Institiúid Teicneolaíochta Cheatharlach

INSTITUTE *of* TECHNOLOGY CARLOW

At the Heart of South Leinster

Computer Games Development CW208
Technical Design Document
Year IV

Josh Tyrrell Browne
C00224660

2020

Section 1 - List of Features Captured from Jira

** The Jira board can be found at:
https://joshbrowne.atlassian.net/secure/RapidBoard.jspa?rapidView=1&projectKey=FYP&atlOrigin=eyJpIjoiODNlZGMwNjk2MDg0NDJhZWI3ZTE2MmVmODM4ZDg5NzMiLCJwIjoiaiJ9

1.1 The following is the list of features based on what is documented on Jira:

Feature 1) Research Cryptocurrency ()

- Task 1) Investigate what exactly it is and how it works
- Task 2) Investigate why this market is so volatile. What are the impacting factors?
- Task 3) What data are people involved in this space looking at when making trades? How do they identify a price trend?
- Task 4) Investigate how much success has been found with sentiment analysis data used in machine learning implementations for cryptocurrency price predicting in the past.

Feature 2) Research Machine Learning Technologies, particularly Recurrent Neural Networks

- Task 1) Investigate the technology and its use cases from previous implementations.
- Task 2) Compare the Tensorflow and PyTorch machine learning technologies to find most suitable.
- Task 3) Investigate methods that have been used in the past to try to make financial predictions.
- Task 4) Follow examples found online to get familiar with Tensorflow models and how data is prepared/fed & how the models are used.

Feature 3) Research Sentiment Analysis

- Task 1) Investigate where it has been used before, particularly related to finance.
- Task 2) Investigate the success rate with previous use cases.
- Task 3) Investigate what technologies are available for me to use.
- Task 4) Find out what is typically involved in the process? data preprocessing techniques to yield the best results.

Feature 4) Acquire Twitter Data

- Task 1) Research the Twitter API (What are the capabilities & limitations)
- Task 2) Research Tweepy, which is a Python library for using the Twitter API
- Task 3) Implement Tweepy calls to retrieve Twitter data
- Task 4) Save retrieved data to CSV files.

Feature 5) Acquire Crypto Price Data

- Task 1) Research API's that can be used to retrieve crypto data
- Task 2) Get familiar with what data can be retrieved and what the data points relate to
- Task 3) Use the Binance API to retrieve some crypto data

Task 4) Save retrieved data to CSV files.

Feature 6) Compare TextBlob & NLTK libraries for use of sentiment analysis.
- Task 1) Discover what are the limitations
- Task 2) Discover what formats can the input text be? String? list of words/strings?
- Task 3) Which best suits my project and why

Feature 7) Build out Website
- Task 1) Create HTML pages (Welcome, Login, Logout etc.)
- Task 2) Create logic for signing in, creating a session
- Task 3) Display Graphs on html page using matplotlib

Feature 8) Preparing Data For ML Model Training
- Task 1) Get the Price data and format it into a CSV with only the columns I need
- Task 2) Step through the columns of this CSV and perform tweet search requests for tweets regarding the focused crypto posted in the datetime interval of the current row's datetime and the next row's
- Task 3) Preprocess the tweets (it's important to clean and preprocess textual data before performing sentiment analysis)
- Task 4) Perform Sentiment Analysis on each (cleaned)tweet separately and record the sentiment value
- Task 5) Find the average sentiment value over all the tweets retrieved from a specific datetime, then add this value to the CSV row with matching datetime
- Task 6) Create a 'Future' column, this is the known currency price from a time distance in the future.
- Task 7) Create a 'Target' column, this value is 1 if future > current and 0 if not

Feature 9) Build the Recurrent Neural Network model

Section 2 - Schedule

The schedule for the development of this project is shown below:

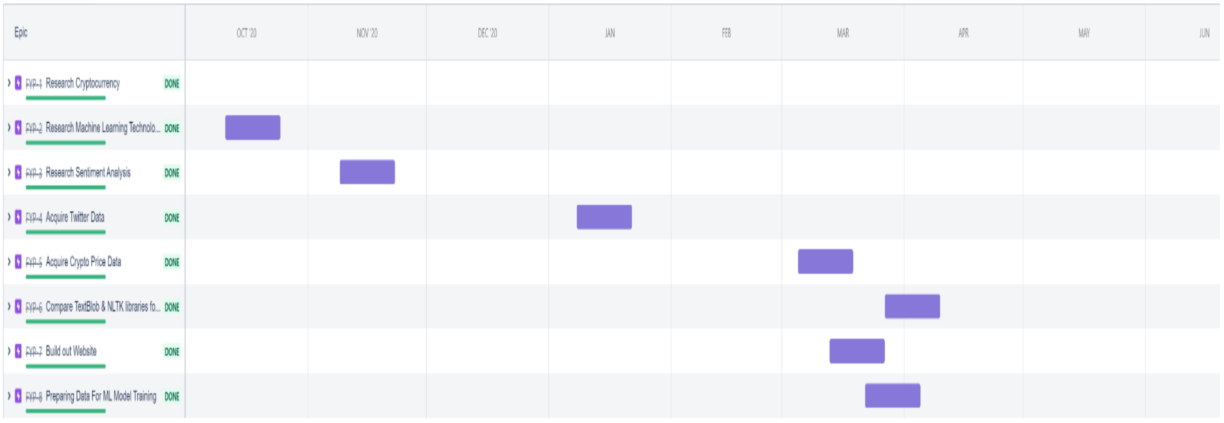


Fig 1. Roadmap (Taken from Jira)

The image shown above is taken from my Jira roadmap. Here we can see the *Epic* features listed, these features are the high level key components that were done for this project. Although these features were worked on continuously throughout the duration of the project development, the bulk of the work involved in each feature was done at the times represented on the monthly calendar shown in Fig 1.

Each of the epic features shown here contain sub-tasks, these are the breakdown of the feature into smaller pieces of work. As previously mentioned, the majority of work for these subtasks were done at the time represented on the calendar and these subtasks were then later revisited for optimisation, cleanup and refactoring purposes.

I found using Jira as a medium for planning extremely helpful. During the development, it was very common to have changes occur and the flexibility of Jira meant altering or deleting and re-making any tickets was easily done. The drag and drop feature Jira offers is another reason why I chose it, I found maneuvering features around my time schedule insightful and it certainly helped me manage my time.

Section 3 - High-level Diagrams to Illustrate Software Design

3.1 Vision Diagram:

Before we dive into a technical breakdown of the software architecture, I would like to present the 'vision' of what my software will try to interpret in theory.

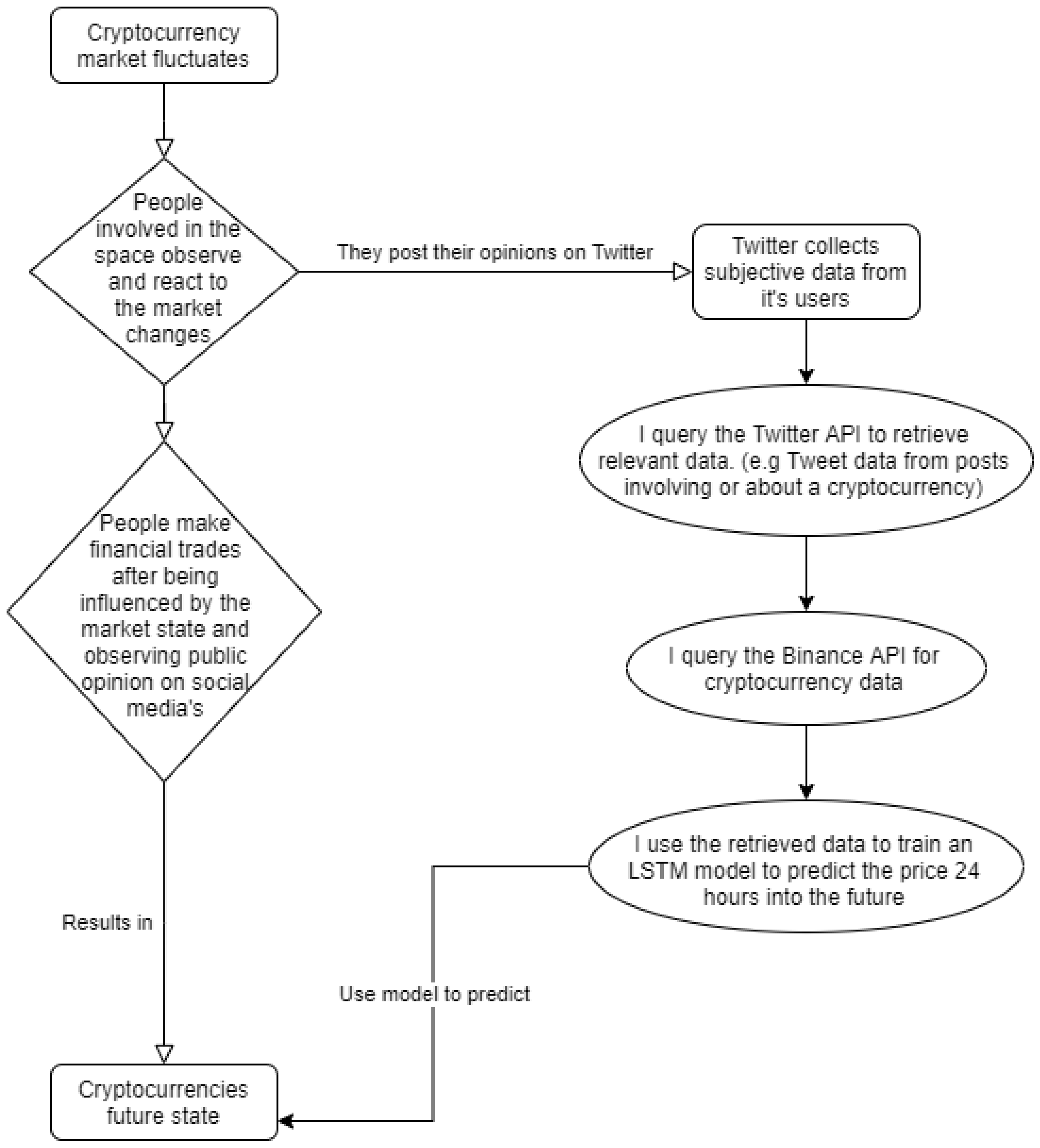

Cryptocurrency market fluctuates
People involved in the space observe and react to the market changes
They post their opinions on Twitter
Twitter collects subjective data from it's users
I query the Twitter API to retrieve relevant data. (e.g Tweet data from posts involving or about a cryptocurrency)
People make financial trades after being influenced by the market state and observing public opinion on social media's
I query the Binance API for cryptocurrency data
I use the retrieved data to train an LSTM model to predict the price 24 hours into the future
Results in
Use model to predict
Cryptocurrencies future state

Fig 2. Vision Diagram

The above is a flow chart detailing the theory of what I am essentially trying to accomplish with my LSTM model. It is in essence an investigation to see if online posts from the public about a cryptocurrency, contain sentiment trends that relate to the price trends of that currency.

The method for this investigation is fundamentally implementing sentiment analysis using currently available tools on tweets containing a cryptocurrency tag, and see if that can give consistent data. If using this data in an LSTM model can give accurate predictions, it could be a sign that public online opinions can be used for financial insight in the cryptocurrency market.

3.2 Technology Diagram:

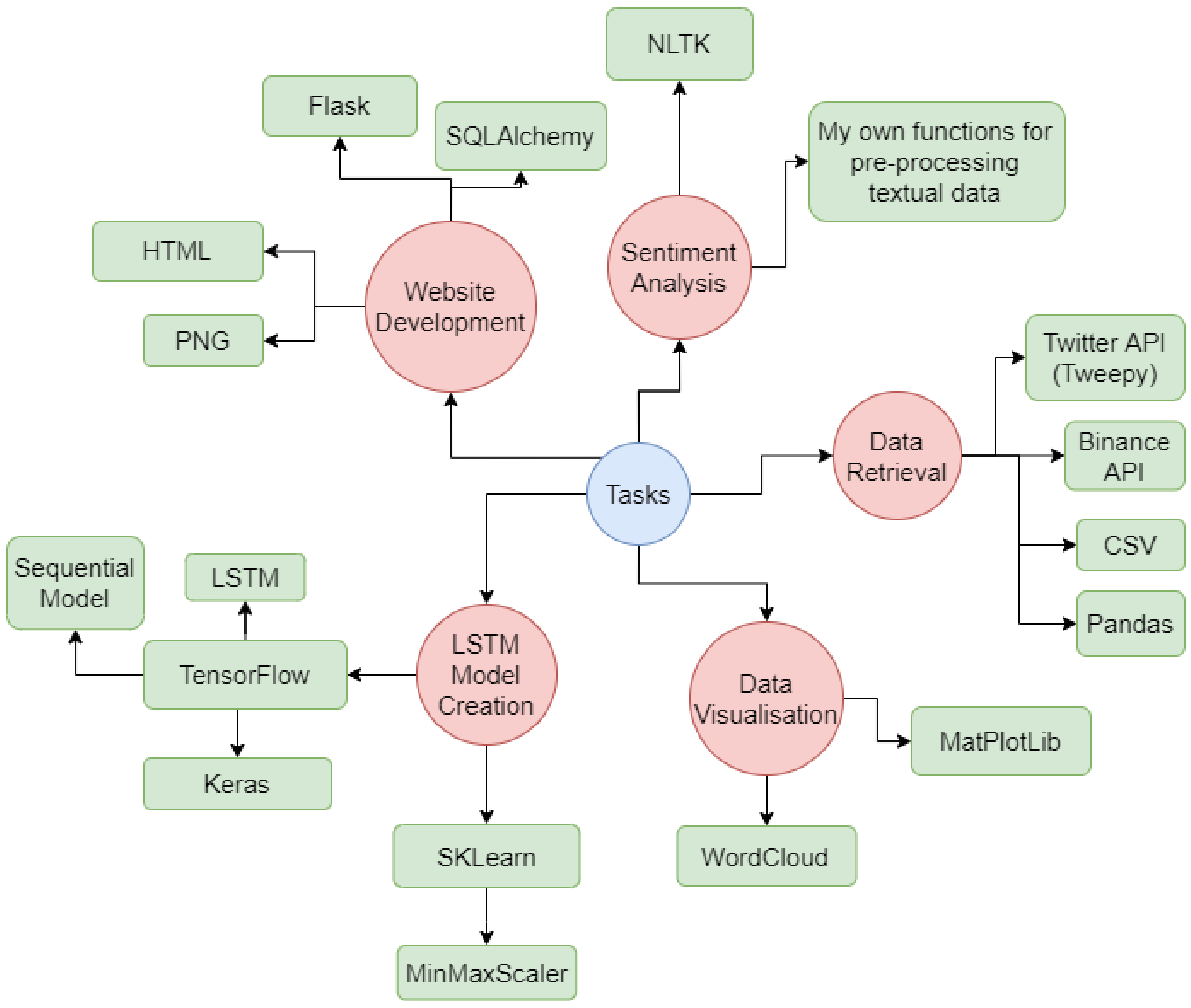


Fig 3. Technology Diagram

The above diagram details an overview of the technology used in this project. The red circles (Fig.3) represent the core tasks and the green rectangles the technologies used in the development of those tasks.

3.3 Data Preparation For Training Model (Sequence Diagram):

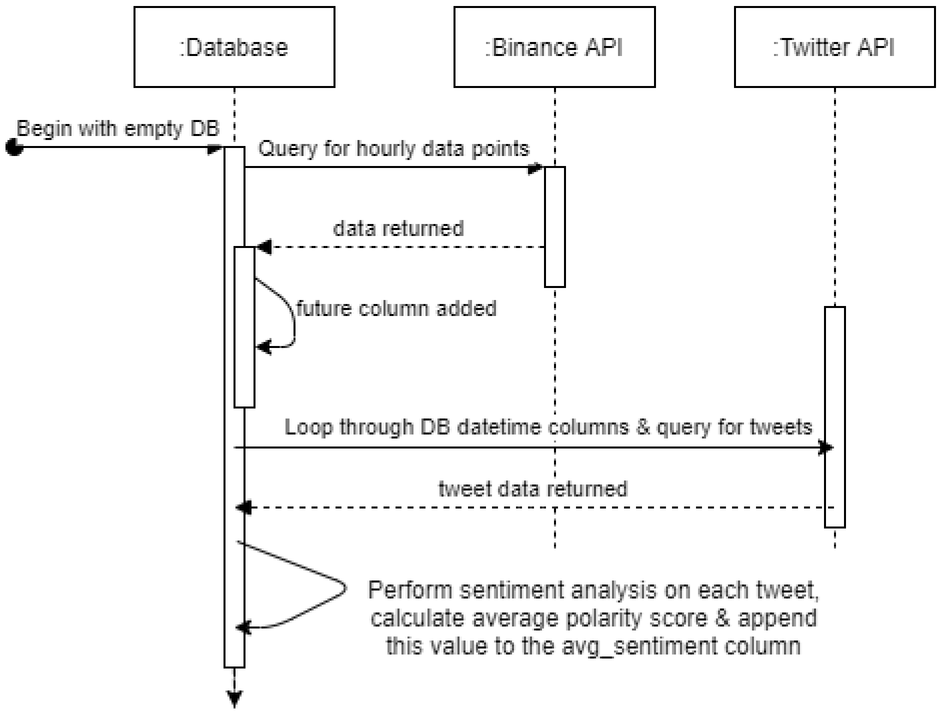

:Database
:Binance API
:Twitter API
Begin with empty DB
Query for hourly data points
data returned
future column added
Loop through DB datetime columns & query for tweets
tweet data returned
Perform sentiment analysis on each tweet, calculate average polarity score & append this value to the avg_sentiment column

Fig 4. Data Preparation Sequence Diagram

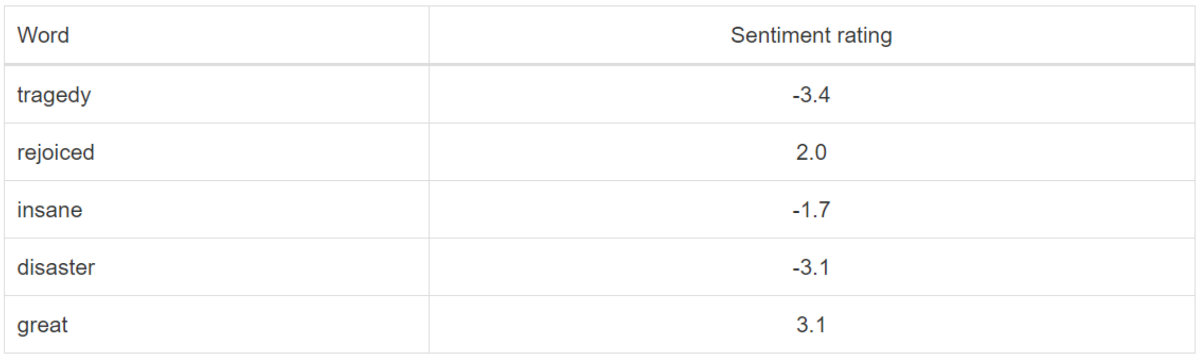
The above sequence diagram shows the processes involved with retrieving and preparing data for use in neural network model training. I will outline each step in greater detail below.

1. Firstly, we create a list of datetimes, the datetimes of the days which we want to get price data for.
2. Next, we loop through each datetime and make a query request to the Binance API. Each request we pass:
    a. Start time (current datetime in list)
    b. Finish time (next datetime in list)
    c. Interval (the time interval for each row of data, in my case it is an hourly interval)
    d. Currency pair (the currency pair we want data for, e.g BTC/USD)
3. Next, we append the returned data to a CSV. We only append the data that is useful to us. That is ‘time_id’, ‘datetime’, ‘close’, ‘volume’, and we add ‘average sentiment score’ but this column is left blank for the moment.
4. Next, we add a new column ‘future close’ which is the closing price 24 hours into the future.
5. Now that we have all the cryptocurrency price data that we need, the next step is to retrieve Twitter data in order to calculate sentiment scoring. We loop through the datetime column of our database and make a query request to the Twitter API for each datetime. We get an estimate of 100-200 tweets returned per request.

6. Finally, we perform sentiment analysis on each tweet using the Natural Language Toolkit(NLTK) library's 'Sentiment Intensity Analyzer' object. Once we have a polarity score for all the tweets, we can then calculate the average score and append it to the 'average sentiment score' column of the CSV at the row matching the datetime used for the tweets retrieval. We repeat until all the rows have a sentiment score value.

Section 4 - Choice of Sentiment Analysis Tool

The Natural Language Toolkit(NLTK) is a leading platform for building Python programs to work with human language data. It provides easy to use interfaces to over 50 corpora and lexical resources such as WordNet [1]. It has comprehensive API documentation which makes it suitable for this project as I am able to find support online with any issues that arose. NLTK has been called "a wonderful tool for teaching, and working in, computational linguistics using Python," and "an amazing library to play with natural language.". NLTK's Sentiment Intensity Analyzer works using something called VADER, which is a list of words that have a sentiment score associated with each of them (Fig. 5). The scores are based on a pre-trained model labeled as such by human reviewers. When using this technology to score the sentiment of a sentence, if the sentence contains more positive words, the sentence will be scored more positively and vice versa.



Word	Sentiment rating
tragedy	-3.4
rejoiced	2.0
insane	-1.7
disaster	-3.1
great	3.1

Fig 5. VADER example (*lexical* approach to sentiment analysis)

In order to score each word accurately, the developers of these approaches need to get a bunch of people to manually rate them, which is costly and time-consuming. Another important requirement for this method is that the lexicon needs to have good coverage of the words in your texts of interest, otherwise it may not give accurate scores. Although, when there is a good fit between the lexicon and the text, this approach can be accurate, and also can quickly return results even on large amounts of text.

The fact lexicons are expensive and time-consuming to produce means they are not updated all that often. This means they lack a lot of current slang terms which people often use in social media posts. One of the things that VADER recognises is capitalisation, which increases the intensity of both positive and negative words [2]. Another factor that increases the intensity of sentence sentiment is exclamation marks, with up to 3 exclamation marks adding additional positive or negative intensity. VADER also takes into account what happens when modifying words are present in front of a sentiment term. For example, “extremely bad” would increase the negative intensity of a sentence, but “kinda bad” would decrease it [2].

TextBlob is a similar library for processing textual data. It is built on top of NLTK, but is easier to use. It’s method of sentiment analysis is based on a separate library caller *pattern*. The lexicon bundled in Pattern focuses on adjectives. It contains adjectives that occur frequently in customer reviews, hand-tagged with values for polarity and subjectivity. It’s similar to NLTK’s VADER, but it specifically looks at words from customer reviews. TextBlob has another option for sentiment analysis which uses a Naive Bayes Analyzer, this is a machine learning technique. When using this option of TextBlob, the sentiment is coming from an NLTK classifier trained on a movie review corpus.

While VADER focuses on content found everywhere, TextBlob’s two options are specific to certain domains. The original paper for VADER noted that it is effective at general use. The methods which are trained on words from a certain field will be good at sentiment in that certain field.

As there are currently none of these domain specific methods available for use on data about cryptocurrency, my decision was to use NLTK’s more general library.

Section 5 - Data Handling & Visualisation Tools

Matplotlib is a comprehensive library for creating static, animated, and interactive visualisations in Python [3]. It is the brainchild of John Hunter, who, along with many of its contributors, have put an immense amount of work into creating this piece of software which is utilized by thousands of scientists worldwide. Pyplot is a Matplotlib module which provides a MATLAB-like interface. Matplotlib is designed to be as usable as MATLAB, with the ability to use Python, and the advantage of being free and open-source [4].

For this project, I needed to visualise my data findings on graphs, so that observations could be made with the data at a visual level. Matplotlib was a useful tool for generating these graphs which are displayed on the website, an example can be seen in Fig. 6 below.

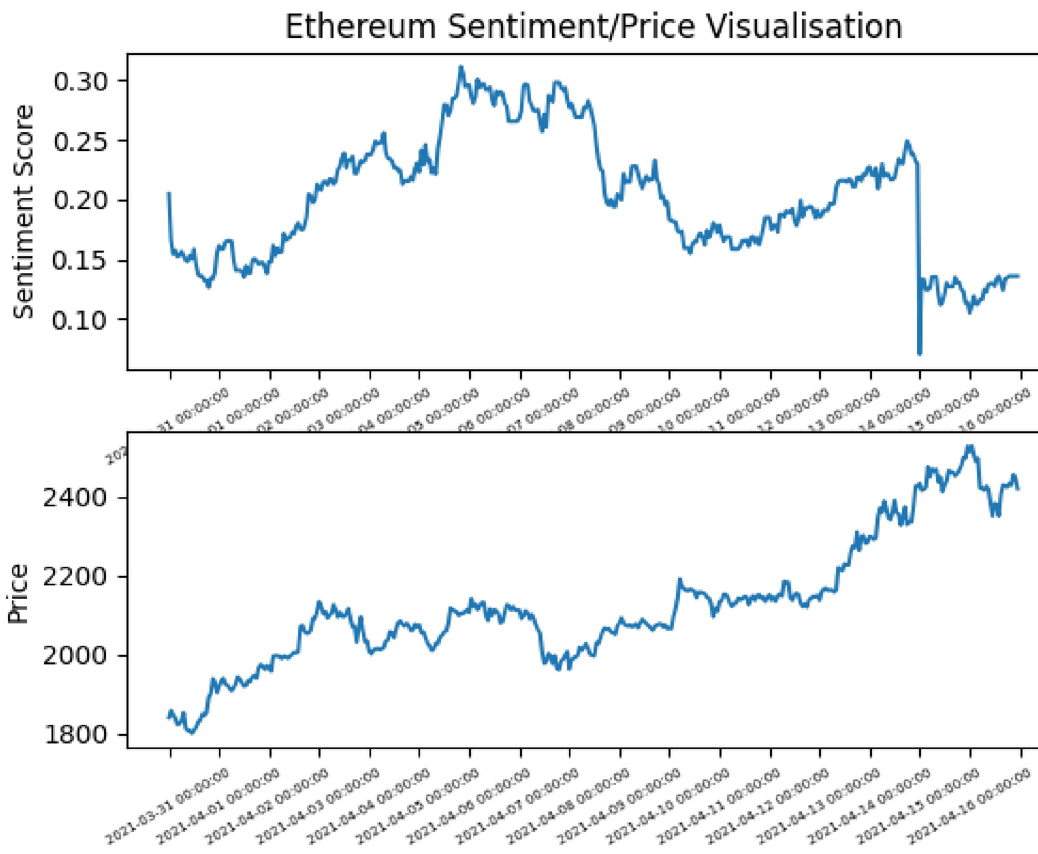


Fig 6. Price & Sentiment data plotted on graphs using Matplotlib

Another tool that was used for data visualisation was WordCloud. This is a word cloud image generator. It works by taking as input some textual data, and generating a word cloud image where the words from the text are randomly placed in the picture. The more commonly a word is found in the text, the larger that word is presented in the generated image. I use this to generate word clouds from the twitter data I use to calculate polarity scores. The idea is that we might gain some insight to the common phrases people use when tweeting about a cryptocurrency. This should highlight the most pertinent parts of the textual data. (Fig. 7)

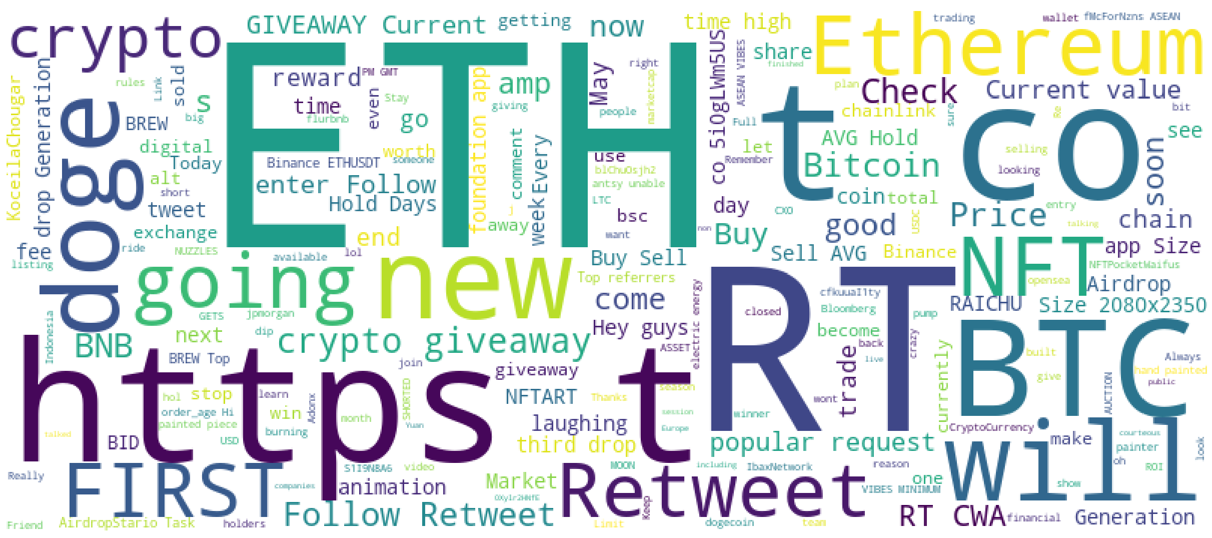


Fig. 7 Word cloud image.

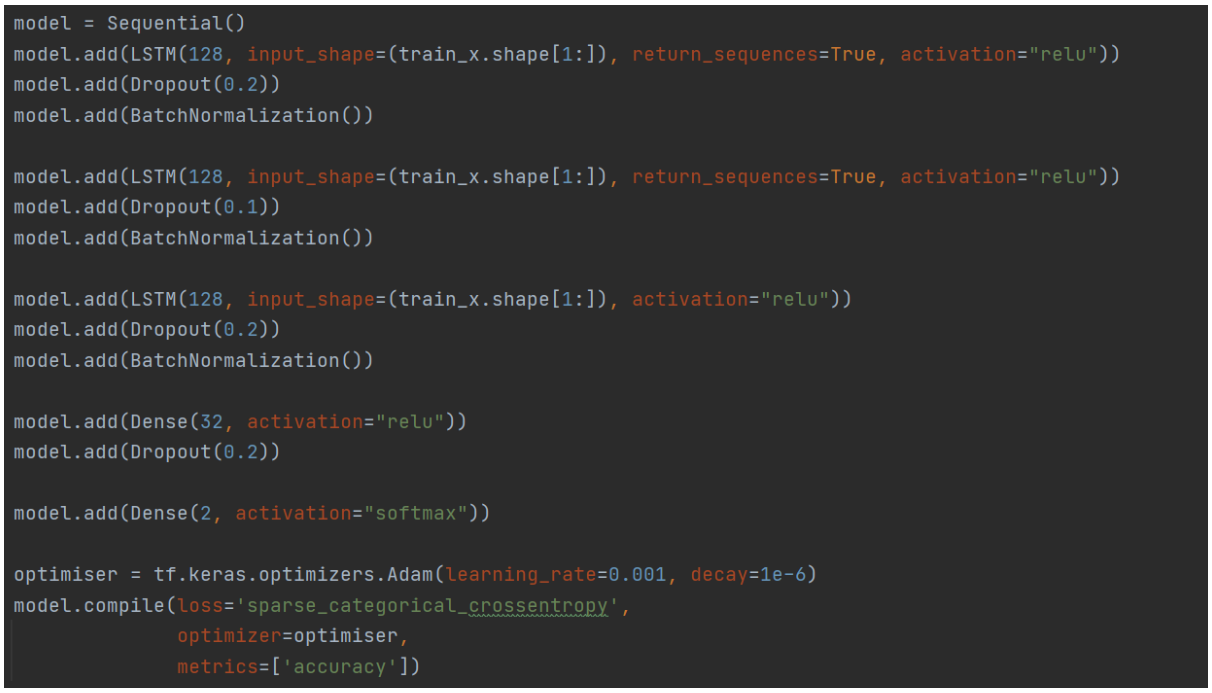
Section 6 - Machine Learning Model Creation & Use

"TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications." - Tensorflow documentation [5]

The technology that was used to build the machine learning models used for this project was TensorFlow. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 in 2015 [5]. Since its release, the Tensorflow technologies have been utilised in a plethora of machine learning projects, ranging from research endeavours to tested and released applications. Tensorflow provides a stable Python API and this is how the technology was used in this project.

It is a symbolic math library that uses dataflow and differentiable programming to perform various tasks focused on training and inference of deep neural networks. In recent years, deep learning has started to outperform all other machine learning algorithms when given massive amounts of data. Google saw this and decided to use these deep neural networks to improve its services, such as Gmail and the Google search engine.

The Tensorflow architecture at a high level, works in three parts. The first being the preprocessing of the data, and this involves scaling the data to an acceptable numerical value (i.e between 0 and 1) and formatting the data to the correct shape. The input shape relates to the multi-dimensional array of the data. The input data shape must match that of the first layer of the network. The second part of the architecture is building the model. This involves specifying the network layer properties such as the type, input shape, activation function and the learning rate of the model. An example of these specifications written in Python can be viewed in Fig. 8 below.



```
model = Sequential()
model.add(LSTM(128, input_shape=(train_x.shape[1:]), return_sequences=True, activation="relu"))
model.add(Dropout(0.2))
model.add(BatchNormalization())

model.add(LSTM(128, input_shape=(train_x.shape[1:]), return_sequences=True, activation="relu"))
model.add(Dropout(0.1))
model.add(BatchNormalization())

model.add(LSTM(128, input_shape=(train_x.shape[1:]), activation="relu"))
model.add(Dropout(0.2))
model.add(BatchNormalization())

model.add(Dense(32, activation="relu"))
model.add(Dropout(0.2))

model.add(Dense(2, activation="softmax"))

optimiser = tf.keras.optimizers.Adam(learning_rate=0.001, decay=1e-6)
model.compile(loss='sparse_categorical_crossentropy',
              optimizer=optimiser,
              metrics=['accuracy'])
```

Fig. 8 Example of model specifications.

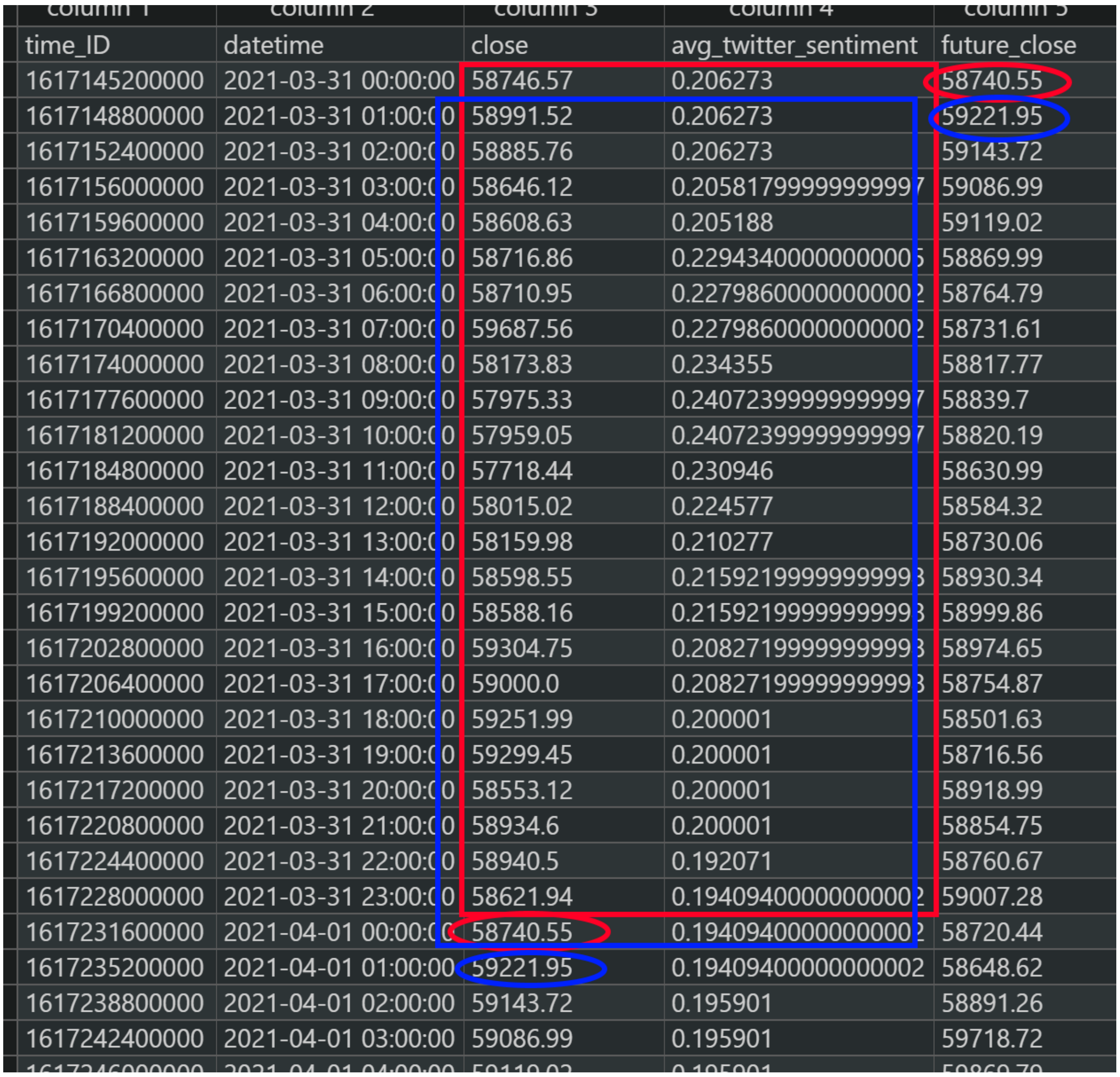
The third part of the Tensorflow architecture is the training and testing of the model. As a machine learning model is a function with learnable parameters that maps an input to a desired output, the optimal parameters are obtained by training the model on data where we know the desired output [5]. The training involves the following steps:

1. Getting a batch of data to the model.
2. Asking the model to make a prediction.
3. Comparing the prediction with the “true” value.
4. Deciding how much to change each parameter so the model would make a better prediction if it were to see that batch of data again.

A well trained model will provide an accurate mapping from the input to the desired output.

I will now detail how I obtained my data batches for training of my model. I wanted my model to make predictions for 24 hours, or one day, into the future. Also, my LSTM model looks at 24 hours worth of hourly data in order to make one prediction of the price 24 hours after the last data row it has seen of that input batch.

To get this done I sorted my input data into batches, each of 24 sets of my 2 input features (closing price & sentiment score). The training input (train_x) is a list of these batches. The labels (train_y) are the future prices for each batch. I try to visualise this in Fig. 9 below, where the red rectangle contains the first input batch and the red ellipse contains the target for that batch. The blue shapes show how the next batch and targets look.



column 1	column 2	column 3	column 4	column 5
time_ID	datetime	close	avg_twitter_sentiment	future_close
1617145200000	2021-03-31 00:00:00	58746.57	0.206273	58740.55
1617148800000	2021-03-31 01:00:00	58991.52	0.206273	59221.95
1617152400000	2021-03-31 02:00:00	58885.76	0.206273	59143.72
1617156000000	2021-03-31 03:00:00	58646.12	0.20581799999999997	59086.99
1617159600000	2021-03-31 04:00:00	58608.63	0.205188	59119.02
1617163200000	2021-03-31 05:00:00	58716.86	0.22943400000000005	58869.99
1617166800000	2021-03-31 06:00:00	58710.95	0.22798600000000002	58764.79
1617170400000	2021-03-31 07:00:00	59687.56	0.22798600000000002	58731.61
1617174000000	2021-03-31 08:00:00	58173.83	0.234355	58817.77
1617177600000	2021-03-31 09:00:00	57975.33	0.24072399999999997	58839.7
1617181200000	2021-03-31 10:00:00	57959.05	0.24072399999999997	58820.19
1617184800000	2021-03-31 11:00:00	57718.44	0.230946	58630.99
1617188400000	2021-03-31 12:00:00	58015.02	0.224577	58584.32
1617192000000	2021-03-31 13:00:00	58159.98	0.210277	58730.06
1617195600000	2021-03-31 14:00:00	58598.55	0.21592199999999998	58930.34
1617199200000	2021-03-31 15:00:00	58588.16	0.21592199999999998	58999.86
1617202800000	2021-03-31 16:00:00	59304.75	0.20827199999999998	58974.65
1617206400000	2021-03-31 17:00:00	59000.0	0.20827199999999998	58754.87
1617210000000	2021-03-31 18:00:00	59251.99	0.200001	58501.63
1617213600000	2021-03-31 19:00:00	59299.45	0.200001	58716.56
1617217200000	2021-03-31 20:00:00	58553.12	0.200001	58918.99
1617220800000	2021-03-31 21:00:00	58934.6	0.200001	58854.75
1617224400000	2021-03-31 22:00:00	58940.5	0.192071	58760.67
1617228000000	2021-03-31 23:00:00	58621.94	0.19409400000000002	59007.28
1617231600000	2021-04-01 00:00:00	58740.55	0.19409400000000002	58720.44
1617235200000	2021-04-01 01:00:00	59221.95	0.19409400000000002	58648.62
1617238800000	2021-04-01 02:00:00	59143.72	0.195901	58891.26
1617242400000	2021-04-01 03:00:00	59086.99	0.195901	59718.72
1617246000000	2021-04-01 04:00:00	59119.02	0.195901	59860.70

Fig. 9 Representation of my batch to target training data

Section 7 - Website Development Tools

I used the Flask Python module to develop my website. Flask is a web application framework which allows you to develop web applications faster. A web application framework represents a collection of libraries and modules that enable developers to write applications without worrying about low-level details such as protocol, thread management etc. Flask is often referred to as a microframework. It is designed to keep the core of the application simple and scalable. Flask is one of the most popular web frameworks, meaning it's up-to-date and modern. You can extend its functionality and scale it up for complex applications.

References:

1. NLTK 3.6.2 documentation. Found at: https://www.nltk.org/

2. Using VADER to handle sentiment analysis with social media text, 2017.
   Link to blog:
   https://t-redactyl.io/blog/2017/04/using-vader-to-handle-sentiment-analysis-with-social-media-text.html

3. Hunter, John D. "Matplotlib: A 2D graphics environment." IEEE Annals of the History of Computing 9.03 (2007): 90-95.
   Found at: Google Scholar

4. Matplotlib documentation. Found at: https://matplotlib.org/

5. TensorFlow documentation. Found at: https://www.tensorflow.org/