Dyson School of Design Engineering | MEng Design Engineering

**Module Exam**

|  |  |
| --- | --- |
| Module code and Name | DE4-SIOT Sensing & IoT |
| Student CID | 01098324 |
|  |  |
| Assessment date | **10th Jan 2019** |
|  |  |

**Presentation URL (publicly accessible link):**

**Code & Data (publicly accessible link):**

# Coursework 1: Sensing

## Introduction and Objectives

The cryptocurrency market has boomed in the past 5 years, with a total market capital currently standing at $130 Bn [1] . According to the random walk hypothesis, predicting how the price of a particular stock will change cannot be done precisely based on its price history [2]. However, there have however been studies that suggest that data available online can help identify future changes in the price of stocks in the stock market, one example being a study of how Google Trends influences the DJIA [3]. There is a suggestion that this data about wide public opinion and interest can give an indication of what trading decisions will be made. Another growing source of online data are social media websites such as Twitter and Facebook. ‘Social sensing’ as it is called is the analysis of social media data for use in understanding real life events and can be used to ‘extract knowledge about upcoming events’ [4]

The objective of this project was to acquire data on the price of the top 6 cryptocurrencies, alongside data on the number of times the name of those cryptocurrencies is tweeted in quasi-real time. The objective was to analyse the data to help gain an understanding of the correlation between the datasets, and to generate a rudimentary level of prediction. The data and it’s prediction were to be displayed as an interface to inform and alert a potential user on increased chances of a shift in the cryptocurrency market.

## Data Sources and Sensing Set-Up

All the code necessary for sensing was written in Python 3. No hardware setup was required for sensing as both data streams were available as APIs.

Cryptocurrency prices were acquired via CoinMarketCap’s API (info available at: [coinmarketcap.com/api/](https://coinmarketcap.com/api/)). CoinMarketCap (CMC) is a popular cryptocurrency listings website. After signing up for a free account, an API key assigned to the account was provided and was stored locally inside a .json file.

In order to handle calls to the CMC API, the python library ‘Requests’ was used, a get request was made using the following statement:

coin\_data = requests.get('api'+'key').json()

This returned a json object that contained a variety of different data and information about the current cryptocurrency market. This json object could then be indexed to acquire the current price of the top 6 ‘coins’ – coin referring to a singular cryptocurrency:

coin['data']['quote']['USD']['price']

The top 6 coins by market cap at the start of sensing were; Bitcoin, Ethereum, XRP, Bitcoin Cash, EOS and Stellar.

Tweet data was sourced through Twitter’s API also - the Requests library could have been used again, however it was found that the process could be made more convenient through a Python library called Tweepy (info available at: [www.tweepy.org/](http://www.tweepy.org/)). The decision was made to stream tweets in real time rather than accessing historical tweets retrospectively as this seemed to be more in-line with the sensing approach. Having signed up for a Twitter dev account, a key and token were provided and stored locally. A twitter stream was then created and filtered according to the names of the coins:

stream = tweepy.Stream(auth=api.auth, listener=listener)

stream.filter(track=columns, is\_async=True)

The API was not case sensitive when it came to filtering, and it also returned hashtags of the filter’s keywords e.g. ‘bitcoin’ as a filter allowed through all tweets being tweeted that contained the term ‘bitcoin’, ‘#Bitcoin’, ‘BITCOIN ’etc. The live incoming tweets were ‘listened’ to by a custom StreamListener object that inherited from the tweepy.StreamListener class:

|  |  |
| --- | --- |
| class StreamListener(tweepy.StreamListener):  def on\_connect(self):  self.connected = True  def on\_disconnect(self):  self.connected = False  def on\_status(self, status):  CountTweets(status.text) | def CountTweets(tweet):  for word in tweet.split():  word.lower().replace(" ", "")  if word.startswith('#'):  word = word[1:]  if (word in list(twitter\_df.columns.values)):  count\_dict[word] += 1 |

Upon the arrival of a new tweet, the on\_status method was invoked which then called the CountTweets function. This function scanned through the tweet and counted how many times each of the coin names were mentioned before adding this count to a dictionary. This meant that if a coin name was featured *n* times in a single tweet, it would be counted as *n* and not as 1. Importantly, a connected Boolean property was featured in the listener class. This Boolean would be set False if the stream disconnected, so that during runtime this problem could be flagged and the stream reopened:

if not listener.connected:

print('Twitter stream dropped... Trying to reconnect...')

time.sleep(2)

stream.filter(track=columns, is\_async=True)

During development it was found that Twitter would occasionally disconnect the stream, this code circumvented that issue.

## Data Collection and Storage Process

Given that the code had to run continuously for several days, the decision was made to run it on a cloud based server. After some research into server providers, Microsoft’s Azure platform was chosen to both run the code and store the data. Azure’s ‘WebJob’ service suited the task of running the code well and required the upload of a zip file containing the sensing script alongside a batch file to run it.

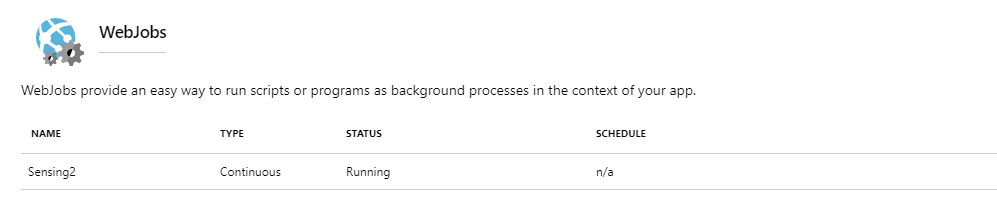


Figure 1 Microsoft Azure's WebJob Service – shown running the Sensing script

Data was sampled at a regular time interval of **4 mins 30 s**. This interval was determined by the limiting factor of allowed number of requests to the CMC API, which was set at 333 req/day for a free account. This time interval was further justified by the fact that; it was longer than the execution time for the request and store process of the script ( ≈ 6 seconds on my computer) and that across 7 days this sampling time would result in:

…which was well below the 10GB storage allocation on Azure’s basic plan.

The data was initially stored in 2 Pandas dataframes, each with a DateTime index, one for the CMC price data and one for the tweet data. Here the data is written to the coin\_df dataframe:

timestamp = pd.Timestamp('now')

for coin in coin\_data['data']:

if coin['name'] in coin\_names:

coin\_df.loc[timestamp, coin['name']] = coin['quote']['USD']['price']

Table 1 First 5 Rows of Cryptocurrency Price Dataset

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **bitcoin** | **ethereum** | **xrp** | **bitcoincash** | **eos** | **stellar** |
| 05/01/2019 16:54 | 3868.129 | 156.7199 | 0.358436 | 162.2812 | 2.727974 | 0.114812 |
| 05/01/2019 16:58 | 3872.382 | 156.7598 | 0.357999 | 162.1221 | 2.73014 | 0.114731 |
| 05/01/2019 17:03 | 3868.375 | 156.8905 | 0.357688 | 162.1638 | 2.730436 | 0.114631 |
| 05/01/2019 17:07 | 3871.104 | 156.8907 | 0.357932 | 162.5431 | 2.737082 | 0.114595 |
| 05/01/2019 17:12 | 3871.428 | 157.1449 | 0.358217 | 162.8878 | 2.740771 | 0.114694 |

The dataframes were then converted to .csv files so that they could be saved. An Azure blob storage container was used to save the .csv files and was interfaced with using the Azure Storage Python library.

coin\_df.to\_csv('CMC\_data.csv')

block\_blob\_service.create\_blob\_from\_path(

creds['azure']['container'],

'CMC\_data.csv',

'CMC\_data.csv',

content\_settings=ContentSettings(content\_type='application/CSV')

)

## Basic Characteristics of the Systems Set Up and Data

# Coursework 2: Internet of Things

## Data Interaction/visualisation/actuation platform

## Data analytics, Inferences and Insights

Visualising the distribution of the number tweets made per sampling time confirms central limit theorem as each coin can be seen to have an approximately normal distribution:

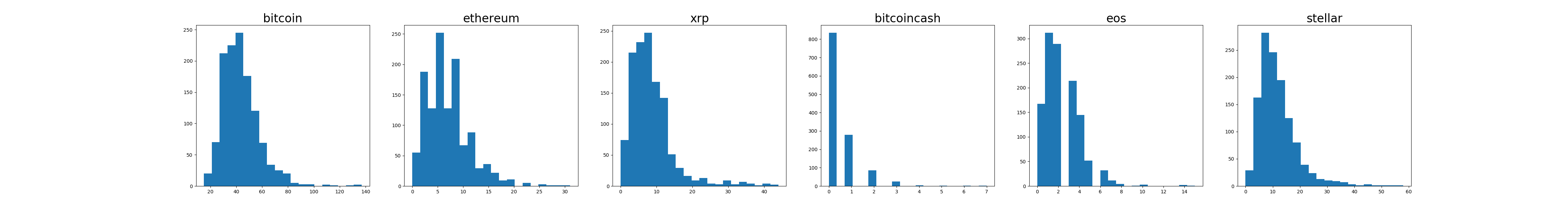


Figure 2 Histograms showing distribution of number of times cryptocurrency name is tweeted per sampling time

Plotting the raw bitcoin data alone, without processing, results in the following:

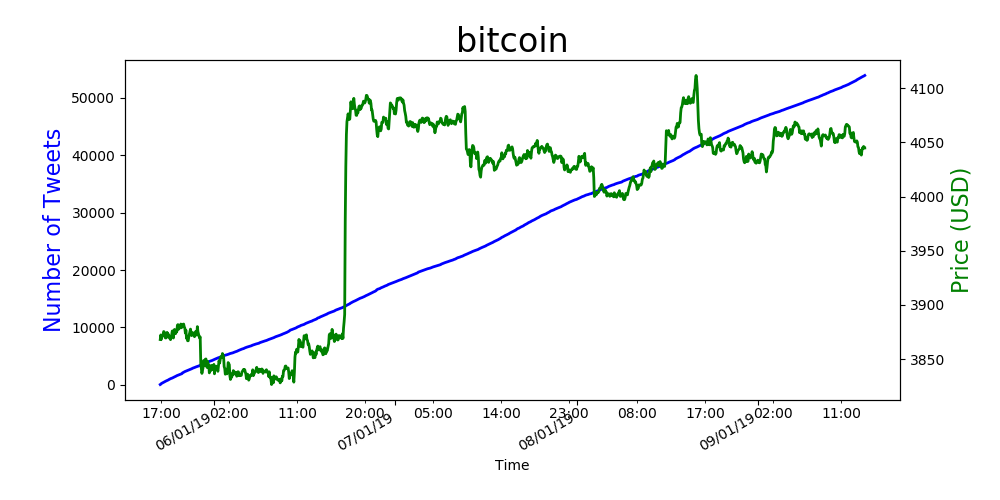
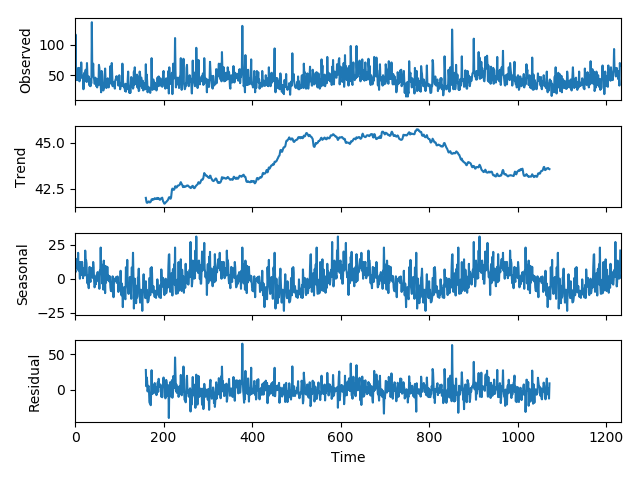


Figure 3 Raw Bitcoin price and cumulative tweet data, plotted against time

The number of tweets rises steadily in Figure 3, due to the fact that the data was logged cumulatively. It can be seen that the price of bitcoin jumped up by a lot on the evening of 06/01/19, relative to the changes in the rest of the series. No clear insights about the relationship between the two signals can be made, however, differentiating the number of tweets and then normalising both the series yields in a better visual comparison:

It is difficult to see a correlation between the two series, however there is a fairly clear seasonality pattern shown in the tweet data, with a frequency of 1 day. To confirm this, a seasonality decomposition was produced:

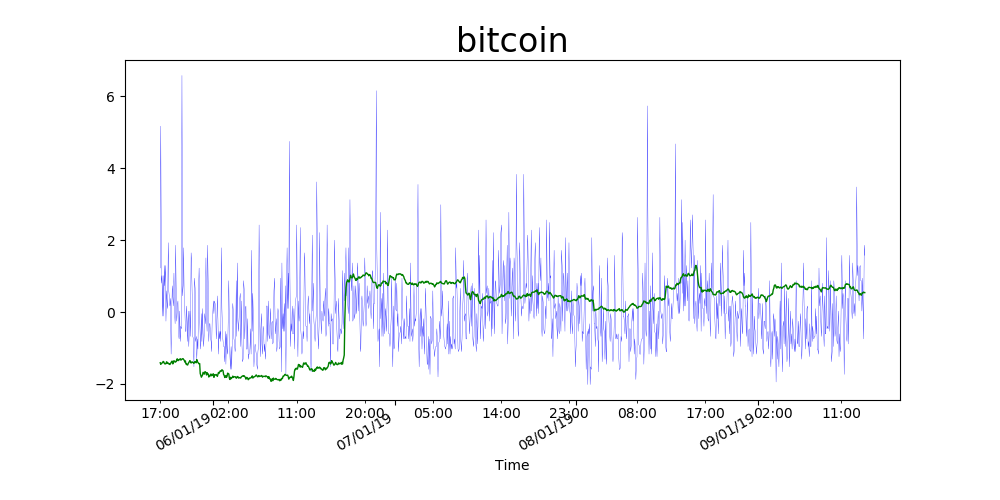


Figure 4 Normalised Bitcoin price and tweet data, against time

## Discussions on the important aspects of the project

## Avenues for future work and potential impact

# References

|  |  |
| --- | --- |
| [1] | CoinMarketCap, "Top 100 Cryptocurrencies by Market Capitalisation," 2019. [Online]. Available: https://coinmarketcap.com/. [Accessed 08 January 2019]. |
| [2] | Investopedia, "Random Walk Theory," 2018. [Online]. Available: https://www.investopedia.com/terms/r/randomwalktheory.asp. [Accessed 8 January 2019]. |
| [3] | T. Preis, H. S. Moat and H. E. Stanley, "Quantifying Trading Behaviour in Financial Markets Using Google Trends," *Scientific Reports,* vol. 3, no. 1684 (2013), 2013. |
| [4] | SoBigData, "Social Sensing," [Online]. Available: http://www.sobigdata.eu/research/social-sensing. [Accessed 08 January 2019]. |