in crime_keywords) else 'non-crime' for text in X_train]

pipeline = make_pipeline(TfidfVectorizer(), LogisticRegression())
pipeline.fit(X_train, y_train)

joblib.dump(pipeline.named_steps['tfidfvectorizer'], 'tfidf_vectorizer.joblib')
joblib.dump(pipeline.named_steps['logisticregression'], 'text_classifier.joblib')

filtered_data = [{'time': item['_source']['created_at'], 'content': item['_source']['content']} for item in data]

vectorizer = joblib.load('tfidf_vectorizer.joblib')
classifier = joblib.load('text_classifier.joblib')

contents = [item['content'] for item in filtered_data]
X = vectorizer.transform(contents)

predictions = classifier.predict(X)

crime_related_records = [content for content, pred in zip(contents, predictions) if pred == 'crime']
```

Figure 3.1.5 Pre-trained BERT

```
dataset = load_dataset('ag_news')

model_name = 'bert-base-uncased'
tokenizer = BertTokenizer.from_pretrained(model_name)
model = BertForSequenceClassification.from_pretrained(model_name, num_labels=4)

def preprocess_function(examples):
    return tokenizer(examples['text'], padding='max_length', truncation=True)

tokenized_datasets = dataset.map(preprocess_function, batched=True)

training_args = TrainingArguments(
    output_dir='./results',
    evaluation_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,
    weight_decay=0.01,
)

metric = load_metric("accuracy")

def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = torch.argmax(logits, dim=-1)
    return metric.compute(predictions=predictions, references=labels)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets['train'],
    eval_dataset=tokenized_datasets['test'],
    compute_metrics=compute_metrics,
)
```

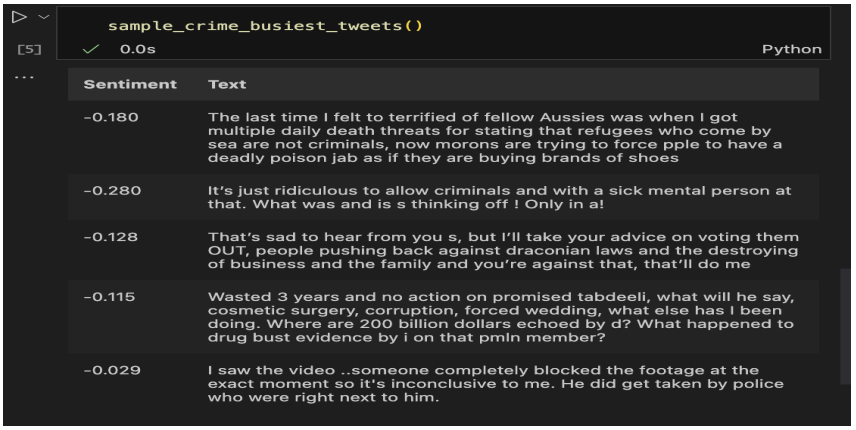
Figure 3.1.6 Pre-trained Transformers

```
classifier = pipeline("text-classification", model="distilbert-base-uncased-finetuned-sst-2-english")


results = classifier([item['content'] for item in filtered_data])

print(results)

for item, result in zip(filtered_data, results):
    print(f"Content: {item['content']}\nClassification: {result}\n")
```

```
[{'label': 'NEGATIVE', 'score': 0.9949318766593933}, {'label': 'NEGATIVE', 'score': 0.9908382296562195}, {'label': 'NEGATIVE', 'score': 0.9964911
341667175}, {'label': 'NEGATIVE', 'score': 0.9955998659133911}, {'label': 'POSITIVE', 'score': 0.9991963505744934}, {'label': 'NEGATIVE', 'scor
e': 0.9988313317298889}, {'label': 'NEGATIVE', 'score': 0.9344297647476196}, {'label': 'NEGATIVE', 'score': 0.9990224838256836}, {'label': 'POSIT
IVE', 'score': 0.9995248317718506}, {'label': 'NEGATIVE', 'score': 0.6469268798828125}, {'label': 'NEGATIVE', 'score': 0.938513994216919}, {'labe
l': 'NEGATIVE', 'score': 0.6865880489349365}, {'label': 'POSITIVE', 'score': 0.7481212019920349}, {'label': 'NEGATIVE', 'score': 0.97313749790191
65}, {'label': 'NEGATIVE', 'score': 0.9828047156333923}, {'label': 'POSITIVE', 'score': 0.9943331480026245}, {'label': 'NEGATIVE', 'score': 0.670
9291338920593}, {'label': 'POSITIVE', 'score': 0.9870933890342712}, {'label': 'NEGATIVE', 'score': 0.7354503870010376}, {'label': 'NEGATIVE', 'sc
ore': 0.9994938373565674}, {'label': 'NEGATIVE', 'score': 0.9899071455001831}, {'label': 'NEGATIVE', 'score': 0.9983448982238777}, {'label': 'NEGA
TIVE', 'score': 0.9142429828643799}, {'label': 'NEGATIVE', 'score': 0.9771742224693298}, {'label': 'POSITIVE', 'score': 0.5602328777313232}, {'la
```

3.2

Figure 3.2.4



```
sample_crime_busiest_tweets()
[5]  ✓ 0.0s                                                    Python
```

Sentiment	Text
-0.180	The last time I felt to terrified of fellow Aussies was when I got multiple daily death threats for stating that refugees who come by sea are not criminals, now morons are trying to force pple to have a deadly poison jab as if they are buying brands of shoes
-0.280	It's just ridiculous to allow criminals and with a sick mental person at that. What was and is s thinking off ! Only in a!
-0.128	That's sad to hear from you s, but I'll take your advice on voting them OUT, people pushing back against draconian laws and the destroying of business and the family and you're against that, that'll do me
-0.115	Wasted 3 years and no action on promised tabdeeli, what will he say, cosmetic surgery, corruption, forced wedding, what else has I been doing. Where are 200 billion dollars echoed by d? What happened to drug bust evidence by i on that pmln member?
-0.029	I saw the video ..someone completely blocked the footage at the exact moment so it's inconclusive to me. He did get taken by police who were right next to him.

Figure 3.2.5

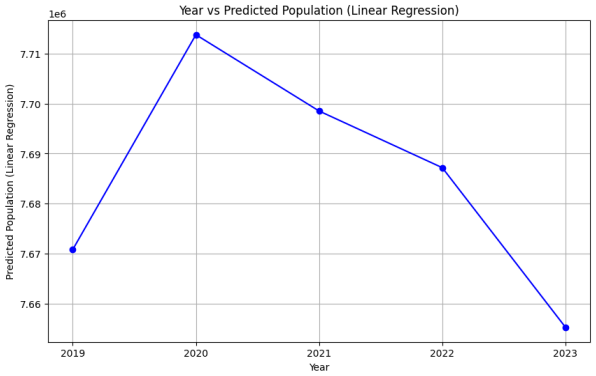
```
Pearson correlation coefficient: 0.30649216705939114
                            OLS Regression Results
==============================================================================
Dep. Variable:              Sentiment   R-squared:                       0.094
Model:                            OLS   Adj. R-squared:                  0.065
Method:                 Least Squares   F-statistic:                     3.214
Date:                Tue, 21 May 2024   Prob (F-statistic):             0.0828
Time:                        08:37:58   Log-Likelihood:                 103.61
No. Observations:                  33   AIC:                            -203.2
Df Residuals:                      31   BIC:                            -200.2
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.1010      0.012     -8.764      0.000      -0.125      -0.078
Crime_Count  7.599e-05   4.24e-05      1.793      0.083   -1.05e-05       0.000
==============================================================================
Omnibus:                        0.473   Durbin-Watson:                   1.849
Prob(Omnibus):                  0.789   Jarque-Bera (JB):                0.294
Skew:                          -0.224   Prob(JB):                        0.863
Kurtosis:                       2.886   Cond. No.                     1.67e+03
==============================================================================
```

3.3

Figure 3.3.4.1

	Year	Sum of Crime Count	Predicted Population	Predicted Population (Random Forest)	Predicted Population (Linear Regression)
0	2019	115623	-29565921	6635613	7670788
1	2020	100088	-32607025	6635613	7713815
2	2021	105602	-31513634	6635613	7698543
3	2022	109700	-30710992	6635613	7687193
4	2023	121236	-28497164	6635613	7655242

Figure 3.3.4.2



3.4

Figure 3.4 Vic Crime Heatmap

```
def read_csv_with_tab_separator(file_path):
    with open(file_path, 'r', encoding='utf-16', errors='ignore') as file:
        content = file.read()

    cleaned_content = ''.join(char for char in content if char.isprintable())

    cleaned_content = cleaned_content.replace('\r\n', '\n').replace('\r', '\n')

    lines = cleaned_content.split('\n')

    data = [line.split('\t') for line in lines if line]

    df = pd.DataFrame(data)

    return df
```

```
tableau_viz_url = "https://public.tableau.com/views/CrimeByAreaMap/CrimebyLocationMap?:language=en-US&publi

html_code = f'''
<div class='tableauPlaceholder' id='vizContainer' style='position: relative; width: 100%; height: 800px;'>
    <noscript><a href='#'>
        <img alt='Tableau Visualization' src='{tableau_viz_url}.png' style='border: none' />
    </a></noscript>
    <object class='tableauViz' width='100%' height='800' style='display:none;'>
        <param name='host_url' value='https%3A%2F%2Fpublic.tableau.com%2F' />
        <param name='embed_code_version' value='3' />
        <param name='site_root' value='' />
        <param name='name' value='CrimeByAreaMap&#47;CrimebyLocationMap' />
        <param name='tabs' value='no' />
        <param name='toolbar' value='yes' />
        <param name='static_image' value='{tableau_viz_url}.png' />
        <param name='animate_transition' value='yes' />
        <param name='display_static_image' value='yes' />
        <param name='display_spinner' value='yes' />
        <param name='display_overlay' value='yes' />
        <param name='display_count' value='yes' />
        <param name='language' value='en-US' />
    </object>
</div>
<script type='text/javascript'>
    var divElement = document.getElementById('vizContainer');
    var vizElement = divElement.getElementsByTagName('object')[0];
    vizElement.style.width = '100%';
    vizElement.style.height = (divElement.offsetWidth * 0.8) + 'px';
    var scriptElement = document.createElement('script');
    scriptElement.src = 'https://public.tableau.com/javascripts/api/viz_v1.js';
    vizElement.parentNode.insertBefore(scriptElement, vizElement);
</script>
```