

National College of Ireland  
BSc in Computing  
2017/2018

Dylan Kilkenny  
x14472008  
x14472008@student.ncirl.ie

## Cryptocurrency Analysis

Technical Report



# Table of Contents

Executive Summary	5
1 Introduction	7
1.1 Background	7
1.2 Aims	8
1.3 Technologies	9
1.4 Structure	11
2 System	12
2.1 Requirements	12
2.1.1 Functional requirements	14
2.1.1.1 Use Case Diagram	15
2.1.1.2 Requirement 1: Gather Data	15
Description & Priority	15
Use Case	15
2.1.1.3 Requirement 2: Scrub and Standardise data	17
Description & Priority	17
Use Case	17
2.1.1.4 Requirement 3: Sentiment Analysis	19
Description & Priority	19
Use Case	19
2.1.1.5 Requirement 4: Data Analysis	21
Description & Priority	21
Use Case	21
2.1.1.6 Requirement 5: Login	23
Description & Priority	23
Use Case	23
2.1.1.7 Requirement 6: Search	24
Description & Priority	24
Use Case	25
2.1.1.8 Requirement 7: View top N cryptocurrencies	26
Description & Priority	26
Use Case	26
2.1.1.9 Requirement 8: Specific Cryptocurrency data	28
Description & Priority	28

Use Case	28
2.1.2 Non-Functional Requirements	30
2.1.2.1 Availability requirement	30
2.1.2.2 Recover requirement	30
2.1.2.3 Reliability requirement	30
2.1.2.4 Environmental requirement	31
2.1.2.5 Extensibility requirement	31
2.1.3 Data requirements	31
2.2 Design and Architecture	33
2.2.1 Database Design	34
2.2.1.1 Collections	34
2.3 GUI	38
2.3.1 Cryptocurrency Overview	38
2.3.2 Individual cryptocurrency section	39
2.3.2.1 Main stats and price/volume/sentiment chart	39
2.3.2.2 Associated currencies and social stats	40
2.3.2.3 Most active users, words and bigrams by date	41
2.3.3 Search	42
2.3.4 Login	43
2.3.4.1 User logged in	44
2.3.4.2 User logged out	44
2.4 Implementation	45
2.4.1 API Server	45
2.4.2 Analysis System	46
2.4.2.1 Environment	46
2.4.2.2 Twitter Stream	47
2.4.2.3 Reddit Stream and Historical data	50
2.4.2.4 Reddit Historical Data	51
2.4.2.5 RedditDB and TwitterDB	53
2.4.2.6 Reddit and Twitter Analyser	55
2.4.3 Price movement prediction	58
2.4.3.1 Random Forest	59
2.4.3.2 K-nearest neighbour	61
2.4.3.3 Logistic Regression	63

2.5	Testing	64
2.5.1	Unit Tests	64
2.5.2	Integration Tests	66
2.5.3	Jest	68
2.5.4	Black Box testing	69
2.6	Customer testing	74
	One of the last stages of a software development lifecycle is user acceptance testing (UAT). UAT involves actual users of the software testing it, in order to make sure it can handle real world tasks and fulfils all the requirements from an end user's perspective. The web app url and a survey was sent around to a number of friends in order to get their opinion on the web application and the information it provides. The survey consisted of a total of 6 questions and got 7 responses.	75
3	Conclusions and Future Work	79
4	References	80
5	Appendix	81
5.1	Project Proposal	81
5.1.1	Objectives	81
5.2	Project Plan	86
5.3	Monthly Journals	87
5.3.1	September	87
	My Achievements	87
	My Reflection	87
	Supervisor Meetings	87
5.3.2	October	87
	My Achievements	88
	My Reflection	88
	Intended Changes	88
	Supervisor Meetings	88
5.3.3	November	88
	My Achievements	89
	My Reflection	89
	Intended Changes	89
	Supervisor Meetings	89
5.3.4	January	89
	My Achievements	89

My Reflection	89
Intended Changes	90
Supervisor Meetings	90
5.3.5    February	90
My Achievements	90
My Reflection	90
Intended Changes	90
Supervisor Meetings	91
5.3.6    March	91
My Achievements	91
My Reflection	91
Intended Changes	91
Supervisor Meetings	91

## **Executive Summary**

The purpose of this project is to perform an analysis on the comments by users on cryptocurrency discussion platforms and the prices of the mentioned cryptocurrencies. Although the blockchain has been around close to 10 years, cryptocurrencies are still a relatively young market. Bitcoin which has the first mover advantage has always been a dominating force within the sphere, but within the last year the market has seen a huge increase in interest. One year

ago, the entire market capitalization of all cryptocurrencies was \$13 billion and bitcoin held a dominance of 88% with a market cap of \$11.5 billion. Compare that to today and all cryptocurrencies have a market cap of \$310 billion with bitcoin only having a 54% dominance. The entire sphere has seen growth of more than 2284% in less than a year and bitcoin is beginning to lose a lot of its market share to newer more promising currencies. Unlike traditional stocks, cryptocurrencies have no tangible value attached and prices fluctuate with perceived value. Cryptocurrency trading relies on speculation more so than fundamental analysis and as a result most of the price speculation takes place online. Due to the abundance of data regarding speculation around prices on various cryptocurrency discussion platforms, I hope to find a correlation between the sentiment of the posts and the prices of the mentioned cryptocurrencies. This project will follow the KDD methodology.

# 1 Introduction

## 1.1 Background

I first heard about bitcoin in 2012 and made my first purchase in 2013. At the time, I had very little understanding of how the underlying technology that powered bitcoin, the blockchain, worked and as far as I was aware bitcoin was the only cryptocurrency. Apart from checking the bitcoin price occasionally I had not looked into the cryptocurrency space much further for a few years.

At the beginning of 2017 I came across an article detailing a new currency called ethereum that promised to dethrone bitcoin. Unlike bitcoin, ethereum isn't just another digital currency, ethereum is a decentralised platform that runs applications on top of the blockchain, opening up the blockchain space to an array of new use cases. One of the most important features of the ethereum blockchain is the ability to create tokens. These tokens are treated the same as digital currency and can be used in the decentralised applications they are associated with as well as traded for other digital currencies. With the discovery of ethereum I began to look into the cryptocurrency space a little further and to my pleasant surprise there was more than just bitcoin and ethereum. Before I knew it, I was a frequent visitor of various cryptocurrency subreddits and forums. Unlike traditional stocks, cryptocurrencies have no tangible value attached and prices fluctuate with perceived value. Cryptocurrency trading relies on speculation more so than fundamental analysis and as a result most of the price speculation regarding cryptocurrencies takes place online. Around May 2017 the cryptocurrency market seen an explosion in interest. As a result, thousands of new users came flooding into the cryptocurrency forums searching for the next best coin to invest in. Most of the cryptocurrencies or tokens being recommended had great potential for making a real-world impact but at the time had no product, or development was a few years away from completion. Despite a lot of the cryptocurrencies having no product or being half finished they began to increase in price anyway. With the influx of new users came a lot of inexperienced traders

putting their money where ever the consensus was. The cryptocurrency market is still quite young so there is a lot of volatility. Regulations are scarce and price manipulation from pump and dumps or traders with a large bankroll is a lot easier than traditional stock markets. Compared to the stock market it does not take a lot of buying or selling power to change the price and with the arrival of novice traders that invest in cryptocurrencies solely because they see them recommended a lot, I believe it may be possible to find a correlation between the sentiment of posts on the various platforms and the prices of the mentioned cryptocurrencies.

## **1.2 Aims**

**Aim 1:** The overall aim of this project is to perform sentiment analysis on the comments of various cryptocurrency discussion platforms. Sentiment analysis is the computational task of determining what opinion a user is expressing in text through NLP. For example, one of the simplest forms of sentiment analysis is to classify words as “positive”, “negative” or “neutral” and then average the values of each word to classify the text. This analysis technique is very useful for gaining an overview of the public's opinion on certain topics. By measuring the sentiment of comments on the most popular cryptocurrency platforms this project hopes to find a correlation between the sentiment of the discussions and the price movements of cryptocurrencies.

**Aim 2:** Although sentiment analysis is great for discovering the overall opinion of a piece of text, it is not perfect and as with any analysis there will be errors in the result. Topic modelling will combat this by adding another layer of classification, allowing us to summarise, understand and organise the dataset. Topic modelling can be described as a method for finding a group of words from within a dataset which best represents the information within. By applying topic modelling to the comments from the various platforms the system will gain useful insight into how topics on the platform are evolving over time.



**Aim 3:** The third aim of this project is to provide a web application for users to access real time social media statistics for the top 200 cryptocurrencies. The web app will provide an overview of the cryptocurrencies similar to current cryptocurrency price aggregation sites, with the added social media mentions and sentiment stats. Users will be able to view more in depth statistics for individual cryptocurrencies, such as historical prices, sentiment and social buzz surrounding the currency. As well as an overview of the most mentioned topics when discussing a given cryptocurrency.

**Aim 4:** The final aim of this project is to utilize the gathered data to create classification models in order to predict the price movements of cryptocurrencies. Social media sentiment and number of mentions for a given cryptocurrency, will be used along with the price and trading volume, in order to create KNN, SVM and Random Forrest classification models. The goal is to be able to determine whether the price will go up or down the following day, based on the current day's data.

### ***1.3 Technologies***

#### **Python 3.6**

For this project, I will be using python to run the entire analysis system. Python is an object oriented, high level language with dynamic typing and binding, making it very useful for rapid development. Firstly, it will be utilised to scrape posts and comments from the various platforms. With regards to twitter.com, coinmarketcap.com and reddit.com I will be getting most, if not all the data, through their API's. Python has a built-in requests module which makes the process of sending GET requests to the API's all that easier. Coinmarketcap API is quite simple and with few endpoints, so I will be querying their API directly with the requests library. Twitter and reddit have quite a lot of resources on their API and can often be tricky to navigate. Secondly, python will be used to transform,

arrange and analyse the data using packages such as *pandas* and *numpy* before adding it to the mongodb database with *pymongo*.

### **BeautifulSoup (Python Third-party Library)**

Coinmarketcap.com API does not provide all the useful information available on their site. Info such as historical prices and social media accounts associated with a given cryptocurrency, are available on the website but not through the API. BeautifulSoup, a web scraping module, with the help of the requests module, will allow me to directly parse this data from coinmarketcap.com html files.

### **Python API wrappers**

To simplify the development approach, I will be using two open source python wrappers found on GitHub for the reddit and twitter API's. For reddit I will be using *PRAW* (Python Reddit API Wrapper), and for twitter I will be using *tweepy*. Both these wrappers will help speed up the development process.

### **R**

R language will be used to perform the data scrubbing, classification and analysis. R offers a wide variety of statistical and graphical techniques and is highly extensible.

### **RStudio**

RStudio is an open source IDE designed for the R language. RStudio will serve as the main development environment for creating the classification models.

### **MongoDB**

MongoDB is a free and open source document-oriented database and uses JSON like documents with schemas. MongoDB will serve the purpose of storing the cryptocurrency statistics.

### **NodeJS**

NodeJS is an open source JavaScript run-time environment for executing JavaScript code server-side. The web server for the web application will be built using the node framework expressJS. Express is a flexible node.js web framework and provides a powerful set of features for web and mobile apps.

## **ReactJS**

React is a JavaScript library for building user interfaces. React allows us to create reusable UI components which can speed up the development process. This project will use React to develop a fast, scalable and simple web application.

## **CSV**

CSV is a text file which separates the values within by commas. This allows data to be saved in a table structured format. CSV files are smaller in size and are easier to generate and parse. They also require less load and setup than a traditional database. All comments, posts, prices along with any other metadata extracted will be saved in CSV format.

## **1.4 Structure**

The structure of this project is broken up into 3 sections:

- **Requirements:** This section details the functional and non-functional requirements of the system along with the use case diagrams.
- **Design and Architecture:** This section contains a diagram of the system architecture from a high-level view and information on the flow of the system.
- **GUI:** This section contains screenshots of the web application mock ups.

## **1.5 Definitions, Acronyms and Abbreviations**

Coin	Cryptocurrency or token
Token	Cryptocurrencies specific to applications built on top of the

	blockchain
KDD	knowledge discovery in databases
R	R programming language
CSV	Comma separated values file
BeautifulSoup	Python third party library for scraping web pages
Altcoin	Cryptocurrencies other than bitcoin
NLP	Natural Language Processing
TA	Technical Analysis
RSI	Relative Strength Index

## 2 System

### 2.1 Requirements

The following requirements will adhere to the KDD methodology. KDD stands for knowledge discovery in databases and refers to process of finding knowledge in data and is essential for a data analysis project. The KDD life cycle is explained below:

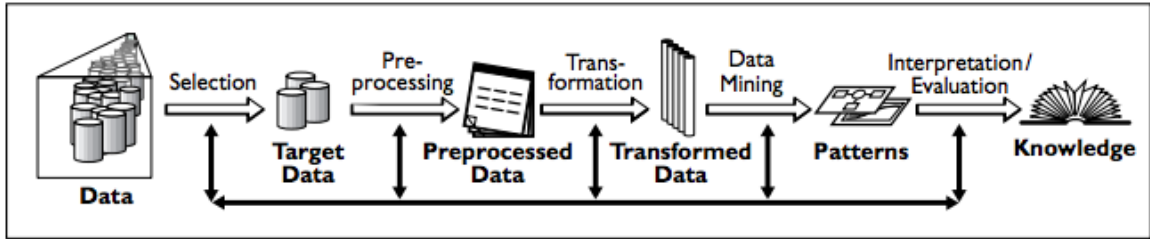


Figure 1: KDD Methodology

### **Selection**

This step involves identifying a suitable dataset for the project. This project will be focused on comments and posts from reddit.com and twitter.com. Twitter and reddit API's offer many endpoints, most of which are not relevant to the scope of this project so this step will also include choosing relevant information.

### **Pre-processing**

Pre-processing step cleans and scrubs the dataset removing any unwanted punctuation, whitespace and special characters that may interfere with the analysis. This step is important as the dataset this project is using is comments and posts from online platforms they are most certainly going to contain emoji's and other special characters that may affect the result. As this project is gathering data from three different sources some of the fields such as date may be represented differently, the pre-processing step will also standardise the datasets into a consistent format.

### **Transformation**

This step within the KDD involves generating better data from our cleaned dataset. Using transformation methods or dimensionality reduction it will reduce the number of variables under consideration or to find invariant representations for the data. This step will also classify the data into either positive neutral or negative sentiment by utilising sentiment analysis techniques.

### **Data Mining**

Data mining is one of the most vital within the project as here is where we will apply algorithms to the dataset to find trends and relevant information. During this

step both cryptocurrency historical prices and comments will be analysed looking for similarities or trends between the sentiment of the comments and price movements of the mentioned cryptocurrencies. Machine learning and statistical algorithms will be utilised within this step.

### **Interpretation**

The last step of the KDD methodology involves interpreting the results of the mined data and documenting the conclusions. The results will also be displayed using visualisations to give users a better understanding of the findings.

#### **2.1.1 Functional requirements**

The functional requirements of the system that will be required to complete the project:

1. The system will gather data and save to in CSV format.
2. The system will scrub and standardise the data.
3. The system will classify the data using text analysis.
4. The system will analyse the data.
5. The user will be able to login with google account.
6. The user will be able to search for a cryptocurrency
7. The user will be able to view data for a specific cryptocurrency.
8. The user will be able to view top N cryptocurrencies.

### 2.1.1.1 Use Case Diagram

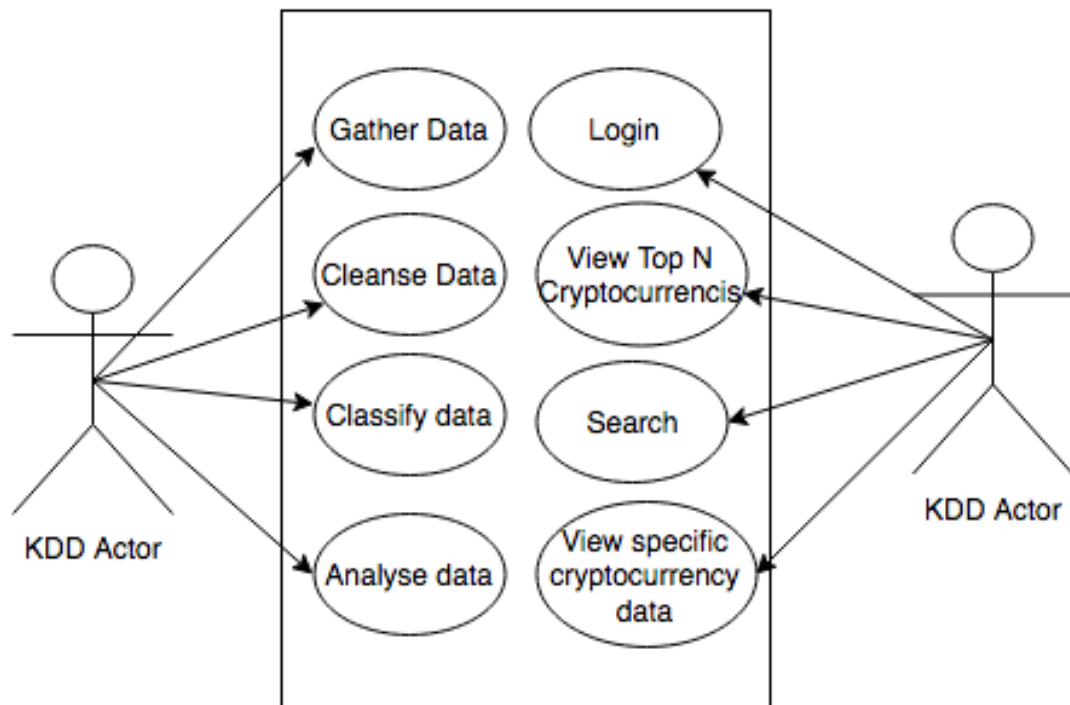


Figure 2: Use Case Diagram

### 2.1.1.2 Requirement 1: Gather Data

#### Description & Priority

This requirement is the most essential for the system. This is the first step and one of the most important as data is required in order for an analysis to be made.

#### Use Case

#### Scope

Identify which information from the platforms is relevant for the purpose of this analysis and gather the relevant data.

#### Description

This use case describes the gathering of data from various cryptocurrency discussion platforms by means of scraping or pulling from API's.

### Use Case Diagram

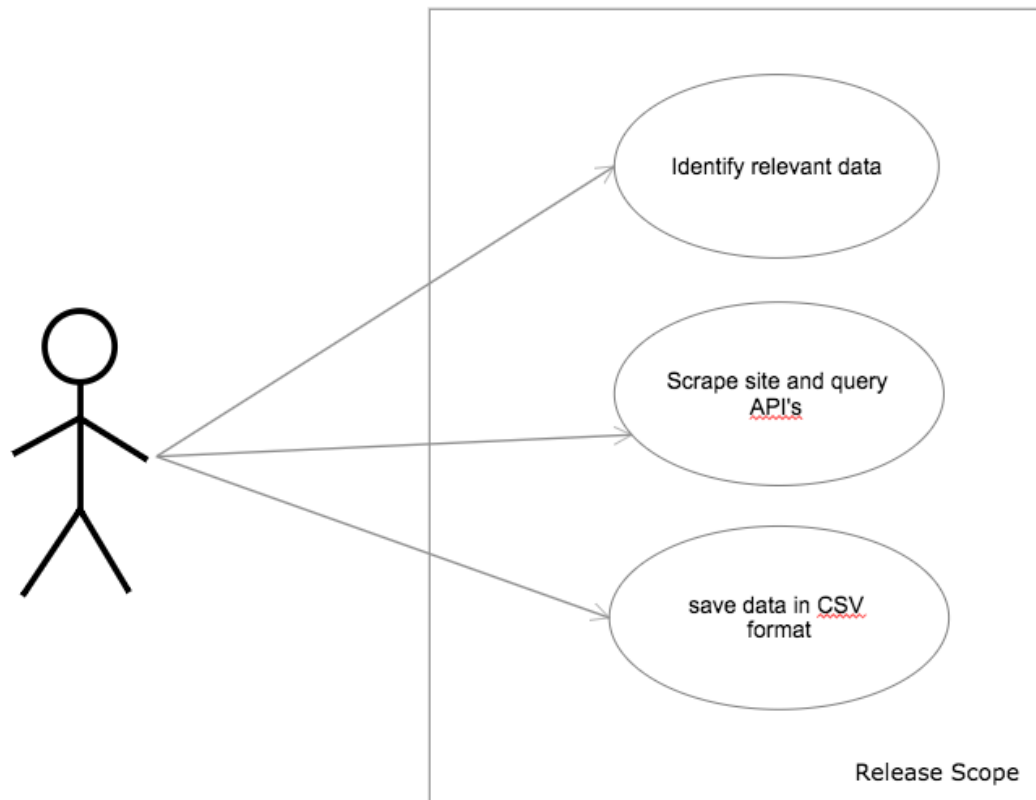


Figure 3: Gather Data

### Flow Description

### Precondition

The system has no data

### Activation

This use case starts when an <Actor> runs the python scripts which will either pull data from the API's or scrape data from the sites with no API.



**Main flow**

1. The <Actor> identifies relevant data
2. The <Actor> runs the script
3. The script pulls the data
4. The data is saved to a CSV file

**Post condition**

The system imports the file to RStudio

**2.1.1.3 Requirement 2: Scrub and Standardise data****Description & Priority**

The data will be scrubbed and standardised into a consistent format by removing punctuation and whitespace. This step is needed to ensure our analysis is as accurate as possible by removing any unwanted data.

**Use Case****Scope**

The scope of this use case is to clean the comments by removing any unwanted characters and standardise the data into a common format

**Description**

This use case describes the scrubbing and standardisation of the comments

**Use Case Diagram**

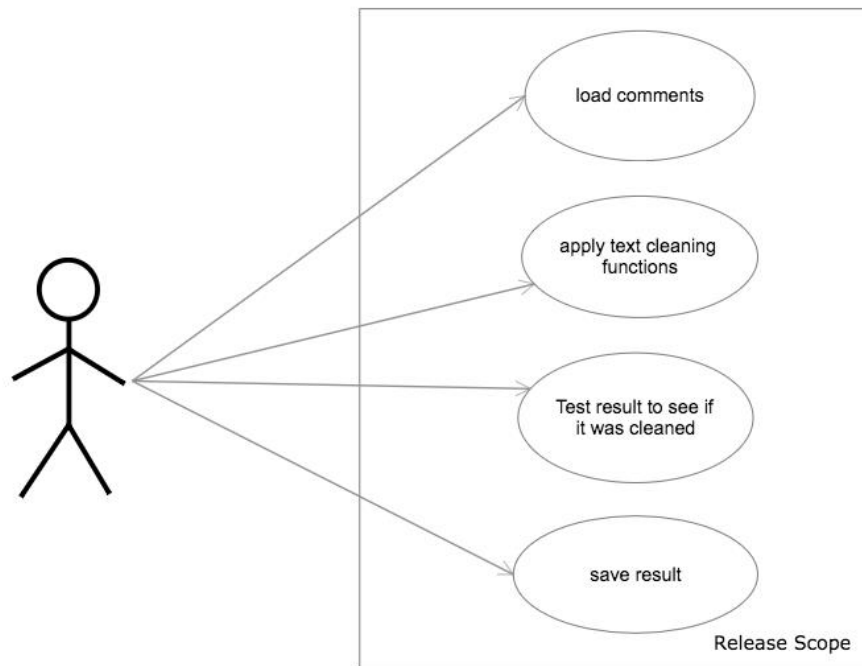


Figure 4: Scrub and Standardise

### Flow Description

### Precondition

The comments contain unneeded whitespace and punctuations

### Activation

This use case starts when an <Actor> imports the dataset to R

### Main flow

1. The <Actor> loads the dataset into RStudio.
2. The <Actor> applies data scrubbing and standardisation functions.
3. The <Actor> checks the result to see if it was scrubbed correctly.
4. The comments are scrubbed and standardised.
5. The result is saved in CSV format.

**Post condition**

The comments are ready for analysis.

**2.1.1.4 Requirement 3: Sentiment Analysis****Description & Priority**

The comments will be classified using sentiment analysis. This requirement is necessary as the focus of the project is on sentiment analysis of the data.

**Use Case****Scope**

The scope of this use case is to categorise the comments into being either positive, negative or neutral.

**Description**

This use case describes the process of performing sentiment analysis on the comments.

## Use Case Diagram

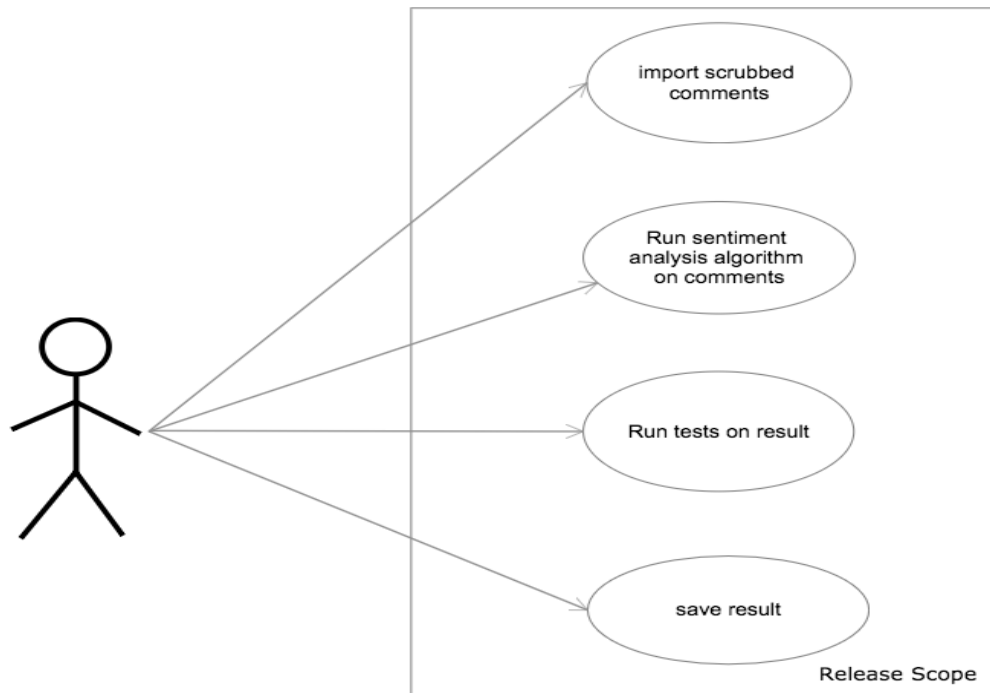


Figure 5: Sentiment Analysis

### Flow Description

#### Precondition

The comments have been scrubbed of any unwanted data that would interfere in the analysis process

#### Activation

This use case starts when an <Actor> imports the scrubbed data to RStudio

#### Main flow

The <Actor> loads the scrubbed data into R.

The <Actor> runs sentiment analysis algorithm on the comments.

The <Actor> reviews the results.

The <Actor> save the results.

**Post condition**

The system is ready to analyse the comments and prices.

**2.1.1.5 Requirement 4: Data Analysis****Description & Priority**

This requirement will make use of statistical algorithms and machine learning to find a correlation between sentiment of user's comments and the prices of cryptocurrencies

**Use Case****Scope**

The scope of this use case is to do an analysis on the user comments using statistical algorithms in order to create models which will allow us to determine if they have any effect on the price of the cryptocurrency they are discussing

**Description**

This use case describes the process of analysing the comments and prices.

## Use Case Diagram

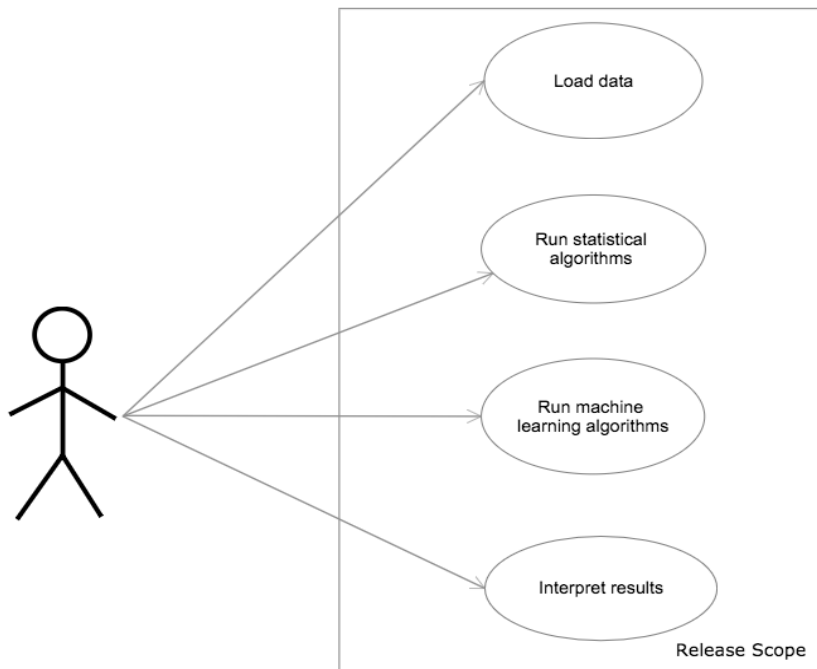


Figure 6: Data Analysis

### Flow Description

#### Precondition

The comments have been classified using sentiment analysis.

#### Activation

This use case begins when an <Actor> loads the data

#### Main flow

1. The <Actor> loads the data into R.
2. The <Actor> runs statistical analysis algorithms on the comments.
3. The <Actor> runs machine learning algorithms on the comments.
4. The <Actor> interprets the results.

### Post condition

The system is ready to visualise the results.

## 2.1.1.6 Requirement 5: Login

### Description & Priority

The user will be able to log into the app with their google account

### Use Case

### Scope

The scope of this use case is to log in the user.

### Description

This use case describes how the user will log into the web app with their email and password.

### Use Case Diagram

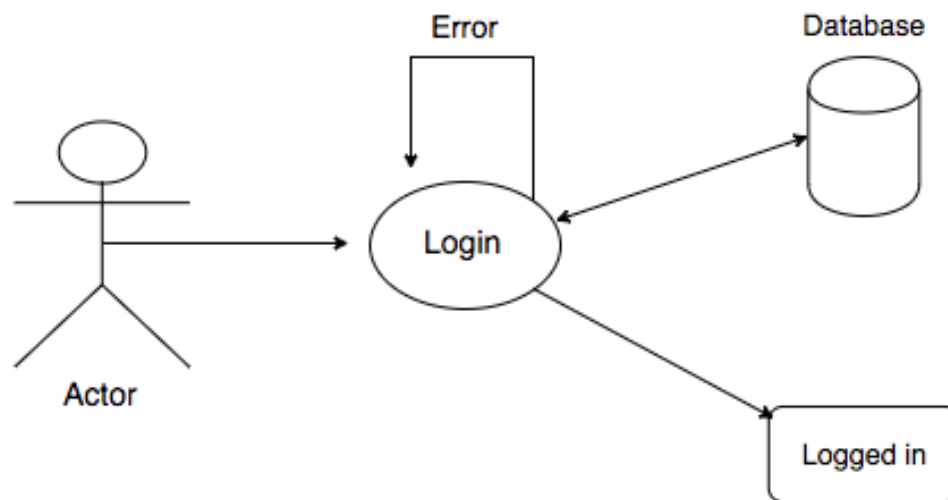


Figure 7: Login

## **Flow Description**

### **Precondition**

The Actor is not logged in

### **Activation**

This use case begins when an Actor clicks the log in button

### **Main flow**

1. The Actor clicks the log in button.
2. The Actor enters their email and password.
3. The Actor is logged in.

### **Exceptional flow**

E1: Invalid credentials

1. The Actor enters their details.
2. The system returns a warning for invalid email or password.
3. The use case continues at step 2 of the main flow.

E2: User not registered

1. The Actor enters their details
2. The system returns an error for email not recognised

### **Termination**

The system logs the user in.

### **Post condition**

The system enters a wait state

## **2.1.1.7 Requirement 6: Search**

### **Description & Priority**

The user will be able to search for a specific cryptocurrency



## Use Case

### Scope

The scope of this use case is for the user to search for a specific cryptocurrency

### Description

This use case describes how the user will search for a cryptocurrency and be redirected to the correct page.

### Use Case Diagram

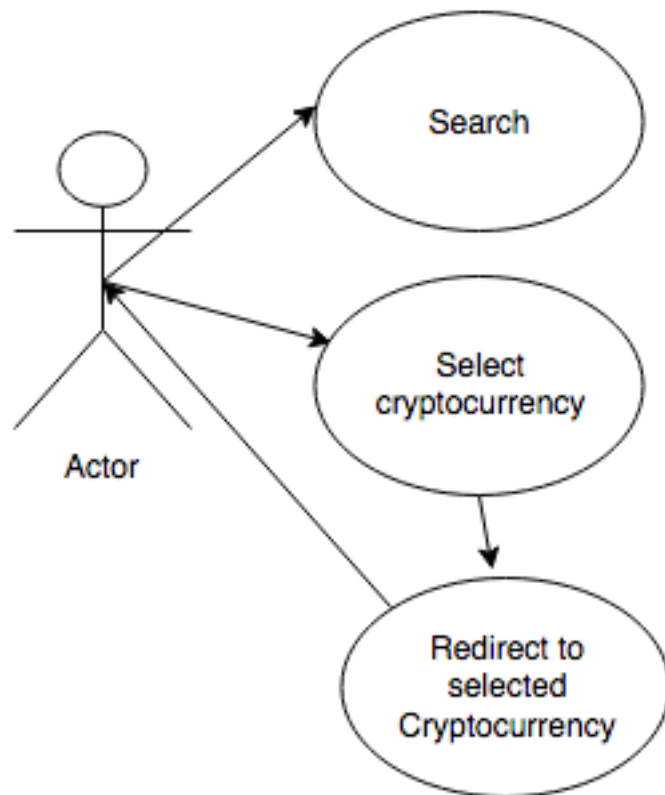


Figure 8: Search

### Flow Description

### **Precondition**

The Actor is on the web app

### **Activation**

This use case begins when an Actor begins typing in the search bar

### **Main flow**

1. The Actor begins typing
2. The system returns suggested results
3. The Actor selects a cryptocurrency.
4. The system redirects to the selected cryptocurrency page

### **Exceptional flow**

E1: No results

1. The Actor enters their details of a cryptocurrency which does not exist.
2. The system returns no results.
3. The use case continues at step 1 of the main flow.

### **Termination**

The system redirects the user.

### **Post condition**

The system enters a wait state

## **2.1.1.8 Requirement 7: View top N cryptocurrencies**

### **Description & Priority**

The user will be able to view the top N cryptocurrencies by market capitalisation

### **Use Case**

### **Scope**

The scope of this use case is for the user to view the top 50 cryptocurrencies by market capitalisation.

### Description

This use case describes how the will view the top N cryptocurrencies.

### Use Case Diagram

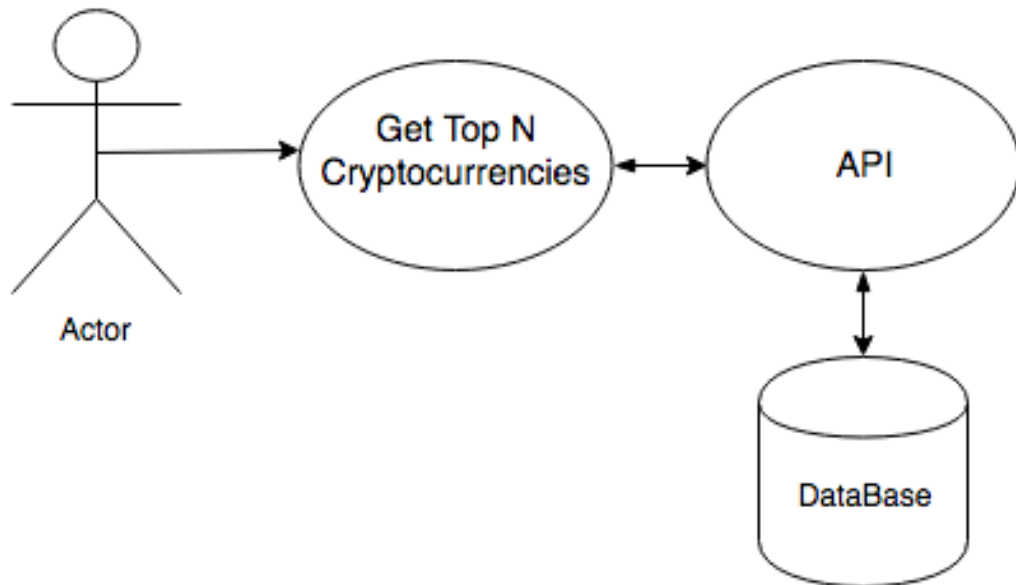


Figure 9: View top N cryptocurrencies

### Flow Description

#### Precondition

The Actor is logged in

#### Activation

This use case begins when an Actor goes to the homepage

#### Main flow

1. The Actor loads the homepage.
2. The system requests the data.

3. The system returns the data.

### **Exceptional flow**

E1: Retrieval error

1. The Actor loads the homepage.
2. The dataset returns an error.
3. The use case begins at step 1 of the main flow.

### **Termination**

The Actor logs out.

### **Post condition**

The system enters a wait state

## **2.1.1.9 Requirement 8: Specific Cryptocurrency data**

### **Description & Priority**

The user will be able to view all data relating to a specific cryptocurrency, such as price, 24-hour mentions volume, sentiment and any other relatable results from the analysis.

### **Use Case**

#### **Scope**

The scope of this use case is for the user to view and interact with all data relating to a specific cryptocurrency.

#### **Description**

This use case describes how the user will view data relating to a specific cryptocurrency.

## Use Case Diagram

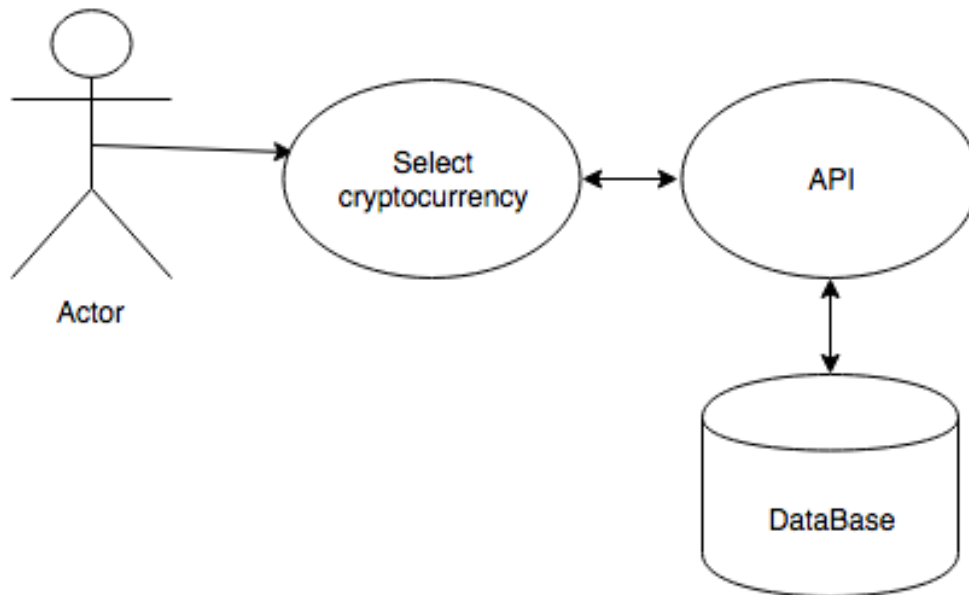


Figure 10: Specific Cryptocurrency data

### Flow Description

#### Precondition

The Actor is logged in

#### Activation

This use case begins when an Actor is on the homepage

#### Main flow

1. The Actor navigates to the homepage.
2. The Actor selects a cryptocurrency.
3. The System redirects to the overview page for that currency
4. The Actor views the data.

#### Exceptional flow

E1: Retrieval error

1. The Actor selects a cryptocurrency.
2. The database returns an error.
3. The use case continues at step 2 of the main flow.

### **Termination**

The Actor logs out or leaves site.

### **Post condition**

The system enters a wait state

## **2.1.2 Non-Functional Requirements**

### **2.1.2.1 Availability requirement**

The system and all its components will be made publicly available through GitHub. The users will be able to access the app through a web browser on mobile, tablet and pc.

### **2.1.2.2 Recover requirement**

All code and datasets will be backed up locally as well as with cloud based storage providers such as Google Drive and GitHub.

### **2.1.2.3 Reliability requirement**

The system will meet all functional requirements. The scripts within the system will run as long as the machine has necessary dependencies installed. The web app will be running at all times and in the event of a server crash the user will have to log back in again.

#### **2.1.2.4 Environmental requirement**

The system will run on mac, windows and Linux. The machine will be required to have python 3 and RStudio installed. The python and R scripts will come with some dependencies which are required to be installed also. The web app is accessible through the web browser and can viewed via any standard internet enabled device.

#### **2.1.2.5 Extensibility requirement**

The system will be extensible by both increasing the size of the data applying more complex algorithms for finding patterns and indicators within the comments. With the use of clustering the system could discover hidden semantic structure within the comments.

### **2.1.3 Data requirements**

To perform the analysis a sufficient dataset is required. Comments will be gathered from the most popular cryptocurrency discussion platforms and prices will be taken from cryptocurrency price aggregator. I chose 2 sources for comments and posts as different platforms cater to different users with different opinions and levels of expertise.

#### **Reddit.com**

Reddit.com is one of the largest sites for newcomer and experienced crypto enthusiasts. The site is built around subreddits or communities devoted to specific topics where users are free to share links and post questions or topics for discussion. A unique feature of reddit is the ability to create a subreddit for any topic you can think of. This has allowed the cryptocurrency community to expand into communities and sub communities devoted to niche areas. A few examples of such would be the bitcoin subreddit [/r/btc](#), which is mainly for bitcoin news and trading discussion, [/r/BitcoinBeginners](#) which is aimed at newcomers to bitcoin who have questions or need advice, and also [/r/CryptoCurrency](#) which is

intended for open discussion on all cryptocurrencies and not just bitcoin. The variety of user base on reddit will allow this project to gather posts and comments from users ranging from first time buyers to blockchain enthusiasts. The most popular cryptocurrency communities on reddit will be chosen and the data will be pulled using reddit's API.

### **Twitter.com**

Although twitter is not solely designed for cryptocurrency discussion it offers a great medium of exchange where news spreads quickly. The users base is much larger than reddit and every level of user from first time buyer to blockchain developers post their opinions there. Twitter offers an API to developers and this project will mainly be focusing on tweets discussing the cryptocurrencies from coinmarketcap.com's list of top 100 cryptocurrencies.

### **Coinmarketcap.com**

Coinmarketcap.com is one of the most popular sources for cryptocurrency prices and statistics and keeps track of cryptocurrency data such as markets, 24-hour volume, market cap and historical data. Coinmarketcap.com offers a public api and this project will utilise it to pull current and historical price data for the top 100 cryptocurrencies.

## **2.1.4 User requirements**

From a client's perspective, the system must display the results of the analysis in a concise and easy to read manner. All visualisations must be clearly labelled along with all relevant information regarding methods and algorithms used to compute the results. The web app requires users to have access to a web browser on an internet connected device. Users are also required to register.

## **2.1.5 Usability requirements**



The web application should have an easy to navigate user interface and no specific level of expertise will be required to use any of its features. The web app should also display clear and concise errors to the user.

## 2.2 Design and Architecture

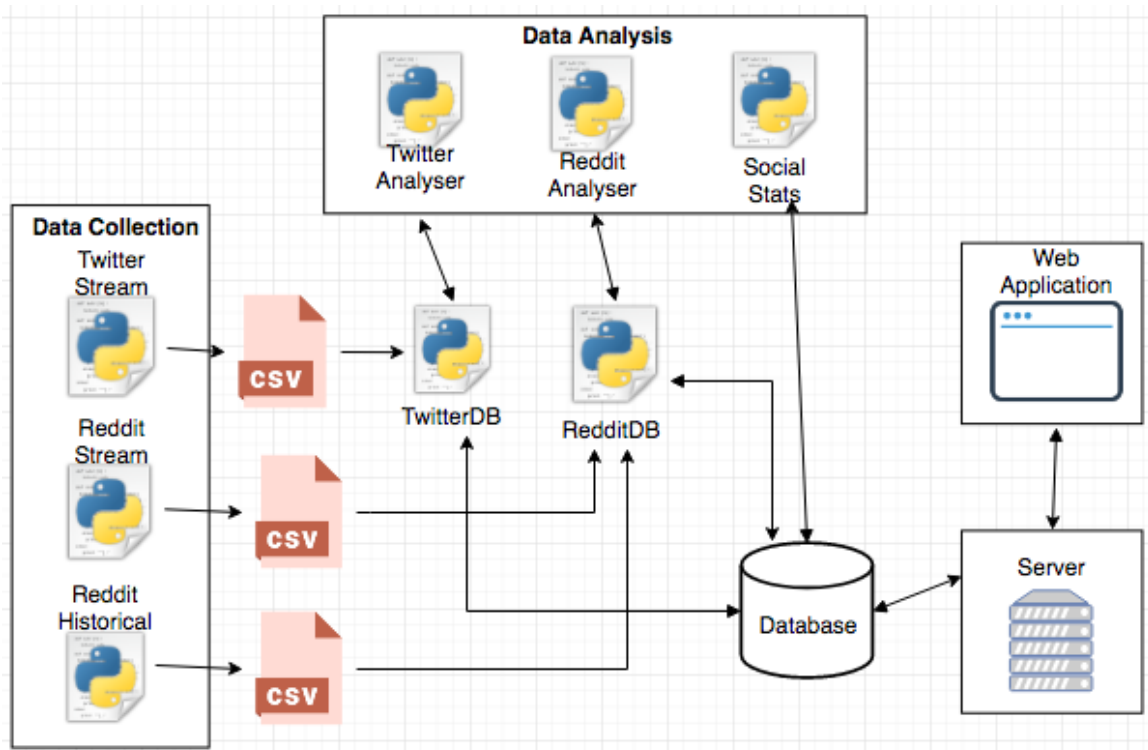


Figure 11: System Architecture

The above architecture diagram shows the system from a high-level view. The system gathers data from both reddit and twitter using their respective API's. Both reddit and twitter offer stream API's for gathering tweets and posts, as they arrive to each site. The system also gathers historical post and comment data from reddit only, as this feature is not available through the twitter API. All gathered data are saved to csv files. The system then analyses the gathered data from reddit and twitter, pulling old data from the database to compare against and update if necessary. The final stage of the data analysis is to generate social stats from the analysed data to provide for the web application.

The web server can then pull this data from the database providing it to the web application.

### 2.2.1 Database Design

The system utilises a MongoDB database for storing the results of the analysis, historical cryptocurrency prices, and various other data which is needed for the web application. Unlike relational databases, which are based on tables, MongoDB is a document oriented schema free database, with each document having key/value attributes. In total, there are 10 different collections with the database, and each collection contains an array documents.

#### 2.2.1.1 Collections

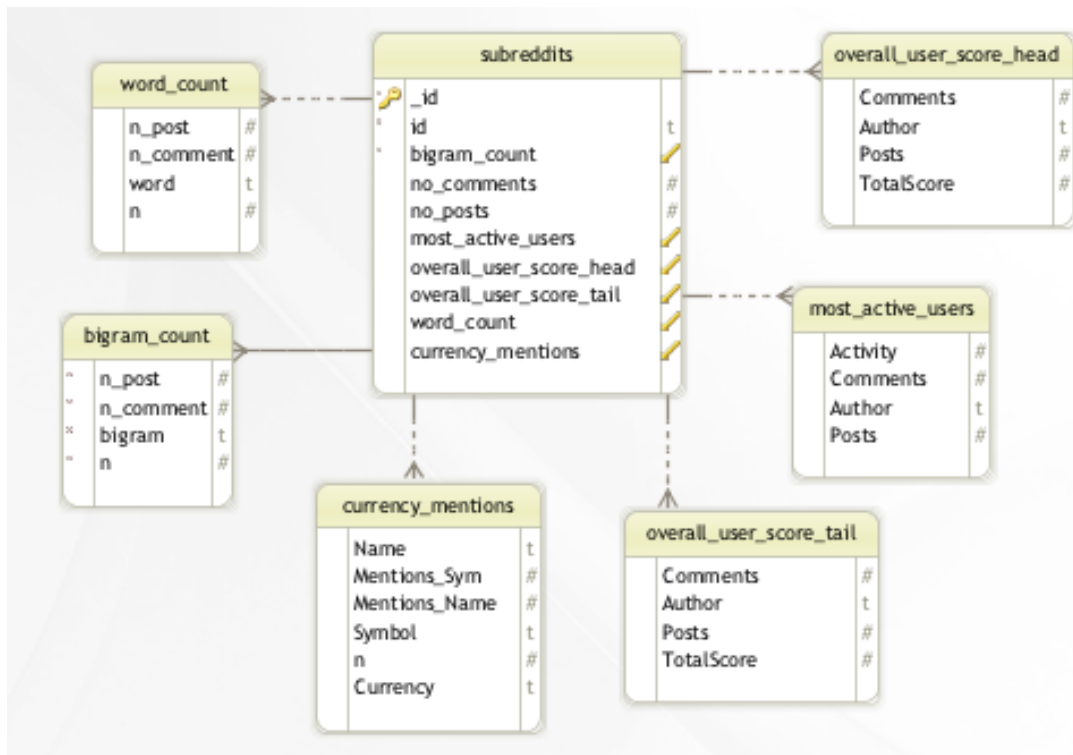


Figure 12: Subreddit collection

Figure 12 provides an overview of the data which is gathered and stored from various cryptocurrency related subreddits. Each document belonging to a

subreddit will contain data on the most popular users, least popular users, most active users, most frequent words, most frequent bigrams and the most popular cryptocurrencies in that subreddit.

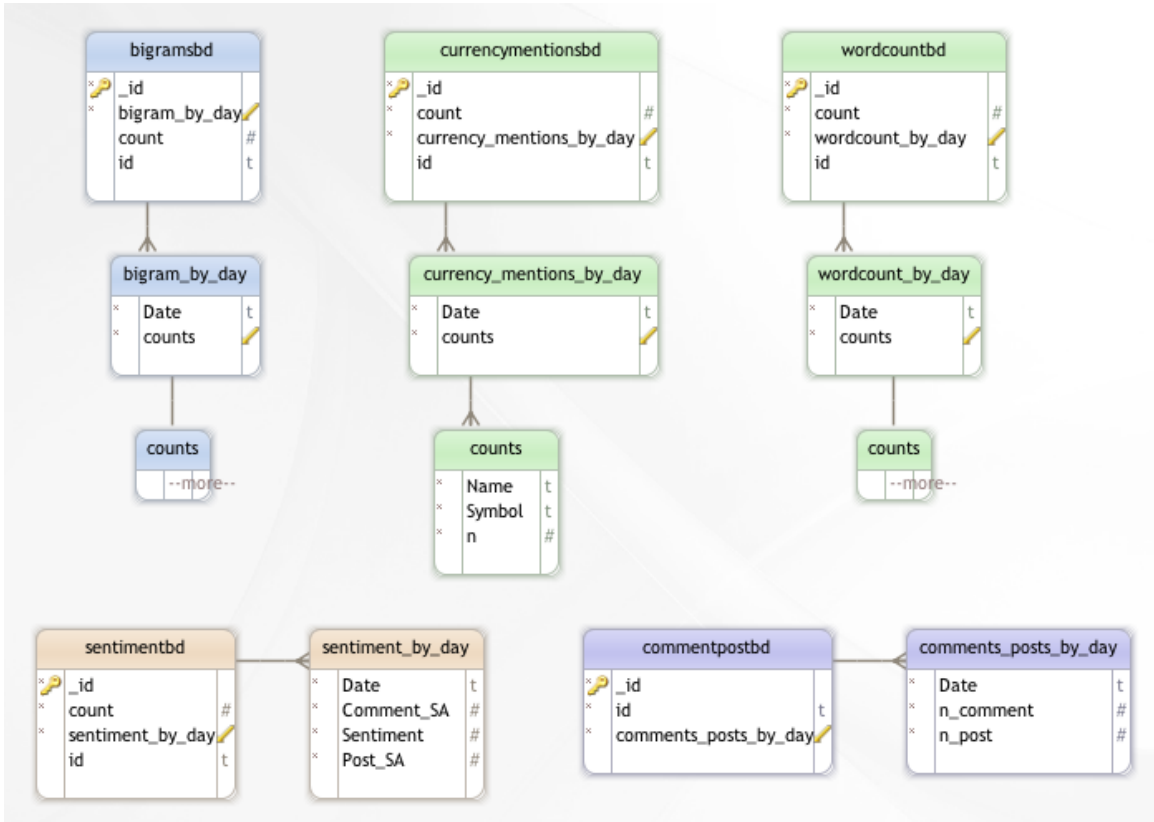


Figure 13: various collections

The 5 collections shown in figure 13 all share a common theme of storing data gathered day to day, from cryptocurrency subreddits and tweets related to cryptocurrencies.

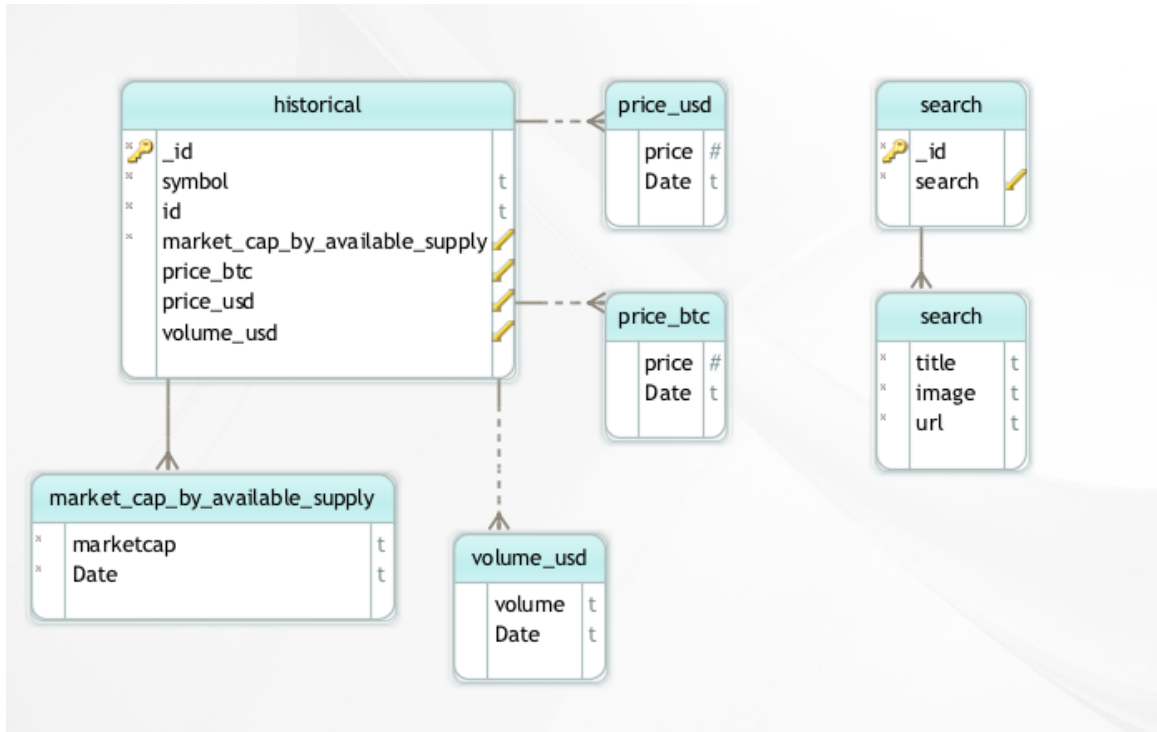


Figure 14: Historical prices and search collections

The historical collection shown in figure 14 contains prices in USD and BTC, as well as market capitalisation and volume for various cryptocurrencies. The search collection is specific to the web application and contains an array of cryptocurrencies available to be searched.

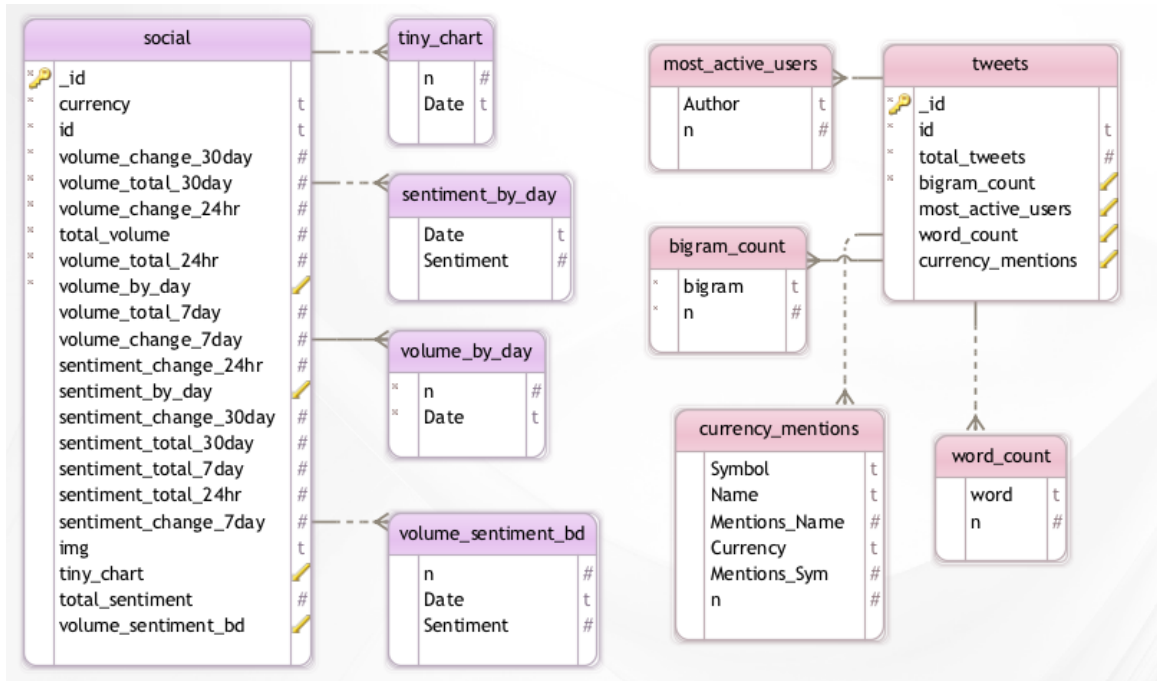


Figure 15: Social and Tweets collections

Similar to the subreddits collection, figure 15 shows the tweets collection which contains data such as word and bigram frequency, most active users and most mentioned cryptocurrencies. The social collection also shown in figure 15 is contains social statistics for various cryptocurrencies, such as sentiment total and change over a 24 hour, 7day and 30-day span, and social media mentions total and change, also over a 24 hour, 7 day and 20-day span. The social collection also contains day to day data on social media mentions and sentiment for a given cryptocurrency.

## 2.3 GUI

The web application was developed using ReactJS and designed with Semantic-UI. ReactJS is an open source JavaScript library developed by Facebook for building user interfaces. It's mainly used for handling the view layer for web application and allows developers to create reusable UI components. The purpose of react is to make simple, scalable and fast web and mobile applications. Semantic-UI is a responsively designed user interface framework, allowing developers to simplify the front end designing experience. Semantic-UI has been fully integrated into ReactJS resulting in a fast and beautiful web app design.

### 2.3.1 Cryptocurrency Overview



The screenshot shows the 'CryptoAnalytics' landing page. At the top, there is a navigation bar with a hamburger menu icon on the left, the title 'CryptoAnalytics' in the center, and a search bar on the right. Below the navigation bar is a table with 12 columns: '#', 'Name', 'Symbol', 'Price', 'Market Cap', 'Social Mentions' (subdivided into 'Change (24h)', 'Total (24h)', and 'Total (7d)'), 'Social Sentiment' (subdivided into 'Change (24h)', 'Total (24h)', and 'Total (7d)'), and 'Mentions Chart (7d)'. The table lists seven cryptocurrencies: Bitcoin, Ethereum, Ripple, Bitcoin Cash, EOS, Litecoin, and Cardano. Each row includes the cryptocurrency's icon, name, symbol, price, market cap, and social media statistics. The 'Mentions Chart (7d)' column contains small area charts for each cryptocurrency. The table data is as follows:

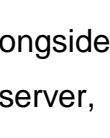




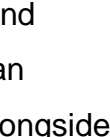
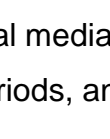
#	Name	Symbol	Price	Market Cap	Social Mentions			Social Sentiment			Mentions Chart (7d)
					Change (24h)	Total (24h)	Total (7d)	Change (24h)	Total (24h)	Total (7d)	
1	Bitcoin	BTC	\$8,603.31	\$146,523,617,858	-44%	12966	96939	2%	17	118	
2	Ethereum	ETH	\$698.19	\$69,404,686,962.0	-25%	4519	37445	7%	13	96	
3	Ripple	XRP	\$0.701185	\$27,479,417,880.0	-49%	3271	29401	-9%	16	110	
4	Bitcoin Cash	BCH	\$1,477.69	\$25,305,884,557.0	-45%	4558	35641	-1%	15	107	
5	EOS	EOS	\$14.5776	\$12,433,023,033.0	72%	1657	13794	-2%	12	89	
6	Litecoin	LTC	\$140.642	\$7,947,871,066.0	-44%	420	2999	8%	6	40	
7	Cardano	ADA	\$0.275225	\$7,135,777,989.0	46%	484	2620	5%	11	75	

Figure 16: Landing Page

The landing page of the site is shown in figure 16. Design inspiration was taken from other cryptocurrency price aggregator sites where the currencies and associated stats are shown in tabular form. The landing page provides an overview of the current cryptocurrency market with social media stats alongside prices. When the page is first loading up the data is requested from the server, and a loading symbol is display to the user until the page is ready. Social media mentions and sentiment values are shown for a 24 hour and a 7 day periods, and

a mentions chart is also provided in the far-right column to give a quick glimpse at the social media buzz for that currency over the last 7 days. Each column of the table is sortable allowing users to sort by mentions, sentiment or price. The default sort is by market capitalisation. Each column in the table also provides a popup when hovered over, which explains the data contained in the column. There is also a search bar in header of the main page which allows users to search for a specific cryptocurrency.

### 2.3.2 Individual cryptocurrency section

Clicking on the row of a cryptocurrency on the main landing page will bring the user to an overview page display an array of information on the clicked cryptocurrency. The whole page is split up into segments, with a grid layout, making the page fully responsive. There are quite a few segments provided for each cryptocurrency so I have broken the overview page down into 3 parts for this demonstration.

#### 2.3.2.1 Main stats and price/volume/sentiment chart

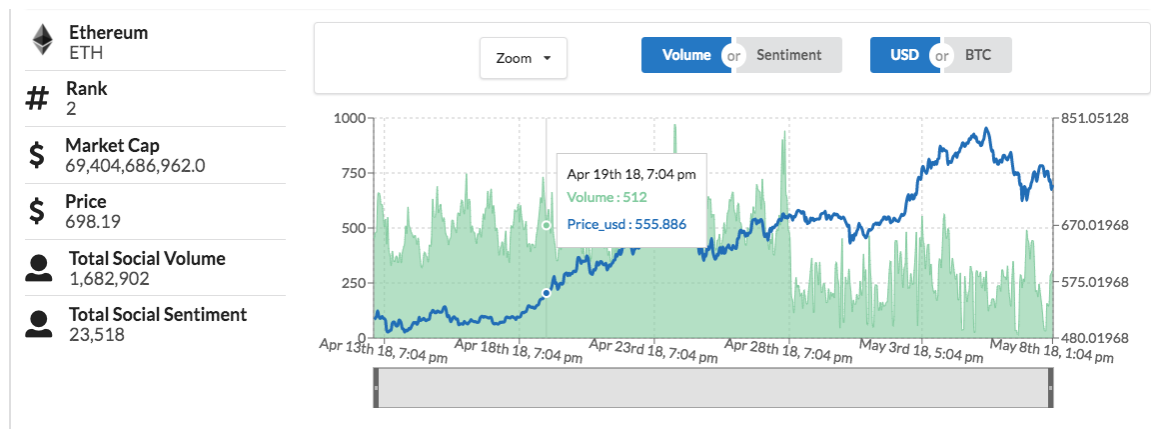


Figure 17: Main stats and chart

The top of the overview page contains current stats for the given cryptocurrency and a chart which users can view historical prices and social media mentions and sentiment. The list on the left provides stats such as the total social volume, which is how many times the currency has been mentioned, and the total sentiment, which is the summed sentiment from all posts this given currency is

mentioned in. The chart provides a way of viewing historical data for the selected cryptocurrency. The zoom dropdown allows users to choose the length of time to view, either 7 days, 1 month or 3 months' worth of data. The chart consists of a line and an area element, the line representing the price and area representing the social media volume. Hovering over a given point in the chart will bring up a legend showing the date, and the price and social media volume for that date. Two buttons in the chart header allow easy switching between displaying social media sentiment and volume on the chart, as well as displaying the price in USD or BTC.

### 2.3.2.2 Associated currencies and social stats

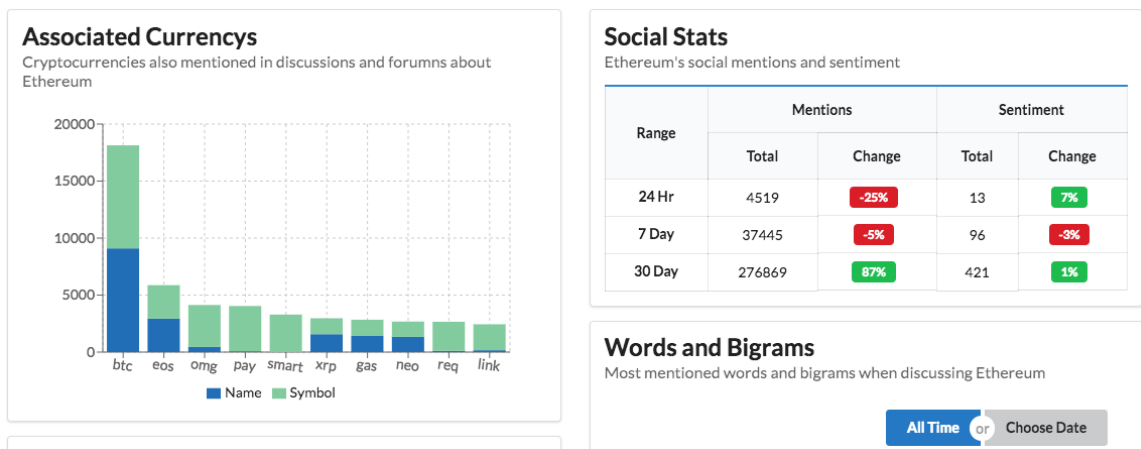


Figure 18: Associated currencies and social stats

Below the chart and main stats are the two segments for associated currencies and social stats. The associated currencies segment displays a bar chart with all other currencies which are mentioned in discussions about a given cryptocurrency. In this case, it's all mentioned currencies in discussions about Ethereum, and Bitcoin is number one mentioned by a good margin. The social stats table provides an overview for the social media sentiment and volume for a given cryptocurrency. The columns Total and Change are provided for mentions and sentiment over a 24 hour, 7day and 30-day time span. Again, hovering over the column header will provide more details to the user on the data within each



column. The change column for both mentions and sentiment displays a different colour depending on whether the change is positive or negative. The change is based on the value in the Total columns compared to the value from the previous period. In this example Ethereum's mentions on social media is up 87% compared to the previous 30 days.

### 2.3.2.3 Most active users, words and bigrams by date



Figure 19: Most active users and words and bigrams

Figure 19 shows the segments for most active users and the words and bigram counts. The Most active users table provides a list of users who discuss the given cryptocurrency the most. The activity column shows how many posts and comments have been captured from this user talking about the given cryptocurrency. Clicking on a user's name will redirect to the user's reddit profile. The words and bigrams bar charts provide the user with the most spoken words and most spoken bigrams (pair of words), when discussing a given cryptocurrency. There is a button at the top of the segment allowing users to

choose between all-time counts, or choose a date. Choosing a specific date will display the most spoken words and bigrams for that date.

### 2.3.3 Search

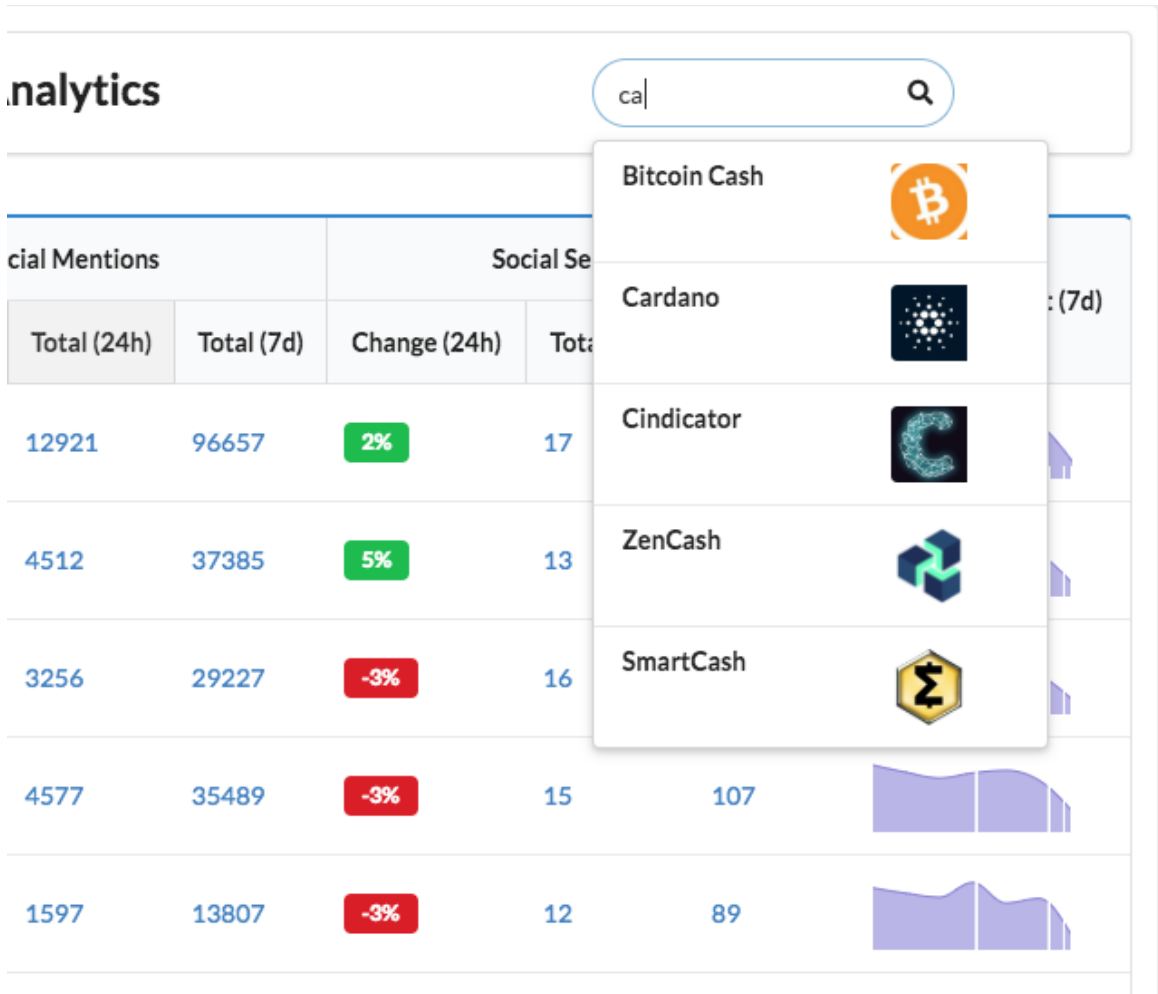


Figure 20: Search bar

The search bar on the top of the page allows users to search for a given cryptocurrency which may not be immediately visible to them on the landing page. As the user inputs the name of the cryptocurrency the list show dynamically changes, filtering in and out cryptocurrency depending on what the user has typed. Clicking on one of the currencies will bring the user to the overview page for that currency.

## 2.3.4 Login

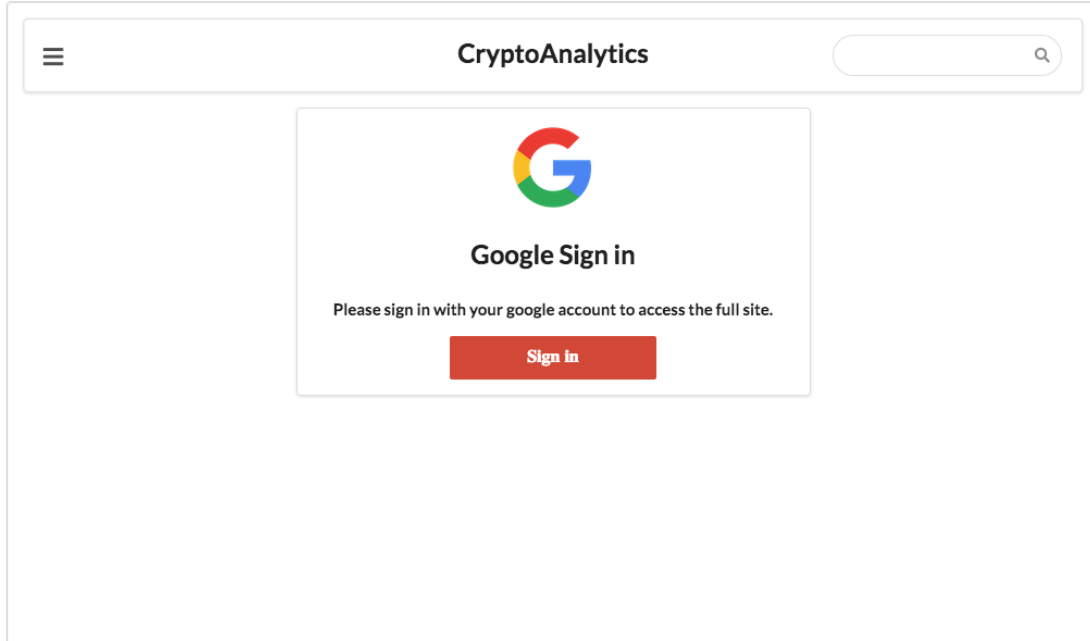


Figure 21: Log in

Before the user can access the overview for a specific cryptocurrency they are required to login using their google account. Once logged in they will be redirected to the currency they previously selected.

### 2.3.4.1 User logged in

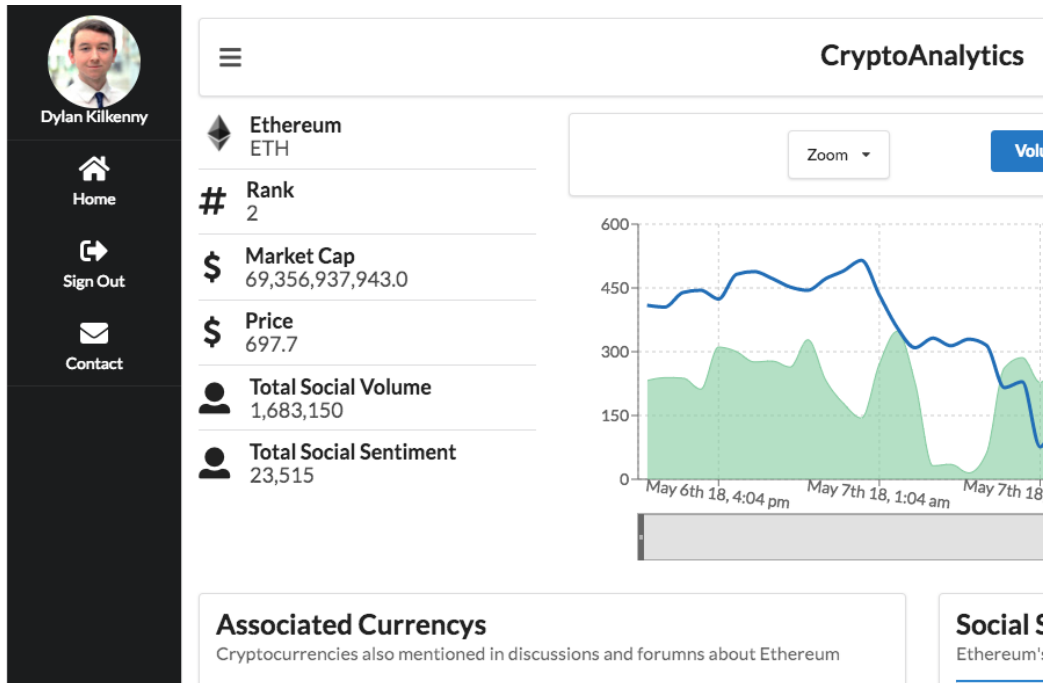


Figure 22: User logged in

Once a user is logged in they can see their profile photo and name when they open the side menu.

### 2.3.4.2 User logged out

**CryptoAnalytics**

#	Name	Symbol	Price	Market Cap	Social Mentions		
					Change (24h)	Total (24h)	Total
1	Bitcoin	BTC	€7,180	€122,288,162,954	-50%	12921	9665
2	Ethereum	ETH	€583	€57,988,295,943.0	-30%	4512	3738
3	Ripple	XRP	€1	€22,940,921,553.0	-52%	3256	2922
4	Bitcoin Cash	BCH	€1,224	€20,954,162,803.0	-50%	4577	3548
5	EOS	EOS	€12	€10,327,246,389.0	20%	1597	1380

Figure 23: User view

If the user is not logged in the side menu will not display anything at all.

## 2.4 Implementation

The main goal of this project was to determine whether the price movements of cryptocurrencies could be predicted using data from social media. The secondary goal was to provide a web application, similar to a cryptocurrency price aggregation site, but with the added features of measuring social media mentions and sentiment for a given cryptocurrency.

### 2.4.1 API Server

A NodeJS server was setup to provide an API for the web application to make requests to the database. ExpressJS is a NodeJS framework which allows the set-up of middleware to respond to HTTP requests.

```
// Retrieves all currencies from db
app.get('/AllCurrencies', function (req, res) {
  // Assign quantity
  var quantity = req.query.quantity
  // Get live cryptocurrency data from coinmarketcap
  getCoins(null, quantity, function (coins) {
    // Retrieve social stats from database
    RedditAPI.Social(db, function (social) {
      // map retrieve cryptocurrency data and social stats on id
      const result = coins.map(val => {
        return Object.assign({ _id: val.id }, val, social.filter(v => v.id === val.symbol)[0]);
      });
      // filter out results without any social stats
      const filter = result.filter(obj => {
        if ('tiny_chart' in obj) {
          if (obj.tiny_chart.length > 0) {
            return true
          }
        }
      })
    })
    // return json object
    res.json(filter)
  })
});
```

Figure 24: /AllCurrencies endpoint

Figure 24 demonstrates one of the endpoints implemented on the API server. The above endpoint is used for the homepage of the web application. A HTTP GET request is made to the endpoint, along with a query parameter for how many currencies to return. The quantity is passed as an argument to the

getCoins function, which will return the specified number of cryptocurrencies from coinmarketcap.com API. From there the database is queried for social statistics for all cryptocurrencies. The two arrays, of cryptocurrencies, and social stats generated for cryptocurrencies in the analysis system, are mapped to one array by the key ID. The merged array is then filtered to remove any cryptocurrencies where no social stats were present, and the resulting array is returned to the web application.

### ***2.4.2 Analysis System***

To begin this project, I had to decide on which cryptocurrencies to keep track of. There are currently over 1500 cryptocurrencies listed on coinmarketcap.com, the majority of which see little price action or recognition by users on social media. To whittle it down I chose to focus on the top 200 cryptocurrencies on coinmarketcap.com. Throughout this project python was heavily utilised to save time on menial tasks, such as aggregating lists of cryptocurrencies and scraping their respective social media accounts and subreddit URLs.

Once I had decided on which currencies to keep track of, and had gathered the information necessary to keep track of them, such as full names, ticker symbols, abbreviations, and social media and subreddit URLs, I began gathering historical and current cryptocurrency data. Before development on the web server or web app even began, the data collection and analysis system needed to be up and running on the cloud 24/7. One of the aims of this project is to determine whether cryptocurrency price movements can be predicted using historical price and social media data related to a given cryptocurrency.

#### ***2.4.2.1 Environment***

The entire system is hosted on an Ubuntu server with 4gb of ram and 2 CPU's, provided by DigitalOcean.com. The server is accessed through remote login over SSH and file uploads and downloads are done over SFTP. Many cronjob's have been setup to keep the system ticking over. Data collection happens in 3 parts,

an ongoing stream of tweets which are saved to csv files, an ongoing stream of reddit posts and comments which are saved to csv files and lastly gathering of historical data from reddit also saved to CSV files. The two streaming scripts for reddit and twitter, which connect to the respective streaming API of each site, are running 24 hours a day, continuously saving tweets, posts and comments to CSV files. Two cronjobs have been setup which call another python script to organise and pass the gathered CSV files to the analyser classes. These cronjobs are executed once every hour, at separate times, in order to avoid resource overload.

#### ***2.4.2.2 Twitter Stream***

A good majority of cryptocurrency discussion takes place on twitter. Usually announcements are made there and eventually shared on reddit and other social media sites. Luckily, Twitter offers a streaming API which allows downloading of tweets in real time and it is very useful for gathering a high volume of tweets related to vast array of subjects. Unlike a REST API, where data is pulled, the twitter streaming API pushes data to a persistent session. This allows more data to be downloaded in real time than could be done with a REST API.

The Twitter API is quite exhaustive and difficult to navigate. In order to speed up development and reduce the number of headaches, Tweepy, a python twitter wrapper was utilised.

```

if __name__ == "__main__":

    hashtags = []
    # Load currencies
    CURRENCYS = pd.read_csv('../data/CurrencySymbols.csv')
    # Create hashtag list
    for i, row in CURRENCYS.iterrows():
        ~
        ~
        |   hashtags.append("#"+row["Name"])
        |   hashtags.append("#"+row["Symbol"])
    # Create OAuth instance
    ~
    AUTH = tweepy.OAuthHandler(PERSONAL_KEY, PERSONAL_SECRET)
    AUTH.set_access_token(ACCESS_TOKEN_KEY, ACCESS_TOKEN_SECRET)
    # Create stream instance
    TWITTER_STREAM = Stream(AUTH, MyListener())
    # Filter based on hashtags
    TWITTER_STREAM.filter(track=hashtags[:400])

```

Figure 25: Stream instance and filter

The above snippet is taken from the python script which implements the tweepy library for streaming tweets. Figure 25 shows the creation of the OAuth instance necessary for communicating with the twitter API, and the creation of the stream instance for receiving live tweets. In order to only gather tweets related to cryptocurrencies the stream needs to be filtered. A csv file containing the names and ticker symbols for 200 cryptocurrencies is loaded in and hashtags are created from the names and symbols which are added to the list and fed to the twitter stream to keep track of.



```

def on_data(self, data):
    # Load tweet json
    tweet = json.loads(data)
    # If a stand alone tweet and not a retweet
    if "extended_tweet" in tweet and (not tweet["retweeted"]) and ('RT @' not in tweet["text"]):
        # print and update counter
        print("Count: " + str(self.count), end="\r")
        self.count += 1
        # transform date
        date = time.strftime('%Y-%m-%d %H:00:00', time.strptime(tweet["created_at"],
                                                              '%a %b %d %H:%M:%S +0000 %Y'))
        # Append relevant data to tweets list
        self.tweets.append([
            date,
            tweet["user"]["screen_name"],
            tweet["extended_tweet"]["full_text"],
            tweet["retweet_count"],
            tweet["favorite_count"]
        ])
    # If length of tweets list == 5000, export to csv
    if len(self.tweets) == 5000:
        t = self.tweets
        self.tweets = self.tweets[:1]
        self.SaveTweet(t)

```

Figure 26: Twitter stream on data function

Figure 26 demonstrates the stream listener class which was implemented from the tweepy library. When a tweet is received, the above function is called and the tweet is transformed to a python object from json. The tweet is then checked for whether it's a reply, a retweet or a standard tweet. The ideal tweet in this case is one which is not a retweet or reply and just a normal tweet. The date of the tweet is then transformed to an appropriate format for the analysis stage. After the required fields from the tweet are added to a list containing all received tweets, the length of the list is checked. If the list contains 5000 tweets, the list of tweets is saved to a CSV file and the list is emptied. This is done for a few reasons. The amount of memory on the Ubuntu server where the system is hosted is quite limited. Preventing the tweets kept in memory from getting too large, and preventing the size of the dataset which will be fed to the data analyser from getting too large, are necessary precautions to keep the system running smoothly.

### 2.4.2.3 Reddit Stream and Historical data

Gathering Reddit posts and comments happens in multiple ways. Most importantly, a reddit stream is setup from receiving a live feed of comments and posts. Similar to the twitter stream api, data is pushed instead of pulled, allowing more data to be gathered at once. In order to speed up development a reddit api wrapper was utilised. PRAW, which stands for python reddit api wrapper, allows simple access to the reddit api and even takes care of a lot of the reddit API rules, such as handling the OAuth, user-agent and requests per minute limit.

```
# Get subreddits
subreddits = reddit.subreddit(subs)
# Stream comments recieved to subreddits
for comment in subreddits.stream.comments():
    # Get post
    submission = comment.submission
    # Add comment to list
    comments.append(
        [comment.id, comment.id, comment.body, comment.created_utc, comment.score, comment.author])
    # add the comment to the appropriate csv file
    SaveComments(str(submission.subreddit), comments)
    # Get postIDs
    PostIDs = pd.read_csv('../data/reddit/PostIDs.csv')
    # Check if already processed
    if PostIDs['ID'].str.contains(submission.id).any():
        continue
    # Add current postID to data frame and save back to csv
    PostIDs.loc[len(PostIDs)] = submission.id
    PostIDs.to_csv('../data/reddit/PostIDs.csv', sep=',', index=False)
    # Add post to the list
    posts.append(
        [submission.id, submission.title,
         submission.created_utc, submission.score,
         submission.num_comments, submission.author])
    # Save post to appropriate csv file
    SavePosts(str(submission.subreddit), posts)
```

Figure 27: Reddit stream

The reddit stream takes a list of subreddits to be loaded as an argument. The submissions for these subreddits are then received and added to a queue for parsing. Figure 27 is a snippet from the core logic of the Reddit stream. The for loop is contained within a try-except clause for handling errors, as well as within a while loop to keep the stream open. When comments are received through the stream they are added to a comment list and then added to the appropriate csv

file. Each subreddit has its own CSV files for comments and posts, so as to not mix up all the subreddits comments and posts in one file. Once the comments have been added to the CSV, the PostIDs file is loaded. This file contains a list of previous posts which have been checked. Since people can comment on a post at any time, and the type of api stream which is open only requests comments, the likelihood of reparsing an already parsed post is extremely high. To prevent such cases of duplicate posts being added, all posts which have been added need to be kept track of, and the current post ID is compared against this list. If it is not present in the list, its post ID is added to the list and the post is saved to the appropriate CSV file for analysis later. The Reddit stream is designed to run continuously, 24 hours a day. Posts and comments get added to CSV files and once an hour a helper script is called to deal with the files. This script will be discussed more in depth in a subsequent section.

#### ***2.4.2.4 Reddit Historical Data***

One of the aims of this project is to determine whether a predicted model can be created to determine the price movements of cryptocurrencies based on social media volume and sentiment. The Twitter and Reddit streams can only gather the latest data, and will not have nearly enough data to base a predictive model on. Therefore, historical data also needs to be collected through Reddits API.

When the script for gathering historical data was originally designed, it used an endpoint available through Reddits API which allowed direct querying for posts between a date range. This was very handy, as for example, I was able to make a request for all posts which happened over the course of 1 month.

Unfortunately, reddit removed this endpoint towards the end of march. This was a major disruption to the development of this system as not nearly enough historical data had been gathered at this stage. I was lucky enough to find a third-party service, pushshift.io, which was keeping track of reddit posts using Reddits streaming API. The workaround I eventually developed was to query pushshift.io for all post ID's between a date range, add all the ID's to a list and then query the

reddit api for each individual post directly. This allowed continuous gathering of historical data without the need for Reddits search API. Though there is a drawback to this solution. Using the original Reddit search API, posts were received in batch. Using this new method with the pushshift API, posts had to be queried individually from reddit, and with a 1 second delay between requests, this drastically slowed down the rate at which historical data was collected.

```
for ID in post_ids:
    # Get reddit post
    submission = reddit.submission(id=ID)
    # Add post to the list
    posts.append(
        [submission.id, submission.title, submission.subreddit_name_prefixed,
         submission.created_utc, submission.score, submission.num_comments,
         submission.author, submission.shortlink])
    # Add post to csv file
    SavePosts(subreddit, posts)
    # reset post list
    posts[:] = []
    #If post does not have any comments skip it
    if submission.num_comments == 0:
        continue
    #If comments have alot of nested replies this will load them also
    submission.comments.replace_more(limit=0)
    #Add comments to list
    for comment in submission.comments.list():
        comments.append(
            [comment.id, submission.id,
             comment.body, comment.created_utc, comment.score, comment.author])
    # add comments to csv file
    SaveComments(subreddit, comments)
    # reset comment list to empty
    comments[:] = []
```

Figure 28: Historical Reddit Data

The core logic in the historical data gathering script is very similar to the Reddit stream logic. Figure 28 shows the for loop within the main function which loops through the post ID's, querying reddit for an individual post and adding the relevant information from that post to a list and then saving that list to the CSV file for that subreddit. Posts are checked if they have any comments, and they too are saved to a CSV file.

#### ***2.4.2.5RedditDB and TwitterDB***

The Reddit Analyser and Twitter Analyser classes house the current gathered dataset, and a call to any given function within each class will return a json object containing summarised data which can be then added to the database. RedditDB and TwitterDB are two helper scripts for controlling the flow of both old and new data, to and from the Reddit and Twitter analyser classes. Again, both RedditDB and TwitterDB are quite similar, but the functions within each are tailored for the type of data which is being processed from each website. Within the main function of each script, a connection is made to the MongoDB database, the relevant analyser class is instantiated with the datasets, and various functions contained in the scripts are called in order to add the summarised data from the analyser class to the database.

```

def SetCurrencyMentions(RA, db, subreddit):
    # Find subreddit document
    cursor = db.subreddits.find({"id": subreddit})
    for doc in cursor:
        # If already exists
        if "currency_mentions" in doc:
            # call reddit analyser function with old data
            cm = RA.CurrencyMentions(doc["currency_mentions"])
            # Remove old data from document
            db.subreddits.update(
                {"id": subreddit},
                { "$unset": { "currency_mentions": ""}})
            # Add new merged data
            db.subreddits.update_one(
                {"id": subreddit},
                {"$set": {"currency_mentions": cm}})
        # else it does not exist
        else:
            # call reddit analyser function without old data
            cm = RA.CurrencyMentions(None)
            # update document with summarised data
            db.subreddits.update_one(
                {"id": subreddit},
                {"$set": {"currency_mentions": cm}})

```

Figure 29: Set Currency Mentions

Figure 29 shows one of the functions within RedditDB which gets an array of cryptocurrencies and how many times they've been mentioned in that specific subreddit. The above function is passed 3 arguments, an instance of the Reddit Analyser, the database connection and the name of the subreddit which is being analysed. The database is queried for any documents relevant to the current subreddit and checks whether the field already exists. If so, the data currently in that field is passed to the analyser class where it will be merged with the latest data. The old data is then removed from the document and the merged data is added. If no old data exists, which can happen with smaller subreddits who do

not receive a lot of posts, then the analyser class only returns the latest data, which is then added to the database.

There are a total of 12 functions within the RedditDB and 9 within TwitterDB which request summarised data from the analyser class, each one providing a unique insight into the given dataset. Each function within RedditDB and TwitterDB are passed the same arguments as the function shown in figure 29, and they all follow the same routine of checking for old data before requesting the latest data from the analyser class.

#### ***2.4.2.6 Reddit and Twitter Analyser***

The most important components of the whole system are the Reddit and Twitter analyser classes. They are both quite similar, but 2 different implementations were required in order to allow for differences in the structure of the data, and also the type of info which would be extracted from the two sites. For the purpose of this explanation I will only refer to the Reddit analyser, but what I am describing applies to both, and there are only few logical differences between the two classes.

Upon instantiation of the Reddit Analyser, 5 arguments are passed, the gathered posts, the gathered comments, a list of cryptocurrencies, a list of stopwords, and a list of banned users. A cleansing function is run on the comments and posts in order to prepare them for analysis.

```

def CleanseData(self, source, posts):
    #Change date format
    source["Date"] = pd.to_datetime(source["Date"],unit='s')
    #Create df object
    if posts:
        data = {"Author": source["Author"], "Text": source["Title"], "Date": source["Date"], "Score": source["Score"] }
    else:
        data = {"Author": source["Author"], "Text": source["Body"], "Date": source["Date"], "Score": source["Score"] }
    #Create df
    data = pd.DataFrame(data=data)
    #Convert datetime to date
    data["Date"] = data["Date"].dt.strftime('%Y-%m-%d %H:00:00')
    #Remove URLs
    data["Text"] = data['Text'].str.replace(r'http\S+', '', case=False)
    #Remove Na's
    data = data.dropna(how='any',axis=0)
    #Remove punctuation
    data["Text"] = data["Text"].str.replace('[^\w\s]','')
    #To lower case
    data['Text'] = data.Text.str.lower()
    #Remove Stop words
    stop = self.stopwords['word'].tolist()
    data["Text"] = data["Text"].apply(lambda x: ' '.join([word for word in x.split() if word not in stop]))
    with open(self.banned_path, "r") as jsonFile:
        users = json.load(jsonFile)
    data = data[~data['Author'].isin(users["users"])]
    return data

```

Figure 30: Cleanse Data

The above snippet in figure 30 shows the cleansing function which is run on each instantiation of the analyser class. The main purpose of this function is to transform the date to an appropriate format which allows easy grouping by hour, remove any urls, punctuation and missing values and convert all text to lower case. Stopwords are also removed from all text based on a predefined list of stopwords passed to the analyser. Lastly users whose names appear in the banned users file are also dropped from the dataset. These users are made up exclusively of bot accounts which are not relevant to this project and have had a noticeable negative effect on the results of previous analysis.

Once the Reddit Analyser has been instantiated the functions within the class can be called to analyse the cleansed data and return summarised data.



```

def SentimentByDay(self, oldsbd):
    # Copy of comments and posts
    cs = self.comments.copy()
    ps = self.posts.copy()
    # Create a new column with sentiment score for each piece of text
    cs['Comment_SA'] = np.array([ self.AnalyseSentiment(text) for text in cs['Text'] ])
    ps['Post_SA'] = np.array([ self.AnalyseSentiment(text) for text in ps['Text'] ])
    # Drop unneeded columns
    cs = cs.drop(['Author', 'Score', 'Text'], 1)
    ps = ps.drop(['Author', 'Score', 'Text'], 1)
    # Group and sum the scores by hour
    cs = cs.groupby('Date')['Comment_SA'].sum().reset_index()
    ps = ps.groupby('Date')['Post_SA'].sum().reset_index()
    # Merged the two dataframes, and drop and NA values
    sbd = pd.merge(cs, ps, on='Date', how='outer')
    sbd.fillna(0, inplace=True)
    sbd["Sentiment"] = sbd["Comment_SA"] + sbd["Post_SA"]
    # If old data
    if oldsbd != None:
        # Create data frame from records
        oldsbd = pd.DataFrame.from_records(data=oldsbd)
        # Merge old and new data
        oldnew_merged = pd.concat([sbd, oldsbd])
        # Group by hour and sum sentiment scores
        oldnew_merged = oldnew_merged.groupby('Date').sum().reset_index()
        # Convert dataframe to json
        oldnew_merged = oldnew_merged.to_json(orient='records', date_format=None)
        oldnew_merged= json.loads(oldnew_merged)
        # Return json array
        return oldnew_merged
    # Convert dataframe to json
    sbd = sbd.to_json(orient='records', date_format=None)
    # Return json array
    return json.loads(sbd)

```

Figure 31: Sentiment by Day

An example of one of the function is the SentimentByDay function in figure 31 which calculates the hourly sentiment for all posts and comments passed to the class upon instantiation. The AFINN analysis lexicon is utilised in order to assign words a score between -5 to +5, with positive scores indicating positive sentiment and vice versa. The posts and comments are grouped by hour and summed in order to get a sentiment score for that hour. The two data frames are then merged and returned. In the event that previous sentiment by day data already exists for the given subreddit, the old data may also be passed to the function as an argument. The old data is then merged with the current data, grouped by hour

and again summed to get the new updated values. All of the functions in the Reddit Analyser class accept 1 or more arguments to allow for old data to be merged with current data.

### ***2.4.3 Price movement prediction***

Once the analysis system was set up, and enough data was gathered, it was time to build some predictive models to determine whether cryptocurrencies price movements can be predicted based on social media data. Throughout the development of the analysis system, quite a few problems were encountered. For one, twitter.com does not provide an easy way of gathering historical tweets. So, for the purpose of this prediction model I had to rely on reddit data exclusively. Reddit also decided to throw some spanners into the works and midway through this project removed a crucial API for the gathering of historical data on their site. To circumvent this issue, I had to rely on a third-party service for gathering the post ID's of historical posts and then querying the Reddit API for each post individually. This method was slower than the previous by a factor of 50, as I could only request 1 post per second. The data gathering process was much slower than originally anticipated, so in order to develop a proof of concept for a predictive model, separate scripts were set up outside the analysis system to gathering only posts related to bitcoin, while the main analysis system continues working on over 200 currencies. This allowed me to get 18 month's worth of posts from reddit related to bitcoin.

The predicted models were created in RStudio using R and the caret library. The dataset was pre-cleaned through the analysis systems. The only necessary transformation was to merge the historical bitcoin prices and historical bitcoin mentions and sentiment into one dataset and create a new column which specified the price movement of bitcoin. For example, If the current day has a higher price than the previous day, the price movement column would contain "higher" for the current day. This column was the dependent variable for all the models. There was a total of 7 independent variables in the dataset: Close, High, Low, Open, Volume, Mentions and Sentiment. Close, High, Low and open all

related to the Bitcoin price at a given point of the day. Volume is how much USD was traded in bitcoin on that given day, and Mentions and Sentiment are how much bitcoin was mentioned on reddit and the sentiment for that given day.

```
# Set seed
set.seed(1234)
#Splitting data as training and test set. Using createDataPartition() function from caret
indxTrain <- createDataPartition(y = bit_forecast$Diff, p = 0.8,list = FALSE)
training <- bit_forecast[indxTrain,]
testing <- bit_forecast[-indxTrain,]
#Checking distribution in origanl data and partitioned data
prop.table(table(training$Diff)) * 100
prop.table(table(bit_forecast$Diff)) * 100
```

Figure 32: Data partition

Figure 32 demonstrates the first step before creating the predictive models. The dataset had to be split up into two separate datasets, one for providing training the model, and one for testing. The caret library provides a useful functional for partitioning data, which was implemented to split the dataset into an 80:20 split. 80% of the dataset was used for training while 20% was left for testing. After splitting up the datasets, the distribution of the training dataset was compared against the original dataset, which returned no inconsistencies.

### 2.4.3.1 Random Forest

The first model created was a Random Forest model. Random Forest can be effectively seen as a large amount of decision trees, that work to help make a decision based on classification. The *mtry* value, which is a randomly selected predictor is used to identify which iteration can be used to give the most accurate reading of classification. This model is an extremely accurate model.

```
set.seed(1234)
ctrl <- trainControl(method="repeatedcv",repeats = 3)
RFFit <- train(Diff ~ ., data = training,
              method = "rf", trControl = ctrl,
              preProcess = c("center","scale"), tuneLength = 20)
RFPredict <- predict(RFFit,newdata = testing )
#Get the confusion matrix to see accuracy value and other parameter values
confusionMatrix(RFPredict, testing$Diff )
```

Figure 33: Random Forest – R code

The above snippet in figure 33 shows the R code used to implement the random forest predictive model. The ctrl variable controls the number of k-fold cross validation sets. The RFFit is the trained model, which is then used to be compared against the testing dataset for accuracy. The confusionMatrix function creates a cross-tabulation of observed and predicted classes, with associated statistics.

```

                Reference
Prediction Higher Lower
Higher          9     15
Lower         20     29

                Accuracy : 0.5205
                95% CI : (0.4004, 0.639)
                No Information Rate : 0.6027
                P-Value [Acc > NIR] : 0.9389

                Kappa : -0.0315
                McNemar's Test P-Value : 0.4990

                Sensitivity : 0.3103
                Specificity : 0.6591
                Pos Pred Value : 0.3750
                Neg Pred Value : 0.5918
                Prevalence : 0.3973
                Detection Rate : 0.1233
                Detection Prevalence : 0.3288
                Balanced Accuracy : 0.4847

                'Positive' Class : Higher

```

Figure 34: Random Forest Confusion Matrix

Figure 34 shows the results of the random forest predictive model, generate by the confusion matrix. The accuracy of the model is only 52%, which is only slightly better than tossing a coin. The kappa score is very small, -0.03, indicating this model is not very reliable. From observation of the cross-tabulation of

observed and expected values it seems the model is performing significantly worse at predicting when the price will go up compared to when the price will go down. The Random forest model shows poor predictability power for cryptocurrency prices based on social media mentions and sentiment.

### **2.4.3.2 K-nearest neighbour**

The second algorithm implemented was k-nearest neighbours. Better known as KNN, it is also a supervised classification algorithm that uses a value of k to be able to classify similar values based on the k value and output a prediction. The k value can be selected in many different ways, for example k could be the number of the square root of the number of instances in a dataset. So, if we have 100 instances, the value of k would be 10. The overall use of k doesn't make a massive difference but it can be used to make the model more efficient if the correct one is selected. The Knn algorithm was implemented much the same as Random forest. All the algorithms were implemented with the caret library so the code was very similar.

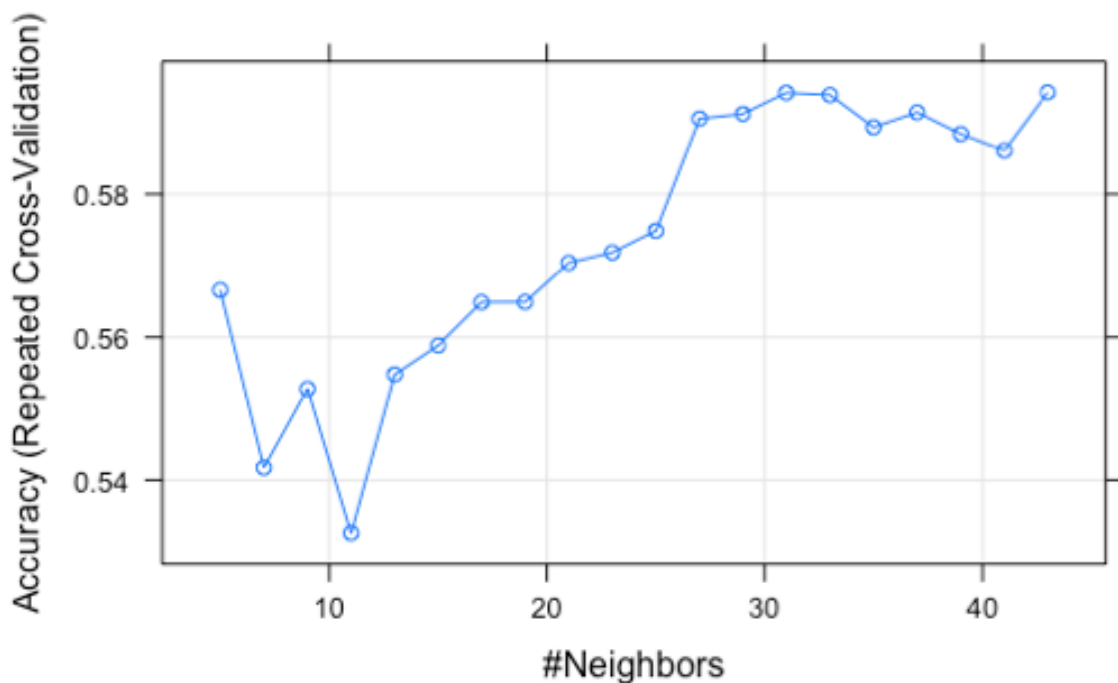


Figure 35: Knn fit plot

The graph shown in figure 35 is the resulting plot from the KNN fit. With a k-value of 42 the model yielded the highest predictive accuracy of around 60%. The accuracy of this model appears to be performing better than random forest, but it is still only slightly better than prediction by chance alone.

```

Reference
Prediction Higher Lower
Higher      4      3
Lower     25     41

Accuracy : 0.6164
95% CI : (0.4952, 0.7279)
No Information Rate : 0.6027
P-Value [Acc > NIR] : 0.4557

Kappa : 0.0801
McNemar's Test P-Value : 7.229e-05

Sensitivity : 0.13793
Specificity : 0.93182
Pos Pred Value : 0.57143
Neg Pred Value : 0.62121
Prevalence : 0.39726
Detection Rate : 0.05479
Detection Prevalence : 0.09589
Balanced Accuracy : 0.53487

'Positive' Class : Higher

```

Figure 36: Confusion matrix for KNN

Upon inspection of the confusion matrix generated for the KNN model, in figure 36, we can see that it is encountering the same problems as the random forest model. With a specificity value of 0.93, the model predicted almost all of the downwards price movements correct, but got barely any of the upwards price movements correct. The accuracy of the model is 61%, but it is extremely towards predicting the price will move downwards, compared to upwards.

### 2.4.3.3 Logistic Regression

Logistic regression is a predictive model that is used to take into account a dependent variable to rule an outcome based on one or many independent variables that are aggregated to construct a probability and output a decision. Logistic regression mainly implements a Boolean format of True or false, 1 or 0. So if the dependent variable was to have more than two values, the solution to use logistic regression would be to dummy code the data frame which would break up all the variables to have 1 or 0 values if categorical. In the price movement prediction case, there are only two outcomes, Higher or Lower, so we can use logistic regression without dummy coding our data frame.

```

                Reference
Prediction Higher Lower
Higher          4      8
Lower          25     36

Accuracy : 0.5479
95% CI : (0.427, 0.6648)
No Information Rate : 0.6027
P-Value [Acc > NIR] : 0.858892

Kappa : -0.0488
Mcnemar's Test P-Value : 0.005349

Sensitivity : 0.13793
Specificity : 0.81818
Pos Pred Value : 0.33333
Neg Pred Value : 0.59016
Prevalence : 0.39726
Detection Rate : 0.05479
Detection Prevalence : 0.16438
Balanced Accuracy : 0.47806

'Positive' Class : Higher
```

Figure 37: Logistic Regression confusion matrix

Figure 37 shows the confusion matrix generated for the final algorithm, logistic regression. Again, the results show the model is biased towards predicting the price will be lower. This may be a result of class imbalance within the dataset. Overall the 3 predictive models performed quite poorly.

## **2.5 Testing**

Software testing, both manual and automated, takes of a good chunk of the development life cycle. Testing confirms that the system meets the various functional, reliability, performance and usability requirements. There are two purposes to software testing, firstly, its done to find bugs and defects within the system, secondly, it's performed to validate the system meets the stated requirements.

### **2.5.1 Unit Tests**

The objective of a unit test is to segregate each component to a program and test the individual parts are working correctly. It isolates a piece of code from the rest of the system and determines whether it behaves as expected.

The analyser classes for reddit and twitter are built with python. Pytest is a python framework which allows the development of scalable and simple tests. Testing with pytest is completely simplified compared to other unit testing frameworks as there is no boilerplate and no required api needed. The pytest package will run all python files prefixed with the word test, checking each functions assertion.



```

### Comments and posts by day
def test_comments_posts_by_day_type_with_data():
    existing = open_desired("comments_posts_by_day")
    comments_posts_by_day = RA.CommentsPostsByDay(existing)
    assert isinstance(comments_posts_by_day, list)

def test_comments_posts_by_day_type_without_data():
    comments_posts_by_day = RA.CommentsPostsByDay(None)
    assert isinstance(comments_posts_by_day, list)

def test_comments_posts_by_day_equal():
    comments_posts_by_day = RA.CommentsPostsByDay(None)
    desired = open_desired("comments_posts_by_day")
    assert comments_posts_by_day == desired

```

Figure 38: Reddit Analyser tests

The above snippet in figure 38 is taken from the Reddit Analyser testing script. There are a total of 38 tests in the script and when pytest is called in the console each function within the script is run. The first two tests check whether the type of the returned value is correct, both with old and without old data being passed to the function. The function which is being called in the analyser class accepts an argument for old data in order to create an updated object with the new and old data. It's important to check the type of the object which results from both with and without the old data being passed, to ensure there are no errors as a result of the passed data. The third function shown in figure 38 checks whether the resulting object is equal to the desired state. Test objects have been created for each function within the analyser classes using test data. This is done to check if the functions are still returning the same outputs when they are given the same data.

```
→ tests git:(master) X pytest
===== test session starts =====
platform darwin -- Python 3.6.5, pytest-3.5.1, py-1.5.3, pluggy-0.6.0
rootdir: /Users/dylankilkenny/Documents/College/FinalYearProject/src/tests, inifile:
collected 58 items

test_reddit_analyser.py .....
test_twitter_analyser.py .....

===== 58 passed in 13.41 seconds =====
```

Figure 39: pytest output

The pytest output in figure 39 is the result of calling pytest in the tests folder. A total of 58 tests were run in both the reddit and twitter analyser class, all of which passed in under 14 seconds.

### 2.5.2 Integration Tests

Integration testing involves testing modules and components as a group, in their complete form, in order to validate their behaviour. Integration tests are done to expose bugs in the interactions between integrated components. The API server developed for providing data to the web application is built using NodeJS and ExpressJS. Mocha, a feature rich JavaScript testing framework, was used for running the tests, and Chai was used as the assertion library.

```

describe('/OneCurrency', () => {
  it('it should get data for one crypto from DB', (done) => {
    var date = new Date();
    date.setDate(date.getDate() - 7);

    chai.request(server)
      .post('/OneCurrency')
      .set('Content-Type', 'application/json')
      .send({ id: 'ethereum', range: date.getTime() })
      .end((err, res) => {
        res.should.have.status(200);
        should.exist(res.body);
        res.body.should.be.a('object');
        res.body.should.have.property('currency_mentions');
        res.body.should.have.property('subreddit');
        res.body.should.have.property('bigrams_words');
        res.body.should.have.property('bigrams_words_by_day');
        res.body.should.have.property('social');
        res.body.should.have.property('historical');
        done();
      });
  });
});

```

Figure 40: Express testing

ExpressJS is a nodeJS framework which allows you to set up middleware to respond to HTTP requests. The above test in figure 40 checks one of the endpoints on the server. Chai makes a HTTP POST request to the /OneCurrency endpoint, which returns data for a specific currency to be used in the overview page. Chai sets the content type as well as sending a payload. Once a response has been received a number of checks are performed. The status of the response is evaluated, the response is checked for a body containing an object and the object is checked for all the valid properties. In total, there are 7 tests for the api server.

```
Server
  /index
    ✓ it should get the text from index
  /AllCurrencies
    ✓ it should get all cryptocurrencies from DB (173ms)
    ✓ it should get <50 cryptocurrencies from DB (114ms)
  /OneCurrency
    ✓ it should get data for one crypto from DB (712ms)
  /Historical
    ✓ it should get historical price data for a crypto (298ms)
  /Subreddit
    ✓ it should get subreddit information from DB (55ms)
  /Search
    ✓ it should get search array
```

Figure 41: npm test output

A test script was setup in the package.json of the NodeJS directory. This allowed the tests to be quickly run when changes have been made and deployed to production. Figure 41 demonstrates the results of running the test script. A total of 6 endpoints were tested, one of which was tested twice with two different parameters. All 7 tests passed.

### 2.5.3 Jest

Jest is a testing framework for ReactJS applications. It was developed and is also used by Facebook. A useful feature of the jest framework is the ability to create snapshots of react components. Jest takes the component which is being tested, renders the component, and takes a snapshot of what it should look like. As updates are made to the project, running the test script will call jest to compare the old snapshot to the new one. If they do not match an error is raised.

```

describe('Table Snapshot', () => {
  test('renders', () => {
    const component = renderer.create(
      <TableContainer />
    );
    let tree = component.toJSON();
    expect(tree).toMatchSnapshot();
  });
});

```

Figure 42: jest snapshot

Figure 42 shows a snippet from the jest testing script. The first time the script is run, jest creates a snapshot of the table container component. On subsequent runs of the test suite jest expects the rendered component to match the snapshot. Snapshots are UI based testing and any changes to the way a component is rendered will raise an error.

#### **2.5.4 Black Box testing**

Black box testing involves examining the functionality of a system without having knowledge of the internal structure of the system. The tester has no access to the source code and only knows what the system is supposed to do rather than how it does it.

The test script I will be using for the black box testing was obtained from Cambridge university press and will have the following sections:

- *Project ID* - the unique project identifier
- *AUT Name* - the definitive name of the Application Under Test (front sheet only)
- *AUT Version* - the definitive version information for the Application Under Test (front sheet only)

- *Iteration ID* - the unique identifier for the iteration this test is being conducted in (front sheet only)
- *Date of Test* - the planned start date of testing (front sheet only)
- Test ID - the unique identifier for the test
- *Purpose of Test* - a brief description of the purpose of the test including a reference where appropriate to the requirement that is to be tested (consider providing references to the requirements specification, design specification, user guide, operations guide and/or installation guide), as well as any dependencies from or to other Test Scripts/Test Cases
- *Test Environment* - a brief description of the environment under which the test is to be conducted (may include a description of the state of the AUT at the start of this test, details regarding the platform or operating system, as well as specific information about data used in this test)
- *Test Steps* - concise, accurate and unambiguous instructions describing the precise steps the Tester must take to execute the test, including navigation through the AUT as well as any inputs and outputs
- *Expected Result* - a brief and unambiguous description of the expected result of executing the test.
- *Actual Result* - a brief and unambiguous description of the actual result of executing the test.

<b>Black Box test 1</b> <span style="float: right;"><i>(front sheet)</i></span>			
<b>AUT Name</b>	Search Cryptocurrency	<b>Version</b>	1.0
<b>Iteration ID</b>	1.0	<b>Date of Test</b>	25/04/2018

<b>Test ID</b>	Blackbox1
<b>Purpose of Test</b>	To ensure that:  Searching for a cryptocurrency in the search bar of the react web app will return suggested currencies based on the input
<b>Test Environment</b>	The test environment is as follows: <ul style="list-style-type: none"> <li>▪ Client hardware: MacBook pro 2012</li> <li>▪ Browser: Chrome</li> </ul>
<b>Test Steps</b>	From the home page the tester should input the name of a cryptocurrency into the search bar.  If the currency inputted is suggested, click on it.
<b>Expected Result</b>	Tester should be redirected to the cryptocurrency page he searched for
<b>Actual Result</b>	Test passed.  The tester was redirected to the overview page for the searched cryptocurrency

<b>Black Box test 2</b> <span style="float: right;"><i>(front sheet)</i></span>			
<b>AUT Name</b>	Login	<b>Version</b>	1.0
<b>Iteration ID</b>	1.0	<b>Date of Test</b>	25/04/2018

<b>Test ID</b>	Blackbox2
<b>Purpose of Test</b>	To ensure that:  The user can log into the web app with their google account
<b>Test Environment</b>	The test environment is as follows: <ul style="list-style-type: none"> <li>▪ Client hardware: MacBook pro 2012</li> <li>▪ Browser: Chrome</li> </ul>
<b>Test Steps</b>	Tester should open the side menu and click sign in.  The tester should then click the log in button and enter their google account details.
<b>Expected Result</b>	The tester is logged in and user profile picture and name appear in the side menu
<b>Actual Result</b>	Test passed.  The tester was successfully logged in



<b>Black Box test 3</b> <span style="float: right;"><i>(front sheet)</i></span>			
<b>AUT Name</b>	Chart Interaction	<b>Version</b>	1.0
<b>Iteration ID</b>	1.0	<b>Date of Test</b>	25/04/2018

<b>Test ID</b>	Blackbox3
<b>Purpose of Test</b>	To ensure that:  The user can interact with the historical prices, volume and sentiment chart on the cryptocurrency overview page.
<b>Test Environment</b>	The test environment is as follows: <ul style="list-style-type: none"> <li>▪ Client hardware: MacBook pro 2012</li> <li>▪ Browser: Chrome</li> </ul>
<b>Test Steps</b>	Tester should navigate to the ethereum overview page. The tester should then click the zoom drop down and change the time span to 30 days. The tester should click the sentiment button The tester should click the BTC button
<b>Expected Result</b>	The chart should zoom out to 30-day view. The sentiment area element should appear. The BTC line element should appear
<b>Actual Result</b>	Test passed.  The chart re-zoomed to 30 days and the sentiment area element and BTC line element appeared on the chart

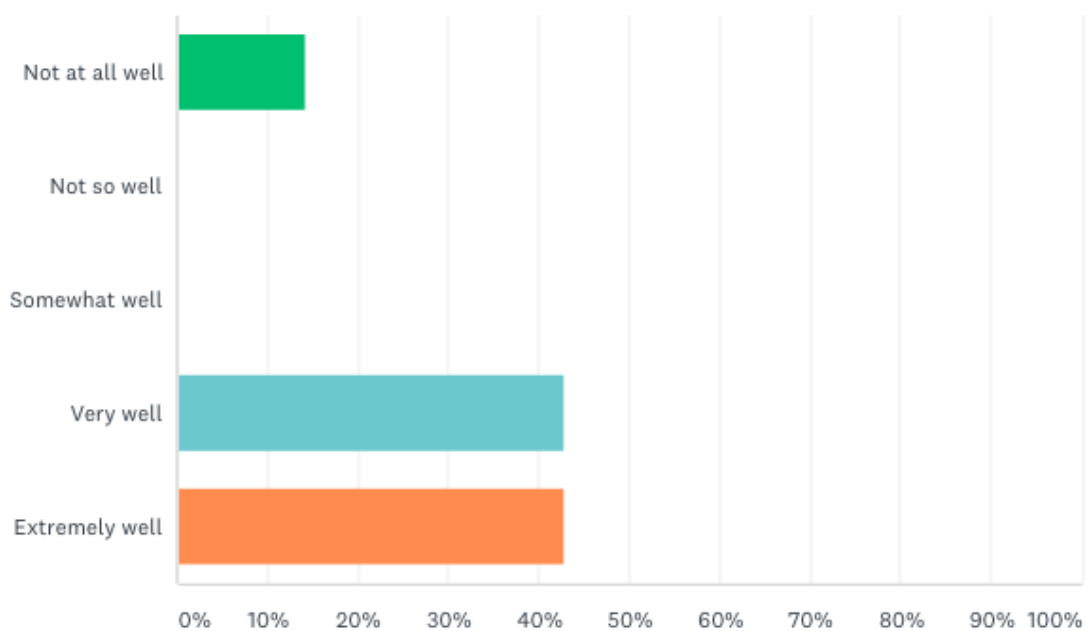
<b>Black Box test 4</b> <span style="float: right;"><i>(front sheet)</i></span>			
<b>AUT Name</b>	Words and Bigram Interaction	<b>Version</b>	1.0
<b>Iteration ID</b>	1.0	<b>Date of Test</b>	25/04/2018

<b>Test ID</b>	Blackbox4
<b>Purpose of Test</b>	To ensure that:  The user can interact with historical Word and Bigram counts by selecting a date to view.
<b>Test Environment</b>	The test environment is as follows: <ul style="list-style-type: none"> <li>▪ Client hardware: MacBook pro 2012</li> <li>▪ Browser: Chrome</li> </ul>
<b>Test Steps</b>	Tester should navigate to the ethereum overview page.  The tester should click the choose date button  The tester should choose a date
<b>Expected Result</b>	The bar charts will re-render with the word and bigram counts for the chosen date
<b>Actual Result</b>	Test passed. the bar charts re-rendered correctly

## **2.6 Customer testing**

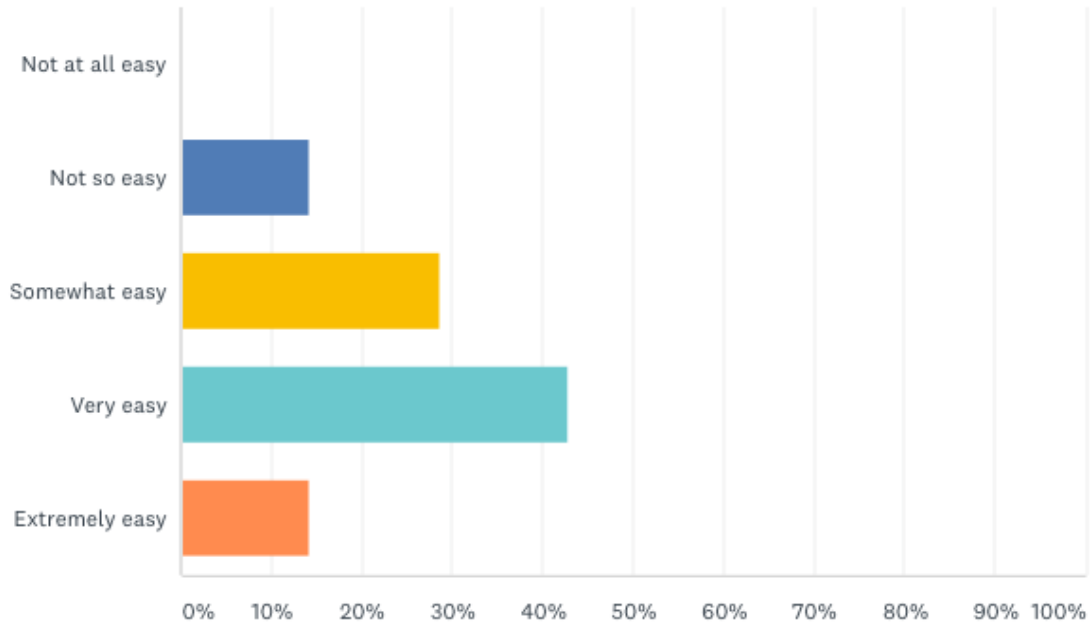
One of the last stages of a software development lifecycle is user acceptance testing (UAT). UAT involves actual users of the software testing it, in order to make sure it can handle real world tasks and fulfils all the requirements from an end user's perspective. The web app url and a survey was sent around to a number of friends in order to get their opinion on the web application and the information it provides. The survey consisted of a total of 6 questions and got 7 responses.

**Q1. Overall, how well does the website fit the needs for a cryptocurrency evaluation tool?**



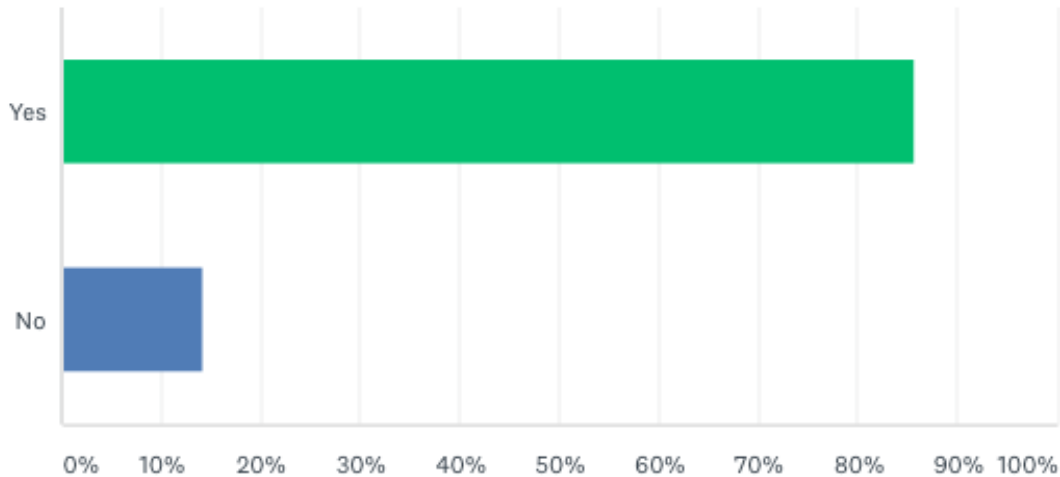
The first question was to determine how well the website might serve as an evaluation tool for cryptocurrencies. The majority of users felt the website was providing a valuable service as a cryptocurrency evaluation tool, with one outlier responding that it does not fit well.

## Q2. How easy was it to find what you were looking for?



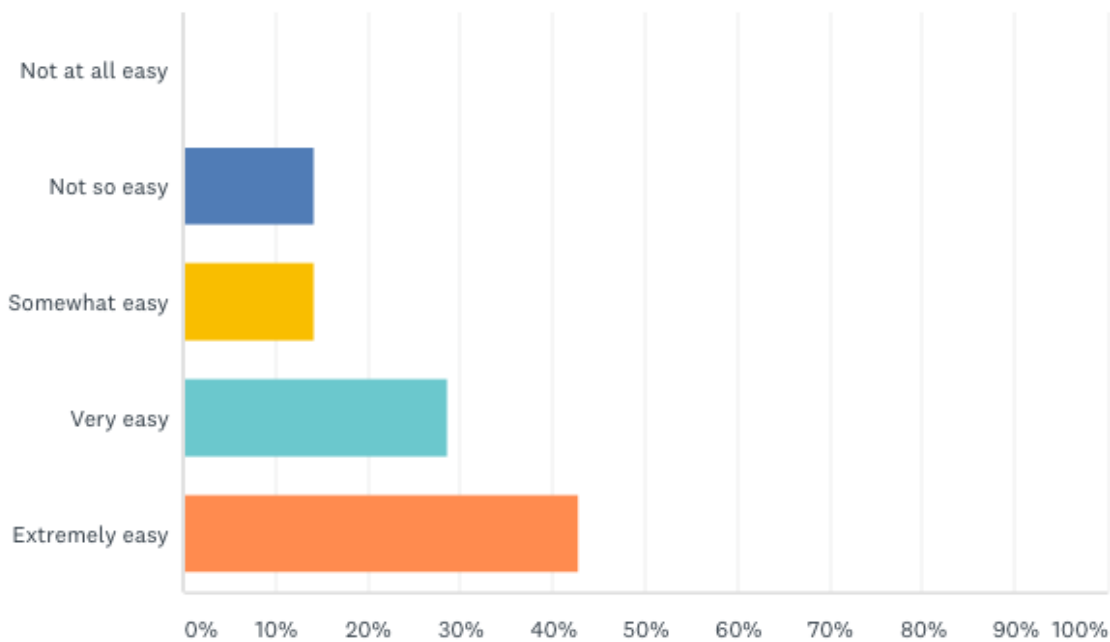
As the site contains a lot of information, it may be hard to dissect for users who do not know what they are looking at. The objective of the second question was to determine whether the information is presented in a straightforward manner. This question received some mixed results, with the majority of responses around somewhat and very easy. It's worth noting that since this survey has been sent out the web app has been updated to include more pop ups and tips for users to understand the information which is being displayed.

## Q3. Do you like the sites design?



The web applications design is quite similar to some existing cryptocurrency price aggregation sites, so the results of the above are not that surprising. Only one responder chose no, that they did not like the design of the site. This is quite positive feedback overall for the sites design.

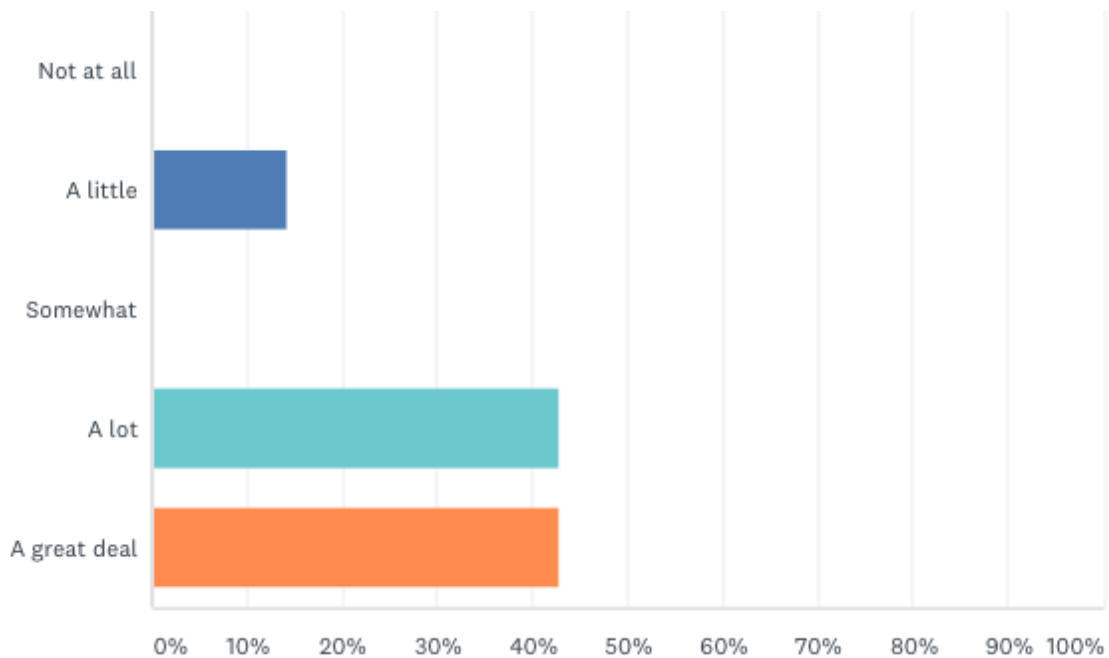
**Q4. How easy did you find it to understand the information on the site?**



This question is quite similar to question two, but is focused on the user's comprehension of the data rather than how easy it is to locate what they are looking for. The results of this question show the majority of users understand the

information. The target market for this web application is users who are interested in cryptocurrencies. Over half of the users that this survey was sent to had some if not a good bit of experience with cryptocurrencies, so the results of this are quite positive as users without a knowledge of cryptocurrencies aren't expected to fully understand the information on the site.

**Q5. Do you trust the information on the site?**



The majority voted they trust the information on this site either a lot, or a great deal. Again, there was one outlier, with one respondent voting they trusted this information a little. Throughout the survey responses there has been a trend of one responder voting more negatively than the rest. This may have been the same user who did not understand the information or find what they were looking for quite easy.

**Q6. How likely are you to recommend this site to a friend?**

Score	1	2	3	4	5	6	7	8	9	10
Chosen	-	-	-	-	1	-	2	-	-	4

The last question of the survey was to gauge how many users would recommend this site to a friend. As previously mentioned over half the respondents have experience with cryptocurrencies, and over half the respondents voted 10 on a scale of how likely they are to recommend this site to a friend. The rest of the respondents also gave quite a high score, the lowest being five.

Overall the results of the customer testing were quite positive. The survey was given to both users who are familiar with cryptocurrencies and users who had no experience with cryptocurrencies. The majority of users liked the sites design and thought it was a useful tool for evaluating cryptocurrencies. Some respondents felt the information was both difficult to understand and find. Since distributing the survey I have updated the site to add in more tips and popups to provide users with a better description of the data they are viewing.

### 3 Conclusions and Future Work

The main objectives of this project were to develop a web application providing an insight about how cryptocurrencies are perceived and how much they were talked about on social media, as well as creating predictive models using the gathered data for the web application. The results of the predictive models analysis show low predictive powers, with the KNN implementation performing best with an accuracy of 61%. The models were heavily biased towards classifying the price as going lower which may have been due to a class imbalance. The customer testing results from the web application were quite positive. Most of the respondents felt that this application is beneficial for evaluation a cryptocurrency. The cryptocurrency market is young, volatile and based on pure speculation. Although the models showed weak predictive powers, I still believe this type of site is not only interesting, but extremely beneficial to traders. In the future, or given more time, I would have liked to

incorporate some sort of AI, or predictive models into the analysis system and web application.

## 4 References

- Coinmarketcap.com. (2017). Historical Snapshots Index | CoinMarketCap. [online] Available at: <https://coinmarketcap.com/historical/> [Accessed 30 Nov. 2017].
- Kdnuggets.com. (2017). Text Mining 101: Topic Modelling. [online] Available at: <https://www.kdnuggets.com/2016/07/text-mining-101-topic-modeling.html> [Accessed 30 Nov. 2017].
- Staff, I. (2003). Technical Analysis. [online] Investopedia. Available at: <https://www.investopedia.com/terms/t/technicalanalysis.asp> [Accessed 30 Nov. 2017].
- Robinson, J. (2017). Text Mining with R. [online] Tidytextmining.com. Available at: <http://tidytextmining.com/sentiment.html> [Accessed 1 Dec. 2017].
- Robinson, J. (2017). Text Mining with R. [online] Tidytextmining.com. Available at: <http://tidytextmining.com/topicmodeling.html> [Accessed 1 Dec. 2017].
- Python, I., Python, I. and Srivastava, T. (2018). *Introduction to KNN, K-Nearest Neighbors : Simplified*. [online] Analytics Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2014/10/introduction-k-neighbours-algorithm-clustering/> [Accessed 13 May 2018].
- Python, I., Python, I. and Srivastava, T. (2018). *Introduction to KNN, K-Nearest Neighbors : Simplified*. [online] Analytics Vidhya. Available at:



<https://www.analyticsvidhya.com/blog/2014/10/introduction-k-neighbours-algorithm-clustering/> [Accessed 13 May 2018].

- Python), I., Python), I. and Srivastava, T. (2018). *Introduction to KNN, K-Nearest Neighbors : Simplified*. [online] Analytics Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2014/10/introduction-k-neighbours-algorithm-clustering/> [Accessed 13 May 2018].
- 

## 5 Appendix

### 5.1 Project Proposal

#### 5.1.1 Objectives

**Objective 1:** Gather Datasets - The platforms this project will focus on are reddit.com, twitter.com and bitcointalk.org. Both reddit and twitter offer API's with an abundance of data, but for the purpose of this project only a subset of relevant information will be required. Bitcointalk.org does not offer an API so this project will make use web scraping technologies to extract the data from the web pages. Historical prices for the top 100 cryptocurrencies will be pulled from coinmarketcap.com's public API.

**Objective 2:** Scrub and Standardise Dataset - As three different platforms will be used the datasets will slightly differ in format. In order to perform the analysis without any hiccups it is good practice to scrub the dataset of any unwanted

characters such as whitespace, punctuations, indentations. The three datasets will also be standardised into a uniform format.

**Objective 3:** Data Classification - By applying sentiment analysis algorithms to the comments they will be classified by either positive, neutral or negative sentiment.

**Objective 4:** Data Analysis - After scrubbing and classifying the dataset statistical analysis will be performed on both prices and comments with the aim of discovering trends or correlations between the two.

**Objective 5:** Visualise Data - Create a visualisation of the research results and display them using tableau for interpretation by the end user.

**Objective 6:** Web Application – After the analysis has been performed the results and dataset should be accessible via a web application. Users will be required to register to view the data and will be able to interact with and interpret the results of the analysis.

### **5.1.2 Background**

I first heard about bitcoin in 2012 and made my first purchase in 2013. At the time, I had very little understanding of how the underlying technology that powered bitcoin, the blockchain, actually worked and as far as I was aware bitcoin was the only cryptocurrency. Apart from checking the bitcoin price occasionally I had not considered the cryptocurrency space much further for a few years.

At the beginning of 2017 I came across an article detailing a new currency called ethereum that promised to dethrone bitcoin. Unlike bitcoin, ethereum isn't just another digital currency, ethereum is a decentralised platform that runs applications on top of the blockchain, opening the blockchain space to an array of new use cases. One of the most important features of the ethereum blockchain is the ability to create tokens. These tokens are treated the same as digital currency and can be used in the decentralised applications they are associated with as well as traded for other digital currencies. With the discovery of ethereum I began to explore the cryptocurrency space a little further and to my pleasant

surprise there was more than just bitcoin and ethereum. Before I knew it, I was a frequent visitor of various cryptocurrency subreddits and forums.

Unlike traditional stocks, cryptocurrencies have no tangible value attached and prices fluctuate with perceived value. Cryptocurrency trading relies on speculation more so than fundamental analysis and as a result most of the price speculation regarding cryptocurrencies takes place online. Around May 2017 the cryptocurrency market seen an explosion in interest. As a result, thousands of new users came flooding into the cryptocurrency forums searching for the next best coin to invest in. Most of the cryptocurrencies or tokens being recommended had great potential for making a real-world impact but at the time had no product, or development was a few years away from completion. Despite a lot of the cryptocurrencies having no product or being half finished they began to increase in price anyway. With the influx of new users came a lot of inexperienced traders putting their money where ever the consensus was. The cryptocurrency market is still quite young so there is a lot of volatility. Regulations are scarce and price manipulation from pump and dumps or traders with a large bankroll is a lot easier than traditional stock markets. Compared to the stock market it does not take a lot of buying or selling power to change the price and with the arrival of novice traders that invest in cryptocurrencies solely because they see them recommended a lot, I believe it may be possible to find a correlation between the sentiment of posts on the various platforms and the prices of the mentioned cryptocurrencies.

### **5.1.3 Technical Approach**

Previous projects I have completed have been focused around software development more so than data analysis. As this is the first project of its kind for me I will begin by researching various techniques within data analysis such as dataset cleansing and text analysis tools. For my dataset, I will be gathering user posts and I have chosen 3 sources; bitcointalk.org, reddit.com, twitter.com. I chose these sites as they are the main hubs for online cryptocurrency discussion. Reddit and twitter offer an API which is very helpful. I will not need every piece of

information they have to offer on a post and to avoid overabundance of unnecessary data I will only gather relevant information. Bitcointalk does not offer an API unfortunately, but this can be circumvented with the use of Python's third-party library Selenium which will allow me to directly query html on Bitcointalk and extract necessary data. To make a comparison between the sentiment of posts and cryptocurrency prices I will need historical and current price data. Luckily this data is easily accessible from many public API's. I chose coinmarketcap.com for the historical and current prices as they are reliable and have a proven track record within the cryptocurrency sphere. Their API is public, requiring no key, and provides helpful data other than price, such as volume and price percent changes over set periods of time. Python will be my language of choice for gathering the dataset and I will gather all relevant posts on the various platforms from the last 12 months. Before performing an analysis on the dataset, it will first need to be cleaned. I will begin by merging my three data sets into one, making sure they are all represented in a consistent format, then identify any missing data and standardising the dataset. To cleanse the data even further I will remove any duplicate data as well as removing any punctuations and unnecessary whitespace. Having scrubbed the dataset, I will begin looking for patterns and performing text analysis on the posts using R. More specifically, applying sentiment analysis to the individual comments using RSentiment. Finally, I will interpret the results and create appropriate visualizations to represent it.

#### **5.1.4 Technical Details**

For the purpose of this project I will be using python as my main scripting language. Python is an object oriented, high level language with dynamic typing and binding, making it very useful for rapid development. I will be using python to scrape posts and comments from the various platforms. With regards to twitter.com, coinmarketcap.com and reddit.com I will be getting most, if not all the data, through their API's. Python has a built-in requests module which makes the process of sending GET requests to the API's all that easier. Coinmarketcap API

is quite simple and with few endpoints, so I will be querying their API directly with the requests library. Twitter and reddit have quite a lot of resources on their API and can often be tricky to navigate. To simplify the approach, I will be using two open source python wrappers found on GitHub for the respective API's. For reddit I will be using *PRAW* (Python Reddit API Wrapper), and for twitter I will be using *python-twitter*. Both these wrappers will help speed up the development process. Bitcointalk.org, notably one of the largest forums for cryptocurrency discussion has no API, to tackle this issue I will write a custom script using a third-party python library called BeautifulSoup. BeautifulSoup, with the help of the requests module, will allow me to extract data from bitcointalk by querying html elements directly. All comments and posts along with any other metadata extracted will be saved in comma separated format (CSV). For my analysis, I will be using R to manipulate and visualise the dataset. R includes a built-in library called RSentiment which will allow me to perform sentiment analysis on the posts I have gathered.

#### **5.1.5 Evaluation**

I will be running various unit tests on all scripts and API connections to verify the quality of the code. Datasets will be checked for credibility as they are being scraped and pulled. Results of the analysis will be displayed using visualisation techniques to allow the end users to interpret the results.

## 5.2 Project Plan

<b>Project Proposal</b>	<b>27 days</b>	<b>Thu 21/09/17</b>	<b>Fri 27/10/17</b>
Brainstorming	10 days	Thu 21/09/17	Wed 04/10/17
Journal Entry	1 day	Thu 05/10/17	Thu 05/10/17
Project pitch	1 day	Fri 06/10/17	Fri 06/10/17
Project proposal document	15 days	Mon 09/10/17	Fri 27/10/17
<b>Requirements Specification</b>	<b>19 days</b>	<b>Wed 01/11/17</b>	<b>Fri 24/11/17</b>
Identify target platforms	6 days	Wed 01/11/17	Wed 08/11/17
Research strategy	2 days	Thu 09/11/17	Fri 10/11/17
Journal Entry	1 day	Fri 10/11/17	Fri 10/11/17
Requirements Spec Document	11 days	Mon 13/11/17	Fri 24/11/17
<b>Mid point presentation</b>	<b>12 days</b>	<b>Sat 25/11/17</b>	<b>Fri 08/12/17</b>
Write Reddit.com script	2 days	Sat 25/11/17	Sun 26/11/17
Test Script	1 day	Mon 27/11/17	Mon 27/11/17
Gather sample data	1 day	Tue 28/11/17	Tue 28/11/17
Clean data	1 day	Wed 29/11/17	Wed 29/11/17
Project prototype	6 days	Thu 30/11/17	Thu 07/12/17
Journal Entry	1 day	Fri 08/12/17	Fri 08/12/17
<b>Build Scripts</b>	<b>45 days</b>	<b>Sat 09/12/17</b>	<b>Wed 07/02/18</b>
Identify required data from API	2 days	Sat 09/12/17	Mon 11/12/17
Write reddit script	3 days	Tue 12/12/17	Thu 14/12/17
Write bitcointalk script	7 days	Fri 15/12/17	Fri 22/12/17
Christmas Break	8 days	Sat 23/12/17	Tue 02/01/18
Exams	4 days	Wed 03/01/18	Mon 08/01/18
Write twitter script	5 days	Tue 09/01/18	Mon 15/01/18
Write coinmarketcap script	2 days	Tue 16/01/18	Wed 17/01/18
<b>Test Scripts</b>	<b>5 days</b>	<b>Thu 18/01/18</b>	<b>Wed 24/01/18</b>
Test coinmarket script	1 day	Thu 18/01/18	Thu 18/01/18
Test bitcointalk script	2 days	Fri 19/01/18	Mon 22/01/18
Test twitter script	1 day	Tue 23/01/18	Tue 23/01/18
Test reddit script	1 day	Wed 24/01/18	Wed 24/01/18
<b>Data Mining</b>	<b>28 days</b>	<b>Thu 25/01/18</b>	<b>Mon 05/03/18</b>
Run Scripts to gather datasets	5 days	Sat 25/11/17	Wed 29/11/17
Scrub and standardise Dataset	5 days	Tue 30/01/18	Mon 05/02/18
Begin data mining, run different algorithms on the data and look for patterns	15 days	Wed 31/01/18	Tue 20/02/18
Create Visualisation of data	7 days	Wed 21/02/18	Thu 01/03/18
<b>Testing</b>	<b>3 days</b>	<b>Fri 02/03/18</b>	<b>Tue 06/03/18</b>
Testing	3 days	Fri 02/03/18	Tue 06/03/18
<b>Technical Report</b>	<b>44 days</b>	<b>Wed 07/03/18</b>	<b>Sat 05/05/18</b>
Document	44 days	Wed 07/03/18	Sat 05/05/18

## **5.3 Monthly Journals**

### **5.3.1 September**

Student name: Dylan Kilkenny

Programme (e.g., BSc in Computing):

Month: September

#### **My Achievements**

This Month we began back at college. We were prepped for our pitch in Eamon's class and he gave us some very helpful advice regarding our projects and our pitch. Luckily, I have been thinking about my project idea since I met with Simon Caton last May while on work placement. I decided to go with a project related to my stream, data analytics, rather than build something. Thankfully it was approved.

#### **My Reflection**

My project revolves around an analysis of forum comments and news article related to cryptocurrency's, with the hope of finding a correlation between the sentiment of these comments/articles and the price movements of motioned currency's. I got some positive feedback within the pitch and it was suggested I place my focus on digital currency's that aren't bitcoin as they aren't studied as much. Overall I am happy with my choice of project and look forward to beginning.

#### **Supervisor Meetings**

No meeting held .

### **5.3.2 October**

Student name: Dylan Kilkenny

Programme : BSHC in Computing

Month: October

### **My Achievements**

This month I began to think about my project a bit more and what is needed to perform the analysis. I decided to focus my attention on three platforms instead of just one as it will give a lot more variety to the dataset. I choose bitcointalk.org, reddit.com and twitter.com as they are the 3 biggest platforms for cryptocurrency discussion. I had a lot of project work this month as well as multiple CA's so I was beginning to feel the pressure of fourth year already. The project proposal for software project was due at the end of the month also so the last two weeks consisted of researching the strategy needed to implement my project idea.

### **My Reflection**

I feel as though I need to manage my time better as towards the end of the month the assignments started to pile up and I felt as though I was rushing to complete them.

### **Intended Changes**

Next month I will be focusing more on my software project. I will gather a sample dataset from one of the platforms and prepare a prototype for December.

### **Supervisor Meetings**

I had a meeting with Adriana Chis on 26<sup>th</sup> October where we discussed some details about my project. Adriana also explained the format of our meetings will be group meetings and they will be held twice weekly.

### **5.3.3 November**

Student name: Dylan Kilkenny

Programme : BSHC in Computing

Month: November



### **My Achievements**

This month I began writing the python scripts to collect the dataset. We have a prototype due for the first week of November and I will be gathering a test dataset from reddit to work on. I also have begun researching sentiment analysis techniques as this is the core focus of my project.

### **My Reflection**

This month the pace is starting to pick up and I am really beginning to feel the 4<sup>th</sup> year work load. My time management needs some improvement.

### **Intended Changes**

Work on time management.

### **Supervisor Meetings**

Absent from meeting

### ***5.3.4 January***

Student name: Dylan Kilkenny

Programme : BSHC in Computing

Month: January

### **My Achievements**

After a break for Christmas, and focusing on my exams, this month I will begin performing some analysis on the data which I have gathered. I will also begin researching the best methods for deploying a full automated analysis system to the cloud, in order to have real time social media sentiment and volume stats for the web application.

### **My Reflection**

I feel the initial data collection stage was a success.

### **Intended Changes**

Improve automation of the system

### **Supervisor Meetings**

No meeting.

### **5.3.5 February**

Student name: Dylan Kilkenny

Programme : BSHC in Computing

Month: February

### **My Achievements**

This month I did the following:

- Began porting existing R code to python
- Created an Ubuntu server for the system to run on
- Made a reactJS prototype

### **My Reflection**

The ReactJS web application needs a lot of work. It is a much steeper learning curve than I originally anticipated.

### **Intended Changes**

Next month I will focus more so on the react application

## **Supervisor Meetings**

No meeting.

### **5.3.6 March**

Student name: Dylan Kilkenny

Programme : BSHC in Computing

Month: March

## **My Achievements**

This month I did the following:

- Fine tuning the automated data gathering and analysis system
- Added improvements to the react web application
- Developed a nodejs server to be the link between the database and the web application
- Worked on implementing the analysis system with a MongoDB back end

## **My Reflection**

The MongoDB backend is very fast and scalable, but has a lot of caveats. The max document size is only 16mb, which has been causing problems with the data consistency.

## **Intended Changes**

Next month I will focus on fine tuning the analysis system with mongoDB

## **Supervisor Meetings**

I had a meeting with Adriana Chis on 8<sup>th</sup> march where I showed her what I have been working on so far. She suggested I begin looking into testing methodologies as it makes up 10% of the marks for the project.

