Bitcoin Prediction using Machine Learning and Time Series Models

Student

Andrei Bogdan Nedelcu

al415019@uji.es

Supervisor

Enrique Salvador Aragó

Degree in Finance and Accounting

2024 – 2025

Abstract

This work provides an in-depth analysis of Bitcoin return prediction using machine learning. We explore a variety of techniques, including natural language processing (NLP), GARCH, and Google Trend to predict the price of Bitcoin, discuss the challenges and opportunities of using machine learning for price prediction, and provide a detailed overview of the algorithm used. We also look at and analyse the efficiency of the algorithm and discuss potential improvements that could be made. Finally, we conclude with a summary of the key points discussed and provide some final thoughts on the topic.

Keywords: Bitcoin, Machine Learning, Time Series, GARCH, NLP

Contents

List of Tables

List of Figures

List of Algorithms

1 Introduction

According to Gunarso & Stephanie (2022) “Cryptocurrency is electronic-based money or virtual currency that exists only on the internet, that may or may not have intrinsic value backed by an institution, and that uses cryptography to secure the generation, transaction, and storage of its value”. Therefore, the increase in the popularity of Machine Learning and Cryptocurrencies have raised the interest of many researchers in the field of quantitative finance. In recent years, we have seen an increase in research focused on statistical models applied in price prediction and volatility. As one of the most volatile assets, Bitcoin grew into the spotlight of the financial world and researchers alike.

This thesis presents a comprehensive study of the development of a framework for the prediction of Bitcoin price using machine learning techniques, with a focus on time series analysis. This framework analyses the historical price of Bitcoin, the sentiment of popular news articles using FinBERT, the volatility of Bitcoin using Generalised AutoRegressive Conditional Heteroskedasticity (GARCH) model and the Google Trends data. The purpose of this thesis is to develop a robust and accurate model for creating profitable trading strategies based on the prediction of the framework.

Therefore, this study is motivated by the significant increase in the number of journals and articles published in the field of price prediction of cryptocurrencies with machine learning. Developing a framework that accurately predicts the price of Bitcoin is an inviting idea for a robust trading strategy. The algorithm developed in this study was designed to be used to trade Bitcoin, with the goal of achieving a statistical advantage over the crypto market.

This paper will focus on Bitcoin prediction using machine learning and time series models by first going through the analysis of the literature related to the topic, with a focus on novel frameworks and models that have been developed in the field of price prediction of cryptocurrencies using machine learning and Natural Language Processing (NLP) algorithms. Then, it presents a brief history of cryptocurrencies and Bitcoin, before moving into the process of data collection and preprocessing, which is elaborated with a focus on the type of data used, the sources of data, and the preprocessing steps taken to prepare the data for analysis. Moving forward, a detailed presentation of the methodology will explain where everything fits in the framework and how it interacts with the machine learning model.

Lastly, the results of the research will be detailed, including how the model performed and what can be done better. It will also include a trading strategy and a backtest of said strategy. This backtesting showed a high performance of the model, achieving a 79.83% return, 3.49 Sharpe ratio and 9.74% drawdown in an environment without costs. While including transaction costs and optimising the features of the model, it was exchanged the high return for lower risk, achieving a 47.11% return, a Sharpe ratio of 3.5 and a maximum drawdown of 6.01%. It concludes that the addition of a size limit and other mitigating risk variables can improve the performance and various metrics of the model.

2 Literature Review

A considerable number of studies, encompassing both older and more recent research, have made reference to cryptocurrencies and sentiment analysis, thereby establishing a foundation for the present study. On this section the focus will be on analysing the current literature on Cryptocurrency price prediction using machine learning and sentiment analysis. The primary focus will be on the methodologies used in the machine learning field since 1997, including the most recent breakthroughs. Diving deeper, we will follow the focus line to the sentiment analysis for crypto price prediction, detailing the importance of the preprocessing and feature engineering. The review will also highlight the key findings and the limitations and challenges faced by researchers in this field.

2.1 Cryptocurrency Price Prediction using Machine Learning

Machine learning in Cryptocurrency sphere became a hot topic in recent years. While crypto markets are known for high volatility, deployment of machine learning models can help understand price fluctuation. The backbone of this field are the linear, non-linear and sequential models (Gurgul et al. 2023). Starting by the simplest method, the linear models are based on the assumption that the input and output have a linear correlation. One of the most common linear model that has been used in the field is the Autoregressive Integrated Moving Average (ARIMA) model. Also, Ordinary Least Squares (OLS) regression having the simplest form might be the most used linear model in the field. To advance, non-linear models capture the non-linear relationships between the input and output variables. The most common non-linear models are Support Vector Machines (SVM), Decision Trees (DT) and Random Forests (RF). Furthermore, sequential models are utilised in order to capture temporal dependencies within the data. These models are particularly useful for time series data, where the order of the data points is of consequence.

Having said that, most Machine Learning models are based on the above-mentioned methods. Recurrent Neural Networks (RNN) are in fact a sequential model, which are designed to process sequences of data (Schmidt 2019). In this case, the sequence of data is of foremost importance because we wil be working with time series data. On one thing that RNNs are failing is the vanishing gradient problem, which means that the model is struggling to learn long term dependencies. Hochreiter & Schmidhuber (1997) came with a solution to this inconvenience, by introducing the Long Short Term Memory (LSTM) model. This model comes with a cell state that can retain information for prolonged periods of time. Even though LSTM model is an improvement compared to RNN, it still came with computational cost. Cho et al. (2014) introduced the Gated Recurrent Unit (GRU) model, which simplifies the LSTM model by combining the forget and input gates into a single update gate. Consequently, GRU has fewer parameters than LSTM, which makes it computationally more efficient.

In 2016 Chen & Guestrin (2016) introduces a new model called XGBoost, this algo-

rithm under discussion implements a decision tree ensemble method that is based on the gradient boosting framework. Its popularity is due to the versatility, efficiency and performance. This particular model is the most frequently employed for large datasets that are characterised by both a structured manner and a high number of features.

Rizwan et al. (2019) are using different types of Machine Learning for Bitcoin price prediction and contrasting the results. The first model chosen was RNN, where this model prioritises the sequence of information given which is the Bitcoin price and puts more emphasis on the recent data. Next model deployed was LSTM, it should be noted that this has numerous parallels with the RNN. However, it possesses a memory cell with the capacity to retain information over an extended time period. The final model employed in this study was GRU, which is a derived version of LSTM as previously stated. For this contrast, ARIMA models were utilised as a benchmark. The results of the study are presented in Table 1.

Models	Accuracy	Error Rate
RNN	85.40%	14.60%
LSTM	92.30%	7.70%
ARIMA	91.00%	9.00%
GRU	94.70%	5.30%

Table 1: Comparison of Machine Learning Models (Source: Rizwan et al. (2019))

From this study, it can be observed that the GRU model outperforms all the other models. GRU is right in between RNN and LSTM in regards of complexity and parameters. It is a simpler model than LSTM and yet more complex than RNN (see Table 2).

Feature	Traditional RNN	LSTM	GRU
Gates:	None	3 gates: Input, Forget, Output	2 gates: Update, Reset
Memory Mechanism:	Simple hidden state	Cell state + hidden state	Single hidden state
Complexity:	Low	High	Moderate
Parameters:	Fewest	Most	Fewer than LSTM

Table 2: Features of models utilised by Rizwan et al. (2019). (Source: *Rizwan et al. (2019)*)

Mohammadjafari (2024) arrived at the same conclusion after comparing GRU and LSTM models. A comparison of Mean Squared Error (MSE) values between the two models demonstrated that the GRU model shows superior performance in comparison to the LSTM model. Furthermore, the GRU model was determined to be more efficient with regard to time by 30%. In terms of MSE values, GRU achieved a value of 4.67 and LSTM achieved a value of 6.25.

A recent study introduces a novel decision framework for which can predict Bitcoin price with minimal error. Din et al. (2025) proposed a decision ensemble framework, one that utilises a combination of Machine Learning models to predict Bitcoin price. It leverages Bidirectional LSTM, eXtreme Gradient Boosting (XGBoost) and an error reciprocal weighting scheme. This paper shows that the BiLSTM model outperforms LSTM and that

XGBoost surpasses LSTM and BiLSTM in terms of accuracy. For this framework, the root mean square error was 106.41.

Jiang (2023) came as well with a novel approach to predict Bitcoin price. He proposed a hybrid model that combines GRU and XGBoost. The input feature for GRU is the historical Bitcoin price, it then extracts the long and short them dependencies. XGBoost comes in afterwards, it takes the output of GRU and classifies the features and predicts the price. This hybrid model achieved a root mean square error of around 0.025.

As indicated in the literature, there is a growing interest in combination of different models. This shift in focus suggests that researchers are recognising the limitations of individual models and seek to optimize their performance benefiting from the computational power of different models. At this moment, the focus is on hybrid models approach to improve the performance of the models. By assigning greater importance to the areas in which a given model excels in comparison with competing models, the usage of hybrid models becomes a viable strategy. This approach enables the capture of a more comprehensive set of features while avoiding the limitations inherent in employing a single model.

2.2 Sentiment Analysis in Cryptocurrency Price Prediction

While we saw in the previous section that some Machine Learning models are great at predicting Bitcoin price, using sentiment analysis can improve the accuracy of the models. Sentiment analysis has been used for gauging public opinions and sentiments on various sectors and topics. It is a Natural Language Processing (NLP) algorithm that extracts information from text data and classifies it as a positive, negative or neutral and given a sentiment score. Big companies started to adapt since early stages of sentiment analysis algorithms to interpret reviews and also track the sentiment of the company and their products (Indurkhya & Damerau 2010).

In order to get an output from the sentiment analysis, the algorithm needs to pass through a few steps. First step is data collection, next step is data processing which includes pre-processing, feature extraction and feature selection (Birjali et al. 2021). Lastly, we get our output which is the sentiment score. There are several types of sentiment analysis, early days of sentiment analysis were based on Lexicon-based methods. These methods work on a pre-defined dictionary of words and phrases and have an already associated sentiment score. Newer methods are based on Machine Learning and Deep Learning, using algorithms such as Naive Bayes, Support Vector Machines (SVM), Recurrent Neural Networks (RNN) and Long Short Term Memory (LSTM) (Birjali et al. 2021) to classify the data and obtain the sentiment score. By using Machine Learning and/or Deep Learning reduces significantly the manual work for feature engineering (Gurgul et al. 2023).

Kim et al. (2023) shows that not only that the sentiment analysis can be used to predict Bitcoin price, but also just by including NLP features in the model, an increase

in accuracy and an increase in profitability can be achieved with no change in volatility obtaining a competitive advantage within the market. The incorporation of a sentiment score into the model has been shown to result in superior performance in comparison to all other models that do not incorporate a sentiment score.

FinBERT is a pre-trained model derived from BERT, which is a transformer-based model that has been trained on financial corpus (Araci 2019). In a similar fashion, the company's financial benefits derived from NLP can also be applied to the domain of financial market analysis. It is vital to understand that in order to understand financial texts, we must first become acquainted with the specific terminology employed. This necessity has led to the emergence of FinBERT, a tool designed to facilitate the dissection of financial jargon and understanding of context. Yang et al. (2020) findings indicate that deployment of specific NLP models such as FinBERT surpasses the conventional BERT approach and places emphasis on the significance of domain-specific pre-training.

Hossain et al. (2024) proposed a model that employs FinBERT and BiLSTM. This model is closely aligned with the subject matter of this work. The process involves the evaluation of news and their sentiment, followed by the input of the resultant score to a machine learning algorithm for cryptocurrency price prediction. The purpose of this study was to evaluate the efficacy of a novel framework, FinBERT-BiLSTM, in predicting the bitcoin price. The experimental results demonstrated an accuracy of 98.07%, which can be considered a highly successful outcome. The authors also employed a trading strategy based on the predictions that his model generated. The results were also highly successful, starting with an initial amount of 100.000$ the model achieved a return of 41.233,62$ in 88 days. These results were achieved with a model that was predicting one day ahead. They also compared these results with a model that was predicting 30 days ahead, the model had a high accuracy but no trading strategy was implemented.

3 Cryptocurrency

This section is dedicated to the explanation of the fundamentals of cryptocurrency. The characteristics that give value to an intangible asset such as crypto will be outlined, and the rationale behind the selection of Bitcoin as the study case will be elucidated. The section will also provide an overview of the historical evolution of cryptocurrencies and the key features that distinguish them.

3.1 Characteristics of Cryptocurrency

The concept of Cryptocurrency or Digital Token has its beginning in the 80s as technology started to evolve and various cryptographers were more and more interested in digitalisation of their work. An example that resembles the cryptocurrencies that we have today is the eCash or Digicash. It was created by David Chaum, a cryptographer, in 1983 (Frankenfield 2021). It used an in-house developed software to send and receive money on the internet and also to withdraw money from ATMs. The users had a designated encryption key that made everything possible. Unfortunately, not even a decade later the company went bankrupt a few years before the dotcom bubble. Although the company failed, it's technology was a stepping stone for the world of cryptocurrencies and electronic payments.

The crypto space started with Bitcoin in 2009, nonetheless today there are thousands of cryptocurrencies. We will see which are the biggest ones but right now our focus will be the characteristics of cryptocurrencies.

One of the major overlap of all cryptocurrencies is that they run on a public ledger also called blockchain. It means that there is no central authority that manages the record of the transactions, instead the transactions are recorded on a peer-to-peer system. Anybody can see the transactions made but not all of the details are visible. You can see the amount and the time of the transactions but not exactly who made them and to whom. For anything that involves cryptocurrencies its needed a digital wallet and said wallet has a unique address assigned and that address is the only thing you can see that is related to the information about a user. Because of this feature, cryptocurrencies are considered to be anonymous and secure.

In the case of the biggest cryptocurrencies, they are created by a process called mining. This process involves the computational power of your computer to solve complex mathematical problems and as a reward you receive a certain amount of the cryptocurrency.

For the storage of the cryptocurrencies, there are only two types of wallets: cold wallets and hot wallets. The differentiating factor between the two is the security. Cold wallets are the most secure ones because they store your cryptocurrencies on a physical device that does not have access to the internet. Meanwhile, hot wallets are stored on a device that has access to the internet and are more vulnerable to cyber attacks.

The crypto sphere has been growing at an exponential rate, at the moment of writing this study, there are around 39.000 cryptocurrencies which are traded across a multitude of exchanges. With an estimated market cap of 2.9 trillion euros, the biggest cryptocurrencies are Bitcoin, Ethereum and Stablecoins such as USDT and USDC. The market cap of Bitcoin is 1.63 trillion euros, which represents about 56% of the total crypto market cap. Ethereum with about 260 billion euros represents only 9% of the total market cap, being the second biggest cryptocurrency.

3.2 Bitcoin

The first cryptocurrency, Bitcoin, was created in 2009 by an unknown person or group of people under the pseudonym Satoshi Nakamoto. It currently holds the distinction of being both the most widely used and the most valuable cryptocurrency globally. It is also the most traded cryptocurrency in the world. The fundamental principle underpinning Bitcoin is that of a decentralised digital currency that functions on a peer-to-peer basis. Its decentralised nature is a key characteristic, as it is not subject to the control of any government or financial institution. This is a key factor in reducing transaction costs (Alstyne 2014). It is a digital currency that can be used to purchase goods and services online.

The total supply of bitcoins is limited to 21 million units. This has the effect of maintaining the supply of bitcoins in a fixed quantity, thereby limiting the circulation of units above 21 million. This limited supply is a primary factor contributing to the high value attributed to bitcoin, and it is also recognised for its volatility. The price of bitcoin has been known to fluctuate substantially over short periods of time. This characteristic has rendered Bitcoin a popular investment medium for traders and speculators. Furthermore, Bitcoin is recognised for its security. The network operates through a verification process referred to as 'proof of work', which ensures its security. The privacy that bitcoin offers is considered a significant benefit, which is a major factor in its popularity. Transactions executed on the bitcoin network are characterised by their pseudonymous nature, whereby the identities of both the sender and recipient of the transaction remain concealed. This ensures that while transactions are recorded on a public ledger, the identities of the parties involved remain undisclosed.

For the purposes of this study, Bitcoin was selected as the primary cryptocurrency for analysis due to its status as the most widely used and valuable cryptocurrency. The historical data for Bitcoin is also readily available, making it an ideal candidate for analysis. Moreover, the popularity of Bitcoin has given rise to extensive discourse in the public sphere, which is why there is interest in studying its price prediction using sentiment analysis.

The present study will focus on the prediction of Bitcoin prices using machine learning and sentiment analysis. The objective of this study is to develop a model that can accurately predict the price of Bitcoin, thereby enabling traders and investors to make informed decisions regarding their investments.

4 Data Collection

For this analysis, the data used is the crucial part of the algorithm. The methodology approached is a mixture of quantitative and qualitative data. We will start first with the qualitative data and then move on to the quantitative data. The qualitative data are the news articles which later will be transformed into a quantitative variable. The quantitative data are the historical price data of Bitcoin and the Google search trends data. All of the combined will be used as descriptive variables in the model. The focus of this analysis will be on the qualitative data, as it is the data that will describe the price return of Bitcoin and the hardest to collect and interpret. A different important aspect of the data is the quality, because the accuracy of the model will depend on it, and the more data we have, the better the model will be. Programming was needed for the data collection, it will be explained in the following paragraphs.

4.1 Quantitative Data

The sample data ranges from 01-August-2024 to 06-March-2025. For this analysis, historical price data for Bitcoin were collected from Yahoo Finance as there is an open source API (Application Programming Interface). To call the API, it was needed to code a Python file that uses the open source library called yfinance to collect the data. This Python file will be used throughout the analysis presented.

In addition, Google search trends data for Bitcoin were used with the help of their Web console. Their Web console is a website that allows you to access Google Trends data, looking for what are the latest search trends, or creating a specific query. Google Trends data measures how often a search term was looked for on Google. We can introduce our keyword, which is Bitcoin, and the console will return the trend data for said keyword. The data that we receive are normalised with a value between 0 and 100. Logically, the higher the value, the more popular the search term is.

4.2 Qualitative Data

To advance with the algorithm, news needed to be collected, for which were collected articles related to Bitcoin from major news outlets such as Reuters, Bloomberg and Yahoo Finance News using the NewsDatahub.com website and their API. NewsDatahub scrapes the news outlets for the latest news articles and provides a sorted JSON file with the details of the news articles. Only the date of publish, headline and description were stored. For this case, a bit more than ten thousand articles were collected.

To preprocess the news articles, they were sorted based on number of views and then the top 10 news articles were selected of that day. The headlines and the description were then combined into one text and the text was cleaned by removing any special characters, numbers, and stop words. This process is important due to the fact that the model is focussing on the sentiment of the news articles, and the more clean the text is, the better the understanding will be as it gives more weight to specific words and

wordings. In Section 5, it will be explained a bit more about how the sentiment score is given to the news articles. In this case, the data will have a value from -1 to 1.

4.3 Storage of the Data

The data was subsequently stored in a structured relational database, which was organised into columns with the following headers: Date, Close, Headlines, Sentiment, Google Trends, GARCH. The collection of the data can be in real time or in batch mode. In this case, the data was collected in a batch mode, meaning that it was collected at the end of the day and stored in the SQL Database mentioned earlier. The batch mode was selected as it is easier to implement and is more efficient in terms of time and resources. However, everything can be called in real time if needed. For example, if something happens during the day that affects the price of Bitcoin, the new price and all new headlines can be collected as they happen, and the model can be updated in real time.

A SQLite database is considered to be the optimal solution for this particular type of data storage, due to its ability to function independently of a server, its lightweight architecture, and its high query processing speed. This further on can be scaled for real-time trading server architecture.

4.4 Ethical Considerations

A further significant element that was given due consideration was the technical aspect of the data. In this instance, the data was sourced from open-source APIs and NewsDataHub.com and was collected in accordance with legal protocols. No personally identifiable information was procured, and no data that required user consent was collected. While scrapping the data is a good method to collect data, it also comes with ethical considerations such as putting pressure on the servers of the websites and potentially causing harm. This implications are important when collecting online data, it can also be considered as a form of cyber attack.

The API used offered a free tier account which gives a limited API calls per day, in this case there were 100 calls per day. It was called every 3 seconds to not burden the server.

5 Methodology

The subsequent section will present a comprehensive overview of the methodological approach employed in the present study. The methodology is divided into four main sections: Normalisation of the data, Natural Language processing, GARCH and Google Trend Data, and lastly machine learning model. The subsequent section will provide an overview of the data preprocessing steps taken to prepare the data for the machine learning model. Finally, the machine learning model section will provide an overview of the ML model that has been utilised for this work.

5.1 Normalisation of the data

In order to predict the price of Bitcoin, the historical price data needs to be transformed into a log return series. This log transformation measures the percentage change in the price of Bitcoin over time. It also removes heteroscedasticity from the data meaning that it stabilises the variance and can make it easier for the machine learning to predict the price return for further dates. The log return is defined as follows:

$$R_t = \frac{Log(P_t)}{Log(P_{t-1})} \tag{1}$$

Where R_t is the log return of Bitcoin price at time t, and P_t is the closing price of Bitcoin at time t. For the case of the analysis, let P be the vector of log returns of Bitcoin defined as $P = [R_1, R_2, ..., R_{t-1}]$.

Bitcoin price can be very volatile and so it is quite important to transform the closing price into log return. It is also important to take in mind that is easier to work with log returns than with the closing price. Also log return is a better measure of the performance of an asset than the simple closing price as it is a continuous variable and it is not affected by the scale of the price for which is simpler to be addressed by the machine learning model. It will be the dependent variable in the model.

5.2 Natural Language Processing

To go further, all the news collected were transformed into a sentiment score using a Natural Language Processing algorithm (NLP). The NLP algorithm used is a pre-trained model built by training Bidirectional Encoder Representations from Transformers (BERT) using a financial corpus. BERT is a transformer-based model introduced by researchers at Google in October 2018 and is widely used in NLP tasks. For example, it is used in text classification, text generation or text prediction. We are using it in our daily lives when we use Google Search, Google Translate or Google Keyboard on our phones or tablets. To simplify, BERT is a model that takes a text, splits it into words and transforms each word into a vector and then uses those vector to predict the relation between words in the text by a cosine function. This train of actions is also called tokenization of the text. It is challenging for a computer to understand the sentiment of a text but it can be done by

helping the computer with associations between words and their meanings. The idea of vectorization can be seen in the Algorithm 1 pseudocode below.

Algorithm 1 Word similarity after tokenization

Variables: Let W_1 and W_2 be a set of words collected from the news articles and sim be the similarity between the words. Let L be a list of tuples containing the words and their similarity.

Require: W_1 , W_2

```
for W1 in words do
    for W2 in words do
        if W1 ≠ W2 then
            sim ← 1 − cosine(W1, W2)
            L ← (W1, W2, sim)
        end if
    end for
end for
```

As mentioned earlier, the BERT model used was trained on a large corpus of financial news articles to predict the sentiment of the news articles. The score is a measure of the sentiment of the news article. The sentiment score could be between -1 and 1, where:

- -1 represents a negative sentiment
- 0 represents a neutral sentiment
- 1 represents a positive sentiment

Those scores that were neutral but they had negative values were transformed into positive values. This transformation was done because as long as there are news about Bitcoin, the price will be influenced in some way and most of the time the price will increase because of the news and its popularity. The sentiment score will be used as an independent variable in the model and will be called S_i.

5.3 GARCH model and Google search trends data

It also includes Generalised AutoRegressive Conditional Heteroskedasticity (GARCH) model to predict the volatility of Bitcoin price. The GARCH model is a time series model that is commonly used to predict the volatility of financial assets because it takes into account the past volatility of the asset and also the squared residuals of the asset. It was considered because the bitcoin price exhibits periods of big swings and periods in which the price is relatively constant. It has been demonstrated the hypothesis that the GARCH model is a reliable estimate for Bitcoin price prediction, while also being reinforced by a number of studies that integrate it with Machine Learning (see Zahid et al. (2022), Arslan et al. (2025)). For the algorithm, the Arch python package was used to implement the GARCH model.

As we know, the price of Bitcoin is highly volatile, so the GARCH model is a good choice for predicting the volatility of Bitcoin price. The GARCH model is defined as follows:

$$\sigma_t^2 = \omega + \alpha\varepsilon_{t-1}^2 + \beta\sigma_{t-1}^2 \tag{2}$$

Where σ_t^2 is the volatility of Bitcoin price at time t, ω is the constant term, α and β are the coefficients, ε_{t-1}^2 is the squared residual at time t-1, and σ_{t-1}^2 is the volatility at time t-1.

Google trends data was used because it gauges the popularity of Bitcoin and its directly linked with the news articles. The Google search trends data measures how often a specific word or phrase was searched for on Google. Smuts (2019) shows in his paper that Google Trends has a predictive value in the case of Bitcoin and Ethereum. Gómez Martínez et al. (2022) also concluded that using AI and Google Trend for Bitcoin can create an edge in the market to create a profitable strategy. The Google search trends data was left as it was, with its value between 0 and 100.

The Google search trends data will be called G_i and the volatility of Bitcoin price will be called V_i.

5.4 Machine Learning Model

It can thus be concluded that the model will be trained with the log return of Bitcoin price as the dependent variable and the sentiment score of the news articles, the Google search trends data and the volatility of Bitcoin price as the independent variables. The model will be trained using a machine learning algorithm called XGBoost. This particular machine learning algorithm has been extensively utilised in numerous research studies and is deemed a suitable candidate for this project due to its capacity to manage substantial datasets with efficiency and expediency. The algorithm is founded upon the gradient boosting framework and finds application in both regression and classification problems.

The predicting function is defined as follows:

$$\hat{p}(x) = f(P, S_{i-1}, G_{i-1}, V_{i-1}) \tag{3}$$

where $\hat{p}(x)$ is the predicted log return of Bitcoin price at time i, P is the log return of Bitcoin price, S_{i-1} is the sentiment score of the news articles at time i-1, G_{i-1} is the Google search trends data at time i-1, and V_{i-1} is the volatility of Bitcoin price at time i-1.

In order to clarify Algorithm 2, line 1 is the input of the independent variables, given as X. Line 2 is the dependent variable, given as y. Line 3 is the splitting of the data into training and test data. The fifth line details the implementation of the XGBoost model, details the model fitting. The final lines shows the accuracy of the prediction of the model, hence the mean squared error of the model is denoted by line 7. Finally, line 8 indicates the root mean squared error of the model.

An additional code was utilised to optimise the model. The purpose of this additional code is to identify the most suitable hyperparameters for the model. The Grid Search al-

Algorithm 2 XGBoost algorithm

Input: Let P be the log return vector of Bitcoin price, S be the sentiment score of the news articles, G be the Google search trends data, V be the volatility of Bitcoin price. Where P be the dependent variable.

Output: Predicted log return price $\hat{p}$, Evaluation metrics MSE and $RMSE$.

Require: P, S, G, V

$X \leftarrow (S, G, V)$
$y \leftarrow P$
$X_{train}, X_{test}, y_{train}, y_{test} \leftarrow train\_test\_split(X, y, test\_size = 0.2)$
$model \leftarrow XGBRegressor()$
$model.fit(X_{train}, y_{train})$
$\hat{p} \leftarrow model.predict(X_{test})$
$mse \leftarrow mean\_squared\_error(y_{test}, \hat{p})$
$rmse \leftarrow np.sqrt(mse)$

gorithm 3 is a systematic approach to hyperparameter tuning that involves the evaluation of all possible combinations of hyperparameters to identify the optimal set of hyperparameters for the model. The algorithm is implemented using the GridSearchCV function from the scikit-learn library. The GridSearchCV function requires the following inputs: the estimator, the parameter grid, the cross-validation, and the scoring metric. The subsequent model fitting to the training data is then evaluated using the specified scoring metric. The identification of the optimal hyperparameters is then undertaken, and these are utilised in the training of the final model. The performance of the final model is then evaluated using the mean squared error metric on a set of test data, and the process is repeated until the model achieves a satisfactory level of performance. Following this evaluation, the model's predictions are stored in a results dataframe, where they can be analysed further. It will be used in the next section to evaluate the model.

5.5 Parametric Configuration of the Model

The parameters used for the training of this model are the the following 3:

max_depth	6	Controls the maximum depth of a tree
eta	0.01	Learning rate of each tree
subsample	1	Random sampling of training instances before each tree
colsample_bytree	1	Feature subsampling for each tree
eval_metric	rmse	Monitors the model performance using Root Mean Squared Error

Table 3: XGboost parameters (Source: *XGBoost Official documentation*)

The time to train the model is based on the computing power of the GPU, RAM and CPU[1] The size of the dataset utilised can also have an impact on training time. The training time can be reduced with a more powerful GPU and more RAM or by reducing

[1]The computer used for the training of this model has the following specifications: CPU: 6 cores, 12 threads, 3.2 GHz; RAM: 16 GB DDR4; GPU: 6 GB GDDR6.

Algorithm 3 Grid Search

Define parameter grid:
$param\_grid \leftarrow \{$
$max\_depth : [3, 6, 10]$,
$eta : [0.01, 0.1, 0.3]$,
$subsample : [0.6, 0.8, 1.0]$,
$colsample\_bytree : [0.6, 0.8, 1.0]$,
$n\_estimators : [100, 500, 1000]$
$\}$

Initialize XGBoost Regressor:
$xgb\_reg \leftarrow XGBRegressor(objective = \text{“}reg : squarederror\text{”}, seed = 42)$

Grid Search for best parameters:
$grid\_search \leftarrow GridSearchCV($
$estimator = xgb\_reg$,
$param\_grid = param\_grid$,
$cv = 3$,
$scoring = \text{“}neg\_mean\_squared\_error\text{”})$
$grid\_search.fit(X\_train\_extended, y\_train\_extended)$

Get best parameters:
$best\_params \leftarrow grid\_search.best\_params\_$
Print “Best parameters: ”, $best\_params$

Train model with best parameters:
$best\_xgb \leftarrow XGBRegressor(**best\_params)$
$best\_xgb.fit(X\_train\_extended, y\_train\_extended)$

Make predictions:
$y\_pred\_best \leftarrow best\_xgb.predict(X\_test\_extended)$

Evaluate model:
$mse\_best \leftarrow mean\_squared\_error(y\_test\_extended, y\_pred\_best)$
Print “Mean Squared Error with best parameters: ”, mse_best

Save predictions:
$results[X\_test\_extended.index, \text{“}predicted\_log\_return\_best\text{”}] \leftarrow y\_pred\_best$

the size of the dataset. In this case the approximate time needed for training was around 15 minutes.

5.6 Evaluation of the model

The algorithm will be evaluated mainly by the mean squared error (MSE) and the root mean squared error (RMSE). The formulas are defined as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(P_i - \hat{p}_i)^2 \tag{4}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(P_i - \hat{p}_i)^2} \tag{5}$$

Where:

- P_i is the actual Bitcoin price
- $\hat{p}_i$ is the predicted price
- n is the number of observations

The root mean squared error is a metric of choice in measuring the discrepancies between the values predicted by a statistical model and the observed values. The value of the RMSE provides clear insight into the accuracy of the model, allowing for a comprehensive evaluation of the discrepancy between predicted and observed outcomes. The RMSE has gained wide acceptance in scientific and mathematical communities due to its capacity to quantify the accuracy of models and assist in the refinement of data analysis techniques.

The model will be evaluated twice, once with the test data and once with the simulated walking forward validation. The walking forward technique is employed to evaluate a simulation of the model in a real world like scenario. In the course of evaluating the model using the test data, a cross-validation technique is used. Cross-validation is a statistical method that is utilised to estimate the proficiency of machine learning models. It is employed to assess the generalisability of the results of a statistical analysis to an independent dataset. The cross-validation technique is employed to evaluate the model by dividing the data into k subsets, where k is a user-defined number. The model is then trained on k-1 subsets, with the remaining subset being used for testing. This process is then repeated a specified number of times, with each individual subset being utilised as the designated test set on one occasion (Weber 2024). The final result is the average of all k results.

For the case of the simulated walking forward validation, the model was evaluated with a set of formulas, such as the Sharpe ratio, Sortino ratio, maximum drawdown and the return of the model. The Sharpe ratio is a measure of risk-adjusted return (Sharpe 1966), which is calculated as follows:

$$SR = \frac{E(R) - R_f}{\sigma} \tag{6}$$

Where:

- $E(R)$ is the expected return of the portfolio
- R_f is the risk-free rate
- σ is the standard deviation of the portfolio returns
- SR is the Sharpe ratio

The Sortino ratio is a variation of the Sharpe ratio, which is used to measure the risk-adjusted return of an investment (Sortino & van der Meer 1991). It helps differentiate cases of volatility, for the strategy implemented in this study, volatility is seen as positive, while downside volatility is seen as negative.

The Sortino ratio is calculated as follows:

$$SOR = \frac{E(R) - R_f}{\sigma_d} \tag{7}$$

Where:

- $E(R)$ is the expected return of the portfolio
- R_f is the risk-free rate
- σ_d is the standard deviation of the downside returns
- SOR is the Sortino ratio

The maximum drawdown is defined as the maximum loss from a peak to a trough of a portfolio, prior to attaining a new peak (Chekhlov et al. 2005). The maximum drawdown is important because it provides a way to measure how much the algorithm can lose before recovering. It is needed to mitigate the risk of the model and to evaluate the performance.

It is calculated as follows:

$$MDD = \frac{P_{max} - P_{min}}{P_{max}} \tag{8}$$

Where:

- P_{max} is the maximum price of the portfolio
- P_{min} is the minimum price of the portfolio
- MDD is the maximum drawdown

6 Empirical Results

This section provides the empirical results, addressing the questions posed in the introduction. The results are divided into the following segments: the first part focuses on the performance of the ML model, while the second part examines the performance in a simulated market. The analysis is based on the data collected and preprocessed as described in Section 5.

6.1 Performance of the ML Model

At first, the model was overfitting the training data. The model was trained on the training set, which consisted of 80% of the data, while the remaining 20% was used for testing. The model's performance was evaluated using root mean square error (RMSE) as the primary metric.

The model achieves a RMSE of 0.4767 on the train set and 0.6234 on the test set. The model's performance was further validated through a 10-fold cross-validation process. This resulted in an average root mean square error value of 0.7813 across all folds. Initially, the model exhibited overfitting, as evidenced by the low RMSE values. However, following the implementation of regularization techniques such as L1 and L2, a substantial enhancement in the model's performance was observed.

L1 regularization, otherwise referred to as Lasso regression, was utilised to reduce the amount of features and improve the interpretability of the model. This technique effectively eliminates irrelevant features, leading to a more efficient model. The L2 regularization, also referred to as Ridge regression, was implemented to address multicollinearity among the features, thereby enhancing the model's robustness (Moore & DeNero 2011)(Kolluri et al. 2020).

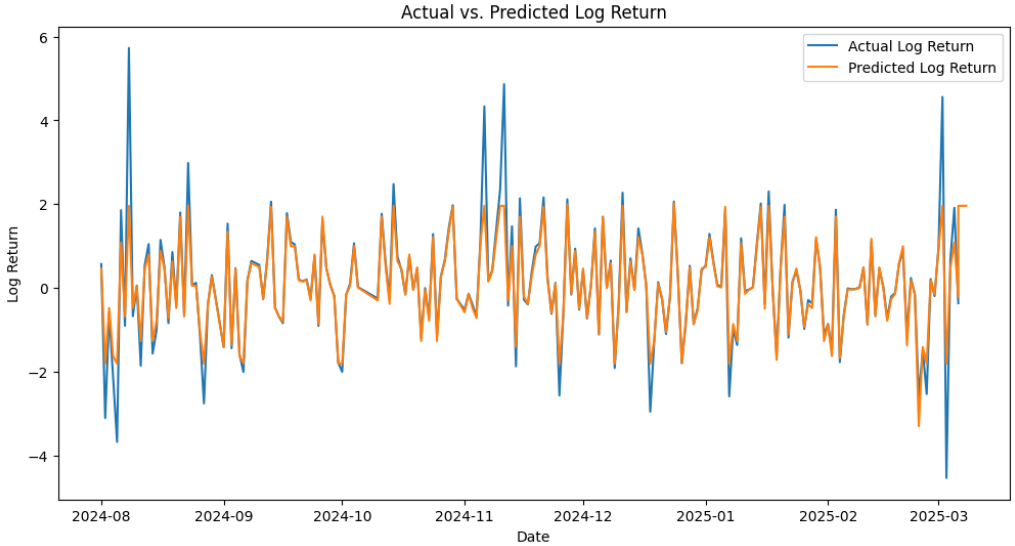

Figure 1: Performance of the ML model on the training and test sets. (Source: *Author*)

As shown in Figure 1, the constructed model has been shown to possess the capacity to predict fluctuations in the market price of Bitcoin with a reasonable degree of accuracy. Because of L1 and L2 regularisation, the model is generalising the unseen data. The prediction is closely aligned with the actual price, unless a sharp increase in price has been observed.

6.2 Simulated Market Performance

In this section, we will evaluate the performance of the model in a simulated market environment. A trading strategy was developed based on the predictions generated by the model. The strategy involves a moving average based on the next predicted prices. The moving average is calculated using the predicted prices for the next x days. If the predicted price is above the moving average, a buy signal is generated, and if it is below, a sell signal is generated. Also the strategy consists only on a buy and no short selling was implemented. The benchmark for the strategy is the buy and hold strategy, very common comparison in the literature. This type of comparison can be used to evaluate if the model is able to beat the market. It consists of buying the asset at the beginning of the period and holding it until the end of the period.

6.2.1 Simulated Market without Size, Cost and Slippage Limitations

For the simulation, the backtest was performed on different premises with different sets of parameters. The first backtest was performed without any size limitations, meaning that the model could buy with all the initial value of the portfolio which was set to $10.000. Also, no slippage and no fee were taken in count. In the case of Bitcoin, the slippage means that the price you want to execute your order is different from the price you actually get from the market. This is an occurring problem in the crypto market because of the high volatility and the low liquidity. The fees are what the crypto exchanges charge you for executing the order. Most common type of fee is the maker fee, expressed as a percentage of the total amount of your buying order at a specific price. In the next subsection, the details of the simulated market with size, cost and slippage limitations will be explained more thoroughly.

It can be observed in Figure 2 that the model was able to beat the benchmark, with a Sharpe ratio of 3.49 and a maximum drawdown of 9.74%. It achieved a return of 79.83% over the period, while the benchmark achieved a return of 37.64%. The Sortino ratio for this case was 6.89, showing that the model achieved high returns with low risk. It had a win rate of 70.73%, meaning that 70.73% of the trades were profitable. The average return per winning trade was 3.2%, while the average return per losing trade was -2.57%. The model was able to generate a total of 41 trades over the period, with an average holding period of 2 days. These results were achieved with a moving average of 4 days, meaning that the algorithm predicted 4 days ahead of the current day and calculated an average of those days. Without taking into account the size, cost and slippage, tests were performed to see the optimal moving average. The results are shown in Table 4.

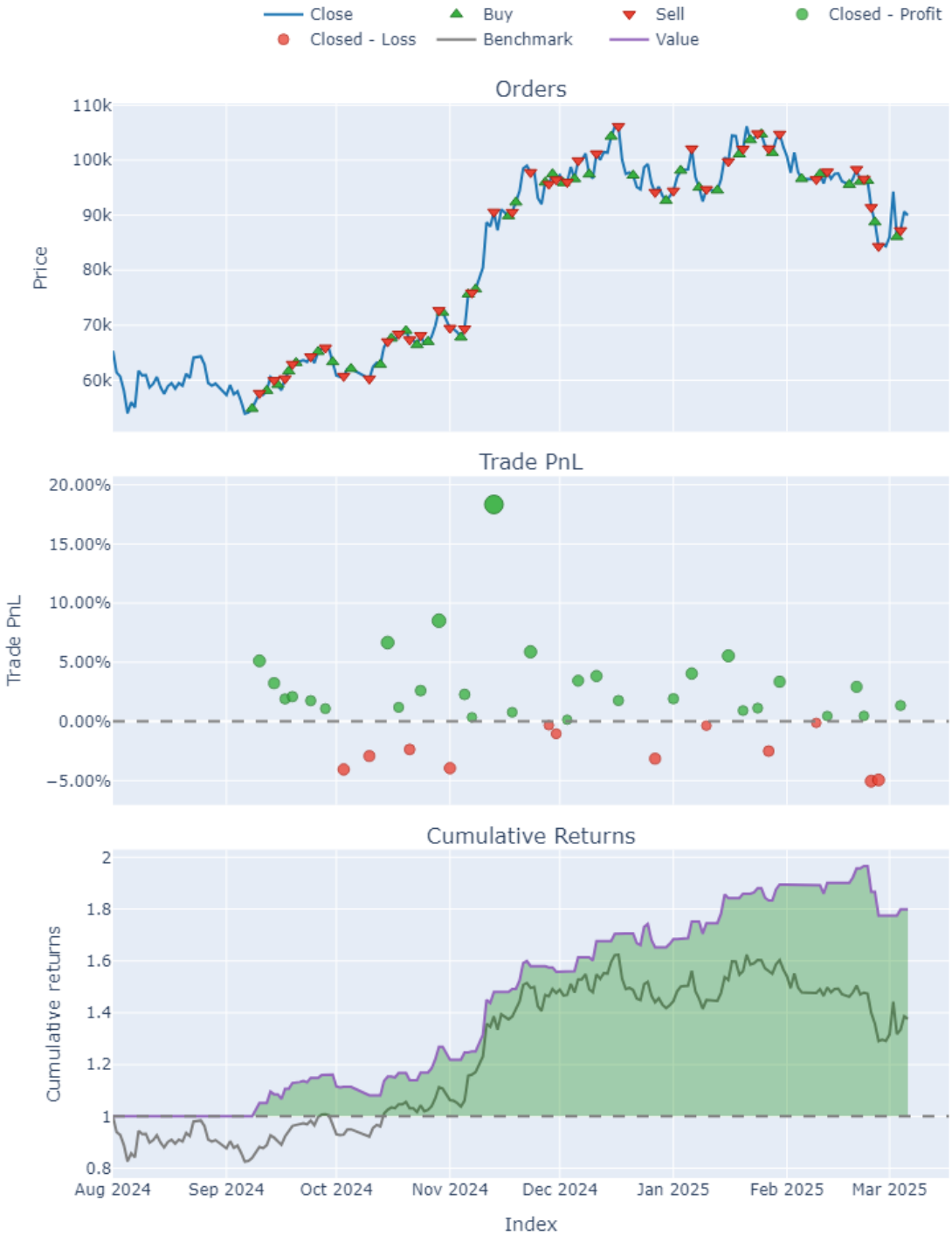

Figure 2: Performance of the simulated market without size, cost and slippage limitations. (Source: *Author*)

It can also be observed that the algorithm did not trade for the first month of the period, as it was not able to generate a buy signal. It can be caused by the fact that the algorithm was not identifying the trend pattern correctly, or that the volatility was not enough to generate the buy signal. On the other hand, when it could generate a buy signal, it was able to generate a higher return than the benchmark with a lower risk in almost all cases (see Table 4).

Features	MA2	MA3	MA4	MA5	MA6	MA7	MA8	MA9
Sharpe	2.07	3.2	3.49	3.47	2.96	3.03	2.44	1.98
Sortino	3.59	6.85	6.89	6.79	5.66	5.89	4.55	3.58
Return	42.26%	72.35%	79.83%	76.09%	59.33%	62.84%	46.22%	35.26%
Drawdown %	13.94%	10.87%	9.74%	9.74%	9.74%	10.02%	10.03%	10.54%
Win Rate %	56.89%	65%	70.73%	74.36%	72.22%	70.59%	67.65%	62.5%
Total Trades	59	41	41	39	36	34	34	32

Table 4: Performance with different moving averages and no size, costs and slippage. (Source: *Author*)

6.2.2 Simulated Market with Cost and Slippage Limitations

As mentioned in the previous section, a more detailed analysis was performed in order to simulate a more realistic simulation of the market. The costs for this simulations were set to 0.22% as a weighted average of the most popular crypto exchanges. Due to its elevated level of volatility, the slippage margin was set at 0.5%.

As expected, the performance of the trading strategy was significantly dragged down by the costs and simulated slippage. On the previous subsection, the algorithm was able to beat the benchmark in 7 cases out of 8. Adding the costs and slippage, the algorithm was able to beat the benchmark only in 6 cases out of 8 while also reducing all the other metrics. The most significant disparity in ratios are observed at the extremities of the table, with the MA2 and MA9. The number of trades was not affected due to the fact that the average of predictions were the same as in the previous tests performed.

Features	MA2	MA3	MA4	MA5	MA6	MA7	MA8	MA9
Sharpe	1.62	2.68	3.17	3.14	2.65	2.74	2.14	1.70
Sortino	2.76	5.11	6.16	6.07	4.99	5.25	3.94	3.03
Return	31.07%	62.87%	69.79%	66.74%	51.50%	55.27%	39.33%	29.24%
Drawdown %	15.67%	12.35%	10.05%	10.05%	10.06%	10.27%	10.27%	11.29%
Win Rate %	56.89%	63.04%	70.73%	74.36%	69.44%	67.65%	64.71%	56.25%
Total Trades	59	41	41	39	36	34	34	32

Table 5: Performance with different moving averages with costs and slippage. (Source: *Author*)

There are different methods to mitigate the costs and slippage. One common method is to set a stop-loss order, meaning that if the price falls to a certain level, the order will be executed. This method helps to limit the losses but also reducing the slippage. In this case, the stop-loss was set to 5% of the price, meaning that it cannot fall more than 5% from the buy price. While doing the backtest, it was observed that the stop-loss was triggered in 1 case out of 41 trades. Luckily, the algorithm does not get to the point to trigger the stop-loss, but it is a good method to mitigate the slippage and costs.

6.2.3 Size Limitations

This sub-section will have a more detailed explanation of the size restrictions. For the other simulations, the size limit of the orders were dynamic and the algorithm was choosing how much to buy based on the volatility and confidence of the prediction. This is not always the best case in real world, one might want to have control over how much to be exposed in the market. For this reason, another simulation was run to find the best size limit.

id	size	entry_price	entry_fees
0	0.174087	57431.023438	1.999600
1	0.180513	54139.687500	1.954581
2	0.169668	62940.457031	2.135797
3	0.177143	60632.785156	2.148140
4	0.168895	63193.023437	2.134592
⋮	⋮	⋮	⋮

Table 6: Dynamic size limitations of the orders based on the volatility and confidence of the prediction. (Source: *Author*)

The size limit was set to 10% of the portfolio. The size was not chosen randomly, the selection was made on the basis that it was smaller than the options that the algorithm had previously identified. Comparing Table 7 with Table 5, it can be observed a significant increase in Sharpe and Sortino ratios. This increase is due to the fact that, with a smaller size limit, the small number of losing trades was able to be compensated by the winning trades.

Adding the size limit, the model was able to beat the benchmark in 5 cases out of 8. But in the cases were achieved a higher return than the benchmark, the Sharpe and Sortino ratios were higher in some cases than the model without any costs and fees. This means that while adding a size limit, the algorithm is achieving a higher return (compared to benchmark) but lower risk. In order to deploy this kind of model, it is essential to recognise that risk represents a significant factor that must also be taken into account, in order to ensure that the objective of achieving a higher return is met compared to the risk taken.

Features	MA2	MA3	MA4	MA5	MA6	MA7	MA8	MA9
Sharpe	1.95	2.86	3.46	3.5	2.98	3.05	2.48	2.04
Sortino	3.39	5.41	7.03	7.19	5.98	6.19	4.86	3.87
Return %	26.46%	40.91%	46.74%	47.11%	39.14%	40.68%	32.60%	26.51%
Drawdown %	10.62%	7.42%	6.02%	6.01%	6.34%	6.44%	6.81%	7.77%
Win Rate %	56.90%	63.04%	70.73%	74.36%	72.22%	70.59%	67.65%	62.5%
Total Trades	59	47	41	39	36	34	34	32

Table 7: Performance with size limit of 10% of the portfolio including costs and slippage. (Source: *Author*)

A decrease in drawdown was also observed and can be seen in the cumulative return panel in Figure 3, which is another positive aspect of the size limit. It is evident from the data that the lowest drawdown is exhibited by MA5, with a drawdown of 6.01%. For the simulation without any limitations the lowest was 9.74%. It is also due to the fact that the algorithm is exposed to smaller risk by having a smaller size limit than what the model was choosing before.

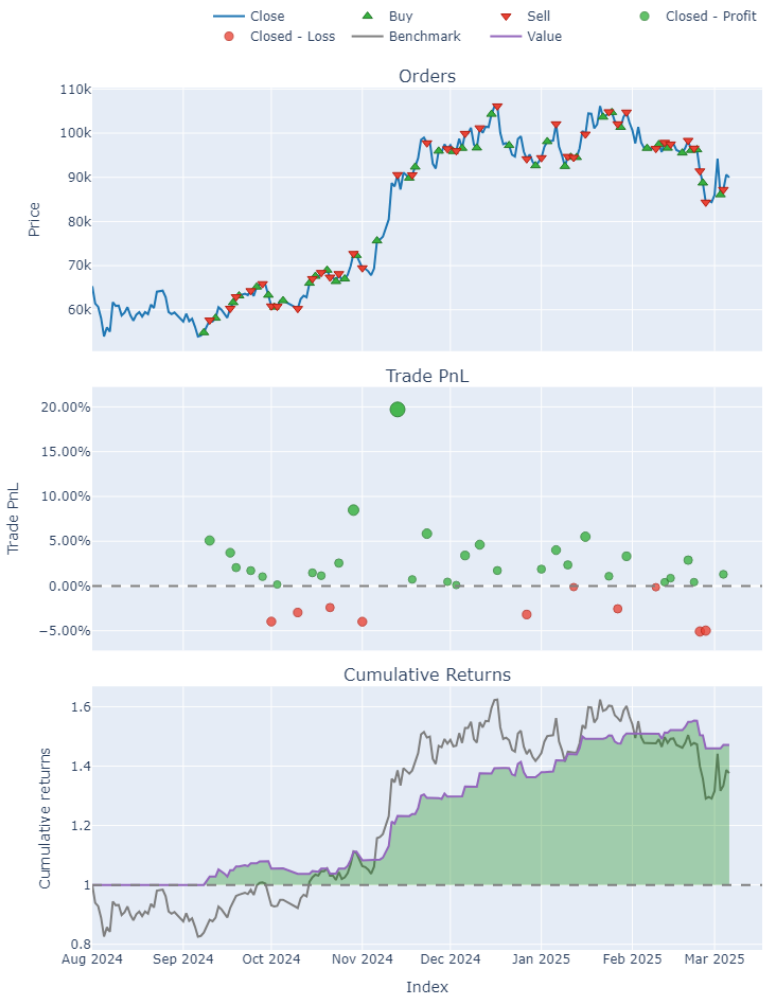

Figure 3: Performance of the simulated market with size limit - MA5. (Source: *Author*)

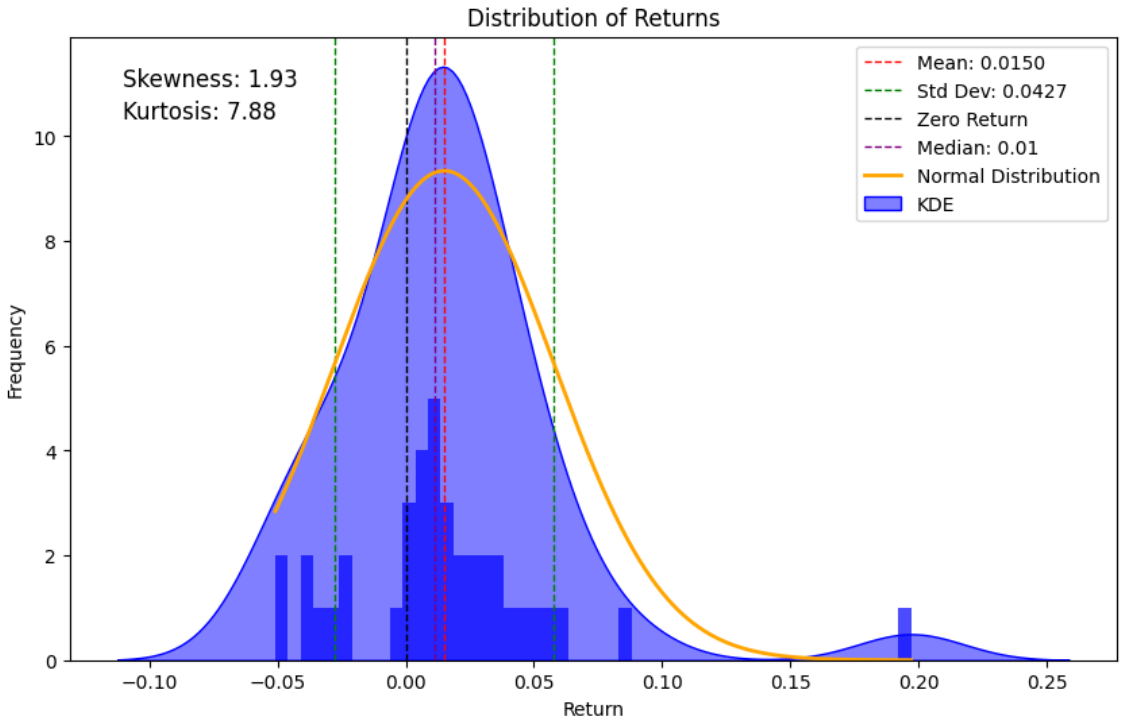

Figure 4: Distribution of return of all trades. (Source: *Author*)

From the previous Figure 4, a significant statistical difference can be observed between the profitable trades and the losing trades. With a total win rate of 74.36%, this high percentage of winning trades is a strong indicator of the model's ability to identify profitable trading opportunities with the help of sentiment analysis, volatility and trend data. The mean return per trade was 1.504%, while the standard deviation was 4.27%, indicating a moderate level of variation in the returns. This implies that the algorithm is able to capture favourable price movements while also mitigating the risk. Figure 4 shows a right skewed distribution of the returns. The skewness of the distribution is 1.93. The kurtosis is 7.88, indicating heavier tails than a normal distribution. It could mean that the model captures the extreme events, one important aspect of trading cryptocurrencies. With all being said, these results indicate that the algorithm is able to generate profitable opportunities and is not mere luck.

7 Conclusion

This study aimed to develop a framework for the prediction of Bitcoin price to create a profitable and scalable trading strategy. It was motivated by the spotlight that cryptocurrencies and machine learning have received in the last decade. The model was developed to be robust and accurate with the goal of being able to be used in real-time trading. Based on the data collected, the sentiment of news articles, the volatility of Bitcoin and the Google Trends data are good predictors of the price of Bitcoin and can be used along with machine learning to create a profitable trading strategy.

To achieve that, this study went through an extensive review of the researches and articles published, related to the topic at hand, where similarities were found in what Hossain et al. (2024) did with the use of FinBERT and BiLSTM to predict the price of Bitcoin and implement a trading strategy. We then shared the back story of cryptocurrencies and Bitcoin, before collection of quantitative and qualitative data from various sources. The methodology used then detailed how the Bitcoin price was transformed and how we get to the sentiment score and the steps of prediction. This led us to the empirical results, showing that the model was able to achieve a 74.36% win rate, with a Sharpe ratio of 3.5 and a maximum drawdown of 6.01%. The results indicate that the model captures very well the price movements.

As a trading strategy, it was able to beat the benchmark in almost all scenarios. When there were no costs and fees involved, the model was able to beat the market in all scenarios. However, when the costs and fees were included, the model was able to beat the market in 5 out of 8 scenarios. This indicates that the model is robust and can be used in real-time trading. The algorithm was capturing the price movements better when there were moments of high volatility, which is a good indicator that it is able to adapt to the market conditions.

The limitation of this study is the fact that good quality data is hard to find or requires an expensive subscription. The data used in this study was collected from various sources, including Yahoo Finance, Google Trends and NewsDataHub. However, the news articles were not always directly linked to Bitcoin, which could have affected the results of the model. Because of this, the model was trained on a small data set and the results may not be generalizable to other time periods of Bitcoin. Future research could focus on collecting more quality data and deploy more advanced machine learning techniques for prediction improvement.

References

Alstyne, M. V. (2014), 'Why bitcoin has value', *Communications of the ACM* .
URL: *https://cacm.acm.org/opinion/why-bitcoin-has-value/*

Araci, D. (2019), 'Finbert: Financial sentiment analysis with pre-trained language models', *arXiv preprint arXiv:1908.10063* .
URL: *https://api.semanticscholar.org/CorpusID:201646244*

Arslan, M., Shahzad, A., Shafique, A. & Ahmed, W. S. (2025), *Forecasting Bitcoin: A Comparative Analysis of Traditional versus Machine Learning Approach*, World Scientific Publishing Co. Pte. Ltd., chapter 10, pp. 257–279.
URL: *doi.org/10.1142/9781800616257_0010*

Birjali, M., Kasri, M. & Beni-Hssane, A. (2021), 'A comprehensive survey on sentiment analysis: Approaches, challenges and trends', *Knowledge-Based Systems* **226**, 107134.
URL: *https://www.sciencedirect.com/science/article/pii/S095070512100397X*

Chekhlov, A., Uryasev, S. & Zabarankin, M. (2005), 'Drawdown measure in portfolio optimization', *International Journal of Theoretical and Applied Finance* **8**(01), 13–58.
URL: *dx.doi.org/10.1142/s0219024905002767*

Chen, T. & Guestrin, C. (2016), Xgboost: A scalable tree boosting system, *in* 'Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', KDD '16, ACM, p. 785–794.
URL: *http://dx.doi.org/10.1145/2939672.2939785*

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. & Bengio, Y. (2014), 'Learning phrase representations using rnn encoder-decoder for statistical machine translation'.
URL: *https://arxiv.org/abs/1406.1078*

Din, R. U., Ahmed, S., Khan, S. H., Albanyan, A., Hoxha, J. & Alkhamees, B. (2025), 'A novel decision ensemble framework: Attention-customized bilstm and xgboost for speculative stock price forecasting', *Plos One* **20**.
URL: *doi.org/10.1371/journal.pone.0320089*

Frankenfield, J. (2021), 'ecash definition'.
URL: *https://www.investopedia.com/terms/e/ecash.asp*

Gómez Martínez, R., Martínez Navalón, J. G. & Prado Román, C. (2022), 'Bitcoin investment strategies based on google trends and ai models', *Tripodos* **52**(52), 129–141.
URL: *https://doi.org/10.51698/tripodos.2022.52p129-141*

Gunarso, G. & Stephanie (2022), 'Cryptocurrency and its state of research', *International Dialogues on Education Journal* **9**, 151–175.
URL: *doi.org/10.53308/ide.v9i1.280*

Gurgul, V., Lessmann, S. & Härdle, W. K. (2023), 'Deep learning and nlp in cryptocurrency forecasting: Integrating financial, blockchain, and social media data', *International Journal of Forecasting* .
URL: *https://arxiv.org/abs/2311.14759*

Hochreiter, S. & Schmidhuber, J. (1997), 'Long short-term memory', *Neural Computation* **9**(8), 1735–1780.
URL: *https://doi.org/10.1162/neco.1997.9.8.1735*

Hossain, M. F. B., Lamia, L. Z., Rahman, M. M. & Khan, M. M. (2024), 'Finbert-bilstm: A deep learning model for predicting volatile cryptocurrency market prices using market sentiment dynamics'.
URL: *https://arxiv.org/abs/2411.12748*

Indurkhya, N. & Damerau, F. J., eds (2010), *Handbook of Natural Language Processing*, 2 edn, Chapman and Hall/CRC.
URL: *https://doi.org/10.1201/9781420085938*

Jiang, Z. (2023), An attention gru-xgboost model for stock market prediction strategies, *in* 'Proceedings of the 4th International Conference on Advanced Information Science and System', AISS '22, Association for Computing Machinery, New York, NY, USA.
URL: *https://doi.org/10.1145/3573834.3573837*

Kim, G., Kim, M., Kim, B. & Lim, H. (2023), 'Cbits: Crypto bert incorporated trading system', *IEEE Access* **11**, 6912–6921.
URL: *https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10014986*

Kolluri, J., Kotte, V. K., Phridviraj, M. & Razia, S. (2020), Reducing overfitting problem in machine learning using novel l1/4 regularization method, *in* '2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)', pp. 934–938.
URL: *dx.doi.org/10.1109/ICOEI48184.2020.9142992*

Mohammadjafari, A. (2024), 'Comparative study of bitcoin price prediction', *arXiv* .
URL: *https://doi.org/10.48550/arXiv.2405.0808*

Moore, R. & DeNero, J. (2011), L1 and l2 regularization for multiclass hinge loss models, *in* 'Machine Learning in Speech and Language Processing (MLSLP 2011)', pp. 1–5.
URL: *https://www.isca-archive.org/mlslp_2011/moore11_mlslp.html*

Rizwan, M., Narejo, S. & Javed, M. (2019), 'Bitcoin price prediction using deep learning algorithm'.
URL: *https://ieeexplore.ieee.org/document/9024772*

Schmidt, R. M. (2019), 'Recurrent neural networks (rnns): A gentle introduction and overview', *arXiv* **abs/1912.05911**.
URL: *http://arxiv.org/abs/1912.05911*

Sharpe, W. F. (1966), 'Mutual fund performance', *The Journal of Business* **39**, 119–138.
URL: *https://www.jstor.org/stable/2351741*

Smuts, N. (2019), 'What drives cryptocurrency prices? an investigation of google trends and telegram sentiment', *ACM SIGMETRICS Performance Evaluation Review* **46**(3), 131–134.
URL: *dx.doi.org/10.1145/3308897.3308955*

Sortino, F. A. & van der Meer, R. (1991), 'Downside risk', *The Journal of Portfolio Management* **17**, 27–31.
URL: *https://dx.doi.org/10.3905/jpm.1991.409343*

Weber, M. (2024), 'Lesson 73: Evaluating model robustness and generalization'.
URL: *https://kindatechnical.com/advanced-llm-topics/lesson-73-evaluating-model-robustness-and-generalization.html*

Yang, Y., Uy, M. C. S. & Huang, A. (2020), 'Finbert: A pretrained language model for financial communications', *arXiv preprint arXiv:2006.08097* .

Zahid, M., Iqbal, F. & Koutmos, D. (2022), 'Forecasting bitcoin volatility using hybrid garch models with machine learning', *Risks* **10**(12).
URL: *https://www.mdpi.com/2227-9091/10/12/237*