Final Report

"Sentiment analysis of social media text and its relationship with the price of cyrptocurrencies over time"

Cardiff University School of Computer Science and Informatics

course ref: Msc Computing

Student: Philip Byrne, Student Number - C210787

Supervisor: Padraic Corcoran

Abstract

The crypto currency market is now an important part of global financial markets, it seems to be driven by a different cohort of investor. These investors are very active on social media. Can understanding the sentiment on social media predict Bitcoin and other cryptocurrencies future price movements? Emotions are an important part of making investment decisions (Devi,2014) Bitcoin and other cryptocurrencies seem more volatile than other investment classes, is social media sentiment a dominant lead indicator?

Sentiment analysis is a key tool in predicting a number of behaviours and consumer demand. It has its origins in studies of the 1950's in the field of linguistics.

Its use grew substantially in the early 2000's with the start of MySpace and then other social media. Initially it was used in politics and for political sentiment but this has evolved into key predictors of consumer and market behaviour. Sentiment analysis involves analysing text to determine whether it is positive, negative, or neutral in sentiment. There are two essential approaches used, using lexicons, or a machine learning approach. This project has focused on various machine learning approaches only.

There is an investment theory called the "cycle of market emotions"(Russell invesments,2021). This means that emotion comes first and reaction follows. If true, using sentiment analysis, can measure investor emotions from social media such as Reddit or Twitter, predict the price (up or down) of crypto currencies such as Bitcoin and other cryptocurrencies that are included in the cryptocurrency index. The following document sets out a series of machine learning approaches to analyse the correlation of one against the other, however, the exercise is subject to the limitations of the data, technology and algorithms used.

Table of Contents

Acknowledgement

I would like to thank my supervisor Padraic Corcoran for his guidance and patient help throughout the many challenges of this project.

Table of Figures

1 Introduction:

This project aims to evaluate if there is a link between social media sentiment (towards digital currencies and Bitcoin in particular) and the historic price of Bitcoin and a price index of cryptocurrencies.

Many younger investors are investing in digital currencies such as Bitcoin and overall, the price has moved very significantly over the last five-year period (Barrett,2018). Over this time, the price has been volatile and has moved both upwards and downwards. This project aims to discover if there is a direct link between sentiment analysis using historical data from January 1st, 2017, to 1st September 2021. This data set has been chosen for cost and resource reasons. The data is in 15 minute periods, as the historic prices are available for free if they are for 15 minute periods. We have matched social media posts within the time periods of the price data. The project has selected the most appropriate technologies that are open source or that offer an amount of memory and processing power for free. The project uses a number of technologies and there are a lot of steps involved, from retrieving the data, parsing, storing and normalising the data. The project examines and then tests a number of text classifiers for sentiment analysis, it trains the models on the data. The data is then normalised, so that it can be compared with pricing data, where both are within 15 minute intervals. It then compares the results of different models used and also attempts to interpret the results. Its objective is to understand if it is possible, using the models, datasets and technology used to determine if social media sentiment can predict the price of Bitcoin and the price of the crypto currency index.

1.1 Social media content as a data source for sentiment analysis:

Social media consists of text, images and video. Both Twitter and Reddit are available through standard API's. This project dealt with fairly large volumes of social media data over a long period of time. (There were also substantial amounts of historic pricing data used). There was considerable effort in selecting free technologies to extract, store and refine the data at different points.
We look at both sources, extracting text from images and video as well as text. While social media texts are considered microblogs, they are widely used in the area of sentiment analysis (Ali et all , 2017). We have used postings from both, Reddit also has sub-reddits and then Twitter has a lot of associated images and video along with text. In total over 625,000 social media posts were used, amounting to 51 GB of memory space for a period of over three years.

1.2 Sentiment analysis:

This was the most intensive part of the project. We used a machine learning approach and we ran a series of tests and trained data on three main classifier models. Classifiers have been initially selected based on their suitability to classify both Reddit and Twitter feeds, the level of training needed on the dataset and their suitability for sentiment analysis. In order to test the classifiers, the dataset was converted into vector representation, using both Scikit-learn and Spacy, both Python libraries.

The classifiers selected were N.L.T.K. which calculates sentiment through VADEAR (Valence Aware Dictionary and Sentiment Reasoner) which constructs a list of lexical features, measuring and grouping intensity of the meaning behind a corpus of text.

The other two methods BERT (DistilBERT) and Flair operate in the same transformer method , where bidirectional encoders speak to other vectors and autoregressive steps inform the next iteration of the previous iteration's progress. Each of these methods strengths and weaknesses are evaluated and a conclusion is formed as to the optimum use of classifiers for sentiment analysis on the social media text.

1.3 Regression analysis of datasets

Once an overall approach for classification and scoring of sentiment was determined, the historic pricing of both Bitcoin and cryptocurrency index were obtained through Binance. All data sets were normalised and a number of regression models were run using a GPU, given the size of the datasets involved. Classification and regression models were used. These show that there is a definite correlation between social media sentiment and the future price trend of both Bitcoin and the cryptocurrency index. Given the relative size of Bitcoin, to the other cryptocurrencies within the index, Bitcoin on other tests of the historic data is the dominant variable within the index, meaning that the Bitcoin price and the cryptocurrency index are strongly related to each other (magas,2020). The models, using this particular set up cannot predict the Bitcoin price, but this method is useful for predicting the price trend.

1.4 Research objectives

The main research objectives of this project is to determine whether the end model is a sound option to use for buying and selling options for crypto currencies. The following objects will help to define overall success of the project which end in measuring accuracy of predictions as success.

- To identify a correlation between sentiment of raw text regarding cryptocurrency from Twitter and Reddit and the future price movements of Bitcoin and other crypto currencies.
- If there is a correlation between the sentiment analysis and Bitcoin prices as above, under what conditions does it have greatest impact?
- To explore if there is any difference between social media sentiment of text from Twitter or Reddit in terms of sentiment or result in Bitcoin price?
- To show causation between Bitcoin price fluctuations and movements in other cryptocurrency prices.
- To build accurate sentiment classification models using BERT and other models to group the text between positive and negative sentiment.
- Create an accurate classification regression model which can predict if the price will go up or down depending on the amount of sentiment posts at a given time.

2 Background

There are a number of key areas of development and technologies used on the project. The main areas of the project use a number of aspects of an n-tier architecture, together with key text analytics methods, in terms of classification models, sentiment analysis models, big data and supercomputing to undertake regression analysis on large volumes of data. Finally, the most appropriate methods for the visualisation of the results are used. The figure below illustrated the steps that have been carried out and set out the methods used and the rationale for these methods.

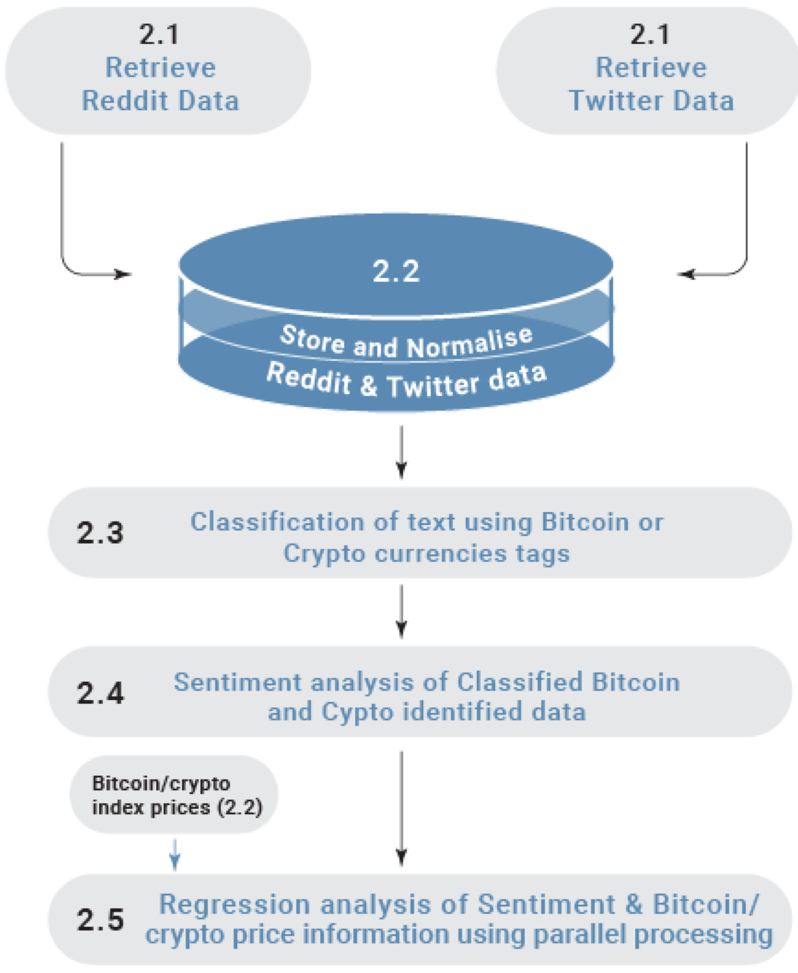

2.1 Retrieve Reddit Data
2.1 Retrieve Twitter Data
2.2 Store and Normalise Reddit & Twitter data
2.3 Classification of text using Bitcoin or Crypto currencies tags
2.4 Sentiment analysis of Classified Bitcoin and Cypto identified data
Bitcoin/crypto index prices (2.2)
2.5 Regression analysis of Sentiment & Bitcoin/ crypto price information using parallel processing

Figure 1: Overall project schema

2.1 Gathering Reddit & Twitter data

A proprietary API was used instead of using a number of standard API's. Standard API's such as PSAW API and the PRAW API were initially considered. They were considered not suitable because of the restrictions in the amount of data that could be queried at one time (e.g. cannot filter by date), the API's have hit limits and cannot guarantee a fixed amount of searches per session.

The reason for deciding to build a proprietary API was due to the need to query any date on Reddit and sub-reddits. The same requirement was needed for Twitter, the ability to query on any date on a Twitter search since its foundation.

jsoup and Spark were selected for the proprietary API, using the search bar web address on Twitter and the individual sub address for Reddit. There is built-in functionality contained in the metadata attached to the web address being queried. This can guarantee one hundred posts in each session/query (maximum per search) for both Reddit, sub-reddit and Twitter sources. The existing API's could not guarantee a fixed number of posts per search.

Jsoup operates from a HTML request library that can retrieve data points from web addresses (crawler). Spark was selected to insert a filter into the meta data, such as specific dates and times and key words through rejects commands.

2.2 Store and normalise Reddit and Twitter data

Initially the posts scraped from either Reddit or Twitter are stored as unstructured data in a datalake. A datalake was chosen, as the posts (or datapoints) are in a variety of text, images and video formats and initially should be in unstructured form. Unstructured data takes up less memory space than structured data, the approach is to initially store the scraped data in unstructured format, before it is filtered and refined, where a smaller number of records can be stored in a database.
Mongo DB was chosen as the datalake, as it offered the highest storage for free (10 GB), which amounts to 3 years of unstructured data. It has many desirable features as an object database, however, the primarily concern is persistence storage at this phase of the process. This allows for additional processing of data to test hypothesis over a period of time.

2.1.1 Filtering and Normalising the data

All records are time stamped and filtered, so that the social media records are stored into 15 minute time intervals. We do this because the historic Bitcoin and cryptocurrency price information is free in intervals of 15 minutes and these datasets will be compared with each other later.
Each post is then sorted into text, image or video clip, or a combination of all three. Each record is also tagged by being either from Reddit, or Twitter, so that analysis can be performed on any variations between Twitter or Reddit user sentiment, or Bitcoin movements.
Once each record has been tagged according to origin, timestamp, form (text, image or video), we start staging through decision trees.

2.1.2 Database:

(Jatana el all ,2021) Given that the records are now provisionally filtered, they need to be stored in a database. We have had three choices of database technology, an RDBMS (a relational database) or sometime referred to as a SQL database, where the data is "related" or joined together. An object database (like MongoDB), where all the data elements are stored and operated as objects. Finally, a graph database was considered. Graph databases are schema-less databases which use graph data structures along with nodes, edges and certain properties to represent data.

Given the requirements of all of the data to be normalised, so that we could run regression analysis against Bitcoin and other price information, PostgreSQL was selected as:

It was the most efficient for the way the social media data needed to be structured.
It was free for the amounts of data and iterations required.
It matched the other technologies used and SQL is very accessible to most other systems.

2.2 Normalisation of text, images and video related to Bitcoin and Crypto currencies:

The diagram below shows how all the social media posts are classified. The diagram also shows the use of the Naïve Bayes model (which will be discussed later). The posts consist of text, video and images. Any text is extracted from posts with video or images, so the text can be classified as relevant to Bitcoin, or crypto currencies. See figure (below)

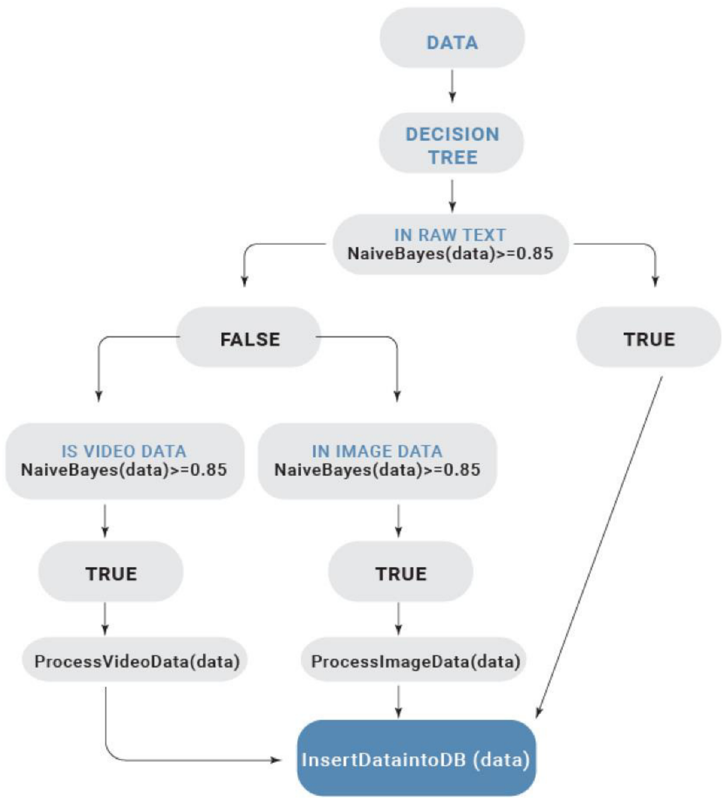

DATA
DECISION TREE
IN RAW TEXT NaiveBayes(data)>=0.85
FALSE
TRUE
IS VIDEO DATA NaiveBayes(data)>=0.85
IN IMAGE DATA NaiveBayes(data)>=0.85
TRUE
TRUE
ProcessVideoData(data)
ProcessImageData(data)
InsertDataintoDB (data)

Figure 2: Data Gathering from source to insertion into DataLake

2.4 Sentiment Analysis

Sentiment analysis, or opinion mining, is a part of natural language processing that analyses opinions, sentiments, attitudes, and emotions in text and some other characters. Its use has greatly expanded as the amount and variety of social text has grown rapidly. (Liu ,2012). Within the scope of this project, the sentiment analysis relates to the classification and production of sentiment scores to the social media posts.

There are two main approaches, either a lexicon approach, or a machine learning approach. The Lexicon approach relies on three proprietary data dictionaries LIWC GI and Hu-Liu data dictionaries , (Taboada et all,2017).

2.4.1 Machine Learning approaches to sentiment analysis:

This relates to learning the sentiment of relevant text. There are a number of methods that we have considered, based on the quantity of messages and data available, versus the fact that the sentiment of Bitcoin and cryptocurrencies is of moderate complexity.

The Naïve Bayes (NB) is a simple classifier that relies on Bayesian probability and the naïve assumption that feature probabilities are independent of one another.

Initial Classification of Bitcoin/cryptocurrency in social media post is used. The figure below shows how the initial social media is filtered for mentions of social media text relevant to Bitcoin. For the filtering of the initial social media posts, a Naïve Bayes has been utilised as per figure2 above.

Bayes for Multinomial classification

This has been used for grouping entities or documents together into a group with predefined rules. Which is perfect for grouping data types together for processing. Notation for Multinomial Bayes or Naïve Bayes

$$be = argmax\, Y\, P(y) \prod_{i\, =\, 1}^{n} P(xi|y)$$

Decision Tree is tree structure where there are leaves and nodes.
Each node ether has an actionable call for the data or passes the data along to the nearest leaf which passes it to the next, where the same action is taken until it finds an actionable node or it falls between an acceptable level probability that the node will accept.

Maximum Entropy (ME) is a machine learning technique using exponential models using multinomial logistic regression. Unlike NB, ME makes no conditional independence assumption between features.

Support Vector Machines (Steinwart, et all,2008) differ from both NB and ME models in that SVMs are non-probability classifiers which operate by separating data points in space using one or more hyperplanes (centrelines of the gaps separating different classes).

2.4.2 Vector Representation:

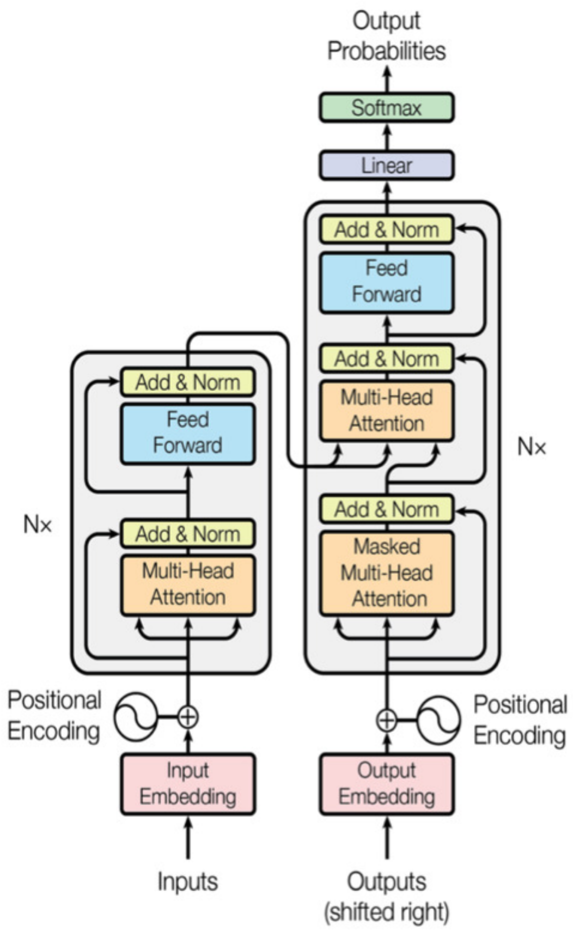
For text to be processed by any of the ML models it needs to convert the text into points in space. It uses cosine or Euclidean distance to measure similarities or map relations in whole text, tokens or the stem of an individual word in a cohort of text.
Cosine distance was chosen as duplicates appearing in the text matter as the goal is to analyse their overall context and sentiment. Cosine distance maps text to vectors by the cosine angle of two different pieces of text and determines whether they are pointing in the same direction and also determines their relationship (Prabhakaran,2018).
The following formula represents cosine distance:

$$\boldsymbol{cos\theta = a \cdot b \div \left|\left|a\right|\right| \left|\left|b\right|\right| = \sum\nolimits_{1}^{n} a\imath b\imath \div \sqrt{}\sum\nolimits_{n} 1a2i \sqrt{}\sum\nolimits_{n} 1b2i}$$

2.4.3 Transformers - neural networks

The network employs an encoder and decoder architecture much like traditional RNN's. The main difference between transformer architecture and RNN's is the sequence can be passed in parallel. Transformers were initially devised for language translation but have been adapted for various NLP tasks including classification (Illia,et all , 2018).

A typical transformer representation:



The Transformer - model architecture.

Figure 3: Transformer Architecture

2.4.4 Input Embeddings –

An input sentence is converted into an embedding in order to represent meaning to the machine using a positional vector in order to track context of that word in the sentence that is fed into the input layer.

The below function to determine positional vector space:

$$PE(pos, 2i + 1 \ldots.) = sin(pos \div \frac{100002i}{dmodel})$$

Figure 4: Function for positional space

2.4.5 Classifiers

N.L.T.K – Natural language Toolkit

NLTK calculates sentiment through VADEAR (Valence Aware Dictionary and Sentiment Reasoner) which constructs a list of gold-standard lexical features, measuring and grouping intensity of the meaning behind a corpus of text. Its result represent a normalized composite score from -1 to +1 where -1 being negative and 1 being positive and 0 being neutral.
It works by grouping re-emerging themes or tracking frequency of a given word root and grouping their appearance and importance to the intensity of the overall corpus inputted(yadav,2020). Flair is a standard BERT model that has been pretrained on a large corpus of social media text. As it is pre-trained, it relies heavily on contextual embeddings , which according to (yadav,2020) can be applicable to any N.L.P tasks while taking little of the client's RAM or memory usage through being available solely through the internet. The client never downloads the model, is only given the ability to access, through the set of libraries given. Other than accessing the off the shelf and pre-trained BERT model, Flair operates in the same transformer method as BERT , where bidirectional encoders speak to other vectors and autoregressive steps inform the next iteration of the previous iteration's progress(yadav,2020). It uses a style of BERT with more bidirectional layers as it tries to classify unseen corpus based on what is has been trained on.

BERT
BERT stands for a Bidirectional Encoder Representation from Transformers with contextual word embeddings and can be easily fine-tuned for NLP tasks including classification, translations and sentence prediction . It was originally trained on the English Wikipedia. There are many variations where the difference relates to the size of the model and how many encoding and decoding blocks there are.
The two original versions are the models on which other versions are based off and follow a similar process.

Below chart Illia et.all 2018

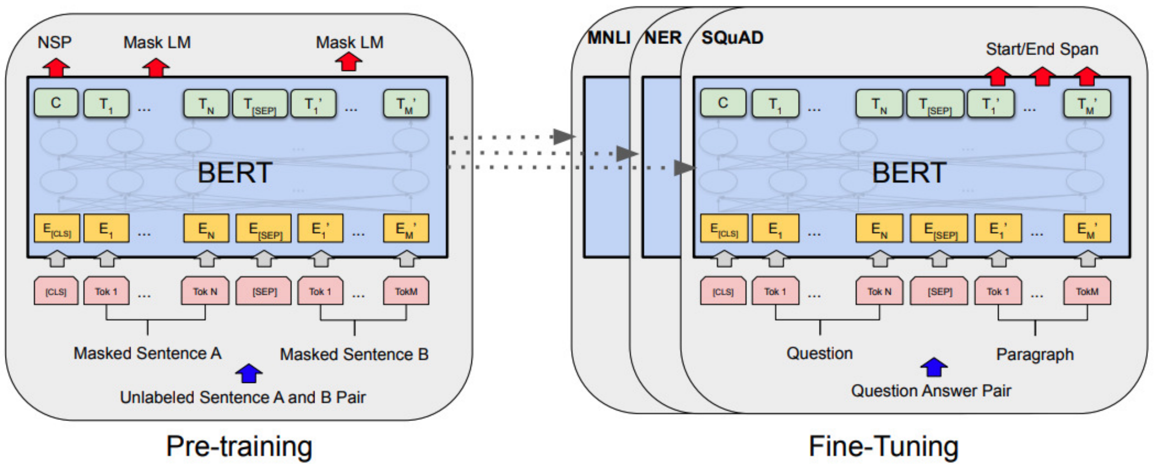


Figure 5: BERT & other models

Another chart from Illia et al , 2018, showing the difference between BERT and other models. BERT leverages transformers with bidirectional layers. It's transformer layers allow for contextual embeddings to be considered between data points.

2.4.6 BERT Models Considered

There are three BERT models that were considered, BERT, DistilBERT and RoBERTa.
The main criteria used to select a method were:

- Memory usage
- Training time
- Size and memory space required.
- Performance degradation

Model	BERT	Distil BERT	RoBERTa
Memory size (in millions)-bytes	110-340	66	110-340
training time	8*V100 X 12	8*V100*3.5	1024*V100*1
Performance	first transformer model	5% degradation from BERT	2-20% Improvement over BERT
Data	16GB	12GB	160GB
Method	Bidirectional transformer with MLM and NSP	BERT distillation	BERT without NSP

Figure 6: BERT Characteristics

Another chart from Illia et al, 2018, showing the difference between BERT and other classification models. As can be seen BERT leverages transformers with bidirectional layers. It's transformer layers allow for contextual embeddings to be considered between data points.

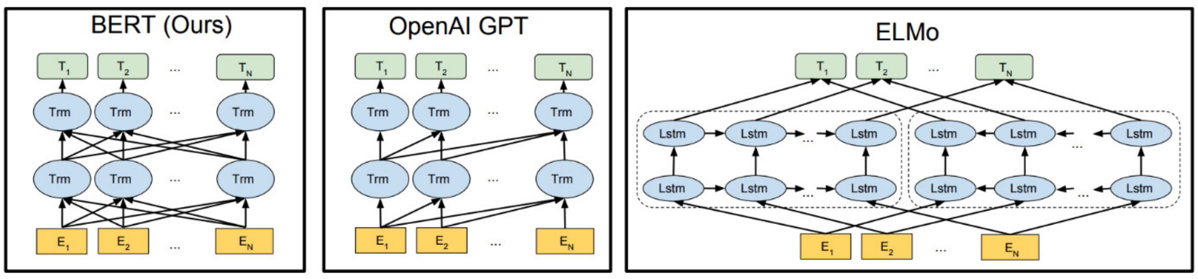


Figure 7: BERT and other classification models

2.4.7 Training BERT for Sentiment Analysis Scoring:

For Training BERT a sample dataset was taken from the data lake and was split into three further

datasets for the purposes of training, validation and testing.
Each of the datasets were given a column based on the text appearing for that data point which was a sentiment score using the Python library TextBlob giving each a sentiment ratio.

The newly given sentiment ratio column was the label for the machine to predict. The features for the model to predict was the text.

Text blob calculates sentiment in the following way:

$$\boldsymbol{polarity = 0.5 * (1 \div 1.3) * P}$$

$$\boldsymbol{subjectivity = (1 \div 1.3) * P}$$

$$\boldsymbol{sentimet = polarity \div subjectivity}$$

Figure 8: Formula for Sentiment

2.4.8 Sentiment Calculations

The sentiment calculations used scores between 0 (positive sentiment), 1 neutral and 2 (negative sentiment).

2.5 Regression Analysis

Models are needed that can input sequential data in order to understand context of the text to determine the correlation between price and sentiment. Given that this will be correlated to time series price data, the ideal model should have strong predictive qualities in this environment.

The following are candidate models:

RNN -> Recurrent Neural network used because

Recurrent neural network is suitable as it can consider sequential data while also being auto regressive it can input the previous token's state and weight in order to update itself during the optimization / training process. (IBM Cloud , 2021)

Success of models with be determined by the following formula:

$$Forecast\ Error(FE) = \frac{(actual - predicted)}{actual}$$

Figure 9: Formula for Model Success

ARIMA –> AutoRegressive Moving Average

- ARIMA models can be an effective at price prediction using large amount of historical data and time series analysis.
- ARIMA model is composed of three parts, Auto Regression, moving average and integration. This process can ensure it fits the data as accurately as possible.

- ARIMA'S model parameters are
  p: represents the order of the autoregressive model
  d: represents the degree of differencing
  q: represents the order of the moving average
-
  Notation shown below:

$$\left(1-\sum_{i=1}^{p}\phi_i L^i\right)(1-L)^d X_t = \delta + \left(1+\sum_{i=1}^{q}\theta_i L^i\right)\varepsilon_t.$$

Figure 10: Formula for Moving Average

- It is effective at predicting movements in established markets and has been a known predictive tool for some time.(whittle,2020)
- Can embedded sentiment analysis into the ARIMA model by using a vectorized B.O.W model (bag of words) with the following equation, with sentiment being calculated as
  Precision: amount of positive rated text
  recall: number of entities rated neutral (unidentifiable to the BERT model)
  F1 score: combination of both

$$F1\ score = \frac{2(Precision\ recall)}{(precision\ \ + recall)}$$

Figure 11: Formula for F1 Sore

- ARIMA can show future prediction of an asset and correlation through a multivariate input

<u>Issues with model:</u>

Cannot consider the sequence in which the words appear. It may be able to detect context through factors of frequency of the word's appearance, but it may not be as powerful as contextual embeddings.

LSTM – Long Short-Term Memory

<u>This is a strong candidate for the following reasons</u>:

- Is an artificial recurrent neural network unlike other models it has feedback and feedforward connections. These characteristics can be extremely useful for time series and sequential data. (Yuan, 2018)
- Long short-term memory (LSTM) networks are a variant of the standard RNN. By replacing the basic hidden neurons with LSTM units in RNN, LSTM networks can better handle the problem of gradient vanishing and explosion of long-term dependencies. Yuan,(2018).
- Every time step feeds the next data point is inputted alongside the hidden layer of the previous data point

- LTSM are more accurate for price prediction where long sequences exist in the data due to their memory gates and tracking the last input of the model , (Yuan,2019)
- LSTM has three gates which update each other after every regressive step.
  These gates are formed of the input gate, the forget gate and the output gate with a cell at its centre which is connected to the gates by hidden layers. Shown below - Yuan, Xiaofeng & Li, Lin & Wang, Yalin. (2019).

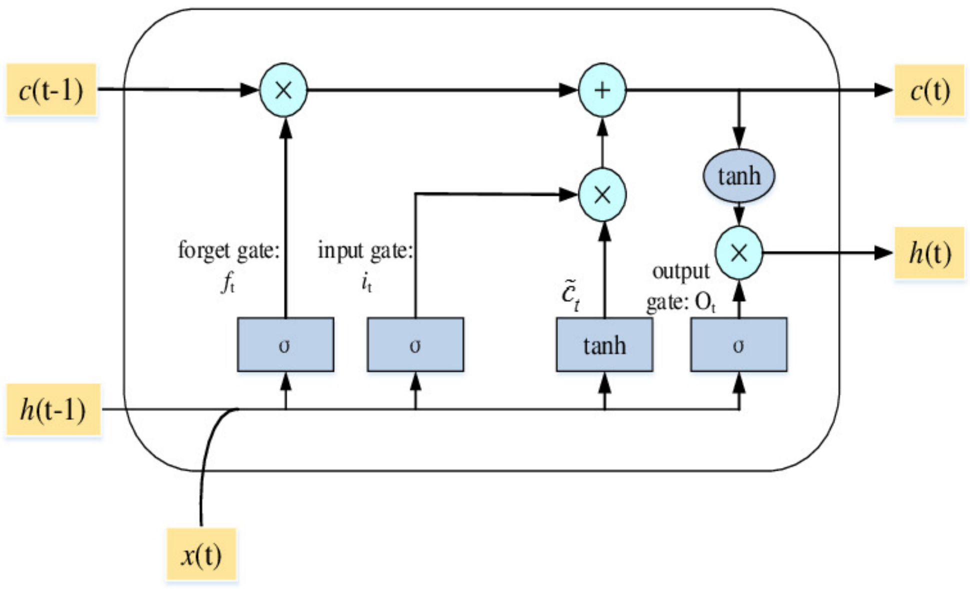


Figure 12 LSTM architecture:

- After each step, each gate stores the gradient during the optimization process to update each hidden layer in the network as it works through the input it has been fed.
- These gates act as a form of memory with their gates and weights updated during the optimization process.
- Gates are a way to let some information pass and blocking information deemed as unimportant to the computation. This is done through a sigmoid neural net layer and a point wise multiplication layer working simultaneously (Yuan,2019).

3.1 API – Data Gathering

3.1.1 Identifying text for the use case

A proprietary API was developed that uses keyword searches using filters connecting to both reddit and twitter API's, as discussed in Section 2.
At first these were chosen at random with one hundred messages seeking random possible words used. A Spark programme was created to consistently review these key words and would filter and delete them until they all had a similar amount of drawback (data retrieved upon A.P.I call) from the API.

3.1.2 Data gathering

In the data acquisition stage of the project, the following considerations need to be taken into account:

Concurrency (access); maximising the number of concurrent sessions (shared resource – over n tiers): mutual accuracy (where one crud action does not affect the integrity of the other); multithreading (multiple processes happening simultaneously).

Memory Usage: Given that large amounts of data (circa 10GB) data classification tools/methods (number of records, structured or unstructured text, image files, video files, large strings of text), memory use was an important consideration in determining classification method used.

There were also a number of data integrity considerations leading to hashing data relating to the identification of the user, including user id, username and the original post being encrypted with a private key so that it could only be read when being processed.

Parallel processing:

Context manager, manages the connection with the API ensuring best practice for API connections to all data feeds (Twitter, Reddit) ensuring each is within a hit limit of 100 data points every ten minutes.

3.2 Data Storage

Before data is pre-processed it must be contained and stored in the datalake as multiple different datatypes (text, gif, videos, etc) need to co-exist in the same eco-system.

A third-party hosted server HEROKU was selected as it offers the most availability in its free tier (what amount of services do they offer before they charge) – which in the case of HEROKU, is free for any interactions below 512MB. There is clearly laid out documentation on how to deploy and manage the server on HEROKU . The jobs sent to the server are tasks such as data acquisition, data wrangling and data processing, which are expensive in terms of memory use, where not a lot of memory is allocated. This means that effective memory management must take place in order to operate within HEROKU'S free tier (one dyno corresponds to 512MB of Memory).

These three problems were addressed with the following:

- To limit memory use, generators were used to pass data intermittently between objects and to the database (MONGO DB). The looping overhead was reduced as much as possible minimising memory usage. Adopting this approach reduced memory use by 30% and eliminated the use of one dyno which represents 512mb of storage on HEROKU memory.
- In order to handle the various types of data and the amount of data, a Spark programme was written to process data across three worker nodes concurrently.
  The Spark thread pooling programme consists of one master node with three worker nodes. Each node runs a connection to the designated API (Reddit, Twitter, Binnace) and classifies the data types with an auto tagging model before being sent to the master node for further staging into the data lake on MongoDB. Edge computing is achieved on the worker nodes having classified the data types it received and logging their order of appearance by index. This ultimately speeds up the processing at a later stage as the result is sent to a shared database . The diagram below illustrates this.

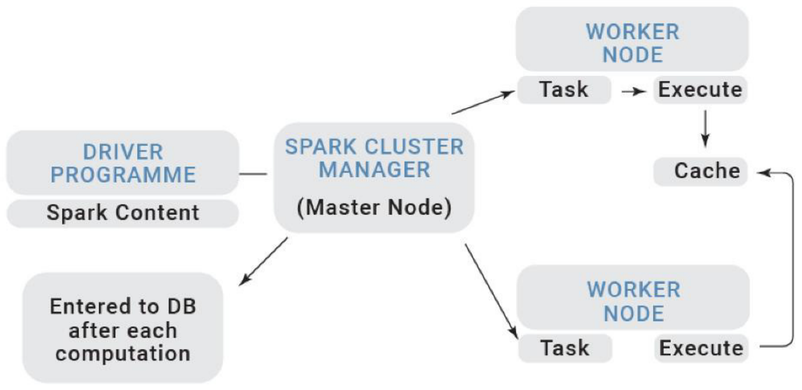


Figure 13: Spark processing

- The Spark programme was deployed to HEROKU with server cronjob running every fifteen minutes. It was set to every fifteen minutes as the processing time could be prolonged due to images and gifs being present in the posts retrieved.
- Fault Tolerance – when a cronjob initializes the master node and receives a response it first checks for three available nodes. It sends a request to previous nodes used to check if they are responsive. If not, it requests HEROKU to make another memory space available until it has three available nodes.
  If there is no response from the master node it requests a new memory space for the master node alongside three worker nodes to receive instructions. The below diagram shows this process

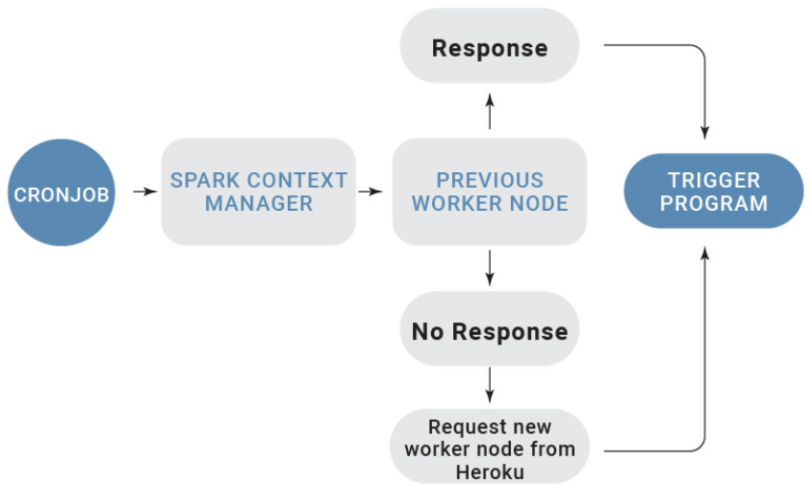


Figure14: Spark observer design

- Each of those worker nodes were given a hit limit of one hundred posts (datapoints) and the server cronjob ran every fifteen minutes in order to adhere to user agreements with each API.
- The data was stored on a MongoDB server in order to handle unstructured data with only a tag describing the type of data inserted alongside the time stamp which will help to identify that data entry at a later point as each timestamp is unique.
- A map reduce function is used to group data types together into the MongoDB, where the datatype id is the key and actual data is the value. It helps to pass a large dataset in this case (82GB dataset) to set of key value pairs which can then be grouped at the end resource. It essentially is a powerful method to group a large dataset and identify groups which can be put into keys with the values then being computed at the end point.
- Memory Management – Caching:
  To achieve load balancing and save memory space, a caching layer was introduced before the un-processed data is staged for submission to the data lake. This helped manage the requests and free up memory for new requests coming in. The un-processed data is stored in a queue style cache, where un-submitted data is held in the queue until enough RAM is free on the server to submit to the MongoDB. In the cache the data is stored in bytes in order to keep memory a low as possible. As BERT would commonly exceed memory usage of a dyno(HEROKU’s term for amount of memory used dyno=512MB) to stay within the confines of the free memory, the cache would free upload on the current server ultimately achieving a fair load balance on the HEROKU nodes.
- The following diagram illustrates the operation of the nodes, the staging for the datalake and the management of the memory using a caching layer.

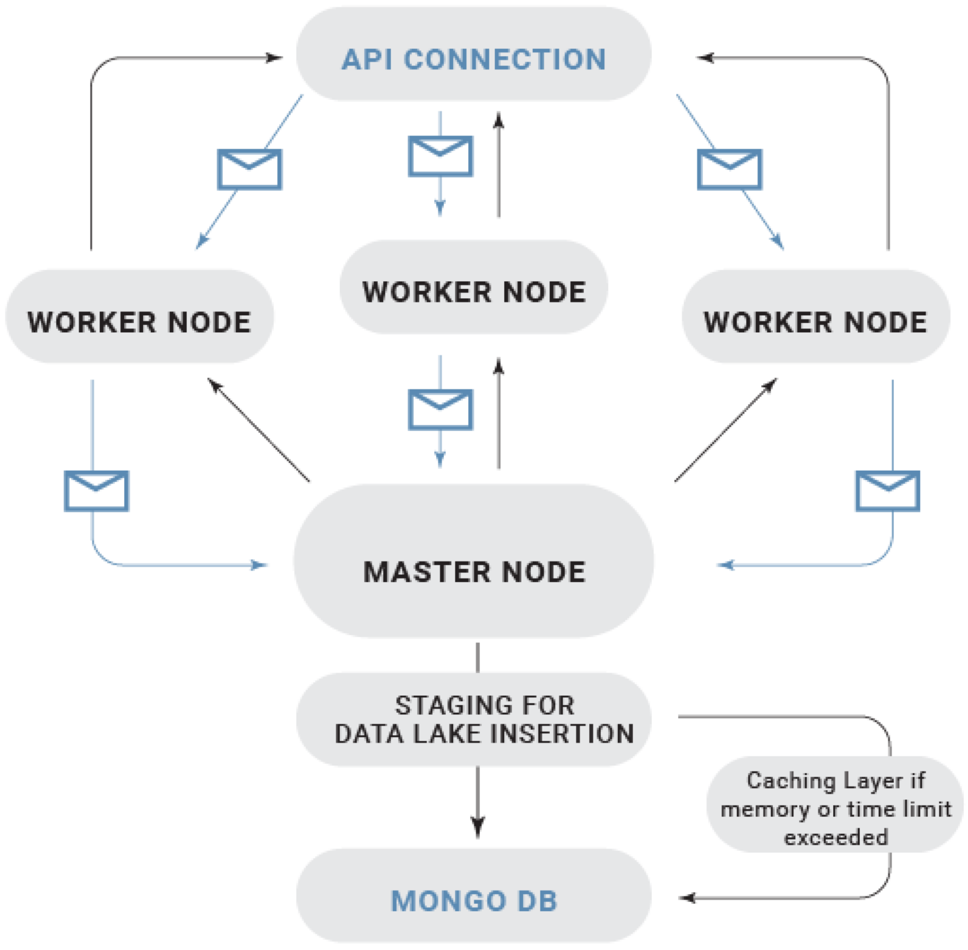

API CONNECTION
WORKER NODE
WORKER NODE
WORKER NODE
MASTER NODE
STAGING FOR DATA LAKE INSERTION
Caching Layer if memory or time limit exceeded
MONGO DB

Figure 15 : API design

3.3 Data wrangling - preparing a dataset for a specific model

As multiple models were assessed for each phase of the project, each potential option required a different processing approach. To help show how each model was prepared, different data pre-processing measures were taken.

As each model has its own requirements and definitions of machine-readable code , data wrangling must be decoupled for preparation on each particular model; meaning each model will also have a unique data wrangling process to prepare the data for that given use-case.
Data wrangling must be decoupled at this stage as if there was a unified approach to this task, not all models would perform at their peak. Memory is a sensitive area for this project due to the HEROKU server memory limitations, so it imperative that the data is leveraged for the strength of each particular model.

This is particularly apparent with a Transformer based architecture, where the goal of data wrangling is to reduce the amount of computation load taken by the model at input stage. The processing of the contextual embedding layer needs to reduce the memory intensity of each model, which is vital for the success of the optimum evaluation of each model.

3.4 Classification Models For Sentiment Analysis:

Three models (N.L.T.K; BERT transformer models; Flair) were considered and tested for sentiment classification compared against the following criteria:
1. Time – time taken to compute the input into output – to compute in 0(n) or better
2. Accuracy – can it predict at confidence level of 90% or greater.
3. Size – can the model be easily deployed onto a memory sensitive server having 512 MB and fed into Docker helping to distribute the model as unified architecture across the distributed system.

Docker is used to create a virtualisation container for the input to interact with the cloud and Kubernetes will help manage the output of the contained model while deployed onto the cloud server. Without Platfrom as a service (Docker); the model size would be inflated and consistently have an overflow; as memory usage would be at least twice as large. For example, BERT memory usage before contamination would be 512GB. This would result in server time-outs and data loss if deployed with these models.

We evaluate and measure all of the main classification methods described below against the above criteria:

3.4.1 N.L.T.K – Natural language Toolkit

This is a well-established model that has been used continuously for some time (Yadav, 2020) and as described in Section2.

N.L.T.K is a vector space model where each word sits in axis or dimension. (Yadav, 2020) the text is represented as a vector in this multidimensional space.
The method although popular and industry standard takes a lot pre-reprocessing to complete.
The process of preparing text for N.L.T.K sentiment classification tool are as follows:

1. Data wrangling
Is the process of preparing raw data in this case to be machine readable by programme(cumminsky, et.al,2019). For N.L.T.K. this means that each piece of text must be normalised in order for that text to be compare against each other when the text corpus is being tokenized.

2. Tokenisation
Is breaking down the given text into groups of either sentences, phrases, paragraphs or their individual words for interpretation depending on the programme being used. In this case NLTK feeds a BOW model meaning that each sentence or data point will be tokenized this also allows for stop words removal to take place simultaneously .(cumminsky,et.al,2019)

4. Lemmatization
Lemmatizing will take words from the previous step and set to group them by context or the actual meaning behind the word itself(cumminsky,et.al,2019) in terms of N.L.T.K this means grouping those

into underlying meaning in a bag of words model and counting their frequency of occurrence in the text.

5. Stemming
This involves removing the suffix of a word and reducing the word to its root word. For example, flying after stemming becomes both “fly” and “ing” . Where the word has now been reduced to its word root is more efficient to vectorize and become machine readable. N.L.T.K searches through the bag of words and groups them further into their stem.

6. Sentiment of the stemmed word
VADER style calculation is used to map frequency with intensity of the word return two sets of numbers both containing different calculations. One being polarity and the other being subjectivity. The sum of these consists of a number between -1 and 1. The following function calculates sentiment in VADER – note alpha is always set to fifteen in VADER which approximates the maximum value of x.

$$x \div \sqrt{x2} + \alpha$$

Figure 16: VADER calculation for sentiment analysis

3.4.2 Text processing for BERT

BERT differs from traditional models in that it only needs three pre-processing steps.

These steps are Tokenization, Stemming and Lemmatization. It takes less processing, as BERT can consider contextual embeddings (Ethayarajh,2020) meaning it can deduce context from a piece of text.

However, tokenisation in BERT is a larger process and consists of three areas which are token embeddings, segment embeddings and positional embeddings.

Therefore, tasks such as lower casing and noise removal may harm the model’s performance depending on what is in the text. As BERT uses contextual embeddings it may give a piece of text a different score if the text was not exactly as it was in the original source of the text.

for example, it could interpret the following differently:

text_one = “no I don’t believe this “, text_two = “NO!! I DON’T BELIEVE THIS “

Produces neutral with confidence interval of 98% for text_one and text_two was negative with 96% confidence interval for the uppercase text with special characters using the BERT Model.

The main part of data wrangling for a BERT model is removing white spaces and other characters which may add to length to the sentence without aiding the predictive power of the model.

An example of this may be the following:

“This is unbelievable ….. Messi Is leaving Barcelona !!!! “

As the BERT model uses contextual embeddings it would assign every character a token after tokenisation. Empty spaces and full stops would hold their own token and would be fed back into the autoregressive model of BERT. This means unnecessary computation is carried while not adding further value or insight into the model.

Data wrangling is still needed for BERT or the transformer model, but the focus of the cleaning process is on how to decrease unnecessary autoregressive steps while maintaining a high predictive power.

After prepossessing the piece of text becomes:

“This is unbelievable. Messi Is leaving Barcelona! “

Both produced the same result (negative 98%) from the trained model. However, the second piece of text ran in 0.27 seconds and the initial text ran in 1.0 second overall which was the time of the unprocessed piece of text.

Instead of implementing stop words removal; some evaluation of the text in terms of what is important to the overall message must be analysed to reduce or remove unneeded autoregressive.

This was implemented using Spark and using regex built in functions regex.sub, regex.findall and regax.split. Spark would take these functions and apply as soon as input was retrieved from the A.P.I to remove future tokens which potentially would not add any value to output.

Diagram for Distil Bert Data wrangling

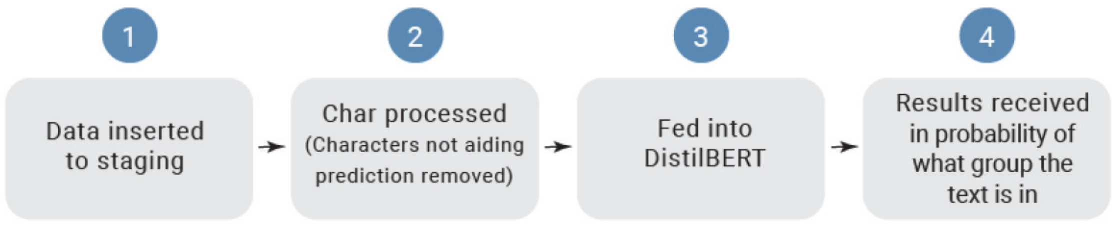

1
Data inserted to staging
2
Char processed (Characters not aiding prediction removed)
3
Fed into DistilBERT
4
Results received in probability of what group the text is in

Figure 17: Data wrangling

<u>Flair</u>
Flair N.L.P is built on-top of Pytorch and has access to various approaches for contextual embedding all including BERT Character Embeddings and uses contextual embeddings for string classification tasks (Saxena,2019).

It essentially acts as a container with access to these models through an easy-to-use Python library. It can be affective depending on the use case but it depends on the task at hand. It may be applicable in this use case as Twitter data has been referenced as being trained on (Saxena,2019). Flair is an out of the box model meaning it has been pre-trained and is ready for production. (Saxena,2019)

In common with BERT, the same processing steps must be taken with a focus on reducing computation that will not add value to the prediction.

3.5 Python Libraries Used

There were many overlapping Python libraries needed including the following:

Json, Tensorflow, Numpy, Pandas, Pysopg2, Pytorch, Math and Pymongo.
Json: used as an intermediary data storage point.
Tensorflow: for its machine learning and A.I. based libraries.
Pandas: to work with data while it is being trained.

Pycopg2: the API layer for PostgreSQL.
Math: to help complete Mathematical elements of the code involved in training.
Pymongo: the Python API to access MongoDB.

Each model needed a Python library not appropriate to the others:
N.L.T.K :
NLTK, Sklearn , Scapy.
NLTK as this library has access to an efficient VADER classficiation tool.
Sklearn is used for its ability of statistical modelling and regression analysis, it was used here for its random forest decision tree algorithm to help group sentiment based on the VADER score.

Scapy is a network manipulation tool used to distribute the work over various nodes in the distributed framework.

The N.L.T.K library was applied to the data after the six steps of processing the data outlined in section 2.4.4. The result of the VADER calculation is then sent to Sklearn libraries for staging for machine learning; before being sent into random forest libraries, the data is normalised in order to be interpretable by Sklearn random Forrest algorithm, where each data point is transformed into a node.

After this has been actioned new data or training data can be inputted into the random Forrest algorithm where it also goes through a normalisation stage before classifying the text inputted.

It is then contained within Scapy in order to be available on the Spark nodes working on the distributed network.

BERT:
The following specific Python libraries were used:

HuggingFace, Keras , Tensorflow-gpu
HuggingFace for access to the untrained BERT MODEL
Keras -> to use as a front end to access and contain the trained model
Tensorflow-gpu -> to use a version of Tensorflow that is purpose built for use on a GPU

Hugging face was used to access the untrained BERT model in a container and to install onto the local environment. Keras was used in order to act in between the inputs layer and the model in order to fine tune the model with pre-set instructions before being trained. Koras also worked alongside Spark in order to pass each epoch through worker nodes along with Spark where each epoch's result could be updated to the master node. Tesnforflow-gpu was installed in-order to take full advantage of the GPU made available through Google Collab .

Flair:
Flair , Genism
Flair library is the access point for the standard BERT (meaning pretrained model) model
Genism helped to manage the pulling of data from the inputs into the Flair model

Genism acted as a wrapper to access the flair model. It contained the data in a format initially instructed in order to prevent less data cleansing and staging when the data is received back whilst also introducing normalization with the other data that was received.

3.6 Implementing the Code

The approach taken was to ensure that each model was run in its own sandbox and that one model running on the GPU did not slow down another. Each model was tested in its own environment on Google Collab which is an executable file on a distributed system which can be controlled locally, with access to its own isolated worker nodes independent of others in the system.

The code of each model was given its own fine-tuned environment to ensure the most effective performance would be achieved.

For BERT-based models this meant HIGH-RAM storage to enable the model to update itself as often as possible. The purpose of Transformer based architectures is to always reduce the amount of autoregressive steps that could occur, as the runtime is usually 0(N) making it an efficient runtime as it predicts the input length.
For Flair this meant having backup worker nodes to ensure a stable internet connection throughout the testing process. This required an observer class to watch the worker node assigned to ensure a stable connection was always available. It would send requests to the Node every thirty seconds during testing and if it did not receive a response in the next thirty seconds it assigns a backup node to the current worker node.

For NLTK this required an approach where each data wrangling stop had a looping overhead and data is stored in bytes, as the VADER model is memory intensive and can be n*2 in terms of runtime. This means that the lead up to processing this must be kept as efficient as possible in order to allow the algorithm (VADER) to perform in an acceptable time frame.

3.7 Testing Approach

Analysis of each method was tested against three selection criteria:

Each model was tested on some text data, each having processed for the specific model. The one needing the least pre-processing was Flair, as this is already a trained model. This would mean that if Flair was selected, it would have the quickest production time.

BERT took the longest to prepare as it needs to be tweaked to be effective. However, after training, the output was over the memory threshold.
3.7.1 Performance & Training of N.L.T.K.
N.L.T.K took a similar amount to produce as BERT. However, although it took less memory compared to the other two both; it's run time and accuracy were far below what would be required forsentiment analysis. This means N.L.T.K was eliminated at this stage of the analysis.

Criteria	Flair	DistilBERT	N.L.T.K
Speed	0(n)	0(nlogn)	0(n*2)
Size (After containerization)	199 MB	505 MB	421 MB
Accuracy	93%	96%	81%

Figure 18: N.L.P model comparisons

3.7.2 Performance and training of BERT:

To maximise results, a BERT model was trained on Google Collab with an access to a GPU.
As mentioned previously BEST base uncased was used i.e. DistilBERT, as it is a smaller model that takes much less memory due it's twelve encoding layers instead of twenty four compared to regular BERT.

Four trained sessions were carried out with their hyperparameters listed below

parameters	run_time_1	run_time_2	run_time_3	run_time_4	run_time_5
epochs	5	10	14	18	22
batch size	20	40	60	80	100
max_len	240	240	240	240	240
random_seed	30	35	40	45	50

Figure 19: Hyperparameters tested

The optimal runtime was run_time_2 with the following training results. Anytime past ten epochs introduced over training and anything below ten epochs did not guarantee accuracy of the overall model.

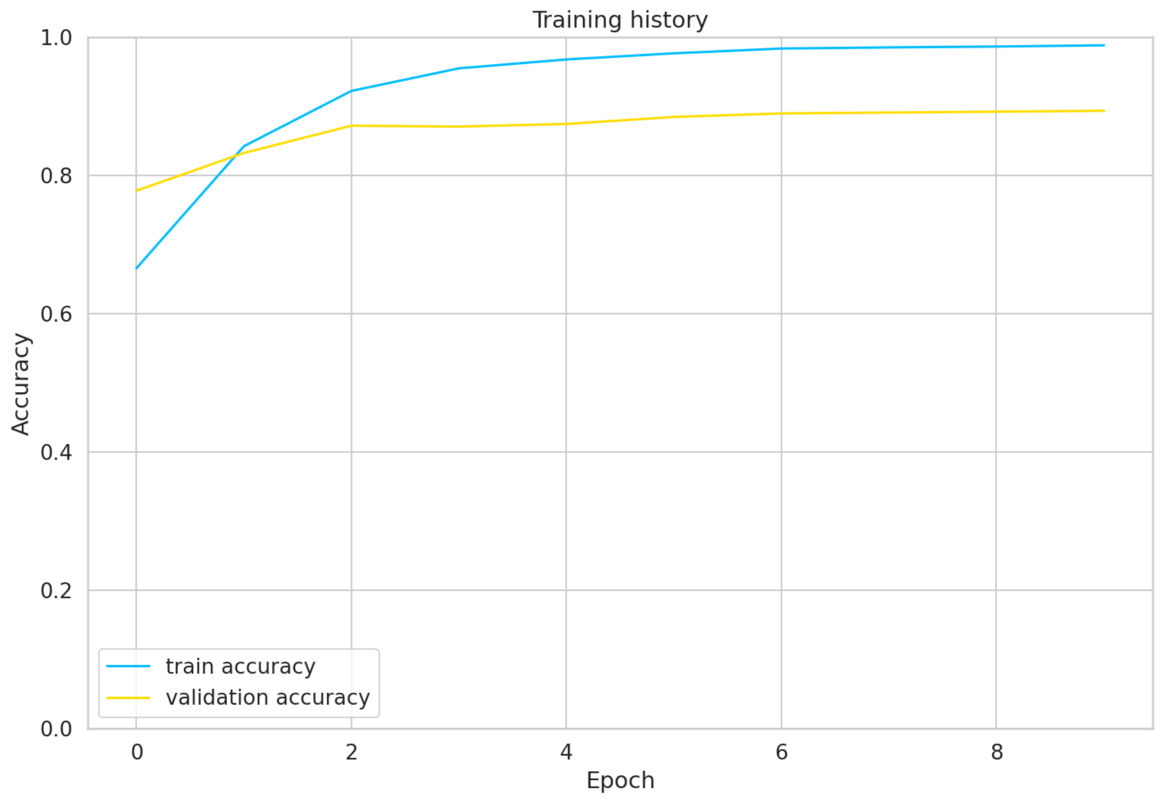


Figure 20: Training result of run_time_3

Confusion matrix for BERT – as you can see from the below confusion matrix, it was successful at group text between selecting sentiment as Positive, Neutral and Negative. From the production model the neutral weighting will be removed. It was included as recommended from the original paper written by Google research Team, 2018 (Devlin,et all, 2018) and edited in production. This was to ensure the training accuracy was as high as possible.

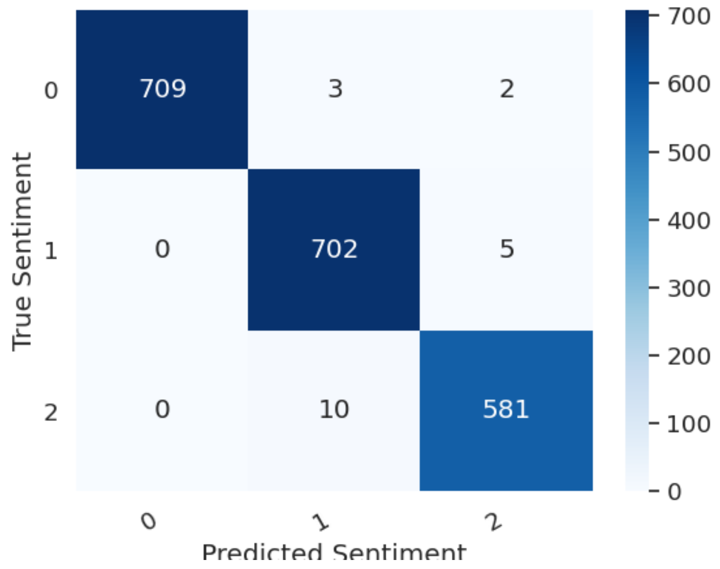


Figure 21: confusion matrix for chosen LSTM model

BERT is the most accurate model. However, it’s shortcomings are its memory usage and computational time. It had an n(logn) computation time and reached the maximum memory limit. This would mean that it may be called upon more than intended and may cause the system to be generally slow. However, as this portion of the process has been allocated more time for being more intensive than other parts; it is a e compromise in order to achieve the accuracy needed. For context, DistilBERT scores sentiment as either (0= positive, 1 = Neutral , 2 = Negative).

Testing Flair

Flair in theory had the best results of all models combined. However, it is also the most unreliable, as what makes it take up little memory, is also its weakness. It needs a stable internet connection to connect through Keras to its pre-trained model. There is a risk of a node failing during the connection would result in a data loss, as each node holds one unique data point, if that node fails that data would be lost when the node fails during processing.

3.8 Solution Chosen based on Test Results:

As both Flair and BERT models have significant strengths and weaknesses it has been decided to integrate both into different stages of the cloud stack. Where BERT would be the initial model used; if it takes up too much memory a caching layer would then store the unprocessed data. Flair would then be used, if a data loss occurs when calling Flair through the cache, there is a record of its previous state before being calling the Flair model. This means both models strengths can be fully utilised in the overall system architecture. Below diagram shows how both models have been integrated together in one stack. This also sits on the same cloud server as the data acquisition stage on HEROKU and run every hour on a cronjob.

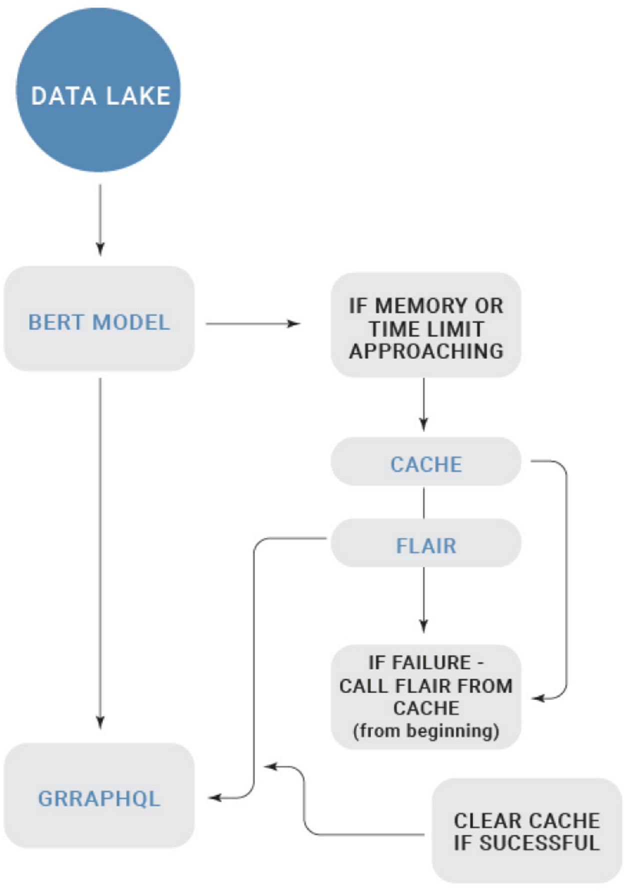

DATA LAKE
BERT MODEL
IF MEMORY OR TIME LIMIT APPROACHING
CACHE
FLAIR
IF FAILURE - CALL FLAIR FROM CACHE (from beginning)
GRRAPHQL
CLEAR CACHE IF SUCESSFUL

Figure 22: How the model is accessed and contained on the cloud

Some potential problems with this approach are, the risk of a continuous loop between calling Flair and sending a new batch job from the beginning of the cache. To mitigate this, if the connection has failed consecutively (twice in a row) the cache is split between two halves and two new worker nodes are requested via Spark in order to process the data and clear the current cache. The original node calling Flair is also replaced with a new node, in order to ensure that node has not failed. Every time a connection is made to the data lake a new node is requested from Spark before being fed into the BERT model, this node will determine if there is enough load or too much load thus pushing what it estimates it cannot manage to the cache.

3.9 Normalised data

Certain data points were normalised using minmaxScaler in Java.
The two main parts of data that needed to be normalised are the dates and the prices from Binnace (the Crypto price feed used).
As the model will be grouping certain time frames together, the time stamps need to be normalised in order that efficient grouping and clustering of text can be carried out.
The normalisation of prices will take rolling mean average for the time period which is trying to be predicted. i.e., for a 24-hour period the RMA for that period will become the price fed into or associated with that data point alongside the ratio of positive to negative sentiment for the time period.

Look back period
The lookback period will start 365 days in the training stage and the model will point to what is an optional lookback period for one day forward prediction.

3.10 Training model & Model Inputs:

The model acts as a python dictionary where the key is the timestamp and contains a list with two elements for that time period. One being the average percentage of positive posts and the other being average rolling mean of the price of Bitcoin for the time period as the key. The time period in this case is defined as for example (3/05/2021 : 00:01 -> 04/05/2021 : 00:00).

$$(total\ posts \div\ total\ positive\ posts\ ) - (total\ posts) * 100$$

Figure 23: overall percentage calculation of positive posts

3.11 Splitting the dataset

The data has been split into training, testing and validation datasets. These three datasets are to help train the classifiers on different datasets. There is a concept known as cheating, whereby using the same dataset used for testing and validation is using test data multiple times to determine the best parameters. It should instead be decided on a validation dataset. The datasets have been broken down into the following sequences for training, testing and validation over different time periods:

Three separate dataset -> (2017-01-2018-01), (2018-02,2019-5),(2019-6-2021-8)
Training dataset is (2019-6-2021-8) sentiment and prices.
Testing dataset is (2018-02,2019-5) sentiment and prices.
Validation dataset is (2017-01-2018-01) sentiment and prices.

3.12 Hyperparameters

The hyperparameters are parameters whose values are used to control the learning process. Values of all other parameter types (typically node weights) are derived via training.

Hyperparameters cannot be inferred while fitting the machine to the training set because they refer to the model selection task, that should have no influence on the performance of the model but affect the speed and quality of the learning process (Radhakrishnan, 2017).

In terms of hyperparameters used for LSTM model figure 22 shows the hyperparameters used in the model tested on different instances trained on the cloud and the optimal training was selected as the model.

3.13 Model outputs/model environment

To easily distribute the model across multiple nodes with an assigned GPU the model used a Keras API with a TensorFlow backed, which means Keras processes the input for the model and contained it and would then pass it to the TensorFlow which trains the model.
Each model was given its own memory space to run on the cloud and then dispersed across nodes in the network.
This allowed for multiple computations happening simultaneously allowing for a short turn over in training the model.
All of the computation was done on Google Cloud and scripts being actioned through Google Collab.

4 Model training and results :

4.1 Model training results

The model was trained with a range of hyper parameters in order to obtain the most accurate model. The epochs and batch size will be trained on various iterations ranging from 100 to 500 epochs and a batch size starting at 60 epochs and increasing by 20 epochs for every iteration of the LSTM model training.

The optimization algorithm chosen was the Adam optimizer, designed to update network weights iteratively during the training process (Brownlee, 2017) meaning it could update weights between layers in the LSTM model using a single alpha during training. It did not need to update the learning rate; leading to a more efficient training process (Bronwlee, 2017). This also introduces the ability to counteract over fitting while training. Overfitting is when a model performs exactly in-line with the training data inputted, meaning it cannot accurately perform on unseen data (IBM-DS, 2021). A cross entropy loss function was used as it can easily group and reduce the weighting of bad predictions in an LSTM model, by measuring the difference between two probability distributions for a set of events (Bronwlee, 2019).

The LSTM model chosen would run on to find the optimal hyperparameters.

parameters	run_time_1	run_time_2	run_time_3	run_time_4	run_time_5
Epochs	100	200	300	400	500
batch size	60	80	100	120	140
random seed	30	35	40	45	50

Figure 24 optimal hyperparameters

After running each experiment on a separate thread on Google Collab the optimal hyperparameters were run_time_3 from the table above.

Below shows the training performance over those epochs.

Below is the graph ‘Results of run_time_3 on the LSTM model’ which shows the result of training on run_time_3 where y-axis shows the percentage of loss at each stage and x-axis shows the current epoch being run. The train tag in the graph explains the performance of the model in the training data through each epoch and the test tag actually shows the amount of loss emitted through each epoch.

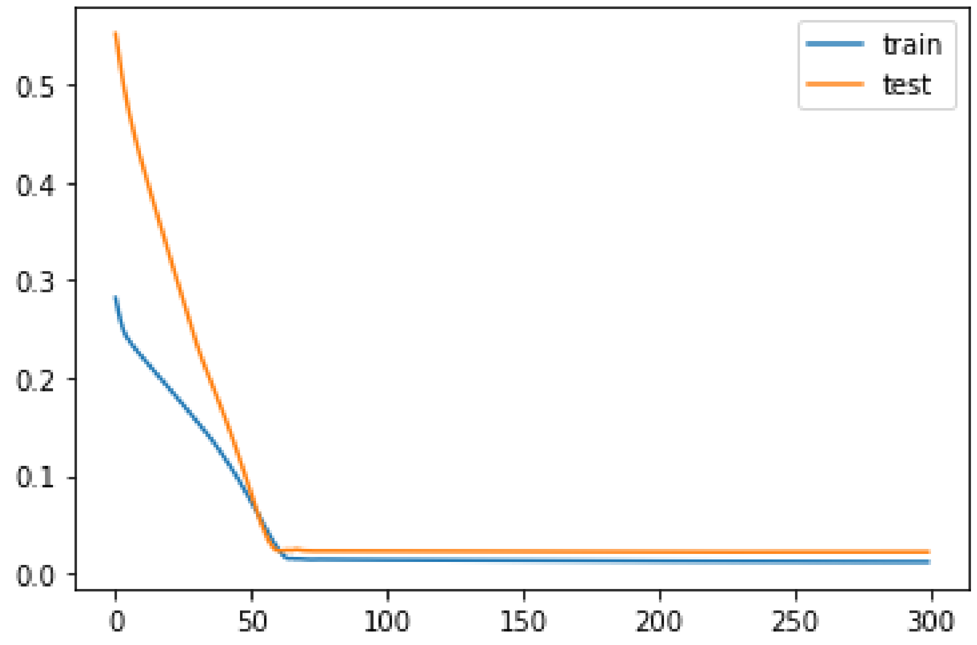


Figure 25: Loss over training results for run_time_3

The accuracy of the model predictions on the training set and using the LSTM trained on run_time_3:

Predicting Prices using Run_Time_3

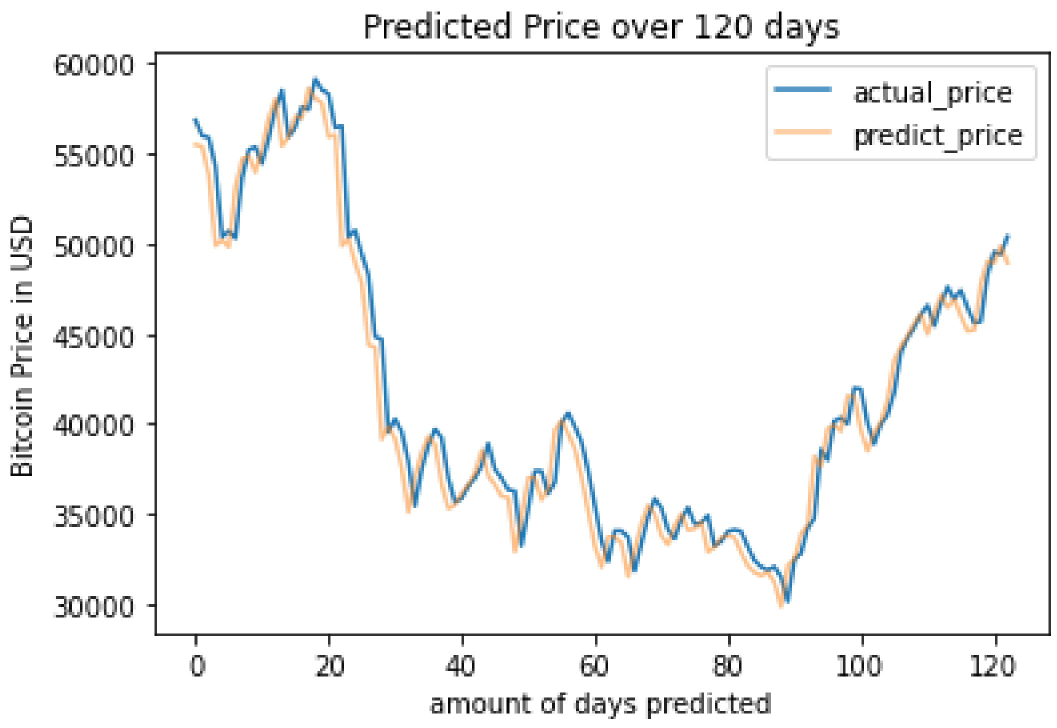


Figure 26: overall predictions power of run_time_3

Initially the LSTM model was intended to take a two hundred day look back period. To make the LSTM model more memory efficient; an input of 120 day lookback period was used instead. This reduced memory usage down by 15%. After making this adjustment the model was less accurate around the actual price predicted. Making the correct price prediction with a confidence interval of (15%) 38% of the time. It predicted accurately whether the price should go up or down accurately. It correctly predicted the price movement 78% of the time whether an increase or decrease would happen in the next 24 hour period.

The graph below is a ‘Model performance on test dataset ‘
where 60k is 60,000 American dollars (price of Bitcoin or the index) on the y-axis and the x-axis represent the timeframe chosen at random.

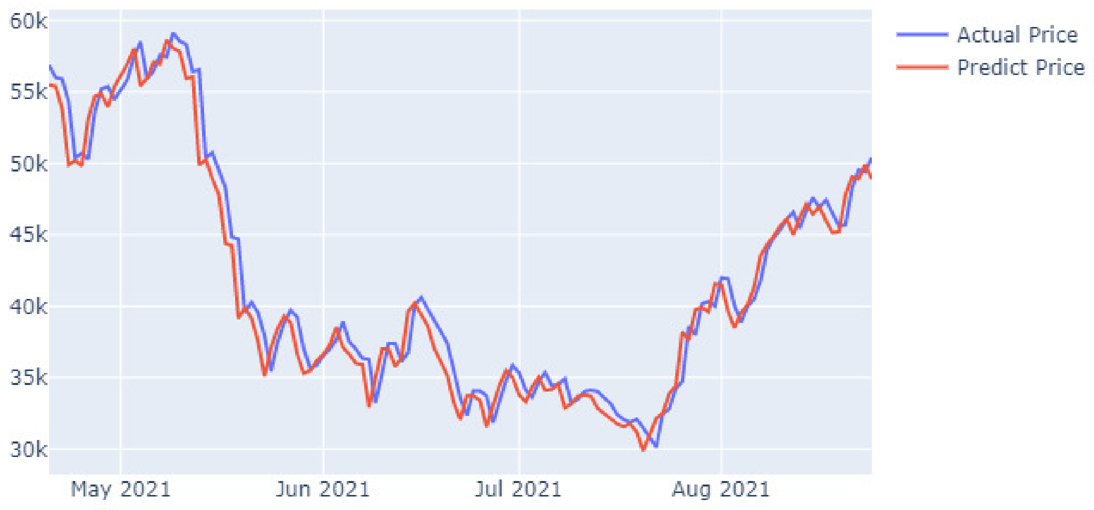


Figure 27: Performance on test data from periods May 2021 – Sept 2021

The model used to test data from when the dataset was split earlier in the process and then run through the algorithm. It ran data for the period from the 3rd of May to the 3rd of September being 120 days in all.
Similar to the results of the training dataset, the model struggled to predict the actual price but performed well on predicting the future price trends for the next 24 hour period.
It correctly predicted the trend 68% of the time which is significantly lower than the training data. However, this time period was chosen to test the rigor of the LSTM model as the time period was the most turbulent in terms of significant price drops in the last ten years (Harr, 2021). Overall, the model performs well on predicting price trends given that there is a lot of price volatility in terms of significant price movements happening quickly. The predictions for the time period are lagging slightly behind when price movements happen but mostly act as a lead indicator from June 2021 to August 2021.

4.2 Model accuracy

Metrics on performance over each dataset was analysed; in order to gauge total model performance over time.

Time Period	Overall Correct Predictions
2017-01-2018-12	72%
2019-01-2020-5 (Training data)	67%
2020-5-2021-6	62%

Figure 28: overall prediction for sole Bitcoin predictions across all datasets

From the above table, some years the model was more effective at predicting than others. This will be investigated further below. The below table shows the results for solely predicting the correct price movement, meaning did it predict correctly or did it not succeed in the prediction.

Training index predictions over each time period

Time Period	Overall Correct Predictions
2017-01-2018-12	53%
2019-01-2020-5	60%
2020-5-2021-6	58%

Figure 29: overall predictions for index predictions across all datasets

As the table illustrates the index prediction is much weaker then predicting the price of Bitcoin. There are multiple contributing factors that will be examined below.

4.3 Overall index predictions accuracy

As indicated earlier in the project the results show that Bitcoin price movements are a good lead indicator for index price movements, which is a tangible measure for predicting a change in a data series (Marr, 2020).

The index is made-up of eight of the main crypto currencies determined by market cap, including Bitcoin.

The testing dataset was inputted into the model, giving the output below for 120 days between 03 May 2021 and 03 September 2021.

The chart below compares the actual index price of crypto-currencies over time, against the predicted price of Bitcoin and the predicted price of the Index using the LSTM model.

The below chart shows the 'Overall price index prediction',
the y-axis is starting at 60k which represents 60,000 USD dollars (value of the cryptocurrencies) and x-axis representing the time frame chosen at random.

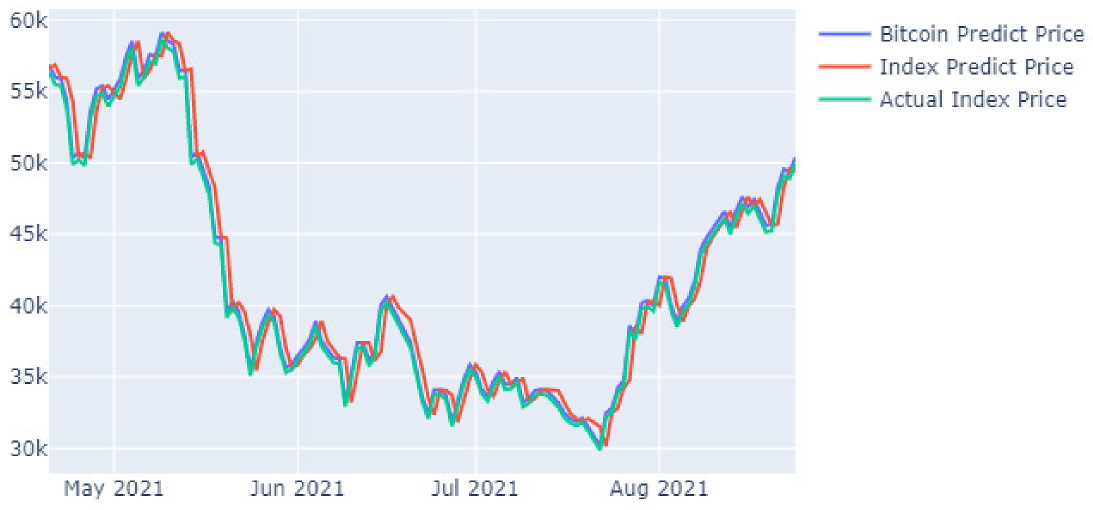


Figure 30: overall index and Bitcoin prediction compared to actual index price

From the above graph, the predicted price of Bitcoin from the model was a strong lead indicator of the index price movements (price increasing or decreasing for the following day). As expected, the model could not successfully predict the actual price of the index throughout the time period. This was tested on the testing dataset as the time periods are still from 03 May 2021 to 03 September 2021.

4.4 Bias in model

Upon further testing of the model, the test dataset was applied to the model for the 120 day period 03 May 2017 – 03 December 2017 to have an element of like-for-like comparison. This showed that apart from the initial testing dataset there was a bias towards the index predictions following the price of Bitcoin far closer than the predicted price of the index.

The below chart 'Measuring performance compared to the actual index and the performance of the index predicted price',
examines the performance of the Index price, Bitcoin price, index predicted price and the predicted price of Bitcoin.

The y-axis shows the amount in US dollars starting at 8,000 USD and the time period chosen is along the x-axis (June 2017 – Nov 2017)

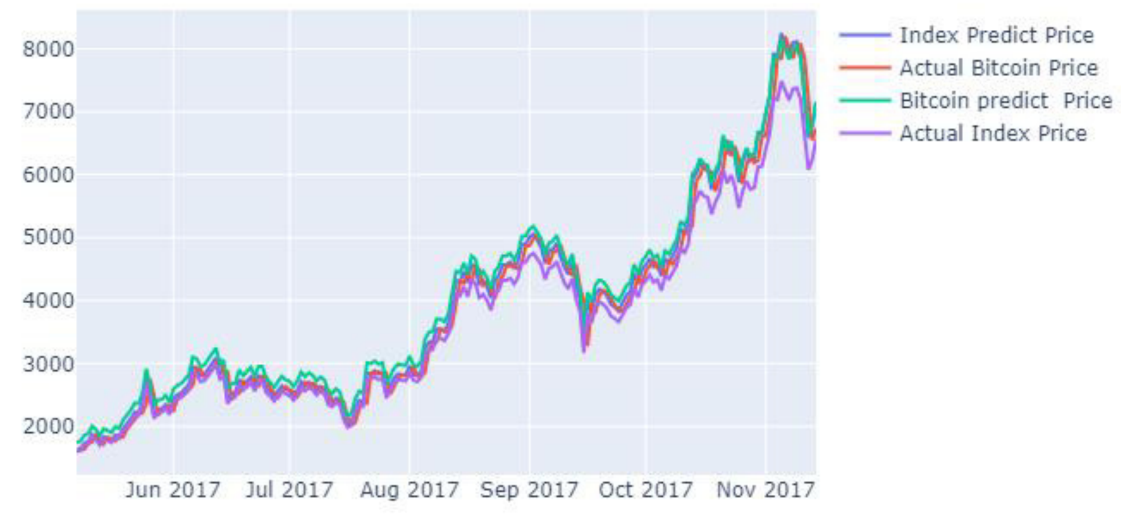


Figure 31: detecting bias when Bitcoin outperforms the index

As Bitcoin was chosen to be a lead indicator to predict the index price, a time period was analysed where Bitcoin outperformed the other cryptocurrencies in the index: June 2017 – November 2017. For the total time period observed, the predicted price movement matched the actual price movement 49% of the time. However, if the time period is split in two datasets (from June 2017 – September 2017 and October 2017 - November 2017) there are different results. In the first dataset, the index price movements were predicted correctly 61% of the time. The second dataset where Bitcoin outperformed the other cryptocurrencies in the overall index prediction was correct 14% of the time; as opposed to 71% of the time when solely predicting the price movement of Bitcoin. This shows that when Bitcoin is outperforming other cryptocurrencies significantly this model has limitations in predicting the total index performance of the model.

4.5 Evaluation of the model used (LSTM):

Using random dates, we tested for a very small time period, (8 hours); social media sentiment as a parameter with the price of Bitcoin for that period and only inputting that price as a parameter.

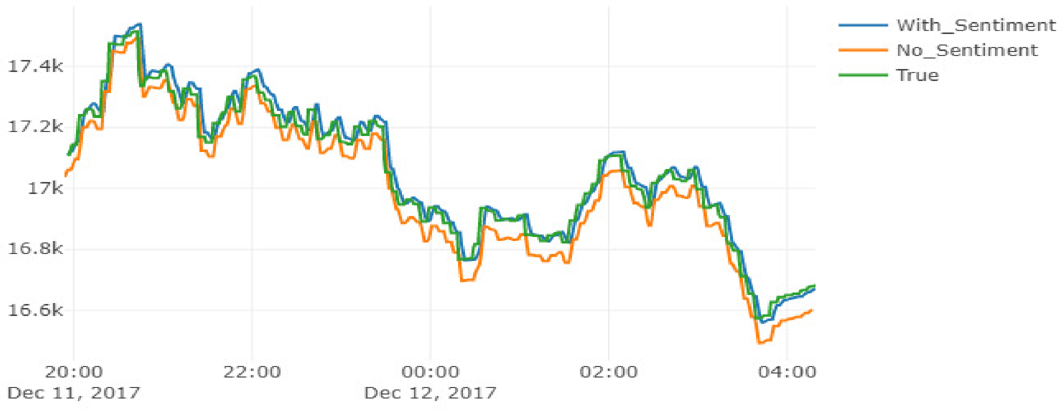


Figure 32: strength of sentiment as a feature

Random data was chosen to predict from the 8 hour period of (Dec 11, 20:00 -> Dec 12, 04:00 , 2017) and along with a 120 day look back period. The model predicted the price with and without the sentiment analysis feature. As the graph shows when the sentiment score is not included as a feature it outperforms when sentiments score is included as a feature. Using historical prices only; acted as a lag indicator which unlike a lead indicator is a tangible measure of data points in the series after the data point has changed (Marr, 2020).

5 Future work

Since starting this project, it has become apparent that there are services using social media and Google searches as a lead indicator to predict the price of Bitcoin and other cryptocurrencies. We would in future add Google search data, together with social media sentiment analysis, to test the prediction of the Bitcoin price.

It would also be worth including the total index price of cryptocurrencies, rather Bitcoin for training the model.

The project, although used large amounts of historic information, was limited in terms of memory usage and a 15 minute delayed price feed. It would be worth evaluating both data sets with deeper time series information (in minutes, rather than 15 minutes).

The project used a number of machine learning classifiers for sentiment analysis. It would be worth testing various lexicon approaches using standard datasets of three proprietary data dictionaries LIWC , (Liu 2012) GI and Hu-Liu data dictionaries. This approach would need to evaluate the technology stack needed in order to run these lexicons against the large volumes of historic data. Lexicon methods to sentiment analysis have a number of drawbacks, but when properly tuned can deliver high accuracy rates. It is not clear using this method, if accurate price predictions could be made, rather than just trend predictions.

In addition to machine learning approaches, some deep learning model might be deployed. Deep learning would need more extensive supercomputing, however, it is far superior at undertaking analysis of video and images, rather than just text. This might change some of the sentiment analysis scores.

Hugging Face auto NLP model, might also be considered, in order to see what model would be a best fit for the data available, rather than trial and error. The auto NLP model would have helped select the most suitable model based off the criteria fed into the model initially and returned a trained model best suited to the data and the desired result (Hugging Face 2021).

There is an analysis method called the fear/greed index, (Bronwlee,2020) that is used as a lead indicator of investor sentiment. This could also be applied to test the models against this index, price movements and against sentiment analysis measurements themselves.

6 Key Challenges Encountered & Learning

- Machine learning on the cloud introduces an added layer of complexity in terms of a container's ability to access the model, due to memory management issues. Manipulating the models was less difficult than the many steps that needed to be taken to constantly manage the memory. It proved to be such a problem, that low memory alternatives should have been considered at the very outset. This is despite the fact that memory availability was one of the initial model selection criteria. The memory usage of the models was also higher than anticipated during the actual implementation of the project.
- All forms of BERT have much higher memory usage than all other models considered. The BERT models introduced a level memory issues that would make simpler traditional NLP method such as feature engineering a serious alternative to consider.
- Using a graph database for data-lake staging introduces an un-necessary layer of complexity when indexing the data at a later stage of the project. Instead of the graph database we could have used a tabular data format to allow for more efficient indexing.
- The rest API used was "over-serving" i.e., too much data was returned and slowed down the program in its later stages. Using a graphQL schema instead would have been more suitable than the restAPI. In a memory sensitive environment, it had a knock-on effect on all parts of the process.
- Spark was an effective way of splitting work across the network but alternative methods should also have been considered. Overall Spark is more suited to big data tasks from a noSQL data source. For example, alternative methods would have Genism, Spacy and regular nodes, which all use less memory and are easier to deploy.
- Google trends is an API whereby Google search information is available and trended, so that searches relating to Bitcoin, or cryptocurrencies could be used as an additional feed for regression analysis. The only issue is that Google charge commercial rates for this information.

7 Results & Conclusions

We examine the results in the light of all of the research objectives as defined in Section 1. The following are the results for each of the research objectives:

• To identify a correlation between sentiment of raw text regarding cryptocurrency from Twitter and Reddit and the future price movements of Bitcoin and other crypto currencies.

The results show that using the dataset used against the machine learning models applied, that sentiment analysis is reasonably good indicator 63% of **future price trends** of Bitcoin and other cryptocurrencies that are either positive or negative.

• If there is a correlation between the sentiment analysis and Bitcoin prices as above, under what conditions does it have greatest impact?

The results show that using the dataset used against the machine learning models applied, that sentiment analysis is not a reliable indicator of **future price predictions** of Bitcoin and other cryptocurrencies within a 24 hour period.

• To explore if there is any difference between social media sentiment of text from Twitter or Reddit in terms of sentiment or result in Bitcoin price.

There was no measurable difference between the two datasets that were used.

• To show causation between Bitcoin price fluctuations and movements in other cryptocurrency prices.

It was noticed that in prices of high price volatility (+/- 50% changes) that the models do not work. When there is low volatility (+/-5%) the models work with 15% confidence intervals.

• To build accurate sentiment classification models using BERT and other models to group the text between positive and negative sentiment.

The optimum BERT model applied was DistilBERT, as it gave reasonably similar results 63% as the other BERT models but had much lower 30% memory usage.

• Create an accurate classification model which can predict if the price will go up or down depending on the amount of sentiment posts at a given time.

This was achieved with an overall accuracy of 69% across all datasets. The project has proved that social media sentiment on Bitcoin and cryptocurrencies is indeed a lead indicator, in that it can predict upward or downward movements in price. It cannot predict future prices, or price levels. This makes it a reasonably reliable tool to predict price trends. These conclusions are based on the limitations of the technology used, the data made available and also on the limitations of the models themselves.

8 References

Polosukhin, Illia; Kaiser, Lukasz; Gomez, Aidan N.; Jones, Llion; Uszkoreit, Jakob; Parmar, Niki; Shazeer, Noam; Vaswani, Ashish (2017-06-12). "Attention Is All You Need.

Nishta Jatana , Sahil Purl , Mehak Ahuja , Ishita Kathuria , Dishant Gosian , 2012 , A Survey and Comparison of Relational and Non-Relational Database , International Journal of Engineering Research & Technology (IJERT)

Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, 2018, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding , Cornell University

Jason Brownlee , 2017 , improving performance in deep learning models , Machine Learning mastery Date accessed : 08/02/21 , weblink : https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

Jason Brownlee, 2017 , Cross entropy in classification problems , Machine Learning Mastery , date accessed : 08/02/2021 , weblink : https://machinelearningmastery.com/cross-entropy-for-machine-learning/

IBM DS , 2021 , How to prevent over and under fitting in your model ,IBM data science blog , IBM , date accessed: 08/12/2021 , web link: https://www.ibm.com/cloud/learn/overfitting

Ryan Harr , 2021 , The Price of Bitcoin Keeps Dropping. Here's What It Means for Investors, According to Experts , nextadvisor , data accessed: 09/24/2021 , web link: https://time.com/nextadvisor/investing/cryptocurrency/Bitcoin-crash-continues/

IBM DS , 2021 , How to use and implement a Recurrent Neural Network , IBM , date accessed: 08/12/2021 , web link: https://www.ibm.com/cloud/learn/recurrent-neural-networks

Whittle , 2020 , using an ARIMA model for forecasting problems , data accessed: 08/12/2021 , web link: https://medium.com/coinmonks/predicting-the-Bitcoin-price-using-seasonal-arima-model-4df3f3a1cd44

Kawin Ethayarajh, 2020, BERT, ELMo, & GPT-2: How Contextual are Contextualized Word Representations? , Stanford A.I blog

Bing Liu, May 2012 , Sentiment Analysis and Opinion Mining, Morgan & Claypool Publishers,

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, Manfred Stede,2017, Lexicon-Based Methods for Sentiment Analysis, . Department of Linguistics, Simon Fraser University

Ingo Steinwart, Andreas Christmann , 2008 , Support Vector Machines Information Science and Statistics , Springer Science & Business Media,

Selva Prabhakaran,2018, Cosine Similarity the math behind the formula , machinelearninsplus.com, data_accessed: 20/09/21, weblink: https://www.machinelearningplus.com/nlp/cosine-similarity/

Sharoon Saxena , 2019 , Introduction to Flair for NLP: A simple yet powerful library , Analytics Vidhya , date_accessed: 20/09/2021

Adam Brownlee, 2020 , Be Fearful when others are greedy , Investipedia , data accessed: 02/07/21, weblink: https://www.investopedia.com/articles/investing/012116/warren-buffett-be-fearful-when-others-are-greedy.asp

Hugging Face , 2021 , AutoNLP , Hugging Face , date accessed : 01/10/21 weblink: https://huggingface.co/docs/autonlp

Bernard Marr , 2020 , What's The Difference Between Lagging And Leading Indicator , forbes.com , data accessed: 01/10/21 , weblink: https://www.forbes.com/sites/bernardmarr/2020/10/23/whats-the-difference-between-lagging-and-leading-indicator/?sh=7a4db65009a4

Hang Yuan , Jin Wang , Xuejie Zhang , 2018, YNU-HPCC at Semeval-2018 Task 11: Using an Attention-based CNN-LSTM for Machine Comprehension using Commonsense Knowledge , Association for Computational Linguistics

Xiaofeng Yuan,Lin Li,Yalin Wang,Chunhua Yang,Weihua Gui, 2019 , Deep learning for quality prediction of nonlinear dynamic processes with variable attention-based long short-term memory network , The Canadian Journal of Chemical Engineering

Clear Barrett, 2021 , Why young investors bet the farm on cryptocurrencies, The financial Times

Russell Investments, 2021, You can't stop the wave, but you can learn to surf , Russell Invesments.com, data accessed : 01/10/2021, weblink: https://russellinvestments.com/ca/resources/cycle-of-investor-emotions

Uma Devi , 2014 , The role of emotions in Investment Decisions, National conference, JNTUH