Data Mining Human Reasoning: Vaccine Hesitancy in the USA

**Taha A. Kashaf**

**ID: 180020889**

**AC40001 Honours Project**

**BSc (Hons) Computing Science**

**University of Dundee, 2022**

**Supervisor: Prof. J. Lawrence**

*The purpose of this project was to use Machine-Learning powered Sentiment Analysis and Natural Language Processing techniques to classify sentiment about vaccination from textual data in the form of tweets.*

*The project endeavoured to present the results of this analysis in a clear, easy-to-approach way. Allowing users/readers to be able to view the results alongside other statistical data and see correlations and comparisons for themselves.*

*The goal being to facilitate an understanding of potential underlying reasons for vaccine hesitancy, to be able to better address it in the future.*

*While many parts of the project were successful – data collections, formatting, cleaning, pre-processing and presenting. Ultimately, the Machine-Learning powered Sentiment-Analysis model wasn’t accurate enough to draw useful conclusions from – however, this was primarily due to the lack of training data, and the model can be improved with simply more training.*

# Introduction

Vaccinations and vaccines have become a controversial talking point in this day and age. In the midst of the worlds best-documented pandemic: COVID-19. Many of us saw the development of COVID-19 vaccines as a fantastic feat of collaboration between the global medical and scientific communities. We believed that with the help of the vaccines and large-scale inoculation, multiple years of lockdowns, restrictions, and sacrifices would come to an end.

However, while that may be the predominant opinion, many people also look at vaccinations and vaccines as a bad thing. Hence the talking point of vaccines being one that brings controversy to the table.

Anti-vaccination rhetoric has been around since long before COVID-19[1]. However, in recent years, and especially with the advent of social media, the visibility of the movement has grown dramatically.

In pre-COVID times, anti-vaccination rhetoric was primarily something we’d hear about and shake our heads at. For most, it did not have any tangible real life effect, it wasn’t likely to cause any changes to our day-to-day lives.

However, currently, in the midst of a pandemic. With reports of hospital urgent care wards being filled with primarily unvaccinated covid patients[2] the effects of the anti-vax movement are bigger and closer than ever. Where even a vaccinated individual may be unable to receive care due to hospital beds being occupied by those who choose not to take vaccines.

The problem is an obvious one: large groups of people carry a negative sentiment towards vaccinations. The goal of this project was simple: To use Machine-Learning powered Sentiment Analysis, in combination with Natural Language Processing techniques to aid in producing a data-driven solution that would help us understand *why* people have the opinions they do regarding vaccines.

This is obviously not a solution to vaccine hesitancy in itself. With a topic as complex as human reasoning, and why people choose the things they do. First, you need a comprehensive understanding of *why*. Only then, once that understanding has been built, can you begin to tackle the problem itself. This project is an attempt at understanding the reasoning behind the problem.

# Background

My background research was split into three distinct categories, Data Analysis, Sentiment Analysis Techniques and Previous Attempts at Classifying Vaccine Sentiment on social media.

## Data Analysis

The field of data analysis is a large and still rapidly expanding one. One that has also come to the forefront of discussion in the last few years, as the power of data-analysis and data-driven solutions has been made clear to the general public – often in a negative light.

Perhaps the most famous example of how data analysis has changed the world is the role it played in recent large-scale political campaigns[3]. These successes have even led to legal debate regarding data privacy, and the uses of data analysis. The most well known result of such debate being the passing of legislation such as the General Data Protection Regulation (GDRP).

As we can see, there is no question about the importance or research potential within big data. The ability to analyse and extract value or meaning from large-scale unsorted data is an ever-growing field with proven real-world results.

## Sentiment Analysis Techniques

For the purposes of this project, I made use of Sentiment Analysis, a form of data analysis on written text. Sentiment Analysis comes in different forms[4], primarily one of two: Lexicon based approaches and Machine-Learning based approaches. Machine-Learning based approaches are then also split further into supervised, unsupervised, and semi-supervised.

Lexicon based sentiment analysis approaches are techniques that make used of a predefined list of words that have been assigned a specific association. These predefined sentiment lexicons are then used to assign a polarity value to each text document (for example, a tweet) by following a basic algorithm[5]. Predefined sentiment lexicons are often referred to as corpora or dictionaries.

Examples of popular lexicon-based sentiment analysis models include the “Valence Aware Dictionary and Sentiment Reasoner”, known simply as VADER[6]. VADER is even specifically tuned to sentiments expressed in social media, however, was not suitable for the project, as discussed in *Implementation and Testing*

Machine-Learning based sentiment analysis techniques do not use any form of pre-defined corpora. Instead, in the case of supervised machine-learning, models are trained using full text documents (tweets) that have been hand labelled into certain categories. In the case of sentiment analysis, these are often positive/negative/neutral. This was the approach I decided to take myself and is again discussed further in *Implementation and Testing.*

Semi-supervised, or hybrid approaches to sentiment analysis use a combination of lexicon, and machine-learning based techniques.

## Previous Attempts at Classifying Vaccine Sentiment

My project is not the first attempt at classifying vaccine sentiment, it is not even the first attempt at classifying vaccine sentiment on twitter specifically. In recent times, similar studies have been carried out[4][7][8]. However, these studies primarily focus on classifying data and presenting the classification results in a purely academic format, such as this report. While this provides an excellent base for that classification data to be used or extended for further study, this was not the direction I wanted to take my project in.

These previous projects and studies typically used a Semi-Supervised Machine-Learning methodology. Meaning a combination of Lexicon-Based and Supervised Machine-Learning techniques. This approach, combined with a high amount of training data lead to a high level of accuracy. In the case of one study, with 10,500 points of training data, a >85% accuracy was achieved in classification[4].

However, even with multiple studies with high levels of classification accuracy, my primary takeaway was that none of the information gathered was presented in a wide and easily accessible scope. It remained within the realms of academia, as mentioned earlier. This is where I wanted my project to be different, deciding from early on that I would present my results in a easily accessible, graphical format for anyone to be able to find and use. I discuss this further in the following *Specification* section.

# Specification

I split my project into four broad, high-level sections early on:

1. Data Collection
2. Data Formatting
3. Data Classifying
4. Data Presenting

Each of these sections also had distinct sub-sections, some of which I was aware of at the outset of the project, others I discovered as the project went on.

Due to this, the nature of the project was very exploratory for me. I was learning what was required as I was progressing through the stages. This made it very difficult to nail down a development methodology early on in the project. I discuss this further in subsection *Development Methodology*.

So, while parts of my specification were pre-planned right from the outset, as the project evolved and I learned more, the initial plans rapidly fell to the wayside. This led to me spending less time in the pre-planning specification writing stages, and more time just working. As time spent planning was often time wasted due to how quickly reality deviated from plans.

## Data Collection

The first part of any Data-Driven solution is to acquire the data. For this project, I would be using the Twitter API[9] for my data collection purposes. The project plan for this section broke down as the following:

1. Acquire Twitter Developer Account
2. Create Twitter API Application (needed for endpoints access)
3. Learn how to use the API 2.0 Endpoints
4. Use API to Collect Data.

However, as stated earlier in *Specification*, reality rarely went to the original specification, so the Data Collection part of the project went more like the following:

1. Acquire Twitter Developer Account
2. Create Twitter API Application (needed for endpoints access)
3. Escape from Twitter Spam Filter detection
4. Learn how to use API 2.0 Endpoints
5. Collect Data
6. Learn how to use API 1.1 Endpoints
7. Collect Data

The details as to why reality deviated from the original specification so dramatically are discussed thoroughly in *Implementation and Testing.*

## Data Formatting

The data formatting section was planned as the following, with no changes during implementation:

1. Data “Cleaning”
   1. Discarding extra information returned by the API that was unnecessary for the project goals.
2. Data Structure “Formatting”
   1. Splitting the tweets into individual text files, organizing them into a two-level folder structure that the Machine Learning model could use.
   2. This stage was done in conjunction with hand-labeling data for training
3. Data “Pre-Processing”[10]
   1. Remove all special characters
   2. Remove all single characters
   3. Remove single characters from start
   4. Replace multiple spaces with single spaces
   5. Removing pre-fixed “b” from loading function
   6. Converting to lowercase
   7. Lemmatization
4. Covert Text to Numbers
   1. Done using “Bad of Words” Model.
5. Finding TFIDF
   1. Term Frequency – Inverse Document Frequency

## Data Classifying

Once all the data has been adequately formatted and pre-processed, the final classifying stage is fairly simple. Only consisting of the following:

1. Train Model
2. Evaluate Model
3. Save (“Pickle”) Model
4. Use Model to Classify remaining large-scale data

A few snags were hit in this area, and are discussed in detail in *Implementation and Testing*.

## Data Presenting

The final part of the project involved Data Presenting. This was making a full front-end website that users could browse to see the results of the project. As well as to see some comparisons of sentiment data alongside other statistics. I arrived at the decision to implement this part of the project based on background research. This was what would set my project apart from previous studies into Vaccine Hesitancy – a clear, non-academic, easily accessible way to view the classified data from the project.

This was further motivated by my first meeting with my project supervisor – Prof. John Lawrence, where we discussed how the project would be split on a back-end/front-end level (see Appendix A).

This would also be used as an opportunity for me to learn and become more familiar with JavaScript, and JavaScript libraries such as React.js[11]. While I had done front end development before, it had not been with simpler technologies, so this was a chance expand my skillset. As a result, I arrived at the following (loose) plan/specification for the website.

1. Learn React.js
2. Create landing page
3. Create “Comparisons” Pages
4. Present data graphically using Graph.js[12]
5. Achieve full deployment for website.

## Development Methodology

While *Development Methodology* is not explicitly related to *Specification* it is important in my case to understand the development methodology I implemented, as it was directly responsible for the very loose style of specification I arrived at and gave myself for the purposes of this project.

The initial plan for the project, before I ever had my first meeting or read my project brief, was to adopt an Agile Methodology to development. I had in previous projects adopted Agile practices to great success, and was a fan of how Agile allowed flexibility and rapid prototyping during development.

However, after my first meeting with my project supervisor, it became apparent to me that I knew very little about the space in which my project was going to be developed. I had never worked with the Twitter API, with Machine-Learning, with Sentiment Analysis or with Natural Language Processing techniques, even my plan to use React.js was so that I could have the opportunity to learn it. Every single stage of my research and development process would be like uncovering a fog-of-war on a map – I could only see what’s in front of me as I came up to it.

As a result, although I did initially plan sprints (see Appendix B) I quickly decided it wasn’t a good investment of time to generate detailed requirements or user stories, as these requirements would quickly be overridden during development – when I actually learned what is required, rather than guessing without much knowledge.

# Design

My design choices were across a number of areas, what languages to adopt, what frameworks to use, what tools would I be able to use, what constraints would I be working with and what systems would I be using for backup and version control.

Each of these decisions was considered for each of the main areas detailed earlier in *Specification* (Data Collecting, Formatting, Classifying, and Presenting).

However, before any of that could be considered, the first and simplest decision needed to be made regarding backup, code storage, and version control.

## Backup, Code Storage & Version Control

This was the easiest design decision out of them all. While consideration was given to services such as SourceForge[13], BitBucket[14], GitLab[15] and even just locally saving all my work. The final decision was very quickly to use the industry standard: GitHub[16].

Although I had been recommended the alternatives by my friends, and even initially wanted to just work locally from my own machine. I had prior experience with GitHub and decided that even thought the project was a solo one, and did not require any code-collaboration, I would use GitHub just because it provides a form of backup – just in case something was to go wrong. (To good effect too, as I did lose all my local work during the year due to a system malfunction on my main machine that required a full re-install)

## Data Collection

For Data Collection, right from the outset I had decided to use the Twitter API. This wasn’t much of a choice, although techniques such as JavaScript Web Scrapping exist for data collection, the Twitter API offers the most control and easiest level of access to Twitter Data.

Within the Twitter API, a decision has to made between using the version 2.0 Endpoints, and the version 1.1 Endpoints.

Initially, I decided on using the 2.0 Endpoints, as the version 1.1 Endpoints were both deprecated, poorly documented and no longer maintained.

To use the 2.0 Endpoints, as recommended by the Documentation I used the Postman API Platform[17] tool. The Postman API Platform is a tool for developers to design, build, test and iterate their APIs. It allows user to make requests to APIs from a graphical user interface. This was perfect for me, as I had never used the Twitter API before and a GUI-centric way of accessing it would be the most simple.

Twitter also supplied a pre-made collection[18] that made access to the Twitter API 2.0 Endpoints from postman very simple. While this worked, due to reasons discussed in detail in *Implementation and Testing* I was forced away from using the Twitter API 2.0 Endpoints, and had to pay for access to the 1.1 Endpoints, and use those.

The version 1.1 Endpoints provided by Twitter are both deprecated, and poorly documented. This meant there was no Postman collection available to allow the API to be accessed graphically. This lead to a steep learning curve as I was forced to adopt Ubuntu (via Windows Subsystem for Linux – WSL[19]) and cURL[20] (a command line tool for transferring data) as my main technologies for Data Collection.

## Data Formatting

The next design choice to be made was what technologies would I use for the expansive amount of data formatting that was required – in fact, this was probably one of the most key parts of the project. As such, the design choices made here would be consequential.

Luckily, these choices were somewhat made for me. During my first meeting with my project supervisor. I was strongly advised to use Scikit-Learn[21] for when I eventually moved on to the Machine-Learning and classifying stages. Scikit-Learn is a Python[22] library. On top of this, my own background research showed that Python was considered the go-to language for entry-level Machine-Learning and Data Analysis. So, Python seemed like the obvious choice, if I used it now during the Data Formatting stages, I would be more familiar with it during the Data Classifying stages.

However, although Python seemed like the obvious choice, I was also going to be using JavaScript later during the Data Presenting stage, when building my front-end. The Data returned from the Twitter API 1.1 Endpoints was also returned as JSON files – JavaScript Object Notation (see Appendix C). So, I was initially tempted to use JavaScript for all the Data Formatting purposes.

Despite this, my final decision was to use Python. As I learned that the Data Formatting and Data Classifying stages would be heavily intertwined, and switching languages between them would just be extra work for no benefit.

Within Python, a number of packages were used to aid in the Data Formatting. These include:

* NumPy[23], a library that adds support for large, multi-dimensional matrices and arrays, as well as a large collection of high-level mathematical functions to operate on those arrays.
* re[24], a base python module that provides regular expression matching operations
* Natural Language Toolkit (NLTK)[25], a suite of libraries and programs for natural language processing for English.
* pickle[26], a base python module for serializing and de-serializing Python object structure – essentially granting the ability to “save” trained Machine-Learning models to a file.

## Data Classifying

As stated in the previous subsection *Data Formatting*, the advice given during the initial project meeting (see Appendix A) was to use Scikit-Learn, and by extension, Python, for the Machine-Learning aspects of the project. My own research also showed that Python was by far and away the most popular language for entry-level Machine-Learning.

However, during my research, I also discovered alternatives to Scikit-Learn. Primarily in the form of TensorFlow[27], TensorFlow being a free and open-source library developed by Google that is available in Python. It can be used for a wide range of tasks but has particular focus on training and inference of deep neural networks and machine-learning.

TensorFlow also had a Sentiment-Analysis tutorial available in their own documentation, whilst Scikit-Learn did not. However, upon discussing with my project supervisor and being provided a number of resources (see Appendix D) for Sentiment-Analysis using Scikit-Learn, I did finally settle on that.

For the purposes of the model itself, I settled on using a Random Forest Classifier algorithm. While the Random Forest Classifier algorithm creates models that are slow to train and require higher processing power. This wasn’t a large problem on my end, as the model would only need to be trained once, before being pickled. Random Forest Classifiers are also good at working on large datasets, my project was designed to be easily extendable, so I would need a model that could cater to a growing dataset, which the Random Forest Classifier was suited for. The following reasons, as well as Random Forest Classifiers being recommended by various tutorials and articles, led to that algorithm being the one I very quickly settled on for training and classification purposes.

## Data Presenting

For the Data Presenting portion of my project, I was certain from the start that I would be making a website as my front end. I also knew that I wanted to use this as an opportunity to learn and work in JavaScript. As a result, although a number of JavaScript frameworks such as Angular.js[28] and Vue.js[29] are considered “better” or “easier”[30], I decided on using React.js[11].

This was because regardless of ease of use, or being told other frameworks were “better”, React.js is by far the most popular web framework in use today[31]. Which means it is a valuable technology to be proficient in as a developer.

As a result, it didn’t get given much further considering, I would be using React.js for my website/front-end. This was not the only technology used however, several packages were used alongside React.js to develop the final website, including:

1. Bootstrap[32], a free and open-source CSS framework for responsive front-end web development
2. Chart.js[12], a free and open-source JavaScript library for data visualization
3. React-Router[33], a library for routing in React.
4. React-Twitter-Embed[34], a library for easily embedding tweets, allowing you to show tweets without violating Twitters style guide/policy.
5. React GitHub Pages[35], scripts that allow deployment of React Apps to GitHub Pages. Allowing a fully-deployed website.
6. React-Select[36], a flexible input select control library for React.js

# Implementation and Testing

You should describe the important aspects of implementation, testing, and debugging that you went through to produce your system. You can structure this in different ways, depending on the development methodology adopted and the needs of your project. You may wish to start with a review and overview of the main features to be implemented and a general, architectural overview of the system. You may then wish to walk through the major features, components, or sub-systems that were created, one after another. These could be sub-sections in your report, e.g., Feature X, Feature Y, etc. Or you may wish to present a time-based review of the implementation process, according to the stages you went through in your project plan. Indeed, if you have adopted an Agile approach, you may wish to structure your discussion around the various Sprints that were undertaken. In your discussion, highlight any important features that were implemented, any major problems that were encountered, and the workarounds that you produced. Your aim is to convince the reader that you are technically competent and that you are capable of problem solving and adapting to needs of the project. The amount / extent of technical contribution is also being assessed and the extent to which you have been able offer original ideas of your own. Regarding the amount of technical contribution. For example, a basic website, with a few, static pages is likely to be rated somewhat poorly. Instead, one would expect dynamic content, a database, more complex code and problems being solved, additional considerations for accessibility, usability, security, etc.

Regarding the implementation section. You may wish to illustrate your discussion with diagrams, or code snippets, that offer additional insights into your work or achievements. You may wish to emphasize user-centred processes, where applicable, and how the system evolved during implementation. For technically oriented projects, it is understood that you may wish to focus more on the performance, accuracy, reliability, or precision in your outcomes, including benchmarking against the work of others. For an additional layer of sophistication, any project can consider additional non-functional aspects of the system which are applicable, e.g., security, scalability, performance, usability, accessibility.

Later in your report, there is a related section: Description of Final Product. This later section is focused around providing a summary overview of your finished product. In contrast, the implementation section focuses on the stages that you went through to achieve and deliver it. There may be some areas of overlap, e.g., when you discuss the implementation of a particular user interface component, and you wish to use a screenshot to highlight the implementation choices made. Meanwhile, it turns out that a similar screenshot is necessary later in the Description of Final Product section, where you are simply presenting what the key aspect of the interface looks like. That is OK. There is just a difference of emphasis here.

For additional sophistication in your implementation, you should consider the use of software testing techniques, e.g., unit testing or similar. If so, the markers would need to see evidence of their use, e.g., in your source code or similar. In addition, you could consider traceability back to your original requirements, and verification or validation that they have been achieved.

As noted above, you may wish to include snippets of code in your report, to accompany your discussion of the implementation. Commonly, these may be included as screenshots of the relevant portions of code. It is best to keep these focused on specific areas of the code, e.g., it may be a specific method or a section of a method. For example, we are developing a web-based system which has a sequence of code for iterating through groups of product items. There is perhaps some reason why this code is noteworthy, e.g., it illustrates a novel approach or solves a tricky problem or is just something you are pleased with. Having discussed the feature, we wish to show a code snippet too. An example of this is below, e.g., please see Figure 2. Code Snippet. Iterating product options. below which illustrates the routine that was implemented to address this challenge. In the code snippet, you will see how the product items are iterated to complete the relevant basket page for the user.

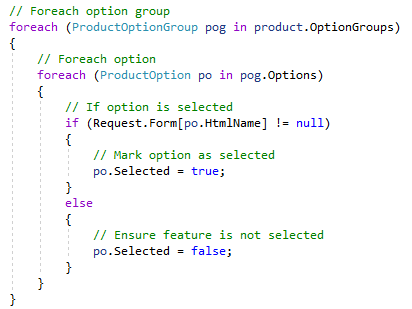


Figure 2. Code Snippet. Iterating product options.

Sometimes, you find that you need to include a larger image in your report that doesn’t fit easily into the two-column layout. In that case, you may be able to use Section Breaks in the document, to temporarily switch to a single-column layout and back again. For example, at this point in the report I have inserted a Continuous Section break immediately below this paragraph. Inside the new section, I have modified the layout to be single-column instead of double-column. I have included an example of a figure that spans the entire page. In this case, the figure is shown on the next page, because it doesn’t fit here. This has left a little but of a gap here on this page. This can be a common problem. Sometimes you can position things in such a way that the gaps are minimised. It is also worth noting that you don’t always need to place a figure immediately beside the text that it refers to. You could place a figure on the next page instead for example, whilst you continue to refer to it here, and afterwards. That way, you can fill up as much of the text here as you can too, without the need to have the figure placed in between. It is also possible to have several figures placed together on one page, and these can be referred to from relevant locations in your text. If you have large images that are impossible to fit into the report without being legible, you can include images in your appendices too.

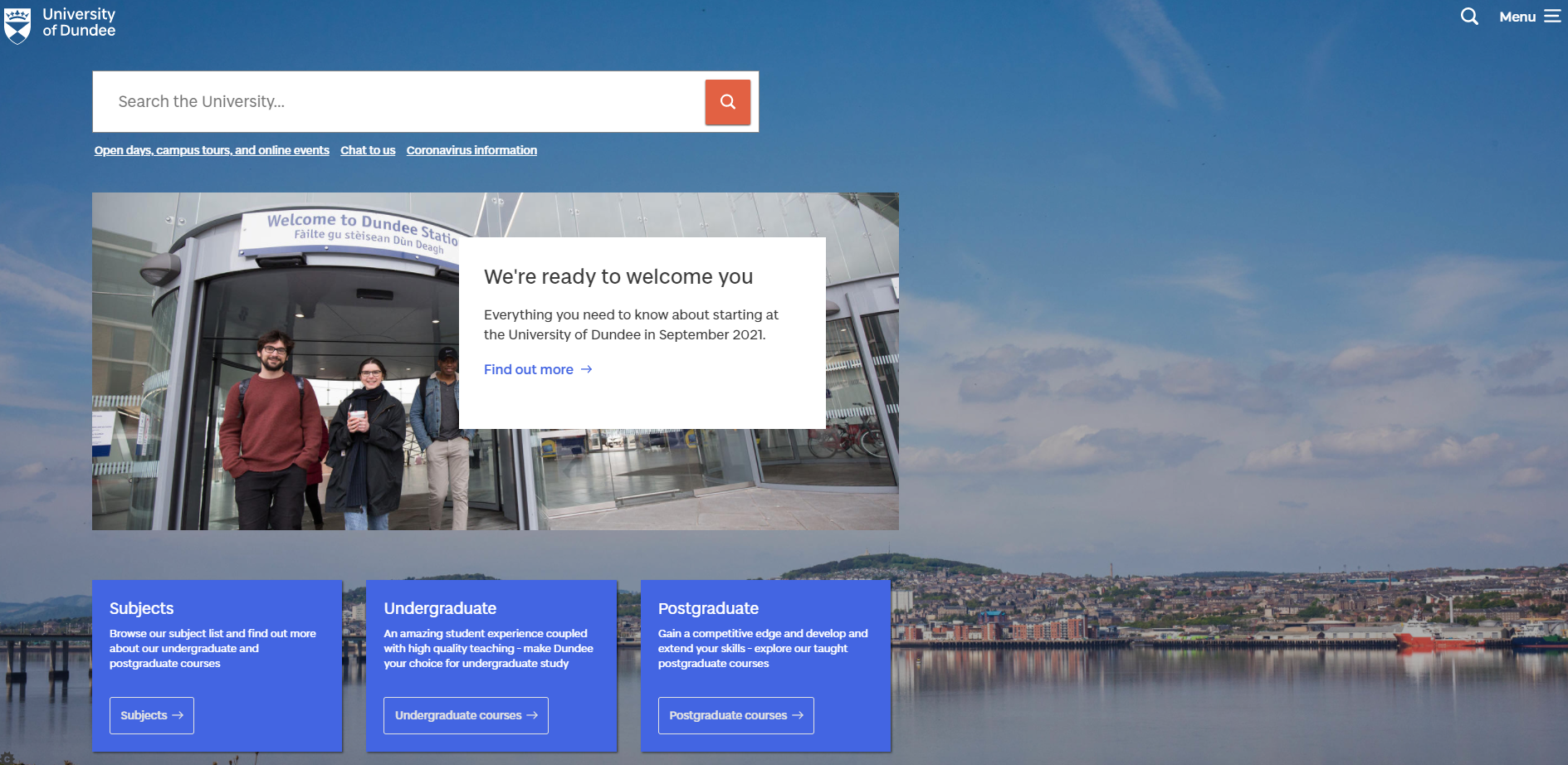


Figure 3. Random Image. In this case, a screenshot of the UoD website.

Now that I have finished what I want to do in this section, I have placed another continuous section break immediately after this sentence, which ends this single-column section and takes me back to the double-column layout afterwards.

# Evaluation / Testing

You must evaluate your system. This will be done in different ways depending on the project. For example, if you are developing a web application or app, it is common to do user testing and you may wish to seek feedback and comments from end users through interviews or questionnaires. If your project has a technical, non-user-based focus, your testing may focus more on benchmarking, comparing different algorithms or parameters, measuring performance or precision, etc. For any type of project, you can consider additional criteria where applicable, e.g., security, performance, accessibility, and computational efficiency. In the case of Cloud-based applications or services, one could also consider the cost implications (e.g., 'x' pence per query) and whether this has influenced the design and testing of the application.

Regardless of the project, you must describe the evaluation or testing of your system in your report, and this must include the following: a presentation of any relevant data; a discussion and analysis of the data; a discussion of the significant results and outcomes you have found. Ideally, you should consider any limitations in your evaluation and the extent to which your outcomes can be generalized to a wider ‘population’, or not.

Consider what you want to evaluate or test, and how you will achieve it. Develop the necessary evaluation plans / materials / methods, and make sure these are described in your report. Be mindful of ETHICS where required and make sure that the relevant Ethics documents are utilised, and it is clear where and how ethics has been adopted in your evaluation. Describe how your tests or evaluations were conducted. You can include the materials you have used in your appendices, e.g., test plans, evaluation checklists or tasks, copies of questionnaires used. Present and discuss the data in your report. You can include copies of the data in the Appendices too. Discuss the main outcome or findings from your evaluation / testing.

## Using subsections in your report

Remember, you can use subsections throughout your report to structure the content. This is often desirable to break up large expanses of text and to aid the reader too. Examples of subsections here, in the Evaluation section could be general such as Methodology or Results, and/or specific such as Usability, Performance, etc. Please remember to use the built-in styles for this that Word provides (Heading 1, Heading 2, etc.). This is necessary to ensure that your document is accessible.

### Here is a sub-subsection

You can use additional layers of hierarchy to progressively structure the content. In this case, if there was a subsection named Methodology, perhaps it could contain sub-subsections such as Participants, Tasks, Ethics, etc.

#### Be mindful of taking the structure too far

Whilst you can use as many hierarchies as you wish in structuring your content, there is usually a limit to what is useful in terms of readability. Aim to go no more than 3 layers deep in the hierarchy, if possible.

# Description of the final product

You should provide a clear description of what the final product looks like and what it does. You do not have to explore every minute detail of the system, you should attempt to convey the key, major areas of functionality. In some ways, you could consider this section to be a cut-down version of a user manual. Even in systems where there is no user interface, there may still be some general aspects that you can mention. However, if it is the case that this section of the report is just not relevant to your project, please just state that or omit this section.

When you are writing your report, you may find that the content of this section could overlap with earlier content in the report too, such as the implementation section. We want to avoid repetition in the report. At the same time, a degree of overlap is OK, bearing in mind that it is other people who are reading your report and they may benefit from a reminder, and a focused overview of what the final product looks like. As noted earlier, this section provides an overview of your finished product whereas earlier sections such as the implementation focus more on how you got to that point, i.e., the stages you went through, the decisions you made, and the problems you had to solve along the way.

# Appraisal

Provide a critical appraisal of the project. The question that I would pose to you here is as follows: if you were doing the whole project again, what would you do differently, what would you do the same, what advice would you give to others if they were doing the same project? Here you should reflect on the entirety of your project including your choice of technologies, your implementation decisions, and the project plan. With the benefit of hindsight, what are the lessons learned during the project and the evaluation of the final product and the process of its production (including a review of the plan and any deviations from it). Also consider what have been the most useful learning aspects for you.

NOTE: the appraisal section could potentially occur after the Summary and Conclusions below, or even as a sub-section within the Summary and Conclusions. See what works best for you and your advisor.

# Summary and Conclusions

Summarise the main points of what your project was and what the report has provided. Provide a summary. Describe the conclusions and outcomes that you have found.

# Future Work

What recommendations do you have for future work? Are there more features that need to be included? More testing? More evaluations? Are there follow-on projects or ideas that could be explored? Do you plan to do any more with the project yourself? Please discuss this here.

NOTE: this section could possibly appear as a sub-section within the preceding Summary and Conclusions.

## Acknowledgments

You can provide acknowledgements here to anyone who has been helpful in your project, or beyond. In some cases, the licensing of certain software products you have used may require you to acknowledge them here, e.g., in return for free use.

# References

[1] Durbach, N. They might as well brand us: Working class resistance to compulsory vaccination in Victorian England. The Society for the Social History of Medicine. 2000;13:45-62.

[2] Article on Unvaccinated Patients: https://northeastlondonccg.nhs.uk/news/almost-90-of-patients-admitted-to-intensive-care-units-innorth-east-london-are-not-fully-vaccinated/

[3] Cambridge Analytica Swinging Elections: https://www.theguardian.com/news/2018/may/06/cambridge-analytica-how-turn-clicks-into-voteschristopher-wylie

[4] Piedrahita-Valdés H, Piedrahita-Castillo D, Bermejo-Higuera J, et al. Vaccine Hesitancy on Social

Media: Sentiment Analysis from June 2011 to April 2019. Vaccines (Basel). 2021;9(1):28. Published 2021

Jan 7. doi:10.3390/vaccines9010028

[5] Almatarneh, Sattam, and Pablo Gamallo. "A lexicon based method to search for extreme opinions." PloS one 13.5 (2018): e0197816.

[6] VADER: https://github.com/cjhutto/vaderSentiment

[7] Neha Puri , Eric A. Coomes , Hourmazd Haghbayan & Keith Gunaratne (2020) Social media and

vaccine hesitancy: new updates for the era of COVID-19 and globalized infectious diseases, Human

Vaccines & Immunotherapeutics, 16:11, 2586-2593, DOI: 10.1080/21645515.2020.1780846

[8] Muric G, Wu Y, Ferrara E. COVID-19 Vaccine Hesitancy on Social Media: Building a Public Twitter Data

Set of Antivaccine Content, Vaccine Misinformation, and Conspiracies. JMIR Public Health Surveill. 2021

Nov 17;7(11):e30642. doi: 10.2196/30642. PMID: 34653016; PMCID: PMC8694238

[9] Twitter API: Twitter API Documentation | Docs | Twitter Developer Platform

[10] Stack Abuse: Text Classification with Python and Scikit-Learn: https://stackabuse.com/text-classification-with-python-and-scikit-learn/

[11] React.js, A JavaScript Library for building user interfaces: https://reactjs.org/

[12] Chart.js, Simple yet flexible JavaScript charting for designers and developers: https://www.chartjs.org/

[13] SourceForge, Compare, Download & Develop Open Source & Business Software: https://sourceforge.net/

[14] Bitbucket, The Git solution for professional teams: https://bitbucket.org

[15] Git Lab, iterate faster, innovate together: https://gitlab.com/

[16] GitHub, Where the world builds software: (https://github.com/)

[17] Postman API Platform: https://www.postman.com/

[18] Postman API Twitter 2.0 Endpoints Collection: https://github.com/twitterdev/postman-twitter-api

[19] Microsoft Windows Subsystem for Linux (WSL): https://docs.microsoft.com/en-us/windows/wsl/about

[20] cURL, a command line library for transferring data: https://curl.se/

[21] Scikit-Learn: Machine Learning in Python: https://scikit-learn.org/

[22] Python Programming Language: https://www.python.org/

[23] NumPy, https://numpy.org/

[24] re, https://docs.python.org/3/library/re.html

[25] Natural Language Toolkit (NLTK), https://www.nltk.org/

[26] pickle, https://docs.python.org/3/library/pickle.html

[27] TensorFlow, an end-to-end open-source platform for machine learning: <https://www.tensorflow.org/>

[28] Angular.js: https://angular.io/

[29] Vue.js: https://vuejs.org/

[30] Frameworks Comparison Video, Fireship: https://www.youtube.com/watch?v=cuHDQhDhvPE

[31] Most Popular Frameworks, according to Statista: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/

[32] Bootstrap, https://getbootstrap.com/

[33] React-Router: https://reactrouter.com/

[34] React-Twitter-Embed:

https://www.npmjs.com/package/react-twitter-embed

[35] React-GitHub-Pages:

https://github.com/gitname/react-gh-pages

[36] React-Select: https://react-select.com

# Appendices

* Appendix A: Hand-Written meeting log for first meeting with John Lawrence.
* Appendix B: Gantt Chart for Semester 2
* Appendix C: Example of JSON Returned by API
* Appendix D: Email with Resources for Scikit-Learn provided by supervisor (John Lawrence)