Algorithmic Cryptocurrency Trading using Sentiment Analysis and Dueling Double Deep Q-Networks

David Gallo

Supervised by Dr. Neha Chaudhuri
MSc Artificial Intelligence and Business Analytics
Toulouse Business School
August 4th, 2022

Executive Summary

This thesis details applications of sentiment analysis and deep reinforcement learning for cryptocurrency price prediction of Ether. The research focused on a highly applicable use case, by clearly detailing data features, data collection methodology, model attributes, and deployment considerations. The goal of this thesis was not to build the highest-performing neural network for cryptocyurrency price prediction but rather to prove the technical feasibility of Deep Q-Networks in a novel application and establish deep reinforcement learning and sentiment analysis as a decision support system for investors.

Over 5 million Tweets were collected and processed in Spark NLP using FinBERT (a BERT extension) for sentiment analysis, and hourly financial data of Ether was collected between January 1, 2017 and July 22, 2022.

The first section of the thesis explores the causal relationships between Twitter sentiment scores, number of tweets, and the closing price of Ether. Granger causality was used to determine that the closing price of Ether forecasts social media sentiment and the number of Tweets. Further, the number of tweets also forecasts changes in closing price.

Next, a Dueling Double Deep Q-Network (DDDQN) was built in Python to trade cryptocurrency by considering the trading process as a Markov Decision Process. The dataset was split using pre-January 1, 2021 as a training set, and the time after the split as a testing set. The trading agent of the DDDQN model generated -33.19% returns during the testing period without Twitter data as feature input, beating the market by 20.28%. When sentiment scores and Twitter volume were added as features, the performance increased by 10.95% to -22.24% returns, beating the market by 31.23%, but with strong variability between runs for all cases.

This thesis addresses four gaps in existing literature. First, a novel combination of deep reinforcement learning and sentiment analysis were combined for cryptocurrency price prediction, something that to date has not been researched in depth. Next, it focuses on Ether, where most cryptocurrency research focuses exclusively on Bitcoin. Third, the theory-heavy focus of prior research papers was extended into practical applications by clearly detailing development and deployment steps for a deep reinforcement algorithmic trader in a cloud-based environment. Finally, a bear market was used as the testing set when measuring model performance, as most previous cryptocurrency research was conducted during a strong bull market (pre-2021).

The conclusion summarizes that the novel application of deep reinforcement learning for cryptocurrency price prediction does indeed merit future research, and provides some suggestions to extend the work presented in this thesis.

Acknowledgement

There are a few people who deserve thanks for making this thesis possible. First, I would like to express my gratitude to my supervisor, Dr. Chaudhuri, who guided me throughout this project. I would also like to thank my professors and classmates for much-needed inspiration. Next, to Layane, thank you for supporting me during long nights of research. And finally, my biggest thanks goes to my parents and grandparents, for encouraging and funding my university education. Without you, this work would not have been possible.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Context

1.1.1 What is cryptocurrency?

A cryptocurrency is a digitally secured currency, without a central authority like a government or bank managing its distribution and validity. Its value is therefore determined by its supply and the demand from people who trade and use it. It operates as both a medium for exchanging value (in the same way that cash is used in a real-world transaction to purchase goods and services), and also as an investment vehicle or store of value (purchased and traded similar to shares in a public company on the stock market). Cryptocurrency is bought, sold, and traded using the internet, and supported by a blockchain to record these transactions. (Narayanan, Bonneau, Felten, Miller, & Goldfeder, 2016)

The most popular cryptocurrency, Bitcoin, was ideated by Satoshi Nakamoto (a psudeonym) in their landmark 2008 paper titled "Bitcoin: A Peer-to-Peer Electronic Cash System." The paper described the first "electronic payment system based on cryptographic proof instead of trust" (Nakamoto, 2008) where transactions are verified and stored on a blockchain. New projects quickly followed Bitcoin, such as Ether (ETH), a cryptocurrency built on the Ethereum blockchain where the cryptocurrency is used to support the underlying smart contract blockchain. (*What is Ether (ETH)?*, n.d.)

A blockchain is an open and immutable ledger that records financial transactions. Most importantly, it is distributed across all users of the blockchain, so that every user has an identical copy. When a new transaction is entered, it is verified and added by all other users of the blockchain. Cryptographic validation of new entries using hash functions prevents malicious actors from adding fraudulent entries; only entries where the consensus of users verify its accuracy are added. The name is derived from blocks of transactions that are linked together for form a blockchain. Trading cryptocurrency using a blockchain is a fundamental example of the "sharing economy", the economic system whereby assets or services are shared between individuals using the internet. (Frenken & Schor, 2017)

This thesis focused on Ether, the cryptocurrency built on the Ethereum blockchain, and the second largest coin by market cap. (Goswami, Borasi, & Kumar, 2021) The Ethereum blockchain offers the unique feature of storing code for execution in the form of "smart contracts". Ether is used as payment for users of the Ethereum blockchain to execute work on the network, and can therefore be thought of as the "fuel" of the network. It differs in this sense from other cryptocurrencies whose blockchains are used simply to record transactions. (Buterin, 2014)

1.1.2 How do cryptocurrency investments differ from equity investments?

Although cryptocurrencies were initially developed as a way to facilitate digital transactions, they have found applications primarily as a investment vehicles. Peer-to-peer exchanges of currency

have been scaled into full-sized global exchanges – platforms like Binance or Coinbase allow individuals to purchase, sell, and trade their cryptocurrency with other users, in a very similar manner to stock exchanges like the NASDAQ.

Equity trading is primarily done through stocks – shares in public companies, bought and sold by investors. As an asset class, stocks are well-established with large financial institutions built solely for managing and trading stocks. On the other hand, cryptocurrencies are much newer investment vehicles that lack the formal structure and governance of stocks, therefore making them much more volatile. While this risk can certainly beget higher returns, trading cryptocurrency should be done more cautiously to avoid significant losses.

Stocks represent partial ownership of a public company, while most cryptocurrencies are not tied to any underlying asset. Therefore, while stocks can be speculative instruments, they are often reflective of the assumed value of a company. If a company is performs better, its share price will therefore increase. Cryptocurrency does not share this trait: it does not come with any ownership of an entity, and its value is determined therefore by public sentiment rather than real-world performance of usability and adoption. (Polasik, Piotrowska, Wisniewski, Kotkowski, & Lightfoot, 2015) Bitcoin for example is rarely used in transactions (its primary use case), but had a $1.49 billion market cap in 2020, with projections of reaching $4.94 billion by 2030 (growing at a CAGR of 12.8% from 2021 to 2030) (Goswami et al., 2021). I believed that observing the sentiment investors have towards cryptocurrency would therefore play an important role in determining when it was profitable to buy or sell.

Until recently, most retail stock market investors have not been especially vocal on social media with their expectations about stock performance. Tesla became an exception to this rule as a company with a cult stock, where herd behaviour may in part have pushed its price to unrealistic valuations (Cheng & Griffin, 2022). Cryptocurrencies are somewhat similar to Tesla in this sense – many proponents of Bitcoin as a smart investment for example tend to be very vocal about their thoughts. Smaller, newer coins too tend to have have more die-hard fanatics, even with no objective reason for their expectations. Studies exploring this phenomenon have highlighted cognitive bias as the likely cause, where new investors are enamored by the possibility of incredible rewards, and publicly support their cryptocurrency of choice even when irrational. (Delfabbro, King, & Williams, 2021)

1.1.3 What is sentiment analysis?

As previously mentioned, cryptocurrency prices are driven primarily by public sentiment – the more investors who want to purchase it as a store of value, the higher the price grows. It therefore seemed beneficial to collect and measure that public sentiment, in order to predict how and when the price might change. Sentiment analysis is an application of Natural Language Processing (NLP) that allowed us to do exactly that.

Sentiment analysis is the collection, processing, and extraction of textual data to systematically determine and quantify emotional state. To do this, a machine learning model (in the case

of this research, an artificial neural network (ANN)) is trained using a collection of sentences labeled with their sentiment and used to classify new data. Advanced sentiment analysis models distinguish between different human emotions (e.g. fear, happiness, sadness, surprise), whereas more general models rank a sentence's sentiment from positive to negative. (Mohammad, 2016) Because overall negative or positive outlook of cryptocurrency was the most important output for the application presented in this thesis, a generalized model that simply highlights whether text was more positive or negative was sufficient. The most important characteristic of the sentiment analysis model that was selected in this thesis was that its training data contained similar sentences as would appear in the data intended for classification.

1.1.4 What is Granger causality?

One of the focuses of this thesis was to include exploration of social media sentiment and its effect on cryptocurrency prices. Determining cause and effect is not a trivial task however – correlation between cryptocurrency price and sentiment does not necessarily mean that one causes the other. One could argue that strong social media support encourages people to buy cryptocurrency, thus inflating the price. On the other hand, one could equally argue that a well-performing cryptocurrency encourages investors to share their positive feelings on social media. Causality goes beyond correlation and asserts that a variable, X (social media sentiment), can be said to cause another variable Y (cryptocurrency prices) if changing X results in a change in Y, but changing Y does not necessarily result in a change in X. When two variables are correlated but not causal however, symmetric changes are expected. (Eichler, 2012)

Because a trading agent does not have control over cryptocurrency prices and social media, statistical approximations can instead be used to measure causality. Of these statistical tests, Granger causality is among the most popular for time-series data, meaning it is applicable to the use case in this research. Econometritian Clive Granger created his method for causality detection with the argument that causality in economics could be tested for by using lagged values of one time series to predict the future values of a different time series. This predictive causality, or precendence, allowed researchers to understand if one variable forecasted another.

Paraphrased from Granger's original 1969 paper: a time series X is said to Granger-cause Y if it can be shown, usually through a series of t-tests and F-tests on lagged values of X (and with lagged values of Y also included), that those X values provide statistically significant information about future values of Y. (Granger, 1969) Granger later clarified the causality relationship based on two principles: (Granger, 1980)

1. The cause happens prior to its effect.

2. The cause has unique information about the future values of its effect.

Given these two assumptions, Granger proposed the following hypothesis test for identification

of a causal effect of X on Y:

$$\mathbb{P}[Y(t+1) \in A \mid \mathcal{I}(t)] \neq \mathbb{P}[Y(t+1) \in A \mid \mathcal{I}_{-X}(t)] \tag{1}$$

where $\mathbb{P}$ refers to probability, A is an arbitrary non-empty set, $\mathcal{I}(t)$ denotes the information available in an environment at time t, and $\mathcal{I}_{-X}(t)$ denotes the information available in the same environment but with X excluded. If the above hypothesis is accepted, it can be said that X Granger-causes Y.

1.1.5 What is a Markov Decision Process?

Markov decision processes (MDP) are a specific types of sequential decisions which are at the foundation for problems that can be solved using reinforcement learning. MDPs provide a mathematical framework to model decision making in a partly-random environment. Fundamentally, an MDP is a stochastic control process, where a decision is made to transition into a future state with discrete time steps. MDPs are not new research, popularized originally by Richard Bellman in his 1957 paper "A Markovian Decision Process", which extended Andrey Markov's notion of Markov chains.

In an MDP, an agent (decision-maker) observes the state of its environment at each discrete time step $t = 0, 1, 2, ...$ to learn the current state, s_t, then takes an action based on this observation, a_t. Based on the action state pair, the agent is granted a reward for the next state, r_{t+1}, a new discount factor for future rewards, γ_{t+1}, and the environment then changes with randomness into a new state, s_{t+1}. In an MDP, $a \in A$ where A is the action space, $r \in R$ where R is the reward space, and $s \in S$ where S is the state space, and $\gamma \in [0, 1]$. The reward function is given by $r(s, a) = \mathbb{E}[r_{t+1} \; s_t = s, a_t = a]$. The probability of moving into a new state s' is given by the state transition function $T(s, a, s') = P(s_{t+1} = s' \mid s_t = s, a_t = a)$. The Markov property is satisfied because the state transitions are considered independent. An MDP can therefore be denonted as the following tuple:

$$(S, A, T, r, \gamma) \tag{2}$$

MDPs differ from Markov chains only by adding a decision point (selecting an action) and providing rewards based on that action. Consider for example an MDP where there is only one action in a given state and that all rewards are the same; in this situation, the MDP simplifies to a Markov chain.

From a state s_t, the discounted sum of rewards is given by:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ...$$

which is more generally represented as:

$$G_t = \sum_{n=0}^{\infty} \gamma^n r_{t+n+1} \tag{3}$$

The discount for a reward n steps in the future is given by the product of all discounts before that step: $\gamma_t^n = \prod_{i=1}^{n} \gamma_{t+1}$. Discounting each term in the sum over time with γ^n is required to prevent the sum from increasing infinitely – assuming a continuing task problem with no end point (such is the case with cryptocurrency trading), the expected sum of rewards would be infinite: $G_t = r_{t+1} + r_{t+2} + ... + r_{\infty}$. The discount factor forces the infinite sum to converge.

The objective of an agent in an MDP is to choose a policy π that maximizes the expected discounted sum of rewards. π is a probability distribution over the actions in each state. For example, at time t, if an agent follows policy π then $\pi(a|s)$ is the probability of $a_t = a$ when $s_t = s$. Next, a value function can be definted that describes the expected future rewards for an agent to select a specific action or to be in a given state. If the policy π denotes the *probability* of selecting an action in a given state, then the value function denotes how *beneficial* it is. Knowing that the policy is trying to maximize the sum of discounted return, these values are of course inherently linked. The state-value function for policy π (denoted as v_π) describes how good a state is for an agent following that policy. The value of state s under policy π is therefore the expected return of discounted rewards when starting in state s and following π. Equation (3) can be used to define $v_\pi(s)$ as:

$$v_\pi(s) = \mathbb{E}_\pi[\sum_{n=0}^{\infty} \gamma^n r_{t+n+1} \mid S_t = s] \tag{4}$$

The action-value function for policy π (denoted as q_π) describes how beneficial it is for the agent to take an action while following policy π. The value of action a in state s under policy π is the expected return of discounted rewards when starting from state s, taking action a, and following policy π from that point forwards. Equation (4) can be used to define $q_\pi(s, a)$ as:

$$q_\pi(s, a) = \mathbb{E}_\pi[\sum_{n=0}^{\infty} \gamma^n r_{t+n+1} \mid S_t = s, A_t = a] \tag{5}$$

This function is colloquially referred to as the q-function, and the output from the function for a state-action pair the q-value. This transitions us directly into Q-learning.

1.1.6 What is Q-learning?

In Q-learning, the notion of MDPs is extended to describe how an optimized policy π can be found. A policy is considered better (more optimal) than another if the expected return of

discounted rewards is greater than another policy for all states:

$$\pi \geq \pi' \iff v_\pi(s) \geq v_{\pi'}(s) \, \forall \, s \in S$$

Returning back to the idea of state-action values, an optimal q-function can be denoted as q_*, and defined as:

$$q_*(s,a) = \max_\pi q_\pi(s,a) \, \forall \, s \in S, \; a \in A \tag{6}$$

In other words, this means that q_* generates the largest expected discounted rewards achievable by any policy π for each possible state-action pair.

One of the most important concept in Q-learning is Bellman's Optimality Equation, which states that q_* must satisfy:

$$q_*(s,a) = \mathbb{E}[r_{t+1} + \gamma \max_{a'} \; q_*(s',a')] \tag{7}$$

What this formula means is that for any state-action pair (s,a) at time t, the expected discounted rewards from starting in state s, choosing action a, and following the optimal policy π_* (as elaborated, the q-value of this state-action pair) is the immediate reward received (i.e.: r_{t+1}), plus the *maximum* expected discounted return that can be achieved by any possible next state-action pair (s',a'). Because the agent follows an optimal policy π_*, the next state s' must be the best possible state for the agent, which allows the agent to select the best possible next action a'.

Using the Bellman Equation to find q_* is valuable, because an optimal policy can be determined or approximated with reinforcement learning. This is because deep learning can be used to determine the action a that maximizes $q_*(s,a)$.

The objective of Q-learning is to find the optimal policy π_* by learning the optimal Q-values for each state-action pair. In a finite and sufficiently small state and action space, this is achievable without using deep learning for approximation. To find the optimal policy in a smaller state-action space, the Q-learning algorithm iteratively updates the Q-values as it tests different state-action pairs by using the Bellman equation until the Q-function converges.

After testing all possible state-action pairs, the agent knows what action to take to maximize reward. However, knowing when to explore the environment to test state-action pairs, or select the best action to maximize reward is a classic exploration/exploitation reinforcement learning problem. Q-learning derives new policies from a state-action value function by acting "ϵ-greedily" when selecting action values. The agent chooses the action with with the highest expected cumulative discounted return (the greedy action) with probability $1 - \epsilon$, and otherwise select an action from the action space A at random with uniform probability. Although randomly selecting actions may provide less reward, the agent cannot learn optimal policies and correct its estimates without experimentation.

To update the q-value of a state-action pair (s,a), the Bellman Equation (Bellman, 1957) is

used to calculate the $q_*(s, a)$ and iteratively compare the loss between $q(s, a)$, and $q_*(s, a)$. The q-value is updated each time the agent sees the same state-action pair by taking the weighted average of the new q-value, discounted by a learning rate α, and the old q-value. Mathematically, the loss function to compare old and new q-values is denoted as:

$$q_*(s, a) - q(s, a) = loss$$

Expanding the equation using equations (5) and (7):

$$\mathbb{E}[r_{t+1} + \gamma \max_{a'} \ q_*(s', a')] - \mathbb{E}_\pi[\sum_{n=0}^{\infty} \gamma^n r_{t+n+1}] = loss \tag{8}$$

The goal of a reinforcement learning algorithm is to minimize this loss function.

1.1.7 What is Deep Q-Network Learning?

Large state or action spaces make it computationally infeasible to learn q-value estimates for each state-action pair to create an optimal policy. In the application presented in this thesis, the state space is effectively infinite since the price of cryptocurrency can be any real number.

In Deep Q-Learning, the same concepts as Q-learning are applied, but a neural network is used to estimate q-values for state-action pairs. Once the loss is calculated according to equation (8) using neural network approximations for q-values, the gradient of the loss is back-propagated to update the weights of the neural network, the same as any other artificial neural network. The last addition to Deep Q-Networks is the idea of an online and target network: two identical neural networks, where the target network is updated more slowly than the online network.

To calculate the loss between the actual q-values and the estimated optimal (target) q-values, two passes through the neural network must be made. This is because in equation (5), the next state-action pair (s', a') must be computed as well as the current state-action pair (s, a), which is used when calculating loss in equation (8). This process leads to unstable learning if the same neural network weights are used in both passes. As the actual q-values approach their target q-values, the target q-values continue to move farther away. Optimization is therefore unstable since the target q-values are always be moving in the same direction as output q-values. To counteract this problem, a second neural network is introduced (the target network), which calculates the target q-values. The weights of this network are periodically copied from the online network; the fixed q targets from this network support much more stable learning. For a deeper understanding about why two neural networks are necessary, the source paper on Deep Q-Networks explored this in greater detail (Mnih et al., 2015)

The architecture of the neural networks as well as the training process for Deep Q-Networks are elaborated in the *Methodology* section of this paper.

1.1.8 What are extensions to Deep Q-Learning?

As Deep Q-Learning was the first major paper exploring a deep neural network model built exclusively for reinforcement learning, various improvements have since been made to increase its performance in standard tests. Six of the most applied extensions to Deep Q-Networks were summarized in Hessel et al's 2017 paper "Rainbow: Combining Improvements in Deep Reinforcement Learning". Of these, the Dueling DQN and Double DQN are among the simplest and most effective changes to the underlying model. Although combining all extensions to use Rainbow DQN for this thesis would have been interesting, the complexity meant that it would be out of scope for a paper focused on experimenting with a simple proof of concept. Instead of using all six extensions, Double DQN and Dueling DQN were selected as two that could be easily added to Deep Q-Networks with minimal code changes, in order to enhance the base Deep Q-Network's performance and provide the best chance of success for this application of reinforcement learning.

Dueling Deep Q-Networks

The Dueling Deep Q-Network (Dueling DQN) is a neural network architecture designed for value-based reinforcement learning, which was the exact use case for this thesis. A dueling network uses two different output streams: the value stream, computing the value of a state (single node), and the action stream, computing the value of each action in the action space (one node per action). These are then merged by a non-trivial linear aggregation. The importance of this aggregation is described in the paper that introduced Dueling networks, "Dueling network architectures for deep reinforcement learning" (Wang et al., 2015). The details are not elaborated here, but suffice to say that simply adding the values of the state to the actions is not sufficient; the average action value must be calculated and used as well. The mathematical representation of a Dueling DQN is as follows:

$$q_\theta(s,a) = v_\eta(f_\xi(s)) + a_\psi(f_\xi(s),a) - \frac{\sum_{a'} a\psi(f\xi(s),a')}{N_{actions}} \tag{9}$$

where ξ is the parameter of the shared encoder f_ξ, η is the parameter of the value stream v_η, and ψ is the parameter of the advantage stream a_ψ. $\theta = \xi, \eta, \psi$ is their concatenation.

As shown, the Deep Q-Network equation is expanded by separating the value of actions from the value of the state. This allows the network to better distinguish between the values of different actions. In most states, the expected reward from all actions are similar, making the decision less important. Purchasing a unit of cryptocurrency for example during a stable economic period is less important than purchasing one right before a sharp price increase. Although a Dueling DQN tends to be more valuable in environments with a large action space, it should still improve learning during periods of stability in the environment, where buying and selling units of cryptocurrency yields similar results to holding. In vanilla Deep Q-Networks, the q-values for each training iteration are updated with q-values only for the specific actions taken in each state. This results in slower learning as the q-values for actions that were not taken yet are not used.

In contrast, the dueling architecture speeds up learning as an agent can start learning the value of a state even if only a single action has been taken in that state.

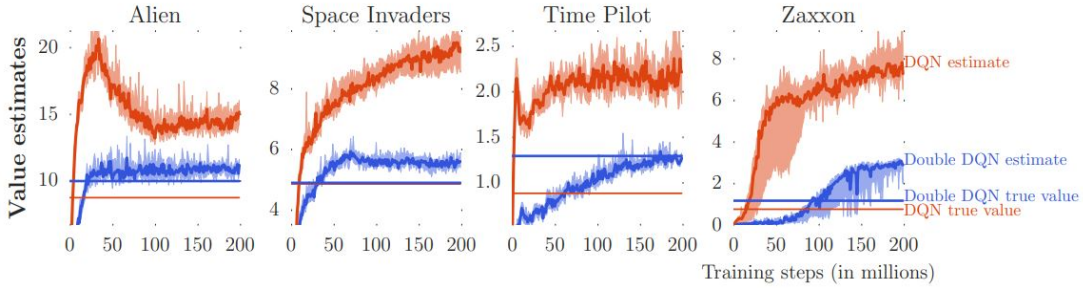
Double Deep Q-Networks

The Double Deep Q-Network (DDQN) architecture architecture is similar to the vanilla DQN but leverages a second neural network to determine the q-value of the maximal action, after that maximal action was first selected by the first neural network.

One of the major mathematical deficiencies of vanilla Deep Q-Networks is the inherent overestimation bias present during q-value calculations. Because of the maximization step in equation (9):

$$\max_{a'} q_\theta(S_{t+1}, a'_t)$$

the network implicitly takes the estimate of the maximum value. For example, consider the situation where each action in a single state has a true q-value of 0. However, because q-values are being estimated, they are distributed around 0, with some above and some below. When the maximum is selected, it therefore chooses a value greater than 0. Because Deep Q-Networks since Q-learning involves bootstrapping, where it learns new estimates from its previous estimates, this overestimation leads to unstable learning. (Wang et al., 2015) illustrated this overestimation bias using Atari game environments:



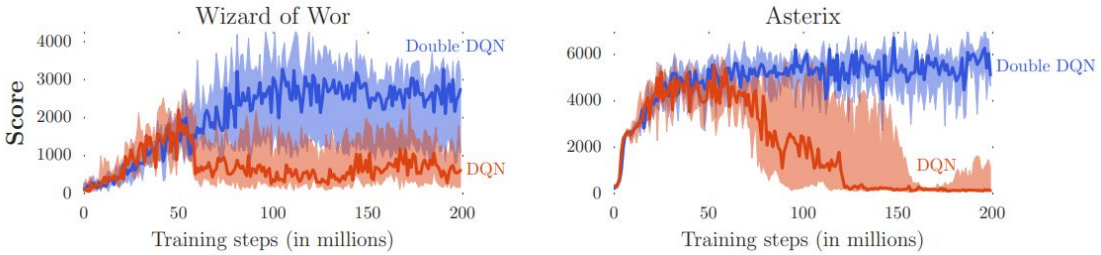
(Wang et al., 2015)

Figure 1: DQN and DDQN estimates versus true values

The authors further showed that overestimation bias resulted in significantly lower performance in some applications:

The solution to overestimation bias is to separate the calculation of maximal action from the estimation of that action's value. This can be accomplished by using two different q-value estimators, each of which is used to update the other, thus reducing the effect of maximization bias.

Hasselt first addressed this overestimation in his 2010 paper "Double Q-learning" by using two separate q-value estimators – using one or the other with 50% probability to select the maximal action and update the estimator not selected. The same author later elaborated his methodology for deep learning in his paper "Deep Reinforcement Learning with Double Q-learning". There,



(Wang et al., 2015)

Figure 2: DQN versus DDQN performance

he used a first neural network, Q (which is referred to from this moment forwards as the "online" network) to select the maximal action, then a second neural network, Q' (which is referred to as the "target" network) to estimate the q-value of that action. Mathematically, this modified Deep Q-Network equation is represented as:

$$q^*(s_t, a_t) \approx r_t + \gamma Q(s_{t+1}, argmax_{a'} Q'(s_t, a_t)) \tag{10}$$

Q' (the target network) copied the weights of Q (the online network) every n time steps, where n was a hyperparameter of the dual network system. Further details about hyperparameter selection can be found in the methodology section.

As described in the methodology section, this research combined both extensions to create a high-performing Dueling Double Deep Q-Network (DDDQN).

1.1.9 How can sentiment analysis and Deep Q-Networks be used to trade cryptocurrency?

As highlighted, reinforcement learning works by identifying patterns in source data to try to determine optimal decisions over time. Many pieces of research have shown that there is a strong correlation between social media sentiment and stock prices, so much so that patterns in sentiment can be used to effectively predict stock market movement (Bharathi & Geetha, 2017). Therefore, I propose that sentiment scores may be an important feature for a reinforcement learning model designed to trade cryptocurrency, since cryptocurrency markets and stock markets share many similarities (Durcheva & Tsankov, 2019). Sentiment analysis can therefore be used as decision-making input when deciding to trade.

1.1.10 Why is this research important?

This research served to provide investors with an alternative decision support system to technical analysis, fundamental analysis, statistical measures, price action, and RNN-based algorithmic traders. Deep learning models using historical price and volume data for cryptocurrency prediction had already been thoroughly developed and tested using supervised LSTM, GRU, and

RNN neural networks, well-described in the literature summary "A comparative study of bitcoin price prediction using deep learning" (Ji, Kim, & Im, 2019). However, reinforcement learning was relatively under-researched as a trading tool, with fewer applications still researching its effectiveness on cryptocurrency markets. This was therefore a novel application of reinforcement learning, which may provide unique decision-making support where other algorithmic trading models did not.

Reinforcement learning, while not necessarily more accurate than supervised learning, has a unique set of benefits for cryptocurrency trading. It is a more "human" style of machine learning, where an algorithm is provided a given scenario and instructed to choose an action, mimicking the behaviour of a human investor (albeit on a much faster scale, with more data). Like a human, an algorithm can "test" the market with different trades, learn from the results, and create a profitable trading strategy. Further, training is inherently sequential, meaning that how the model learns is also more intuitive to understand than supervised neural networks.

Although machine learning and artificial intelligence (AI) have become more prevalent in the business world, there are still large gaps in understanding about how these tools can be used. Unfortunately, the "black box" nature of most deep learning algorithms can prevent managers from adopting them. Humanizing complex calculations through more intuitive methods like reinforcement learning can encourage adoption. (Markus, Kors, & Rijnbeek, 2021)

There are also three main gaps in existing research that were addressed with this thesis. First, a novel combination of deep reinforcement learning and sentiment analysis were combined for cryptocurrency price prediction, something that to date had not been researched in depth. Next, the performance of a deep learning model was measured in the recent cryptocurrency bear market, something that prior research had not been able to do. Finally, the theory-heavy focus of prior research papers was extended into practical applications by clearly detailing development and deployment steps for a deep reinforcement algorithmic trader in a cloud-based environment.

1.2 Research objectives

There were two clear research objectives from this thesis:

1. Determine whether there is a causal relationship between social media data and the price of cryptocurrency, using big data.

2. Create a proof-of-concept automated cryptocurrency trading algorithm using Deep Q-Networks, with sentiment analysis and price data as feature input.

It is important to note that the goal of this thesis was not to build the highest-performing neural network for cryptocyurrency price prediction; if that were the case, selecting a supervised learning model like LSTM would likely yield stronger results. Instead, this thesis served to show whether sentiment analysis was valuable for reinforcement learning, and to prove the technical feasibility of Deep Q-Networks in a novel application, with the goal of promoting future research

in this area. For this reason, this thesis did not experiment with different model parameters or architectures, leaving optimization as a piece of future studies.

1.3 Research questions

Breaking down the research objectives, there were four categories of questions posed before research began:

It is clear from observation that there is a correlation between social media discussion of cryptocurrency and the price of the underlying cryptocurrency. Can causality be determined between sentiment and price? If so, what is the lag between one variable and the other? Further, can an indicator like the number of people talking about a cryptocurrency be used for the same comparison as social media sentiment?

Second, what are the technical requirements to collect sentiment data and train a Deep Q-Network for prediction? In order to create a very applied thesis, clearly detailing system requirements should be a direct output of this research

Third, can a Deep Q-Network model generate profit trading cryptocurrency? Ultimately, it is unlikely that a reinforcement learning model can outperform traditional supervised learning models on financial data, but it is valuable to understand if profit is generated at all in order to act as a valuable trading strategy decision metric.

Finally, how important are sentiment scores and social media discussion volume as feature input to a Deep Q-Network with regards to profitability?

2 Literature Review

The keywords searched for in the literature review were summarized thematically into the following sections. All searches included some variation on the keywords "Cryptocurrency", "Bitcoin", "Ether", and "Ethereum".

1. Social media sentiment and cryptocurrency prices, where keywords including "Sentiment Analysis", "Social Media Prediction", and "Market Sentiment" were used.
2. Sentiment analysis and ANNs for cryptocurrency price prediction, where focus was placed more heavily on using ANNs with sentiment analysis as feature input. Keywords included the previous list, as well as "Deep Learning", "Neural Network", "LSTM", and "RNN".
3. Reinforcement learning for cryptocurrency price prediction, where keywords included "Reinforcement Learning", "Q-learning", "Stock Markets", and "Price Prediction".

A handful of the most important papers from each theme were selected and discussed below.

2.1 Social media sentiment and cryptocurrency prices

In an era where social media usage has exploded and cloud computing has made NLP more accessible, sentiment analysis using social media data as input has become a major focus of current academic papers. One application with significant focus is to use sentiment analysis to understand financial markets – cryptocurrency markets included, although research predominantly focuses on Bitcoin. One of the most cited works is the 2018 paper "How Does Social Media Impact Bitcoin Value? A Test of the Silent Majority Hypothesis" (Mai, Shan, Bai, Wang, & Chiang, 2018), where the authors showed that trends in social media posts supporting purchasing bitcoin are associated with higher future bitcoin values. This correlation was strongest among posts from the 95 percent of users who were less active and whose contributions amounted to less than 40 percent of total messages. Importantly, focused cryptocurrency forums had a stronger predictive power on future cryptocurrency prices compared to Twitter. One gap identified in their methodology however was to use more complex causal inference tools to justify their claims for predictive power. That being said, the journal article still suggested that social media sentiment was an important predictor in determining Bitcoin prices, even if the source of the sentiment varied in importance. Because this thesis used big data as the source for sentiment analysis, it significantly limited the possibility of extracting data from specialized cryptocurrency forums, where the traffic of posts may not have been significant enough to qualify as big data. Further, extracting huge corpuses of data from a variety of discreet forums would have required significant effort, outside the scope of this thesis. That being said, the authors still clearly established platforms like Twitter as valuable predictors for Bitcoin price, even if less effective than subject-specific forums.

Selecting Twitter as a data source was also supported by "Does Twitter predict Bitcoin" (Shen, Urquhart, & Wang, 2019), where the authors compared the number of Tweets to trading

volume and volatility of Bitcoin. The authors used Granger causality tests to show that the number of Tweets were an important predictor. It was interesting therefore to see if Granger causality also appeared for Ether price changes. Given how closely Ether and Bitcoin were visually correlated, I predicted that I would find similar results. The paper also affirmed the methodology of using Granger causality testing on cryptocurrency data.

A related paper was "Coin Market Behavior using Social Sentiment Markov Chains" (Kim, Lee, & Assar, 2021), which explored social media sentiment's effect on cryptocurrency by treating market behaviour as a Markov chain, with this concept supported by other published research including (Ballis & Drakos, 2020), (Li, 2021), (Nascimento, Santos, Jale, Júnior, & Ferreira, 2022), and (Ramadani & Devianto, 2020). This approach was interesting, as it affirmed the possibility of using Deep Q-Learning to predict cryptocurrency prices, since Q-learning is based on Markov Decision Processes. The authors distinguished the effect of social sentiment during a bull versus a bear market. They used a similar approach as the one proposed in this thesis, by extracting social media sentiment from Twitter, once again validating the methodology in this paper. The authors found that social sentiment was more relevant at predicting cryptocurrency prices during a bull market than during a bear market. This research was interesting, because most existing cryptocurrency research considered market performance since inception, or within some set number of years. Up until late-2021, cryptocurrency had seen a a strong bull market. This meant that the findings of this thesis, which included test data after 2021, may not have had the same predictive power of sentiment analysis on cryptocurrency prices as previous studies, but it could be interesting to compare against them. One significant limitation identified was the use of social media data only from South Korea – although a relatively large country, the vast majority of cryptocurrency investors come from English-speaking countries (Goswami et al., 2021). It should be cautioned that over-generalizing based on an culturally-biased data subset may have led to incorrect assumptions. As discussed in the conclusion, this thesis was not above this flaw either; only English data was collected, but the hope was that the enormous volume of data would make cultural bias more fuzzy. Social sentiment data was collected by crawling Bitcoin-related posts on Twitter.

2.2 Sentiment analysis and ANNs for cryptocurrency price prediction

(Critien, Gatt, & Ellul, 2022) extended the work of (Abraham, Higdon, Nelson, & Ibarra, 2018) to predict not only the direction of cryptocurrency movement, but the magnitude of change. They chose to use both the sentiment of Tweets as well as the volume of Tweets. They used two types of ANNs for the goal of identifying price change: convolutional neural networks (CNNs) and recurrent neural networks (RNNs), the two most popular ANNs for financial market prediction. They then added an additional multi-classification model to predict the magnitude of change. Their approach affirmed that neural networks were capable of relatively high accuracy in price prediction. The same methodology of collecting both Twitter sentiment scores and number of Tweets as features was used in this thesis, as they showed the latter to be an important feature

of their data.

"Cryptocurrency Price Prediction Using Tweet Volumes and Sentiment Analysis" (Abraham et al., 2018) was an especially interesting paper as the authors looked at not only Bitcoin but also at Ether for price prediction. It was therefore important input for this thesis, as it showed that Ether exhibited similar behaviour under analysis as Bitcoin. However, the authors neglected to show the results of training and testing their model on Ether, leaving room for this thesis to add to their research.

"Time Series Analysis of Cryptocurrency Prices Using Long Short-Term Memory" (Fleischer, von Laszewski, Theran, & Bautista, 2022) built on (Kwon, Kim, Heo, Kim, & Han, 2019) and (Critien et al., 2022) (previously discussed) by building an LSTM model for cryptocurrency price prediction using Bitcoin, Ether, Dogecoin, and EOS. Unlike the data used in this these, they included only closing price values in their model as feature input. Their results showed that using a neural network performed better than ARIMA models for all cryptocurrencies, although the lowest improvement over ARIMA models was for Ether, at +13.9% root-mean-squared error. This was an interesting result for this thesis, because if Deep Q-Learning can generate meaningful returns, it may be able to compete with supervised learning models like LSTM.

Finally, CNNs have become especially prominent for price prediction of cryptocurrency, as highlighted in "CNN-based multivariate data analysis for bitcoin trend prediction" (Cavalli & Amoretti, 2021) and "An advanced CNN-LSTM model for cryptocurrency forecasting" (Livieris:2021, Kiriakidou, Stavroyiannis, & Pintelas, 2021). In both papers, the authors used CNNs to generate state-of-the-art returns in bull markets. Although CNNs were not be applied in this paper's methodology, it was important to mention them in the literature review as they mark the highest performance for ANNs in cryptocurrency price prediction – and as such, they could be integrated in future Deep Q-Learning research by adding convolutional layers to Deep Q-Networks.

2.3 Reinforcement learning for cryptocurrency price prediction

Limited literature existed that discussed applications of reinforcement learning for price prediction. Of the existing research, most was quite recent, with papers published in the last two years.

"Deep reinforcement learning for the optimal placement of cryptocurrency limit orders" (Schnaubelt, 2022) examined a number of state-of-the-art reinforcement learning algorithms not to determine price movement, but to decide where at what price to place limit orders for Bitcoin and Ether. The algorithms they selected are backward-induction Q-learning, deep double Q-networks (applied in this thesis, with the addition of a dueling architecture), and proximal policy optimization. Like this thesis, the authors leveraged big data to build their feature set, with 300 million historic trades and more than 3.5 million order book states from major exchanges and currency pairs. However, unlike this thesis, they did not consider social media sentiment in any respect, nor did they attempt to optimize both purchases (market orders, rather than limit orders) and sales (market sales). The results of the paper showed that reinforcement learning could

indeed be used to generate profits from a cryptocurrency portfolio, although Deep Q-Learning fell behind proximal policy optimization as an optimal strategy.

"Recommending cryptocurrency trading points with deep reinforcement learning approach" (Sattarov et al., 2020) used a deep reinforcement learning algorithm similar to Deep Q-Networks. The model the authors built classified results as either a price increase, decrease, or no move in cryptocurrency data, using a 5-layer densely-connected neural network, with a single output node denoting the best action to take. The trading agent in their model achieved 14.4% net profits within one month of Bitcoin trading during a bull market, and an impressive 41% profit for Ether in the same market. Their research suggested that Deep Q-Learning could likely exhibit similar performance, although there were some notable gaps in the methodology of the researchers. First, the models were not back-tested on bear market time series, and second, the dataset was limited to pre-2020, before the huge volatility of cryptocurrency markets took effect. It was necessary therefore to extend the research into present day, using post 2020-data for training and testing.

The only meaningful research published that uses Deep Q-Learning to predict when to buy and sell cryptocurrency at market prices was "Deep Q-learning for Trading Cryptocurrency" from the Journal of Financial Data Science (Ma, Wang, & Fleiss, 2021). The authors used three different cryptocurrencies in their model: Bitcoin, Ether, and Litecoin (the last one being a fork from Bitcoin), and achieved portfolio returns of 66% over 2000 episodes. However, the time series selected for training and testing was mostly during a strong bull market, and as such, the Deep Q-Network was not evaluated in the current recession of cryptocurrency prices. The authors also noted that the variance between episodes was very significant, due to the high volatility of cryptocurrency prices and Deep Q-Network's stepwise nature. The authors concluded that Deep Q-Learning for cryptocurrency prices warranted further research, and this thesis extended their work by adding sentiment analysis as a feature, using more data, and data during a bear market.

2.4 Literature discussion and gaps

Based on previous research, sentiment analysis and social media volume metrics would likely play an important role in understanding price fluctuations in financial markets. Because cryptocurrencies (more than equity markets) are strongly tied to investor sentiment for future rewards, and as social media usage continues to grow, sentiment analysis should display important trends that could be used to predict cryptocurrency prices.

Next, it seemed that reinforcement learning was unlikely to provide strong financial returns when used as a trading tool. This was primarily because of the strong volatility of markets and the lack of control an algorithmic trading agent can exert on its environment. I hypothesized therefore that a Deep Q-Network could likely be built and trade as competently as an uninformed trader, but not generate any substantial returns.

Based on my literature review, there were some key gaps in existing research to highlight. First, reinforcement learning was rarely used in financial market applications outside of portfolio management. Deep Q-Learning especially saw very few practical applications – because of its

novelty, it was still heavily rooted in academia. Most applied examples for Deep Q-Networks had traditionally been to mimic humans playing videos games, or to train human-like robots. This thesis therefore explored a relatively unknown application of Deep Q-Networks on financial data. Proposing alternative, practical use cases for the algorithm (as done in this paper) may contribute to the wider adoption of reinforcement learning in financial institutions as a valid alternative supervised machine learning and technical analysis.

Another key gap valuable to point out was the abundance of research on Bitcoin and not smaller coins like Ether. Although the second biggest coin, fewer papers referencing Ether or similar cryptocurrencies had been written by a factor of nearly ten. It was therefore valuable to add an additional perspective to cryptocurrency research by exploring a secondary (but still widely popular) coin.

Next, papers that studied social media sentiment scores of cryptocurrency focused primarily on the correlation between sentiment and price, and did not create end-to-end practical models to predict cryptocurrency prices. With Deep Q-Networks specifically, there was no research that combined this type of reinforcement learning with sentiment, volume. and historical price as feature inputs. As such, this thesis explored a previously unresearched combination of features and machine learning model.

Additionally, the goal of this thesis was to be deeply applied, by including sections dedicated to explaining how the model was developed and deployed. Many prior research papers neglected implementation steps, hyper-focusing on algorithm development and results. This paper should serve as a starting point for further research using Deep Q-Networks in financial applications. In the methodology, industry-standard tools and technology were selected (e.g. Python, Spark, and Tensorflow) to facilitate adoption in companies' existing technology infrastructure. Very few academic papers detailed their hardware/software environment, with fewer still describing considerations for cloud deployment. In contrast, this thesis detailed a practical approach to training and deploying the deep learning model. Finally, using a cloud environment maximized the accessibility of this work by allowing real-time scaling of hardware resources with minimal capital expenditure. This meant that businesses or individuals who do not own adequately strong computing hardware to locally train and deploy the Deep Q-Network model can follow the methodology in this paper and do so in the cloud.

Next, this thesis used higher volumes of data than other papers had used in their research. By leveraging big data tools like Apache Spark, hourly financial data was collected and analyzed instead of daily data, and millions of Tweets were used for sentiment analysis where other papers examined only thousands.

Finally, the vast majority of research on cryptocurrency markets were written during a strong bull market. Although there had been occasional price corrections (e.g. January 2018, March 2020, May 2021), cryptocurrency prices had until recently trended upwards, meaning that most trading algorithms developed would perform well on average, regardless of actual algorithmic merit. Given the recent market crash from November 2021 to date (reaching 24-month lows in

June 2022), one goal was to see if a Deep Q-Network model could still be profitable in a recession economy.

3 Methodology

3.1 Hardware/software environment

Two distinct cloud environments were used in this project – one to prepare the data, and another to train the machine learning model.

Environment 1

- Hosting Service: Microsoft Azure
- Operating System: Ubuntu 18.04 LTS
- Software: Databricks Runtime 10.4 LTS
- Cluster configuration:
  - Mode = Standard
  - Worker Type = Standard_D12_V2 (28 GB memory, 4 vCPU cores) [min workers 2, max workers 8], autoscaling enabled
  - Driver Type = Standard_D12_V2 (28 GB memory, 4 vCPU cores)
- Python Version: 3.8.10
- Spark Version: 3.2.1
- Spark NLP Version: 4.0.2

Environment 2

- Hosting Service: Microsoft Azure
- Operating System: Ubuntu 20.04 LTS
- Virtual Machine configuration: Standard_NC6s_v3 (112 GiB memory, 6 vCPU cores, NVIDIA Tesla V100 GPU)
- Python Version: 3.10.2
- EViews Version: 12, July 19 2022 build
- Tensorflow Version: 2.9.1 (GPU distribution)

The first environment served to collect, clean, and process the big data used as input. It was also used to run Spark NLP sentiment analysis and Granger causality analysis between Ether price data and social media sentiment. Databricks software was deployed in Microsoft Azure's cloud to create a cluster of virtual machines and manage Spark jobs. See *Azure Databricks Setup* in the Appendix for detailed instructions on how set up this environment in Azure.

The second cloud environment was used to train and test the DDDQN model and for Granger causality testing. A single GPU-attached Microsoft Azure virtual machine was used. The reason data collection and processing were performed in a separate environment from training was simply for cost management; DDDQN network training with Tensorflow would not benefit from a distributed Spark workflow, and maintaining a GPU cluster with Databricks was more expensive than provisioning a single GPU-attached virtual machine.

3.2 Cryptocurrency selection

Selecting an appropriate cryptocurrency was an important caveat of this project. Ultimately, Ether (ETH) was selected for this research, which was the best option based on my three-pillar selection framework. The options were main cryptocurrency coins (Bitcoin and Ether), mid-sized coins (e.g. Tether, Binance Coin, XRP, Cardano), or one of many smaller start-up coins.

The first feature of my selection framework was that the cryptocurrency had significant trading volume with reasonable variance. Many smaller coins simply did not have enough purchases and sales to show consistency in hour-by-hour price changes. The inherent variability with low volume trading would have made training a reinforcement learning model unstable. On the flip side, stable coins that tracked the US dollar were also not interesting for research (Tether and USDC for example), since their price was pegged to an existing asset with limited variance that did not fluctuate with supply/demand of the cryptocurrency itself. Of the two large coins, Ether and Bitcoin, Ether displayed more stability that Bitcoin during cryptocurrency market price shocks. A more stable price graph would likely result in better Deep Q-Network learning.

The second feature was that there was sufficient real social media discussion to establish statistically resilient and accurate trends in sentiment. One of the biggest issues discovered exploring alt-coins was that it was difficult to verify the veracity of Tweets. From anecdotal experience, it seemed that smaller start-up coins had a much more vocal positive social media following – Reddit's cryptocurrency subreddits for example were inundated with daily posts about new coins hitting the market, often with strong sales pitches emphasizing reasons to buy them. Unfortunately, many of these coins were tools for malicious actors to exploit new cryptocurrency investors, as the coins had no real market value. Their prices were artificially inflated by intense buying by a small number of people, followed by a rapid coordinated sell-off by the majority wallet holders. (Liebau & Schueffel, 2019) These malicious actors used bots and automated scripts to post positive messages about their scam coins. For this reason, smaller, newer coins were not considered. Further research could extend the sentiment analysis portion of this thesis to predict which coins are scams and which have merit (with real people discussing them on social media rather than bots). Bitcoin and Ether both had the largest collections of social media posts on Twitter, making them prime candidates for research.

The last pillar of my selection framework was to select a cryptocurrency with academic merit to research. Although mid-sized coins fit the previous two criteria, their long-term success was questionable. Longer standing coins with a higher likelihood to remain successful were more

interesting should this research be used in the future. Of the two largest coins, Bitcoin and Ether, the latter is traded on the Ethereum network, which has more usability that Bitcoin by natively supporting smart contracts. Further differences between the two cryptocurrencies were well-described in "An Overview of Ethereum & Its Comparison with Bitcoin" (Jani, 2017). In short, I believed that Ether had a stronger value proposition than Bitcoin due to its higher applicability, and as such would become more valuable in the long term. Lastly, most academic papers that researched cryptocurrency defaulted to Bitcoin. This thesis provided a new perspective on the huge cryptocurrency market by focusing on another, similarly large but under-researched cryptocurrency.

3.3 Financial data collection

Collecting the historic prices of Ether was straightforward as there were native Python packages that allowed historic price APIs to be queried for this information. The open-source Historic-Crypto package by David Woroniuk (Woroniuk, 2021) was used, which queried the Coinbase Pro API for hourly price data of Ether. For each time step, the data contained the opening price, closing price, high price, low price, and volume. All the features were kept, as they all had merit as input to a neural network. The open and close price were valuable to identify a trend in intra-hour pricing, the high and low price summarized the variance within each hour, and the volume could itself be a valid predictor for future price.

3.4 Social media text collection

Apache Spark allowed data processing to be scaled across multiple virtual machines, significantly speeding its performance especially with a large dataset. First, Twitter data was collected. There were two approaches that could be used for this: the official Twitter API, or a third-party Python scraping library like snscrape or Twint. The official Twitter API was significantly limited in the number of Tweets that could be queried without an application for Academic Research, and was also limited only to recent Tweets (generally the last 7 days). On the other hand, a scraping solution was much slower at collecting data, since it mimicked a human scrolling a Twitter search and could only collect Tweets as it virtually scrolls. The data returned was also missing some of the richer metadata that the official Twitter API included in its responses.

Ultimately, the Academic Research level of access to the official API was the best option, as it allowed for the full archive of Tweets to be queried, and 10 million Tweets to be collected per month. Important to note however is that the process of applying for and receiving access took many weeks, and approval of the request was not guaranteed.

With the authentication keys for Twitter's V2 API endpoints, the Tweepy library was used to query English language Tweets containing the keywords (#ETH OR $ETH OR Ether OR Ethereum), filtered from January 1, 2017 00:00, to July 18, 2022 00:00. This range was specifically selected as it represented the highest volatility and volume in Ether trading (pre-2017, Ether was

relatively unknown and stable in price, which would not have significantly contributed to training a model or provided any valuable insights from reinforcement learning). Collecting these Tweets took a significant amount of time as the API was limited to 1 request per second, with a maximum of 100 Tweets returned per request. There were over 145 million Tweets in the selected period, so the Python script was configured to query a set representative proportion of Tweets per hour, with approximately 5 million Tweets collected in total. This meant that 4% of the Tweets each hour were collected. In order to ensure there was enough data in each hour of the data, a minimum number of Tweets to collect was set to 100. This meant that between 2018 and 2020, when Ether was relatively unpopular, a large enough sample of Tweets was still collected for accurate sentiment analysis. In total, 5,675,203 Tweets were collected.

After running the API querying script for around 28 hours, the Tweets were exported into a UTF-8 encoded CSV file. Although other data format types including parquet files alloed for smaller file sizes, interoperability was the key focus when exporting data to allow different software to visualize results, where nonstandard file types may have been unsupported. The CSV file was loaded into the Databricks FileStore directly via the Azure Blob Storage interface (to circumvent the Databricks 2GB local I/O API's 2GB file limit), then processed from UTF-8 encoded strings to Python datatypes using the Abstract Syntax Trees standard library. Once completed, the Tweets and their metadata were loaded into Spark dataframes using PySpark, with the following schema displaying the data and metadata retained:

```
root
 |-- id: long (nullable = false)
 |-- created_at: timestamp (nullable = true)
 |-- author_id: long (nullable = true)
 |-- conversation_id: long (nullable = true)
 |-- source: string (nullable = true)
 |-- geo_coordinates: array (nullable = true)
 |    |-- element: double (containsNull = false)
 |-- geo_place_id: string (nullable = true)
 |-- mentions: array (nullable = true)
 |    |-- element: long (containsNull = false)
 |-- hashtags: array (nullable = true)
 |    |-- element: string (containsNull = false)
 |-- urls: array (nullable = true)
 |    |-- element: string (containsNull = false)
 |-- text: string (nullable = true)
```

Figure 3: Spark Dataframe schema for Twitter data

See the Appendix for the output displayed when calling Pyspark's *show()* method on the dataframe.

Ultimately, only the aggregate sentiment per hour and number of Tweets per hour were used as features for the DDDQN, with the remaining metadata used for exploratory analysis and data visualization.

3.5 Sentiment analysis

There were many different NLP model architectures to select among for sentiment analysis, but the focus of this research was to apply a state-of-the-art method in order to achieve the most accurate results for this research. Cutting edge NLP models natively supported by Spark NLP primarily fell into one of the three categories: Bidirectional Encoder Representations from Transformers (BERT) and BERT extensions (ALBERT, RoBERTa, DistilBERT, Universal Sentence Encoder with CMLM), XLNet, and ELMo. Ultimately, a BERT model fine-tuned on financial communication text, named FinBERT (Huang, Wang, & Yang, 2020) was selected, as it was expected to yield the highest relative performance given its pre-training dataset.

BERT is an NLP pre-training model created by Google employees that achieved state-of-the-art performance when released. It serves as the foundation for many of the more complex NLP algorithms achieving state-of-the-art results today. It uses a transformer-based machine learning technique, created in 2018 and formally published in 2019 (Devlin, Chang, Lee, & Toutanova, 2019)

From the abstract of their paper: "Unlike recent language representation models, BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications. BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement)."

Since its development, BERT had become ubiquitous in the NLP space for both academic and professional application. Google announced on Twitter in October 2019 that it had begun using BERT for US English web searches (Google-SearchLiaison, 2019b) and posted another announcement December the same year that BERT was rolling out to over 70 languages worldwide (Google-SearchLiaison, 2019a).

BERT's architecture differs quite significantly from previously-preferred RNN models. At its core, BERT is a set of Transformer encoder neural network layers (Vaswani et al., 2017), each with multiple self-attention "heads" (terminology used in the original paper). Transformers are novel neural network models that can process words in relation to all the other words in a sentence, rather than individually in sequential order. BERT models can therefore consider the full context of a word by looking at the words surrounding it.

Before transformers, neural networks usually processed language by creating vector-space representations (e.g., word2vec based on "Efficient Estimation of Word Representations in Vector Space" (Mikolov, Chen, Corrado, & Dean, 2013)). Reading one word at a time, RNNs must perform multiple steps to make decisions that depend on words far away from each other, and prior research has shown that the more such steps decisions require, the harder it is for a recurrent network to learn how to make those decisions. (Devlin et al., 2019). Sequential training like this also does not benefit from the significant speed advantages of GPUs and TPUs which excel in parallel floating-point processing.

Transformers on the other hand process text data all at once, with a small number of iteration steps. At each step, a transformer model uses a self-attention mechanism which directly models relationships between all words in a sentence, regardless of their position. This combats RNN's issue with finding connections between words far away in a sentence. For example, suppose a BERT model is fed the sentence "The player threw the baseball at the pitcher". To decide that the word "pitcher" is referring to the position on a baseball team and not a receptacle for water, the transformer model can learn to immediately pay more attention to the word "baseball" and make this decision in a single step.

To compute the representation of a given word, the transformer compares it to every other word in the sentence. The result of these comparisons is an attention score for every other word in the sentence. Taking the previous example, these attention scores determine how much each of the other words should contribute to the representation of "pitcher". "Baseball" would receive a high attention score, while "threw" would not. The attention scores for each word in the sentence are then used as weights for a weighted average of all words' representations. This is then fed into a fully-connected neural network to decide a new representation for "pitcher", showing that the sentence is talking about a baseball pitcher.

In a BERT model, each self-attention "head" of the transformer encoder layers computes key, value, and query vectors for every input token in a sequence of words. Then, as mentioned, it creates a weighted representation of the input. The outputs of all heads in the same layer are combined and run through a fully connected layer, skip connection (to skip over some layers of the network), then the layers are normalized. As the FinBERT model used in this paper was trained additionally on financial text data, additional fully connected layers were added on top of the final encoder layer.

The base English-language BERT models used 12 encoders with 12 bidirectional self-attention heads and was pre-trained from unlabeled data extracted from two sources. The first is BooksCorpus, a collection of 800 million words extracted from a large collection of free novel books written by unpublished authors (Zhu et al., 2015). The second is English Wikipedia with 2,500 million words (Devlin et al., 2019). As previously mentioned, FinBERT extends BERT's training corpus with financial communication text from the following three financial communication corpus:

- Corporate Reports 10-K & 10-Q: 2.5 billion tokens
- Earnings Call Transcripts: 1.3 billion tokens

- Analyst Reports: 1.1 billion tokens

As the application for the model in this thesis was sentiment analysis, the FinBERT-tone release of the model was selected, which further fine-tunes the FinBERT model on 10,000 manually annotated (positive, negative, neutral) sentences from analyst reports, achieving superior performance on financial tone analysis task. Because it was pretrained on financial text, many social media posts with positive or negative sentiment about price movements were more accurately captured.

While training a custom BERT model would of course have been possible, finding a labelled dataset (or labelling one) using cryptocurrency Tweets was out of scope of this thesis. The focus for this research was primarily about Deep Q-Learning for cryptocurrencies, so a pretrained model was selected to avoid bloating the scope of this paper. Of the pretrained models available in Spark NLP, FinBERT was the most popular and trained on the largest meaningful dataset for this use case.

The FinBERT model was loaded into a Spark NLP pipeline so that the computation could be performed across 8 worker nodes. The FinBERT pre-trained pipeline was used to process the Tweet text and generate an overall sentiment score for each Tweet. The results were then cleaned using Spark NLP's *Finisher()* method. Distributed processing significantly reduced the time it took to classify 4 million Tweets, to only 12 hours. Finally, the average sentiment and Tweet volume columns by hour were summarized using PySpark's SQL functions, making them ready for Deep Q-Network training alongside financial data.

3.6 Feature engineering

The first step of feature engineering was to merge the financial data with the sentiment data. Both were subset by hour with a timestamp as the index, so combining them was trivial. The resulting features dataset for the model is summarized in the Results Analysis section.

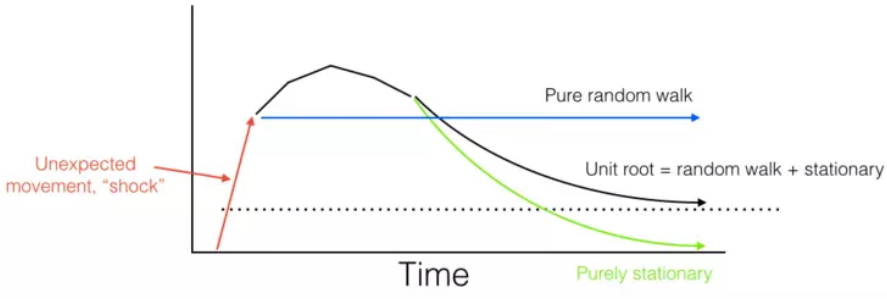
Neural networks benefit from cleaned and scaled data. Although the data did not contain any missing values, the scale was not standardized. As such, the last important step in feature engineering was to use a Min-Max Scaler to scale the values of the features dataset in the range [0, 1].

3.7 Granger causality

With hourly sentiment scores and hourly financial data collected, Granger causality testing could be performed. The data was exported from Spark notebook to EViews, an industry-standard and feature-rich software solution for econometric time series data analysis and processing. The goal of causality testing was to understand whether sentiment has predictive causality for closing price, or vice-versa. It was also to determine if the number of Tweets was caused by closing price or sentiment, or inversely if either of those two variables were the cause for the number Tweets.

The data was explored in EViews to understand correlations between variables, which was summarized in the results analysis. There were some important assumptions that must have been considered true before pairwise Granger causality tests could be performed: first, variables were stationary, and second, that they were linearly cointegrated. The latter assumption was inherent since the closing price and Tweets are about the same subject, Ether. The former assumption required some data manipulation before the time series could be considered stationary.

As a precursor to this methodology, it was important to understand unit roots, and what they meant for stationarity. Unit roots are generalized in this diagram:



(Cochrane, 2015)

Figure 4: Effect of a unit root in a time series after a shock

A unit root is a feature of stochastic processes, where the process has a unit root if 1 is a root of the process's auxiliary equation. A detailed explanation of unit roots and the reason why they suggest non-stationary data was outside the scope of this thesis; for a thorough explanation, consult (Baumol, 1970).

There are two tests that can be done to ensure stationarity by considering a unit root, which approach the problem from two different angles:

1. The Augmented Dickey–Fuller (ADF) test (Dickey & Fuller, 1979), whose null hypothesis is that the time series has a unit root (which means the data is non-stationary). The ADF statistic computed during the test is a negative number, where the more negative it is, the stronger the rejection of the hypothesis that there is a unit root. In order to ensure stationarity in the data, the goal therefore was to **reject the null hypothesis** with a p-value close to 0. 5% was used as the alpha level for statistical tests, meaning that the alternative hypothesis of stationarity was considered at the 95% confidence level if the t-statistic from the test was less than the 5% critical value.

2. The Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test (Kwiatkowski, Phillips, Schmidt, & Shin, 1992), whose null hypothesis is that the time series is stationary (with the alternative that the time series has a unit root). With KPSS, the goal was to **consider the null hypothesis** with a p-value far from 0, once again using a 5% alpha level.

The ADF and KPSS tests complemented each other when testing for stationarity. If the ADF test did not find a unit root, but the KPSS test did, the series was difference-stationary. If the KPSS test did not find a unit root, but the ADF test did, the series was trend-stationary. In both cases, taking the first-order differences and repeating the tests was a solution.

Although anecdotally clear the data was not stationary from viewing the plotted time series, the tests were used for numeric verification. Unsurprisingly, the tests failed to show stationarity. To make the data stationary, the first-order differences were taken of the closing price, sentiment scores, and number of Tweets, after which the tests successfully passed. Results from this testing can be found in the Appendix. EView's pairwise Granger causality tests were then performed on the three variables.

With causality testing complete, the next step was to use the data as training input for deep reinforcement learning and build an algorithmic trader.

3.8 Trading rules

Many applications of algorithmic trading for stock market portfolios grant a trading agent a predetermined starting balance and trade until that balance is exhausted. This is a realistic approach when stocks are sold in single units at a fixed price per unit. While cryptocurrencies can be purchased in any fractional amount, the implementation in this thesis still treated a "unit" of cryptocurrency as indivisible. For this reason, a starting balance of $100 was included, and the trading agent was allowed to purchase and sell as many units of cryptocurrency as it desired until it reached the end of the time series or ran out of balance.

The next decision was to grant a reward to the trading agent only during the sell action. This meant that purchasing a unit effectively granted a reward of 0, and selling the unit returned either a positive reward (if the price the unit was purchased at was less than the current market price), or a negative reward (in the contrary case). From initial testing, it was found that training the Deep Q-Network using this approach was much more stable.

Next, the action space of the agent was limited to only three options: Buy, Hold, or Sell. In this way, the agent could only buy or sell a single unit of Ether at each time step. A more complex Deep Q-Network could have increased the action space by $2n$ by allowing the agent to buy or sell n number of units. Alternatively, if the goal was also to decide how much of a cryptocurrency to buy or sell at each time step (instead of the arbitrary unit used here), model-based acceleration Deep Q-Learning (Gu, Lillicrap, Sutskever, & Levine, 2016) or actor-critic reinforcement learning (Lillicrap et al., 2015) allow for modeling in a continuous action space. However, the complexity of the environment would likely prevent the Q-Network from finding an effective trading strategy.

If a training episode was completed with inventory left over, the value of the inventory was added to the overall reward from the episode by "selling" all the units at the market price of the cryptocurrency in the last time step. The added value to the balance was therefore the market price for those remaining units, minus the sum of their purchase prices.

January 1 2017 00:00 to December 31 2021 23:59 was used as the training data period. Before 2017, the price of Ether was quite stable and low, with very few Tweets referencing Ether (generally less than 100 per hour). Its relative unpopularity and slow price movement would likely not have been valuable inputs to the neural network. January 1 2022 00:00 to July 21 2022 23:59 was then used as the testing range. Although it was a fairly small range, it included a high amount of variation, and provided an accurate window to gauge the performance of the model.

Some Deep-Q-Network applications allow the agent to choose an action that the environment does not permit, and sets the reward to 0 for that action. Instead, the trading agent's action set was limited based on the agent's inventory. If the agent's inventory of Ether units purchased was 0, it was prevented from selecting the sell action, even if the neural network determined it to be the most profitable. This was a closer representation of a real-life trading agent, who rather than receiving 0 reward for trying to sell something they did not have, could not have done so in the first place. The inventory of the agent, denoted as I, was a first-in-first-out model, where units of Ether recorded in the inventory were sold in reverse order of purchase. This meant that the first unit of Ether sold was always the most recent unit purchased.

3.9 Experience replay memory

Experience replay memory enhanced Deep Q-Learning by improving how the weights of the neural network were updated based on past experiences. As previously discussed, Deep Q-Learning uses past combinations of states, next states, rewards, and actions ("experiences") to predict optimal future states and actions. At each time step t, the trading agent took an action based on its current state, was granted a reward (either positive or negative depending on the action taken), and the environment moved to the next state if it was not a terminal state. These four elements comprised an experience to store in memory, which was used to train the trading agent's neural network. After making a trading decision, the trading agent's online network was updated to better predict the optimal decision. To do this, the agent's replay memory was sampled and historical experiences were used to estimate the value of the best action to take, adjusting the weights of the network accordingly.

In the code, the trading agent's experiences were stored at each time step in numpy arrays within the ExpReplay class. Although Python natively supported the deque datatype for similar memory buffer applications, numpy arrays tended to be faster, more robust solutions for storing regularly-accessed data in a Tensorflow workflow. Mathematically, the agent's experience at time t could be denoted as the tuple e_t:

$$e_t = (s_t, a_t, r_{t+1}, s_{t+1}, d_t) \tag{11}$$

e_t contained the state of the environment, s_t, the action taken from that state, a_t, the reward given to the agent as a result of the state/action pair, r_{t+1}, the next state of the environment, s_{t+1}, and whether the experience is a terminal state, d_t. The experiences were stored in replay

memory for later sampling, up to a maximum memory size (denoted as N). This memory size cap prevented the trading agent from sampling experiences that were too old and which may have lost their predictive importance.

3.10 DDDQN model architecture

The next two pages show a detailed diagram of the DDDQN model used in this thesis, and the Tensorflow layers of the online network as summarized by the compiled model (in the first trial, without sentiment features).

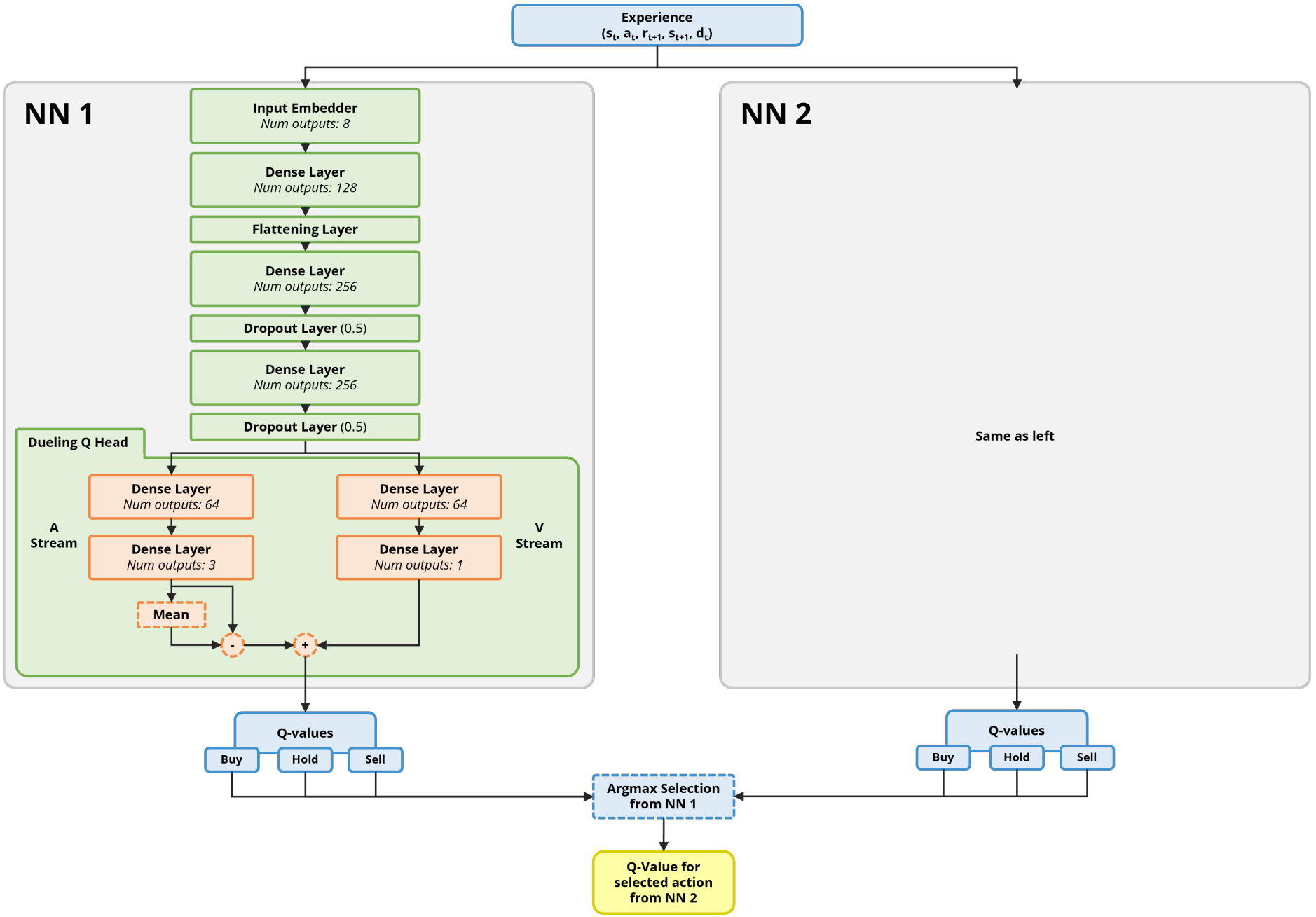


Figure 5: Dueling Double DQN model architecture

```
self.online_net.summary()
Model: "dddqn"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                multiple                  768

flatten (Flatten)            multiple                  0

dense_1 (Dense)              multiple                  1573120

dropout (Dropout)            multiple                  0

dense_2 (Dense)              multiple                  65792

dropout_1 (Dropout)          multiple                  0

dense_3 (Dense)              multiple                  16448

dense_4 (Dense)              multiple                  16448

dense_5 (Dense)              multiple                  65

dense_6 (Dense)              multiple                  195

=================================================================
Total params: 1,672,836
Trainable params: 1,672,836
Non-trainable params: 0
_________________________________________________________________
```

Figure 6: Keras *summary()* method called on the compiled online network

The DDDQN started with an input layer containing 8 nodes (one per feature of the state). The input nodes accepted data as an (64, 48, k) dimension numpy array, where 64 represented the batch size, 48 represented the window size and k was the number of features (5 without sentiment data, and 7 with sentiment scores and number of Tweets included). The data was then passed to a 128-node densely-connected layer, followed by a flattening layer to eliminate the second dimension. This meant that the data was now structured as a (64, 6144) matrix, which was necessary to allow the model to predict the value of actions taken at the current time step only, and not for each time step of the 48-hour window.

Next, the flattened data was passed through two 256-node dense layers, with a dropout layer of 0.5 after each one (a regularization method to prevent overfitting). Then, the network was split into the dueling architecture, where both branches had an additional 64-node dense layer. The state value stream terminated at a single-node output layer (representing the value of the state), and the action stream terminated at a three-node output layer (where each node represented the

value of an action in the action space: buy, hold, or sell).

Finally, as explained in the introduction to Dueling DQNs, the state value and action values were linearly combined to create a single expected q-value for each action.

The relatively high number of hidden layers and nodes of the DDDQN were incorporated to try to capture small nuances in highly-volatile data. Additionally, because the dataset was so large, it was unlikely that the neural network would memorize (and thus overfit) the data.

3.11 DDDQN parameters

DDDQNs have a number of hyperparameters that can be tweaked to change learning performance. Many of these were explained in prior sections of this thesis.

First, the discount rate γ was set to 0.95. As a reminder, the discount rate was an exponential function that decided how much future rewards effected decision choices. When $\gamma^n = 0.5$, the reward at time $t + n + 1$ was half as important as the reward at $t + 1$ (the immediate reward). For example, if $\gamma = 0.9$, this was nearly 7 steps, but with $\gamma = 0.99$ it was closer to 70 steps. This meant that for $\gamma = 0.9$, the reward in about 7 steps was half as important as the immediate reward, but for $\gamma = 0.99$, the same was valid for about 70 steps. Because the dataset was quite granular with small time steps (one hour), and significant changes in cryptocurrency markets generally happened over multiple days, a relatively high γ rate of 0.95 was selected. It was important to avoid setting it any higher however, as the high variability of markets meant that immediate rewards remained more important than future rewards as it was difficult to predict what the future rewards may have been. The agent was therefore empowered to select an action that generated more immediate value, unless it was certain that future rewards outweighed this benefit.

Next, ϵ started at 1 (typical in most reinforcement learning applications) but decayed exponentially until it reaches its minimum value in the final time step of the final episode. It therefore decayed more quickly in earlier episodes before slowing to the minimum ϵ value of 0.02. Because the model could never have perfectly predicted future states and rewards, it always had the opportunity to select an exploration instead of exploitation action. Given the intense volatility of cryptocurrency markets, it was better to ensure that the model could continue to experiment even after ϵ had converged on a minimum value.

The window size was set at 48 time steps, meaning that the neural network was fed the last two days of data at each training step. A smaller window size would have been applicable for daily data, but because hourly data was being used, a larger window was more appropriate. Because cryptocurrency market trends tended to propagate over days rather than hours (barring sudden shocks), having access to a larger window of time led to more effective learning. Increasing the window size further could have been detrimental however, as the high variability in historical data may have prevented the neural network from discovering meaningful patterns or trends in too much noisy data. A larger window size may have been effective in a more complex densely-layered neural network, and could be explored in future research.

N was set to 100000 as the size of the replay memory, since experiences from trading did not lose significant predictive importance over time. As such, it was not necessary to significantly limit the number of past experiences the model could learn from.

In neural network applications, the learning rate for gradient descent scales the amount network weights are updated in order to minimize a loss function. The learning rate for the model was set at 0.01, which after some naive testing at different magnitudes provided the best results. It was necessary to ensure that the data would converge without becoming stuck at a local minimum.

As suggested in many Deep Q-Network papers and extensions, the Adam loss function was used, details for which can be found in the source paper "Adam: a method for stochastic optimization." (Kingma & Ba, 2014). In short, it is an improvement on stochastic gradient descent used to optimize an objective function, which became popularized in machine learning applications due to its simplicity and speed. Note that the original Deep Q-network paper used stochastic gradient descent.

Finally, the update interval for the target network was set to 96 time steps, meaning that the weights of the online network were copied to the slower-moving target network after it had seen four days of data.

3.12 DDDQN training process

With the online and target neural networks defined, the Python training models could then be fed data. A trading agent object was created using the aforementioned parameters, which started with a balance of $100 (an arbitrary figure, selected as a "stop losses" number to prevent the trading bot from making a long series of poor decisions). Then, the agent began processing each row of data in the dataset. Because the window size was 48, the first row of data was duplicated $48 - t$ times when $t < 48$, where t was the current time step. This gave the agent a synthetic "history" for its window that assumed the price of Ether was stable before the first row of data.

The python script looped through the data 30 times, each step representing an episode of training. The procedure in each loop was as follows:

1. The trading agent observed the state of the environment, s_t (the row of data corresponding to time step t and the preceding 48 rows)

2. The trading agent chose an action to take. A random number x between $[0, 1]$ was generated: if x ¡= ϵ, a random action was selected. On the other hand, if $x > \epsilon$, the state at time t was fed to the online network for prediction of q-values for each action; the agent then chose the maximal action. In either situation ($x > \epsilon$ or $x <= \epsilon$), if the agent had an inventory $I > 0$ (meaning there was at least one unit of cryptocurrency the agent had purchased but not yet sold), the action space was $[0, 1, 2]$ where 0 represented the sell action, 1 was the hold action, and 2 was the buy action. If the inventory was empty, the action

space was shrunk to $[1, 2]$. Action selection began randomly, but became more informed as ϵ decayed and the trading agent learned more about the environment.

3. The trading agent observed the next state, $s_t + 1$, which was required to train the neural networks based on Bellman equation estimations (see equation (7))

4. The trading agent executed the selected action:

   (a) If the action was 0, sell, one unit of inventory I was removed. A reward r was granted to the trading agent computed as the current close price of Ether at time t minus the price for which the unit of Ether was purchased. If the trading agent bought Ether at \$1 per unit at some time $t - n$, and the close price at time t was \$1.20 per unit, the reward was be \$0.20.

   (b) If the action was 1, hold, no change was made and the reward was 0.

   (c) If the action was 2, buy, the agent added one unit of Ether to its inventory I and recorded its price. As discussed in *Agent Rules*, the reward for this action was 0.

5. The reward was added/subtracted from the agent's balance. If the balance was below the current closing price of Ether at time t, the simulation was considered to have reached a terminal state where the trading agent could no longer buy new units of Ether. In this circumstance, the training loop was exited and the "done" flag is set to True.

6. The trading agent updated its replay memory with the state, action, reward, next state, and done flag. Remember that the tuple describing an experience was denoted as $(s_t, a_t, r_{t+1}, s_{t+1}, d_t)$, where the parameter d was added as a boolean representation of terminal states. The reason was simply to distinguish the last state before a trading agent "lost" so that it would learn not to repeat the same errors. The experience was stored in numpy arrays, using a pointer to fill the next available slot in memory. Once the number of items in memory reached N (the maximum number of items), it began overwriting its earliest memories.

7. The trading agent trained its neural networks on its experience:

   (a) If the number of items in memory did not yet meet the batch size, training was skipped to avoid learning from data that did not yet exist (this occurred in the first 64 training steps).

   (b) If the training step was a multiple of 96, copy the weights of the online network to the target network.

   (c) The replay memory was then sampled for a batch of data (64 entries). For each item in the batch, the online network was trained by doing the following:

      i. The q-values were predicted for the next state s_{t+1} of the sampled memory using both the online and target networks.

ii. The maximal action for the next state s_{t+1} was selected based on the q-value outputs of the aggregated a and v streams of the online dueling Q network using an argmax function to find the index of the best action. (*n.b.: the online network was not finding the* ***value*** *of the action, simply* ***which was the best action****. The target network was later used to calculate the value of the maximal action.)*

iii. The q-values were predicted for the current states in the sampled memory using the online network.

iv. The loss function was computed according to equation(8). The results from this prediction were edited by selecting the q-value associated with the action actually taken, and updating the predicted q-value with the true q-value. To do this, the actual rewards received (r_{t+1}) were added to the predicted q-value of the next state s_{t+1} according to the target network, discounted by γ. If the sampled memory was a terminal memory, the future rewards were multiplied by 0 to cancel them out.

v. This updated batch of data was then fed to the online neural network to update its weights via back-propagation. To better understand this step, review the equations in *What is Q-learning* and *What is Deep Q-Network Learning* from the Introduction.

8. Finally, the agent is transitioned to the next state and the process is repeated until the trading agent reaches one of two stop conditions:

   - The trading agent reached a terminal state where its balance dropped too low
   - The time series reached the penultimate time step.

9. If there were any remaining units of Ether in the inventory I when a stop condition was reached, the units were sold at the current closing price and their value added to the balance from the episode. The balance was updated with the market price for those units, minus the sum of their purchase prices. See *Trading Rules* for more details.

Once the agent had trained on 30 episodes of training data, it was then fed the testing set of data and repeated the same process as training, without steps 3 or 7, and with ϵ set to 0 so that the agent always chose the greedy action. To build a suitable collection of results from testing data, this whole process was repeated for ten runs (each run with a new set of 30 training episodes and a new trading agent).

4 Results Analysis

4.1 Data exploration

Before beginning statistical analysis and reinforcement learning, it was important to understand the data in more depth. As previously mentioned, 5,675,203 total Tweets were collected, about 3.5% of the total Tweets. This gives some context to the scale of big data available on Twitter.

4.1.1 Ether price data

Starting with summary statistics for the Ether hourly prices:

	low	high	open	close	volume
0	8.25	8.30	8.26	8.30	1610.315200
1	8.30	8.53	8.30	8.47	3139.987090
2	8.45	8.60	8.45	8.59	3503.826085
3	8.49	8.60	8.58	8.53	1693.233010
4	8.34	8.54	8.53	8.38	2223.611356

Table 1: Pandas *head()* method called on the Ether price dataframe

	low	high	open	close	volume
count	48647.000000	48647.000000	48647.000000	48647.000000	48647.000000
mean	949.099156	963.295755	956.512399	956.546269	7841.863990
std	1179.649125	1195.776955	1188.051274	1188.050337	9900.998488
min	0.100000	8.160000	8.090000	8.100000	0.568451
25%	183.665000	185.695000	184.710000	184.715000	2496.999621
50%	327.700000	332.400000	330.310000	330.340000	4818.732141
75%	1331.635000	1357.740000	1345.720000	1346.585000	9273.992657
max	4835.150000	4867.810000	4849.040000	4849.040000	179904.541935

Table 2: Pandas *describe()* method called on the Ether price dataframe

Price data was expressed in USD, with volume in thousands. Based on the tables above, there was intense variability, as expected given the well-known volatility of cryptocurrency. The closing price is plotted on the next page.

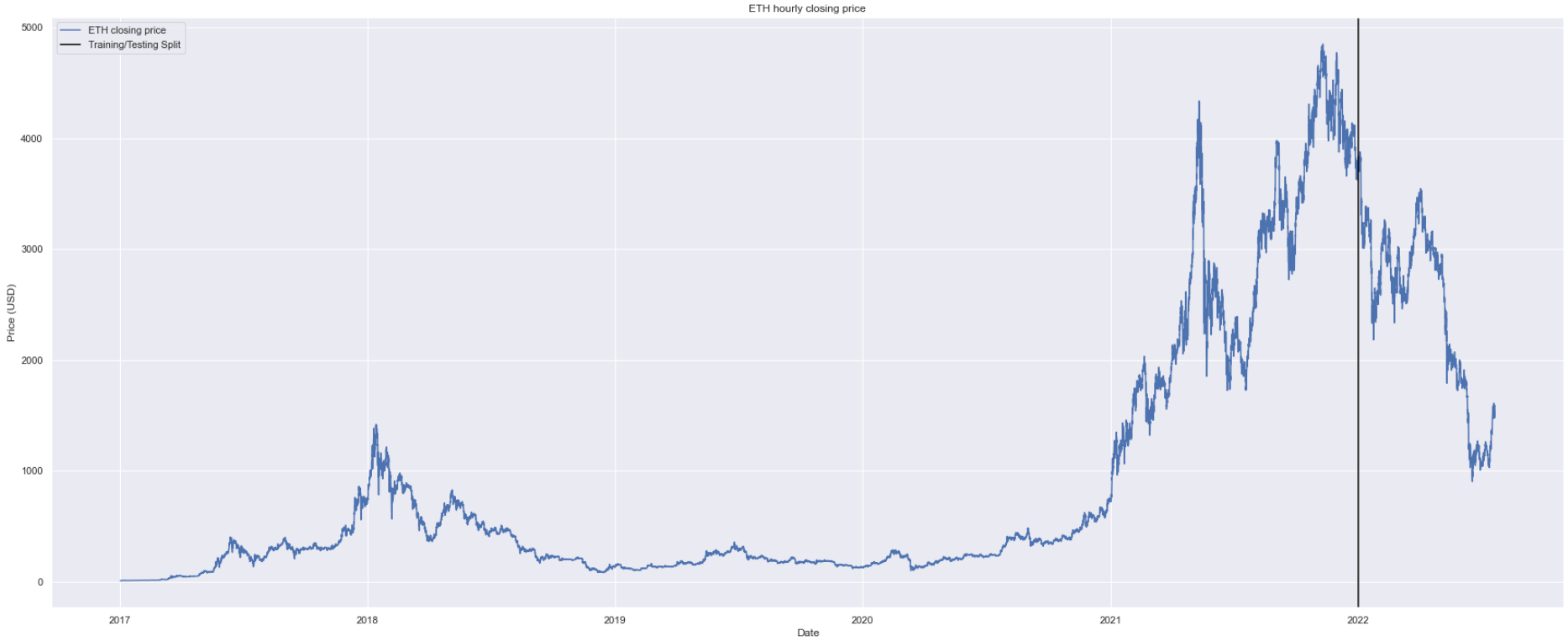


Figure 7: Ether hourly price

4.1.2 Ether Twitter sentiment scores

Continuing with summary statistics for the Ether hourly social media scores:

	Average Sentiment Score	Number of Tweets
0	0.000000	32
1	0.000000	19
2	0.052632	38
3	0.000000	22
4	0.000000	40

Table 3: Pandas *head()* method called on the Ether sentiment scores dataframe

	Average Sentiment Score	Number of Tweets
count	48647.000000	48647.000000
mean	0.084733	2986.499949
std	0.055137	4566.850529
min	-0.820000	19.000000
25%	0.050000	481.000000
50%	0.080000	836.000000
75%	0.110000	2746.000000
max	0.690000	30809.000000

Table 4: Pandas *describe()* method called on the Ether sentiment scores dataframe

Much like the price data, there was intense variability in the number of hourly Tweets. However, sentiment seems relatively neutral – given that sentiment scores were measured in the range of $[-1, 1]$, a mean of 0.08 and a standard deviation of 0.06 shows that Twitter opinions were mildly positive but rarely strong in any direction. The outliers were notable however – there was one hour in the data where Twitter users collectively voiced significant dislike for Ether, with a -0.8 sentiment score. On the flip side, there was at least one period with significant support as well, with a score of 0.69. The hourly sentiment scores were quite noisy however, so these outliers likely do not have strong predictive importance.

Although an adequate sample of Tweets was selected per hour to gauge sentiment, there was intense hourly variation in social media sentiment. This was rather unexpected – I had predicted that the Twitter sentiment would be much more stable and follow the price of Ether. Instead the data seems random, and required statistical tests to understand correlation and causality. The hourly social media sentiment scores and number of Tweets are plotted on the next two pages in figures (8) and (9). To reduce some of the noise and outliers, a 2-day rolling average was overlaid on the plots. Exploration into the outliers of figure (8) show that there was often one or more highly-followed social media influencers sharing an opinion about Ether, which was quickly retweeted within the same hour.

Next, the two graphs were combined to visually understand any correlation. The sentiment

scores and price were scaled to the same axis, then plotted together, using the 2-day rolling average for sentiment scores to reduce noise. As shown in figure (10), no correlation was immediately visible. This did not necessarily mean there was no Granger causality however – because the first-order differences needed to be taken to transform the time series data, there were deeper hidden correlations.

The process was repeated, this time with the number of Tweets against the closing price, in figure (11). It was much clearer that there was similarity in the data: it seemed that the shape of the number of Tweets mimicked the closing price, but lagged behind by a few weeks.

In the last plot, the top 500 hours with positive and negative sentiment (smoothed with the rolling 2-day mean) were overlaid onto the closing price, to understand visually if strong negative or positive sentiment corresponded with a change in price. As shown in figure (12), the strongest negative sentiment occurred around Ether's inception, especially after the first price drop. Examining the Tweets, it seemed that most users were unimpressed with the future adoption of cryptocurrency after it began to fall in price, thinking that the short rise and fall marked the end of the useful life of Bitcoin and Ether. Negative sentiment also occurred most strongly around present day, where the beginning of a recession economy after Covid-19 and the pullback of cryptocurrency markets added significant uncertainty among Ether investors.

The strongest positive sentiment on the other hand occurred a bit more sporadically after Ether shot up in price in Q1 and Q2 of 2021. There, the strongest positive sentiment seemed to appear right after a large spike in price. Looking at the Twitter data again, it showed that investors succumbed to a few cognitive biases that manifested as strong positive social media sentiment. Namely, participation bias (Hsieh & Kocielnik, 2016), where investors encouraged each other to continue holding and purchasing Ether in the unfounded certainty that it would continue to increase in price, the Bandwagon effect (Nadeau, Cloutier, & Guay, 1993) (also known as group-think), where investors mimicked others regardless of actual financial merit, and probability neglect (Kahneman, 2011), where investors believed (incorrectly) that the price of Ether would continue to rise regardless of statistical likelihood.

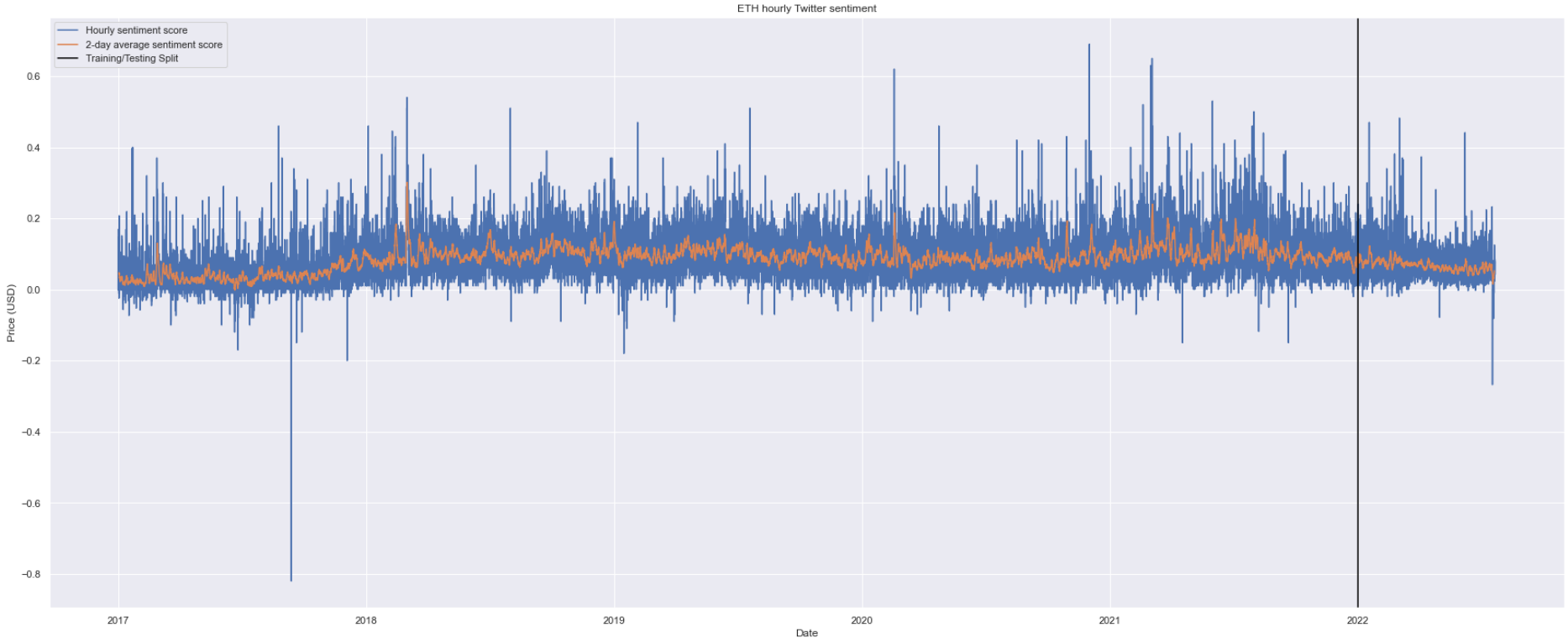


Figure 8: Ether hourly Twitter sentiment scores

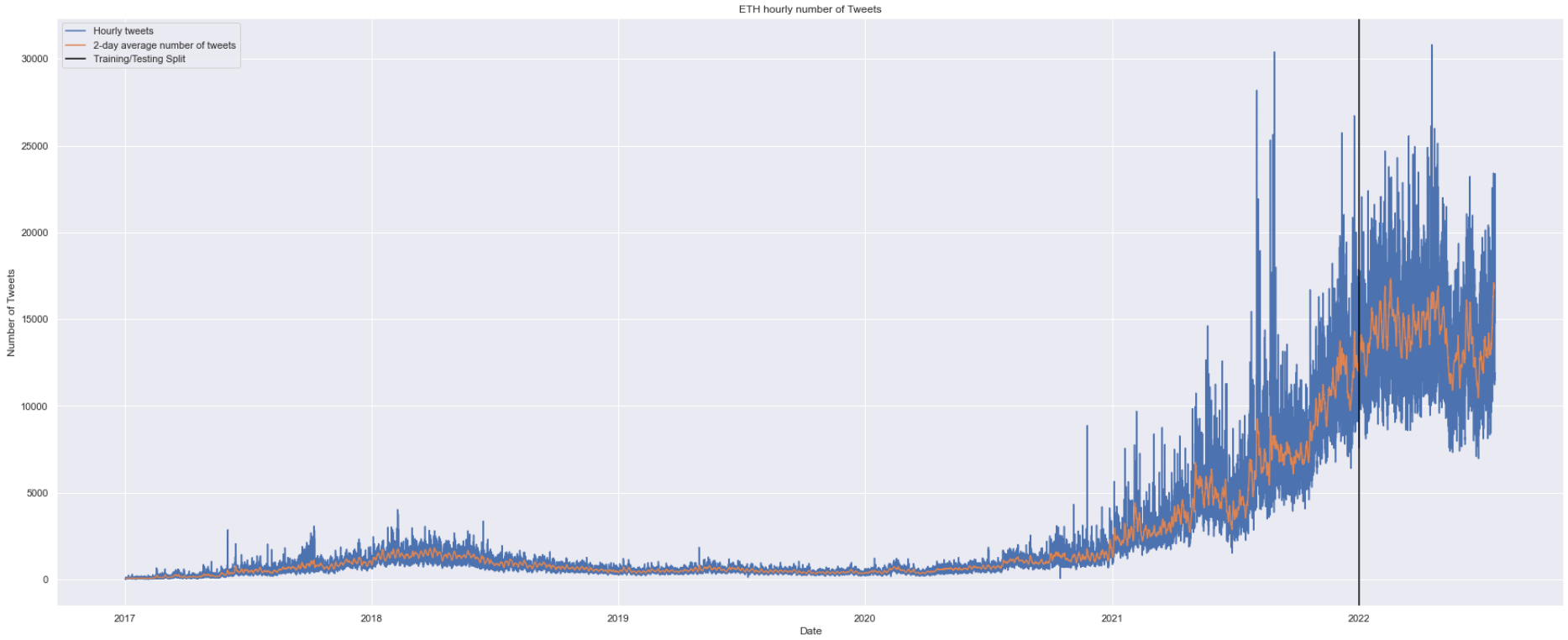


Figure 9: Ether hourly number of Tweets

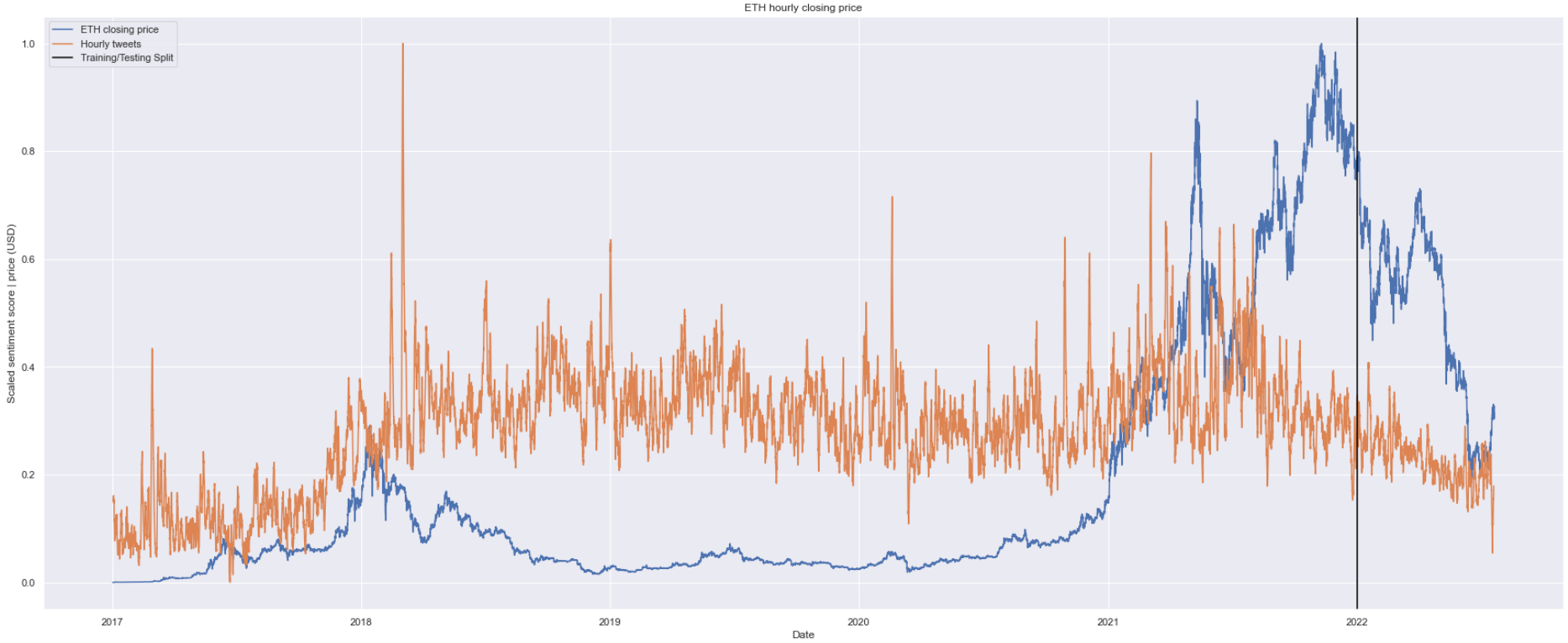


Figure 10: Ether hourly Twitter sentiment scores plotted against closing price

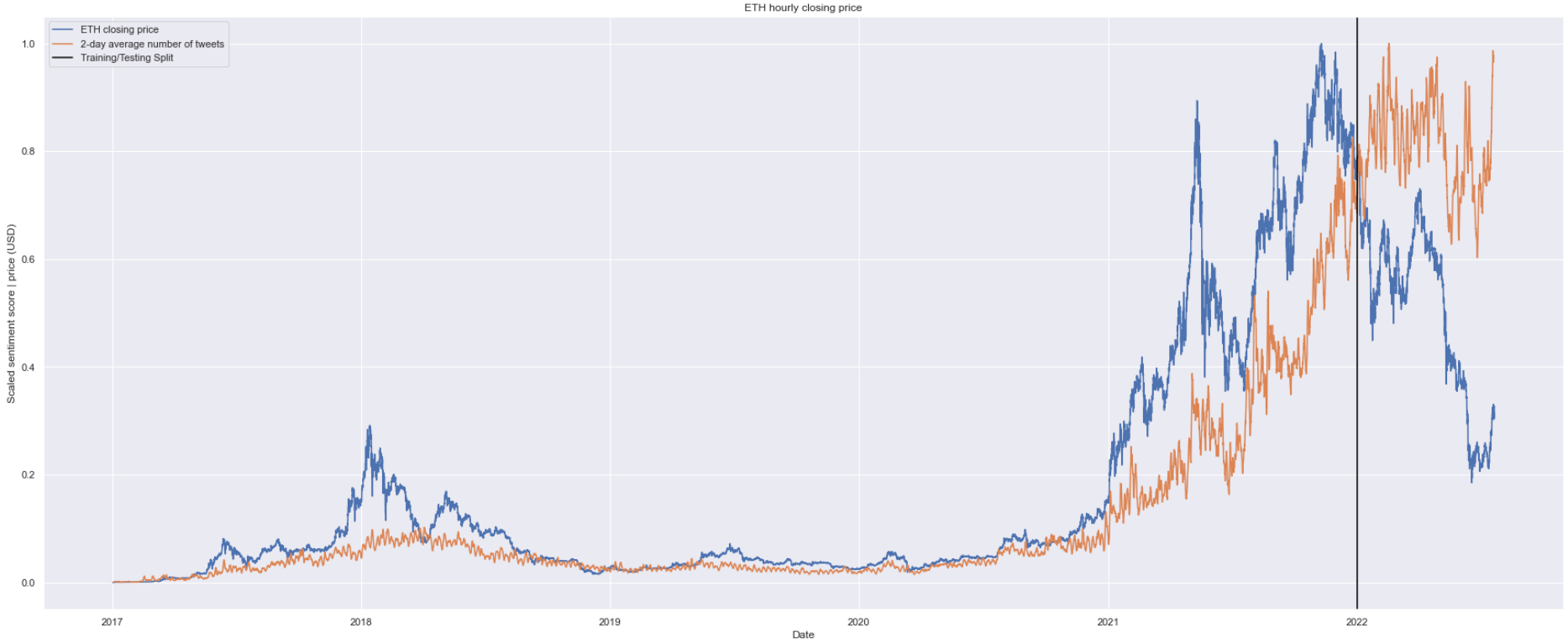


Figure 11: Ether hourly number of Tweets plotted against closing price

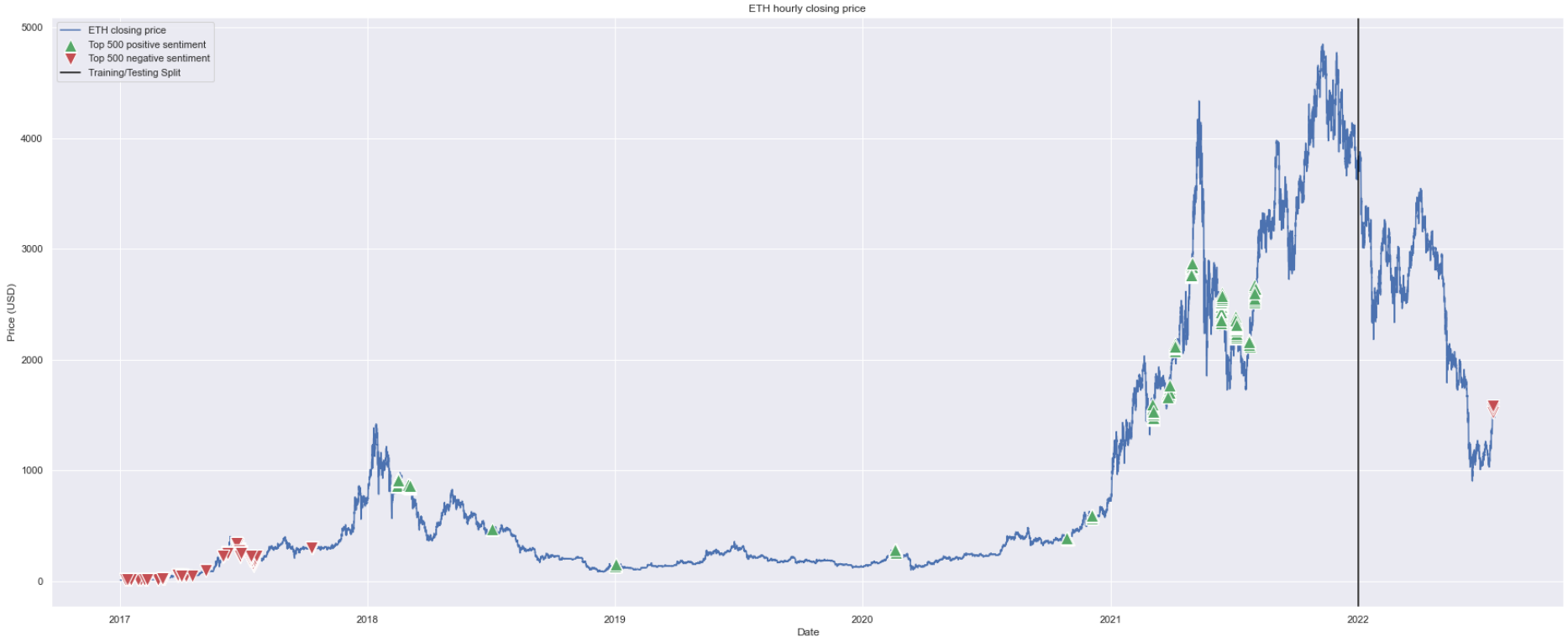


Figure 12: Ether hourly number of Tweets with top positive and negative sentiment plotted against closing price

Finally, the correlation between variables could be explored. Listing pairwise correlations, the close price was visibly correlated with the number of Tweets. However, sentiment scores were not correlated with any other variable.

Covariance Analysis: Ordinary
Date: 07/31/22 Time: 02:19
Sample: 1/01/2017 00:00 7/20/2022 22:00
Included observations: 48630
Balanced sample (listwise missing value deletion)

Correlation	TWEET_CO...	CLOSE	AVG_SENTI...
TWEET_COUNT	1.000000		
CLOSE	0.788315	1.000000	
AVG_SENTIMENT_...	-0.016696	0.083147	1.000000

Figure 13: Correlation between number of Tweets, sentiment scores, and closing price

The first-order differences of all three variables were distinctly uncorrelated, as expected after removing any trend component.

Covariance Analysis: Ordinary
Date: 07/31/22 Time: 02:16
Sample: 1/01/2017 01:00 7/20/2022 22:00
Included observations: 48616
Balanced sample (listwise missing value deletion)

Correlation	D(TWEET_C...	D(CLOSE)	D(AVG_SEN...
D(TWEET_COUNT)	1.000000		
D(CLOSE)	0.002129	1.000000	
D(AVG_SENTIMEN...	-0.000174	-0.003201	1.000000

Figure 14: Correlation between the first order difference of the number of Tweets, sentiment scores, and closing price

Given the general lack of correlation between variables, it was interesting to see that Granger causality was still present.

4.2 Granger causality testing

As mentioned in the methodology, ADF and KPSS were used tests to establish stationarity in the data. The results of the tests before and after taking the first order difference can be found in the Appendix, in Figures (29) through (40). After differenciating the data to achieve stationarity, pairwise Granger causality tests were run.

The null hypothesis that X does not cause Y was rejected for the closing price on Tweet count, Tweet count on closing price, Tweet count on sentiment, and closing Tweet count on sentiment. For all other relationships, no Granger causality was present.

Pairwise Granger Causality Tests
Date: 07/31/22 Time: 02:18
Sample: 1/01/2017 00:00 7/21/2022 15:00
Lags: 96

Null Hypothesis:	Obs	F-Statistic	Prob.
D(CLOSE) does not Granger Cause D(TWEET_COUNT)	48550	3.71253	2.E-31
D(TWEET_COUNT) does not Granger Cause D(CLOSE)		3.19923	8.E-24
D(AVG_SENTIMENT_NUM) does not Granger Cause D(TWEET_COUNT)	48153	1.17884	0.1116
D(TWEET_COUNT) does not Granger Cause D(AVG_SENTIMENT_NUM)		1.31369	0.0214
D(AVG_SENTIMENT_NUM) does not Granger Cause D(CLOSE)	48152	0.75793	0.9629
D(CLOSE) does not Granger Cause D(AVG_SENTIMENT_NUM)		1.41931	0.0044

Figure 15: Granger causality between the first order difference of the number of Tweets, sentiment scores, and closing price

Breaking it down, the results conclusively show that closing price was not caused by sentiment scores at the 95% confidence level; rather, the inverse was true. As the price of Ether fluctuated, social media sentiment responded in turn. This meant that sentiment analysis, while a useful to better understand Ether, likely did not have predictive power on its future price. Instead, changes in price caused changes in score.

Next, relationships with Tweet counts were interesting. Tweet counts were Granger-caused by closing price, meaning that changes in the price of Ether caused a change in the number of Tweets. On the other hand, closing price was also Granger-caused by the number of Tweets, meaning that a change in the number of Tweets could effectively predict a change in closing price. This was an interesting two-way feedback loop between the variables; adding the number of Tweets to the Deep Q-Network as a feature likely increased its performance if the neural network detected and responded to this Granger causality. The number of Tweets Granger-causing price changes supported the hypothesis discussed in the literature review where similar results appeared for Bitcoin in (Shen et al., 2019).

Finally, Tweet counts and sentiment were, as expected, related by Granger causality. Because sentiment Granger-causing Tweet counts had a p-value (0.1116) above the alpha level of 0.05, null hypothesis was not rejected. On the the other hand, the number of Tweets did Granger-causes sentiment. This meant that changes in sentiment score could be predicted by changes in the number of Tweets, an interesting relationship. I postulate that more Tweets were usually indicative of some big event in the market, which would have had a strong sentiment reaction, thus promoting this Granger causality.

4.3 Deep Q-Networks

4.3.1 Twitter data omitted from feature input

Plotted below is a histogram of actions the agent selected during training. Only the last episode was examined here as it had the lowest ϵ value. As a reminder, ϵ started at 1 and decayed exponentially until the last time step of the final episode, meaning that actions were selected mostly at random early in the training. The last episode showed the agent's best action selection decisions for the training data with an ϵ near 0.02. As shown in the figure, the agent primarily chose to hold its position when given a choice between all three actions, which reflects similar decisions of human traders.

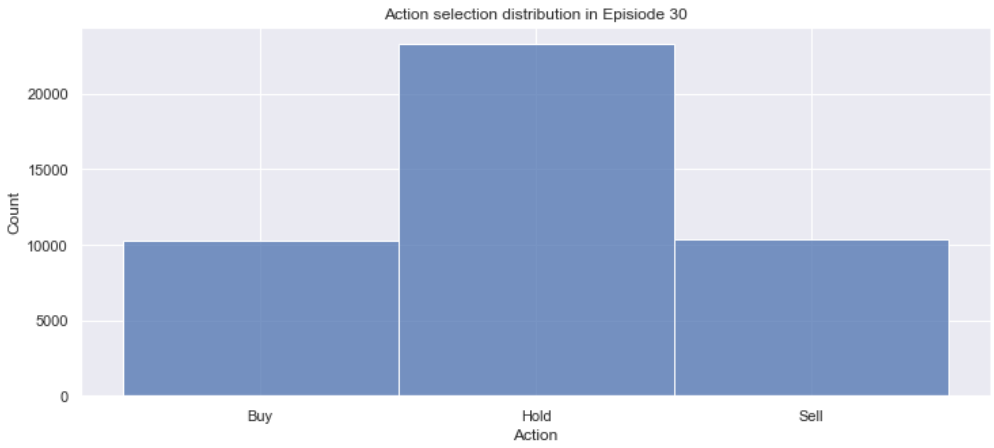


Figure 16: Actions selected by the agent during training in episode 30 (no Twitter data)

Next, the inventory held over time was plotted in figure (17). The fact that the agent never accumulated an inventory greater than 25 units suggested that it prioritized immediate rewards, and was either unable to predict long-term price increases or believed that short-term rewards were more beneficial to overall profit. The subsequent figure overlays the closing price of Ether scaled on the same graph. It was difficult to spot clear trends in the data due to the fine granularity of the x-axis. On a daily or weekly time scale, changes in inventory might have become easier to correlate to changes in Ether price, but the rapid transaction speed made inventory purchases and sales seem arbitrary. Figure (19) was perhaps the most important plot, which shows how the portfolio value of the trading agent changed over time. It became clear how the trading agent generated its profit: while it was able to capitalize on some mraket price increases, it primarily avoided losing value during sudden drops. Clearly, the trading agent was able to detect signals in the data for upcoming market downturns, and was able to mitigate them appropriately. Finally, figure (20) shows the profits for each episode of the training data, plotted against ϵ. Profitability is predictably low when the agent was exploring the environment with random actions, but increased as epsilon decayed.

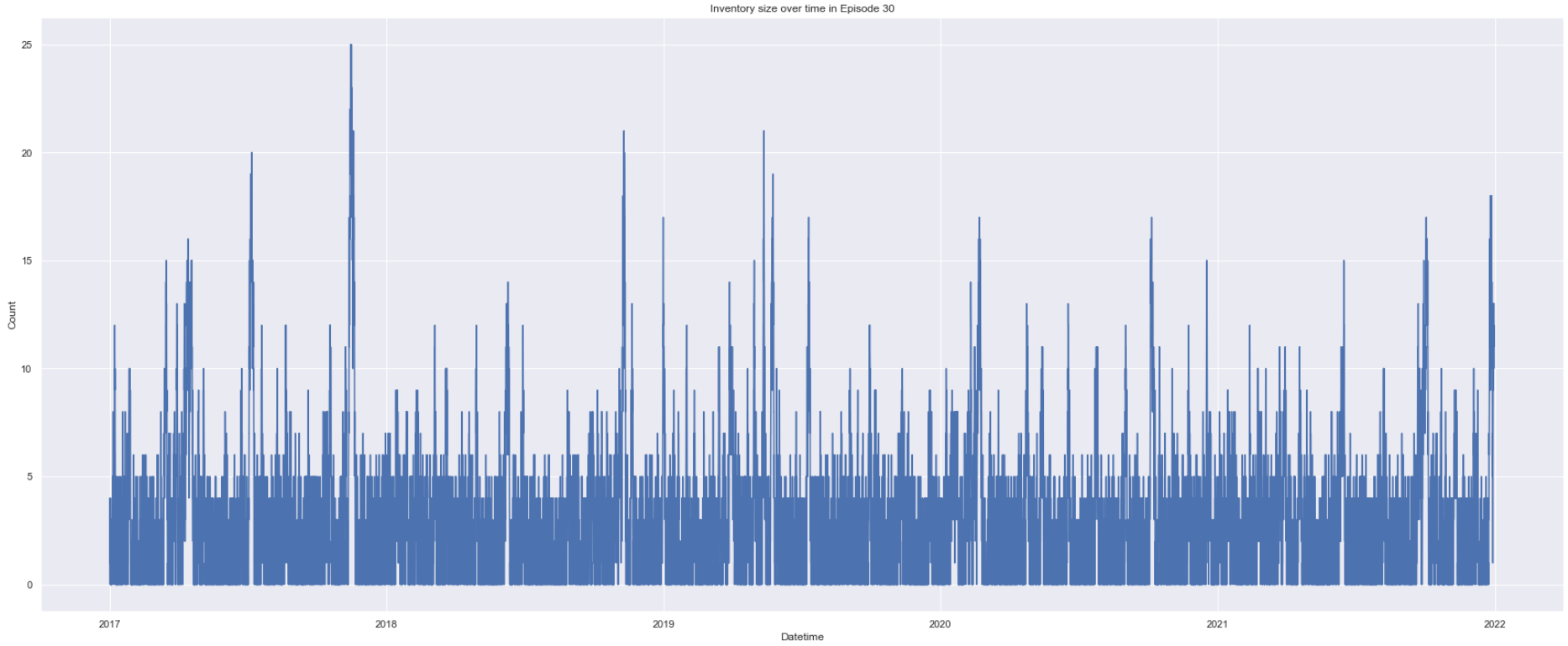


Figure 17: Inventory held by the agent during training in episode 30 (no Twitter data)

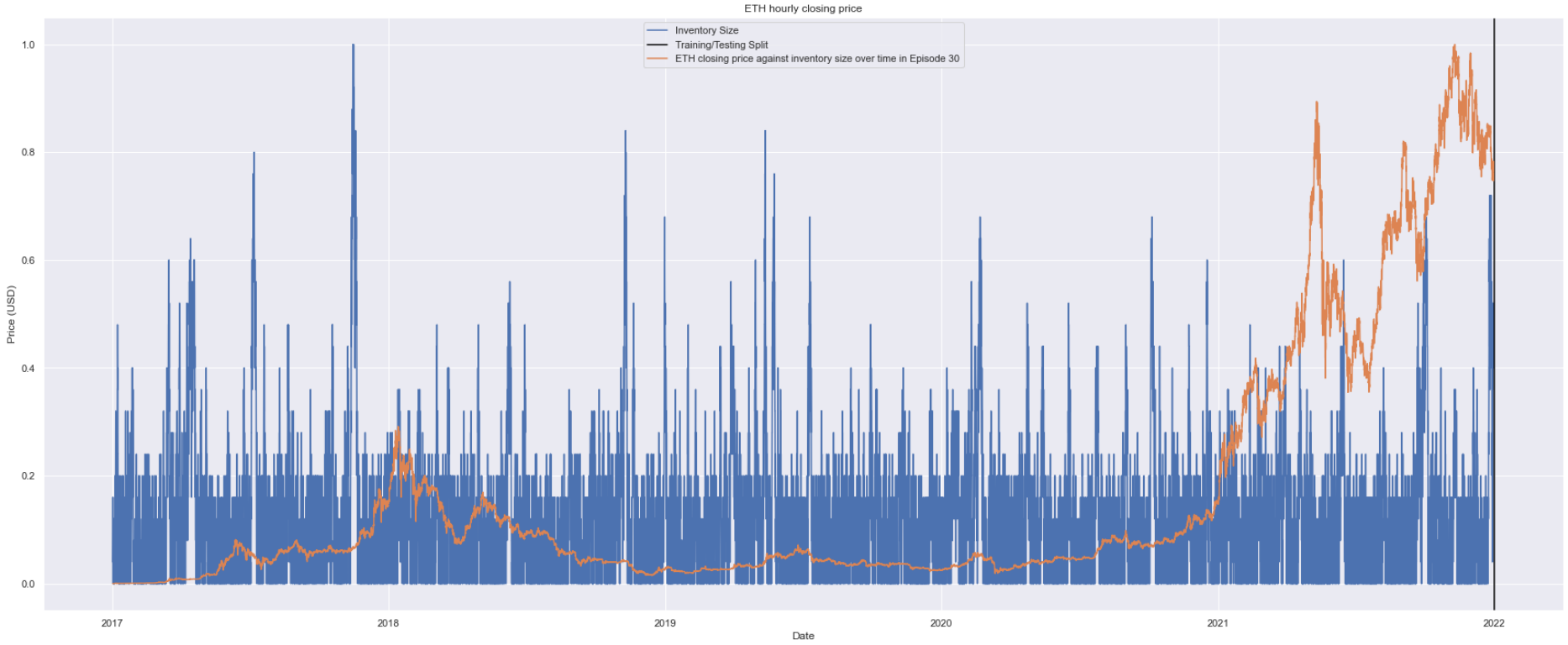


Figure 18: Inventory held by the agent during training in episode 30, plotted against Ether closing price (no Twitter data)

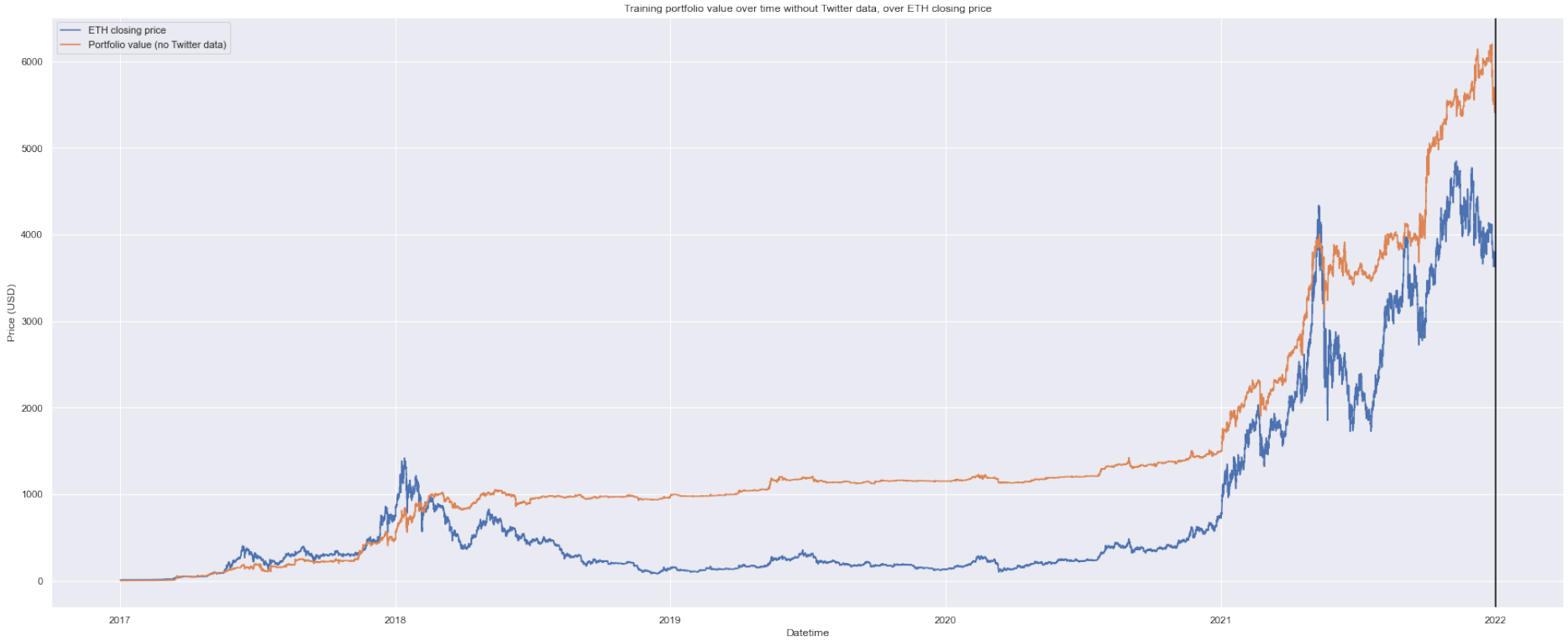


Figure 19: Portfolio value of time during training episode 30, plotted against Ether closing price (no Twitter data)

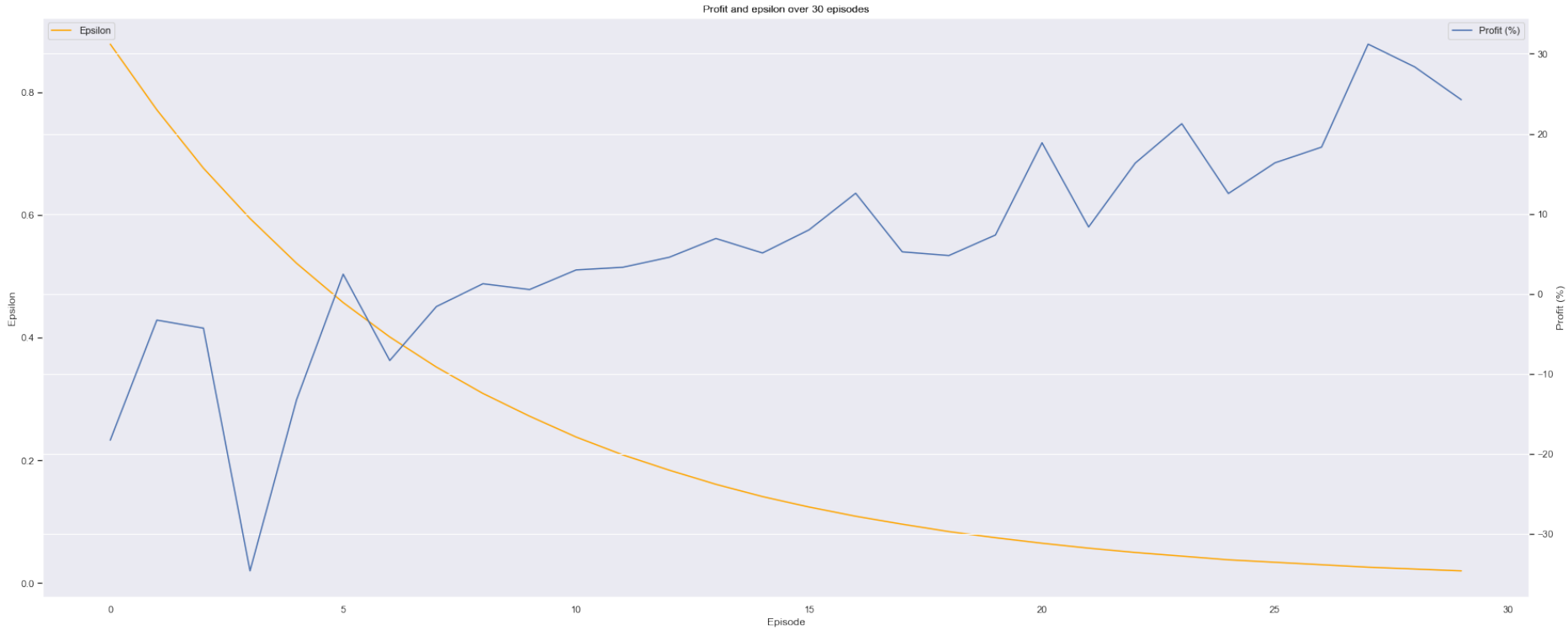


Figure 20: Profit of the agent for each training episode (no Twitter data)

The data in the figure (20) was derived from this table of training results:

Episode	Cumulative Runtime (Hours)	Profit (%)	Epsilon
0	1.80	-18.3000	0.878
1	3.60	-3.2894	0.771
2	5.44	-4.2940	0.676
3	7.29	-34.6300	0.594
4	9.12	-13.2350	0.521
5	10.97	2.4500	0.457
6	12.83	-8.3400	0.401
7	14.72	-1.6000	0.352
8	16.62	1.2570	0.309
9	18.54	0.5300	0.272
10	20.49	2.9810	0.238
11	22.46	3.3140	0.209
12	24.43	4.5660	0.184
13	26.42	6.9120	0.161
14	28.41	5.1010	0.141
15	30.39	7.9940	0.124
16	32.42	12.5660	0.109
17	34.43	5.2430	0.096
18	36.43	4.7780	0.084
19	38.41	7.3460	0.074
20	40.42	18.8780	0.065
21	42.46	8.3470	0.057
22	44.51	16.3300	0.050
23	46.57	21.2590	0.044
24	48.70	12.5300	0.038
25	50.67	16.3670	0.034
26	52.58	18.3300	0.030
27	54.48	31.2020	0.026
28	56.40	28.3530	0.023
29	58.34	24.2520	0.020

Table 5: Training summary (no Twitter data)

With the trading agent fully trained, the Deep Q-Network could then be fed testing data, with ϵ set to 0 so that the agent would always pick the action it believed would generate the highest profit.

Figures 21 through 24 show for the first testing run, respectively, the actions selected by the agent, the inventory held by the agent, the inventory held plotted against Ether's close price during the test period, and the portfolio value over time during testing. During the test set, the agent chose the hold action more than training, and the maximum inventory size decreased to 16, but it was similarly difficult to understand how changes in inventory correlated to changes in closing price due to the fine granularity of the time series. Figure (24 shows a similar result for the testing set as the training set: the agent avoided losing value during sharp market losses,

but was unable to capitalize on moments of market recovery.

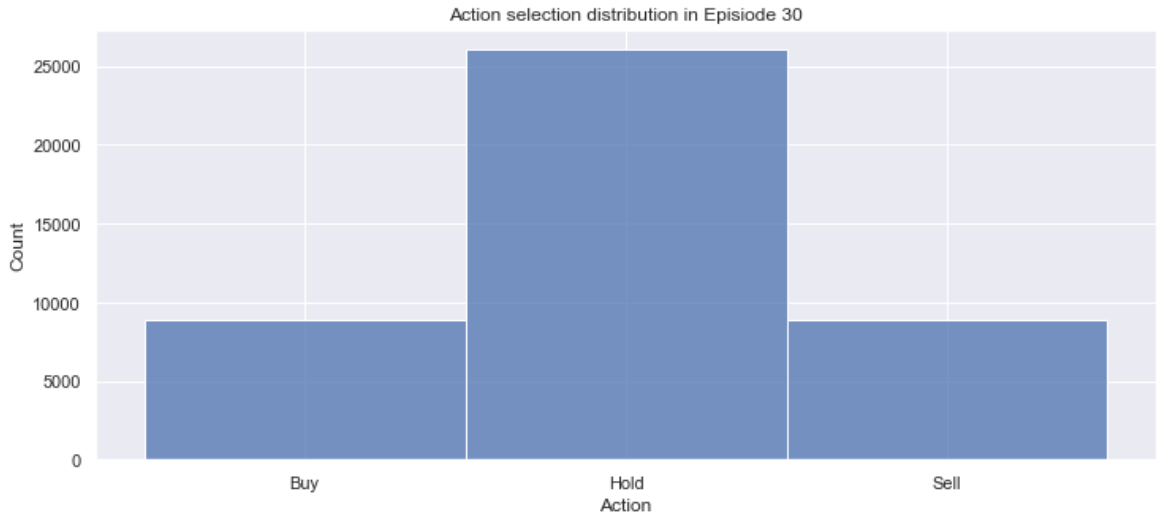


Figure 21: Actions selected by the trading agent during the first testing run (no Twitter data)

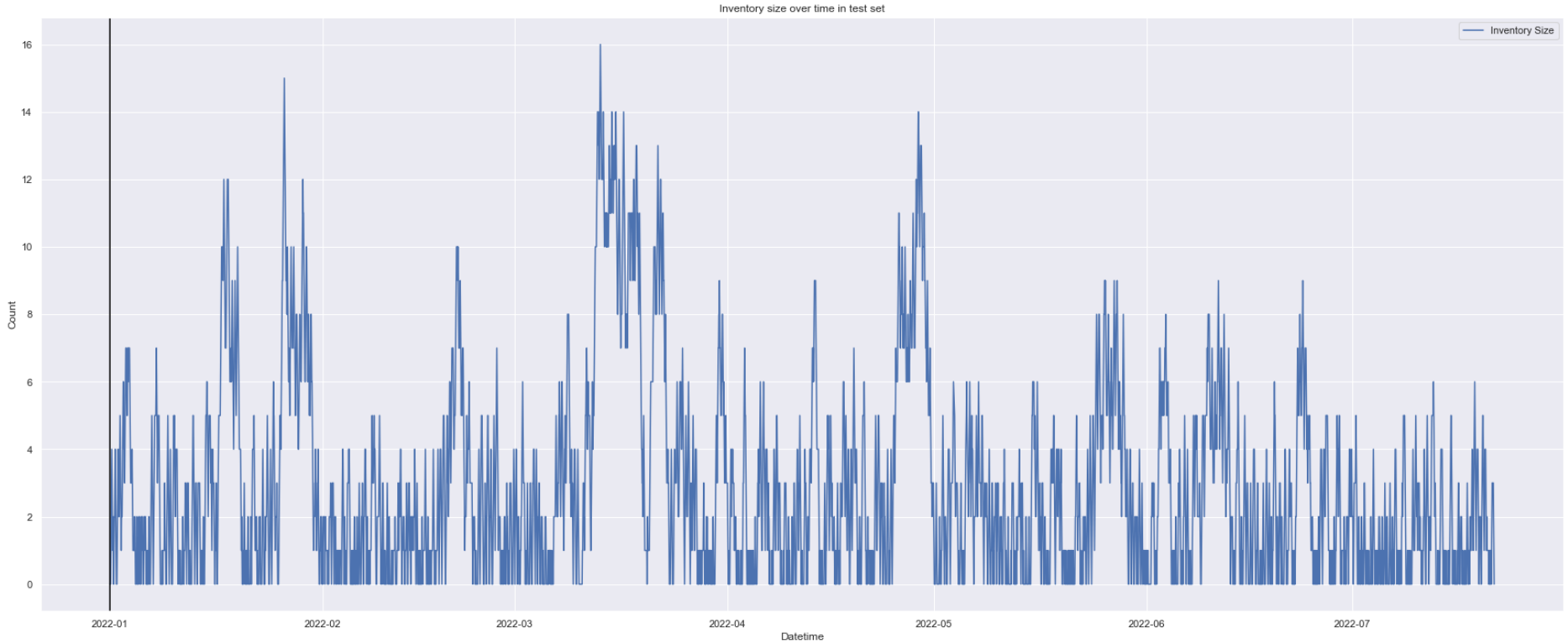


Figure 22: Inventory held by the agent during the first testing run (no Twitter data)

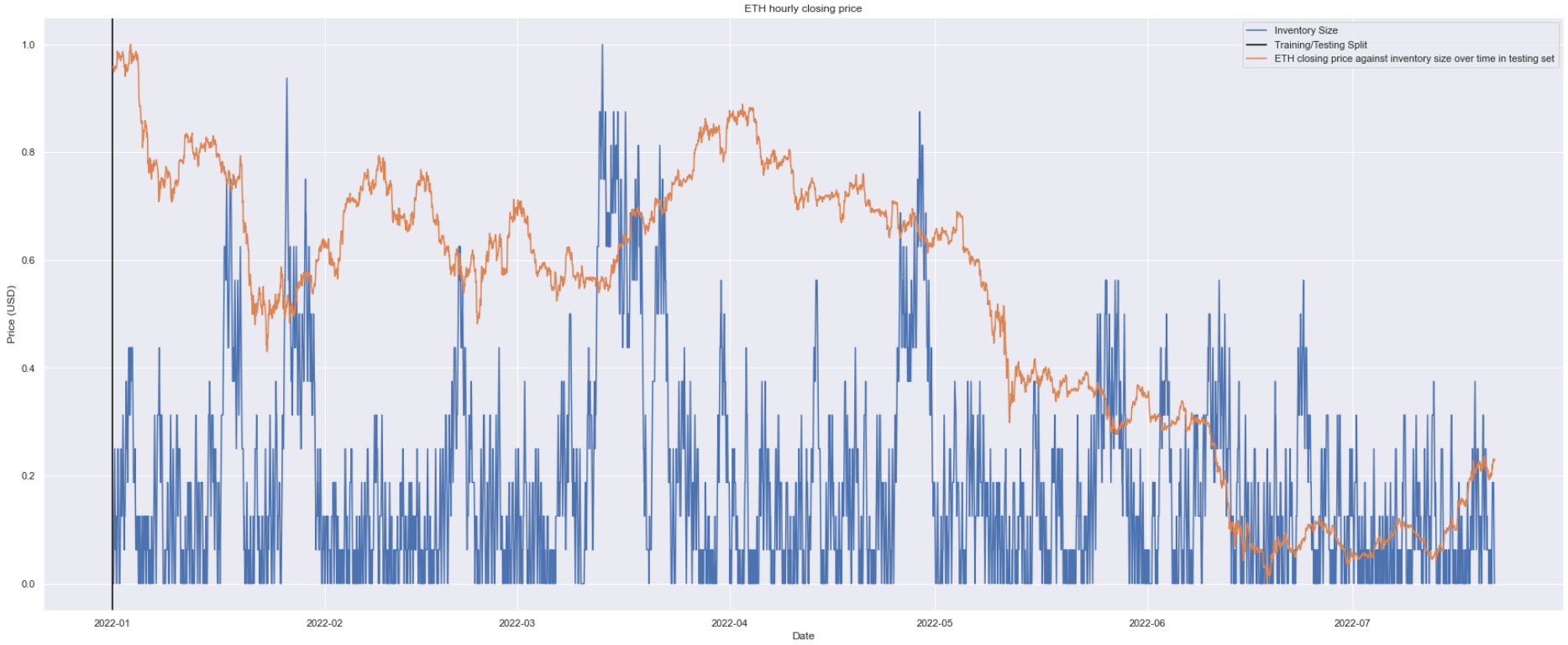


Figure 23: Inventory held by the agent during the first testing run, plotted against Ether closing price (no Twitter data)

Figure 24: Portfolio value over time during the first testing run, plotted against Ether closing price (no Twitter data)

The entire process was then repeated 10 times (training the agent for 30 episodes, then running the trading agent on the testing data). The profits from each distinct run of the test data was summarized in the table below:

Run	Profit (%)
0	-24.968
1	-31.802
2	-22.628
3	-22.516
4	-49.915
5	-43.038
6	-42.399
7	-46.292
8	-30.465
9	-17.846

Table 6: Profit over 10 test runs (no Twitter data)

The mean profit from the trading agent trained without sentiment scores or Twitter volume was -33.19%, but displayed strong variability with returns ranging anywhere from -49.92% to -17.85%. These figures may have seemed unimpressive, but it was important to consider the bear market of the testing set. The market's overall performance during the testing period was -53.47%, meaning that the agent effectively outperformed the market by **20.28%**. Using just historical price data, the Deep Q-Network was able to perform better than a buy-and-hold strategy for the duration of the test set, showing its power as a decision-making support tool for investors trading cryptocurrency in a bear market, and especially as a tool to avoid losses during market price shocks.

The next step was to add sentiment scores and social media volume (number of Tweets) as features, and re-run the 10 testing runs.

4.3.2 Twitter data included from feature input

Some descriptions in this section were skipped if figures were previously described.

The histogram of actions selected by the agent was nearly indistinguishable from the training set with no Twitter data added, and as such, was not included here. Much like the figures in the preceding section, plotting the agent's inventory over time or against the closing price of Ether was difficult to interpret, and such plots were also omitted here. Figure (25) shows the portfolio value over time, plotted against the previous training data (with no Twitter features) and Ether closing prices. Unlike the training without Twitter data, the agent seemed more susceptible to sudden price drops, where the previous agent was more resilient. On the other hand, the trading agent with Twitter data was much more capable of benefiting from price increases, and far surpassed the trading agent without Twitter information during strong price increases. Figure (26) shows the average profits for each episode of the training data against ϵ.

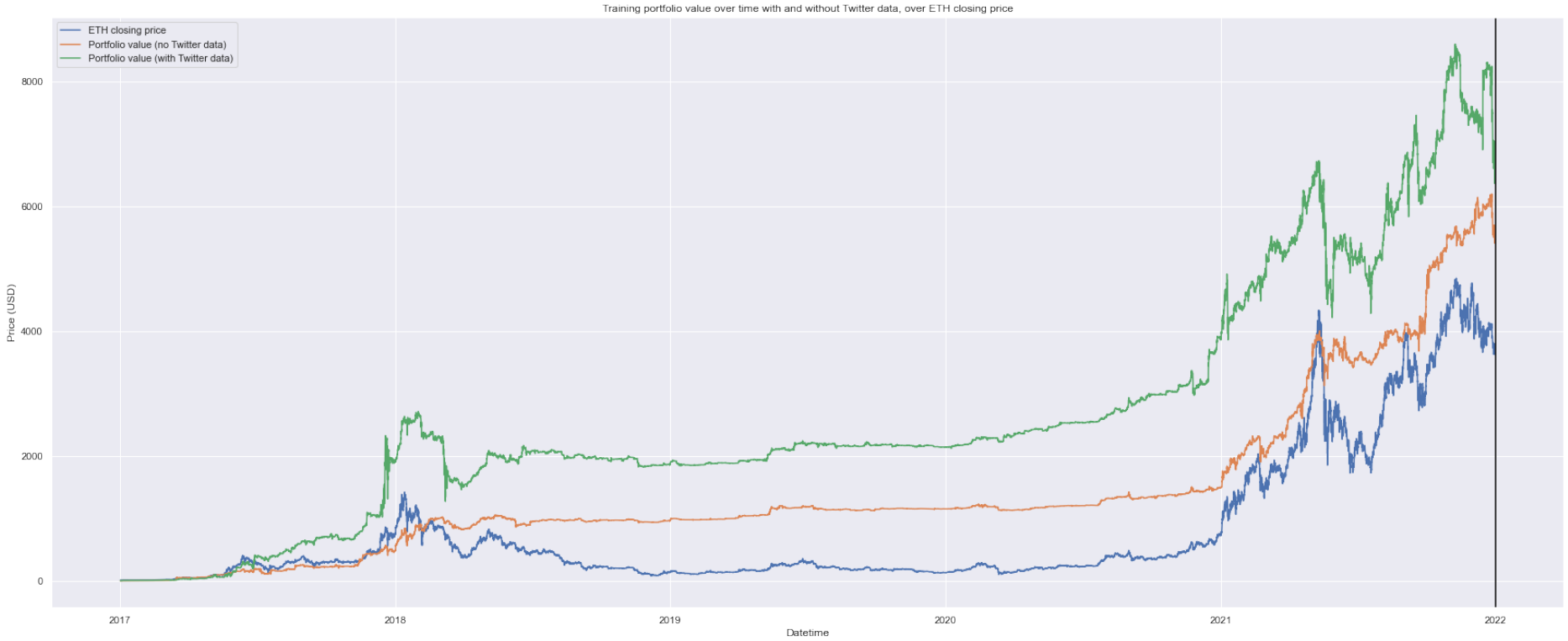


Figure 25: Portfolio value over time during training episode 30, plotted against Ether closing price (with and without Twitter data)

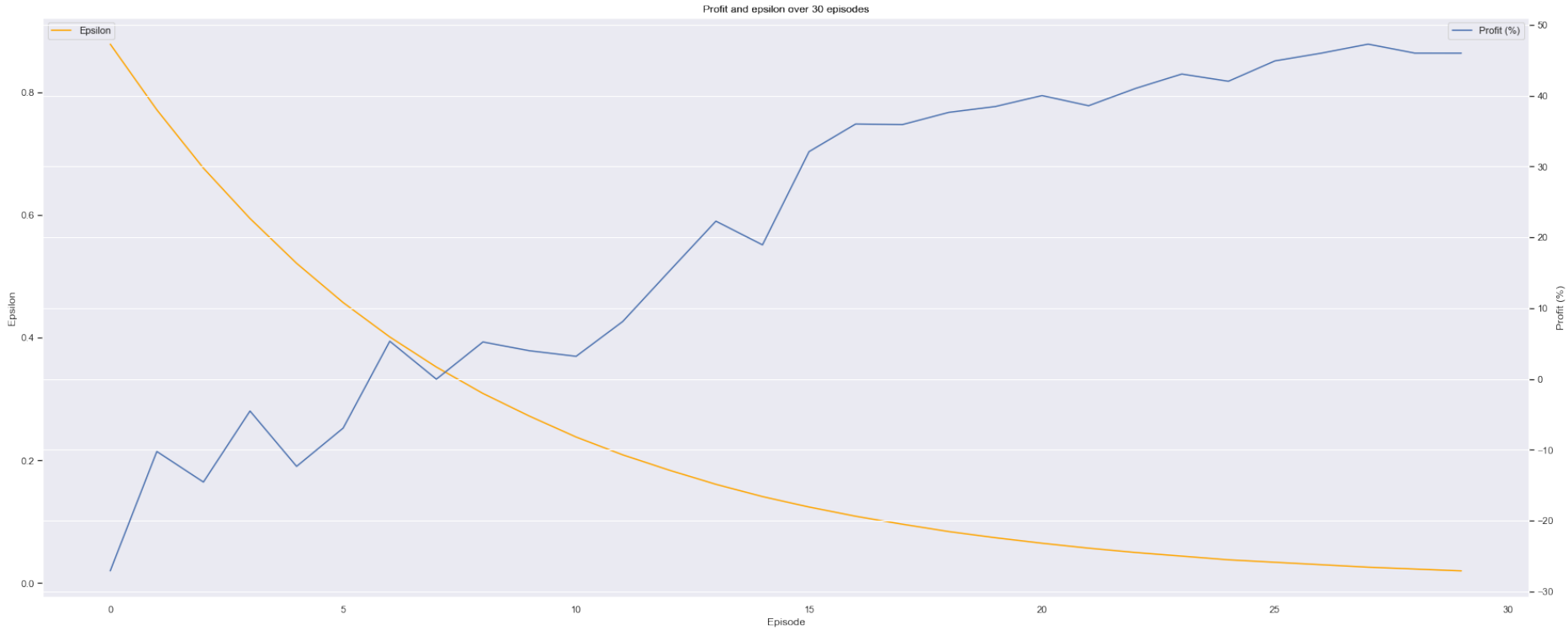


Figure 26: Profit of the agent for each training episode (with Twitter data)

The data from figure (26) was derived from this table of training results:

Episode	Total Runtime (hours)	Profit (%)	Epsilon
0	2.03	-27.090	0.878
1	4.07	-10.230	0.771
2	6.15	-14.550	0.676
3	8.24	-4.515	0.594
4	10.31	-12.333	0.521
5	12.40	-6.923	0.457
6	14.50	5.340	0.401
7	16.63	-0.020	0.352
8	18.78	5.233	0.309
9	20.96	3.990	0.272
10	23.15	3.209	0.238
11	25.38	8.123	0.209
12	27.61	15.233	0.184
13	29.86	22.293	0.161
14	32.11	18.944	0.141
15	34.34	32.112	0.124
16	36.64	36.012	0.109
17	38.91	35.923	0.096
18	41.16	37.652	0.084
19	43.41	38.484	0.074
20	45.68	40.023	0.065
21	47.98	38.593	0.057
22	50.30	41.020	0.050
23	52.63	43.067	0.044
24	55.03	42.044	0.038
25	57.25	44.923	0.034
26	59.41	46.022	0.030
27	61.56	47.290	0.026
28	63.73	46.022	0.023
29	65.92	46.019	0.020

Table 7: Training summary (with Twitter data)

Moving on to the test set, figure (27) shows the portfolio value over time for the first testing run, plotted against the first testing run of the agent with no Twitter data and the Ether closing price. The agent exhibited similar behaviour as in testing – it was not able to mitigate price drops as well as the agent without Twitter data, but was much more capable of generating returns during short periods of market recovery.

Figure 27: Portfolio value over time during the first testing run, plotted against Ether closing price (with Twitter data)

The profits from each distinct run of the test data were summarized in the table below:

Run	Profit (%)
0	-11.752
1	-7.665
2	-40.591
3	-10.338
4	-33.067
5	-18.681
6	-24.908
7	-13.523
8	-37.887
9	-23.981

Table 8: Profit over 10 test runs (with Twitter data)

The mean profit from the trading agent trained including sentiment scores and Twitter volume was -22.24%, but once again displayed strong variability with returns ranging from -40.59% to -7.66%. As a reminder, the market's overall performance during the testing period was -53.47%, meaning that the agent effectively outperformed the market by **31.23%**. This was a very promising result for an untuned Deep-Q Learning network, and showed that adding sentiment and Twitter volume as inputs to the data improved the performance of the trading agent by **10.95%**. The improvement in profitability simply from adding sentiment scores and social media volume suggested that these variables could likely be applied to many different machine learning applications that predict cryptocurrency price movement. This made sense, given the Granger-causality detected between social media volume and Ether price.

The results from this study highlighted the power of deep reinforcement learning in financial markets. Practical applications of Deep Q-Learning can be developed and implemented as a decision support system in a business context; business recommendations will be elaborated in the next section.

5 Recommendations

Based on the findings of this thesis, some key business-level managerial recommendations were developed for people interested in trading cryptocurrencies using Deep Q-Networks and sentiment analysis.

First, the results showed that sentiment analysis and social media discussion volume were effective predictive tools to understand the direction of cryptocurrency. Therefore, regardless of the algorithm or machine learning model being used to predict price changes, this data is valuable to include in input features. I would argue that sentiment scores and social media discussion volume alone can be used as valuable decision-making tools because of their clear meaning for managers.

Second, although Deep Q-Networks were shown to generate a positive returns, the performance of the model was not exceptional. As such, managers who leverage Deep Q-Learning should use the buying, selling, and holding signals of the network as inputs to decision-making when accompanied by other analysis. Rather than relying entirely on the Deep Q-Network for trading, augmenting trading decision using its output is a much more valuable application. Neural networks can be used to potentially find deep hidden patterns in data, in conjunction with traditional fundamental and technical analysis which allow managers to see more visible trends. It may also be valuable for a manager to look at the expected benefit of each action (buy, hold, or sell) using the three-node output layer of the advantage stream rather than just the trading signal alone.

Related to my previous recommendation, machine learning algorithms require tuning for optimal results. Therefore, should a Deep Q-Network algorithm be implemented in a business context, adequate time should be allocated to properly tune and test the model with a range of hyperparameters and architectures. The same testing methodology described in this paper can be used, but a more robust testing strategy should include a larger window of data to test on (for example, using random slices of time to test model performance rather than a set slice at the end of the time series).

Finally, it is important to address the black-box nature of ANNs. Although the results of a neural network can be valuable, the reason why a neural network chooses to send a buying signal over a selling signal for example can be extremely nuanced. Managers are therefore cautioned from trying to identify patterns where they do not exist in the outputs of ANNs. In a real-world context, many financial decisions must be justified due to attributes of the data, which is impossible using an ANN that simply tells the investor what action to take. It should be reiterated therefore the importance of using this research as a decision support tool, rather than as a decision-maker on its own.

6 Conclusion

Overall, the results of the study were mostly on par with the hypotheses described during the literature review, and this thesis successfully answered all the research questions presented in the introduction. Granger causality was indeed present between variables, however it was the closing price of Ether that predicted sentiment, not the other way around. The number of Tweets was also shown to Granger-cause closing price and vice versa.

Deep Q-Networks are excellent models in environments with a small action space and where future states are based on previous actions. However, the inherent variability of cryptocurrency markets and lack of control trading agents have on the market made Deep Q-Networks an imperfect application. Whether or not the trading agent bought or sold a unit of Ether for example did not change the price of Ether in future states. That being said, the objective of this study was never to create the most profitable trading strategy, but rather to determine whether reinforcement learning for financial markets merited future research through a proof of concept model. In this sense, this thesis was successful in establishing a baseline approach researchers can take to use sentiment analysis and reinforcement learning for financial market prediction. The results of this study showed that DDDQNs have practical applications outside of academia, and social media sentiment data and volume were found to be important features when predicting cryptocurrency prices.

An important outcome from this study to highlight was its technical feasibility in a cloud-based era. A decade ago, cloud computing was rarely used for personal research projects. Now, the scale of the computations and data used in this thesis were only made possible by highly-scalable and cost-effective cloud solutions offered by the likes of Amazon, Microsoft, and Google. The Apache Spark workflow used to process over 2 gigabytes of Tweets for sentiment analysis required a heavily-distributed network of machines that quickly scaled up and down as needed, which was infeasible for individuals who cannot be expected to purchase and maintain this level of computer hardware. Cloud computing has empowered individuals and small companies to perform complex research and process big data in ways that were inaccessible a decade ago.

There are additionally some limitations in this study that could be explored more deeply in future papers. First, the trading environment was over-simplified for the sake of building a proof of concept model. As described in the trading rules section of the methodology, the agent was restricted to buying or selling an individual unit of Ether at each time step. A more accurate representation of reality would allow the agent to buy any number or fraction of Ether units at a given time step, using more advanced Deep Q-Learning models or different reinforcement learning techniques altogether.

Next, sentiment analysis was limited to data collected from English-speaking Twitter. This introduced bias to the dataset by focusing only on opinions from mostly non-AMEA regions where the predominant first or second language was English. Although ethical bias was not especially relevant in this use case, it can be important to consider when conducting sentiment analysis in more sensitive applications. Twitter data collection was also quite naive, by weighing

each Tweet equivalently, and counting retweets to emphasize more popular opinions. A more tuned approach could use public Tweet metrics available in the Academic API (such as likes and shares) to weigh the sentiment of popular Tweets more heavily when calculating sentiment scores. This would likely promote clearer trends in the data and remove some noise.

Another limitation in the scope of this study was the time spent tuning the model. This thesis served as a proof of concept to determine whether Deep Q-Networks could work as a decision support system, and to decide whether the topic merited further research. As such, optimizing the DDDQN architecture and parameters was not in the scope of this work. A necessary extension to this work before implementation in a real business scenario would therefore be to spend the appropriate time building and optimizing a high-performing DDDQN. An excellent continuation to this thesis will be to tune the DDDQN in order to generate the highest possible profits.

Moving on to surprising results, it was primarily the lack of correlation between sentiment and closing price that was unexpected. Although Granger causality was present, the shape of the sentiment scores plot was unintuitively different from the closing price. It exhibited apparent randomness at a first glance, requiring differenciation and deeper statistical analysis to discover the causal link between sentiment and price. It was also surprising to see that the number of Tweets had a stronger causal relationship on closing price that sentiment scores.

In summary, this paper demonstrated a novel application of deep reinforcement learning for cryptocurrency price prediction using sentiment analysis as feature input, and established a causal relationship between cryptocurrency price and sentiment, as well as between social media conversation volume and cryptocurrency price. Further research can extend this work to optimize and implement the DDDQN model using a similar methodology.

7 References

Abraham, J., Higdon, D., Nelson, J., & Ibarra, J. (2018). Cryptocurrency price prediction using tweet volumes and sentiment analysis. *SMU Data Science Review*, *1*(3).

Ballis, A., & Drakos, K. (2020). A markov chain analysis for capitalization dynamics in the cryptocurrency market. Munich Personal RePEc Archive. Retrieved from `https://mpra.ub.uni-muenchen.de/109329/`

Baumol, W. J. (1970). *Economic dynamics*. Macmillan.

Bellman, R. (1957). A markovian decision process. *Journal of Mathematics and Mechanics*, *6*(5), 679–684. Retrieved from `http://www.jstor.org/stable/24900506`

Bharathi, S., & Geetha, A. (2017). Sentiment analysis for effective stock market prediction. *International Journal of Intelligent Engineering and Systems*, *10*(3), 146–154. doi: 10.22266/ijies2017.0630.16

Buterin, V. (2014). *Ethereum: A next-generation smart contract and decentralized application platform.* Retrieved from `https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf`

Cavalli, S., & Amoretti, M. (2021). Cnn-based multivariate data analysis for bitcoin trend prediction. *Applied Soft Computing*, *101*. Retrieved from `www.scopus.com` (Cited By :18)

Cheng, Y., & Griffin, C. H. (2022). *Tesla vs. its stock price: "herd theory" at work?* (Vol. 16) (No. 1).

Cochrane, J. H. (2015, Apr). *Unit roots in english and pictures.* Retrieved from `https://johnhcochrane.blogspot.com/2015/04/unit-roots-in-english-and-pictures.html`

Critien, J. V., Gatt, A., & Ellul, J. (2022). Bitcoin price change and trend prediction through twitter sentiment and data volume. *Financial Innovation*, *8*(1). Retrieved from `www.scopus.com` (Cited By :1)

Delfabbro, P., King, D., & Williams, J. (2021, 06). The psychology of cryptocurrency trading: Risk and protective factors. *Journal of Behavioral Addictions*, *10*. doi: 10.1556/2006.2021.00037

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, June). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)* (pp. 4171–4186). Minneapolis, Minnesota: Association for Computational Linguistics. Retrieved from `https://aclanthology.org/N19-1423` doi: 10.18653/v1/N19-1423

Dickey, D., & Fuller, W. (1979, 06). Distribution of the estimators for autoregressive time series with a unit root. *JASA. Journal of the American Statistical Association*, *74*. doi: 10.2307/2286348

Durcheva, M., & Tsankov, P. (2019, Nov). Analysis of similarities between stock and cryptocurrency series by using graphs and spanning trees. *PROCEEDINGS OF THE 45TH INTERNATIONAL CONFERENCE ON APPLICATION OF MATHEMATICS IN ENGINEERING AND ECONOMICS (AMEE'19)*. doi: 10.1063/1.5133581

Eichler, M. (2012). Causal inference in time series analysis. In *Causality* (p. 327-354). John Wiley & Sons, Ltd. Retrieved from `https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119945710.ch22` doi: https://doi.org/10.1002/9781119945710.ch22

Fleischer, J. P., von Laszewski, G., Theran, C., & Bautista, Y. J. P. (2022). Time series analysis of cryptocurrency prices using long short-term memory. *Algorithms*, *15*(7). Retrieved from `www.scopus.com`

Frenken, K., & Schor, J. (2017). Putting the sharing economy into perspective. *Environmental Innovation and Societal Transitions*, *23*, 3-10. Retrieved from `https://www.sciencedirect.com/science/article/pii/S2210422417300114` (Sustainability Perspectives on the Sharing Economy) doi: https://doi.org/10.1016/j.eist.2017.01.003

Google-SearchLiaison. (2019a, Dec). *Bert, our new way for google search to better understand language, is now rolling out to over 70 languages worldwide. it initially launched in oct. for us english. you can read more about bert below; a full list of languages is in this thread.... https://t.co/nukvdg6hym.* Twitter. Retrieved from `https://twitter.com/searchliaison/status/1204152378292867074`

Google-SearchLiaison. (2019b, Oct). *Meet bert, a new way for google search to better understand language and improve our search results. it's now being used in the us in english, helping with one out of every 10 searches. it will come to more counties and languages in the future. pic.twitter.com/rj4ptc16zj.* Twitter. Retrieved from `https://twitter.com/searchliaison/status/1187732030399889409`

Goswami, A., Borasi, P., & Kumar, V. (2021, Jul). *Cryptocurrency market size, share and analysis: Forecast - 2030.* Retrieved from `https://www.alliedmarketresearch.com/crypto-currency-market`

Granger, C. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, *37*(3), 424–438. Retrieved 2022-07-27, from `http://www.jstor.org/stable/1912791`

Granger, C. (1980). Testing for causality. *Journal of Economic Dynamics and Control*, *2*, 329–352. doi: 10.1016/0165-1889(80)90069-x

Gu, S., Lillicrap, T. P., Sutskever, I., & Levine, S. (2016). Continuous deep q-learning with model-based acceleration. *CoRR*, *abs/1603.00748*. Retrieved from http://arxiv.org/abs/1603.00748

Hsieh, G., & Kocielnik, R. (2016, 02). You get who you pay for: The impact of incentives on participation bias. In (p. 821-833). doi: 10.1145/2818048.2819936

Huang, A., Wang, H., & Yang, Y. (2020, Aug). Finbert—a deep learning approach to extracting textual information. *SSRN Electronic Journal*. doi: 10.2139/ssrn.3910214

Jani, S. (2017, 12). An overview of ethereum & its comparison with bitcoin. *International Journal of Scientific & Engineering Research*, *10*.

Ji, S., Kim, J., & Im, H. (2019). A comparative study of bitcoin price prediction using deep learning. *Mathematics*, *7*(10). Retrieved from https://www.mdpi.com/2227-7390/7/10/898 doi: 10.3390/math7100898

Kahneman, D. (2011). *Thinking, fast and slow*. New York: Farrar, Straus and Giroux. Retrieved from https://www.amazon.de/Thinking-Fast-Slow-Daniel-Kahneman/dp/0374275637/

Kim, K., Lee, S.-Y., & Assar, S. (2021, 11). The dynamics of cryptocurrency market behavior: sentiment analysis using markov chains. *Industrial Management & Data Systems*, *ahead-of-print*. doi: 10.1108/IMDS-04-2021-0232

Kingma, D., & Ba, J. (2014, 12). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Kwiatkowski, D., Phillips, P. C., Schmidt, P., & Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics*, *54*(1), 159-178. Retrieved from https://www.sciencedirect.com/science/article/pii/030440769290104Y doi: https://doi.org/10.1016/0304-4076(92)90104-Y

Kwon, D., Kim, J., Heo, J., Kim, C., & Han, Y. (2019). Time series classification of cryptocurrency price trend based on a recurrent lstm neural network. *Journal of Information Processing Systems*, *15*(3), 694-706. Retrieved from www.scopus.com

Li, M. (2021). Prediction of bitcoin price based on the hidden markov model. In *Proceedings of the 2021 3rd international conference on economic management and cultural industry (icemci 2021)* (p. 2962-2967). Atlantis Press. Retrieved from https://doi.org/10.2991/assehr.k.211209.481 doi: https://doi.org/10.2991/assehr.k.211209.481

Liebau, D., & Schueffel, P. (2019, 01). Crypto-currencies and icos: Are they scams? an empirical study. *SSRN Electronic Journal*. doi: 10.2139/ssrn.3320884

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... Wierstra, D. (2015). *Continuous control with deep reinforcement learning.* arXiv. Retrieved from https://arxiv.org/abs/1509.02971 doi: 10.48550/ARXIV.1509.02971

Livieris:2021, I. E., Kiriakidou, N., Stavroyiannis, S., & Pintelas, P. (2021). An advanced cnn-lstm model for cryptocurrency forecasting. *Electronics (Switzerland)*, *10*(3), 1-16. Retrieved from www.scopus.com (Cited By :16)

Ma, Y. c. C., Wang, Z., & Fleiss, A. (2021). Deep q-learning for trading cryptocurrency. *The Journal of Financial Data Science*. Retrieved from https://jfds.pm-research.com/content/early/2021/06/08/jfds.2021.1.064 doi: 10.3905/jfds.2021.1.064

Mai, F., Shan, Z., Bai, Q., Wang, X. S., & Chiang, R. H. L. (2018). How does social media impact bitcoin value? a test of the silent majority hypothesis. *Journal of Management Information Systems*, *35*(1), 19-52. Retrieved from www.scopus.com

Markus, A. F., Kors, J. A., & Rijnbeek, P. R. (2021). The role of explainability in creating trustworthy artificial intelligence for health care: A comprehensive survey of the terminology, design choices, and evaluation strategies. *Journal of Biomedical Informatics*, *113*, 103655. Retrieved from https://www.sciencedirect.com/science/article/pii/S1532046420302835 doi: https://doi.org/10.1016/j.jbi.2020.103655

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, *abs/1301.3781*. Retrieved from http://dblp.uni-trier.de/db/journals/corr/corr1301.html#abs-1301-3781

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. doi: 10.1038/nature14236

Mohammad, S. M. (2016). 9 - sentiment analysis: Detecting valence, emotions, and other affectual states from text. In H. L. Meiselman (Ed.), *Emotion measurement* (p. 201-237). Woodhead Publishing. Retrieved from https://www.sciencedirect.com/science/article/pii/B9780081005088000096 doi: https://doi.org/10.1016/B978-0-08-100508-8.00009-6

Nadeau, R., Cloutier, E., & Guay, J.-H. (1993). New evidence about the existence of a bandwagon effect in the opinion formation process. *International Political Science Review / Revue internationale de science politique*, *14*(2), 203–213. Retrieved 2022-07-31, from http://www.jstor.org/stable/1601152

Nakamoto, S. (2008). *Bitcoin: A peer-to-peer electronic cash system.* Retrieved from http://www.bitcoin.org/bitcoin.pdf

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

Nascimento, K. K. F., Santos, F. S., Jale, J. S., Júnior, S. F. A. X., & Ferreira, T. A. E. (2022, Feb 11). Extracting rules via markov chains for cryptocurrencies returns forecasting. *Computational Economics*. Retrieved from https://doi.org/10.1007/s10614-022-10237-7 doi: 10.1007/s10614-022-10237-7

Polasik, M., Piotrowska, A., Wisniewski, T., Kotkowski, R., & Lightfoot, G. (2015, 09). Price fluctuations and the use of bitcoin: An empirical inquiry. *International Journal of Electronic Commerce*, *20*, 9-49. doi: 10.1080/10864415.2016.1061413

Ramadani, K., & Devianto, D. (2020, Nov). The forecasting model of bitcoin price with fuzzy time series markov chain and chen logical method. *INTERNATIONAL CONFERENCE ON SCIENCE AND APPLIED SCIENCE (ICSAS2020)*. doi: 10.1063/5.0032178

Sattarov, O., Muminov, A., Lee, C. W., Kang, H. K., Oh, R., Ahn, J., ... Jeon, H. S. (2020). Recommending cryptocurrency trading points with deep reinforcement learning approach. *Applied Sciences (Switzerland)*, *10*(4). Retrieved from www.scopus.com (Cited By :11)

Schnaubelt, M. (2022). Deep reinforcement learning for the optimal placement of cryptocurrency limit orders. *European Journal of Operational Research*, *296*(3), 993-1006. Retrieved from www.scopus.com (Cited By :5)

Shen, D., Urquhart, A., & Wang, P. (2019). Does twitter predict bitcoin? *Economics Letters*, *174*, 118-122. Retrieved from www.scopus.com (Cited By :120)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *CoRR*, *abs/1706.03762*. Retrieved from http://arxiv.org/abs/1706.03762

Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., & de Freitas, N. (2015). Dueling network architectures for deep reinforcement learning. arXiv. Retrieved from https://arxiv.org/abs/1511.06581 doi: 10.48550/ARXIV.1511.06581

What is ether (eth)? (n.d.). Retrieved from https://ethereum.org/en/eth/

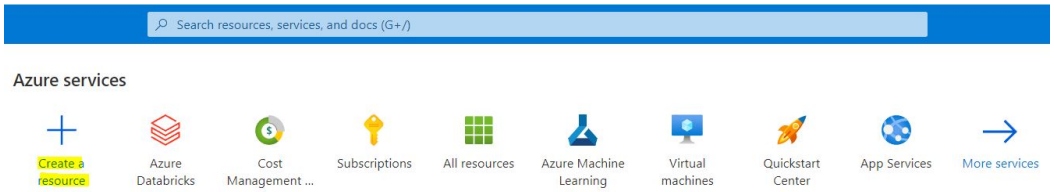
Woroniuk, D. (2021). *Historic crypto.* https://github.com/David-Woroniuk/Historic_Crypto. GitHub.

Zhu, Y., Kiros, R., Zemel, R. S., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *CoRR*, *abs/1506.06724*. Retrieved from http://arxiv.org/abs/1506.06724

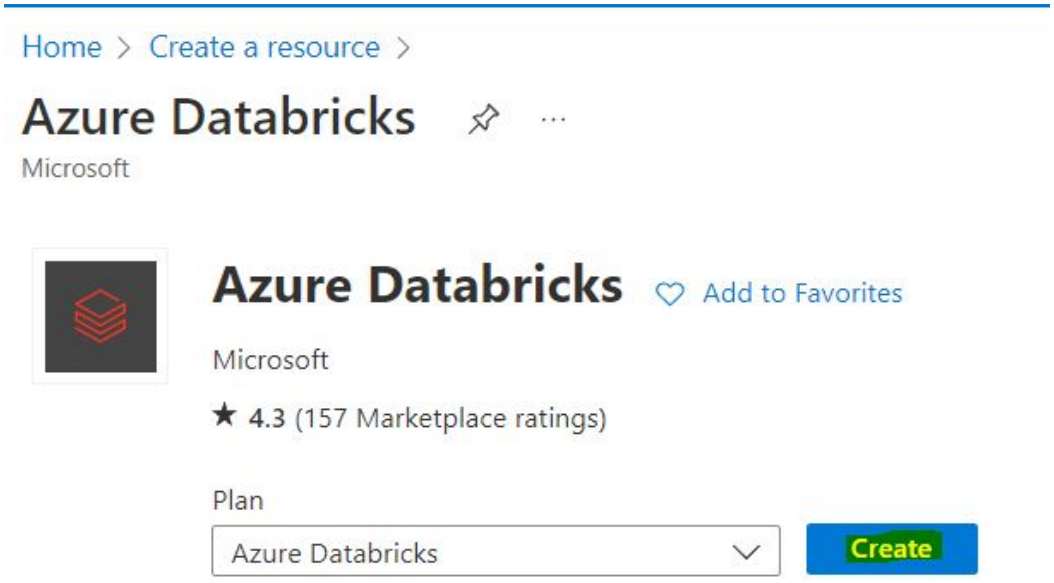
8 Appendices

8.1 Azure Databricks setup

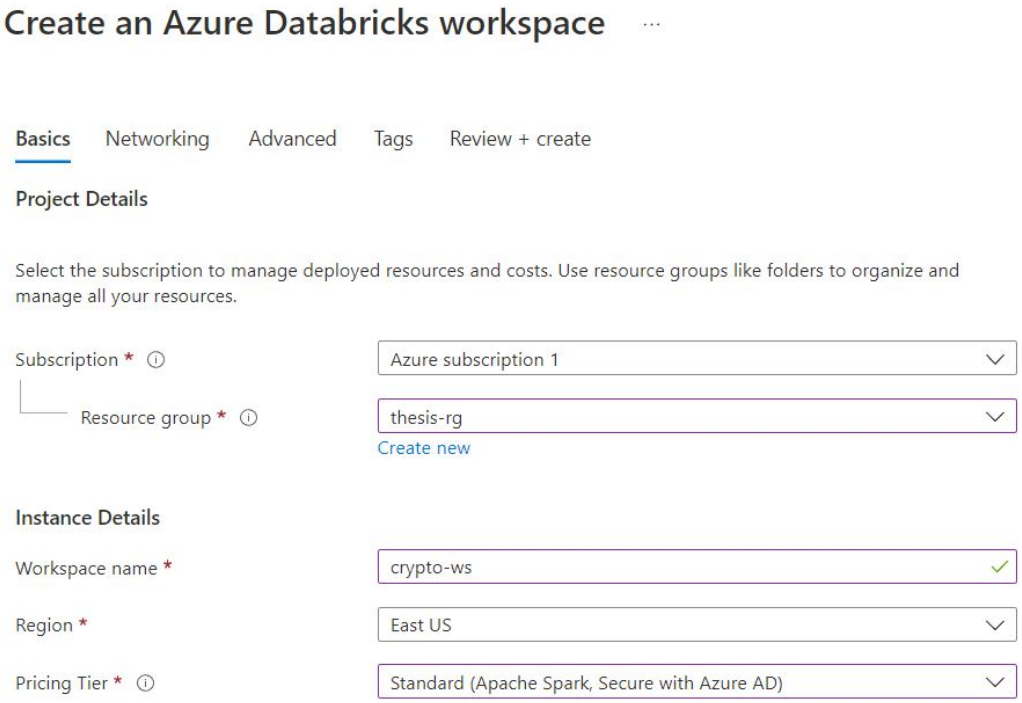
1. Navigate to https://portal.azure.com/ and login or create an account.
2. Click "Create a Resource" from the dashboard page:



3. Search for "Databricks", select the Azure Databricks plan, and click "Create":

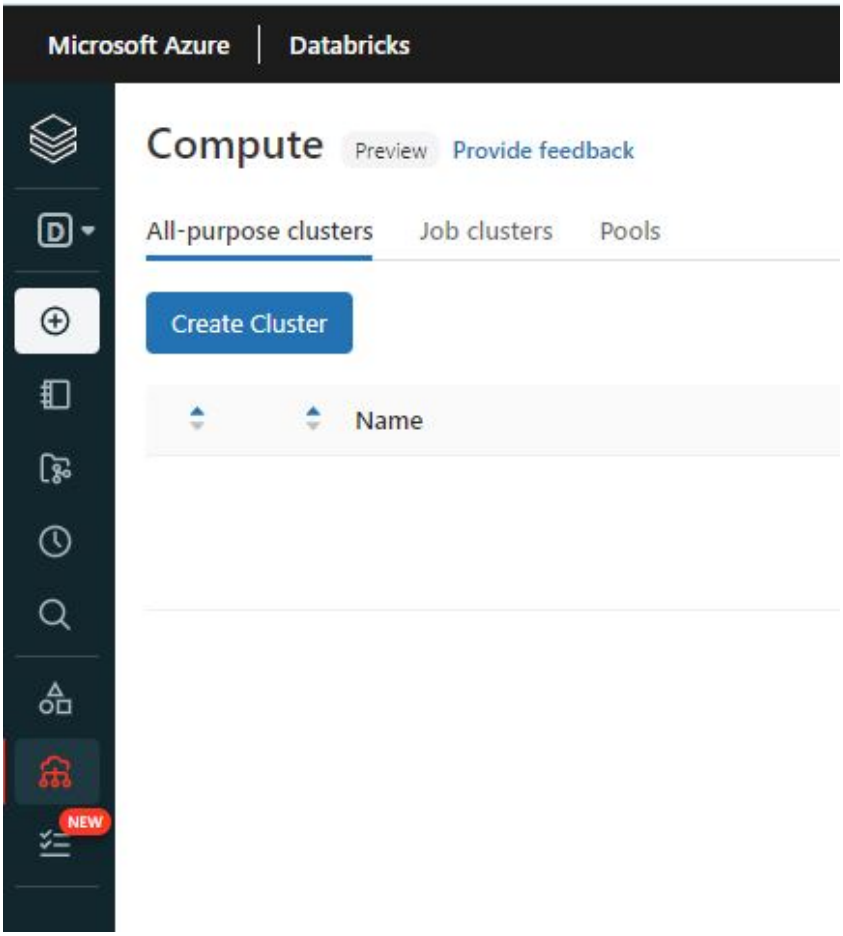


4. Select an Azure Subscription (used as a source of payment) and resource group. Then, fill in the instance details, noting that region pricing for virtual machines can differ significantly. If the Spark workflow does not require network streaming, It is encouraged to select the cheapest region for per-hour pricing of high-memory virtual machines. Finally, in the pricing tier, select Standard.

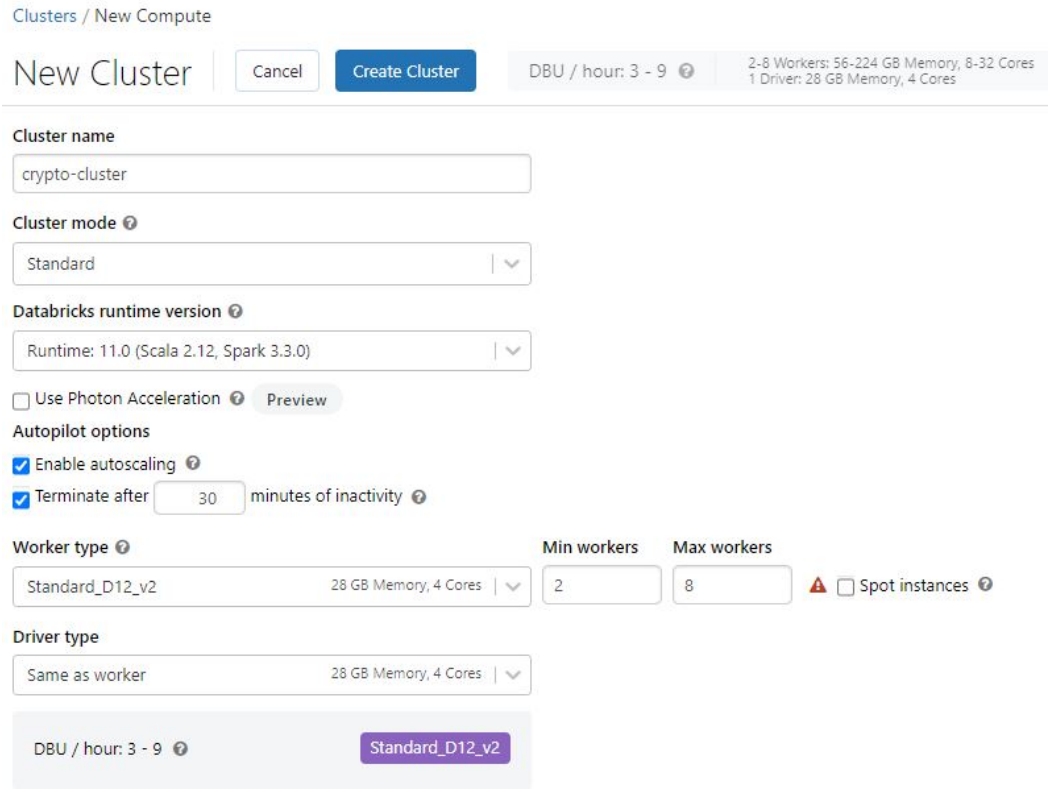


5. Wait for the deployment to complete (this may take several minutes), click "Go to resource", then "Launch Workspace".

6. Once Databricks opens and is signed in using Azure AD, select the Compute tab and click "Create Cluster" under the "All-purpose clusters" tab:

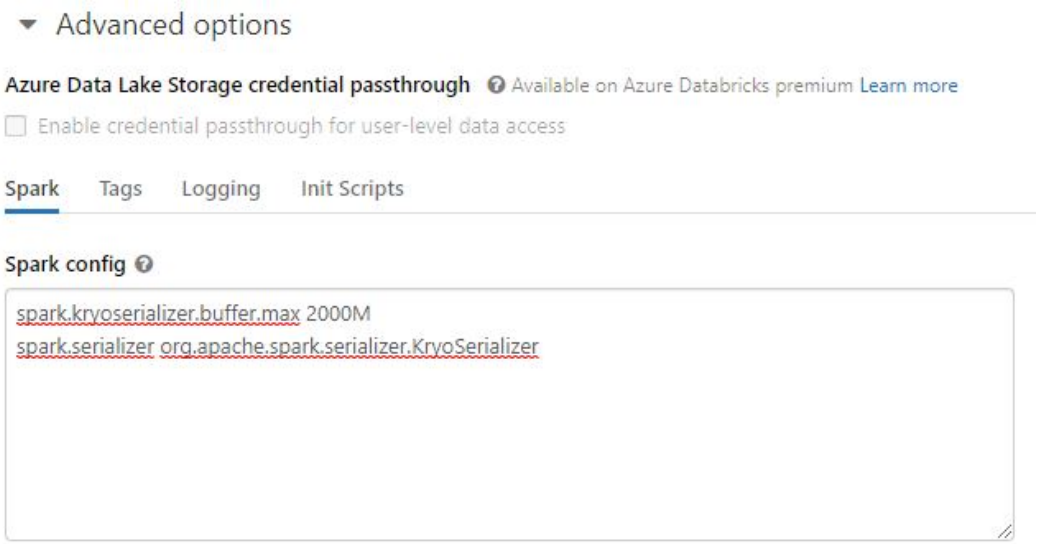


7. Complete the cluster configuration details, selecting "Standard_D12_V2" as the worker type, with 2-8 workers and autoscaling enabled. The driver type should be the same as the worker type:



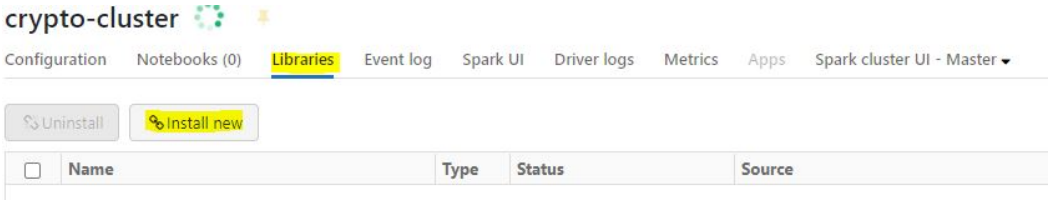
8. In the Advanced options, add the following two lines of Spark config code (from the Spark NLP installation guide), then click "Create Cluster":

    - spark.kryoserializer.buffer.max 2000M
    - spark.serializer org.apache.spark.serializer.KryoSerializer

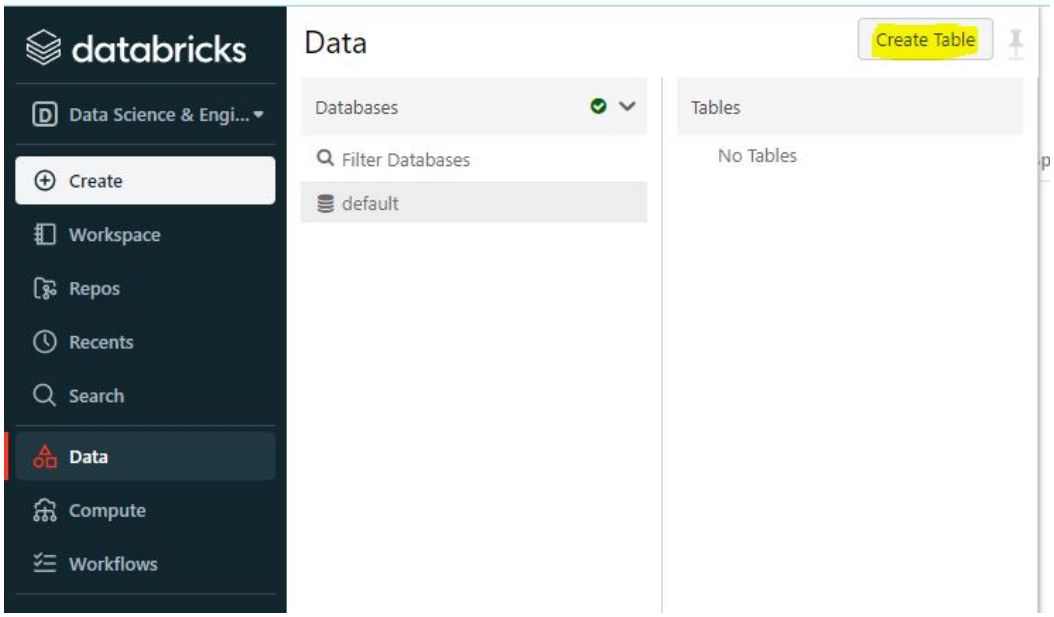


9. Select the "Libraries" tab, then "Install new". Add the following packages to the PyPi and Maven tabs respectively:

   - PyPi Package: spark-nlp
   - Maven Coordinates: com.johnsnowlabs.nlp:spark-nlp_2.12:4.0.2

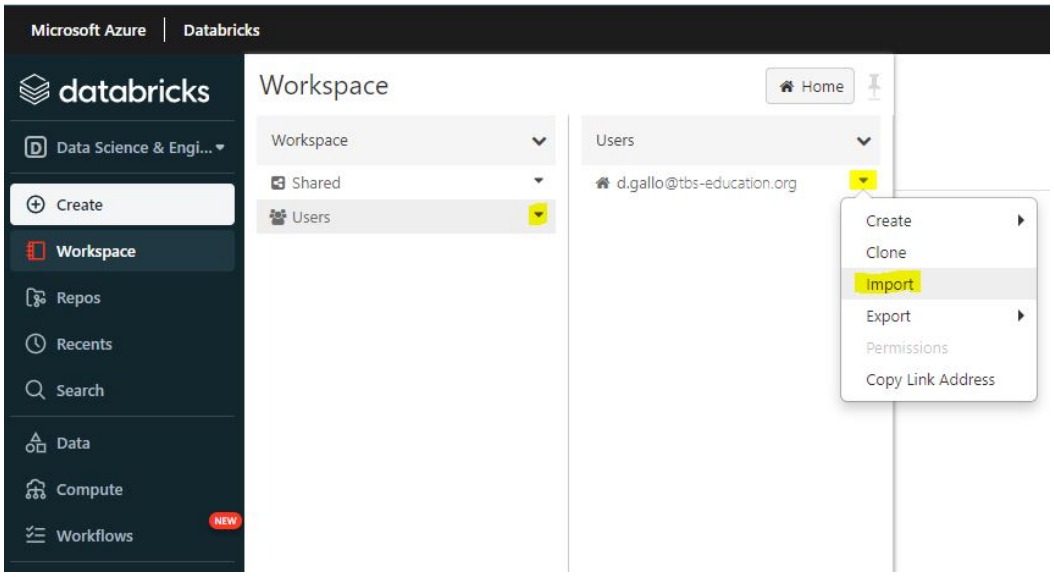


10. Select the "Data" tab, and click "Create Table". Upload the .csv file of collected Tweets, making note of the 2GB file size limit for the local I/O API:

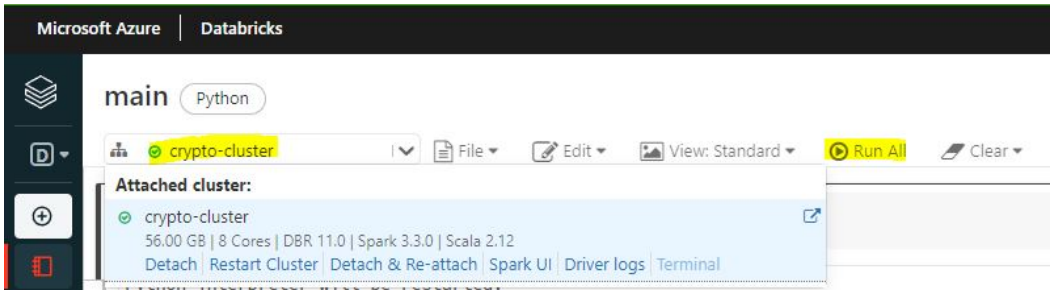


11. If the file is too large, it can be directly upload to the Azure Blob Storage bucket for the Databricks instance. If someone else is hosting the file, it can also be uploaded to Google Drive and downloaded to the DBFS FileStore using the *get_from_gdrive()* function in the code, being sure to make the sharing link public and filling in the required parameter for file id.

12. Select the "Workspace" tab, and navigate to the user space. Import the .ipynb Jupyter notebook or DBC archive for the project:



13. Open the notebook, attach the notebook to the compute cluster that was created, then run the code:



8.2 Spark dataframe *show()* method

```
+------------------+-------------------+------------------+------------------+------------------+--------------+-----------+--------------------+--------------------+--------------------+--------------------+
|                id|         created_at|         author_id|   conversation_id|            source|geo_coordinates|geo_place_id|            mentions|            hashtags|                urls|                text|
+------------------+-------------------+------------------+------------------+------------------+--------------+-----------+--------------------+--------------------+--------------------+--------------------+
|815361680775335936|2017-01-01 00:59:08|          98215866|815361680775335936|Twitter Web Client|          null|       null|        [2150123534]|[SimpleFX, ethere...|[http://www.newsb...|RT @newsbtc: Ethe...|
|815361254558695424|2017-01-01 00:57:26|745668049454374912|815361254558695424|             IFTTT|          null|       null|                null|          [Ethereum]|[http://ift.tt/2i...|What's the curren...|
|815361253971521536|2017-01-01 00:57:26|          196082542|815361253971521536|Twitter Web Client|          null|       null|                null|[eth, ethereum, b...|[https://www.redd...|What's the curren...|
|815360683864915968|2017-01-01 00:55:10|         2309346044|815360683864915968|         TweetDeck|          null|       null|        [2309346044]|[RT, blockchain, ...|[http://bit.ly/2f...|RT @Vindyne8: #RT...|
|815360660225871872|2017-01-01 00:55:04|         2309346044|815360660225871872|         TweetDeck|          null|       null|                null|[RT, blockchain, ...|[http://bit.ly/2f...|#RT Join ChronoBa...|
|815360291504463872|2017-01-01 00:53:36|809972460040945670|815360291504463872|      twittbot.net|          null|       null|                null|[bitcoin, blockch...|[https://goo.gl/q...|free ethereum eve...|
|815359388827402240|2017-01-01 00:50:01|         1491962095|815359388827402240|            Buffer|          null|       null|                null|                null|[http://buff.ly/2...|Create invoice sm...|
|815359086053261312|2017-01-01 00:48:49|          16997715|815359086053261312|         RoundTeam|          null|       null|[745668049454374912]|          [Ethereum]|[http://ift.tt/2i...|RT @r_ethereum: E...|
|815355255600926721|2017-01-01 00:33:36|          28471339|815355255600926721|Twitter for iPhone|          null|       null|         [841437061]|[Ethereum, Blockc...|                null|RT @DeepLearn007:...|
|815354951606140928|2017-01-01 00:32:23|796447976071827456|815354951606140928|       SocialOomph|          null|       null|                null|[blockchain, bitc...|[http://buff.ly/2...|How does #blockch...|
|815353076106178560|2017-01-01 00:24:56|809972460040945670|815353076106178560|      twittbot.net|          null|       null|                null|[bitcoin, blockch...|[https://goo.gl/S...|free ethereum eve...|
|815352664330534912|2017-01-01 00:23:18|         4845410254|815352664330534912|     EthereumVibes|          null|       null|        [1912522274]|     [CryptoTrading]|[http://dlvr.it/N...|RT @SportsbookBTC...|
|815352661956526081|2017-01-01 00:23:17|         4845410254|815352661956526081|     EthereumVibes|          null|       null|        [1388071068]|                null|                null|RT @julia_vaingur...|
|815352659444121600|2017-01-01 00:23:17|         4845410254|815352659444121600|     EthereumVibes|          null|       null|[1534275283888529...|[news, bitcoin, c...|[https://btcmanag...|RT @btc_manager: ...|
|815352507648053253|2017-01-01 00:22:40|          28471339|815352507648053253|Twitter for iPhone|          null|       null|[8381595149102653...|          [Ethereum]|[https://www.ethn...|RT @EthereumPress...|
|815352116638314498|2017-01-01 00:21:07|         2339916366|815352116638314498|Twitter Web Client|          null|       null|                null|   [ethereum, miner]|[https://www.indi...|Donate to our pro...|
|815351798349316097|2017-01-01 00:19:51|          22938169|815351798349316097|  Tweetbot for iOS|          null|       null|        [1068950834]|                null|                null|RT @IAmNickDodson...|
+------------------+-------------------+------------------+------------------+------------------+--------------+-----------+--------------------+--------------------+--------------------+--------------------+
```

Figure 28: Spark Dataframe with rows of Twitter data

8.3 Unit root testing

8.3.1 Ether closing price

Null Hypothesis: CLOSE has a unit root
Exogenous: Constant
Lag Length: 25 (Automatic - based on SIC, maxlag=96)

		t-Statistic	Prob.*
Augmented Dickey-Fuller test statistic		-1.390816	0.5885
Test critical values:	1% level	-3.430315	
	5% level	-2.861409	
	10% level	-2.566740	

*MacKinnon (1996) one-sided p-values.

Augmented Dickey-Fuller Test Equation
Dependent Variable: D(CLOSE)
Method: Least Squares
Date: 07/30/22 Time: 22:36
Sample (adjusted): 1/02/2017 02:00 7/20/2022 22:00
Included observations: 48621 after adjustments

Figure 29: ADF test on Ether closing price

Null Hypothesis: CLOSE is stationary
Exogenous: Constant
Bandwidth: 163 (Newey-West automatic) using Bartlett kernel

		LM-Stat.
Kwiatkowski-Phillips-Schmidt-Shin test statistic		16.69800
Asymptotic critical values*:	1% level	0.739000
	5% level	0.463000
	10% level	0.347000

*Kwiatkowski-Phillips-Schmidt-Shin (1992, Table 1)

Residual variance (no correction)	1411435.
HAC corrected variance (Bartlett kernel)	2.30E+08

KPSS Test Equation
Dependent Variable: CLOSE
Method: Least Squares
Date: 07/30/22 Time: 22:42
Sample: 1/01/2017 00:00 7/20/2022 22:00
Included observations: 48647

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	956.5463	5.386502	177.5821	0.0000

R-squared	0.000000	Mean dependent var	956.5463
Adjusted R-squared	-0.000000	S.D. dependent var	1188.050
S.E. of regression	1188.050	Akaike info criterion	16.99804
Sum squared resid	6.87E+10	Schwarz criterion	16.99822
Log likelihood	-413450.7	Hannan-Quinn criter.	16.99809
Durbin-Watson stat	0.000169		

Figure 30: KPSS test on Ether closing price

Null Hypothesis: D(CLOSE) has a unit root
Exogenous: Constant
Lag Length: 24 (Automatic - based on SIC, maxlag=96)

		t-Statistic	Prob.*
Augmented Dickey-Fuller test statistic		-46.39527	0.0001
Test critical values:	1% level	-3.430315	
	5% level	-2.861409	
	10% level	-2.566740	

*MacKinnon (1996) one-sided p-values.

Augmented Dickey-Fuller Test Equation
Dependent Variable: D(CLOSE,2)
Method: Least Squares
Date: 07/30/22 Time: 23:16
Sample (adjusted): 1/02/2017 02:00 7/20/2022 22:00
Included observations: 48621 after adjustments

Figure 31: ADF test on first order difference Ether closing price

Null Hypothesis: D(CLOSE) is stationary
Exogenous: Constant
Bandwidth: 20 (Newey-West automatic) using Bartlett kernel

		LM-Stat.
Kwiatkowski-Phillips-Schmidt-Shin test statistic		0.081050
Asymptotic critical values*:	1% level	0.739000
	5% level	0.463000
	10% level	0.347000

*Kwiatkowski-Phillips-Schmidt-Shin (1992, Table 1)

Residual variance (no correction)	238.7369
HAC corrected variance (Bartlett kernel)	224.1141

KPSS Test Equation
Dependent Variable: D(CLOSE)
Method: Least Squares
Date: 07/30/22 Time: 22:40
Sample (adjusted): 1/01/2017 01:00 7/20/2022 22:00
Included observations: 48646 after adjustments

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	0.032508	0.070055	0.464035	0.6426

R-squared	0.000000	Mean dependent var	0.032508
Adjusted R-squared	-0.000000	S.D. dependent var	15.45127
S.E. of regression	15.45127	Akaike info criterion	8.313280
Sum squared resid	11613596	Schwarz criterion	8.313461
Log likelihood	-202202.9	Hannan-Quinn criter.	8.313337
Durbin-Watson stat	1.991913		

Figure 32: KPSS test on first order difference Ether closing price

8.3.2 Sentiment scores

Null Hypothesis: AVG_SENTIMENT_NUM has a unit root
Exogenous: Constant
Lag Length: 24 (Automatic - based on SIC, maxlag=96)

		t-Statistic	Prob.*
Augmented Dickey-Fuller test statistic		-18.53457	0.0000
Test critical values:	1% level	-3.430316	
	5% level	-2.861409	
	10% level	-2.566741	

*MacKinnon (1996) one-sided p-values.

Augmented Dickey-Fuller Test Equation
Dependent Variable: D(AVG_SENTIMENT_NUM)
Method: Least Squares
Date: 07/30/22 Time: 22:55
Sample (adjusted): 1/02/2017 01:00 7/21/2022 15:00
Included observations: 48457 after adjustments

Figure 33: ADF test on sentiment scores

Null Hypothesis: AVG_SENTIMENT_NUM is stationary
Exogenous: Constant
Bandwidth: 150 (Newey-West automatic) using Bartlett kernel

		LM-Stat.
Kwiatkowski-Phillips-Schmidt-Shin test statistic		6.552636
Asymptotic critical values*:	1% level	0.739000
	5% level	0.463000
	10% level	0.347000

*Kwiatkowski-Phillips-Schmidt-Shin (1992, Table 1)

Residual variance (no correction)	0.003040
HAC corrected variance (Bartlett kernel)	0.128169

KPSS Test Equation
Dependent Variable: AVG_SENTIMENT_NUM
Method: Least Squares
Date: 07/30/22 Time: 22:56
Sample: 1/01/2017 00:00 7/21/2022 15:00
Included observations: 48647

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	0.084733	0.000250	338.9507	0.0000

R-squared	0.000000	Mean dependent var	0.084733
Adjusted R-squared	0.000000	S.D. dependent var	0.055137
S.E. of regression	0.055137	Akaike info criterion	-2.957973
Sum squared resid	147.8878	Schwarz criterion	-2.957792
Log likelihood	71949.26	Hannan-Quinn criter.	-2.957916
Durbin-Watson stat	1.048110		

Figure 34: KPSS test on sentiment scores

Null Hypothesis: D(AVG_SENTIMENT_NUM) has a unit root
Exogenous: Constant
Lag Length: 47 (Automatic - based on SIC, maxlag=96)

		t-Statistic	Prob.*
Augmented Dickey-Fuller test statistic		-49.79536	0.0001
Test critical values:	1% level	-3.430316	
	5% level	-2.861409	
	10% level	-2.566741	

*MacKinnon (1996) one-sided p-values.

Augmented Dickey-Fuller Test Equation
Dependent Variable: D(AVG_SENTIMENT_NUM,2)
Method: Least Squares
Date: 07/30/22 Time: 23:17
Sample (adjusted): 1/03/2017 01:00 7/21/2022 15:00
Included observations: 48361 after adjustments

Figure 35: ADF test on first order difference sentiment scores

Null Hypothesis: D(AVG_SENTIMENT_NUM) is stationary
Exogenous: Constant
Bandwidth: 1.6e+03 (Newey-West automatic) using Bartlett kernel

		LM-Stat.
Kwiatkowski-Phillips-Schmidt-Shin test statistic		0.169250
Asymptotic critical values*:	1% level	0.739000
	5% level	0.463000
	10% level	0.347000

*Kwiatkowski-Phillips-Schmidt-Shin (1992, Table 1)

Residual variance (no correction)	0.003186
HAC corrected variance (Bartlett kernel)	4.59E-06

KPSS Test Equation
Dependent Variable: D(AVG_SENTIMENT_NUM)
Method: Least Squares
Date: 07/30/22 Time: 23:18
Sample (adjusted): 1/01/2017 01:00 7/21/2022 15:00
Included observations: 48633 after adjustments

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	-3.27E-06	0.000256	-0.012766	0.9898

R-squared	0.000000	Mean dependent var	-3.27E-06
Adjusted R-squared	-0.000000	S.D. dependent var	0.056448
S.E. of regression	0.056448	Akaike info criterion	-2.910964
Sum squared resid	154.9614	Schwarz criterion	-2.910783
Log likelihood	70785.45	Hannan-Quinn criter.	-2.910907
Durbin-Watson stat	2.925599		

Figure 36: KPSS test on first order difference sentiment scores

8.3.3 Number of Tweets

Null Hypothesis: TWEET_COUNT has a unit root
Exogenous: Constant
Lag Length: 96 (Automatic - based on SIC, maxlag=96)

		t-Statistic	Prob.*
Augmented Dickey-Fuller test statistic		0.348248	0.9807
Test critical values:	1% level	-3.430315	
	5% level	-2.861409	
	10% level	-2.566740	

*MacKinnon (1996) one-sided p-values.

Augmented Dickey-Fuller Test Equation
Dependent Variable: D(TWEET_COUNT)
Method: Least Squares
Date: 07/31/22 Time: 12:04
Sample (adjusted): 1/05/2017 01:00 7/20/2022 23:00
Included observations: 48551 after adjustments

Figure 37: ADF test on number of Tweets

Null Hypothesis: TWEET_COUNT is stationary
Exogenous: Constant
Bandwidth: 161 (Newey-West automatic) using Bartlett kernel

		LM-Stat.
Kwiatkowski-Phillips-Schmidt-Shin test statistic		17.44867
Asymptotic critical values*:	1% level	0.739000
	5% level	0.463000
	10% level	0.347000

*Kwiatkowski-Phillips-Schmidt-Shin (1992, Table 1)

Residual variance (no correction)	20857704
HAC corrected variance (Bartlett kernel)	3.12E+09

KPSS Test Equation
Dependent Variable: TWEET_COUNT
Method: Least Squares
Date: 07/31/22 Time: 02:12
Sample (adjusted): 1/01/2017 00:00 7/20/2022 23:00
Included observations: 48648 after adjustments

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	2986.724	20.70643	144.2414	0.0000

R-squared	0.000000	Mean dependent var	2986.724
Adjusted R-squared	0.000000	S.D. dependent var	4567.071
S.E. of regression	4567.071	Akaike info criterion	19.69115
Sum squared resid	1.01E+12	Schwarz criterion	19.69133
Log likelihood	-478966.6	Hannan-Quinn criter.	19.69121
Durbin-Watson stat	0.019934		

Figure 38: KPSS test on number of Tweets

Null Hypothesis: D(TWEET_COUNT) has a unit root
Exogenous: Constant
Lag Length: 96 (Automatic - based on SIC, maxlag=96)

		t-Statistic	Prob.*
Augmented Dickey-Fuller test statistic		-34.65435	0.0000
Test critical values:	1% level	-3.430315	
	5% level	-2.861409	
	10% level	-2.566740	

*MacKinnon (1996) one-sided p-values.

Augmented Dickey-Fuller Test Equation
Dependent Variable: D(TWEET_COUNT,2)
Method: Least Squares
Date: 07/31/22 Time: 02:11
Sample (adjusted): 1/05/2017 02:00 7/20/2022 23:00
Included observations: 48550 after adjustments

Figure 39: ADF test on first order difference number of Tweets

Null Hypothesis: D(TWEET_COUNT) is stationary
Exogenous: Constant
Bandwidth: 384 (Newey-West automatic) using Bartlett kernel

		LM-Stat.
Kwiatkowski-Phillips-Schmidt-Shin test statistic		0.120814
Asymptotic critical values*:	1% level	0.739000
	5% level	0.463000
	10% level	0.347000

*Kwiatkowski-Phillips-Schmidt-Shin (1992, Table 1)

Residual variance (no correction)	415782.2
HAC corrected variance (Bartlett kernel)	4575.097

KPSS Test Equation
Dependent Variable: D(TWEET_COUNT)
Method: Least Squares
Date: 07/31/22 Time: 02:11
Sample (adjusted): 1/01/2017 01:00 7/20/2022 23:00
Included observations: 48647 after adjustments

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	0.284601	2.923542	0.097348	0.9225

R-squared	0.000000	Mean dependent var	0.284601
Adjusted R-squared	-0.000000	S.D. dependent var	644.8184
S.E. of regression	644.8184	Akaike info criterion	15.77584
Sum squared resid	2.02E+10	Schwarz criterion	15.77602
Log likelihood	-383722.5	Hannan-Quinn criter.	15.77589
Durbin-Watson stat	1.936679		

Figure 40: KPSS test on first order difference number of Tweets

9 About the author

This thesis was written exclusively by myself, David Gallo. You can read more about me at https://dgallo.ca/

I am a technology consultant and data scientist by profession, often working in positions that need both strong data and programming knowledge as well as business acumen. I was born and raised in Ottawa, Canada, where I completed an Honours Bachelors of Commerce specialized in Business Technology Management, with the French immersion and Co-operative Education options. I graduated with highest distinction, Summa Cum Laude, before moving to France for a Master of Science in Artificial Intelligence and Business Analytics at the Toulouse Business School. This thesis serves as the last requirement for my master's candidacy.

My past work experience includes business technology consultancy at Deloitte. As an analyst and consultant with the firm, I delivered a wide variety of technology transformation projects, from strategy to implementation. My projects focused on digital modernization for organizations, mostly for data and information management.

I had previously worked at CIRA, where I was the data scientist for a new DNS firewall product working with an agile team. I built machine learning workflow with Logstash, ElasticSearch, MySQL, Python, and Kibana to process DNS data in real-time, and hierarchical density-based clustering techniques with Python libraries to analyze millions of daily DNS requests, in an integrated AWS environment.

Before that, I worked at the Department of National Defence for the Government of Canada as a database administrator, at the Department of Fisheries and Oceans as a legal correspondence officer, at MNP as a developer/analyst, and at the Department of National Defence as a records manager.

On the side, I enjoy making things with my 3D printer, programming, playing music (both drums and piano), D&D, and playing soccer.