A DEEP LEARNING APPROACH TO CRYPTOCURRENCY PREDICTION

CWD-12

by

JAKE ANDERSON

A dissertation project accepted and approved in partial fulfillment of the requirements for the degree of

Computer Science and Artificial Intelligence

in Department of Computer Science

Supervised by,

Dr Christian W Dawson

Loughborough University

Summer 2024

DISSERTATION ABSTRACT

Jake Anderson

B.Sc in Computer Science and Artificial Intelligence

Title: A deep learning approach to cryptocurrency prediction
CWD-12

On the macro level, the project looks to provide insight into the viability of contemporary machine learning-backed methods for cryptocurrency, while on a micro level provides market insight for investors. A long short-term model (LSTM) uses historical time series data of opening and closing prices of coins such as BTC or ETH in order to estimate closing prices. This model is measured against more traditional models such as linear regressions in order to compare results. Another approach is deployed where sentiment analysis with NLP methods is used to get investor sentiment indicating if a trader should buy or sell a coin.
The source can be found here: https://github.com/Kodoh/CWD12_F132339

ACKNOWLEDGMENTS

It goes without saying Chris Dawson's input has been incredibly valuable and the success of this project has partially been a result of his feedback. I want to personally thank Chris for helping me achieve a lifelong dream of working on a long-term machine learning academic project.
I would also like to extend my thanks to Lawrence M Stewart for open-sourcing a large portion of the sentiment data obtained (*ProfitsBot_ V0_ OLLM_*). Recent updates in the privatisation of media data looked to complicate the project however Lawrence and collaborators have allowed academic projects such as this to continue their research without the need of large funding.

For my family

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Introduction

As of 2023, it has been suggested that over 420 million people own a form of cryptocurrency, which represents roughly 5% of the world population (Triple-A 2023). Of these owners, it is noted that over 40% are inexperienced and happy to make uninformed decisions when it comes to their investments (Vega 2021). Taking this into account in addition to a rising lack of trust in cryptocurrency investing, such as the case with the recent collapse of FTX (Tidy and Geoghegan 2023), it poses the question: 'is cryptocurrency prediction a viable investment strategy?'.
The concept of cryptocurrency prediction is not a new one, in fact, it's expected to grow even further with a reported 46% of hedge funds looking to expand on their cryptocurrency portfolios going into Q423 (Cohen 2023). Successful prediction models can yield great wealth for investors, leading to a series of attempts (some of which are discussed later), the results of which heavily impact the welfare of day traders and the returns that funds can boast to their clients, a key driver for the following research. Another reason for this research is the cost cutting benefit for private funds looking to reduce high payouts for quantitative analysts and traders (Yates 2022) or enable reallocation of manpower and resources given that more of their trading operations can be automated.
Due to the chaotic nature of cryptocurrency markets, this report also looks towards less traditional approaches such as analysing sentiment of investors in order to make better judgments by incorporating NLP and deep learning methods. There are notable attempts in this domain yet they mostly rely on publicly accessible APIs for social media sites in order to gather data. 2023 has seen the rise of private access API keys (Collins 2023), resulting in a halt in sentiment-based research. This report aims to provide a free of charge sentiment based solution accessible for academics and traders alike, aiming to expand on previous works post API privatisation.
The results of this report aim to shed light on the potential viability of recent models as well as delve into sparsely documented areas such as post-privatised API sentiment analysis. By contributing to this area, it is hoped that the research will promote the usage of machine learning approaches among investors.

1.2 Motivation

The main motivation behind this research is to discover where contemporary methods can be applied in an increasingly volatile market (Tambe 2023) as well as highlight some key blockers experienced while aiming to achieve this. While acknowledging the academic implications of the discussed project, I personally have a great deal of interest in finance and am especially curious about how accurate cryptocurrency predictions can evolve investment as we know it. I hope for the results of this report to form the basis for future research while contributing to econometric and machine learning conversations.

1.3 Aims

Goals

This report aims to give insight into the technology currently accessible to academia and prove its viability in contemporary cryptocurrency investing. Readers will be able to analyse a range of graphs and results highlighting the strengths of different models and approaches on a range of popular coins. It is hoped that these results can be used to promote further research as well as expand on projects aiming to automate cryptocurrency portfolio management. By building two separate models, there is also the opportunity to test the feasibility of sentiment-based predictions (post API restrictions) and deep learning approaches.
The produced results look to be reproducible by academics and traders alike by optimising model implementation (with the likes of parallelisation and Bayesian optimisation). With rapid discoveries in the field, the reports findings and implementation should be understandable and salable to allow for future technology to be used within the scope of the following report.
Aside from the benefits a reader may find in these results, it is also hoped within this process I can enhance my knowledge in deep learning, sentiment and financial data.

Project Objectives

Objective	**Description**	**Deadline**
Analyse Problem	Review problem and plan prediction approach.	W2-3, S1
Conduct Literature Review	Research cryptocurrency prediction, learn from past attempts.	W9, S1
Market Data Retrieval	Gather daily prices for BTC, ETH, and XRP using free API.	W7, S1
Sentiment Data Retrieval	Collect news articles, cleanse, and explore social media data.	W8, S1
Develop DL Model	Build accurate market data-based model for closing price prediction.	W11, S1
Develop Sentiment Model	Build binary classifier using sentiment data. Display results.	W3, S2
Evaluate Models	Analyze results, discuss advantages, disadvantages, and improvements.	W6, S2

Table 1.1. Project Objectives. Key: $\{W_i : Week, S_j : Semester\}$.

Result Objectives

Objectives related to the performance of the models.

Objective	**Rational**	**Desired Result**
Market Accuracy	Market data model should hold its own against contemporary attempts.	$MSE \leq 0.75$[*]
Sentiment Accuracy	More correct than incorrect classifications, otherwise human intuition may be more efficient.	$TP + TN > FP + FN$
Optimisation	If the models are not efficient then they become less reproducible (goal).	$R_{op} \leq R_{default}$
Profitability	Model should be practical.	$I \leq P$

Table 1.2. Key: $\{MSE$: Mean Squared Error, I: £ Invested, P: £ Post Investment, R: Run time, TP, TN, FP, FN: Classification types (Google 2018)$\}$

[*]MSE comes from a recent popular student attempt at the same task (Yu 2023)

1.4 Gantt Chart

During the early stages of the project, a Gantt chart was drawn in order to plan the development process. However, this is subject to change (notable modifications will be stated later).

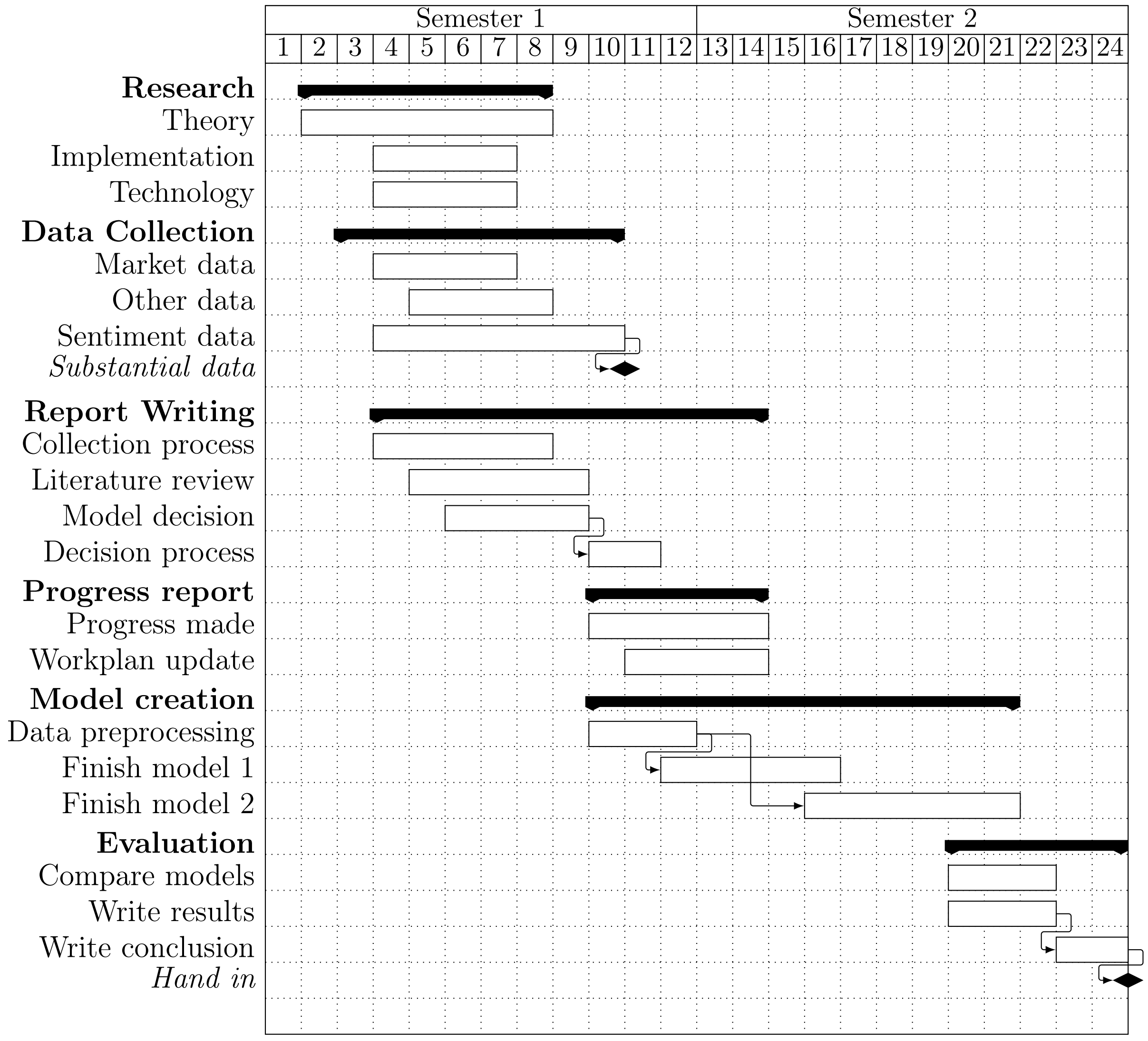


Figure 1.1. Workflow Gantt Chart

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Financial forecasting has long been an interest of academic as well as private research and continues to be due to its complexity and high stakes. It should be noted that the literature discussed may not contain the most contemporary solutions to said dilemma as a result of many firms keeping winning models a secret, however, the results of the research give a foundational insight into the problems that plague cryptocurrency predictions and how academia has attempted to solve them. The reviewed content can be roughly split up into three distinct parts; Traditional approaches to forecasting, Deep learning attempts, and Sentiment. This is then summarised within a conclusion which helps form the basis for this report and how the solution is eventually formed.

2.2 Traditional Approaches To Forecasting

Traditionally, the process of financial prediction can be split into two broad methodologies: Fundamental Analysis and Technical Analysis (Majaski 2023), however with the rise of AI usage in the private sector (Haan 2023) a new range of machine learning attempts have also been adopted. Fundamental Analysis involves assessing a firm's equity value based on published financial statements. Graham and Dodd (1934) expand this further by emphasising the difference between intrinsic value being the true worth of a security based on its fundamentals and market price which is simply the prevailing market value easily swayed by supply and demand. The likes of Seng and Hancock (2012) have shown that fundamental Analysis can be paired with a form of context appreciation to see promising long and short-term gains. However, like many other Fundamental attempts, the paper results with relatively low R^2 (regression) values, implying that an increased number of factors must be considered for more robust portfolio management.
Technical Analysis is focused on the idea of finding a trend early on and then holding that position until evidence suggests that the trend has ended (Edwards and Magee 1992). In terms of the profitability of the approach, Park and Irwin (2007) show that out of 95 studies employing technical strategies, 56 had positive results indicating, the advantages of its use. However, it should be noted such an approach

can be easily influenced by bias due to its human involvement implying a need for more technological approaches (typically machine learning-inspired).

There is also a range of quantitative approaches that adopt some general discrete-time market model (Elliott and Kopp 2005) which gives way to many statistical attempts. Tay and Shen (2002) point out a range of statistical attempts that rely on the concept of rough sets (Pawlak 1982) such as Ruggiero (1994) who was able to predict movements in the S&P 500 with 70% accuracy over a 5 week period. Similarly, shraf, Félix, and Serrasqueiro (2019) use a 3 probit model which saw 73% accuracy in predicting distress in emerging markets, however, it should be noted this is only a short-term model as results showed diminishing accuracy further in the future it predicted. While impressive it's important to consider many attempts of this nature struggle when faced with times of financial crisis and as of recently more firms have opted towards machine learning (like the approach I will highlight within this paper).

2.3 Deep Learning Attempts

With the change of financial regulations on the supply side and technological evolution in recent years, consumers have taken a shift to more automated systems (Board 2017) suggesting the use of solely traditional methods is insufficient for modern-day forecasting. Gamboa (2017) discusses the potential of deep learning within time series data and stresses that previous attempts depended on hand-crafted features and or required professional assistance of some sort. The paper highlights early attempts such as the modelling of flour prices over an eight-year period (Chandra and M. Zhang 2012) as well as the promise of contemporary attempts such as Schulz and Matthies (2014) who take a deep learning approach to model the amount of littering in the North Sea. Such attempts highlight that there is great potential to leverage these techniques within time series-related problems such as financial forecasting.
Heaton, Polson, and Witte (2016) argue that deep learning in finance can be used in a range of contexts while seeing greater results than statistical and econometrics approaches such as shallow factor modeling. It should be noted while showing great potential the paper accepts that such models tend to have a single use and building a fully automated portfolio manager is not fully achievable currently (why this report opts for two separate forecasting models). The group concludes its findings by

warning economists that we are likely to see a world where foundational economic logic cannot compete with deep learning models in the near future.

Academia as well as the private sector have shown great interest in using this potential in deep learning for a variety of stock and cryptocurrency prediction models. Singh and Srivastava (2017) use 2 Directional 2 Dimensional Principal Component Analysis ($2D^2PCA$) with a Radial Basis function neural network (RBFNN) in order to make short-term predictions on the closing price of stocks on the NASDAQ. Their results saw a 4.8% increase in accuracy compared to $2D^2PCA$ using a deep neural network and a 15.6% increase on an RNN model. The paper uses long time frames to make short-term predictions to acknowledge any volatility. It is important to consider the differences between stock prediction and cryptocurrency prediction (as is the case for this report) in this case as the features and target of the model may deviate. In light of this, it was important to consider cryptocurrency attempts such as D'Amato, Levantesi, and Piscopo (2022) who used a Jordan Neural Network (Jordan 1986) in order to predict daily volatility in popular cryptocurrencies. The group was able to achieve a 0.000527 MSE on bitcoin over a several-year period while smaller coins such as XRP had a 0.086333 MSE implying newer coins may be harder to predict due to their volatile nature. The paper concludes by stating their interest in LSTM approaches and acknowledging their power. LSTM (Hochreiter and Schmidhuber 1997) was the chosen model for this report due to its impressive results within cryptocurrency forecasting. Long Short-Term Memory was utilised by Chen, Zhou, and Dai (2015) in their stock prediction on the Chinese stock market seeing a 12.9% increase in accuracy compared to the random prediction method.

2.4 Sentiment

Chen, Zhou, and Dai (2015) conclude their findings by pointing to financial news and social media as key drivers for market volatility, suggesting sentiment analysis is an important consideration in cryptocurrency forecasting. L. Zhang, Wang, and Liu (2018) discussed a multitude of applications of deep learning-based sentiment analysis highlighting the applicability of the process within the context of financial predictions. The paper points to several sentence-level sentiment classification successes such as Guggilla, Miller, and Gurevych (2016) who present an LSTM and CNN-based deep learning solution to the classification of sentences being factual or feeling. The paper concludes by emphasising the promising future of NLP and sentiment analysis while acknowledging complexities such as sarcasm analysis and resource-poor language sentiment analysis.

Mishev et al. (2020) look into the applications of sentiment analysis within finance. Their review points to use cases within stock forecasting, securities trading as well as portfolio management and even goes as far as to say NLP-based sentiment analysis is comparable if not better than an expert. The paper does however warn readers that finance tends to have a lot of large unlabelled data sets and domain-specific language leading to attempts having increased miss classifications. Regarding prediction, there have been several attempts with varying success. Mittal and Goel (2012) continue the research done by Bollen, Mao, and Zeng (2011) to achieve 75.56% accuracy on stock predictions using a self-organisational fuzzy neural network analysing Twitter sentiment. Their research also found SVM (59% accuracy) and logistic regression (60%) are unsuitable models in such a domain with both models having a lower classification rate than linear regression (71%). The paper can be criticised in the fact that Twitter stock sentiment is only a small subset of the greater public sentiment, it is limited to those who own Twitter, those who actively post about investments and users who speak English indicating there is much more at play.
Mohan et al. (2019) were able to predict the closing prices of companies on the S&P500 using sentiment analysis on news articles related to the company. The group shows RNN (specifically LSTM) trained models to be best for using sentiment in time series data. RNNs saw MAPE (Mean absolute percentage error) values between 2.03 and 2.17 outdoing traditional approaches such as Autoregressive Integrated Moving Average (ARIMA) models. An interesting observation from the study is that larger companies such as Apple (AAPL) and Microsoft (MSFT) had lower MAPE values overall compared to American Airlines (AAL), suggesting news sentiment can be harder to make predictions with on smaller companies (due to sparsity in data).

Lamon, Nielsen, and Redondo (2017) use a combination of news and social media sentiment gathering $\sim$ 30,000 tweets and roughly 3600 news articles to make binary classifications distinguishing whether a given cryptocurrency coin will drop or increase in value. Their model boasted an impressive 75.8% accuracy using a Bernoulli Naive Bayes classifier. It is important to acknowledge the model's reliance on Twitter's API which has since been privatised, a potential problem this report addresses.

2.5 Conclusion

While research in the review remains promising, it emphasises the disparity in private funds research to that of academia as top funds have been averaging over 60% on returns (Gyol 2023) indicating researchers are playing catch-up with the newest models. This report aims to leverage the contemporary results discussed above to build models to bridge the gap between these two worlds.

CHAPTER 3

DESIGN

The following chapter is used to discuss and justify the technology, tooling and concepts incorporated within the project. There is also discussion for potential future modifications within the design of the project.

3.1 Data Sources

The project relies on processing large quantities of data in order to improve the accuracy of results. This data for the most part is handled in terms of data frames (provided by the Pandas library) providing simplified handling of large data sets, in addition to large multi-dimensional arrays from the scientific package NumPy. These libraries were selected due to their high levels of documentation and versatility compared to other popular packages such as Vaex or Dask.

Market Data

Yahoo Finance (https://developer.yahoo.com/oauth2/guide/) was the chosen API to gather daily market updates on BTC, ETH and XRP. This decision was made due to the free access to the API and daily accurate prices of several popular coins with the option for British sterling to be used as the counter currency. It was also discovered that in order to get a suitable amount of data it would be important to pick from popular and well-documented tokens, giving preference to the chosen coins. For any given coin (from its first day of trading to the day of the request) the API returns (in JSON format):

$$\{Date, Open, High, Low, Close, Volume, Dividends, Stock\ Splits\}$$

Date, Open and Close were decided to be the most important of these and thought to be the key drivers of volatility for almost all coins. However, a future aspiration would be to incorporate more features such as High and Low prices as well as Volume which can often prove to be useful in predicting longer-term trends or volatility cap.

Sentiment Data

Concerning sentiment, both social media posts and news article headlines were taken into consideration. This is because other sources for sentiment such as polls, video content and blogs are harder to come by and potentially have less impact on market volatility and are less quantifiable. The social media data is from a collection of Reddit posts selecting only popular finance and cryptocurrency-focused subreddits. As a result of recent changes in privatisation with Reddits API (Spez 2023), the data has been sourced from a public repository storing collections of posts for each subreddit pre-API-monetization (https://github.com/getorca/ProfitsBot_V0_OLLM). The news data was sourced from numerous popular news articles and provided by the Cryptopanic API (CryptoPanic 2024). The Guardians news API (Guardian 2024) was also utilised due to its large request rate, while also coming free of charge for student-related purposes.

Other Data

SEC (U.S. Securities and Exchange Commission) report data was also collected which can provide market sentiment as well as regulatory information. This can be incredibly useful, especially in the case of FTX or with smaller coins facing criminal charges leading to increased volatility. However, due to a lack of publicly available reports, it was concluded that the volume of data would not be substantial enough to form the basis of a model.
CoinGecko (CoinGecko 2024) is another popular API service which was tested to see if getting data such as 'open interest' a measure on popular exchanges to see how much a coin is in demand and 'known holdings' (how much a company holds of a certain coin) could add to sentiment data. Due to limited free API requests and poorly documented historical data, it seemed both of these options had become unviable however remain an ambition for future research.

3.2 Closing Price Model

The closing price model was written solely in Python due to its selection of libraries to support machine learning programs. The libraries employed included TensorFlow (with Keras) to provide an interface for the models, Scikit-learn for model evaluation, and finally Matplotlib for visualising the resulting data.

Model Selection

Due to the nature of time series data, it was decided to implement an RNN (Recurrent Neural Network), which revolves around the idea of having loops in the network to consider previous time steps. RNNs has proven to work well when working with financial data (Mohan et al. 2019) as it has the ability to capture long-term Dependencies and is built for sequential data. In financial data, important events or market conditions might have occurred several time steps in the past, losing such data can cause the model to have less informed predictions. This can happen when using RNNs as a result of the 'vanishing gradient problem' where gradients exponentially vanish during backpropagation as time goes on. To combat such an issue LSTM (Long Short-Term Model) was selected (Hochreiter and Schmidhuber 1997), an RNN which utilises 'gates' to enable the network to retain information for longer. With previous attempts proving that markets can be increasingly chaotic (D'Amato, Levantesi, and Piscopo 2022) it was important to choose a model which captures longer trends hence the decision for a BiLSTM Bidirectional Long Short-Term Memory (al 2016). BiLSTM is simply a modified LSTM allowing for forward as well and backward processing, meaning any given coin can consider past and future trends, providing higher accuracy.

Hyperparameters

The model is passed a series of hyperparameters which heavily determine the quality of the prediction. These include:

$$\{layers, units, activation, length, epochs\}$$

Where layers refers to the number of layers used in the network, with units being the number of individual nodes within each layer. Length, in this context, is the number of previous closing prices to consider when making a prediction and was made a parameter due to its role in generalising the data. It is hoped fine-tuning length will help avoid over or underfitting the data. In order to optimise this process Bayesian optimization was used, a hyperparameter turner which uses a surrogate model to check for any given hyperparameter what is the level of 'uncertainty' and return the most optimal. This was chosen as it saves on computational processing power (consider a loop for each parameter) and it is more scalable, especially when considering more coins. GPyOpt, a Python open-source library for

Bayesian Optimization (https://sheffieldml.github.io/GPyOpt/) was used due to its compatibility with LSTMs and its comprehensive documentation.

Further optimisation

With this report looking at use cases for funds that potentially may have hundreds of coin predictions taking place it was chosen to use Pythons 'Pool' module (Python 2023) which allows for parallelisation of predictions on multiple cores. This also becomes scalable when considering the use of multiple GPUs over several devices for example.
Two dropout layers are incorporated into the model in order to prevent overfitting, this is because it has been seen in financial market prediction models which emphasise more recent prices tend to be more accurate (Singh and Srivastava 2017). This is likely due to shorter time frames being more predictable (as seen in the case with Ashraf, Félix, and Serrasqueiro 2019) with closing price trends proving to be highly nonlinear. These dropout layers mean that a random subset of the neurons within the training set will be ignored helping the model generalise the trends of price fluctuations.

Overview

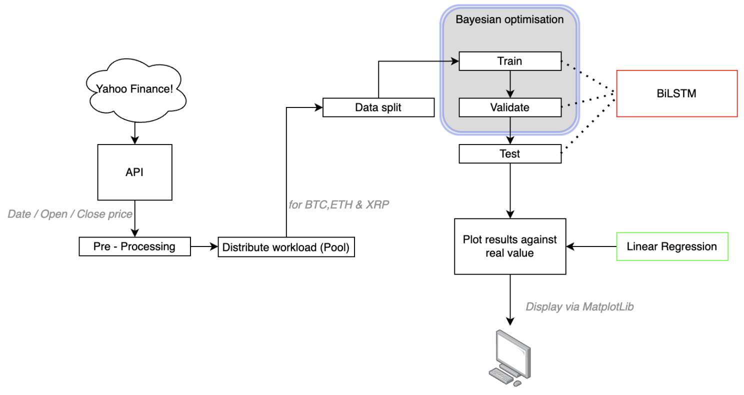


Figure 3.1. System Overview

3.3 Sentiment Model

Much like the model discussed in section 3.2, Python is predominantly used, its rich NLP documentation and plethora of libraries make it better suited for the project, when compared to alternatives such as Javas Apache OpenNLP (Apache 2024).

Preprocessing

The ease of handling large quantities of data provided by data frames (Pandas) made it an obvious choice to work with the data as opposed to manually accessing the raw data.

Data Cleansing

Reddit posts by nature tend to be informal, unstructured and for the most part unsuitable for models to consume directly, while typically more formal news articles come from diverse sources which can result in inconsistent formatting. For these reasons, a range of sanitising techniques were required. NLTK (Natural Language Tool Kit) was used to automate a large portion of this process as a result of its simplicity and compatibility with other used Python libraries such as NumPy or Pandas.

Tokenisation and Stop Words

Tokenisation allows the model to begin to treat the titles and articles as vectors in which the model can handle, hence making it an important step in the data-handling process (further discussed in section 5.3). With the tokenised data it makes it possible for more sophisticated means of preprocessing such as lemmatisation and stop word removal.
Removing stop words (words bringing no value to the sentiment) was a priority given the lack of formality in Reddit titles. For instance, consider the following token stream t and the result of removing stop words t_{stop} (actual example of a post):
t = *[‘Just’, ‘used’, ‘my’, ‘new’, ‘shift’, ‘card’, ‘to’, ‘buy’, ‘sushi’, ‘felt’, ‘awesome’]*
$\rightarrow t_{stop}$ = *[‘used’, ‘new’, ‘shift’, ‘card’, ‘buy’, ‘sushi’, ‘felt’, ‘awesome’]*. This emphasises more emotionally charged words such as *‘awesome’* and *‘new’* helping better form judgment on the sentiment.

Stemming vs Lemmatisation

The final step of preprocessing involves converting tokens into their base form, this makes handling data a simpler process (eg. *car* $\leftarrow_{base}$ *cars, car's, cars'...*). The most popular way of doing this is using stemming or lemmatisation algorithms.

Stemming is the more basic of the two and simply looks to cut the ends of words in order to obtain their root form (Porter 1980). However, stemming can cause nonwords and may encounter issues when working with foreign languages (poor accuracy). Lemmatisation (Gillis 2023) looks to reduce words to their lemmas, meaning words of similar nature reduce to the same lemma (returning to their base dictionary form). With this in mind and the volume of diverse titles and articles, lemmatisation was the more suitable approach.

Vectorisation

Vectorisation refers to the process of taking the pre-processed token streams and turning them into meaningful numerical vectors which can in turn be passed as input to the model. For instance, consider again the token stream t, this could be reduced to [0.125,0.75845,0.23432,0.32423,0.121,0.9432] where the values represent the emotional intensity associated with each token. This is now in a format consumable by the model. There are several ways of performing such a task, popular algorithms include: TF-IDF (Sparck Jones 1972), GloVe (Pennington, Socher, and Manning 2014), USE (Cer et al. 2018) with a range of others such as Word2Vec (Mikolov et al. 2013), BERT (Devlin et al. 2018), and Bag of Words (BoW) (Harris 1954).
It was decided to incorporate as many of these algorithms as possible to use for performance comparison while understanding which works best in the context of sentiment. To do this without compromising runtime, TF-IDF, GloVe and USE were selected due to their diversity in approaches. Alternatives follow similar processes to the ones mentioned above making differences in the results marginal and not worth adding large overheads to the runtime for.
The general idea behind TF-ID is the values within the vector are typically based on an 'importance score'. Formally this is defined as (Jha 2021):

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

Where TF(Term Frequency) is defined as:

$$\text{TF}(t,d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

And IDF (Inverse Document Frequency) as:

$$\text{IDF}(t,D) = \log\left(\frac{\text{Total number of documents in } D}{\text{Number of documents containing term } t+1}\right)$$

In contrast, GloVe (and other word embedding approaches) stores a dictionary of word-value pairs to capture the semantic relation between words within the vector and obtain their overall value. Google's Universal Sentence Encoder (USE) is a sentiment-focused encoder which looks to create fixed-sized vectors for sentence (or short text) level content. Given the sentiment focus of the project with the limited size of titles and posts, USE looks to be a promising approach.

Model Selection

The desired output can have one of two values ($sell \in [0,1]$), reducing the set of possible models to those supporting binary classification. Similarly to the vectorisation process, a diverse range of models was chosen in order to capture their impact on results, namely KNN, Support Vector Machines (SVM) and Random Forest.

Random Forest (Breiman 2001) is an ensemble model which builds multiple decision trees (with different structures) and recursively combines these results based on a majority vote. Random forest is a simpler implementation, making it a perfect base case to compare other models to. Random forest, due to its ensemble nature, helps prevent overfitting which will help when testing.

K-Nearest-Neighbours (KNN) (Cover and Hart 1967), similar to Random Forest, looks to classify each point based of a majority vote from its k nearest (euclidean distance) neighbours (other features). KNN has been known to work well with noisy data (Menhour, Abidine, and Fergani 2018) making it a good fit for working with the unpredictability and scale of the Reddit posts.

Support Vector Machines (Cortes and Vapnik 1995) being the more complex of the three, look to find a 'hyperplane' which can split the data set into two classes (in this case a 1 or a 0). While also looking to maximise the margin between points

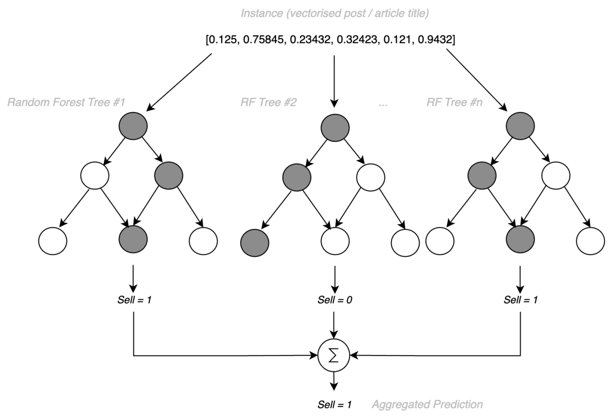


Figure 3.2. Random Forest example. Grey nodes refer to chosen path.

and the hyperplane (for better generalisation), formally we:

$$\begin{aligned} \underset{w}{\text{Minimize}} \quad & \tfrac{1}{2}\|w\|^2 \\ \text{subject to} \quad & y_i \left(\langle \boldsymbol{w}, \mathbf{x}_i \rangle + b\right) \geq 1 \end{aligned} \tag{3.1}$$

SVM has been known to work well with unstructured and limited data sources (Akkaya and Çolakoğlu 2019) which will complement both article and Reddit posts alike.

Initial Attempt

An initial approach was to predict targets based on single posts (given a single post or article a valid sell prediction is expected). This approach proved much simpler, however, relies heavily on our ability to generalise the training data. The unpredictability of forum posts puts into question how much sentiment can be extracted from a single post. As a result of the viability of the strategy and the discovered results (section 6.2), a new attempt was considered (aggregation).

Aggregation Attempt

A potential pitfall of the initial attempt was the lack of context for any given prediction. Consider the following example from the initial attempt, where t_i refers to some article or Reddit title and c_i refers to the relevant classifier.

t_1= *'BTC losing key investors'* $\rightarrow c_1 = 1$ (sell)

A more context (data leveraging) approach is to consider all posts related to a given day. This method means we shift focus from the post to the date (an enhanced classification leveraging more information), achieved by aggregating the results of individual features and reclassifying each based on the majority vote. Consider the following titles created on the same date:

$$t_1 = \textit{'BTC losing key investors'} \rightarrow c_1 = 1$$

$$t_2 = \textit{'Big opportunity for new Bitcoin investors'} \rightarrow c_2 = 0$$

$$t_3 = \textit{'BTC-GDP sits strong amidst new SEC regulation'} \rightarrow c_3 = 0$$

$$c^* = \arg\max_i(c_i) = 0$$

A further optimisation was then made to consider the likeliness (probability) of each classifier for each post ($0 \leq c_i \leq 1 \in \mathbb{R}$). From here the average for each day is calculated and reclassified based on if it is higher than some threshold γ. Formally:

$$c_i = \begin{cases} 1, & \text{if } ((\frac{1}{n}\sum_{i=0}^{n}(\{c_i, c_{i+1}, ..., c_n\} \in d_i)) > \gamma \\ 0 & \text{otherwise} \end{cases}$$

This logic can be extended to the case where multiple days are considered, namely we can take the average of all $c \in d_i, d_{i-1}, ..., d_{i-n}$ and check if it passes some threshold γ. This may be a useful tool when considering the impact of previous sentiment on future market movements, in addition to providing insights into whether media tends to be predictive or responsive.

Hyperparameters

Each of the models chosen has varying hyperparameters, all of which play a role in the overall accuracy, hence, it is of great interest to explore different variations and look to understand what is most suitable and will yield the best results. KNN has a single hyperparameter k, referring to the total number of neighbours to consider. It is thought that larger values of k can generalise the sentiment well, however, it may miss out on more specific parts of the title which could be necessary for sentiment classification.

Random forest has the hyperparameters *n_ estimators* and *max_ depth*, referring to the number of trees and depth of the trees respectively. A larger depth and n value imply greater accuracy when predicting whether to sell however will likely impact runtime performance (Probst, Wright, and Boulesteix 2019) hence a compromise will need to be found.

SVM results has the hyperparameters *kernel type* and C value (a regulation parameter). Kernel type here refers to the function used to transform features to a higher dimensional plane, used to find the aforementioned 'hyperplane'. Lower values of C may lead to overfitting hence finding a reasonable value that can classify a diverse range of sentiment is the goal. While changing the kernel one may expect to see runtime performance differ (Bergstra et al. 2013) hence its of interest to optimise this as well.
It should be noted that all of these models have more hyperparameters available, however, it was thought due to their relevance to sentiment classification and rich documentation that these would be the best to tune.

Optimisation

In order to search through the hyperparameter space more efficiently in addition to comparing the performance of hyperparameter optimisation algorithms (compared to Bayesian optimisation), different algorithms were explored.
An initial attempt was made using grid search CV (Sklearn), this was an interesting base algorithm to compare against due to its simplicity. Grid search is an exhaustive search algorithm which will search through every possible parameter within a given bound and return the best performing (based on loss) (Hsu, Chang, and Lin 2010). The CV here refers to cross-validation used to test the generalisation of the model with the given parameters. While useful it is clear that it is a runtime-expensive algorithm, consider the following where p is */parameters/*, n is */data points/*, k is */folds/ in KFCV* and finally $f(1)$ is the runtime of the model on a single data point.

$$O(p) \cdot O(n) \cdot O(k) \cdot O(f(1)) \in O(pnk) \rightarrow \lim_{p,k \rightarrow \inf} \approx O(n^x)^*$$

It was evident in order to satisfy the optimisation objective for the project (table 1.2) another approach was needed. For this an evolutionary optimisation algorithm was chosen. Evolution optimisation revolves around the idea of creating and evolving populations ($h = |populations|$) and finding the fittest based on tourna-

*Simply expressing the function becomes exponential (Joel 2021) ment selection over some g number of generations. The algorithm is also known for being highly configurable (Devansh 2023), making it a perfect match when optimising the models mentioned previously (KNN, SVM and Random Forest). Runtime can be simplified to:

$$O(h) \cdot O(g) \cdot O(f(1)) \in O(hg) \lessapprox {}^{\dagger}O(pnk)$$

It was also decided with suitable gene embedding, the model could be used to find optimal vector embedding and model selection, this will not only better runtime but hopefully yield higher performance.

Data splitting

With an emphasis on binary classification, it was important to also consider representation within the training set (so the model avoids overfitting on a certain class). Given the BTC dataset, it was found that $\approx 54\%$ of the classifiers were 'buy' or 'hold' (i.e $c = 0$), hence it was decided that the Synthetic Minority Oversampling Technique (SMOTE) algorithm should be used. SMOTE aims to create synthetic minority class samples (posts on a day when it is best to sell) and then uses KNN to create new observations, balancing class distribution (Chawla et al. 2002).

Base-Model

The proposed model (discussed prior) relies mainly on classification, a diverse set of learners and ensemble methods (aggregation), traditionally concepts that have been associated with boosting methods. For comparison, the now popular XGBoost system is used which proves to be a highly scalable and configurable ensemble learning tool (T. Chen and Guestrin 2016). XGBoost much like other boosting methods relies on building confidence in classification via the aggregation of results yielded by weak learner models. It is hoped that the results obtained from the model show improved accuracy than that of XGBoost proving that there is a need for pre-defined classifiers compared to low-level weak learners associated with boosting.

[†]Due to the chaotic nature of genetic algorithms there may be edge cases where this does not hold

CHAPTER 4

IMPLEMENTATION

4.1 Data Gathering

Both models use large amounts of data, the success of the project heavily replies on several data sources, the following section discusses how this process was optimised.

Market Data Retrieval

Yahoo Finance limits API GET requests to 8000 per day. Bitcoin started being recorded on Yahoo Finance in 2014 and as of gathering the data, $\sim$3300 days of active trades were held. XRP and ETH have a combined $\sim$4000 entries meaning all days were able to retrieved without exceeding API rate limits. A Python script was used to loop through each coin and store the returned price from the API after checking the absence of API overuse and any network-related errors when accessing the data. The Pandas library was utilised to store the retrieved data in data frames, making it simpler to store within a CSV file, accessible by the deep learning model.

Sentiment Data Retrieval

Reddit Data

ProfitsBot V0 (Stewart 2023) provides both the raw data sets and a template Python repository used to replicate the data set. Due to an interest in only cryptocurrency related forum posts and the non-CV-compatible format of the raw data, it was decided to make a bespoke script to gather only relevant data and download it to a suitable format (CSV). Three separate scripts were used, one in order to generate the JSONL files for all of the posts from the subreddits of interest, another for all of the comments on each of these posts and then finally a script used to combine both datasets and convert it into a CSV file. Python was also used in order to group the CSVs (subreddits) by the coin the posts are relevant to and then add a '*Sell*' (binary) attribute to each record within these CSVs.

The sell attribute came from mapping the market data for a day (d) with all given posts related to a coin (C) i.e

$$Sell = \begin{cases} 1, & \text{if } C^d_{open} > C^d_{close} \\ 0 & \text{otherwise} \end{cases}$$

News Data

A script was written in Python to collect the 'title', any 'votes' the article may have had and its 'published date'. The API has different endpoints for the news of each coin, hence, a loop was used to gather data for each coin of interest. The Cryptopanic API limits users to daily news data meaning that a Cron job was needed to run the script each day in order to collate a substantial amount of data. Much like the Reddit data retrieval, a '*Sell*' attribute was added to the CSV files and then a single sentiment file was created storing both Reddit and news data for each coin.

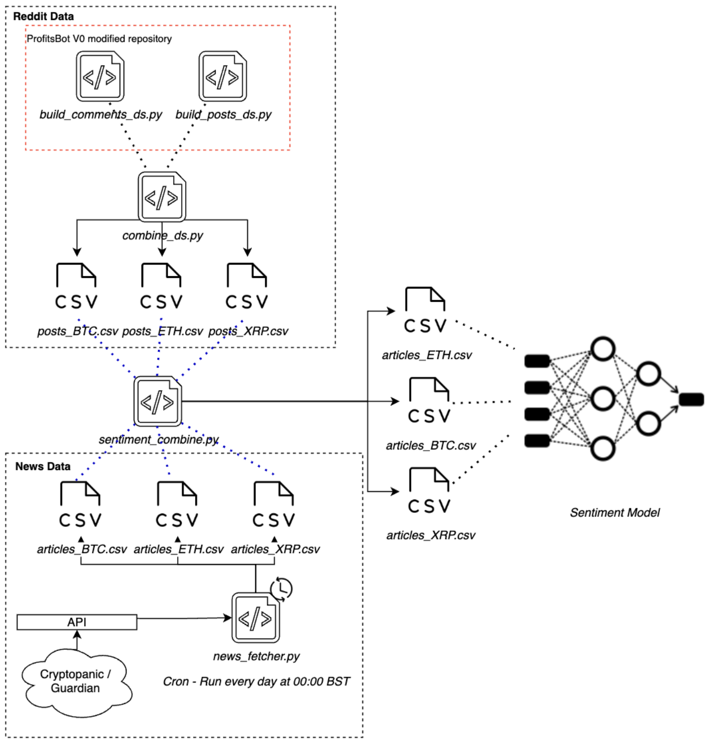


Figure 4.1. Sentiment data retrieval overview*

4.2 Closing Price Model

The following explores how the the closing price model was configured and problems that were encountered on the way and how these were overcome or why they may not have been.

Model Structure

Pre-Processing

Each coin is passed into a loop where the CSV holding the market for that coin is passed into a data frame. After careful consideration, it was chosen not to remove any values (i.e 'anomalies') due to the chaotic nature of market prediction meaning many data points can be seen as anomalies and are worth tracking. The data is then split 70% for training and 15% each for validation and testing, this means that the model is trained on the first 70% of closing prices, then validated on the last 15% and then finally tested against the closing prices between both sets. These sets are then normalised using minMax scaling provided by Sklearn, see below for the normalisation function.

$$x^i_{norm} = \frac{x^i - \min(x)}{\max(x) - \min(x)} \in [0, 1]$$

By using this form of normalisation it allows for maximums and minimums to still be considered (important within the context). The normalised sets are then split into two further subsets features (i.e. opening prices) and targets (closing prices), for example, the set *test_ data_ target* holds all closing prices within the test set. By doing this it is then possible to convert the sets into 'time series generators' provided by TensorFlow.

Training

Time series generators can be thought of as batches of feature target pairs used as suitable inputs for RNN's (LSTM is a modified RNN). The generator takes the features and targets, a 'length' parameter indicating the number of previous features to consider for any given target (*see below for visualisation*), a sampling

*The repository only stores articles_BTC as it's the only coin being used in the end rate (1 by default) and then finally the desired size of each of these batches (TensorFlow 2024).

$$TSG(X_f, X_t, L, SR, BS) \rightarrow \forall x_t^i \in X_t : x_t^i : [x_f^{i-1}, x_f^{i-2}, ..., x_f^{i-L}]$$

In order to prevent under or overfitting, through visualisation it was discovered that setting the length to 20 was optimal and the batch rate to 16. The training and validation (and test) sets are then given to separate time series generators and passed onto the model.

Model Creation

To ensure the scalability of the program, the process of model creation is modularised accounting for length, dropout rate, activation function, number of units and number of layers (num_layers) as parameters (simplifying the Bayesian optimisation and overall testing). Tensorflow is used to make a base Sequential model (as an LSTM is used), where individual layers and dropouts are then added through a loop based on the inputted num_layers. Once all layers have been added and it has been compiled it is then returned to calculate the loss.

Performance Checking

The loss function chosen was Root Mean Squared Error (*see below*).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

After each iteration of the validation model, the best_model is compared by checking the RMSE, this is also used when searching through the Bayesian Optimisation space at each stage looking for the lowest RMSE.
While useful, any loss function is dependent on the data set and split provided, meaning it is a less generalised measure. With this in mind, K-folds cross-validation was also implemented to gain insights into model performance within constantly changing markets, highlighting the importance of checking diversified data sets.
The basic idea is partitioning the market data into 'k' subsets or folds. The model is then trained and evaluated 'k' times, using a different fold as the test set in each iteration. While the remaining folds are used for training, an average RMSE is then provided (see Wong et al (2019) for further reading).

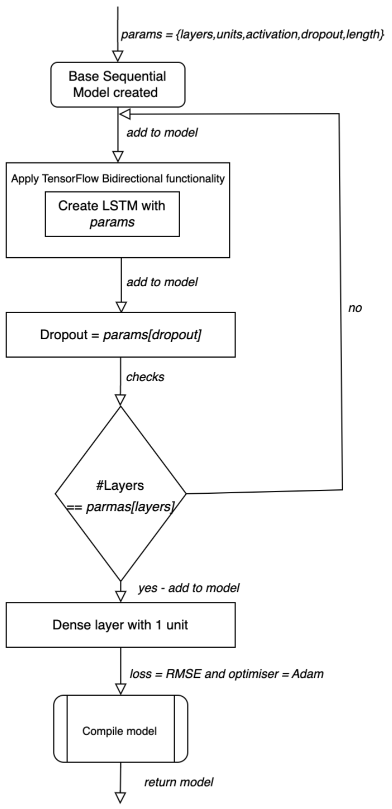

params = {layers,units,activation,dropout,length}
Base Sequential Model created
add to model
Apply TensorFlow Bidirectional functionality
Create LSTM with params
add to model
no
Dropout = params[dropout]
checks
#Layers == parmas[layers]
yes - add to model
Dense layer with 1 unit
loss = RMSE and optimiser = Adam
Compile model
return model

Figure 4.2. Flowchart for the implementation of the closing price model function

Optimisation

Baysian Optimiser

A function is used to perform optimisation, taking both training and validation time series generators as parameters. The search space is defined as an array holding different parameters for the model and their bounds namely: $|layers|$ (2 or 3), $|units|$ (50, 100 or 150), *activation* (ReLu, Tanh or Sigmoid), *length* (10,20 or 30), *dropout* (0.1 to 0.5) and $|epochs|$ (50 to 100). In order to get the optimal from this space an objective function is needed, calling the model creation method with parameter values from the search space. The losses for these configurations are then gathered by running the created model against both training and validation sets for the given number of epochs in the parameter. The function will then return the configuration with the lowest loss (RMSE). GPyOpt is used to invoke the objective function within the given domain (search space), adopting an expected

Algorithm 1 K-Fold Cross Validation Implementation Pseudocode

```
1: function K_FOLD_CROSS_VALIDATION($k, open\_prices, close\_prices, params$)
2:     $tscv \leftarrow$ TimeSeriesSplit($n\_splits = k$)          ▷ $k = 5$ found to be optimal
3:     $rmse\_list \leftarrow []$
4:     for $train_i, test_i$ in $tscv.split(open\_prices)$ do
5:         $train_{open} \leftarrow open\_prices[train_i]$          ▷ Coin prices assigned to folds
6:         $test_{open} \leftarrow open\_prices[test_i]$
7:         $train_{close} \leftarrow close\_prices[train_i]$
8:         $test_{close} \leftarrow close\_prices[test_i]$
9:         $tsg\_train \leftarrow$ TimeseriesGenerator($train_{open}, train_{close}, ...$)
10:        $tsg\_val \leftarrow$ TimeseriesGenerator($train_{open}, train_{close}, ...$)
11:        $loss \leftarrow$ train_model($params, tsg\_train, tsg\_val$)
12:        $rmse\_list$.append($loss$)          ▷ $params$ provided by Baysian Optimiser
13:    end for
14:    $avg\_rmse \leftarrow$ avg($rmse\_list$)          ▷ Mean over RMSE
15:    return $avg\_rmse$
16: end function
```

improvement acquisition strategy looking to balance exploration and exploitation using a Gaussian process as the underlying surrogate model. This is then run for a set number of iterations in order to cover the domain as best as possible while not compromising run time ($iteration \in [5, 20]$).

Parallelisation

The Pool module in Python is used to distribute workload across multiple devices (replicas) on the machine. The program was run on a single CPU, spreading the workload over multiple cores ($|cores| = |coins| = 3$) however with larger-scale implementations (more devices) it is expected to exponentially reduce runtime. Issues were faced initially as a result of continuous distribution within parallelised cores (recursive workload splitting), hence, $freeze\_support()$ (from Pool) was employed in order to only distribute on the initial core. With *epochs* going up to 100 (see search space) it is useful to optimise run time and reduce the workload of individual cores. As can be seen in the graph below parallelisation on average achieved a speedup of $\sim 51.28\%$, making the result formulation more efficient and contributing to the scalability of the project. This helped complete objective 3 of the project (table 1.2).

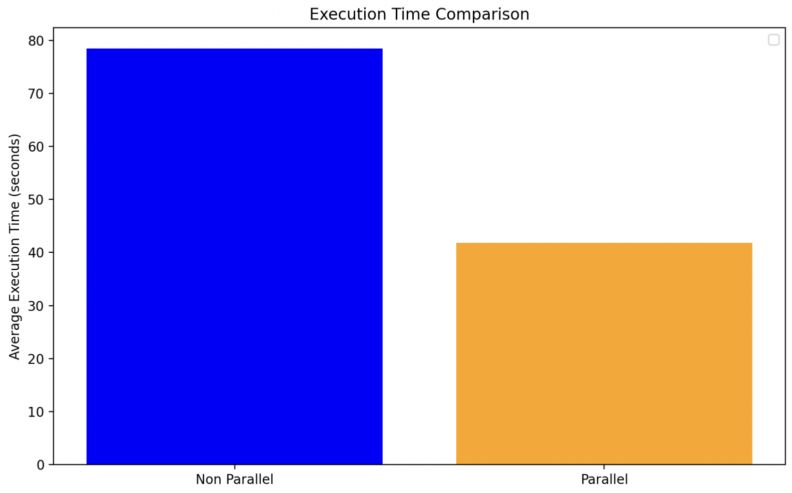


Figure 4.3. Execution time comparison per stock average

Adam

Tensorflows implementation of the Adam optimiser is used to compile the LSTM model. Adam (Adaptive Moment Estimation) is a stochastic gradient descent algorithm used to dynamically adjust learning rates (i.e. the size of steps to adjust weights) of parameters (TensorFlow 2023b). It was decided to manually set the learning rate to 0.001 as an initial value in order to gain better coverage and avoid local minima using RMSE as the loss function.

Visualisation

The Matplotlib Python library was used in order to visualise results. Before any computation is done an empty plot instance is created at the end of each loop (going over each coin) the results are recorded and added to the plot. In order to compare the actual closing price is then mapped the same graph (the $test\_features$) as well as a linear regression (provided by the Sklearn library). Finally, in order to simplify the review process each subplot is titled in the following form:

{Stock,Config,RMSE,K_Fold_RMSE}

4.3 Sentiment Model

The following discusses the techniques, tools and difficulties found when developing the sentiment-orientated model. It should be noted that the development of this model remains autonomous with that of the attempt discussed in section 4.2.

Pre-Processing

The raw data is put into data frames (Pandas) labelled '*id*', '*title*' (of post) and '*date*', easing data handling. Before any form of vectorisation on the posts can be applied, it was vital to apply suitable sanitisation and tokenisation. The implementation of this process can be split into 7 distinct parts.

1. Removal of capital letters
2. Removal of links
3. Removal of whitespace
4. Removal of digits
5. Tokenisation
6. Removal of stop words
7. Lemmatise Tokens

NLTK (Natural Language Tool Kit) was a Python module used to automate a lot of the process of sanitising the titles. Steps 1 to 4 can be simply thought of as pattern-matching tasks in which NLTK searches through the data frames and employs a regular expression.
Tokenisation (5) refers to the process of taking a sentence or paragraphs, say $s =$ *'BTC is amazing'* and breaking it into sub sentences (words) or subparagraphs (sentences) giving $w =$ *['BTC', 'is', 'amazing']* for example. While a stop word (6) refers to common 'non-essential' words that help human understanding yet tend not to bring any value to the sentiment, examples of this include: 'the' or 'an' or 'but'.
Finally, lemmatisation reduces inflections of words to their base form (eg 'stocks' becomes 'stock' and 'performing' becomes 'perform') which are known as lemmas. Pseudocode explaining the implementation of this can be found below.

Algorithm 2 Lemmatise Tokens

```
function LEMMATISE_TOKENS(tokens)                                ▷ Tokens from post title
    lemmas ← []
    function GET_WORDNET_POS(word)                               ▷ Part of Speech Tag
        tag ← pos_tag([word])[0][1][0].upper()                   ▷ Tuple of word POS pairs
        tag_dict ← {"J": ADJ, "N": NOUN, "V": VERB, "R": ADV}
        return tag_dict.get(tag, NOUN)
    end function
    for each token in tokens do                                  ▷ NLTK lemmatiser function
        lemma ← lemmatiser.lemmatise(token, GET_WORDNET_POS(token))
        lemmas.append(lemma)
    end for
    return lemmas
end function
```

Part of Speech (POS) here refers to which category each token in the title belongs to, i.e Noun (N),Verb (V), Adjective(J) or Adverb (R).

Data Splitting

Due to an imbalance in classes (i.e. more sell than buy), Imblearn (from SkLearn) is used in order to provide a simple implementation for SMOTE. Simply, the training features and classifiers are fitted to the provided SMOTE function returning, the desired class distribution on the training set. This results in the following distribution.

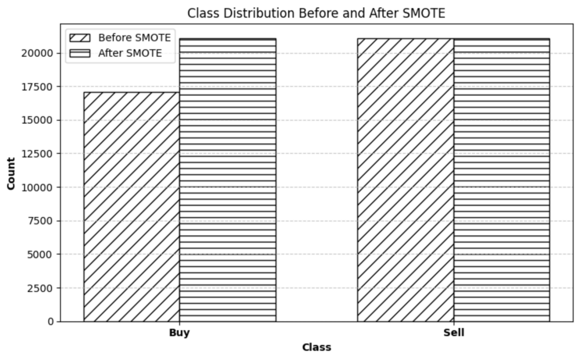


Figure 4.4. Class distribution before and after SMOTE.

Vectorisation

In order to simplify the implementation of TF-ID, Sklearn was used to automate a large portion of the process. An instance of TfidfVectorizer (Sklearn) is set up and applied to each title in the data frame. This is then passed onto the model to start training.

In order to obtain the GloVe embedding (word-value pairs) a local Python dictionary is created to store the pairs. The pairs are downloaded from https://nlp.stanford.edu/projects/glove/ and looped through then passed onto the dictionary. A function iterates through each token within the token stream and receives its embedded GloVe value, it will then find the average value of the post or article and return this to a separate data frame which can be used for model consumption. USE embeddings are loaded in from https://tfhub.dev/google/universal-sentence-encoder/4 using TensorFlow Hub; these are then applied to each title within the data frame. Then, the USE title vectors can be utilised within the model training.

Model Creation

When testing the sentiment (as mentioned prior) there was a lot of preprocessing involved hence Jupyter Notebook was used to isolate the model creation code segment making runtime when testing much lower.

Initial Attempt

Classifications are made within a single evaluation function, acting as the target for the genetic hyperparameter optimisation algorithm. Taking the input of an individual (from the optimiser) the form of the embedding is gathered, and then convert the titles (features) into the appropriate format. The data is then randomly split into testing and training sets (80:20 via Scikit-Learn) and parameters are obtained by using the parts of the individual as indexes to a dictionary storing all parameters. All three classifiers used are provided by Scikit-Learn and hence have the ability to fit into the training set and then make predictions based on this. The accuracy of the model is returned, and useable by the optimiser.

Aggregation Attempt

The updated attempt follows the structure of the evaluation function mentioned within the initial implementation, with some crucial differences. As discussed, there is a difference within model outputs, namely, we now care for the probability p_i of a sell call for a given d_i. This is achieved by no longer randomly obtaining test and training sets, but rather manually splitting the data to maintain the sequential nature of the data. Scikit-Learn's *predict_proba()* function (as opposed to simply using *predict()*) is utilised in order to return the confidence of a given classification. With the assurance of posts sequentially (ordered by date), a dictionary is used to match the indexes of the test set with the original data frame, allowing us to store the dates with associated predictions. This dictionary is then looped through and averaged for each day to determine if γ (threshold) has been surpassed and updates a results array appropriately which can then be compared to the C_{actual} to obtain the loss. The implementation is described in Algorithm 3.

Boosting

The most comprehensively documented and easiest to use module for boosting was 'Xgboost' (xgb), this allowed an easy implementation which fits, trains and predicts based on the data with built-in functionality. It should be noted that as the data is being handled with Pandas data frames, it was important to convert to an Xgboost compatible input, hence, a dense matrix was used to store training and testing data (provided by xgb). Parameters were set to: *'max_depth': 3, 'learning_rate': 0.1, 'n_estimators': 100*, to highlight the need for optimisation.

Algorithm 3 Evaluation Function Pseudocode

```
1: procedure EVAL(individual)
2:     SPLIT_INDEX ← $|titles| \cdot 0.8$        ▷ Training and test set built with index
3:     date_predictions ← {}
4:     titles_embedded ← titles.apply(individual$_{embed}$)
5:     X_train, y_train ← apply_smote(X_train, y_train)
6:     classifier ← classifiers[individual$_{classifier}$]
7:     model ← classifiers[classifier].set_params(individual$_{params}$)
8:     model.fit(X_train, y_train)        ▷ Train
9:     sell_confidence ← model.predict_proba(X_test)
10:    sell_dates = {}
11:    for $i \in \{0, ..., |sell\_confidence|\}$ do        ▷ Assign confidence to days
12:        if $d_i$ in sell_dates then
13:            sell_dates[$d_i$].append(sell_confidence$_i$)
14:        else
15:            sell_dates[$d_i$] ← [sell_confidence$_i$]
16:        end if
17:    end for
18:    res ← []
19:    $\gamma$ ← thresholds[individual$_\gamma$]
20:    for $d_i$ in sell_dates do        ▷ Aggregate results for each day
21:        average ← $\frac{1}{|d_i|}\Sigma_j^n(\forall c_j \in d_i)$
22:        if average $\geq \gamma$ then
23:            res.append($0 * |d_i|$)
24:        else
25:            res.append($1 * |d_i|$)
26:        end if
27:    end for
28:    $f$ ← accuracy(res, y_test)        ▷ return fitness of aggregated results
29:    return $f$
30: end procedure
```

Optimisation

Grid Search

The initial attempt to use Grid Search CV was realised by creating an instance of each of the three models and then only returning each model found to have optimal parameters. This was accomplished using Scikit-Learn's GridSearchCV built-in functionality and creating a dictionary storing each of the bounds for each hyperparameter mentioned in Section 3.3 (defining the search space).

Genetic Optimisation

The genetic optimisation algorithm, on the other hand, relies heavily on a highly flexible and distributed evolutionary Python library known as DEAP (Fortin et al. 2012). First of all, individuals within the population are defined, i.e. an individual i may be '01120' representing the index of classifier, hyperparameters, γ and embedding, an example can be seen below.

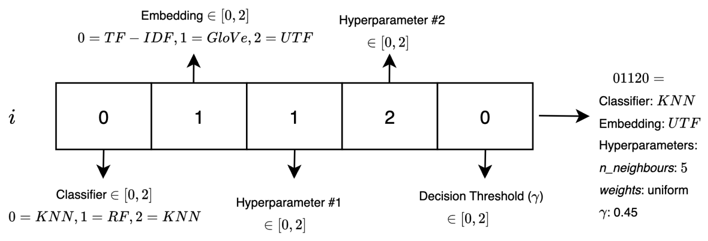


Figure 4.5. Individual Example

A base fitness (known as the fitnessMin) is created to serve as a baseline for future selection. To determine the fitness of an individual a custom evaluation is used to run the model (within the previously discussed loop) and return the accuracy for each; which is run separately within a function and bound to the individuals.

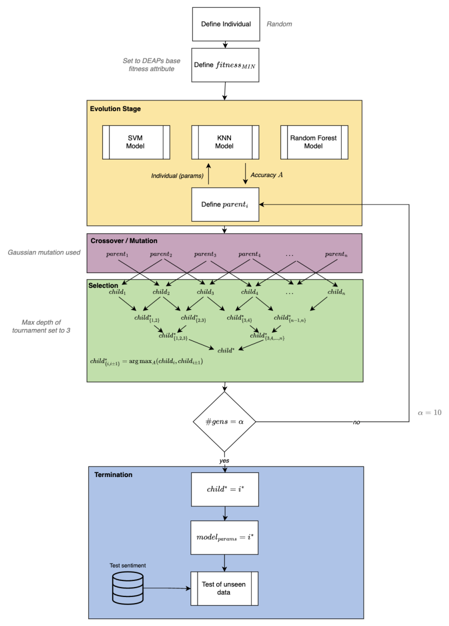

Define Individual
Random
Set to DEAPs base fitness attribute
Define $fitness_{MIN}$
Evolution Stage
SVM Model
KNN Model
Random Forest Model
Individual (params)
Accuracy A
Define $parent_i$
Crossover / Mutation
Gaussian mutation used
$parent_1$ $parent_2$ $parent_3$ $parent_4$ … $parent_n$
Selection
$child_1$ $child_2$ $child_3$ $child_4$ … $child_n$
Max depth of tournament set to 3
$child^*_{\{1,2\}}$ $child^*_{\{2,3\}}$ $child^*_{\{3,4\}}$ $child^*_{\{n-1,n\}}$
$child^*_{\{1,2,3\}}$ $child^*_{\{3,4,\ldots,n\}}$
$child^*$
$child^*_{\{i,i\pm1\}} = \arg\max_A(child_i, child_{i\pm1})$
$\#gens = \alpha$
no
$\alpha = 10$
yes
Termination
$child^* = i^*$
$model_{params} = i^*$
Test sentiment
Test of unseen data

Figure 4.6. Optimiser visualised

The following shows the values associated with the individuals defined. The *eval* function uses this parameter grid in order to extract the specific values from the embeddings.

Table 4.1. Individual table

Individual Index	**Index Value** (v)		
	0	**1**	**2**
Classifier(i_0)	KNN	RF	SVM
Embedding(i_1)	TF-IDF	GloVe	UTF
Threshold(i_4)	0.45	0.5	0.65

Parameters defined in i_2 and i_3 vary depending on the classifier. These represent the specific hyperparameters for each model within the classifier given. i_2 is defined below:

$$i_2 = \begin{cases} k[v] \text{ where } k = [3, 5, 7], & \text{if } i_0 = KNN \\ n[v] \text{ where } n = [50, 100, 200], & \text{if } i_0 = RF \\ C[v] \text{ where } C = [0.1, 1, 10], & \text{if } i_0 = SVM \end{cases}$$

We define i_3 as[†]:

$$i_3 = \begin{cases} w[v] \text{ where } w = [uniform, distance], & \text{if } i_0 = KNN \\ d[v] \text{ where } d = [0, 10, 20], & \text{if } i_0 = RF \\ kernal[v] \text{ where } kernal = ['linear','rbf','poly'], & \text{if } i_0 = SVM \end{cases}$$

[†]if $w[2]$ then $v = v \mod 2$ $(v \in [0, 1])$

CHAPTER 5

RESULTS

5.1 Market Data Results

The Bayesian Optimiser was called 10 times, setting $iteration_{Max}$ to 12, leading to $\sim$ 120 configurations being conducted. Of these configurations, the top 10 best performing (i.e. lowest RMSE) were documented. It should be noted that all RMSE scores have been left normalised in order to make comparisons between coins fair. In order to shorten run-time, the K-folds generated RMSE was not used for each configuration. *See Table 6.1.*

Table 5.1. Model Configurations and Performance. All values rounded to 4 d.p.
Diff = $|RMSE_{Linear} - RMSE_{Produced}|$

Rank	#Layers	#Units	Activation	Length	Dropout	Epochs	RMSE	Diff
1	2	50	Tanh	30	0.2133	50	0.00021	+0.1325
2	3	100	Tanh	10	0.2132	100	0.00023	+0.1325
3	3	100	Tanh	10	0.1913	80	0.00028	+0.1324
4	2	100	Tanh	30	0.2332	50	0.00031	+0.1324
5	2	50	Tanh	30	0.3286	90	0.00074	+0.1320
6	2	50	Tanh	20	0.4918	100	0.00400	+0.1287
7	3	100	ReLU	10	0.4535	70	0.00912	+0.1236
8	2	50	ReLU	20	0.3286	60	0.01675	+0.1159
9	3	50	Sigmoid	20	0.4764	50	0.02076	+0.1119
10	2	50	Sigmoid	10	0.2132	60	0.02183	+0.1109
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Note: For simplicity the configuration at rank 1 will be referred to as BTC^*

BTC^* was ran on the K-Fold algorithm to test its generalisation ($k = 5$).

Table 5.2. RMSE for each fold in k-Fold Cross-Validation and K-Fold average

Fold	1	2	3	4	5
RMSE	0.00051	0.00024	0.0142	0.0002	0.05
K-Fold Average					**0.01303**

The performance of BTC^* is visualised with the MatplotLib instance below in Figure 5.1. This provides better insight into when the model was performing well compared to when there were significant losses.

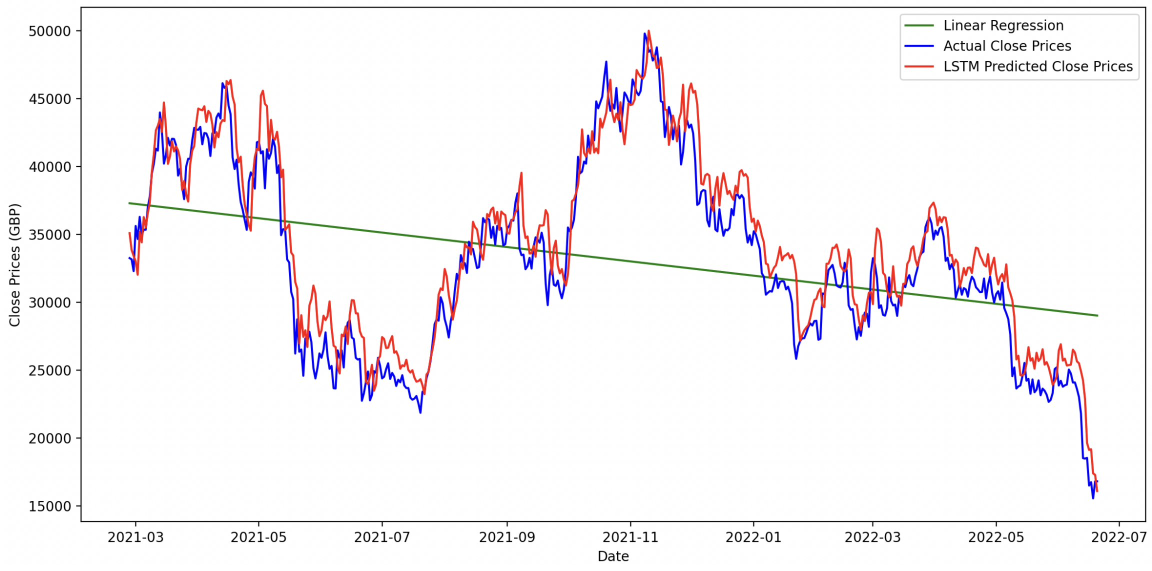


Figure 5.1. C^* vs $BTC actual closing vs linear regression over 18 month period.

To further investigate where the loss was being found, the residuals (i.e. the difference in predictions and actual $BTC close price) were mapped within a histogram.

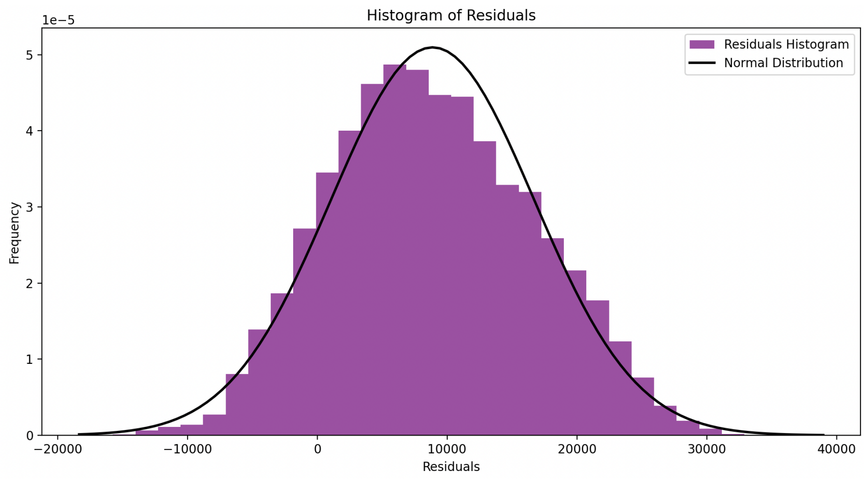


Figure 5.2. Residual plot mapped against normal distribution for $BTC

Table 6.3 shows the best-performing configurations provided by the Bayesian Optimiser for Ethereum and XRP (i.e ETH^* and XRP^*).

Table 5.3. ETH^* and XRP^* Configurations and Performance. All values rounded to 4 d.p.

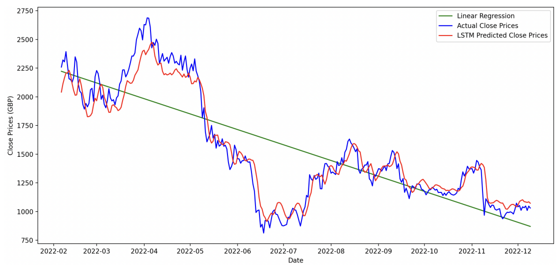
Coin	#Layers	#Units	Activation	Length	Dropout	Epochs	RMSE
ETH	2	100	Tanh	20	0.3902	50	**0.0002**
XRP	3	150	Tanh	30	0.436	100	**0.0001**

The RMSE provided by the K-folds algorithm can be seen in Table 6.4 for ETH^* and XRP^*.

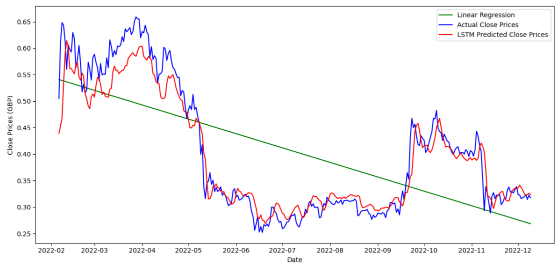
Table 5.4. K fold average for ETH and XRP.When $k = 5$.

	Fold					K-Fold Average
	1	2	3	4	5	
ETH	0.0031	0.00002	0.00004	0.0212	0.0016	**0.005**
XRP	0.00005	0.0009	0.002	0.0003	0.0001	**0.0006**

The performance of ETH^* & XRP^* over the testing period is better visualised below in Figure 6.3. Note the testing period referred to is not consistent for all coins here due to varying quantities of available market data.



(a) ETH^* vs ETH_{actual} vs ETH_{linear} closing price.



(b) XRP^* vs XRP_{actual} vs XRP_{linear} closing price.

Figure 5.3. Coin performance over the testing period.

A final measure for the coins is sMAPE (Sarra 2022) (Symmetric Mean Absolute Percentage Error), which takes the average of the percentage difference between predicted and actual values. While RMSE is useful for measuring the model performance, this project seeks to provide profitability for traders (see section 1.3 for more), hence a low average percentile difference is critical. We define sMAPE as:

$$\text{sMAPE} = \frac{1}{N}\sum_{i=1}^{N} \frac{|\hat{y}_i - y_i|}{(|\hat{y}_i| + |y_i|)/2} \times 100 \tag{5.1}$$

Table 6.5 provides the found sMAPE values for each coins optimal configuration.

Table 5.5. sMAPE Values

	BTC^*	ETH^*	XRP^*
sMAPE	22.767%	32.978%	29.944%

5.2 Sentiment Results

We first look at the performance of XgBoost compared to the initial attempt, providing a baseline to how well the aggregated attempt performed. The values stated are the best results found from each embedding using said method.

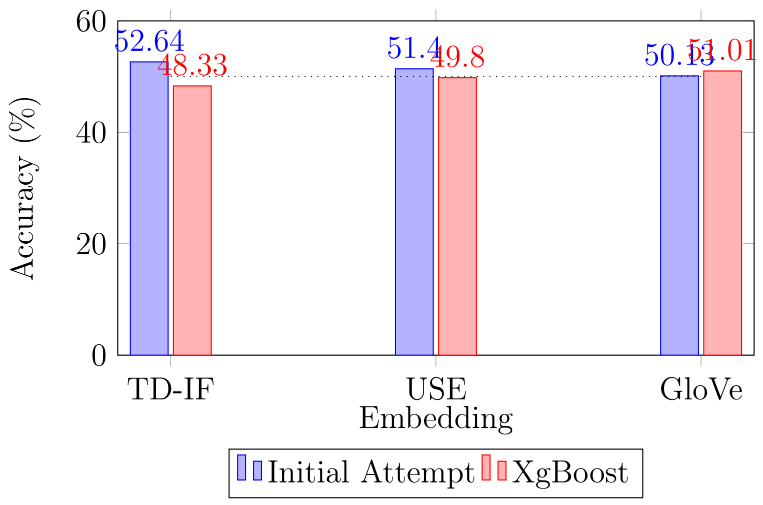


Figure 5.4. Accuracy comparison of XgBoost and Initial Attempt

The following results were obtained by assigning the genetic algorithm the parameters $\mu = 10, \lambda = 20, p_{xo} = 0.7, p_m = 0.2, \alpha = 10^*$. This meant appropriate samples were taken while not causing excessive runtime. See table 6.6 for the updated models performance, where comparison refers to the best counter model (i.e linear or XGB or initial attempt).
For individual decoding refer to table 5.1.

Table 5.6. GA Optimizer Hall of Fame

Individual	**Fitness (f)**	**Comparison**
11211	0.5299	+0.0035
02002	0.5278	+0.0014
01202	0.5241	-0.0023

$^*\mu$: Parents considered, λ: Children produced, p_{xo}: Chance of crossover, p_m: Chance of mutation , α: Number of generations

We also consider cases where more than a single day's results are aggregated in order to make predictions. Table 5.7 presents the cases where $n = 2$ and $n = 3$.

Table 5.7. Hall of fame for n days considered

(a) 2 days considered

Individual	**Fitness (f_2)**	**Comparison**
10202	0.5366	+0.0102
01101	0.5285	+0.0021
01121	0.5285	+0.0021

(b) 3 days considered

Individual	**Fitness (f_3)**	**Comparison**
21011	0.5387	+0.0123
10102	0.5338	+0.0074
10002	0.5325	+0.0061

For simplicity i_n^* denotes the best performing individual when n days are considered. And to refer to overall best performing individual i^* (namely 21011) is used.

Consider the confusion matrix of i^* which highlights where exactly the model is making errors. The predicted set has size 9544.

	Predicted	
Actual	**Buy**	**Sell**
Buy	2087	1903
Sell	2499	**3055**

Table 5.8. i^* Confusion Matrix

Given the table 6.8 it is possible to compute other useful metrics for i^* such as:

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{3055}{3055 + 1903} = \mathbf{0.6162}$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{3055}{3055 + 2499} = \mathbf{0.5501}$$

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \times \frac{0.6162 \times 0.5501}{0.6162 + 0.5501} = \mathbf{0.5813}$$

Figure 6.5 visualises the monthly returns for the year of 2022 of the best performing aggregated model (i^*) and the initial model. This was calculated by finding the % error for each month of both models.

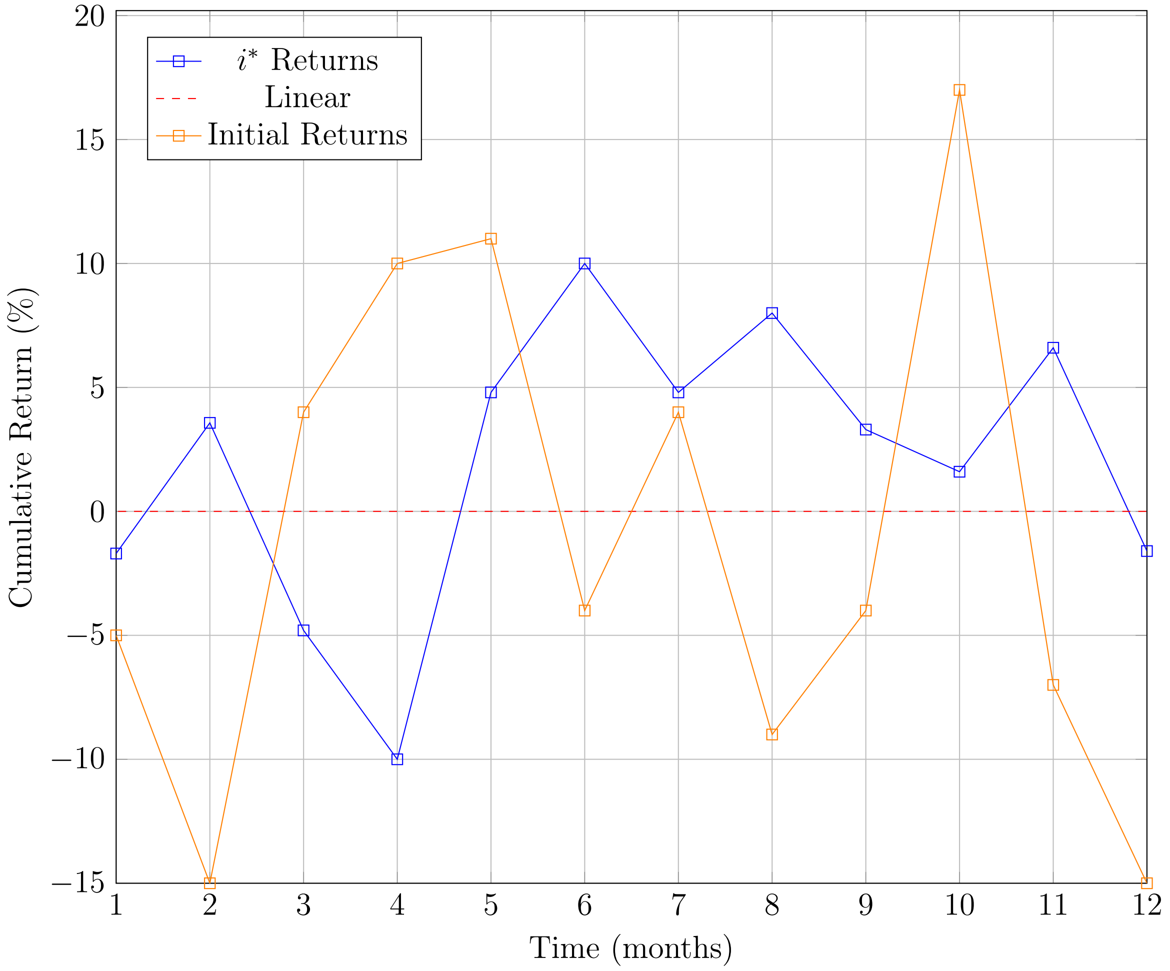


Figure 5.5. Monthly performance for i^*

CHAPTER 6

DISCUSSION

Chapter 5 is informative yet poses several questions. In regard to the closing price model, there is the question of activation dominance, generalisation capability, visual performance and smaller coin success. While the sentiment model results highlight the need for aggregation, i^* performance, attempt comparison and profitability. The following chapter aims to answer and expand on these topics.

6.1 Market Data Results Discussion

BTC findings

Why Tanh?

Table 6.1 provides useful insight into how the model is learning and what its optimal configurations may be. It is evident that Tanh (see 7.1) is the dominating activation function, this could be down to several reasons including information loss and optimisation strategy.

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \in [-1, 1] \tag{6.1}$$

The Sigmoid function (see 7.2) suffers from its derivative exponentially reducing in output space (see 7.3). Potentially resulting in information loss as model layers increase and values eventually converge (Himanshu 2021).

$$\sigma(x) = \frac{1}{1 + e^{-x}} \in (0, 1) \tag{6.2}$$

$$\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x)) \ \in (0, 0.25] \tag{6.3}$$

Due to the project implementing a deep learning model, this will reduce the range to at least $(0, 0.25]^*$, when considering the heavy fluctuation of BTC closing prices over the training and testing periods, this must be captured by the model which may converge in this case. With a larger range of $[-1, 1]$, Tanh looks to resolve this issue potentially helping it outperform Sigmoid.

*] is used to denote exclusivity and (for inclusivity

As well as RMSE (result objective 1), execution time is considered when running through configurations for the Bayesian optimiser. When running LSTM layers in Tensorflow it is possible to accelerate run time with GPU optimisation tools such as CUDNN when using Tanh (TensorFlow 2023a), however, the criteria for this are only met with Tanh meaning other activation functions such as ReLU will use a generic GPU kernel as fallback (much slower). As a result, it is likely the optimiser will produce more Tanh configurations making it the most explored activation. With scalability, an aspiration for the project would be to produce more results for activations such as ReLU, Sigmoid or even LeakyReLU by exploiting the parallelisation referred to in section 4.2.

Optimal Configuration & Generalisation

BTC^* seems to differ in many ways from subsequent configurations however (apart from Tanh) it seems a lower dropout is preferred in general, this is likely to due any higher rates leading to underfitting. BTC^* does however run the risk of overfitting with a large length and low dropout, looking at table 6.2 initially suggests better generalisation may be needed, yet for $k = \{1, 2, 4\}$ it seems to show it works best in specific market states. It is also reassuring to see the MSE produced outperforms the baseline set out in section 1.3 (objective 1).

Visualisation

Looking at Figure 6.2, the model does not seem to make any sizeable misclassification, however, a slight shift between the actual and predicted closing prices can be seen. This can be explained through how the model is trained, consider some closing price cp^d on day d, the opening price for the next day will be roughly $op^{d+1} \approx cp^d$ (there is also after-hour trading however this is negligible). As each prediction is based on $op^d, ..., op^{d-length}$ it is in theory lagged, however, this is to be expected of the model due to the nature of the problem. Another interesting observation is how the model deals with volatile times such as the flash crash of May 2021 (Morrow 2021), due to it making day trades it can keep up with the volatile essence of the market. While useful, it still seems to slightly overestimate in chaotic periods hence prompting a sentiment approach (section 4.3).
Figure 5.2 shows the residual histogram resembles a normal distribution plot. This indicates the model is unbiased and when errors occur they are of random nature

(Pardoe 2023), suggesting the model struggles with the volatility of the system and not with the learning process itself.

ETH & XRP findings

From Figure 5.3, it is evident that BTC^* in general looks to be more accurate. This is likely due to an increase in volatility with smaller coins, with fewer investors markets can swing faster and unexpectedly. This does, however, lead to better generalisation of prices (as seen in Figure 5.3), further backed up by the lower K-Fold average as well. With this generalisation in mind, it is interesting to note the optimiser's choice of larger dropout for both ETH and XRP in order to prevent overfitting when price fluctuations are common.

Market Data final thoughts

It is positive to see that sMAPE is far less than 50% for all coins (see section 1.3) indicating that it is a much better alternative to guessing close prices based on instinct. This also proves to be a useful tool for investors for all given coins, however, seems to be a harder task as the size of the coin decreases (as seen with ETH and XRP). Conversely, stock forecasting (generally deemed to be more stable) using this model is an area of interest for future research.

6.2 Sentiment Results Discussion

Attempts overview

"Rule No. 1: Never lose money. Rule No. 2: Never forget rule No.1"
- Warren Buffett

The first and most obvious observation is that models are at minimum breaking even, however, what is of interest is comparatively how much the model is accurately predicting.

XgBoost and Initial attempt

When observing Figure 5.4 it's clear that on average the initial attempt performs better than XgBoost, and in fact, XgBoost often produces a negative return.

This suggests that weak learners may not be able to handle more complex patterns which pertain to financial markets, it also points to a need for parameter optimisation (as seen in the initial attempt). While positive, the initial attempt looks to be on average close to 50%, suggesting that there is a significant aspect of randomness and that the calls for aggregation were warranted due to a lack of context.

Aggregated results

When looking at the initial results for $n = 1$ (table 5.6) it becomes apparent that the fitness of the top-performing individuals is similar and all produce accuracy over 50%. Another interesting observation is that better-performing individuals (same for $n = 2$ and $n = 3$) use higher γ values, implying that buying is typically more preferred.
When considering $n > 1$, we see better-performing results, suggesting that the greater level of data accumulated increases the confidence by expanding the context. However, it can also be noted that as n increases, f gains are apparent yet marginal, potentially showing that f_n gains begin to plateau or even reduce as n increases. This could be a cause of increased noise making it even harder to make predictions, with old posts having too much weight in current market news.

i^* analysis

Similar to i^*_1 , there is a preference towards $\gamma = 0.5$ and GloVe embedding, giving the idea that with suitable embedding there is no need to change the threshold. It is also positive to see that i^* is $\sim$ 4% greater than 50%, meaning in long-term investment, we are likely to see reasonable returns. With only 28% of American investors stating they had profited off cryptocurrency in 2022 (Evans 2023), it shows the promise of the model.
When considering how the model classifies the data it is clear from the confusion matrix (table 5.8) that it is best at predicting when to sell. Selling at the correct time means that money is not being lost, whereas not buying at the right time can mean less gains, as mentioned before at minimum the goal is not to lose money, only then profiting can be considered. As a result, money-making opportunities are often missed out by the model however this could be a result of an uneven distribution of classes within the test set (more sell) compared to the training due to SMOTE.

With the cost of misclassified sell calls, it is encouraging that the precision is above 60%, however, due to the worse performance of buy classifications, it is reflected within the recall score. As a result, an F1 score can be seen that lies in between the two and remains relatively high considering the highly chaotic nature of the classification task.

Comparison

As is evident from Figure 5.5, i^* looks to have a lower standard deviation as opposed to the initial attempt. In the case of October, the initial attempt has much higher returns, however, as can be seen in several other months there are much lower drops in returns. As mentioned previously a possible explanation for this could be due to the lack of context and with the 3 days' worth of post data the returns show more stability.
When considering the difference between i^* and the linear attempt, hypothesis testing is used to prove that the produced model is significantly better than a random or passive approach.
Null Hypothesis (H_0): There is no difference in accuracy between the two models.
Alternative Hypothesis (H_1): The accuracy of i^* is higher than the accuracy of the linear model.
Significance level (α): 0.05[†]
Formally checking:

$$\text{Decision} = \begin{cases} \text{Reject } H_0 & \text{if } p < \alpha \\ \text{Fail to reject } H_0 & \text{if } p \geq \alpha \end{cases}$$

First, the test statistic (z) is calculated as follows:

$$z = \frac{(\hat{p}_{i^*} - \hat{p}_l)}{\sqrt{\hat{p}(1-\hat{p})\left(\frac{1}{n_{i^*}} + \frac{1}{n_l}\right)}} = 5.254$$

where:

- $\hat{p}_{i^*}$ and $\hat{p}_l$ are the sample proportions (accuracy) of both models 0.5387 and 0.5 respectively.

[†]Different values of α could be used but 0.05 is the most common Moore, McCabe, and Craig 2017

- n_{i^*} and n_l are the sample sizes (number of observations) for the models 9544 for both.
- $\hat{p}$ is the pooled sample proportion.

Then the p value can be formulated as follows:

$$p = P(Z \geq |z|) = 7.45 \times 10^{-8}$$

Where Z is a standard normal random variable and $|z|$ represents the absolute value of the calculated test statistic z.
With these values, it is possible to obtain the decision of the hypothesis:

$$p < \alpha \Rightarrow \text{Decision} = \text{Reject } H_0$$

It can be concluded with confidence that the produced model is significantly better performing than a linear (random or neutral) approach showing that objective 2 has been met (section 1.3).

Returns

With day-traders in mind consider that an investor initially puts in £1000 (P_0) in January 2022 and wants to see how much profit (P) can be made with both attempts discussed within section 5.2 by December 2022. The linear (non-investment strategy) is trivial and we get:

$$P_{linear} = P_0 - 0 = 1000$$

$$ROI_{linear} = \frac{\text{Net Profit}}{\text{Cost of Investment}} \times 100\% = \frac{0}{1000} \times 100\% = 0\%$$

Next, the initial attempt is considered. For simplicity, assuming an automated process, a consistent amount equivalent to 1% of the current holdings is purchased, and the entire position is sold upon selling. Additionally, trading occurs daily, irrespective of market conditions. Market data on daily percentage returns is referenced, and model performance is evaluated leveraging Figure 5.5.

Let P_t represent the current investment amount at time t.

The daily percentage change in stock prices, denoted by δ_t, is calculated as:

$$\delta_t = \frac{\text{stock price}_t - \text{stock price}_{t-1}}{\text{stock price}_{t-1}} \times 100$$

The current investment amount at time t is adjusted based on the percentage change as follows:

$$P_t = P_{t-1} \times \left(1 + \frac{\delta_t}{100}\right)$$

$$\text{buy amount} = 0.01 \times P_t$$

The current amount is updated after the buy operation:

$$P_t = P_t + \text{buy amount}$$

The result gives us the following profit:

$$P_{inital} = \sum_{t=0}^{N} P_t = 1506.38 - 1000 = 506.38^{\ddagger}$$

$$ROI_{inital} = \frac{506.38}{1000} \times 100 = 50.64\%$$

Following the same methodology we get the following profit for the aggregation model.

$$P_{i^*} = \sum_{t=0}^{N} P_t = 2527.25 - 1000 = 1527.25$$

$$ROI_{i^*} = \frac{1527.25}{1000} \times 100\% = 153\%$$

The results here suggest that following the model decisions, there is the ability to be profitable (objective 4). It should be noted that these results are obviously impacted by volatility in the market, hence, it is ill-advised to solely rely on ROI (also consider Figure 6.2).

6.3 Final thoughts

It is reassuring to see that both models have achieved positive metrics and show clear strengths. The findings are a testament to the recent evolution of accessible AI tooling. However, there are still areas for improvement such as market forecasting with the aim of optimising a portfolio. A future effort may look to exploit the two models in order to achieve this by combining their capabilities.

‡Algorithm built to compute this value

CHAPTER 7

CONCLUSION

This project has been the result of ∼seven months' worth of coding, documenting and theorising. The following chapter reflects on some of the aims, improvements and personal developments of the project as well as a brief insight into what the future may hold for deep learning in cryptocurrency prediction.

7.1 Aims reflection

Goals

The main goal of the project is to answer the following,
'Does machine learning as a whole have a role in modern-day cryptocurrency portfolio management?' Based on the research and results found the answer is undoubtedly yes, but there is the question of whether it helps day traders or is useful within academic circles. The sentiment model showed that a long-term loss-tolerant strategy is needed which for many day traders is unviable. The models also prove that external factors play a large part which requires an extensive amount of data typically making it harder for academics to replicate (see literature review). It is thought that future profitable attempts within this field will remain within established funds. However, the success of optimisation within both approaches supports the notion that at its core it is possible for individuals as well as corporations to experiment with a range of forecasting methods (along with the help of tool automation progression). The promising results of parallelisation also suggest the potential for larger-scale attempts with appropriate hardware distribution strategies.

Objectives

All objectives of the project were met and done according to the set deadline. While relatively modest goals it was felt that initial estimates were overestimated for a large portion of the planning and preparation phases with model creation (as to be expected) taking up the largest amount of time. Overall, the planned time estimates did not affect the project drastically due to the agile nature of the project allowing for flexibility in the creation.

It is also a reflection of the success of the project that all initial result-based objectives were met.

7.2 Future improvements

There are several areas where the project could be expanded, the following provides some of the concepts that were considered on development.

1. **Predicting close price over x days**

   - **Rationale:**
     - In conventional trading environments, the concept of long positions is often associated with the expectation of higher returns.
     - Previous attempts in this area have fallen short (see Chapter 2), indicating a need for further exploration.
   - This idea expands the current understanding of trading models and strategies.

2. **Exploit KDB+ for data collection**

   - **Rationale:**
     - KDB+ (Kx Systems 2003) is a database handling tool that simplifies the process of working with large amounts of historical data.
     - Most banks and funds rely on KDB (Q) for data management and analysis (Insights 2024).
   - Leveraging KDB+ would facilitate efficient data collection and analysis, enhancing the accuracy and robustness of the models.

3. **Expand coin set**

   - **Rationale:**
     - Currently, only a limited number of coins are being considered in the analysis.
     - Exploring a wider range of coins, including smaller and potentially more volatile ones, could lead to discovering new insights and opportunities.
   - Diversifying the coin set would enable a more comprehensive understanding of the cryptocurrency market dynamics.

7.3 Personal Development

Throughout the project, a number of skills were acquired, and many lessons were learnt. The task at hand requires a range of knowledge from basic financial concepts to contemporary machine learning tools, as a result, the quick and effective analysis of papers, articles and raw data was vital to the success of this project. In turn, the initial progression was halted due to wasting time on irrelevant or outdated works and forced an updated approach in which resources were skimmed and then further analysed dependent on their potential significance to the project.

On a technical level, there are countless skills which have been acquired, especially in the context of using Python for data handling and model creation. These tools in turn helped speed up much of the project management process (as can be inferred from the Gantt Chart and the changes made) and prove the rising power of machine learning tool automation.
It is hoped that this refined skill set will become useful for future works and can be applied to a range of applications.

7.4 Closing Remarks

As of writing the importance of cryptocurrency can't be stressed more with roughly 10% of all UK adults holding money in coins and close to 70% stating they are likely to keep investing (HMRC 2022). This project looks to make sense of current cryptocurrency markets and answers questions on viability, scalability and optimisation which show a range of interesting results. However, the project has also created some further questions such as 'How will funds fuel quantitative approaches with rising hardware costs?' or 'Is it possible to create a recession-prone model?'.
It is hoped that further research extends upon this effort and looks to answer some of these ever-challenging questions with the abundance of contemporary machine-learning techniques.

"AI allows people to utilize this technology to do things they never imagined that they could do before" - *Neil Jacobstien*

The results of this project with the research conducted suggest we are not too far from a world where Wall Street is powered by GPUs rather than traders.

REFERENCES CITED

Akkaya, B. and N. Çolakoğlu, *Comparison of Multi-class Classification Algorithms on Early Diagnosis of Heart Diseases.*

al, Z. et, *Attention-based bidirectional long short-term memory networks for relation classification*, Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 207–212.

Apache, *Documentation*, Brand, URL: https://opennlp.apache.org/docs/, Accessed: 18 April 2024.

Ashraf, S., E. G. S. Félix, and Z. Serrasqueiro, *Do traditional financial distress prediction models predict the early warning signs of financial distress?*, Journal of Risk and Financial Management **12**, no. 2, 55.

Bergstra, J. et al., *A Tutorial on Hyperparameter Optimization for Machine Learning Algorithms*, arXiv preprint arXiv:1308.0971.

Board, F. S., *Financial stability implications from fintech: Supervisory and regulatory issues that merit authorities' attention*, Financial Stability Board.

Bollen, J., H. Mao, and X. Zeng, *Twitter mood predicts the stock market*, J. Comput. Sci. **2**, no. 1, 1–8.

Breiman, L., *Random forests*, Machine learning **45**, 5–32.

Cer, D. et al., *Universal Sentence Encoder*, arXiv preprint arXiv:1803.11175.

Chandra, R. and M. Zhang, *Cooperative coevolution of Elman recurrent neural networks for chaotic time series prediction*, Neurocomputing **86**, 116–123.

Chawla, N. V. et al., *SMOTE: Synthetic Minority Over-sampling Technique*, J. Artificial Intelligence Res. **16**, 321–357.

Chen, K., Y. Zhou, and F. Dai, *A LSTM-Based Method for Stock Returns Prediction: A Case Study of China Stock Market*, 2015 IEEE International Conference on Big Data (Big Data), IEEE, 2823–2824.

Chen, T. and C. Guestrin, *XGBoost: A Scalable Tree Boosting System*, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, ACM, DOI: 10.1145/2939672.2939785.

Cohen, N., *Unlocking the future: How hedge funds are shaping the crypto market*, URL: https://www.nasdaq.com/articles/unlocking-the-future-how-hedge-funds-are-shaping-the-crypto-market.

CoinGecko, *Crypto API documentation*, CoinGecko, URL: https://www.coingecko.com/api/documentation, Accessed: 18 April 2024.

Collins, B., *Death by API: Reddit joins Twitter in pricing out apps*, URL: https://www.forbes.com/sites/barrycollins/2023/06/01/death-by-api-reddit-joins-twitter-in-pricing-out-apps/.

Cortes, C. and V. Vapnik, *Support-vector networks*, Mach. Learn. **20**, 273–297.

Cover, T. and P. Hart, *Nearest neighbor pattern classification*, IEEE Transactions on Information Theory **13**, no. 1, 21–27.

CryptoPanic, *If you're going to panic, panic early.* CryptoPanic, URL: https://cryptopanic.com/developers/api/, Accessed: 18 April 2024.

D'Amato, V., S. Levantesi, and G. Piscopo, *Deep learning in predicting cryptocurrency volatility*, Physica A: Statistical Mechanics and its Applications **596**, 127158.

Devansh, *Why You Should Implement Evolutionary Algorithms in Your Machine Learning Projects*, URL: https://medium.com/mlearning-ai/why-you-should-implement-evolutionary-algorithms-in-your-machine-learning-projects-ee386edb4ecc, Medium.

Devlin, J. et al., *Bert: Pre-training of deep bidirectional transformers for language understanding*, arXiv preprint arXiv:1810.04805.

Edwards, R. D. and J. Magee, *Technical Analysis of Stock Trends*, 1948th ed., New York Institute of Finance, New York.

Elliott, R. J. and P. E. Kopp, *Mathematics of financial markets*, Springer Finance.

Evans, J. R., *38% of investors have lost more in crypto than made it*, LendingTree.

Fortin, F.-A. et al., *DEAP: Evolutionary Algorithms Made Easy*, The Journal of Machine Learning Research **13**, no. 1, 2171–2175.

Gamboa, J., *Deep learning for time-series analysis*, arXiv preprint, eprint: 1701.01887.

Gillis, A., *What is lemmatization?: Definition from TechTarget, Enterprise AI*, URL: https://www.techtarget.com/searchenterpriseai/definition/lemmatization.

Google, *True/False Positive/Negative*, URL: https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative, 15/12/23.

Graham, B. and D. Dodd, *Security Analysis*, McGraw-Hill.

Guardian, T., *The Guardian*, URL: https://open-platform.theguardian.com/documentation/, Accessed: 18 April 2024.

Guggilla, C., T. Miller, and I. Gurevych, *CNN- and LSTM-based Claim Classification in Online User Comments*, 2740–2751.

Gyol, S., *Renaissance Technologies Returns, AUM, CEO and Top Energy Stock Picks*, URL: https://finance.yahoo.com/news/renaissance-technologies-returns-aum-ceo-145524029.html.

Haan, K., *How businesses are using Artificial Intelligence in 2023*, URL: https://www.forbes.com/advisor/business/software/ai-in-business/.

Harris, Z. S., *Distributional structure*, Word.

Heaton, J., N. Polson, and J. Witte, *Deep learning in finance*, arXiv preprint, eprint: 1602.06561.

Himanshu, *Activation functions: SIGMOID, Relu, leaky relu and Softmax Basics for neural networks and deep...* Available at https://himanshuxd.medium.com/activation-functions-sigmoid-relu-leaky-relu-and-softmax-basics-for-neural-networks-and-deep-8d9c70eed91e.

HMRC, *Individuals Holding Cryptoassets: Uptake and understanding*, GOV.UK.

Hochreiter, S. and J. Schmidhuber, *Long Short-Term Memory*, Neural Comput. **9**, no. 8, 1735–1780, DOI: 10.1162/neco.1997.9.8.1735.

Hsu, C.-W., C.-C. Chang, and C.-J. Lin, *A practical guide to support vector classification*, Technical Report, Available at: https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

Insights, H., *Companies using KDB+, market share, customers and competitors*, Available at: https://discovery.hgdata.com/product/kdb-plus.

Jha, A., *Vectorization Techniques in NLP [Guide]*, https://neptune.ai/blog/vectorization-techniques-in-nlp-guide.

Joel, *What is the computational complexity of hyper-parameter tuning using grid search?*, https://stackoverflow.com/questions/57535932/what-is-the-computational-complexity-of-hyper-parameter-tuning-using-grid-seach, Stack Overflow.

Jordan, M., *Serial order: A parallel distributed processing approach*, Advances in Psychology **121**, 471–495, DOI: 10.1016/S0166-4115(97)80111-2.

Kx Systems, *KDB+*, Kx Systems, New York, NY.

Lamon, C., E. Nielsen, and E. Redondo, *Cryptocurrency Price Prediction Using News and Social Media Sentiment*, SMU Data Sci. Rev **1**, no. 3, 1–22.

Majaski, C., *Fundamental vs. technical analysis: What's the difference?*, URL: https://www.investopedia.com/ask/answers/difference-between-fundamental-and-technical-analysis/.

Menhour, I., M. B. Abidine, and B. Fergani, *A New Framework Using PCA, LDA and KNN-SVM to Activity Recognition Based SmartPhone's Sensors*, 2018 6th International Conference on Multimedia Computing and Systems (ICMCS), 1–5, DOI: 10.1109/ICMCS.2018.8525987.

Mikolov, T. et al., *Distributed representations of words and phrases and their compositionality*, Advances in neural information processing systems **26**, 3111–3119.

Mishev, K. et al., *Evaluation of sentiment analysis in finance: from lexicons to transformers*, IEEE Access **8**, 131662–131682.

Mittal, A. and A. oel, *Stock prediction using Twitter sentiment analysis*, **15**, 2352.

Mohan, S. et al., *Stock price prediction using news sentiment analysis*, 2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService), IEEE, 205–208.

Moore, D. S., G. P. McCabe, and B. A. Craig, *Introduction to the Practice of Statistics*, WH Freeman.

Morrow, A., *A crypto crash wiped out $1 trillion this week. Here's what happened.* URL: https://www.cnn.com/2021/05/22/investing/crypto-crash-bitcoin-regulation/index.html.

Pardoe, I., *4.6 - normal probability plot of residuals: Stat 501.*

Park, C. H. and S. H. Irwin, *What do we know about the profitability of technical analysis?*, Journal of Economic Surveys **21**, no. 4, 786–826.

Pawlak, Z., *Rough sets*, International Journal of Computer Information Sciences **11**, 341–356.

Pennington, J., R. Socher, and C. D. Manning, *Glove: Global vectors for word representation*, Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 1532–1543.

Porter, M., *An algorithm for suffix stripping*, Program **14**, no. 3, 130–137.

Probst, P., M. N. Wright, and A.-L. Boulesteix, *Hyperparameters and tuning strategies for random forest*, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **9**, no. 3, e1301.

Python, *Multiprocessing - process-based parallelism.*

Ruggiero, M., *Rules are made to be traded*, AI in Finance Fall, 35–40.

Sarra, D., *How to interpret smape just like Mape*, Medium.

Schulz, M. and M. Matthies, *Artificial neural networks for modeling time series of beach litter in the southern North Sea*, Marine Environmental Research **98**, 14–20.

Seng, D. and J. R. Hancock, *Fundamental analysis and the prediction of earnings*, International Journal of Business and Management **7**, no. 3, 32.

Singh, R. and S. Srivastava, *Stock prediction using deep learning*, Multimedia Tools and Applications **76**, 18569–18584.

Sparck Jones, K., *Statistical interpretation of tf-idf for term weighting in information retrieval*, Journal of documentation **28**, no. 1, 11–21.

Spez, *Addressing the community about changes to our API*, URL: https://www.reddit.com/r/reddit/comments/145bram/addressing_the_community_about_changes_to_our_api/, Accessed: 18 April 2024.

Stewart, L. M., *ProfitsBot_ V0_ OLLM*, https://github.com/getorca/ProfitsBot_V0_OLLM, GitHub repository.

Tambe, N., *Why Is The Crypto Market Down In October 2023?*, Forbes Advisor INDIA, URL: https://www.forbes.com/advisor/in/investing/cryptocurrency/why-crypto-market-is-down/.

Tay, F. E. and L. Shen, *Economic and financial prediction using rough sets model*, European J. Oper. Res. **141**, no. 3, 641–659.

TensorFlow, *Tf.keras.layers.LSTM : tensorflow V2.14.0*, TensorFlow.

———, *Tf.keras.optimizers.Adam: tensorflow V2.14.0*, https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam.

———, *tf.keras.preprocessing.sequence.TimeseriesGenerator*, https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/sequence/TimeseriesGenerator.

Tidy, J. and T. Geoghegan, *Who is Sam Bankman-fried?*, URL: https://www.bbc.co.uk/news/business-66990173.

Triple-A, *Cryptocurrency Ownership Data – Triple-A (2023)*, URL: https://triple-a.io/crypto-ownership-data/.

Vega, N., *More than 1 in 3 cryptocurrency investors know little to nothing about it, survey finds*, URL: https://www.cnbc.com/2021/03/04/survey-finds-one-third-of-crypto-buyers-dont-know-what-theyre-doing.html.

Wong, T. and P. Yeh, *Reliable accuracy estimates from k-fold cross validation*, IEEE Transactions on Knowledge and Data Engineering **32**, no. 8, 1586–1594.

Yates, T., *Quants: The Rocket Scientists of Wall Street*, Investopedia, URL: https://www.investopedia.com/articles/financialcareers/08/quants-quantitative-analyst.asp.

Yu, Z., *Stock Prediction Model Based on Machine Learning*, https://rpubs.com/ZHANGYU/StockPrediction.

Zhang, L., S. Wang, and B. Liu, *Deep Learning for Sentiment Analysis: A Survey*, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **8**, no. 4, e1253.