Developing an automated cryptocurrency trading bot using Twitter data and supervised machine learning

By

Luke Magnet

Supervised by

Phil Smith

A thesis submitted to
the University of Birmingham
for the degree of
MSc COMPUTER SCIENCE

School of Computer Science
University of Birmingham
January 2022

ABSTRACT

This purpose of this project is to test the feasibility of creating algorithmic trading software that makes automated trades, based on the outputs of supervised machine learning models that use Twitter data as their inputs. This is done by firstly exploring whether Twitter can be used to effectively predict short-term price movements of Bitcoin, Ethereum, Cardano, and Dogecoin. Those models which offered the most promising results were implemented in the algorithmic trading software and their performances analysed against a buy-and-hold strategy. Price movements are infamously volatile in the world of cryptocurrency, so making buy and sell decisions at the right time can greatly affect an investor's returns. Supervised machine learning models were created using WEKA, developed by the University of Waikato, in which sentiment analysis, word embedding, and tweet volume attributes were extrapolated from Twitter data. While Twitter is increasingly being used as a tool to analyse general market sentiment within both traditional and non-traditional financial markets, it was found to not be a good indicator of price percentage change, or price direction within almost all of the models created in this study. The one exception to this was a model that used word embedding data to predict the hourly price direction of Dogecoin with 79.17% accuracy. These findings do not necessarily indicate that Twitter data is useless, just that they are not a good indicator of price direction when looked at alone; other models not used in this study such as LSTM may offer more promising results. Despite the machine learning models being somewhat weak, the algorithmic trading mechanism itself runs successfully, and is an effective proof of concept. That being said, the trading algorithm used needs improving. The software and its code can act as a useful tool and/or framework for those wanting to build their own algorithmic trading systems, especially those new to the field of machine learning looking to build applications using machine learning tools such as WEKA.

KEYWORDS

Cryptocurrency; blockchain; algorithmic trading; machine learning; sentiment analysis; word embedding; supervised learning; social media; Twitter

ACKNOWLEDGMENTS

I would like to thank my supervisor, Phil Smith, for his help, guidance, and support throughout the project process. I would also like to thank those staff at Heroes Bar in Worcester for covering so many of my shifts so that I would have the time to work on this project. A special mention must also go out to Billy Markus, for personally answering a lot of the questions I had relating to Dogecoin and Dogecoin Core.

Contents

List of Figures

List of Tables

Chapter One
Introduction

The prices of Bitcoin and other related cryptocurrencies have risen exponentially since their inception, meaning those that invested in them early have made lucrative, exponential returns on their investments. As the popularity of cryptocurrency in general rises, some have referred to their use as the start of a new censorship-resistant financial era, while others have denounced cryptocurrency as a scam, calling it a speculative bubble (Chaim and Laurini 2019). This study makes no comment on the effectiveness of cryptocurrency as either a medium of exchange or an investment overall, though offers an overview of its underlying technology, its potential impacts, and use within an automated trading framework.

1.1 Research Aims

The main aim of this study was to produce a piece of algorithmic cryptocurrency trading software that used machine learning to make automated buy, hold, and sell decisions. The machine learning models in question would use quantitative data from Twitter using sentiment analysis and word embedding techniques, as well as data concerning tweet volume to make cryptocurrency price predictions.

While the overall goal of this research is realised through the software creation phase described in Chapter 5, the most complex and detail-oriented part of this project lies in the actual setup of the underlying machine learning models, due to the use and study of language-based metrics. Hence, cryptocurrency price prediction models are the main focus of this document, as opposed to the broader goal of creating the algorithmic trading software, which was much more simple.

1.2 Motivation

The main motivation behind creating such software lies in the fact that algorithmic trading bots are especially useful within the world of trading. The use of bots ensures that its developer's emotions are kept out of the decision-making process, with successful implementation of algorithmic trading software providing a great source of passive income for developers if they work correctly. Nonetheless, profit margins depend on the software's ability to predict the market accurately. Algorithmic trading is especially useful within the field of cryptocurrency as it is tradable 24/7, so software can be left to run indefinitely, as opposed to stock

markets which are closed for more time than they are open.

1.3 Structure

Relevant literature concerning previous price prediction models within cryptocurrency and other financial fields are explored and analysed within the literature review. High-level overview of the more advanced topics follow, with overviews of cryptocurrency, machine learning, and natural language processing. A description of the methodology used to create the machine learning models follows, before the penultimate chapter describes the process of creating the trading software. The last chapter provides a discussion of the project as a whole, offering insight into what could have been done differently and what developers should know before designing their own algorithmic trading software.

Chapter Two
Literature Review

Numerous machine learning techniques have been discussed in financial literature concerning how best to predict the prices, or price movements, of cryptocurrencies, as well as other financial assets such as individual stocks, stock market indices, commodities, and foreign currency exchanges. Moreover, such financial literature also deliberate ideas concerning *market efficiency* and whether price prediction may be an impossible (or, at the very least, unsustainable) task to begin with. This section aims to discuss these ideas, and provide some examples of previous price prediction studies that can used as a basis for the machine learning model development phase of this software design.

2.1 Financial Asset Price Prediction

As mentioned, machine learning based price prediction is not just limited to cryptocurrency. Examples of studies in which machine learning algorithms are used to predict the price of other financial assets include Chiroma et al. (2014) discussing how machine learning can be used to predict oil prices, Ranjit et al. (2018) concluding that *long-short term memory* (LSTM) can be used to predict foreign exchange prices, as well as the plethora of studies explored by Somanathan and Rama (2020) that explore stock market price prediction. Since cryptocurrency is still a relatively new asset, such studies provide valuable insight into how the prices of other asset classes can be predicted.

2.2 Efficient-Market Hypothesis

Financial literature concerning asset price prediction commonly refers to the *efficient-market hypothesis* (EMH), mostly associated with the work of Fama (1970). This hypothesis states that financial markets are *efficient*, i.e. an asset's current price is a reflection of all known information regarding the asset in question, and reacts to new information accordingly. What this means in practice is that investors and traders cannot use historic data and trading strategies to generate excess profit on a consistent basis, and that asset price prediction is a futile exercise in the long run.

Whilst EMH has been heralded as a milestone in modern financial theory (Urquhart 2016), it is important to note that Fama's work was written long before the invention of cryptocurrencies, thus it is debatable whether it can be applied to cryptocurrency at all.

Cryptocurrency efficiency was first explored by Urquhart (2016), who concluded that Bitcoin was highly inefficient under EMH; however, did comment that it was maturing. More recently, López-Martin et al. (2021) offer a mixed response, showing that degrees of efficiency of cryptocurrency markets vary over time, but are overall still maturing. This perhaps hints that cryptocurrency price prediction may be possible over inefficient periods, explaining the contrast in conclusions made within cryptocurrency price prediction literature.

EMH is not without its critics though. Weng et al. (2018) comment on how "an increasing number of studies [...] provide evidence contrary to what is suggested by the EMH", as well as on how some investors, such as Warren Buffet, are able to consistently beat stock indices, indicating that the EMH hypothesis is flawed.

2.3 Social Media & Sentiment Driven Pricing

The advance of social media has allowed investors to express sentiment and share information regarding various financial assets on a level never seen before, with social media playing an increasingly significant role in people's investment decisions (Umar et al. 2021). At its most extreme, online communities have used social media to consciously affect stock prices, such as in the 2021 GameStop short squeeze, when prices of GameStop shares rose from a low of $17.08 on 8th January 2021 to an intraday high of $483.00 on 28th January 2021 (Yahoo Finance 2022; Umar et. 2021; Hasso et al. 2021).

Such social media driven price fluctuations are well documented within the world of cryptocurrency (Tandon et al. 2021). However, the use of social media to influence cryptocurrency prices has made investors vulnerable to what are called *pump and dump* scams, in which social media hype is artificially created to drive a smaller cryptocurrency's price up, before the scammers sell, causing the cryptocurrency's value to plummet (Li et al. 2021; Xu and Livshits 2019).

Several studies have examined sentiment data extrapolated from various social media sites to explore their effect on asset prices, with most of the literature focusing on their relation to stock market or cryptocurrency prices (Nguyen et al. 2015; Mao et al. 2012; Jiao et al. 2020; Abraham et al. 2018; Kraaijeveld and De Smedt 2020; Jain et al. 2018; Colianni et al. 2015; Pano and Kashef 2021). Those studies relating to cryptocurrency are discussed in more detail in Section 2.4, forming the basis and inspiration for the methodology used in this paper.

2.4 Similar Studies

With the continued growth and use of cryptocurrency, cryptocurrency price prediction has become a trending research topic (Mezquita et al. 2021). The data gathered, models used, and prediction periods vary greatly; however, overlapping themes exist. As mentioned, this study will aim to use sentiment data from Twitter, as well as tweet volume data to make cryptocurrency price predictions; hence, this section aims to provide a general overview of the methodologies used and the conclusions reached of similar studies, as well as the limitations that exist within such studies.

2.4.1 Data Collection

Those studies that made use of data from Twitter generally collected tweets using the Twitter API or libraries that use the Twitter API such as the Python library Tweepy. This was true of Abraham et al. (2018), Kraaijeveld and De Smedt (2020), Colianni et al. (2015), and Jain et al. (2018). Of course, Twitter is not the only site that could have been used, with papers having been published using data from Weibo (Huang et al. 2021), Telegram (Smuts 2019), and Reddit (Wooley et al. 2019) exploring sentiment data and price movements.

As mentioned, tweet volume data are also referred to in this paper. Abraham et al. (2018) make use of an excellent resource, BitInfoCharts, to collect data concerning tweet volume numbers due to Twitter's API limits, discussed in more detail in Section 4.1.

The use of VADER (Valence Aware Dictionary for sEntiment Reasoning), a feature of the NLTK (Natural Language ToolKit) Python library, is prominent throughout the literature, used by Abraham et al. (2018), Kraaijeveld and De Smedt (2020), and Pano and Kashef (2021). VADER is a sentiment analysis tool which is designed for use within social media and microblogging contexts, making it an excellent tool to use in this context. Nevertheless, Kraaijeveld and De Smedt (2020) deliberate that sentiment analysis tools used within the literature fail to incorporate lexica used within cryptocurrency and financial circles. Huang et al. (2021) also comment on this, stating that sentiment models that do not include such lexica are "not applicable in the crypto domain". This is taken into account by Kraaijeveld and De Smedt (2020) and Smuts (2019), who all use customised sentiment scoring methods that take cryptocurrency slang and jargon into account. Huang et al. (2021) and Pant et al. (2018) take this one step further, using machine learning to build sentiment analysers for use within crypto domains.

2.4.2 Findings Summary

The results of studies aiming to predict cryptocurrency price movements using Twitter data and machine learning are generally mixed. Linear regression is used by Abraham et al. (2018) to conclude that Twitter sentiment data are an unreliable indicator of daily cryptocurrency price direction as "[p]eople who tweet about cryptocurrencies have an interest in them beyond investment opportunity", recommending the use of proxies to measure overall interest instead such as tweet volumes or Google Trends.

Similar conclusions were drawn by Jain et al. (2018), whose work using multiple linear regression models inferred that Bitcoin's price movements are not affected by Twitter sentiment. Their results did infer that this may not be the case for other cryptocurrencies, as more successful results were obtained when predicting the price of Litecoin, a different cryptocurrency not used in this study.

One earlier study from Colianni et al. (2015) claimed to be able to predict the price direction of Bitcoin with 59.0% accuracy, with this figure climbing to 76.23% accuracy when additional data preprocessing was undertaken.

Kraaijeveld and De Smedt's (2020) results are more varied, using Granger-causality tests to explore whether various cryptocurrency prices can be predicted, as opposed to creating price prediction models themselves. They discovered that it is possible to analyse Twitter

sentiment to predict the prices of Bitcoin, Bitcoin Cash[1] and Litecoin, but not for the other cryptocurrencies used in their analysis.

Pant et al. (2018) report more successful results, detailing that their recurrent neural network (RNN) model predicts the daily price of Bitcoin with 77.62% accuracy using sentiment data from Twitter as well as historic price data. However, little insight is provided on how this 77.62% figure was reached. Given that prices are continuous variables, it is unclear from the text whether this figure refers to price direction, some other variable, or a metric such as MAPE (see Section 3.2.4); hence, these data should be viewed cautiously.

2.4.3 Limitations

The limitations of these studies, as well as cryptocurrency price prediction literature in general, are discussed in detail by Kraaijeveld and De Smedt (2020). Firstly, they describe how literature on the topic is scarce: cryptocurrency's status as a new field means that literature is limited compared to that of the stock market.

Furthermore, the literature that does exist heavily focuses on Bitcoin, with literature on other cryptocurrencies hard to come by (Kraaiejveld and De Smedt 2020). Whilst this is unsurprising given Bitcoin's status within the cryptosphere as a whole, other cryptocurrencies are somewhat neglected.

As aforementioned in Section 2.4.1, Kraaijeveld and De Smedt (2020) also deliberate that sentiment analysis models within the literature fail to incorporate the lexicon used within cryptocurrency and financial circles.

Finally, there is a tendency within the literature to report optimised figures meaning there is a risk that certain models overfit (see Section 3.2.5) and are not, in fact, generalisable.

[1]Bitcoin Cash (BCH) is a separate cryptocurrency to Bitcoin, developed as a Bitcoin fork.

Chapter Three
Background

This section aims to provide a high-level analysis of relevant terms and concepts used throughout this paper, exploring their key features and history within an appropriate scope for those with computer science and/or mathematical backgrounds.

3.1 Cryptocurrency

Cryptocurrencies, the singular of which is often called crypto for short, can be described as digital assets that use a combination of cryptography, peer-to-peer networks and open-source distributed online append-only ledgers called *blockchains* to function as a secure, virtual currency (Narayanan et al. 2016). The first cryptocurrency, Bitcoin, was developed by the pseudonymous programmer Satoshi Nakamoto in the aftermath of the 2008 global financial crisis, in which public trust in conventional banking systems was damaged (Rejeb et al. 2021). Other cryptocurrencies have been launched since, based upon Nakamoto's model (albeit with numerous differences), with cryptocurrencies that have been created since Bitcoin referred to as *altcoins* (Narayanan et al. 2016). The total market cap of cryptocurrencies at the end of 2021 was around $2.2 trillion (CoinMarketCap 2022).

Cryptocurrency transactions can take place between two parties without the involvement of third parties such as banks and credit card companies, using blockchain technology and its underlying cryptographic key system instead (Narayanan et al. 2016). In most cases, these blockchains are decentralised and exist outside the authority of central banks and governments, meaning such entities have no control on their supply (Inci and Lagasse 2019). Regulation is possible to an extent, with various governments enacting cryptocurrency legislation, such as banning/authorising their possession and/or use (Hendrickson and Luther 2017; Geiregat 2018). However, the effectiveness of such anti-cryptocurrency legislation is debated (Chokor and Alfieri 2021; Kiviat 2015).

Wallets that contain a user's cryptographic keys enable users to use their crypto. Cryptocurrency is technically not stored in these wallets: they are stored on the blockchain but can be accessed through these keys (Narayanan et al. 2016; Nakamoto 2008). Each user has a *public key* which allows them to receive transactions, and a *private key* (also called a *secret key*) that allows a user to digitally sign transactions which are then verified on the blockchain using a *blockchain consensus protocol* (Narayanan et al. 2016). Bitcoin uses a consensus protocol called *proof of work* (PoW) (Nakamoto 2008) described in more detail in Section 3.1.1; however, other consensus protocols have been developed (Zhang and Lee

2020). Cryptocurrency transactions have a transaction fee associated with them as well, with the level of fees and protocols associated with them varying from cryptocurrency to cryptocurrency (Chepurnoy et al. 2018; Judmayer et al. 2018).

The advance of cryptocurrency has be heralded by some as the start of a new financial era outside the traditional banking system, cutting out middlemen and providing anonymity and decentralisation to the masses (Houben and Snyers 2018). It has been argued that cryptocurrency adoption can offer financial solutions to the un(der)banked, as well as a way of sending money internationally without high transaction fees, with El Salvador becoming the first jurisdiction to adopt Bitcoin as legal tender for these reasons (Gorjón 2021).

The unregulated and anonymous nature of cryptocurrency, however, makes it controversial. Darknet markets – online black markets that operate on the dark web – use cryptocurrency as a medium of exchange for this reason (Desmond et al. 2019). Stoll et al. (2019) ratiocinate, in fact, that individuals wanting to purchase illicit items such as drugs, weapons, or child pornography need to be protected from themselves and others, thereby justifying centralised financial controls. Moreover, cryptocurrency is also used to facilitate money laundering, tax evasion, and funding terrorism (Houben and Snyers 2018).

Additionally, there is criticism of cryptocurrency's enormous energy output and subsequent carbon footprint, especially of those cryptocurrencies who make use of the PoW consensus protocol (Stoll et al. 2019; Howson and de Vries 2022). Carbon emissions from Bitcoin usage alone, should Bitcoin's rate of adoption and dependence on carbon-emitting resources continue, could push global warming above the 2°C threshold agreed upon at the Paris Climate Accords within thirty years (Mora et al. 2018).

Moreover, the vast amount of energy used by cryptocurrency mining (see Section 3.1.1 for a full description mining) means that in some countries, locals compete against cryptocurrency miners for electricity (Howson and de Vries 2022). It should be noted that these concerns relating to electricity use are criticisms of PoW rather than of cryptocurrency in general, as highlighted by Howson and de Vries (2022).

Whilst cryptocurrency was not initially intended to be used as an investment tool, many people use it as such (Inci and Lagasse 2019). Cryptocurrency prices are infamously volatile, making them incredibly high-risk investment assets; however, the returns of such investments can be much larger than those of standard investments, with the global cryptocurrency market cap increasing by 196% in 2021, compared to a 27% increase in the S&P 500 (CoinMarketCap 2022; Yahoo Finance 2022).

The five cryptocurrencies used in this study are Bitcoin (BTC), Ethereum[2] (ETH), Cardano[3] (ADA), Dogecoin (DOGE), and Tether (USDT), described in detail in the following subsections.

[2]Ethereum is technically the name of the blockchain rather than the native cryptocurrency that runs on it. Ether is the name of the currency; however, most of the literature, as well as all the online resources consulted in this study refer to the currency as Ethereum, except when discussing other features of Ethereum's blockchain. This name is therefore used throughout this paper, except in Section 2.1.3 in which other features of the Ethereum blockchain are discussed.

[3]Similar to the difference between Ethereum and Ether, Cardano is the name of the blockchain and Ada is the name of the internal cryptocurrency that runs on it. Again, most the literature and all the online resources consulted in this study refer to the currency as Cardano unless discussing other features of Cardano's blockchain so that name is used here to refer to the cryptocurrency.

3.1.1 Bitcoin

Bitcoin's white paper was published by the pseudonymous Satoshi Nakamoto in 2008 and launched in 2009. As mentioned, it was the first cryptocurrency to be developed and is the root of all blockchain technology that exists today. New bitcoins enter the market through a process called *mining*, given as a reward to miners for being the first to verify a block of transactions through Bitcoin's PoW consensus protocol (Nakamoto 2008). The reward for mining a block is 6.25 bitcoins at time of writing, with this reward halving every 210,000 blocks mined (approximately every four years). As well as this block reward, miners also receive the transaction fees for each block's transactions.

Each block in Bitcoin's blockchain contains a *nonce* (number used once) that, when *hashed* alongside a list of transactions using the SHA-256 *cryptographic hash function* (i.e. inserted into an encryption algorithm called SHA-256), produces a *hash* (output) that begins with a certain number of zeroes (Nakamoto 2008). The goal of miners is to find the nonce of the most recent block that results in such a hash. As this nonce must be guessed, Bitcoin mining requires a lot of computational work, hence why the PoW protocol is named as such. The number of zeroes required at the beginning of a hash depends on a moving average that targets an average of six blocks being mined per hour, with the difficulty of finding the correct hash (i.e. the number of zeroes at the start of a hash needed) increasing when blocks are generated too quickly and decreasing when blocks are generated too slowly. An example of this cryptographic process, where nineteen zeros were required at the start of a hash to mine the 715,160$^{\text{th}}$ block of the Bitcoin blockchain, can be seen in Figure 3.1 below.

Whilst it is not possible to fake transactions due to the lack of digital signatures, it is theoretically possible for malicious miners to prevent new transactions being added to the chain, reverse/remove transactions, or double-spend coins should they be the first to find a block's valid hash. However, these transactions will only be accepted if the network falls victim to what is referred to as a 51% attack: a type of attack in which more than 50% of a network's computing power is controlled by the attackers (Ye et al. 2018; Nakomoto 2008).

Bitcoin's maximum supply is 21 million bitcoins (Nakamoto 2008), with its block reward continuing to decline each halving. At time of writing, nearly 19 million bitcoins have been mined and are in circulating supply (Blockchain.com 2022). These coins can be divided into units of hundred-millionths, or 10^{-8} of a bitcoin, with these units colloquially referred to as satoshis (Judmayer et al. 2017).

Bitcoin is, to date, the only cryptocurrency that has been made legal tender, with El Salvador adopting it as such in 2021 (Gorjón 2021).

3.1.2 Ethereum

Ethereum was theorised by developer Vitalik Buterin in 2013 and launched in 2015. It is the second most valuable cryptocurrency after Bitcoin, with a market cap of around $400 billion at time of writing (CoinMarketCap 2022).

Unlike Bitcoin, Ethereum uses its blockchain for other purposes outside of cryptocurrency. The Ethereum blockchain takes the decentralised payment system created by Bitcoin and takes it one step further by allowing users to build/use *decentralised applications* (DApps) that run on the Ethereum network using its own global decentralised virtual

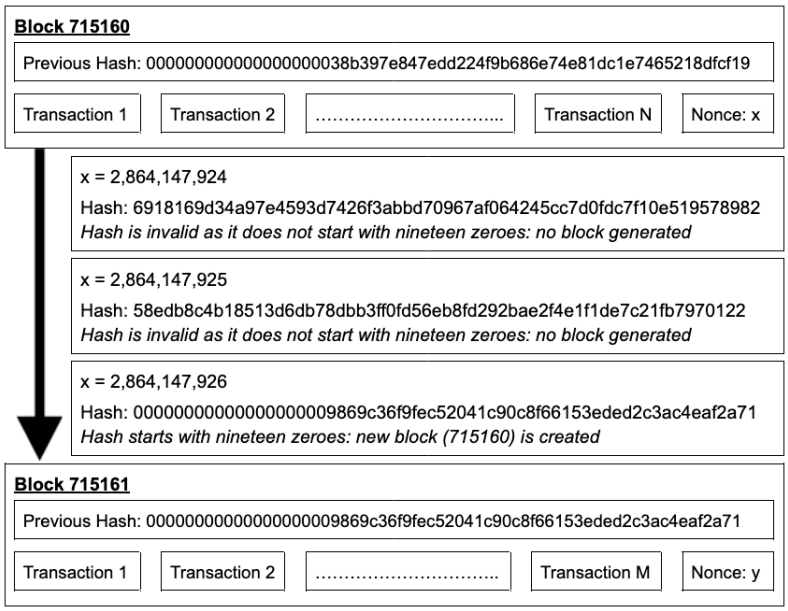


Figure 3.1: A representation of how the 715,160$^{\text{th}}$ Bitcoin block was mined. While a Bitcoin miner is unlikely to guess nonces in sequential order, this is done so in the figure to demonstrate how SHA-256 provides no hint that you are close to the right nonce. Adapted from Nakamoto (2008). Hashes and nonces acquired from the Bitcoin blockchain (Blockhain.com 2021).

machine, called the *Ethereum Virtual Machine* (EVM) (Antonopoulos and Wood 2018; de Azevedo Sousa 2021). DApps that operate on the Ethereum blockchain use its native cryptocurrency, Ether, for financial transactions; however, other cryptocurrencies that run on Ethereum's blockchain may be used (Ethereum 2022). Similarly, Ethereum is also used to host *non-fungible tokens* (NFTs): unique and immutable representations of blockchain-based assets such as collectibles, digital art, and video game memorabilia (Antonopoulos and Wood 2018). Sales of NFTs are made in Ether when hosted on the Ethereum blockchain.

As Ethereum is not just used for cryptocurrency transactions, there are also fees associated with using the network's computation in addition to Ether transaction fees, with these collectively being referred to as gas fees on the Ethereum network (de Azevedo Sousa 2021).

Ethereum currently uses PoW as its consensus protocol, with new blocks being mined on average every 14 seconds (Kiffer et al. 2017; Ethereum 2022). Miners are incentivised to add new blocks with a block reward in much the same way as Bitcoin, i.e., a successful miner will receive a set block reward, in addition to some of the block's gas fees (Antonopoulos and Wood 2018). There is no set supply cap for Ether; however, Ethereum's most recent upgrade implemented a burning mechanism, in which some Ether used for transaction fees are taken out of circulation, to combat inflation (Ethereum 2022).

Ethereum is in the process of changing consensus algorithms due to the large carbon footprint associated with PoW and moving to *proof of stake* (PoS) (Park et al. 2020). PoS

is described in more detail in Section 3.1.3 as Cardano uses the same protocol.

3.1.3 Cardano

Cardano was developed by one of the co-founders of Ethereum, Charles Hoskinson, with Cardano being founded in 2015 and initially released in 2017 (Hoskinson 2017).

Like Ethereum, Cardano uses its blockchain for other purposes outside of cryptocurrency, with NFTs and DApps both being implemented within the system (Cardano 2021).

The Cardano Foundation consider Cardano to be a third-generation cryptocurrency that solves scalability, interoperability and sustainability issues present in earlier generations (Cardano 2021). Akram et al. (2020) refer to such currencies as *Blockchain 3.0*, the successors to *Blockchain 1.0* (first-generation cryptocurrencies such as Bitcoin that run basic PoW decentralised ledgers for handling cryptocurrency transfers) and *Blockchain 2.0* (second-generation cryptocurrencies such as Ethereum that extend the uses of blockchain to DApps). Cardano is able to handle more transactions per second (TPS) than Bitcoin and Ethereum without affecting network performance, with Cardano able to handle 257 TPS, compared to the 7 TPS and 15 TPS of Bitcoin and Ethereum respectively (Akram et al. 2020).

Cardano uses a PoS consensus algorithm as opposed to PoW, specifically the Ouroboros protocol described in Kiayas et al. (2017). In PoS, transactions are validated by miners (often called *validators* within PoS frameworks) who are randomly selected to validate blocks, with the chances of being selected proportional to a user's stake: the amount of coins used by a miner as collateral, which can be taken away in the event of the validator approving malicious activity. As PoS does not rely on large amounts of computational power, it is much more energy efficient than PoW, with validators being able to use standard CPUs as opposed to specialised mining hardware that runs on expensive GPUs (Kiayas et al. 2017).

3.1.4 Dogecoin

Dogecoin was the first example of a meme coin: a cryptocurrency created for satirical purposes that originate from Internet memes (Tandon et al. 2021). It was created by Billy Markus and Jackson Palmer in 2013 as a way of making fun of the speculative nature of cryptocurrency investment, although it has since attracted legitimate investors of its own (Chohan 2021).

Before 2021, Dogecoin was mostly used as a tipping mechanism within the Dogecoin community, given the coin's low value; however, 2021 saw a surge of interest in Dogecoin with large price rallies and celebrity endorsements (Chohan 2021). Dogecoin's price rose from \$0.0012 at the start of 2021 to an all-time high of \$0.71 in August 2021, with the price crashing down to around \$0.17 by the end of the year (CoinMarketCap 2022).

Dogecoin proponents, such as Tesla CEO Elon Musk and billionaire entrepreneur Mark Cuban, have argued in the mainstream media that Dogecoin is more suited to being a medium of exchange than Bitcoin and Ethereum due to its low transaction fees and set inflation rate (Locke 2021a, 2021b). Dogecoin Core, the official Dogecoin wallet, recently lowered transaction fees to 0.01 DOGE in its most recent upgrade (Dogecoin Core 2021), and while Dogecoin has no supply cap, it does have a deterministic inflation rate of 10,000

dogecoins per block mined (Chohan 2021). A new Dogecoin block is mined every minute, with five billion dogecoins being added each year from the mining process (Chohan 2021).

While Dogecoin uses PoW as its consensus algorithm, it differs from Bitcoin in that it uses Scrypt as its hash function as opposed to SHA-256, a hash function designed to be more memory-intensive than SHA-256 (Chohan 2021; Dogecoin Core 2021).

3.1.5 Tether

Tether is a type of cryptocurrency called a *stablecoin*, those coins whose values are pegged to traditional assets, such as fiat currencies or commodities, and aim to solve the problem of traditional cryptocurrencies being unsuitable as a medium of exchange due to their volatility (Chohan 2020; Bullman et al. 2019). Stablecoins offer a way of using fiat currencies on blockchain-based services and are commonly used within *decentralised finance* (DeFi), *peer-to-peer* (P2P) based financial services that have no central intermediaries that run on DApps (Bullman et al. 2019). Cryptocurrency exchanges make use of stablecoins as a means of easily converting between fiat and crypto, with Tether having a higher trading volume than any other cryptocurrency at time of writing (CoinMarketCap 2022).

3.2 Machine Learning

Machine learning refers to a discipline within computer science in which self-improving algorithms can be used to make predictions or perform tasks, based on input data called *training data*. Machine learning works on the assumption that previous data (i.e. training data) can be used to predict future events. This assumption is argued to use the same logic by which humans make predictions, and is therefore considered to be a subset of *artificial intelligence* (AI) (Knox 2018).

3.2.1 Supervised, Unsupervised, & Reinforcement Learning

There are a profusion of different machine learning algorithms, which can generally be divided into those that use *supervised learning*, *unsupervised learning*, and *reinforcement learning*.

Supervised machine learning algorithms refer to those that make use of labelled data, with solutions being provided during the training phase (Knox 2018). An example of a supervised machine learning algorithm may be one that predicts weather events based on parameterised data such as atmospheric pressure, wind direction and relative humidity. Supervised learning problems can further be broken down into those that predict discrete variables (i.e. binary or categorical variables), referred to as classification models, and those that predict continuous variables, referred to as regression models (Knox 2018).

Unsupervised machine learning algorithms are those that use unlabelled data. They are common within the field of cluster analysis (Rodriguez 2019), in which data points are analysed and segregated on the basis of how close their positions are.

Reinforcement learning is considered to be a third paradigm outside of supervised and unsupervised learning, in which algorithms explore an environment aiming to maximise a cumulative *reward* using trial and error and learning from mistakes (Sutton and Barto

2018). Reinforcement learning has been deployed successfully in areas such as game theory (Nowé et al. 2012) and autonomous vehicle design (Pan et al. 2017).

While these three approaches are considered to be the three core paradigms of machine learning, hybrid approaches also exist that make use of a combination of these (Knox 2018).

3.2.2 Optimisation and Cost Functions

Machine learning algorithms self-improve using *function optimisation*, a mathematical concept in which a real function[4] is minimised and/or maximised by using specific inputs. This function, in the context of machine learning, is called a *cost function* or *loss function*, and is a way of measuring a machine learning algorithm's performance. Cost functions represent cumulative error, therefore it is the goal of a machine learning algorithm to minimise its cost function through function optimisation. Examples of cost functions include *mean squared error* (MSE) and *mean absolute error* (MAE), both of which are described in more detail in Section 3.2.4 through their use as evaluation metrics[5].

3.2.3 Test Data, Training Data, and Cross-Fold Validation

As mentioned, the data that are used to develop machine learning models are called *training data*, with data used to evaluate a model's performance being called *test data*. When these are gathered from the same data set, most common in supervised learning environments, testing and training data can be collected using a *train-test split*, in which a data set is divided into two test data and training data subsets. There is no consensus on how best to split these data, although 80%-20% and 70%-30% are common (Knox 2018).

An alternative to only splitting the data once is to use a technique called k-fold cross-validation, in which multiple models are created using a data set that resamples training and test data, before aggregating the models' performance results. The k in k-fold stands for the number of folds used, and is replaced with a number when specified, e.g. 10-fold cross-validation. Similar to the train-test split ratio, there is no general consensus on how many folds should be used, although 10-fold cross-validation is common (Berrar 2019). While a higher number of folds provides a larger training set, it does require more computational power (Berrar 2019).

The first stage of k-fold cross-validation is to shuffle a data set, then split the data set into k subsections. For each subsection, a model is then created using the data outside the subsection as training data, and the data within it as test data. A diagram of this can be seen in Figure 3.2, in which 10-fold cross-validation is used.

[4]Defined as a function f(x) in which $D \subseteq \mathbb{R}$, where D is the function's domain,

[5]Evaluation metrics, discussed in detail in Section 3.2.4, are similar to cost functions, though differ in their purpose. Cost functions are used during the training (i.e. for function optimisation) process to build a machine learning model. Evaluation metrics are used to analyse a complete model's performance.

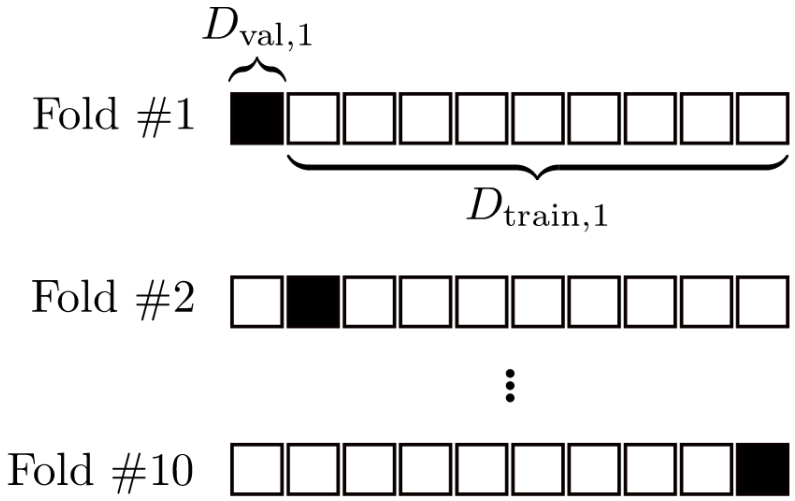


Figure 3.2: A representation of 10-fold cross-validation from Barrer (2019), in which a data set D is split into ten disjoint subsets that are then iteratively used as test data. D_{train} represents training data while D_{val} represents test data.

At its most extreme, k can equal the respective data set's length. This is called *leave-one-out cross-validation* (LOOCV), a diagram of which can be seen in Figure 3.3.

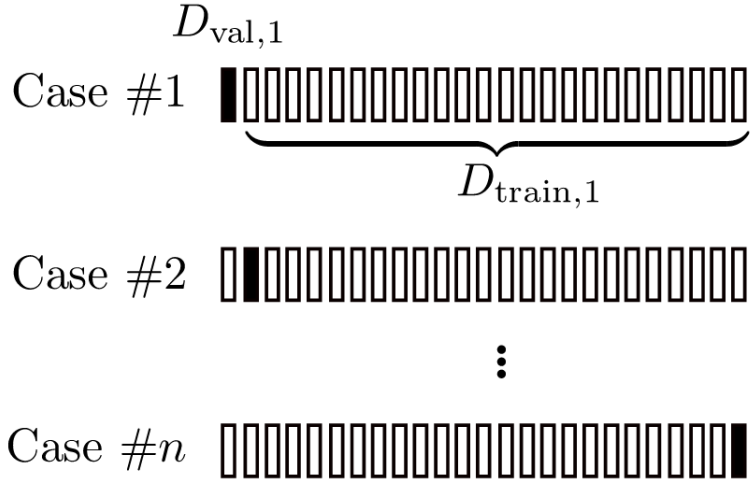


Figure 3.3: A representation of leave-one-out cross-validation from Barrer (2019), in which the data set is of length $n = 25$ cases. Each case is used as a hold-out test while a model is built using the remaining cases.

3.2.4 Confusion Matrices and Evaluation Metrics

When classification machine learning models are used to predict binary outcomes, it is possible to express their performance using a *confusion matrix*, in which correctly classified results are referred to as *true positives* (TP) and *true negatives* (TN). Incorrectly classified results are referred to as either *false positives* (FP), in which a model predicts a positive that in reality is negative, and *false negatives* (FN), in which the reverse is true. The structure of such tables are demonstrated in Table 3.1.

	Predicted positive	*Predcited negative*
Actually positive	TP	FN
Actually negative	FP	TN

Table 3.1: A reprsentation of a confusion matrix's structure.

The performance of binary classification machine learning models can be evaluated using various metrics. One of the most common ways to evaluate their performance is using their accuracy, the formula for which is displayed in Eq. (3.1).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

Using accuracy as a performance can be criticised as a metric, as it can be misleading when working with data sets with imbalanced classes. Consider, for example, a model that predicts the presence of COVID-19 in a patient. The confusion matrix in Table 3.2 shows an extreme example in which the model always predicts a patient is negative. The accuracy of this model is 96.5%, despite missing 100% of positive COVID-19 cases.

	Predicted positive	*Predcited negative*
Actually positive	TP (0)	FN (51)
Actually negative	FP (0)	TN (1424)

Table 3.2: Confusion matrix of a hypothetical machine learning algorithm that always predicts a negative COVID-19 diagnosis.

More appropriate measures to measure the performance of such machine models include its $F_1 score$, or *Matthews correlation coefficient* (MCC), the formulae for which are displayed in Eqs. (3.2) and (3.3) respectively. These metrics take imbalanced classes into account, making them more appropriate for evaluating binary classification machine learning models.

$$F_1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{3.2}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{3.3}$$

Wider confusion matrices and adjusted evaluation metric formulae can be used to measure the performance of classification models that predict non-binary categorical variables, e.g. models that predict martial status or eye colour. Such models are outside this study's relevance and are therefore not discussed here.

Regression models require different evaluation metrics given that they predict continuous variables. When discussing these metrics, it is first necessary to discuss the concept of *error*, that is, the difference between a model's predicted output and the output's true value. The *absolute error* refers to this difference's magnitude, while the *relative error* expresses this difference as a ratio or percentage of the true value. The formulae for absolute error and relative error are expressed in Eqs. (3.4) and (3.5) respectively, in which x represents a true value, and y represents its corresponding prediction.

$$Absolute\ Error = |x - y| \tag{3.4}$$

$$Relative\ Error = |\frac{x - y}{x}| \cdot 100\% \tag{3.5}$$

Mean absolute error (MAE) and *mean absolute percentage error* (MAPE) are two evaluation metrics for regression models that, given a set of errors, work out the mean value for the absolute error and relative error respectively. Generally, the smaller this figure, the better the model is perceived to be. The formulae for these metrics are shown in Eqs. (3.6) and (3.7), in which:

- $x_i \in X$, where X is a set of a regression model's true values
- $y_i \in Y$, where Y is a set of the corresponding predicted values
- n is the cardinality of sets X and Y, where $|X| = |Y|$ is assumed

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - y_i| \tag{3.6}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}|\frac{x_i - y_i}{x_i}| \tag{3.7}$$

MAPE, due to its use of relative error, has the advantage of representing data sets with large scales of data more fairly. That being said, it is not possible to use data sets where zero is an actual value with MAPE, as it requires division by zero. Similarly, MAPE can exhibit bias as it applies heavier penalties on *negative errors*, those where a value is overestimated, than positive errors, especially when actual values fall close to zero (Kim and Kim 2016). These problems are addressed by Kim and Kim (2016), who present *mean arctangent absolute percentage error* (MAAPE) as an alternative to MAPE, in which the relative error formula is presented as a trigonometric function, justified in Figure 3.4 and shown in Eq. (3.8).

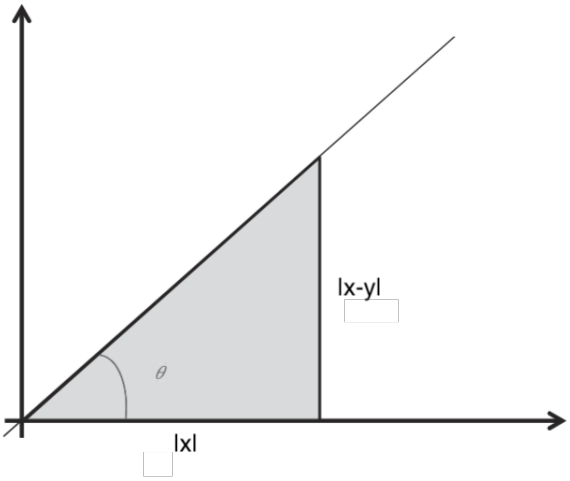


Figure 3.4: Trigonometric conceptual justification of arctangent absolute percentage error (AAPE) from Kim and Kim (2016), where x represents a true value and y represents its corresponding prediction.

$$\theta = tan^{-1}(|\frac{x-y}{x}|) \tag{3.8}$$

The formula for MAAPE is derived from this function, and presented in Eq. (3.9), in which x_i, y_i, and n represent the same variables as in Eqs. (3.6) and (3.7).

$$MAAPE = \frac{1}{n}\sum_{i=1}^{n} tan^{-1}(|\frac{x_i - y_i}{x_i}|) \tag{3.9}$$

The metric values of MAE, MAPE and MAAPE all significantly increase when a data point with a large[6] amount of error is added, though this increase is applied using a linear scale. Other evaluation metrics, such as *mean squared error* (MSE) have been devised that use formulae to magnify these larger errors using exponential scales. In MSE's case, this is done through squaring, as can be seen in its formula in Eq. (3.10). Again, x_i, y_i, and n represent the same variables as in Eqs. (3.6), (3.7) and (3.9). Note that while the use of absolute error is inferred, squaring the difference between the predicted and true values means that it is not necessary to take an absolute value.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2 \tag{3.10}$$

Similar to MSE is *root means squared error* (RMSE), an expansion of MSE in which the square root of the MSE is value is taken, shown in Eq. (3.11).

$$RMSE = \sqrt{MSE} \tag{3.11}$$

[6]Large in the sense defined by whether relative or absolute error are used in each metric's formula.

MSE coefficients are expressed as square units, e.g. the MSE of a regression model that predicts a value in euros may have an MSE of 10,000 euros squared. RMSE therefore allows the use of a more logical unit in this case. MSE and RMSE are currently two of the most popular evaluation metrics used within regression learning (Kim and Kim 2016; Chai and Draxler 2014), though its inflated punishment mechanism can be criticised as being unnecessarily punitive. Some researchers advocate MAE as an alternative for this reason (Chai and Draxler 2014); however, MAE is not as scalable as MAPE and MAAPE (Kim and Kim 2016).

3.2.5 Generalisation, Underfitting, and Overfitting

Machine learning models are said to have good *generalisation* if they can accurately classify unseen data points (Knox 2018). However, focusing too heavily on optimisation may result in a machine learning model *overfitting*, in which a model fits too closely to the training data due to outliers and/or noise data being captured. The opposite of this is *underfitting*, in which a machine learning model is too simple to capture a more complex pattern. Figure 3.5 shows an example of classification models using the same data set that are generalised, overfitting, and underfitting respectively.

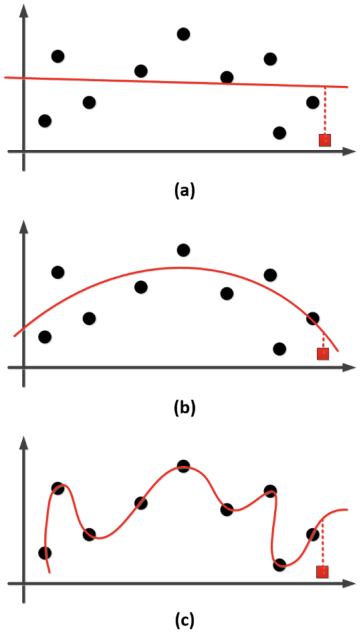


Figure 3.5: An example for (a) overfitting, (b) generalised, and (c) overfitting data, in which black circles represent training data, the red square represents a test sample, and the red line represents a regression model's fitted line. Taken from Ghojogh and Crowley (2019)

3.2.6 Feature Selection

Before machine learning models are created, they sometimes have to go through a process called feature selection in which some variables are removed from an input data set. This is done so that models do not overfit due to containing too many variables (Hawkins 2004), as well as remove variables that have negligible effects on the output. Added benefits from this process include less computation time as well as producing simpler models that can be more easily interpreted by humans (Hawkins 2004).

Ways of selecting features within supervised machine learning models include the use of correlation statistics, an example of which is the *Pearson Correlation Coefficient* (PCC). This coefficient measures the linear correlation of a population between two data sets, using the formula in Eq. (3.12), in which:

- X and Y refer to a pair of variables where $|X| = |Y|$
- $cov(X, Y)$ refers to the covariance of X and Y
- σ_X and σ_Y refer to the standard deviations of X and Y respectively

$$PCC = \frac{cov(X, Y)}{\sigma_X \sigma_Y} \tag{3.12}$$

The range of Eq. (3.12) is [-1, 1], where 1 represents direct positive correlation, -1 represents direct negative correlation, and 0 represents no (linear) correlation, as shown in Figure 3.6. Figure 3.6 also demonstrates that PCC only measures linear correlation, it may miss correlations that are not linear.

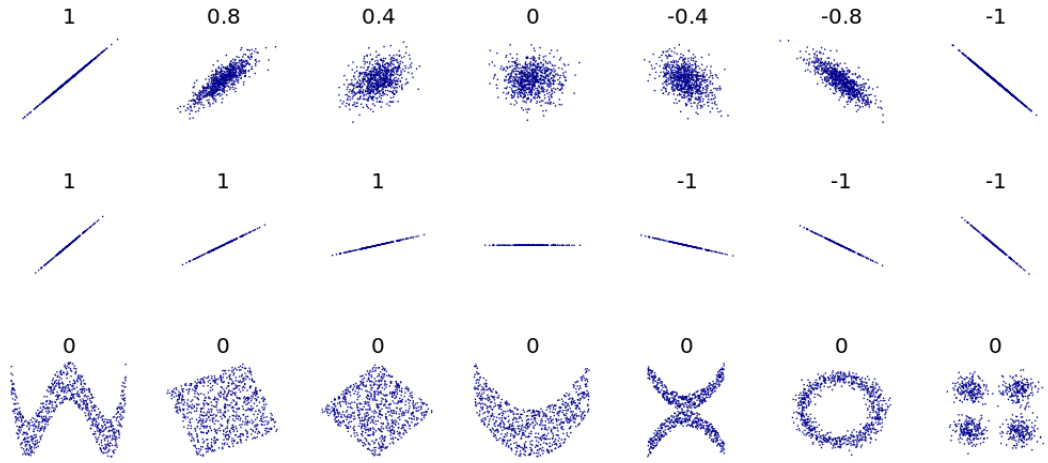


Figure 3.6: A representation of the PCC for a number of different data sets. Taken from Boigelot (2011).

After correlation statistics are collected, features can therefore be selected based on a predetermined criterion, i.e. only selecting those variables over a certain correlation coefficient threshold, the most correlated k variables, or those variables within a certain top percentile.

3.2.7 WEKA

WEKA is a Java-based piece of software developed by the University of Waikato (Frank et al. 2016) that allows users to easily run machine learning tasks either using the inbuilt GUI, Java code, or command line prompts. At time of writing, WEKA 3.8.5 is the most recent stable release, and WEKA 3.9.5 is the most recent developer release. WEKA 3.9.5 was the version used for this research.

While additional functionality is available through downloading extension packages since WEKA 3.7.2 (Frank et al. 2016), the core functionality of WEKA allows users to do the following:

- Upload CSV, ARFF, JSON and similar file types and preprocess data through removing attributes (decisions of which can be aided through in-built feature selection tools) and/or converting between data types.
- Create models using various supervised machine learning algorithms such as Naïve-Bayes, J48 Decision Trees, and K-Nearest Neighbours, as well as adjust each model's hyperparameters: those parameters that control the learning process as opposed to those derived through the learning itself. Over 50 different supervised algorithms are available to choose from, and therefore cannot be sufficiently discussed in this paper. Note that neural network models such as LSTM are not available.
- Analyse the performances of machine learning models created in WEKA, using a variety of evaluation metrics.
- Perform clustering algorithms on data sets and visualise the results.

Note that this is merely a brief overview of the core functionality that mainly focuses on the areas relevant to this research. Other core functionality may be more relevant to users depending on their research objectives. A screenshot of the WEKA GUI is available to view in Figure 3.7.

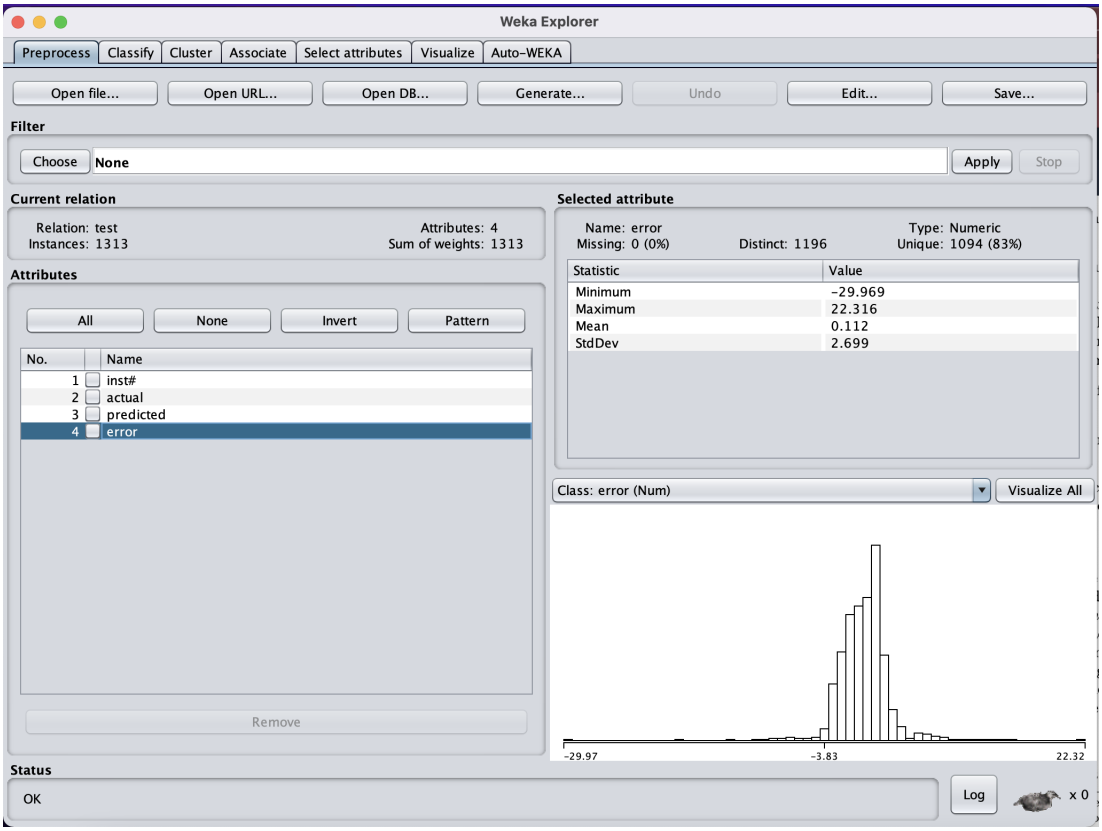


Figure 3.7: A screenshot of WEKA's GUI interface.

3.2.8 Auto-WEKA

One of the packages that can be downloaded for WEKA is called Auto-WEKA, developed by Thornton et al. (2013), to address what they refer to as the *combined algorithm selection and hyperparameter optimization* (CASH) problem.

Summed up briefly, for each machine learning algorithm available to use on WEKA, there exists a set of possible tunings for the algorithm's hyperparameters. As finding the most optimised machine learning algorithm for a problem is, in reality, finding the most optimised algorithm-hyperparameter combination, a large amount of computational effort is required as the set of all possible combinations is of considerable size.

More formally, Thornton et al. (2013) define the CASH problem as computing Eq. (3.13), given the following[7]:

- A set of algorithms $A = \{a^{(1)}, a^{(2)}, ..., a^{(k)}\}$ in which a^* is the optimised algorithm
- An associated set of hyperparameter spaces $\{\Lambda^{(1)}, \Lambda^{(2)}, ..., \Lambda^{(k)}\}$ exists for A in which each $\Lambda = \{\lambda^{(1)}, \lambda^{(2)}, ..., \lambda^{(n)}\}$, where λ^* is the optimised hyperparameter space

[7]Note that some variables used here and within Eq. (3.13) are given different symbols than those used by Thornton et al. (2013) in this paper to ease formatting issues.

- a_λ refers to an algorithm a with hyperparameter settings λ where $a \in A$ and $\lambda \in \Lambda$
- δ refers to training data $\delta = \{(x_1, y_1), ..., (x_n, y_n)\}$, split into disjoint training and validation sets $d^{(i)}_{train}$ and $d^{(i)}_{valid}$
- $\zeta(a, \delta^{(i)}_{train}, \delta^{(i)}_{valid})$ refers to an unspecified loss achieved by a when trained on $\delta^{(i)}_{train}$ and evaluated on $\delta^{(i)}_{valid}$

$$a^*_{\lambda^*} \in \underset{a^{(j)} \in A, \lambda \in \Lambda^{(j)}}{argmin} \frac{1}{k} \sum_{i=1}^{k} \zeta(a^{(j)}_\lambda, \delta^{(i)}_{train}, \delta^{(i)}_{valid}) \tag{3.13}$$

Auto-WEKA calculates Eq. (3.12) for data sets uploaded to WEKA, using 10-fold cross-validation in the process. The loss function can be specified in Auto-WEKA's settings, and exists as an additional hyperparameter with which users can tinker.

Due to the exceedingly large amount of computation required to calculate Eq. (3.12), Auto-WEKA must be left running for several hours to get truly reliable results. Nonetheless, as Auto-WEKA approaches $a^*_{\lambda^*}$ at a logarithmic rate, allowing Auto-WEKA to run for shorter periods of time can suffice in most circumstances.

Models created using Auto-WEKA (as well as vanilla WEKA) are saved as *.model* files that can then be run using WEKA's GUI, Java code, or the command prompt through use of Java's Virtual Machine (JVM).

3.3 Natural Language Processing

Natural language processing (NLP) is the subfield in which linguistics and computer science are used to provide and analyse computational mechanisms for processing, examining, and producing natural language data (Indurkhya and Damerau 2010). There are a wide variety of NLP use cases, examples of which include speech recognition and machine translation, as well as the concepts discussed in this section.

3.3.1 Sentiment Analysis

Also called *opinion mining*, sentiment analysis is an area of NLP in which subjective and/or emotive information is computationally extracted from pieces of text (Liu 2020). A popular task within the field is determining the *polarity* of a text: that is, whether it is positive, negative, or neutral in nature. More complex models can be said to explore sentiment *beyond polarity* when they explore more advanced emotional states, such as happiness, anger, or fear.

The rise of social media has led to a large amount of sentiment analysis research concerning such platforms, due to its abundance of publicly available, opinion-heavy data (Liu 2020). Much of this research focuses on Twitter, although research concerning other social sites such as Reddit and Weibo exist (Liu 2020; Lubitz 2017; Huang et al. 2021).

The most important indicators of a text's sentiment are what Liu (2020) refers to as a text's *sentiment words*. The concept of a sentiment word is best explained through the examples presented by Liu (2020), in which "*good*", "*wonderful*", and "*amazing*" are examples

of positive sentiment words, and "*bad*", "*poor*" and "*terrible*" are examples of negative sentiment words. Nonetheless, relying on sentiment words alone does not account for linguistic features that make sentiment analysis a complex endeavour. Liu (2020) highlights some of the difficulties in relying on sentiment words alone:

- Negation: a word can be used with a negative to express its opposite meaning, e.g. "*not good*".
- Interrogatives and conditionals: sentiment words may not express sentiment at all when used in structures such as "*Is this a good camera?*" or "*If I can find a good camera, I will buy it*".
- Polysemy and homonymy: words can have different meanings in different contexts, compare "*This camera sucks*" and "*This vacuum cleaner really sucks*".
- Sarcasm: sentences such as "*What a great car! It stopped working after two days!*" that use sarcasm are complex for a sentiment analysis model to understand.

This dissertation makes use of a sentiment analysis tool called VADER, a Python library used for carrying out NLP tasks that is commonly used in the literature (see Section 2.4.1). Its common usage in the literature, as well as its tailored use within social media and microblogging contexts such as on Twitter, make it an appropriate tool to use.

3.3.2 Word Embeddings

A word's meaning can be represented computationally as a vector through a technique called *word embedding*. While representations vary depending on the model used to create them, they are generally dimensionally heavy. For example, SpaCy, a Python library used for NLP used in this research, has an in-built word embedding feature that uses 300-dimension word vectors.

Representing words as vectors allows you to compare word similarity, in which a figure between 0 and 1 demonstrates how close two words are semantically by comparing their word embeddings. This is demonstrated in Table 3.3, in which the similarities of the word vectors, created using SpaCy, for "*dog*", "*cat*", "*lion*", "*tiger*", "*wolf*", "*cow*", "*pig*", and "*sheep*" are compared and contrasted.

	dog	cat	lion	tiger	wolf	cow	pig	sheep
dog	1.000	0.801	0.474	0.436	0.520	0.519	0.569	0.464
cat	0.801	1.000	0.526	0.541	0.508	0.474	0.549	0.383
lion	0.474	0.526	1.000	0.735	0.647	0.478	0.490	0.479
tiger	0.436	0.541	0.735	1.000	0.592	0.400	0.434	0.315
wolf	0.520	0.508	0.647	0.592	1.000	0.451	0.459	0.498
cow	0.519	0.474	0.478	0.400	0.451	1.000	0.738	0.753
pig	0.569	0.549	0.490	0.434	0.459	0.738	1.000	0.640
sheep	0.464	0.383	0.479	0.315	0.498	0.753	0.640	1.000

Table 3.3: The word similarity of various animals according to SpaCy

Interesting patterns and clusters can be discovered from comparing the similarity scores in Table 3.3. For each farmyard animal, vectors representing other farmyard animals are more similar (i.e. closer) than non-farmyard animals, while dogs and cats (both domestic pets), and lions and tigers (both wild big cats) are semantically the most similar to each other.

Word embedding data from larger pieces of text or linguistic corpora, when used in combination with clustering algorithms, can be used for purposes such as topic extraction and automatic document organisation, amongst other things.

Representing words as vectors also allows them to be used as parameters in various functions. For example, arithmetic can be performed using the vector representations of "*king*", "*man*", and "*woman*" to create a vector where "*queen*" is (theoretically) the most similar word to the vector created (Srinivasa-Desikan 2018). This is shown in Eq. (3.14) where $V(example)$ represents the word embedding of "*example*".

$$V(king) - V(man) + V(woman) \approx V(queen) \tag{3.14}$$

As word embeddings use a large number of dimensions, it can be difficult to visualise their vector data. Algorithms such as *t-distributed stochastic neighbour embedding* (t-SNE) (Van der Maaten and Hinton 2008) can be used to aid such visualisation, as shown in Figure 3.8.

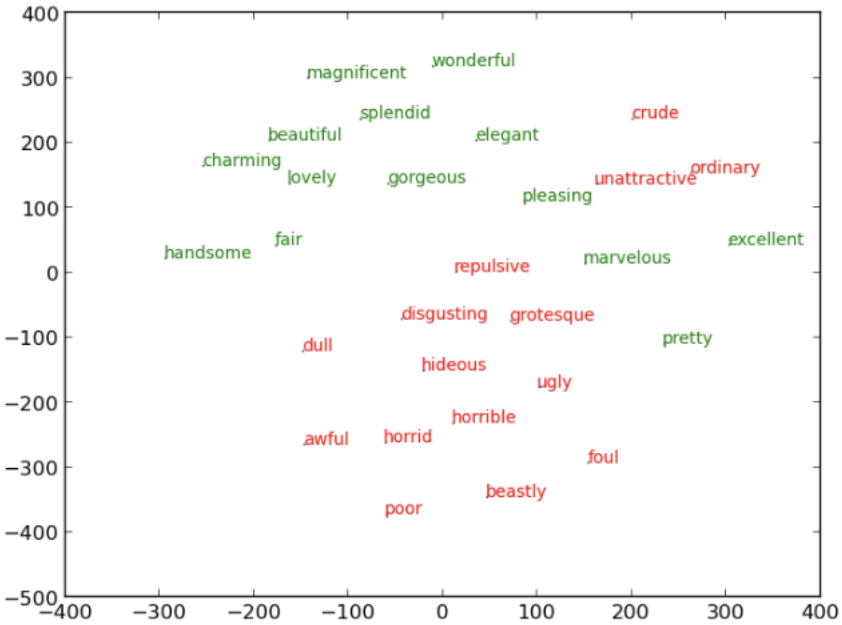


Figure 3.8: A 2D t-SNE projection of various word embeddings created by Chung and Glass (2018). Words with positive sentiment are coloured in green and words with negative sentiment are coloured in red, with positive/negative clusters more easily visible thanks to t-SNE visualisation.

Examples of word embedding tools include Word2Vec, FastText and WordRank (Srinivasa-Desikan 2018). SpaCy's in-built word embedding tool uses GloVe vectors.

3.3.3 Data Preprocessing Techniques

When raw text data are extrapolated for NLP purposes, it requires preprocessing through a normalisation technique called *cleaning*, in which the raw text data are manipulated to be suitable/meaning for NLP (Srinivasa-Desikan 2018).

The exact cleaning algorithms will differ depending on a text's source, with Kraaijeveld and De Smedt (2020) commenting that researchers in this field use *tokenisation*, *stemming*, and *stop-word removal* to clean Twitter data.

This section aims to describe those techniques described by Kraaijeveld and De Smedt (2020), as well as some additional techniques used in the field, as well as some of those used by NLTK and SpaCy as part of their own data preprocessing.

Tokenisation is a process in which text is split into meaningful segments called *tokens* (Srinivasa-Desikan 2018). These segments may not just be limited to words, with many tokenisation algorithms, such as the one used by SpaCy, taking punctuation and clitics into account. Figure 3.9 shows an example of tokenisation algorithm.

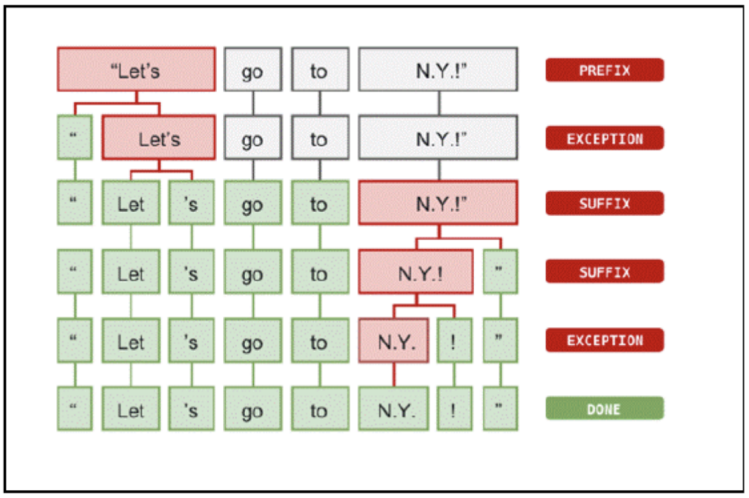


Figure 3.9: An example of how a string is split into several tokens using a tokenisation algorithm, taken from Srinivasa-Desikan (2018).

Stemming refers to the process in which inflected and derivational suffixes are removed from words, so that only the stem of a word is left. Whilst such morphemes can generally be removed through pattern matching, irregular forms can make this difficult (Srinivasa-Desikan 2018). A related technique is called *lemmatisation*, in which instead of a stem, the root word, called a *lemma*, is left instead.

Stop words are words which are judged, depending on a criterion, to have little to no significant semantic value, such as "*a*", "*the*", and "*my*". Stop words are therefore removed during the preprocessing phase.

Twitter data in particular contain information such as links and non-alphanumeric characters that are not relevant for sentiment or semantic analysis; part of the cleaning process is therefore removing these characters. That being said, information from the use of non-alphanumeric characters may be extracted using more advanced models (Chen et al. 2018; Srinivasa-Desikan 2018) and are recognised as tokens by NLTK and SpaCy.

Other preprocessing techniques include *part-of-speech* (POS) *tagging*, in which tokens are marked with the grammatical category they represent (e.g. noun, verb, adjective) and *named entity recognition*, in which real-world objects such as people, countries and companies are marked according to their noun type (Srinivasa-Desikan 2018).

Chapter Four
Machine Learning Model Creation

In order to create profitable algorithmic trading software, it is first necessary to find appropriate machine learning algorithms that can use data from Twitter to predict cryptocurrency prices effectively.

4.1 API Limits

In an ideal setting, it would have been possible to create machine learning models that use both tweet numbers and sentiment/vector data as their input variables; however, API limits and the time frame of this study did not allow such models to be created without severe data implications.

Historic data concerning the number of tweets posted per day are available to access from BitInfoCharts (BitInfoCharts 2022); however, these data cannot be broken down into smaller time intervals. Other ways of gathering data concerning tweet numbers in smaller time intervals were explored, such as collecting primary data using Twitter's API. This was deemed infeasible, though, due to Twitter API limits on the number of search queries one can make in given time intervals.

Similarly, specific tweets can be accessed through Twitter's API search function; however, Twitter limits its search function to the past seven days on the standard API. It is possible to apply for an academic developer account on Twitter, in which permission is granted to use the search function beyond the last seven days. An application for this was submitted to Twitter; however, it was unfortunately rejected.

The decision was therefore taken to explore the modelling of two different time intervals that use a different set of inputs. The first of these, described in Section 4.2, explores hourly price prediction methods using raw Twitter data using sentiment analysis and word vector data. Section 4.3 explores daily price prediction methods with data gathered from historic tweet numbers.

4.2 Part 1: Hourly Twitter Analysis

The goal of this section is to describe the process of creating a machine learning model that can predict hourly cryptocurrency price movements to generate a profitable algorithmic trading strategy based on hourly analyses of raw Twitter data, and analyse its performance.

4.2.1 Data Collection

Tweets were collected using Tweepy, a Python library used for accessing the Twitter API. A script was written that could collect up to 400 tweets, all written in English, per hourly interval: a maximum of 100 for each of the keywords "*bitcoin*", "*ethereum*", "*dogecoin*", and "*cardano*" respectively[8]. These tweets were then added to corresponding PostgreSQL tables (one for each cryptocurrency) within a single database.

The first version of this script simply collected any English language tweet that contained these keywords; however, this resulted in the data set being overwhelmed by spam. Common features of the spam-filled database include the overuse of hashtags, cashtags[9], links, retweets, as well as words like "*giveaway*" and "*donate*".

A second version of the script, a demonstration version of which is able to be viewed in the GitLab repository under *tweet_ collector_ demo.py* was written to filter this spam, so that only tweets that did not contain the aforesaid spam features were added to the database[10]. Tweets were collected between 9$^{\text{th}}$ November 2021 00:00:00 and 21$^{\text{st}}$ November 2021 23:59:59 UTC.

This approach can be criticised as many non-spam tweets would be excluded from the analysis, including hashtags and retweets which are core components of the culture that exists on Twitter (Kraaijeveld and De Smedt 2020). However, it can be argued that it was necessary to filter out such tweets to avoid the spam.

As well as sentiment data, hourly price data also needed to be collected. Raw price data were collected using CSV files downloaded from CryptoDataDownload (2022), that contained the opening and closing prices for each hourly interval (as well as other information), with price change and price direction[11] being extracted using the formulae in Eqs. (4.1) and (4.2) respectively, using opening and closing price data.

$$price\ change = \frac{closing\ price - opening\ price}{opening\ price} \cdot 100\% \tag{4.1}$$

$$price\ direction = \begin{cases} 1, & closing\ price > opening\ price \\ 0, & closing\ price \leq opening\ price \end{cases} \tag{4.2}$$

The question as to which of these price metadata variables can be predicted more effectively will be explored through creating machine learning models for each of these variables.

[8] A maximum of 100 was set as when these data were collected, Twitter only allowed API users to collect 500,000 tweets per month. The version of the script used in the final analysis could have theoretically collected 124,800 tweets in total over its collection period, and was written after several test programs and earlier script prototypes had been written. This limit has since been raised to 2,000,000 tweets per month with the launch of Twitter Dev 2.0.

[9] Cashtags refers to the use of company or (crypto)currency's ticker symbol on Twitter, e.g. $TSLA, $GBP, or $BTC. Cashtags are used on Twitter in much the same way as hashtags.

[10] This technically does not include cashtags as filtering for cashtags is an advanced filter not available for use on the standard API. Tweets containing cashtags were filtered after collection. For more information, see *tweet_ collector_ demo.py* in the GitLabs repository

[11] More accurately, this can be described as *positive price direction*. Price direction is stored as $\in \{0, 1\}$ as it is a binary variable

4.2.2 Tweet Cleaning

After the tweets were collected, each tweet had to be preprocessed through a process called cleaning (see Section 3.3.3). A diagram of showing the process of how the raw tweets were cleaned can be viewed in Figure 4.1.

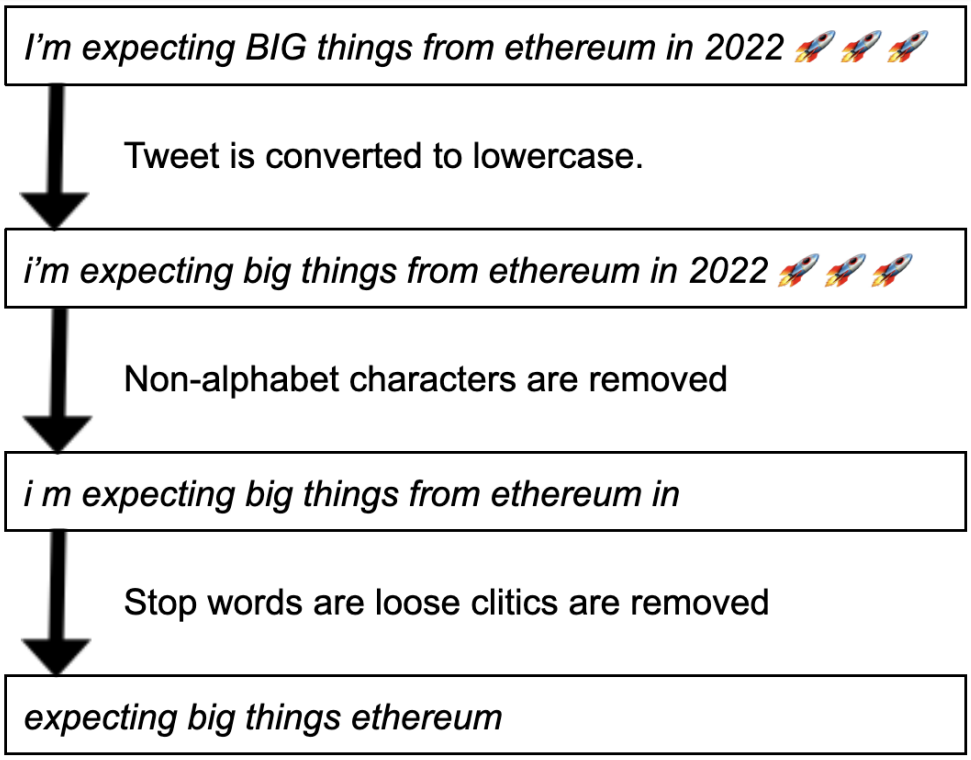


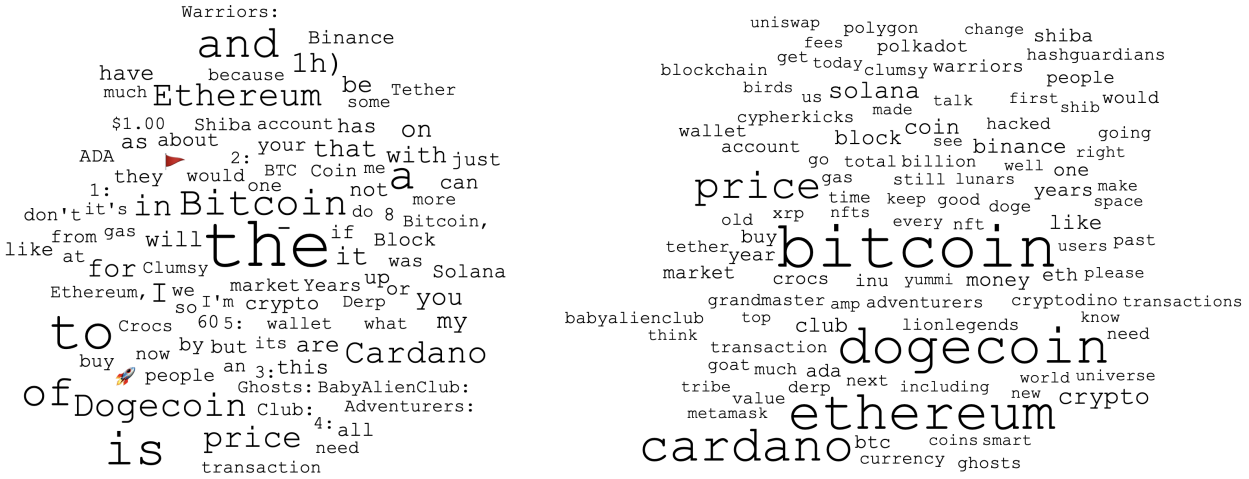
Figure 4.1: Diagram showing how tweets were cleaned.

This process was realised using the *clean_tweet* function, as part of the *tweet_cleaner.py* file available to view on GitLabs. The exact code snippet of *clean_tweet* is replicated with comments in Figure 4.2, in which *t* refers to a tweet represented as a string, and *stop_words* is an array of stop words and loose clitics provided as part of NLTK Python library. The string is split and then joined again to remove excess whitespace.

```
def clean_tweet(t):
    t = t.lower()                                #Makes entire string lowercase
    t = re.sub('[^a-z]',' ',t)                   #Removes non-alphabet characters
    t = t.split()                                #Turns string into an array
    t = [w for w in t if not w in stop_words] #Removes stop words
    t = ' '.join(w for w in t)                   #Turns array into string
    return t
```

Figure 4.2: Python code snippet of a tweet cleaning function.

The stark differences between the content of the Twitter database before and after tweets being cleaned can be seen in the word clouds in Figure 4.3.



(a) Word cloud before cleaning

(b) Word cloud after cleaning

Figure 4.3: Word cloud representation of the tweet database before and after the tweets were cleaned.

4.2.3 Sentiment Analysis and Word Embeddings

After each of the tweets in the database had been cleaned, more data preprocessing was required to extract NLP analysis data and create data sets of numeric inputs suitable to use in machine learning algorithms.

VADER, as aforementioned in Sections 2.4.1 and 3.3.1, was used to generate a sentiment polarity using its in-built *sentiment intensity analyser* (SIA). This SIA takes a string as an input, and outputs a dictionary of sentiment polarity scores between 0 and 1 for positive, negative, and neutral sentiment.

As described by Kraaijeveld and De Smedt (2020) and Huang et al. (2021), cryptocurrency communities use their own dialect features and slang, as well as financial terms. The SIA's lexicon is therefore adjusted slightly before use, in order to accommodate these terms and their connotations. The full list of terms added are available to view in the *tweet_cleaner.py* code, and are gathered from the database itself, as well as informal online sources. Examples include *bear* and *bearish* (words used in the world of economics to describe pessimistic investors and traders who expect the value of a stock, commodity, or cryptocurrency, or markets overall to fall) as well as *moon* (a term used in online trading communities that refers to the value of an asset rising by a large amount).

The positive, negative and neutral scores of each tweet were collected, with each tweet also being assigned a "*positive*", "*negative*", or "*neutral*" label depending on which polarity score was highest. The scores were then aggregated on an each-hour basis to create the data labels in Table 4.1 for each hourly data point.

Data label	Description
POS_PCT	The percentage of tweets labelled as having positive sentiment.
POS_AVG	The average positive sentiment score.
NEU_PCT	The percentage of tweets labelled as having neutral sentiment.
NEU_AVG	The average neutral sentiment score.
NEG_PCT	The percentage of tweets labelled as having negative sentiment.
NEG_AVG	The average negative sentiment score.

Table 4.1: Data labels created from the sentiment analysis of tweets.

In addition to the sentiment analysis data, data labels were created for each hourly interval based on word embedding data. This was done using SpaCy's inbuilt vector attribute, in which a string's average word embedding is mapped as a 300-dimension vector.

The average word vector was collected for each tweet, with the average word vector for tweets posted in an hourly interval stored to create the data labels in Table 4.2, where `v` refers to the 300-dimensional average word vector for tweets posted in an hourly interval.

Data label	Description
v0	The value of `v[0]`.
v1	The value of `v[1]`.
...	...
v299	The value of `v[299]`.

Table 4.2: Data labels created from the word embeddings of tweets.

4.2.4 Machine learning model creation

To create a data set suitable for use within a supervised machine learning algorithm, the average hourly sentiment and vector data described in Section 4.2.3 were merged with corresponding timestamp and price data to form a data set, stored as a CSV file, for each cryptocurrency.

It is important to note at this stage that the goal of this research is to analyse whether cryptocurrency prices can be predicted using Twitter data, not explore whether there is a general correlation between Twitter data and price metadata overall. While these aims are similar in nature, the marked difference here is the use of *lag*. Hence, for each timestamp in the data set, sentiment and vector data are collected from tweets posted within the hour *before* a timestamp, while price data refer to closing prices collected one hour *after* a timestamp. This is best demonstrated through the timeline in Figure 4.4.

An extract from Bitcoin's data set is available to view in Table 4.3, in which the data set's labels and a row of data are visible.

These data sets were then *normalised*, with three separate data sets created in each case so that price, price change, and price direction predictions can be analysed separately.

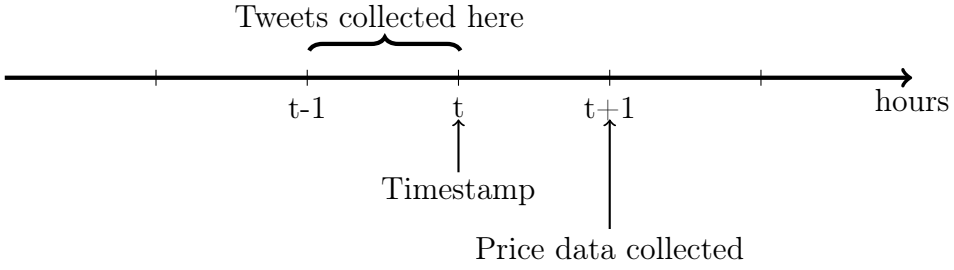


Figure 4.4: Timeline demonstrating the lag between the time tweets are collected and the time prices are predicted against a timestamp label.

Timestamp	**POS_ PCT**	**POS_ AVG**	**...**	**v0**	**v1**	**...**	**Price**	**Price_ Change**	**Price_ Direction**
2021-11-09 01:00:00	0.057164	0.1334536	...	-0.0885112	0.1216791	...	67508.40	-0.140377	0

Table 4.3: Example of a row from the machine learning data set created from Bitcoin tweets and hourly price data.

Twelve normalised data sets were created in total: one for each combination of cryptocurrency and price prediction metric, from the sets {*Bitcoin*, *Ethereum*, *Cardano*, *Dogecoin*} and {*price*, *price change*, *price direction*} respectively.

WEKA, described in Section 3.2.7, was used to carry out feature selection for of the normalised data sets. After a normalised data set was uploaded to WEKA, the attribute variables' PCC was then evaluated against price metrics, with the top 5 best performing attributes, i.e. those with the highest absolute PCC values, shown in Table 4.4.

Word embeddings are much more prominent in Table 4.4 than sentiment data, with negative tweet percentage the only sentiment attribute to appear, which it does so in only one correlation ranking. While there is insufficient evidence to draw major conclusions from this, it may be an indicator that word embedding data are more useful as attributes than sentiment data when used in cryptocurrency price prediction models. Nonetheless, there are considerably more word embedding attributes than sentiment attributes overall, so more evidence would be needed to justify this claim.

The large correlation scores for cryptocurrency price data can best be explained by looking at the charts in Figure 4.5 that show cryptocurrency prices over the period of the data collection period.

Cryptocurrency prices in general show a clear negative trend in Figure 4.5, meaning any attribute showing consistent growth or decline over the collection period would have a high absolute PCC score, even when such trends are less obvious when the attribute and output are compared directly. Such an observation can be visualised in Figure 4.6, in which *price*, `v104`, and timestamp are compared for Bitcoin tweets.

Models were then created for each of these data sets using Auto-WEKA, with MAE used as the loss function. Auto-WEKA's algorithm was allowed to run for 15 minutes each time, at which point the differences between loss function outputs per iteration were minuscule.

Bitcoin Price

Data label	MCC
timestamp	-0.913
v104	0.900
v291	0.892
neg_pct	-0.884
v103	0.883

Bitcoin Price Change

Data label	MCC
v15	-0.103
v271	0.064
v71	0.055
v282	0.052
v189	0.052

Bitcoin Price Movement

Data label	MCC
v147	0.087
v226	0.076
v24	0.071
v160	0.068
v91	0.064

Ethereum price

Data label	MCC
v137	-0.914
v146	0.897
v145	-0.859
v289	-0.854
v19	0.851

Ethereum price change

Data label	MCC
v91	-0.081
v34	-0.076
v296	-0.069
v171	0.061
v97	0.061

Ethereum price movement

Data label	MCC
v91	0.092
v34	0.086
v171	0.063
v276	0.054
v61	0.053

Dogecoin price

Data label	MCC
timestamp	-0.923
v236	0.784
v80	0.780
v170	-0.749
v4	-0.694

Dogecoin Price Change

Data label	MCC
v262	-0.113
v23	0.109
v58	0.103
v64	-0.097
v117	0.094

Dogecoin Price Movement

Data label	MCC
v19	0.109
v151	0.097
v262	0.091
v281	0.089
v23	0.083

Cardano price

Data label	MCC
v215	-0.928
v237	-0.924
v51	0.923
v86	0.922
v83	-0.922

Cardano Price Change

Data label	MCC
v263	0.143
v84	-0.121
v163	-0.109
v91	-0.093
v29	0.093

Cardano Price Movement

Data label	MCC
v262	0.149
v84	0.124
v82	0.106
v163	0.104
v83	0.102

Table 4.4: Top 5 most correlated attributes for each cryptocurrency and price prediction metric combination.

(a) Bitcoin (BTC) price chart

(b) Ethereum (ETH) price chart

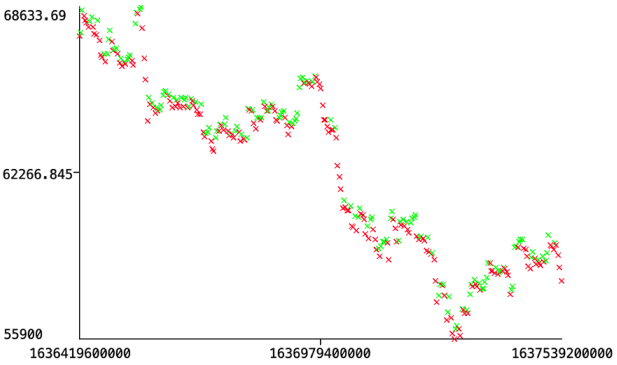
(c) Dogecoin (DOGE) price chart

(d) Cardano (ADA) price chart

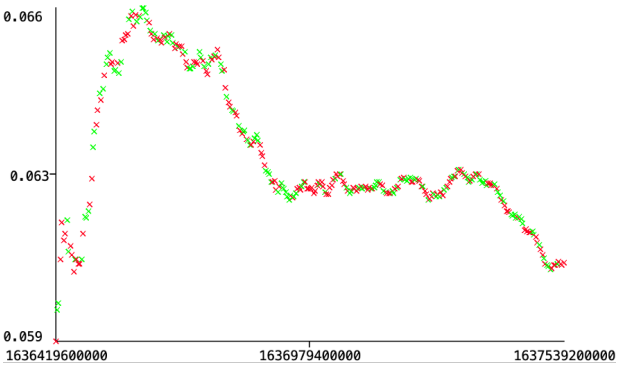
Figure 4.5: Price charts for each of the cryptocurrency used in this study compared against the US dollar (USD). Chart shows prices between 9th and 22nd November 2021. Figures taken from Yahoo Finance (2022).

4.2.5 Model performance

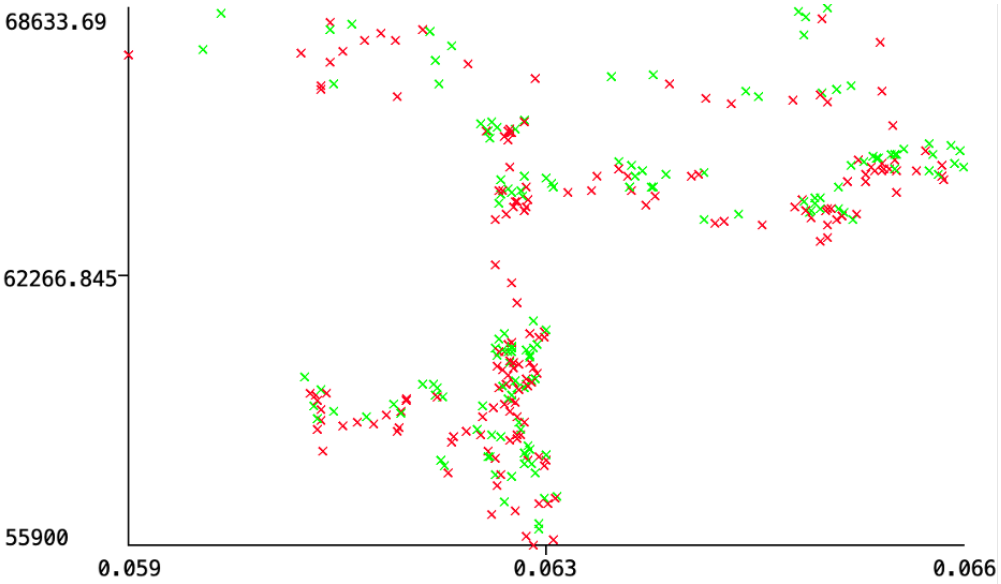
After the models were created, the training data were used to analyse each model's performance. These reevaluations can be seen in Figures 4.7 and 4.8, and Tables 4.5 and 4.6, in which the regression models (those used to predict *price* and *price change* performances are graphed, while the classification models (those used to predict *price direction*) performances are shown via confusion matrices. The evaluation metrics discussed in Section 3.2.4 are also shown.



(a) Bitcoin price (y-axis) plotted against timestamp (x-axis)



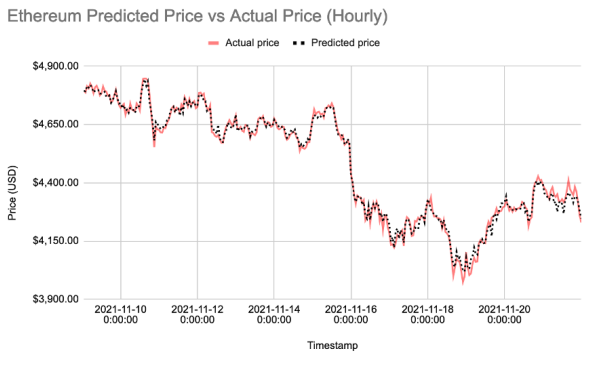
(b) `v104` value (y-axis) plotted against timestamp (x-axis)



(c) Bitcoin price (y-axis) plotted against `v104` value (x-axis)

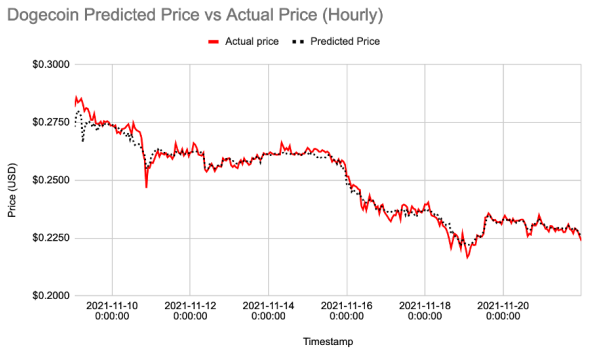
Figure 4.6: Chart exploring the correlation between Bitcoin price and `v104` value. Timestamps are in epoch (millisecond) form. Hourly positive (green) and negative (red) price direction is also plotted for each sample. Subfigures (a) and (b) show that both Bitcoin price and `v104` value generally decline throughout the data collection phase and have negative correlation. However, subfigure (c) shows there is little true correlation between the two variables. Graphs generated using WEKA (Frank et al. 2016).

(a) Bitcoin (BTC) hourly price prediction and actual price
MSE: 173,398.23
RMSE: 417.01
MAE: 264.11
MAPE: 0.43%
MAAPE: 0.0043 rad



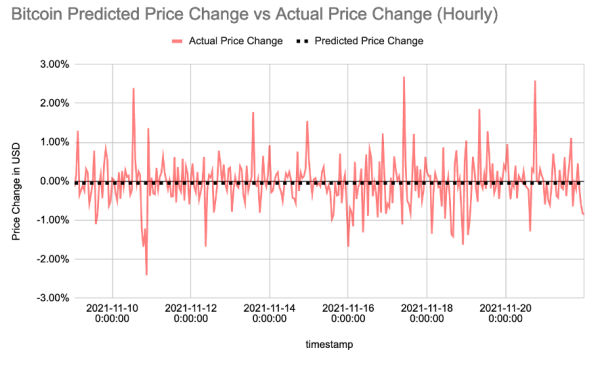
(b) Ethereum (ETH) hourly price prediction and actual price)
MSE: 326.38
RMSE: 18.07
MAE: 12.89
MAPE: 29%
MAAPE: 0.29 rad

(c) Cardano (ADA) hourly price prediction and actual price
MSE: $4.7 * 10^{-5}$
RMSE: 0.0069
MAE: 0.0054
MAPE: 0.27%
MAAPE: 0.0027 rad

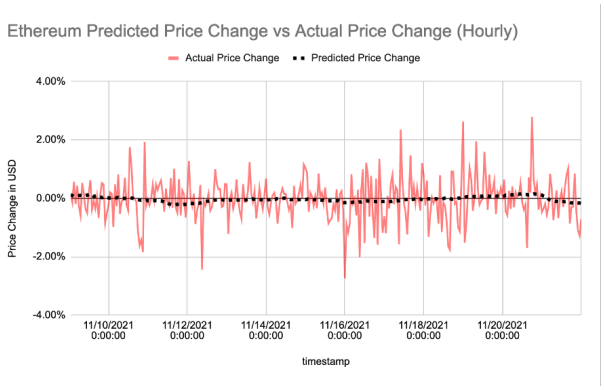


(d) Dogecoin (DOGE) hourly price prediction and actual price
MSE: $6.1 * 10^{-6}$
RMSE: 0.0024
MAE: 0.0016
MAPE: 0.11%
MAAPE: 0.0011 rad

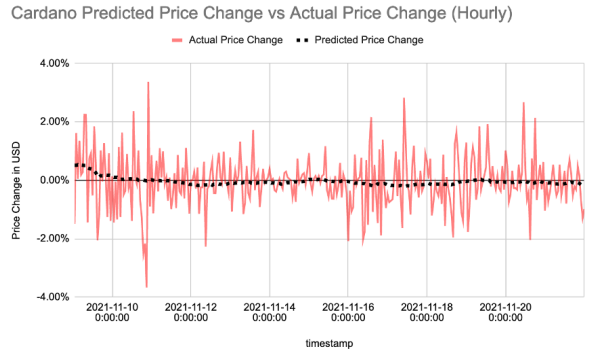
Figure 4.7: Hourly price chart for each cryptocurrency (red) plotted against the predictions made by Auto-WEKA's regression model for the evaluated data (black, dotted).



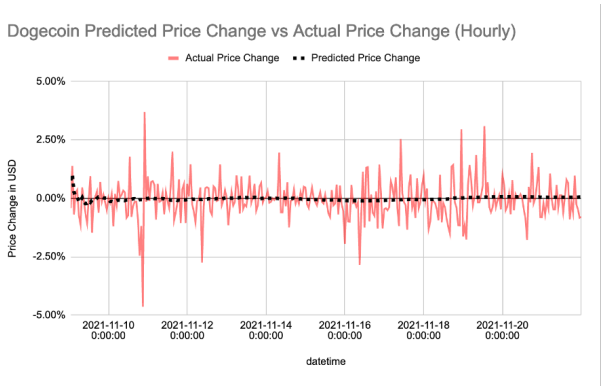
(a) Bitcoin (BTC) hourly price change prediction and actual price change
MSE: 0.39
RMSE: 0.62
MAE: 0.44
MAPE: 107%
MAAPE: 0.77 rad



(b) Ethereum (ETH) hourly price prediction and actual price)
MSE: 0.53
RMSE: 0.73
MAE: 0.53
MAPE: 139%
MAAPE: 0.76 rad



(c) Cardano (ADA) hourly price prediction and actual price
MSE: 0.82
RMSE: 0.90
MAE: 0.66
MAPE: undefined
MAAPE: 0.77 rad



(d) Dogecoin (DOGE) hourly price prediction and actual price
MSE: 0.66
RMSE: 0.90
MAE: 0.66
MAPE: undefined
MAAPE: 0.77 rad

Figure 4.8: Hourly price chart for each cryptocurrency (red) plotted against the predictions made by Auto-WEKA's regression model for the evaluated data (black, dotted).

BTC	*Actually positive*	*Actually negative*
Predicted positive	0	0
Predicted negative	142	170

ETH	*Actually positive*	*Actually negative*
Predicted positive	28	18
Predicted negative	121	148

ADA	*Actually positive*	*Actually negative*
Predicted positive	0	0
Predicted negative	136	176

DOGE	*Actually positive*	*Actually negative*
Predicted positive	107	36
Predicted negative	29	140

Table 4.5: Confusion matrices for each cryptocurrency, showing to what extent models could predict the (positive) price direction of a cryptocurrency.

	Bitcoin	*Ethereum*	*Cardano*	*Dogecoin*
Accuracy	54.5%	55.45%	56.41%	79.17%
F_1 *measure*	N/A	0.261	N/A	0.787
MCC measure	N/A	0.091	N/A	0.579

Table 4.6: Evaluation metrics for the hourly classification models based off the confusion matrices in Table 4.5.

4.2.6 Performance Analysis

Those classification models which aimed to capture price movement suffered as a result of the downward market trend, causing imbalances within the training data, with hourly closing prices being down 54.3% of the time[12]. This, combined with the fact that attribute correlation was low in general, meant that most of these models were biased towards predicting negative price movements regardless of attribute figures in order to optimise the cost function. That being said, the Dogecoin model appears to be the exception to this and performs remarkably well. More preprocessing should be carried out in future work to balance databases such as these.

The regression models which aimed to capture price change performed poorly, as predictions tended to be near-zero values. While cryptocurrency prices are volatile, hourly price movements are generally very small, with the average absolute price change being 0.57% within the data set. The average price change in the data set is -0.05%, though this figure is distorted due to positive and negative price changes balancing each other out. This distortion, as well as the fact hourly price volatility is generally low to begin with, means that these models trend towards picking close-to-zero values to minimise cost if predictions

[12]The exact figures work out as follows: Cardano and Ethereum hourly closing prices were down 53.2% of the time; Bitcoin was down 54.5% of the time; Dogecoin was down 56.4% of the time.

cannot be made.

The regression models that predict price appear to work sufficiently well from the charts; however, these data are overfitting. Reevaluated data are being used as opposed to unseen data, so stronger conclusions can be made when they are applied to unseen data. Using k-fold cross-validation with a model that predicts price was the wrong decision to make, as the price prediction problem can then be viewed as a time-forecasting one. As well as this, the data were collected at a point when cryptocurrency prices were generally declining; it is therefore possible that potential attributes that display stronger correlation over varying long-term market conditions were overlooked in favour of those exhibiting short-term, yet overall consistent, growth or decline throughout the data collection period.

4.3 Methodology Part 2: Daily Twitter Analysis

This subsection's goal is to describe the process of creating the second round of machine learning models, used to predict daily cryptocurrency price movements using tweet volume data. The data collection procedure is described, and the performance of the models created analysed.

4.3.1 Data collection

Collecting daily data was much simpler than collecting hourly data due to the types of attributes being collected. The only attributes used in the daily data concerned tweet volume, thus all that was required was finding these figures.

Data concerning tweet volume were collected from BitInfoCharts (2022), where daily figures that show a cryptocurrency's tweet volume, i.e. the number of tweets posted on Twitter in a given day that include a cryptocurrency's name, are graphed. A screenshot of the website is shown in Figure 4.9, in which the site is being used to display and compare the tweet volumes of Bitcoin, Ethereum, Cardano and Dogecoin over time.

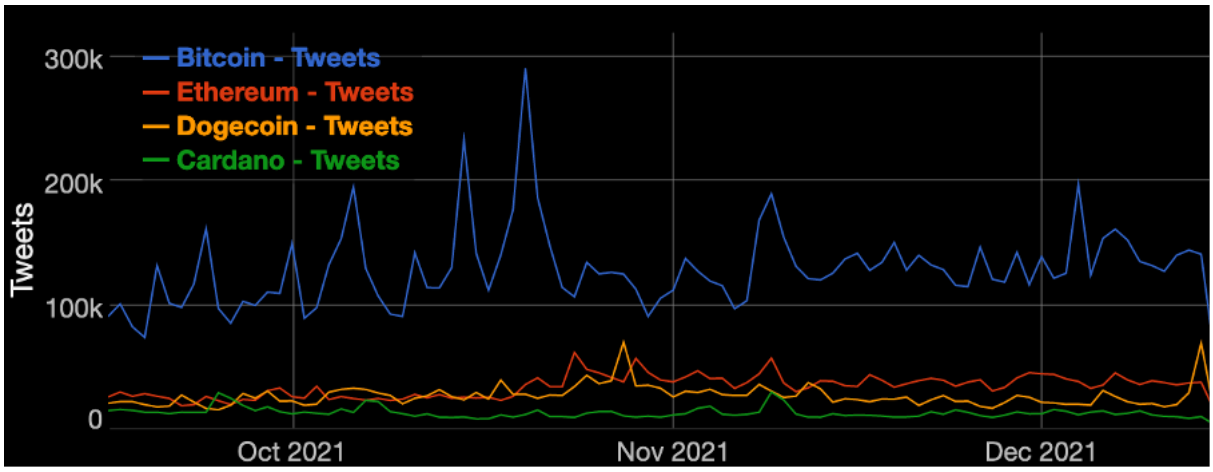


Figure 4.9: A screenshot from BitInfoCharts (2022) comparing the tweet numbers of Bitcoin, Ethereum, Dogecoin, and Cardano over a three month period.

While these data are publicly available, they are not available in a downloadable format, thus the data had to be scraped from the page's HTML source code using a Python script. This is done using the *get_most_recent_datapoint(currency)* function within the *trading_bot.py* script as can be seen in GitLabs.

As in Section 4.2, cryptocurrency price data were collected from CSV files available from CryptoDataDownload (2022), with daily price data either extracted from the files directly or using the formulae in eqs. (4.1) and (4.2).

4.3.2 Machine learning model creation

As mentioned, the goal of this research is *prediction*, so a similar lag that was applied in Section 4.2.4 is also applied here, in which volume data refer to the number of tweets posted in the 24 hours previous to a timestamp, while price data use the closing statistics collected 24 hours after a timestamp. The timeline used in Figure 4.4 can also be used here, but a *daily* timescale needs to used.

In order to create a data set that was suitable for use with Auto-WEKA, tweet volume data scraped from BitInfoCharts using Python was then merged with timestamp and price data to form a CSV data set for each currency. An extract from Ethereum's data set is available to view in Table 4.7, in which the data set's labels and example data are visible.

Timestamp	**Tweet_Volume**	**Price**	**Price_Change**	**Price_Direction**
2017-08-18	23809	293.96	-2.68%	0

Table 4.7: Example of a row from the machine learning data set created from Bitcoin tweets and hourly price data.

A similar normalisation process to that described in Section 6.2.4 then took place, although an additional step was added. A further variable, *Tweet_Change*, was created, referring to the percentage increase/decrease in the number of tweets posted referring to a cryptocurrency, calculated using Eq. (4.3).

$$tweets\ change = \frac{tweets\ today - tweets\ yesterday}{tweets\ yesterday} \cdot 100\% \tag{4.3}$$

Feature selection was less of an issue with these data sets, due the fact only three attribute variables are used: *timestamp*, *tweet volume*, and *tweet change*. It was decided on this basis to use all three attributes, as feature selection was deemed unnecessary.

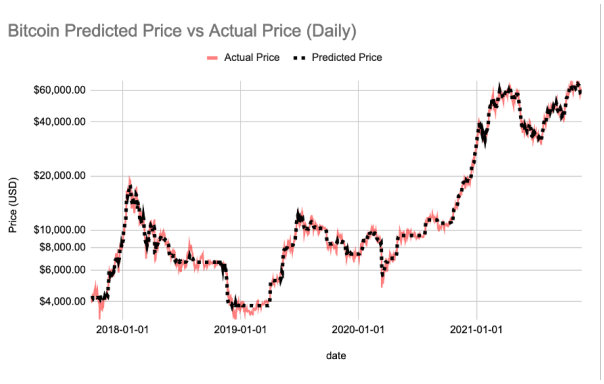
As was done with the hourly models, each data set was uploaded to WEKA for use in Auto-WEKA, which was allowed to run for 15 minutes each time with MAE used as a loss function.

4.3.3 Model performance

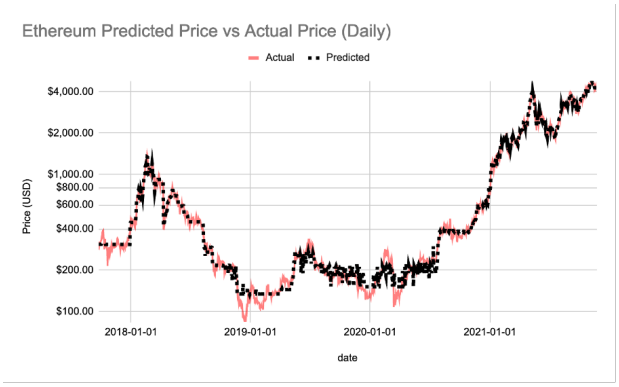
Again, after the models were created, the training data were used to analyse the performance of each model. Although Auto-WEKA created a model for price changes for each cryptocur-

rency's, in all cases the model could not run. Given the poor performance of the hourly price change model and the amount of noise in the data, it is hardly surprising.

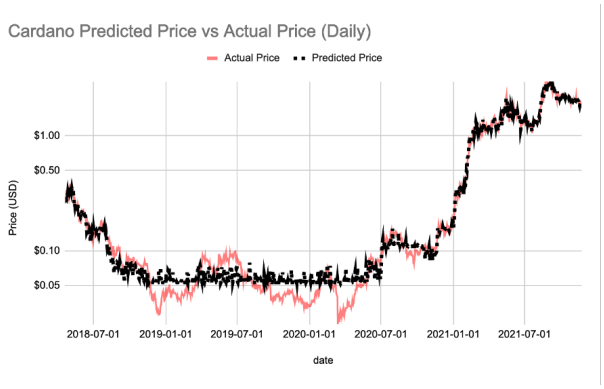
The performance of each model is available to view in Figure 4.10 and Tables 4.8 and 4.9.



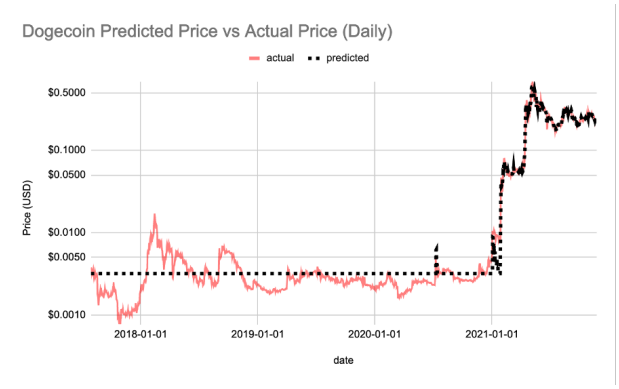
(a) Bitcoin (BTC) daily price change prediction and actual price change
MSE: 229562
RMSE: 479.13
MAE: 362.20
MAPE: 4.61%
MAAPE: 0.046 rad



(b) Ethereum (ETH) hourly price prediction and actual price)
MSE: 1009.81
RMSE: 32.71
MAE: 21.77
MAPE: 4.4%
MAAPE: 0.044 rad



(c) Cardano (ADA) daily price prediction and actual price
MSE: $2.3 * 10^{-4}$
RMSE: 0.015
MAE: 0.013
MAPE: 19.55%
MAAPE: 0.18 rad



(d) Dogecoin (DOGE) hourly price prediction and actual price
MSE: $8.4 * 10^{-6}$
RMSE: 0.0029
MAE: 0.81
MAPE: 81%
MAAPE: 0.87 rad

Figure 4.10: Daily price chart for each cryptocurrency (red) plotted against the predictions made by Auto-WEKA's regression model for the evaluated data (black, dotted). Y-axes use logarithmic scales in each subfigure.

BTC	*Actually positive*	*Actually negative*
Predicted positive	810	707
Predicted negative	0	42

ETH	*Actually positive*	*Actually negative*
Predicted positive	806	729
Predicted negative	4	20

ADA	*Actually positive*	*Actually negative*
Predicted positive	669	644
Predicted negative	0	0

DOGE	*Actually positive*	*Actually negative*
Predicted positive	0	0
Predicted negative	798	836

Table 4.8: Confusion matrices for each cryptocurrency, showing to what extent models could predict the (positive) daily price direction of a cryptocurrency.

	Bitcoin	*Ethereum*	*Cardano*	*Dogecoin*
Accuracy	54.65%	52.98%	50.95%	51.16%
F_1 *measure*	0.696	0.687	0.675	N/A
MCC measure	0.173	0.088	N/A	N/A

Table 4.9: Evaluation metrics for the daily classification models based off the confusion matrices in Table 4.8.

4.3.4 Performance Analysis

Similar problems existed for the daily classification models as the hourly classification models as again, the training data were imbalanced. Dogecoin was again the outlier as its model predicted all negative price movements, while the other cryptocurrency's models predicted mostly (or all in Cardano's case) positive price direction. That being said, this is only due to Dogecoin having more negative price changes than positive ones overall. Again, the imbalanced data set should be corrected in future work.

The issue with using MAE as a loss metric became apparent when analysing the daily price data. Predictions for Dogecoin and Cardano were flat when the currency was worth less than \$0.01; however, since their rise, the predictions have become more and more accurate. This is presumably due to larger losses being applied to more recent predictions when the price is higher. The price data can therefore be said to overfit the more recent data, and underfit the older data.

4.4 Model comparison & overall analysis

The data for hourly and daily models are very similar, with no major differences between the two sets. For both sets of models, predicting price change is too difficult a task due to

the choatic and noisy nature of the data.

Price direction data are also generally too difficult for a supervised machine learning to pick up a pattern, although Dogecoin's hourly price direction model did offer some interesting results that should be looked at further in future work.

While Abraham et al. (2018) argue sentiment tweet volume data are better than sentiment data at predicting price direction; the research here was unable to make that same conclusion.

4.5 Picking a model

At the end of the machine learning process, it was necessary to choose a machine learning model to use within the final software. As the hourly models did not perform well enough to justify a potentially 24-fold increase in trading fees, it was decided to use a daily model. None of the daily models worked particular well either; however, as the models predicting price had not been tested on any unseen data points yet, it was decided to use these within the software to test them on unseen data.

Chapter Five
Developing the trading bot

This section aims to describe the manner in which the machine learning tests in Chapter 4 were used to develop a simple piece of automated trading software. The runnable trading software is available to view in *trading_ bot.py*, and corresponding interface available to view by running *interface.py* - full instructions are available to view in the GitLab's *README.md* file.

5.1 Cryptocurrency Exchanges and API Key Setup

In order to write a program that makes automated cryptocurrency trades, it was necessary to sign up for a cryptocurrency exchange website that allowed users to make automated trades using APIs. Popular cryptocurrency exchanges include Coinbase, Kraken, and Gemini - though Binance was chosen as the exchange to use due to having lower trading fees and an extensive developer portal compared to other trading websites.

After API keys were set up, the account was funded with 100 USDT (see Section 3.1.5), with 25 USDT allocated for use with each cryptocurrency. Tether was used as opposed to a fiat currency due to the fees associated with converting between fiat and crypto on exchange websites.

5.2 Trading Algorithm

The algorithm that is used to make automated trades is fairly simple. If a model predicts that the price of a cryptocurrency is going to rise, then the cryptocurrency is bought if it has not already been done so. Similarly, if the price of a cryptocurrency is expected to fall, then the cryptocurrency is sold.

In terms of how this is actually carried out, *file_ name.py* does the following for each cryptocurrency when it is run:

- BitInfoCharts is scraped for the most recent information on tweet volume concerning the cryptocurrency in question[13]
- A temporary ARFF file is created using this information

[13]It is assumed that the file is run at 00:00:00 UTC as instructed. In the event it is not, tweet volume information will be underrepresented.

- A shell command that is executed that runs the cryptocurrency's associated machine learning model using WEKA and the data stored in the temporary ARFF file
- The model's prediction is scraped from the shell command's output
- The portfolio's current holdings are loaded from a YAML configuration file
- A trade decision is made (and if necessary, executed)
- The portfolio's configuration file is updated

This trading bot was designed to be run at 00:00:00 UTC each day, taking tweet volume data from the past 24 hours to make a prediction of the price 24 hours later. This was done locally using UNIX cron jobs and ensuring the computer was running at 00:00:00 UTC each day for the duration of the study; however, other options are available such as uploading the script to a cloud-computing service, or simply running the script manually at 00:00:00 UTC each day if the user is willing (though this can obviously be labour-intensive).

5.3 Interface

As the automated trading software is relatively simple in its design, an interface was created using Python, HTML, CSS, and JavaScript to display some of its core components. The main feature of the interface is a portfolio tracker, in which the portfolio's performance is assessed against a buy-and-hold strategy over time. This performance can be broken down into the performances of individual cryptocurrencies. Additional features include a market breakdown chart, using data gathered from the Binance API, that also shows whether a user is HODL-ing[14]. There is also a semi-live Twitter feed, in which 60 recent tweets (15 for each cryptocurrency) are cyclically shown on the feed, with the sentiment scores for each tweet shown underneath the feed. A screenshot of this interface can be viewed in Figure 5.1.

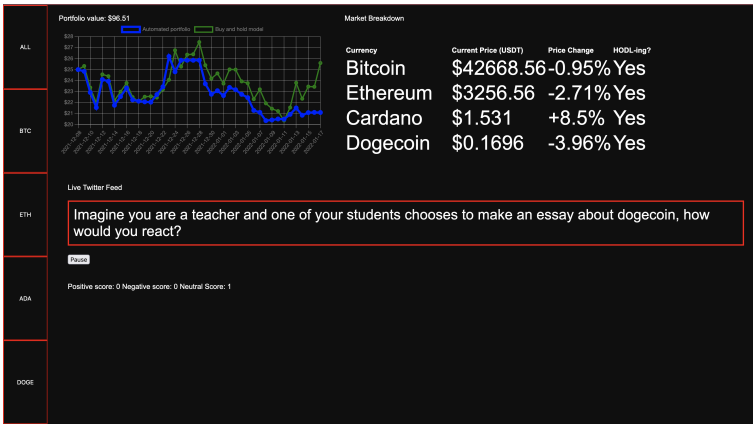


Figure 5.1: Screenshot of the web interface.

[14]HODL refers to a meme in the cryptocurrency community which originated as a misspelling of "*hold*" on a Bitcoin forum, the backronym "*hold on for dear life*" has since been coined.

As mentioned, this interface can be loaded by running *interface.py* - this seems strange given a HTML document is opened in the default browser at the end of this process, though there is good reason for this. Local files cannot be accessed using JavaScript for security reasons, thus data from configuration files are added to the *interface_ script.js* file using a file rewriting mechanism. Similarly, this file rewriting mechanism is also used to send data from the Tweepy, Binance-Python, and NLTK libraries to the JavaScript file, as these libraries cannot be used in JavaScript.

Code snippets and functions/methods from *interface.py* could easily be uploaded as backend code if the interface were hosted on the web.

5.4 Results so far

The performance of the algorithm have been somewhat disappointing, as very little movement has taken place. Since the 10th December 2021, following a price fall in the cryptocurrency market overall, the portfolio automated buy decisions for every cryptocurrency it was not already holding, and has not made a sell decision since. This is due to a combination of cryptocurrency prices falling despite tweet volume remaining consistent, and the model not correctly predicting the current bear market. This shows that a more advanced trading strategy is needed so that the model does not get left behind when prices shoot up or down. Possible ways of doing this could be avoiding using the "*all-in*" method used here and instead using formulae to allocate percentages of money.

It remains to be seen whether either cryptocurrency prices will rise or tweet volume will fall enough for the algorithm to make another sell decision. That being said, a buy-and-hold strategy also would have performed disappointingly, over the same period. The value as can be seen in Figure 5.1.

Chapter Six

Project Overview, Limitations, Future Work, & Reflection

The project overall is a good attempt at creating algorithmic trading software, and offers a useful proof of concept for those wanting to create algorithmic trading of their own. That being said, several improvements must be made before such software can be functionally used. This section aims to reflect on the project as a whole, and discuss those areas that can be improved. The main areas of improvement mainly resolve around the use of pre-built software, such as WEKA and VADER, and unsophisticated ways of dealing with some of the project's hurdles. The project's wide range comes at the cost of project depth, with more sophisticated methodologies recommended for future projects.

6.1 Data collection & Twitter

While the Twitter API proved to be a very useful tool, the ways in which it was used in this research can be improved in future studies. The rejection of an application for an academic Twitter account was the most damaging part of the data collection phase, as perks included higher API limits and the use of a search function that extended beyond the last seven days. The consequences of this, as well as ways data collection techniques can be improved are discussed throughout this section.

As highlighted by Kraaijeveld and De Smedt (2020), API limits mean that only a fraction of the full scope of tweets can be explored using the Twitter API. Potential ways of increasing this fraction include only focusing on one cryptocurrency so API resources do not have to be divided between coins, or creating additional developer accounts to bypass this limits (though doing so is ethically ambiguous since it this would constitute as spamming). Furthermore, this study only makes use of tweets in English for practical purposes. Higher-level analysis could make use of tweets in multiple languages to more accurately capture *globalised* sentiment.

An additional problem concerning Twitter was the problem of spam tweets infiltrating the database. The solution used in this project was to simply avoid searching for tweets that included "*spammy*" features; however, more sophisticated and reliable criteria could be applied. Machine learning has been successfully deployed in the field of spam-detection (Knox 2018), thus similar solutions could potentially be implemented. Duplicate or near-duplicate tweets also made their way into database, so in future work, developers should make use of

Levenshtein distance to counteract this (Yujian and Bo 2017). Furthermore, data relating to spam (i.e. volume of spam tweets or spam tweet percentage overall) may be useful attributes in price prediction models, especially given the role bots play in creating publicity of smaller coins in cryptocurrency pump and dump schemes (Li et al. 2021; Xu and Livshits 2019). Whilst technically not falling under the remits of spam, some tweets referring to Bitcoin Cash ended up part of the Bitcoin tweet database due to the use of "*Bitcoin*" in the forked currency's name. A similar issue arises with Cardano, in which tweets about the blockchain's namesake, Gerolamo Cardano (1501-1576) were added to the Cardano database. Whilst this problem was only minor and did not appear to have a major effect on the database, more sophisticated search queries could be designed to eliminate this problem completely.

6.2 Machine Learning & WEKA

The use of WEKA and Auto-WEKA allowed machine learning models to be created easily without needing to write several lines of code, which is beneficial for those developers wanting to learn more about how machine learning works who feel they lack the necessary coding skills. However, while this tool saved lots of time, this was counteracted by the fact the study was inhibited in other ways. Firstly, as WEKA uses Java, models could not be run in Python directly. While wrapper libraries such as python-weka-wrapper3 exist, as well as cross-language implementations such as Jython, these had there own drawbacks; the only feasible solution that was found was running shell commands that ran Java and ergo WEKA, which was very inefficient. Furthermore, while WEKA offered a wide range of machine learning algorithms, the literature makes references to RNNs and LSTM models that would have been beneficial to explore.

The choice of machine learning algorithms can also be criticised. While a case can be made that predicting price change percentages and/or price direction does not constitute as time-forecasting, this cannot be said of those models that predicting raw price values. More sophisticated algorithms that are more suitable for time-forecasting should be used, as opposed to the standard algorithms used by Auto-WEKA, especially when timestamps are used as attributes, as was done for all the daily models and some of the hourly ones.

Upon reflection, WEKA would have better been used as a preliminary testing tool as opposed to a tool used throughout the whole project.

6.3 Sentiment Analysis and VADER

While VADER proved to be a useful sentiment analysis tool, it was not perfect. Many valid criticisms can be made of the use of VADER, despite its popularity within the literature as a go-to tool.

One problem with using VADER is the polarity scores refer to the sentiment of a tweet overall, as opposed to the sentiment of the cryptocurrency within it. This leads to potential discrepancies when a positive opinion is expressed using vocabulary that VADER perceives to be negative or vise-versa. This can be an especially prominent issue when sarcasm is used; take the following tweet from the Bitcoin database as an example:

> "Love the fact bitcoin uses more energy than switzerland, fuck crypto man what a fucking joke"

This tweet clearly expresses negative sentiment; however, this negative sentiment is not picked up by VADER, which gives the tweet a positive score of 0.385 compared to a negative score of 0.148.

Furthermore, some tweets were actually irrelevant to the cryptocurrency mentioned, or were affected by the mention of the cryptocurrency as a comparative tool. Consider an example taken from the Dogecoin database, in which a different cryptocurrency, Shiba, is being discussed:

> "I knew about shiba when nobody was talking about it .. when they was calling it the dogecoin killer ... I hope this listing on robinhood causes another spike"

As can be seen, Dogecoin is not the focus on this tweet at all, yet the tweet still featured in the Dogecoin tweet database.

While adding additionally lexica from financial and cryptocurrency online communities can be seen as a major advantage of this study, the scores were assigned somewhat arbitrarily, with each positive word given a score of +2.0, and each negative word given a score of -2.0. Future work should using VADER should use a more detailed process when doing this. Furthermore, it may be more beneficial to use a different sentiment analysis tool altogether, or take inspiration from Huang et al. (2021) and Pant et al. (2020) and use machine learning to create a novel sentiment analysis method specific to the field in which it is being deployed.

6.4 Word Embeddings

SpaCy proved to be a very useful tool in terms of collecting word embedding data, although the way in which these word embeddings were collected and used can be debated. The idea behind creating a model that used average vector values was that underlying concepts or ideas that correlated with price movements would have been picked up on by the machine learning models; however, separating the vectors' dimensions in this way, while improving computational performance, did not work as intended. More effective uses of word embeddings could have been achieved in this study by using clustering algorithms to find underlying concepts or ideas instead of splitting vectors up. Additionally, the concept of an "average word vector" over an entire sentence arguably dilutes the vectors themselves with outlying data.

6.5 Trading Software

Since the trading software was designed to show a proof of concept, the fact that a system has been implemented that uses Twitter data to make automated trade decisions is effective. However, the fact that not a single trade decision has been made since 10th December 2021

is alarming. As mentioned, a more effective trade strategy is needed so that the portfolio can be adapted during bull and bear markets.

Moreover, more advanced features could be added to the interface such as a truly live Twitter feed instead of a semi-live one, as well as a cleaner design. The interface could also be uploaded to the internet so that backend Python code can be run to generate those features not available in JavaScript, instead of having to run the interface using a Python workaround.

6.6 Conclusion

Looking back, there are many aspects of this study that I would like to change, given the limitations that have been described in this section as well as knowledge I have acquired since running the machine learning models. Creating this software proved to be a tremendously difficult challenge, given the amount of time it took to learn unfamiliar techniques and technologies. That being said, the software I have designed works as a successful proof of concept that algorithmic trading software can be developed to trade cryptocurrency based on Twitter statistics.

Appendix

The GitLab repository for this project is available to view at https://git-teaching.cs.bham.ac.uk/mod-msc-proj-2020/lam045.

Requirements

- Python 3.9.1, with the following libraries:
  - Tweepy (`pip install tweepy`)
  - YAML (`pip install pyyaml`)
  - Binance-Python (`pip install python-binance`)
  - NLTK (`pip install nltk`)
  - SpaCy (`pip install spacy`), requires follow up vocab download that can be run using `python -m spacy download en`$_c ore_w eb_l g$.
- Java 11.0.9.1
  - WEKA 3.9.0: `weka.jar` has been included, so files should run on any machine with Java installed once files are unzipped. Run `java -jar weka.jar` to unzip. If this is not sufficient, please view https://waikato.github.io/weka-wiki/downloading_weka/ for troubleshooting guide to installing WEKA.
- Jupyter Notebooks 6.4.6
- UNIX shell command compatability: Windows users may need to install PowerShell for these commands to work.

Runnable files

IMPORTANT NOTICE: *trading_bot.py* is included for viewing purposes only. Do **not** run this code as it will enact a cryptocurrency trade.

tweet_collector_demo.py

This file is a demonstration version of the file orginally used to collect tweets and store them in a PostgreSQL database. The functionality has been changed somewhat for demonstration purposes. Where applicable, old code has been commented out in the file so it is available

to view in its originally format. Running this file prints a collection of tweets posted from the previous day about Bitcoin, Ethereum, Dogecoin, and Cardano. The only instruction is that it needs to be run where console outputs can be viewed.

trading_bot_demo.ipynb

This file is Jupyter Notebook version of *trading_bot.py* that displays this shows the functionality of *trading_bot.py* and how trades are made using the automated script. Run each cell sequentially.

tweet_cleaner.py

This is a demo version of how tweets are cleaned. A small selection of tweets are pre-loaded instead of being loaded from a database. The program prints the clean version of the tweets, as well as the polarity scores and vector data.

interface.py

Running this file will load the interface in the default browser. Takes some time to load so progress updates are provided in the terminal.

Other files

- `.apikeys.yaml`: API keys stored in a hidden YAML configuration file
- `weka.jar`: java package for WEKA functionality, requires unzipping via `java -jar weka.jar`
- `assets/interface.html`: HTML code for interface, used by `interface.py`
- `assets/stylesheet.css`: CSS stylesheet for interface, used by `interface.py`
- `assets/interface_script.js` JavaScript for interface, used/edited by `interface.py`
- `portfolio/default_portfolio.yaml` a YAML configuration file used to store and append data concerning previous portfolio holdings, used/edited by `trading_bot.py` and `interface.py`
- `portfolio/portfolio.yaml`: a YAML configuration file used to store the current portfolio holdings, used/edited by `trading_bot.py`, `trading_bot_demo.ipynb` and `interface.py`
- `portfolio/portfolio_DEMO.yaml`: a demo version of `portfolio/portfolio.yaml` for use in `trading_bot_demo.py`
- `portfolio/portfolio_history_DEMO.yaml`: a demo version of `portfolio_history.yaml` for use in `trading_bot_demo.ipynb`
- `models/...`: daily model files used by WEKA to make predictions

References

Abraham, J., D. Higdon, J. Nelson, and J. Ibarra (2018). "Cryptocurrency price prediction using tweet volumes and sentiment analysis". In: *SMU Data Science Review* 1.3, p. 1.

Akram, S. V., P. K. Malik, R. Singh, G. Anita, and S. Tanwar (2020). "Adoption of blockchain technology in various realms: Opportunities and challenges". In: *Security and Privacy* 3.5, e109.

Antonopoulos, A. M. and G. Wood (2018). *Mastering ethereum: building smart contracts and dapps*. O'reilly Media.

Azevedo Sousa, J. E. de, V. Oliveira, J. Valadares, G. Dias Goncalves, S. Moraes Villela, H. Soares Bernardino, and A. Borges Vieira (2021). "An analysis of the fees and pending time correlation in Ethereum". In: *International Journal of Network Management* 31.3, e2113.

Berrar, D. (2019). "Cross-Validation". In: *Encyclopedia of Bioinformatics and Computational Biology*. Ed. by S. Ranganathan, M. Gribskov, K. Nakai, and C. Schönbach. Oxford: Academic Press, pp. 542–545.

BitInfoCharts (2022). *Bitcoin, Litecoin, Namecoin, Dogecoin, Peercoin, Ethereum stats*. URL: https://bitinfocharts.com/ (visited on 01/13/2022).

Boigelot, D. (2011). *Correlation Examples*. URL: https://commons.wikimedia.org/wiki/File:Correlation_examples2.svg (visited on 01/13/2022).

Bullmann, D., J. Klemm, and A. Pinna (2019). "In search for stability in crypto-assets: are stablecoins the solution?" In: *ECB Occasional Paper* 230.

Cardano (2021). *Cardano Docs*. URL: https://docs.cardano.org.

Chaim, P. and M. P. Laurini (2019). "Is Bitcoin a bubble?" In: *Physica A: Statistical Mechanics and its Applications* 517, pp. 222–232.

Chen, Y., J. Yuan, Q. You, and J. Luo (2018). "Twitter sentiment analysis via bi-sense emoji embedding and attention-based LSTM". In: *Proceedings of the 26th ACM international conference on Multimedia*, pp. 117–125.

Chepurnoy, A., V. Kharin, and D. Meshkov (2018). "A systematic approach to cryptocurrency fees". In: *International Conference on Financial Cryptography and Data Security*. Springer, pp. 19–30.

Chiroma, H., S. Abdulkareem, and T. Herawan (2015). "Evolutionary Neural Network model for West Texas Intermediate crude oil price prediction". In: *Applied Energy* 142, pp. 266–273.

Chohan, U. W. (2019). "Are stable coins stable?" In: *Notes on the 21st Century (CBRi)*.

— (2021). "A history of Dogecoin". In: *Discussion Series: Notes on the 21st Century*.

Chokor, A. and E. Alfieri (2021). "Long and short-term Impacts of Regulation in the Cryptocurrency Market". In: *The Quarterly Review of Economics and Finance*.

Chung, Y.-A. and J. Glass (2018). "Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech". In: *arXiv preprint arXiv:1803.08976*.

CoinMarketCap (2022). *Cryptocurrency Prices, Charts And Market Capitalizations*. URL: https://coinmarketcap.com (visited on 01/13/2022).

Colianni, S., S. Rosales, and M. Signorotti (2015). "Algorithmic trading of cryptocurrency based on Twitter sentiment analysis". In: *CS229 Project*, pp. 1–5.

Desmond, D. B., D. Lacey, and P. Salmon (2019). "Evaluating cryptocurrency laundering as a complex socio-technical system: A systematic literature review". In: *Journal of Money Laundering Control*.

Dogecoin Core (2021). *Dogecoin Core*. Version 1.14.5. URL: https://github.com/dogecoin/dogecoin.

Ethereum (2022). *Ethereum Developer Documentation*. URL: https://ethereum.org/en/developers/docs/ (visited on 01/13/2022).

Fama, E. F. (1970). "Efficient Capital Markets: A Review of Theory and Empirical Work". In: *The Journal of Finance* 25.2, pp. 383–417. ISSN: 00221082, 15406261. URL: https://www.jstor.org/stable/2325486.

Frank, E., M. A. Hall, and I. H. Witten (2016). *The WEKA Workbench. Online Appendix for" Data Mining: Practical Machine Learning Tools and Techniques*.

Geiregat, S. (2018). "Cryptocurrencies are (smart) contracts". In: *Computer law & security review* 34.5, pp. 1144–1149.

Ghojogh, B. and M. Crowley (2019). "The theory behind overfitting, cross validation, regularization, bagging, and boosting: tutorial". In: *arXiv preprint arXiv:1905.12787*.

Gorjón, S. (2021). "The role of cryptoassets as legal tender: the example of El Salvador". In: *Banco de Espana Article* 35, p. 21.

Hasso, T., D. Müller, M. Pelster, and S. Warkulat (2021). "Who participated in the GameStop frenzy? Evidence from brokerage accounts". In: *Finance Research Letters*, p. 102140.

Hawkins, D. M. (2004). "The problem of overfitting". In: *Journal of chemical information and computer sciences* 44.1, pp. 1–12.

Hendrickson, J. R. and W. J. Luther (2017). "Banning bitcoin". In: *Journal of Economic Behavior & Organization* 141, pp. 188–195.

Holmes, G., A. Donkin, and I. H. Witten (1994). "Weka: A machine learning workbench". In: *Proceedings of ANZIIS'94-Australian New Zealnd Intelligent Information Systems Conference*. IEEE, pp. 357–361.

Hoskinson, C. (2017). *Why Cardano*. URL: https://why.cardano.com (visited on 01/13/2022).

Houben, R. and A. Snyers (2018). *Cryptocurrencies and blockchain: Legal context and implications for financial crime, money laundering and tax evasion*.

Howson, P. and A. de Vries (2022). "Preying on the poor? Opportunities and challenges for tackling the social and environmental threats of cryptocurrencies for vulnerable and low-income communities". In: *Energy Research & Social Science* 84, p. 102394.

Huang, X., W. Zhang, Y. Huang, X. Tang, M. Zhang, J. Surbiryala, V. Iosifidis, Z. Liu, and J. Zhang (2021). "LSTM Based Sentiment Analysis for Cryptocurrency Prediction". In: *arXiv preprint arXiv:2103.14804*.

Inci, A. C. and R. Lagasse (2019). "Cryptocurrencies: applications and investment opportunities". In: *Journal of Capital Markets Studies*.

Indurkhya, N. and F. J. Damerau (2010). *Handbook of natural language processing*. Vol. 2. CRC Press.

Jain, A., S. Tripathi, H. D. Dwivedi, and P. Saxena (2018). "Forecasting price of cryptocurrencies using tweets sentiment analysis". In: *2018 eleventh international conference on contemporary computing (IC3)*. IEEE, pp. 1–7.

Jiao, P., A. Veiga, and A. Walther (2020). "Social media, news media and the stock market". In: *Journal of Economic Behavior & Organization* 176, pp. 63–90.

Judmayer, A., N. Stifter, K. Krombholz, and E. Weippl (2017). "Blocks and chains: introduction to bitcoin, cryptocurrencies, and their consensus mechanisms". In: *Synthesis Lectures on Information Security, Privacy, & Trust* 9.1, pp. 1–123.

Kiayias, A., A. Russell, B. David, and R. Oliynykov (2017). "Ouroboros: A provably secure proof-of-stake blockchain protocol". In: *Annual International Cryptology Conference*. Springer, pp. 357–388.

Kiffer, L., D. Levin, and A. Mislove (2017). "Stick a fork in it: Analyzing the Ethereum network partition". In: *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, pp. 94–100.

Kiviat, T. I. (2015). "Beyond bitcoin: Issues in regulating blockchain tranactions". In: *Duke LJ* 65, p. 569.

Knox, S. W. (2018). *Machine learning: a concise introduction*. John Wiley & Sons.

Kraaijeveld, O. and J. De Smedt (2020). "The predictive power of public Twitter sentiment for forecasting cryptocurrency prices". In: *Journal of International Financial Markets, Institutions and Money* 65, p. 101188.

Li, T., D. Shin, and B. Wang (2021). "Cryptocurrency pump-and-dump schemes". In: *Available at SSRN 3267041*.

Liu, B. (2020). *Sentiment analysis: Mining opinions, sentiments, and emotions*. Cambridge university press.

Locke, T. (2021a). *Elon Musk says dogecoin is better to buy things with than bitcoin*. URL: https://www.cnbc.com/2021/12/13/elon-musk-says-dogecoin-is-better-to-buy-things-with-than-bitcoin.html (visited on 01/13/2022).

— (2021b). *Mark Cuban says dogecoin is the 'strongest' cryptocurrency as a medium of exchange*. URL: https://www.cnbc.com/2021/08/13/mark-cuban-dogecoin-is-the-strongest-crypto-as-a-medium-of-exchange.html (visited on 01/13/2022).

López-Martın, C., S. Benito Muela, and R. Arguedas (2021). "Efficiency in cryptocurrency markets: new evidence". In: *Eurasian Economic Review* 11.3, pp. 403–431.

Lubitz, M. (2017). "Who drives the market? Sentiment analysis of financial news posted on Reddit and Financial Times". In: *University of Freiburg: http://ad-publications. informatik. uni-freiburg. de/theses/Bachelor_ Michael_ Lubitz_ 2018. pdf*.

Mao, Y., W. Wei, B. Wang, and B. Liu (2012). "Correlating S&P 500 stocks with Twitter data". In: *Proceedings of the first ACM international workshop on hot topics on interdisciplinary social networks research*, pp. 69–72.

Mezquita, Y., A. B. Gil-González, J. Prieto, and J. M. Corchado (2021). "Cryptocurrencies and Price Prediction: A Survey". In: *International Congress on Blockchain and Applications*. Springer, pp. 339–346.

Mora, C., R. L. Rollins, K. Taladay, M. B. Kantar, M. K. Chock, M. Shimada, and E. C. Franklin (2018). "Bitcoin emissions alone could push global warming above 2 C". In: *Nature Climate Change* 8.11, pp. 931–933.

Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. URL: https://bitcoin.org/bitcoin.pdf (visited on 01/13/2022).

Narayanan, A., J. Bonneau, E. Felten, A. Miller, and S. Goldfeder (2016). *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press.

Nguyen, T. H., K. Shirai, and J. Velcin (2015). "Sentiment analysis on social media for stock movement prediction". In: *Expert Systems with Applications* 42.24, pp. 9603–9611.

Nowé, A., P. Vrancx, and Y.-M. De Hauwere (2012). "Game theory and multi-agent reinforcement learning". In: *Reinforcement Learning*. Springer, pp. 441–470.

Pan, X., Y. You, Z. Wang, and C. Lu (2017). "Virtual to real reinforcement learning for autonomous driving". In: *arXiv preprint arXiv:1704.03952*.

Pano, T. and R. Kashef (2020). "A Complete VADER-Based Sentiment Analysis of Bitcoin (BTC) Tweets during the Era of COVID-19". In: *Big Data and Cognitive Computing* 4.4, p. 33.

Pant, D. R., P. Neupane, A. Poudel, A. K. Pokhrel, and B. K. Lama (2018). "Recurrent neural network based bitcoin price prediction by twitter sentiment analysis". In: *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*. IEEE, pp. 128–132.

Park, D., Y. Zhang, and G. Rosu (2020). "End-to-end formal verification of ethereum 2.0 deposit smart contract". In: *International Conference on Computer Aided Verification*. Springer, pp. 151–164.

Ranjit, S., S. Shrestha, S. Subedi, and S. Shakya (2018). "Comparison of algorithms in foreign exchange rate prediction". In: *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*. IEEE, pp. 9–13.

Rejeb, A., K. Rejeb, and J. G. Keogh (2021). "Cryptocurrencies in Modern Finance: A Literature Review". In: *Etikonomi* 20.1, pp. 93–118.

Rodgriguez, M. Z., C. H. Comin, D. Casanova, O. M. Bruno, D. R. Amancio, L. D. F. Costa, and F. A. Rodrigues (2019). "Clustering algorithms: a comparative approach". In: *PloS one, 14(1)*, e0210236.

Smuts, N. (2019). "What drives cryptocurrency prices? An investigation of google trends and telegram sentiment". In: *ACM SIGMETRICS Performance Evaluation Review* 46.3, pp. 131–134.

Somanathan, A. R. and S. K. Rama (2020). "A Bibliometric Review of Stock Market Prediction: Perspective of Emerging Markets". In: *Applied Computer Systems* 25.2, pp. 77–86.

Srinivasa-Desikan, B. (2018). *Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras*. Packt Publishing Ltd.

Stoll, C., L. Klaaßen, and U. Gallersdörfer (2019). "The carbon footprint of bitcoin". In: *Joule* 3.7, pp. 1647–1661.

Sutton, R. S. and A. G. Barto (2018). *Reinforcement Learning: An introduction*. MIT press.

Tandon, C., S. Revankar, and S. S. Parihar (2021). "How can we predict the impact of the social media messages on the value of cryptocurrency? Insights from big data analytics". In: *International Journal of Information Management Data Insights* 1.2, p. 100035.

Thornton, C., F. Hutter, H. H. Hoos, and K. Leyton-Brown (2013). "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms". In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 847–855.

Umar, Z., M. Gubareva, I. Yousaf, and S. Ali (2021). "A tale of company fundamentals vs sentiment driven pricing: The case of GameStop". In: *Journal of Behavioral and Experimental Finance* 30, p. 100501.

Urquhart, A. (2016). "The inefficiency of Bitcoin". In: *Economics Letters* 148, pp. 80–82.

Van der Maaten, L. and G. Hinton (2008). "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11.

Weng, B., W. Martinez, Y.-T. Tsai, C. Li, L. Lu, J. R. Barth, and F. M. Megahed (2018). "Macroeconomic indicators alone can predict the monthly closing price of major US indices: Insights from artificial intelligence, time-series analysis and hybrid models". In: *Applied Soft Computing* 71, pp. 685–697.

Wooley, S., A. Edmonds, A. Bagavathi, and S. Krishnan (2019). "Extracting Cryptocurrency Price Movements from the Reddit Network Sentiment". In: *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE, pp. 500–505.

Xu, J. and B. Livshits (2019). "The anatomy of a cryptocurrency pump-and-dump scheme". In: *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pp. 1609–1625.

Yahoo Finance (2022). *Yahoo Finance - Stock Market Live, Quotes, Business & Finance News*. URL: https://finance.yahoo.com (visited on 01/13/2022).

Ye, C., G. Li, H. Cai, Y. Gu, and A. Fukuda (2018). "Analysis of security in blockchain: Case study in 51%-attack detecting". In: *2018 5th International Conference on Dependable Systems and Their Applications (DSA)*. IEEE, pp. 15–24.

Yujian, L. and L. Bo (2007). "A normalized Levenshtein distance metric". In: *IEEE transactions on pattern analysis and machine intelligence* 29.6, pp. 1091–1095.

Zhang, S. and J.-H. Lee (2020). "Analysis of the main consensus protocols of blockchain". In: *ICT express* 6.2, pp. 93–97.