LUISS

MSc Data Science and Management

Course of Data Driven Models for Investment

Optimising Financial Decision-Making: An Integrated Approach to Automated Investment Models through Machine Learning and Advanced Financial Indicators

Prof. Elio Stocchi

SUPERVISOR

Dr. Emanuele Ricco

CO-SUPERVISOR

Spencer Marcinko

CANDIDATE

Academic Year 2023/2024

Abstract

This dissertations aims to delve into the nuances of algorithmic trading (algo-trading)- exploring and expanding upon existing trading strategies, whilst attempting to create a near autonomous and customised algorithm for users of varying backgrounds. Using a combination of machine learning techniques- specifically reinforcement learning and supervised learning- and considering financial technical indicators, we aim to an optimal strategy for trading stocks. The adaptability of the proposed algorithm extends to both long-term strategies, such as weekly or monthly trading intervals, and can be adapted to work for more short-term approaches, catering to hourly or daily transactions.

Our research tackles this intricate challenge by treating historical stock prices as a dynamic environment. The environment is embedded with information pertaining to the technical indicators on the data, which aim to guide and aid the agent in making more informed decisions. This results in a robust and adaptable algorithm that is applicable to a number of different situations and stock markets. These indicators play a pivotal role in gauging the health of the stock, providing crucial insights into potential stock trends and assisting in risk mitigation in stock trading.

The algorithm employs a multi-layered perceptron (MLP), as the agent. This robust algorithm navigates the dynamic environment to learn and refine an optimal policy for buying, holding and selling stocks. The MLP provides a significant advantage over standard learning algorithms such as q-learning in the form of increased adaptability and improved learning rate. This is vital for capturing the complex patterns demonstrated in the stock market.

The overarching goal of the algorithm is to maximise returns given an opening balance. The return in this case is considered as a combination of the current balance and the current price of the held stocks. The algorithm can be modified to either act more aggressively- buying more stocks when the conditions for buying are favourable- or more conservatively- restricting the purchases to a certain percentage of the current balance

under specified conditions.

The study culminated in the development and comparison of three distinct but related algorithms each with their own merits of use. Each algorithm revolved around using reinforcement leaning with proximal policy optimisation (PPO) as the agent. The primary differences reside in the inclusion and method of use of the financial technical indicator, 'Moving Average Convergence Divergence' (MACD). The use of these indicators proved invaluable in risk mitigation, but often resulted in lower returns than the algorithm that did not include the constraints concerning the values.

Overall, each of the algorithms demonstrated their utility by drastically increasing the user's net worth over the allotted time period, without requiring in-depth understand of the intricacies of stock trading. This serves to lower barrier of entry imposed around the stock market, allowing access to individuals with varying levels of trading knowledge.

Acknowledgements

This dissertations was created with the help of Quaternion Technology, a company that deals in financial stock trading using high level algorithms including machine learning techniques. Without the teachings presented by this company, this dissertation would not be possible. In particular, Dr. Ricco and Prof. Stocchi, who provided the necessary ground work of understanding in the field of data driven models for investment. I would like to thank my parents for their extremely important role in affording me the opportunities of studying further throughout my life. My sister and brother who have always played a role in motivating and helping me in what ever way they are able. The University of LUISS and its lecturers for accepting me into their prestigious academy- allowing me the opportunity to reside in Europe - furthering not only my understand of academics but my cultural understanding too. Finally, I would like to thank all of friends from Italy and all around the world, without whom I would not be where I am today.

Contents

1 Introduction **1**

1.1 Background of the study . . . 2
1.2 Problem statement . . . 2
1.3 Study objectives . . . 3
1.4 Research questions . . . 4
1.5 Significance of the study . . . 4
1.6 Scope and limitations . . . 5

2 Theoretical Framework **7**

2.1 Reinforcement learning fundamentals . . . 7
2.1.1 Markov chains . . . 8
2.1.2 Markov decision processes . . . 10
2.1.3 Dynamic programming . . . 11
2.1.4 Reinforcement Learning . . . 12
2.2 Supervised learning fundamentals . . . 13
2.2.1 Artificial Neural Networks . . . 13
2.2.2 Single layered perceptron forward pass . . . 14
2.2.3 Single layered perceptron back pass . . . 15
2.2.4 Multi-layered perceptron conceptualised . . . 16
2.3 Financial technical indicators . . . 16
2.3.1 Efficient Market hypothesis . . . 17
2.3.2 Types of financial indicators . . . 17
2.3.3 Technical indicator functions . . . 18
2.4 Integration of Techniques . . . 19
2.5 Application on financial markets . . . 20

3 Literature review **21**

3.1 The use of financial technical indicators . . . 21
3.2 Deep reinforcement learning in portfolio management . . . 23
3.3 Proximal Policy Optimisation Algorithms . . . 25
3.4 Considerations from literature review: . . . 27

4 Materials and Methods 28
4.1 Implementation Details . . . 28
4.2 Data Sources and Preprocessing . . . 30
4.3 Algorithm Implementation . . . 33
4.4 Evaluation Metrics and Performance Analysis . . . 34
4.5 Experiment Design and Validation . . . 36
4.6 Implementation Challenges and Solutions . . . 37
4.7 Ethical Considerations and Regulatory Compliance . . . 38
4.7.1 European Union . . . 39
4.7.2 United States . . . 39
4.7.3 Compliance Measures Implemented . . . 40

5 Experiments and Results 41
5.1 Deep RL Method . . . 41
5.2 Pure Technical Indicator Method . . . 43
5.3 Strategic Technical Indicator Method . . . 45

6 Discussion 48
6.1 Evaluation of Reinforcement Learning Algorithm . . . 48
6.2 Impact of Technical Indicators . . . 50
6.3 Effectiveness of Strategic Technical Indicators . . . 51
6.4 Comparison of Algorithms . . . 51
6.5 Addressing Research Questions . . . 53
6.6 Market Conditions, Algorithmic Design and Data Quality . . . 54
6.7 Limitations and Future Work . . . 55

7 Conclusion 57

A The First Appendix 59

List of Figures

2.1 Single layer perceptorn model 14

4.1 MACD for Apple dataset 31

5.1 Graph indicating the actions of the highest yielding model using deep RL over 100 iterations . 42
5.2 Graph showing the net worth over time of the highest yielding model using deep RL over 100 iterations 42
5.3 Graph showing the net worth over time of the highest yielding model using the Pure Technical Indicator method over 100 iterations 44
5.4 Graph indicating the actions of the highest yielding model using the Pure Technical Indicator method over 100 iterations 44
5.5 Graph showing the net worth over time of the highest yielding model using Strategic Technical Indicator method over 100 iterations 46
5.6 Graph indicating the actions of the highest yielding model using Strategic Technical Indicator method over 100 iterations 46

A.1 Scatter plot showing the resulting netwoths of all of the 100 iterations of running the pure RL algorithm 59
A.2 Scatter plot showing the resulting netwoths of all of the 100 iterations of running the pure technical indicator algorithm 59
A.3 Scatter plot showing the resulting netwoths of all of the 100 iterations of running the strategic technical indicator algorithm 60

List of Tables

3.1 Differences between PPO and DDPG algorithms 24
3.2 Generalised explanation of nn policy update 26
3.3 Key Considerations from the Literature Review 27

4.1 Environment design 33
4.2 Starting values in the environment 33
4.3 Constraints for buying stocks in each of the algorithms 34
4.4 Table showing the MACD bin and the corresponding percentage of stocks to trade. 35

5.1 Descriptive statistics for the generated net worth of the Deep RL method over 100 trials 41
5.2 Descriptive statistics for the generated net worth of the Pure Technical Indicator method over 100 trials 43
5.3 Descriptive statistics for the generated net worth of the Strategic Technical Indicator method over 100 trials 45

6.1 Transaction Fees vs. Risk Levels 52

Chapter 1

Introduction

- *Project rationale:* Traditionally, the stock market has been largely dominated by person-to-person (p2p) transactions. By denotation- if an individual had an interest in entering the stock market- they would likely have to approach a professional in this field, or devote a large amount of time to studying in order to properly understand the stock market for themselves. In this dissertation, we aim to formulate an automated and optimised stock trading algorithm that combines the power of financial techniques and machine learning, in order to allow laypersons to be able to buy, sell and hold stocks with the understanding that these decisions are based purely on statistically significant results- based on historical data. This can not only save the interested party a large amount of time in learning stock trends- but can prove to be a more effective and possibly lucrative method for undertaking this particular activity.

- *Project scope:* This project will delve into the world of finance and machine learning- explaining the fundamental concepts necessary for the generation, and merging of the different machine learning techniques and financial techniques used. It will compare the results of applying different variations of the algorithm, providing each of the algorithms with the same information. Each variation will differ from the other only in constraints added to the ability for the code to take actions in the environment.

- *Project organisation:* Following this introductory section, there will be a deep dive into to the concepts and functionalities of the different ML techniques and financial technical indicators used in the creation of these algorithms. Followed by a comprehensive literature review section that will incorporate the findings and lay the ground work of understanding on some of the important techniques of previously used financial techniques for stock trading- and algo-trading. Chapter 4 will be focused on the work that was done on the creation of the algorithm, from data collection to algorithm implementation. This

will be followed by a section that explains the results and findings of using the algorithm under test conditions. The culmination of the paper being a section concluding thought and recommendations- which will consider each of the algorithms and consider the advantages and disadvantages of employing each of the strategies for future use cases.

1.1 Background of the study

Throughout the evolution of stock trading market, there has been an increasing use of technology, in particular, the demand for increased information and market insights are in significantly high demand. According to an article entitled; 'Algorithmic Trading History: A Brief Summary', , Smigel (2023) The earliest form of information arbitrage that was used for this purpose was awarded to Nathan Mayer Rothschild. Who reportedly used carrier pigeons to learn about the goings on in the war between England and France- this information proved highly lucrative for the Rothschilds- who were able to make more informed decisions- and chose to hold their stocks during a period where people where selling their assets in a state of panic. This occurred in the early 1800's and the race for acquiring more information pertaining to the stock market has been growing increasingly competitive.

It was not until over 100 years later, that we saw a firm using an algorithm for trading for the very first time. In 1949, 'Richard Donchian launched the first trading rule-based fund' , (Smigel 2023). This algorithm was based on moving averages and was a far cry from the complexities of current day trading algorithms. All of these algorithms and trading strategies have one underlying, crucial dependence that came about over years of technological advancements. An increase in information. This information comes in the form of data. Stock traders learned to consider this new data, and attempted to formulate educated guesses about the future state of the stocks. This increased information was seen as a valuable commodity for stock traders, and it was subsequently made more available. The ability to trade stocks online and the increased accessibility to historical data on a number of stocks allows users to create and fine tune predictive algorithms that rely on powerful concepts and statistical principles to trade in these environments.

1.2 Problem statement

Amidst the wealth of newly available financial data, and with an understanding of some of the most sophisticated statistical tools at our disposal- this dissertation endeavors to merge one of the oldest trades in the world with some of the newest technologies. The overarching goal being to find the most suitable ways to leverage these technologies in order to better understand how to utilise them for decision making processes on the stock market. From , Smigel (2023) it is evident that despite there being a plethora of attempts to automate this process- many have fallen short of the mark.

An insight shared by a seasoned stock trader Troy Murray highlights a fundamental aspect of the dynamics of the stock market: ”The only true measure of the markets is from fear and greed.” This emphasised the importance of market sentiment in this sector, and highlights the need for more sophisticated methods for extracting useful information on when to trade. It is, however, crucial to realise that the this sentiment is not the only factor that might affect the stock prices. ”Hedge funds and institutional investors have a significant impact on market dynamics due to their large trading volumes and sophisticated strategies. Their actions can create trends that other market participants follow.” , Lo (2010). Additionally, economic indicators such as interest rates, inflation and employment data may serve as useful insight into the general state of the economy, which likely has some underlying connection about the sentiment of stock investments, which could prove to shape the stock environment itself.

It is therefore important to note that the proposed algorithm should not simply buy, sell or hold based on simplistic thresholds of stock prices- although this is a powerful tool to consider- one should endeavor to capture the sentiment of the market at certain times and use this information to make more informed decisions on what to do. Whilst incorporating information about the movements of large stockholders and key market indicators such as interest rates and inflation could potentially enhance the algorithm’s performance, this study will focus specifically on leveraging market sentiment and technical indicators like the MACD. This approach allows for a more streamlined analysis and practical implementation, ensuring the development of a robust algorithm without the additional complexity of integrating diverse market factors.

With the above in consideration, this research explores how more modern techniques namely, deep reinforcement learning can be leveraged in order to capture complex patterns expressed in historical stock data. Utilising these traits in order to formulate a robust strategy for trading in a dynamic market environment. The proposed model offers a solution that is effective to those with little to no financial know how- ensuring that individuals without extensive trading experience can still benefit from this algorithm, thereby paving the way for a more democratised stock market.

1.3 Study objectives

This study aims to bridge the gap between traditional stock trading and modern technologies by leveraging some advanced statistical techniques and predefined machine learning algorithms in order to learn some of the highly complex patterns generated in stock price movements in the past- in order to improve stock trading in future transactions. In order to achieve this goal, this paper will provide insight into the fundamentals of Markov decision processes- and the subsequent development of reinforcement learning. It will also explain the concept of an MLP in order to ensure the techniques used are fully understood before they are employed.

Further elaboration will be made upon some basic statistics, and the development and implementation of some financial technical indicators- exponential moving average, and MACD. It is important to illustrate the underlying concepts that were used to derive the algorithms and functions used in this paper in order to emphasise the role of statistics in financial analysis and highlight their importance in these information trading decisions.

By elucidating the theoretical foundations of these algorithmic trading techniques, this paper aims to 'lift the veil' on their operations, providing readers with the essential knowledge needed to understand the algorithms used throughout this study. Real-world examples will be presented in several case studies, demonstrating the practicality of these techniques and further explaining their functionality and significance in the field.

The study continues by applying and testing the three algorithms, comparing their results and discussing the advantages and disadvantages of each of them. The goal being to generate an algorithm that fits the user's preference in terms of associated risk compared to algorithmic greed. The algorithms generated are intended to require minimal interference from the user in order to run efficiently, making them easy for any person to use. Risk in the algorithms is a big factor throughout this paper and thus in each algorithm, there is an attempt to mitigate some of the risk while still ensuring a good return on investment for the user.

1.4 Research questions

The fundamental question that will be explored throughout this research paper is :"how effective is reinforcement learning in algo-trading?". This question will be broken down into three sub-questions.

1. Does the reinforcement learning algorithm work consistently for increasing the user's net worth in a stock trading environment?

2. Can the algorithm be enhanced with the help of technical indicators in a way that mitigates risk?

3. How much risk is mitigated and how do the algorithms compare to each other in general performance?

Additionally, the paper will strive to look at questions pertaining to market conditions, algorithmic design and data quality to help provide a more concrete analysis on the efficacy of the algorithms in question.

1.5 Significance of the study

By providing evidence of the effectiveness of these algorithms, the implication of a more democratised stock market may prove extremely valuable in the current economic climate. The empowerment of the individual to

enter the stock market with a statistically powerful algorithm- even with little to no prior knowledge of any of the information provided in this paper- is considerable. This could allow for major economical advancements to be made by individuals who may otherwise have not had the opportunity to acquire the necessary expertise or resources to navigate the market on their own.

Further, the innovative methods explained throughout this paper not only serve to pave a new way for users to invest stocks, but the techniques and algorithms displayed may be used in manors hither-to unexplored. Considering the general volatility of the stock market- if these seemingly random patterns are 'discerned' by the algorithms provided by this paper, then the idea that modified variations of these algorithms may be used in other sectors is not unfounded. This could prove highly useful for future analyses to be done on the trading market as well as investment profiles, fostering innovation and entrepreneurship within the sector. This may change the way future financial sectors operate.

In addition to increasing individual empowerment and innovation within the financial field, the usefulness of the algorithms presented in this case study extend far beyond the limits of financial trading. With the necessary modifications, this algorithm can be used to view other sets of complex data with irregular models. The use of reinforcement learning algorithms can be extended from examples such as city planning to medical usage. For instance, they could be used to optimise transportation routes or identify optimal delivery pathways for drug treatments. The variety of possibilities of use for these algorithms extend themselves to the user's imagination. This study, however, will be solely focused on the use of these algorithms in stock trading, but it could facilitate studies into further fields in the future.

When one considers the potential for empowerment emphasised above, it leads to the idea of more investors entering the stock market. This surge of investors entering into the stock market, has the potential to stimulate economic growth. This increased investment would likely lead to increased innovation- as companies strive to make their stock options more viable, and can result in a net job creation- within both the sector of algo-trading investors and the companies listed in the stocks.

1.6 Scope and limitations

The scope of this paper is focused on developing and testing trading algorithms within the stock market, utilising open source data from companies such as NASDAQ, Dow-Jones and APPLE to formulate trading strategies. The goal is to formulate and evaluate three different algorithms that generate policies for trading. The algorithms will be trained on historical data and their performance will be tested on unseen data from subsequent time periods. To accomplish these goals, this study will:

- Use open source stock market data to train and test algorithms.
- Focus on the return generated by a theoretical starting capital to evaluate the policies of the algorithms.
- Ensure that the policies generated by each algorithm are used only for the stocks they are trained on.
- Ensure there is sufficient usable data to be used for both training and testing phases of the algorithm.

It is important that the training data for the stock is up-to-date. This is to ensure that any new trends and traits pertaining to this data are relayed to the algorithm for effective learning. Cross application of policies between different stocks should be avoided. The policies learned by the model will reflect the information learned about the stock being traded and is likely to produce poor results when applied to another stock with different trends.

Since the generated algorithms are made to be robust and semi-autonomous, requiring minimal intervention and hyper-parameter tuning, there is little insight into these hyper-parameters in this paper. There may exist better algorithms to be used as actors in the stock market environment, such as Deep Deterministic Gradient Policy (DDGP) which will require hyper-parameter tuning, but the implementation of that algorithm will fall outside the scope of this paper.

It is also imperative that the data used when applying this algorithm is accurate and complete enough to ensure that the patterns of the stock market can be learned by the algorithm. As with all machine learning algorithms, the concept of 'garbage in, garbage out' applies. This means, that if the algorithm is given bad quality data, the results will be poor. This is based on the fact that the algorithm will learn the inaccurate traits of poor quality data, and reflect these inaccuracies in the results.

Using historical data in an attempt to make decisions on stock trading in the future means that there is an assumption made about the relationship between past information and future prices. This may not always prove to be true when considering market anomalies and unexpected economic events.

Chapter 2

Theoretical Framework

This section is dedicated to explaining the underlying statistics involved in creating the algorithms used throughout this paper. A base level of statistics may be required in order to fully understand the nuances of the techniques, but this should provide a useful overview of the methods and techniques used from beginning to end.

2.1 Reinforcement learning fundamentals

Reinforcement learning techniques operate on the principle of agents interacting with an environment to achieve a predefined goal through a reward system that evaluates their actions , (Sutton & Barto 2018). The environment encapsulates all relevant information about the system a reward signal, which evaluates the desirability of actions taken within it. The agent, conceptualized as a sequence of actions within this environment and receives feedback in the form of rewards or penalties based on its actions. Upon reaching specified termination conditions, such as task completion or excessive negative reward generation, the algorithm updates its strategies based on accumulated rewards. This process of iterative policy refinement based on feedback to optimise decision making strategies is central to the field.

Despite its apparent simplicity, reinforcement learning draws upon various statistical principles, notably Markov statistics, to guide the learning process. Through iterative processes, policies derived from experienced interactions with the environment, are refined. Reinforcement learning algorithms aim to optimize decision-making strategies. These optimized policies can then be applied to solve similar problems in new environments.

Whilst this introduction provides a basic framework for understanding reinforcement learning, the field encom-

passes diverse applications and continues to evolve with advancements in artificial intelligence and machine learning.

2.1.1 Markov chains

Markov chains are a branch of statistics that focus on discrete-time stochastic processes in a specific way. In a discrete-time stochastic process, the expected outcome at a certain time is determined by a function of the outcome of an experiment at a specified time. Mathematically:

$$r(t) = \phi(s,t) \quad t \in Z^{+} \tag{2.1}$$

Where $r(t)$ is a random variable at time t, and s is the outcome of the experiment. From equation 2.1, if $t = constant$ then the equation would result in a singular random variable at time $t = constant$. If, conversely, $s = constant$, the resulting output would be a sequence of time-dependant random variables over times $\{t = 0, t-1, t-2...t-t\}$. This sequence of random, time dependant variables is the result of a stochastic process.

In a standard stochastic process, the outcome at a specific time is considered as a probability. Given the state space $\{s_t, t = 0, 1, ..\}$, the probability that the state at time $t+1$ is $s_{t+1} = j$ depends on the probabilities at previous time steps. This can be visualized as:

$$P(s_{t+1} = j) = P(s_{t+1} = j | s_t = i, s_{t-1} = i_{t-1}, ...s_0 = i_0) \tag{2.2}$$

Markovian statistics enforces the principle that the probability of moving to the next state depends on the previous states only through the current state , Ross (2014). Therefore, equation 2.2 can be rewritten as:

$$P(s_{t+1} = j) = P(s_{t+1} = j | s_t = i) = P_{ij} \tag{2.3}$$

Thus, there is a simplification in a sense of equation 2.1. The state space of a Markov process is then the collection of all possible states the process can occupy. P_{ij}, above is used to notate the *transition probability* of the system being in state i at time t, and state j at time t+1.

Considering the state space of the Markov chain, and the notation used above we can say that at time t, the

system can evolve from state $i \in \{s_1, s_2, ...s_N\}$ to another state $j \in \{s_1, s_2, ...s_N\}$ at time $t+1$. The transition probabilities of these states can be collected in a stochastic matrix as follows:

$$P = \begin{bmatrix} P_{11} & P_{12} & ... & P_{1N} \\ P_{21} & P_{22} & ... & P_{2N} \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ P_{N1} & P_{N2} & ... & P_{NN} \end{bmatrix} \tag{2.4}$$

This matrix, being made up of probabilities, has the constraint:

$$\sum_j P_{i,j} = 1 \tag{2.5}$$

The constraint comes from the fact that the cumulative probability of transitioning from one state to another must be one. Considering, now, the stock market. If we were to consider the closing prices of a stock, s_t at day t, and assume that the rate of return are independent and ideally distributed (i.i.d) and normal (Gaussian).

$$s_{t+1} = s_t(1+r_t) \quad r_t \sim \mathcal{N}(\mu, \sigma^2) \tag{2.6}$$

In this way, stock prices are an example of a Markov chain with a continuous state space, because: s_{t+1} depends on all of the previous stock prices through only s_t,and can adopt any positive real number. Considering the state space of a stock market rather to be $s_t \in \{Bull, Bear, Stagnant\}$. This can be considered as a transition probability matrix as in 3.4 as follows:

$$P = \begin{bmatrix} P_{Bear,Bear} & P_{Bear,Stagnant} & P_{Bear,Bull} \\ P_{Stagnant,Bear} & P_{Stagnant,Stagnant} & P_{Stagnant,Bull} \\ P_{Bull,Bear} & P_{Bull,Stagnant} & P_{Bull,Bull} \end{bmatrix} \tag{2.7}$$

Knowing the state of the system at time t, Markov chains can be analysed through the framework of matrix theory in order to make predictions about the state at time $t' > t$. Considering the following:

$$P_{ij}^{(n)} := P(s_{t+n} = j | s_t = i) \tag{2.8}$$

In the above equation, P_{ij}^{n} is the probability that at time t, the state is i and at time $t+n$, the state is j. Using the Chapman Kolmogorov equation as a foundation as well as some mathematical induction, one can demonstrate that the transition matrix:

$$P^{(n)} = \begin{bmatrix} P_{11}^{(n)} & P_{12}^{(n)} & \dots & P_{1N}^{(n)} \\ P_{21}^{(n)} & P_{22}^{(n)} & \dots & P_{2N}^{(n)} \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ P_{N1}^{(n)} & P_{N2}^{(n)} & \dots & P_{NN}^{(n)} \end{bmatrix} \tag{2.9}$$

Takes the form:

$$P_{ij}^{(n)} = (P * P * P * P...)_{ij} = (P^n)_{ij} \tag{2.10}$$

Thus, the transition matrix after n-steps, as notated in equation 2.9, is equal to the n-power of the original one step transition matrix 2.4.

2.1.2 Markov decision processes

Markov chains, although useful and powerful in their own right, are not enough to try and capture the true nature of the market. They do, however, provide much needed understanding for some of the models that underpin reinforcement learning. This is because many reinforcement learning algorithm are modeled as *Markov decision processes* MDPs. MDPs are an extension of Markov chains that incorporate actions and rewards, allowing for the modelling of decision-making in uncertain environments , Puterman (1994).

Considering the Markov chain with its transition matrix displayed in equation 2.7. The state space is: $\mathscr{S} = \{Bull, Stagnant, Bear\}$. In order to extend this Markov chain into an MDP, one would have to give the agent the ability to make a 'choice' at each given state. Thus an action set can be defined for this model as being $\mathscr{A} = \{Buy, Hold, Sell\}$. Following this, a reward function will have to be defined. These functions will be

dependant on both the state, and the action taken in that state. One should not only consider the reward of applying the action at the state, but should consider the utility of the state to incur a more complete image. Thus two reward functions for the process should be considered.

$$g(s_t, a_t) : \text{reward of applying action } a_t \text{ in state } s_t \tag{2.11}$$

$$G(s_T) : \text{utility of being in state s at time T} \tag{2.12}$$

Since the goal is to maximise the expected rewards,

$$\max E[\sum_{t=0}^{T-1} g(s_t, a_t) + G(s_t)|s_0 = s] \tag{2.13}$$

The introduction of the action set and a set of reward functions has evolved the problem into an MDP, with the overarching goal of finding the set:

$$\vec{a} = \{a_0(s_0), ...a_{T-1}(s_{T-1})\} \tag{2.14}$$

That maximises the total reward given by:

$$J_0^*(s) = \max E[\sum_{t=0}^{T-1} g(s_t, a_t) + G(s_T)|S_0 = s] \tag{2.15}$$

The solution to this can be found using dynamic programming techniques.

2.1.3 Dynamic programming

Dynamic programming can be described as breaking down a problem into a set of smaller, more easily solvable problems, storing the solutions and rebuilding the total optimal solution from the modular sub-solutions , Bellman (1957). The usefulness of this solution comes from the *Bellman Principle of Optimality*. 'An optimal policy has the property that, whatever the initial state and initial decision are, the remaining decisions must

constitute an optimal policy with regard to the state resulting from the initial decision' , Thomas et al. (2019)

2.1.4 Reinforcement Learning

Reinforcement learning builds on the concepts of MDPs and dynamic programming, where agents learn optimal policies through interaction with the environment , Kaelbling et al. (1996). As reinforcement learning is said to be an extension of the techniques explained above, so too should the following sections be considered in conjunction with the previous sections.

RL environment

The environment of an RL algorithm is used to define a system with which an agent interacts in order to find an optimal policy for a specific problem. The environment contains information related to, but not limited to:

1. **State space:** Analogous to the state space described in MDPs, the state space is described by the set of all possible configurations the environment can be in.

2. **Action space:** This set describes all of the possible actions the agent can take within the environment in a given state, it is akin to the set of actions available in each state of an MDP.

3. **Reward function:** This provides feedback about the utility of the actions taken by the agent. This function maps state-action pairs to designated scalar rewards, indicating the immediate cost or benefit of performing a particular action in a specific state. This is just as in an MDP, where rewards guide the agent's decision-making process.

For the task of algorithmic trading at the individual level, the working assumption is that the actions of the agent do not affect the state of the environment. Thus, further environment dynamics are not required, and will not be described. But it can be noted that one can create a dynamic environment with which an agent interacts.

RL agent

The agent in the paradigm of reinforcement learning is the theoretical 'entity' that learns to navigate the environment by learning what decisions to make and which actions to take at specific states in the environment- in order to achieve a specified objective. The agent should contain and or decipher information pertaining to:

1. **Policy:** This refers to the set of rules that govern the agent's actions in different states of the environment. This maps states to actions and stores each state-action pair to be used for determining the optimal policy. It aims to optimise long-term rewards, similar to the optimal policy sought in MDPs.

2. **Value functions:** These functions help estimate the utility of remaining in a state, versus performing an action in a state. Much like the value functions used to evaluate policies in MDPs.

3. **Learning algorithm:** This algorithm is a method used by the agent to update its policy based on prior experience with the environment. Again, akin to learning algorithms such as Q-learning or value iteration used in MDPs.

4. **Exploration probability:** This random variable is used to determine when the agent should explore new actions, versus leveraging the knowledge it has amassed from previous policies.

5. **Reward signal:** This is the feedback provided to the agent about the effects of its actions. It guides the learning process by indicating the immediate desirability of the actions taken.

6. **Policy evaluation and improvement:** Once an iteration is completed, the policy learned by the agent is used to estimate the value function, the result of that iteration is compared to that of other iterations and, depending on the results, the policy of the system is updated.

It should be noted that agents are not limited to the above characteristics.

2.2 Supervised learning fundamentals

Supervised learning is another of the three sectors of Machine learning, along with unsupervised and reinforcement. Supervised learning invloves training models on labelled data to learn underlying patterns and make predictions , Bishop (2006).

This section is called Supervised learning fundamentals, but since the algorithms in this paper primarily make use of an MLP's, the primary focus of this section will be on Artificial Neural Networks (NN).

2.2.1 Artificial Neural Networks

NN's were conceptualised as mathematical analogues of a neuron (brain-cell). The idea was conceived in the early 1940's by Warren McColloch and Walter Pitts in their paper entitled, "A Logical Calculus of Ideas Immanent in Nervous Activity" , McCulloch & Pitts (1943). But it wasn't until 1958, when Frank Rosenblatt released his work entitled "The perceptron: A Probabilistic Model for Information Storage and Organization in the Brain" , Rosenblatt (1958), that there was an actual working model.

NN's have come a long way since Rosenblatt's one layered perception, with multi-layered perception's having the ability to learn non linear and complex relationships with relative ease. To understand the concept fully, it may be beneficial to focus on the concept of a single layered perceptron at first.

2.2.2 Single layered perceptron forward pass

When referring to the layers of a perceptron model, one typically excludes the input and output layers. The 'single layer' typically denotes what is also known as the 'hidden layer' of the model.

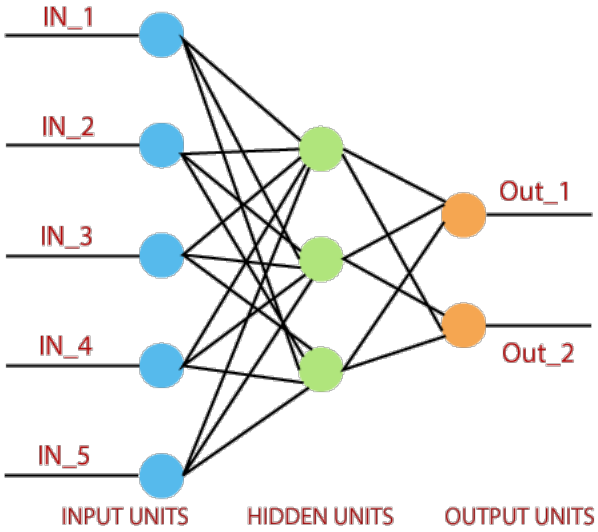

Figure 2.1: Single layer perceptorn model

From 2.1 one can visualise the concept of a single layer perceptron. To understand how it works, one should consider the image in detail. The first layer in the perceptron model is the layer of blue circles. These circles denote the input data. This data can be made up of any *numerical* data. Of course the data should be properly prepared and cleaned, but we will assume this is the case. Following the input of the data, the data is 'transferred' to the second hidden layer. This can be visualised by the lines that connect the two layers. These lines are known as the 'weights' of the model. Each of the weights represent a number, by which the input data is multiplied.

The second layer of the model is denoted by the green circles. This layer is the hidden layer of the model, so called, because the numbers that are attached to them are not directly perceived, but instead are used by the model to produce a result. Each of the different green circles represent something called a 'bias'. The bias is a number that is added to sum the inputs of the hidden layer(The product of the data and weights).

The result of each of these summations with the added biases will then be 'passed' to the output (orange circles) through an activation function (second set of lines). This activation function will generally serve to produce a non-linear output. This output represents the 'confidence' that the neuron belongs to a specific class. In the case of 2.1 there are only two outputs. In this case, one could consider using a decision boundary condition

of 0.5. If the confidence is greater than 0.5, then the output can be classified as 'Out_1'. Otherwise, it will be classified as 'Out_2'. The activation function used to generate this 'confidence' varies and depends on the particular problem at hand.

Once all of the outputs of the hidden layers have been calculated, the class with the larger number of votes will be used to classify the data. This process is known as the forward pass of the algorithm.

2.2.3 Single layered perceptron back pass

Once a classification on the data has been made, the label associated to the data is compared to the output that was generated by the algorithm. The similarity of the output from the expected value is measured using a loss function. This is what separates supervised learning from unsupervised. If the classification matched the label, the weight and the biases of the model remain the same, and the model is ready to be used to classify on unseen data.

Sub-optimal approach

If the classification does not match the label of the data, each of the weights associated to the 'connection' between the input and the hidden layer will be randomly updated. The same update will occur for the biases associated with the hidden layer. The process of the forward pass will then be repeated to generate new classifications, which will then be compared to the labels again, and condition for updating the model will be considered again. This will repeat until the model makes predictions that relate to the correct label of the data.

Optimal approach

For a more optimised approach to updating the weights and biases of the system, one can use an optimisation algorithm to speed up the rate of convergence. Algorithms such as gradient descent, or more complex momentum algorithms such as ADAM (Adaptive Momentum) are examples of optimisation algorithms used in the back pass.

Both of the algorithms work by computing the gradient of the loss function, with respect to the weights of the network. This is done by back propagating the error though the layers of the network and calculating the gradient. With multiple layers, this is done using the chain rule in calculus.

The optimisation algorithm used for this process (ADAM) will not be fully explained, but can be conceptualised by considering the result of rolling a ball down a hill. As the steepness of the hill increases, so too does the velocity of the ball. As the ball reaches the bottom of the hill, the velocity will start to reduce and it will start coming to a halt. Eventually the ball will settle at the bottom of the hill, this will occur at the lowest point

of the hill.

Analogous to the hill in the above concept, the loss function's gradient is used, the momentum of a theoretical ball is then used with Newtonian physics equations to navigate this function's gradient in order to find the global minima.

2.2.4 Multi-layered perceptron conceptualised

Now that the concept of a single layered perceptron has been introduced, it should be noted that the model does not come without limitations. Due to the structure of the model, the single layered perceptron is only effective at classifying linearly separable data. This limitation can be rectified, however by the introduction of more hidden layers.

Multi-layered perceptrons (MLPs) extend single layered perceptrons by adding hidden layers, enabling the modeling of complet, non linear relationships , Rumelhart et al. (1986) the perceptron can handle the separation of non-linear data. This makes the MLP a very powerful tool for tasks such as regression and classification. It also means that the algorithm is a good tool for pattern identification.

The ability for the MLP to recognise patterns makes it a useful tool to use as an agent in a reinforcement learning algorithm. This can take the place of more rudimentary agents such as Q-learning. Not only will it likely provide a more useful result, but it will speed up the processing time of the algorithm.

2.3 Financial technical indicators

Financial technical indicators such as the Exponential Weighted Moving Average (EWMA) and Moving Average Convergence Divergence (MACD) are used to analyse market trends and predict stock prices , Murphy (1999). They typically work by comparing the prices of the stocks along different periods and try to predict the trend of the stocks in the present periods. Although one may be able to visualise the growth and decay of stocks, there exist trends that may not be innately visible by the human eye.

Technical indicators attempt to highlight these invisible traits, to allow for a more informed decision to be made about the stocks. These functions can serve to lower the risk of stock trading- allowing for modifications to be made on the risks taken on the stocks by the user. This can individualise the algorithm and allow the user to parameterise their risk margins.

2.3.1 Efficient Market hypothesis

In order for these indicators to be efficient, one should consider the efficient market hypotheses:

- **Weak form**- All past information is contained in securities. No pattern exists, prices are totally random, no fundamental or technical analyses are useful.

- **Semi-strong form**- Price quickly soaks up new information, so analysis is useful to determine the long - term trend of prices.

- **Strong form**- The information set that determine price trend can be evaluated with past news about data.

In order for these functions to serve their purpose properly, they should be used under the assumption that the stocks they are being used on do not satisfy the weak condition of the hypothesis.

2.3.2 Types of financial indicators

Since the market is not solely governed by rational means, there exist a number of different types of indicators, each of which try to discern different kinds of information about the state of the market.

- **Sentimental indicators**-They deal with the feeling of professional investors about current situation of financial markets

- **Flow of funds indicators**-Macroeconomic metrics used by financial institutions to have an overall view about tracking flow of national economy.

- **Market indicators**- Quantitative information useful to interpret and forecast financial data. They are divided in:

  - Chart Patterns: visual analysis of data by both a quantitative and qualitative approach

  - Price trends: mathematical manipulation of time series data to understand price oscillations.

  - Volume and momentum indicators: identify the strength of the market measuring the variation of prices.

For this project, the focus will be more directed on using market indicators as the technical indicators.

2.3.3 Technical indicator functions

Exponential Weighted Moving Average (EWMA). This function assigns each price a time dependant weight. The weight associated to the price reduces exponentially over time. This ensures that the more current prices play a more important role in determining the sentiment of the market. The EWMA is calculated as follows:

$$EWMA_t = (1-\alpha)EWMA_{t-1} + \alpha P_t \tag{2.16}$$

Where,

$$\alpha = \frac{2}{k+1} \tag{2.17}$$

α is the weight of the most current price, and k is the length of the chosen window.

EWMA is a good algorithm to determine trend direction in the market, but it may be further leveraged to arrive at what many would consider to be an even more useful function. The **Moving Average Convergence Divergence (MACD)** is represented by 3 distinct interconnected indicators, whose relative position determines the trading indication. By considering the EWMA with a window (k) of 12, 26, and 9 one can generate both an MACD line and a signal line in the following way:

$$MACD\ line = EWMA(k=12) - EWMA(k=26) \tag{2.18}$$

$$Signal\ line = EWMA(k=9) \tag{2.19}$$

This is an example of a momentum indicator. It allows traders to understand when the market is over-bought, and over-sold by considering the position of the signal line relative to the MACD line. Since the signal line considers the short term return of the EWMA function, while the MACD line represents the difference between the EWMA(k=12) in the short term, and the EWMA(k=26) in the long term. If the MACD line crosses above the signal line, there is an indication that the momentum of the asset is increasing. This crossover suggests that the short-term moving average is rising faster than the long-term moving average, signaling a potential

uptrend and generating a bullish signal for traders. The opposite is true when the MACD line dips below the signal line.

2.4 Integration of Techniques

Throughout this section, we have considered the origins of a number of powerful techniques in the area of data science. If one were to try and apply these techniques to a specific task, a number of specifications should be met. Following the explanation of a reinforcement learning algorithm, one should consider how to define an environment. The environment should consist of data that follows the Markov principles so as to ensure the algorithm can be used.

As previously stipulated, the financial market fits this characteristic. Thus if the data pertaining to the prices of stocks were used as the environment of the algorithm, one could consider the prices of the stocks at specific times to be the set defining the continuous state space of the algorithm. In order to transform this Markov chain into an MDP, the process needs both an action set and a reward function of sorts.

The action set attached to this chain will be the set containing the actions of buying, selling and holding. These actions can be completed in any state and the result will be stored as a state-action pair. These pairs will then be passed into a reward function that will generate an immediate reward for the system.

Considering the financial technical indicators described in section 3.3.3, one can determine the utility of performing an action in a given state. Thus the reward pertaining to the state-action pair can be described by using these indicators as the environment's reward function. This will generate an immediate reward scheme for each action taken at each state and will provide the agent with more information on the system.

Once the environment has been defined, the definition of an agent is necessary to traverse the environment. Since the environment is large and continuous, the use of functions such as q-learning or value iteration may impede the processing time and limit the use of the algorithm. This can result in slow convergence of the algorithm with high variance. To compensate for this, a variation of the MLP from chapter 2.2.4 is used as the agent.

The MLP is a good choice as it can handle high-dimensional state spaces, and is a powerful tool for learning complex patterns, as previously explained. This algorithm, however, does come with potential risks. Since it is a more complex system, the tuning of hyper-parameters is vital and thus may require more attention in design. This should be done to ensure the algorithm does not over or under fit the provided problem. The amount of data required for the algorithm to return viable results is also higher.

The input of the MLP should consist of the historical data, the current state, and the reward functions of the environment. The output of the MLP will be will be the policy of the algorithm. This output can be interpreted as being the probabilities of taking different actions given the current state. Therefore, an output layer of size 3 should be used for this problem (buy,sell,hold).

The value function of the MLP is described by net worth of the individual at the time of the action. This is a combination of the seller's current bank balance in addition to the amount of money the seller currently holds in the stock. Thus the reward will be considered as the direct result of the action taken by the function in a given state.

The reward signal of the algorithm will be based on the profitability of the chosen actions in a given state. Policy optimisation will be done using the back pass of the algorithm. The result will be a neural network trained on subset of historical data with a policy that should prove useful for future trading on the same stock market.

2.5 Application on financial markets

Provided the algorithm is properly set up and used, the use case in the financial markets can serve to democratise the markets. This can allow for individuals with little prior knowledge on how to properly navigate the stock markets to freely enter into stock trading. The freedom to enter the markets for individuals en masse can serve to empower the individual, allowing them to engage with the market with the knowledge that their money is being invested using some of the most powerful techniques available to them. The influx of investments within the markets can also serve to bolster the economic growth not only within the stock market, but in the companies that are listen on the markets.

The increased investment in the companies can lead to increased growth within the company. The results being an increased sense of urgency for innovation within the company. This could result in either horizontal, or vertical integration. This may result in new job opportunities emerging, increased competition and an increase in ingenuity.

Chapter 3

Literature review

3.1 The use of financial technical indicators

The role of financial technical indicators in the field of trading is considered to be highly important and its practice is well established. In the paper entitled; 'technical analysis: An asset allocation perspective on the use of moving averages' , Yingzi et al. (2009). In this paper, Yingzi et al. (2009) analyse the utility of using financial technical indicators in the paradigm of asset allocation. In their paper, they aimed to explore the potential benefits of using one of the more simple indicators- the Moving Average (MA) technical indicators- which happens to form the basis for the technical indicator used in this paper- Moving Average Convergence Divergence (MACD).

The authors explain that, although widely used in the field of trading, papers exploring the utility and efficacy of the techniques are few and far between. Papers that exist, such as those done by Eugene Fama and Marshal Blume , (Fama & Blume 1966) and Alfred Cowles , (Cowles 1933) do not yield any definitive conclusions about the utility of the functions. Further research into the use of technical indicators has started to illuminate the presence of utility in their use- as seen in the work done by Andrew Lo et al , Lo et al. (2000), but further exploration is required.

The authors of Yingzi et al. (2009) aim to provide additional information on the subject in a manner that was previously unexplored. By providing statistical proofs and concurrently exploring the theoretical rationales behind using MA in asset allocation, the authors aim to produce a more comprehensive study on this subject than their predecessors.

The authors critique the use of MA for asset allocation revolved around an all-or-nothing strategy. This strategy includes full monetary commitment, or complete avoidance to a stock option dependant on the MA signal. This- the authors explain, is a sub-optimal use of this function. Instead, they propose MAs as a model for investor risk aversion. This approach can lead to a less binary approach to stock allocation, which may prove useful in optimising returns.

The authors explain that they intend to derive functions that allow investors to use the MA in conjunction with other predictive and fixed rule functions in an optimal manner- through a number of methods. The combined metrics will be tested in order to gauge their utility. It is explained that the use of an 'optimal dynamic strategy' is likely to produce the best results in portfolio modelling, but it is acknowledged in the article that the derivation of truly optimal strategies can be extremely complex, and requires vast amounts of data. The simplicity and robustness of MA's is offered as an advantage over attempting to derive the true model of the portfolio.

Within the findings of this paper- the authors show that the use of an (estimated) optimised MA outperforms an incorrectly chosen optimal model. This further highlights the benefit of its utility due to its simplicity and thus comparative robustness.

Analytical testing derived important proposition in the paper:

1. Combining the optimal MA with a fixed strategy generally improves returns unless stock returns are entirely unpredictable.

2. MA combined with a fixed rule can create a new optimal function, which can serve to enhance the utility of the function when stock returns are predictable.

3. When considering a pure MA strategy, using an all-or-nothing strategy is unlikely to yield optimal results.

These propositions illustrate the importance of using MA in stock allocation (particularly when returns are predictable) and highlight some important information with regards to their proper usage in different situations. Proposition 3, above, highlights the author's advocacy that the MA should be considered as a function of the investor's risk aversion. This concept will be explored further in this paper when considering the implementation of the MACD technical indicator.

The further sections of Yingzhi et al. (2009) explore the optimisation of the risk aversion function γin different approximate

solutions to optimal trading strategies. They consider the power-utility case with first order and second order approximate solutions. In the former case, the optimal approximate utility function was said to be inversely proportional to the risk aversion function i.e:

$$U_1 \propto \frac{1}{\gamma} \tag{3.1}$$

Whilst in the latter case, the function linking the two values is far more convoluted and involves solving a complex differential equation in order to arrive at the solution. This paper will consider the first order approach. This will likely result in sub-optimal results, but it will help to derive a simple and practical use of the technical indicator function as a risk aversion measure.

The empirical section of Yingzi et al. (2009) derived evidence to support the theoretically backed claims mentioned above. Their experiments compared different functions across four different starting conditions. The starting conditions include; comparison under complete information, comparison under parameter uncertainty, comparison under model uncertainty, the effect of lag lengths. In each of the cases, the results of applying fixed rules, a combination of fixed rules with MA and optimal strategies- when available- were observed and recorded for comparison.

The results of these experiments served to bolster the author's claims with regard to combining the fixed rules with MA. The use of combined fixed rules with MA consistently outperformed the fixed rules alone. Whilst in the case of model uncertainty, the combined metric served to outperform the optimal solution in terms of robustness- as the optimal solution proved to be far more computationally expensive and difficult to derive.

The reviewed paper demonstrated that the use of technical indicators provide a significant impact on the return of investments in algorithmic trading. This is especially true when the user is not aware of the optimal strategy of the portfolio and when the stock is predictable. The use of MA and by extension, MACD described in section 2.3.3, proved to serve as a practical and robust risk aversion function. The emphasis being on using the results as a function in order to generate more optimal results. This underscores the importance of considering these technical indicators when generating a robust trading strategy.

3.2 Deep reinforcement learning in portfolio management

In the article entitled 'Deep reinforcement learning in portfolio management' , Liang et al. (2018), the authors explore the use of deep reinforcement learning algorithms in managing stock portfolios. Whilst the use of

reinforcement learning has been extensively studied in fields such as robotics and gaming, their application in financial markets are relatively unexplored. They explain that deep-reinforcement learning algorithms are renowned for being able to handle non-linear and complex domains without extensive feature engineering. By extension, this should make them compatible with the complex and dynamic domain of financial markets.

This paper explores two prominent deep reinforcement learning algorithms; Deep Deterministic Policy Gradient (DDPG) and Proximal Policy Optimisation (PPO). The two algorithms are used and compared to each other in order to define a more optimal policy in the use case of deep RL strategies in the stock markets. Both DDPG and PPO work by combining an RL environment with a multi-layer perceptron (MLP) as the actor, as discussed in section 2 of this paper. The key differences between thes algorithms are summarised in table 3.1.

Algorithm:	PPO	DDPG
Algorithm type:	Policy-based	Actor-critic
Action space:	Discrete	Continuous
Training stability:	Stable algorithm	May incur hyperparameter tuning for stability
Exploration v.s. Exploitation:	Balanced through optimization	May require explicit strategy
Policy update:	Updates policy during current roll-out	Uses replay buffer to store and sample from

Table 3.1: Differences between PPO and DDPG algorithms

Table 3.1 highlights the key differences between the two deep learning policies. The authors find evidence that the DDPG strategy proves more optimal in portfolio management due to the algorithm's continuous action space and potential for hyper-parameter fine tuning. However, the discrete action space of PPO makes integration with financial technical indicators more simplistic, which is why this paper focuses on PPO. The evidence that they accumulated shows that both algorithms show great potential in the fields of algorithmic trading. PPO is also known as a stable algorithm, which means that the parameters of the model are updated in a self contained manner- this is further reason for the exploration of this algorithm, as this may aid in the generation of a more autonomous algorithm.

In the paper, the authors explain that they considered the cost of obtaining a stock as a percentage of the price of the stock. They explained further that this transaction fee is vital in producing a more realistic representation of capital gain or loss in the trading environment. Liang, et al. aim at diversify stock options, using deep RL in order to choose which stock options to favour. They aim to achieve this using multiple risky assets and one risk-free asset. In contrast, this paper investigates the efficacy of applying a deep RL trading strategy whilst only considering one risky asset(the stock), and one risk-free asset(the account balance). The transaction, as shown in Liang et al.'s study, adds significant complexity to the reward signal generation. This paper recommends that the transaction cost be considered post algorithmic application. It can be considered as fixed value- where the user agrees to set aside a portion of their expected earnings before applying the algorithm, or it can be

derived by limiting the number of transactions in the environment to a set limit, paying the standard amount for these transactions.

The paper concludes with the notion that deep RL algorithms, in particular, PPO and DDPG, showed promise in capturing complex market movement patterns, with limited data and features, but require further attention in order to be fully adapted for use on the financial market. They conclude by stating that more modifications are needed for the algorithms- stating that the algorithm "tends to buy only one asset at a time". This limitation may possibly be overcome by using an ensemble of algorithms rather than applying one algorithm to multiple stocks. Considering the algorithms generated in this paper work only one one risky asset, by refining the action space (possibly with the inclusion of the technical indicators), one may generate a function that mitigates risk and assigns a set portion of the net work to the stock. This could potentially be used on more than one stock option at a time, resulting in an ensemble of algorithms for different assets rather than using one environment for all of the stocks.

The authors' of , Liang et al. (2018) provide valuable insight into the potential for deep RL use in portfolio management, but their findings suggest that the algorithms in question are better served for their original applications in robotics and gaming. The requirement for extensive parameter tuning and the tendency for the algorithm to invest in only one stock option at a time are among the author's primary concerns. Future studies in this sector could benefit from focusing primarily on the action state and reward state of the environment, enforcing conditions that incorporate risk management functions in order to enhance the ability for the algorithm to adapt properly to the financial markets.

3.3 Proximal Policy Optimisation Algorithms

Proximal Policy Optimisation (PPO) is an algorithm that aims to optimise the policy generated by applying an artificial neural network (nn) (the agent) to a reinforcement learning (RL) environment. It does so by optimising a surrogate function that serves to approximate the agent's reward, whilst enforcing constraints to ensure that the updated policy does not deviate too much from the previous policy. PPO is an open source algorithm developed by openai. According to , Schulman et al. (2017) the algorithm works by focusing on a 'surrogate' function representing the total reward of the policy and passing information about said function into the ann that was used to generate the policy to update the parameters of the ann.

In the article entitled,'Proximal policy optimization algorithms.' , Schulman et al. (2017), it is explained that there exist a number of ways to consider optimising the policy generated by a neural network on an RL environment. In order to explain them, the process of the policy generation will first be explained.

1. **State input:** The state vector containing the state of the environment at the current step is passed into the nn.
2. **Action generation:** The output of the nn relates to the generated action to take in the environment by the nn.
3. **Action execution and reward generation:** The resultant action is then taken by the environment, with the environment providing feedback in the form of a reward signal.
4. **State Transition:** The state of the environment is then updated.
5. **Policy update and backpass:** Dependant on the optimisation method, the policy will there after be updated and a back pass will be used in order to fine tune the parameters of the nn.
6. **Iteration:** Steps 1-5 are repeated in iterations, constantly updating the environment's policy. This is done until some stopping criteria is met.

Table 3.2: Generalised explanation of nn policy update

Schulman et al. discuss two different methods for updating the policy in a reinforcement learning environment. Gradient policy methods and trust region policy methods are explained in the article. In gradient policy methods, as shown in table 3.2 at step 5: an nn (the actor) will generate a batch of experiences from the environment. From batch of experiences, the performance of the policy will be evaluated using a performance measure, such as cumulative reward. Following this, the gradient of this performance measure with respect to the policy parameters (i.e. the weights and biases of the nn) is calculated. This gradient provides insight into how the performance measure will react to the changes made to the policy parameters. Using this information, the nn updates the policy parameters in order to maximise the reward of the system. In this way, the policy is updated and refined in a manner that considers the agent's behaviour and its impact on the reward function.

In contrast, Trust Region's methods serve as the policy optimiser in this deep learning environment. Considering step 5 of table 3.2 and focusing on Trust Region Policy Optimisation (TRPO), the algorithm aims to construct an objective function, based on experiences collected during the interactions with the environment. In TRPO, the objective function can be considered as a surrogate function that acts like the true reward function(cumulative reward) but is constructed to be easier to optimise, whilst still providing a good approximation of the true objective function. This function serves to guide the process by which the policy is updated. TRPO introduces a constraint on the magnitude by which the policy may change during updates.This constraint is typically measured by considering the amount of divergence that occurs before and after the update of the policy, this is usually measured using the Kullback-Leibler(KL) divergence. This promotes stability in training by ensuring that the updated policy does not deviate by too large a value from the previous policy (preventing large policy changes). Once the function and the constraint are defined, the algorithm updates the policy parameters based on the gradient of the surrogate function's objective, with respect to the policy parameters. As above, the gradient will direct the policy changes to follow the highest change with respect to small changes in the policy parameters. These changes will then be scaled according to the parameters contained in the constraint. These scaled changes optimise the overall policy whilst maintaining stability. Further, , Schulman et al. (2017) refers to a 'clipped surrogate objective'.

TRPO with a clipped surrogate objective works similarly to the original TRPO method. The clipped TRPO method aims to further stabilise the learning of the function by comparing the probabilities of the algorithm performing specific actions in the current policy to that of the subsequent policy. The algorithm is penalised when this ratio deviates from 1. The clipping method prevents the algorithm from making significant changes to the policy and promotes even more stability in the training process. Additionally, the paper suggests that, without considering the clipped surrogate objective, one can consider an adaptive KL model in order to stabilise the training process.

The TRPO with a clipped surrogate function is what is known as the Proximity Policy Optimisation or PPO. Throughout the paper, the authors of , Schulman et al. (2017) continuously showed the advantage of using PPO over other, well defined, policy update methods. The benefits include, the relative stability of the algorithm, ease of implementation and increased optimality in a number of different worked cases.

3.4 Considerations from literature review:

Based on the literature reviews completed above, there are a number of factors that should be considered when considering the creation of a deep RL trading algorithm.

Category	**Key Considerations**
Financial Technical Indicators	- MAs as a risk model (Yingzi et al. (2009)) - Simple, robust MAs often outperform complex models
Deep RL in Portfolio Management	- PPO and DDPG show promise (, Liang et al. (2018)) - PPO preferred for stability, easy integration
Policy Update Methods	- Gradient methods use performance gradients (, Schulman et al. (2017)) - TRPO uses surrogate objectives for stability
Proximal Policy Optimisation	- PPO limits policy deviation for stability - Clipped surrogate objectives enhance training stability
Future Research Directions	- Explore ensemble methods for multiple assets - Improve DRL algorithms for risk management
Implications	- Indicators and DRL are valuable in finance - Stability, implementation ease, and indicator integration are key

Table 3.3: Key Considerations from the Literature Review

Chapter 4

Materials and Methods

4.1 Implementation Details

In order to implement the algorithms used throughout this paper, one should be at least somewhat familiar with the Python coding language. The choice of editor is up to the user. Our choice to use visual studio code is based on its simple integration with github, but other editors such as spider, jupyter or even using the terminal will work. Python was chosen as the primary coding language for the project due to its vast collection of powerful libraries. These libraries empower the user to implement complex functions and algorithms in a more concise and efficient manner, assisting the programmer in developing more intricate and stable algorithms in a more timeous manner. Although Python is not recognised to be the fastest coding language, its comparatively simplistic syntax and versatility make it a compelling choice for the tasks presented. The algorithm can be adapted to be used for languages such as Java, or C if the user is so inclined, but for this study, that conversion will not be considered.

The libraries used throughout this project can be found below. Each of the libraries can be downloaded through pip, and all of them work on the current version of the Python 3 interpreter (3.10.11).

- **numpy v.1.26.1**- This is a library for handling data in the form of arrays. It is useful for manipulating data, and is embedded within a number of other packages.

- **pandas v.2.1.2**- This library works as an extention of numpy and allows the user to easily and efficiently manipulate DataFrames and series data.

- **yfinance v.0.2.33**- This library contains functions that calculate the technical indicators referred to in section 3.3.3 above.

- **sklearn v.1.3.2**- This package contains a number of machine learning and data cleaning algorithms. In this code, the functions used are MinMaxScalar- which is used to scale the data, and train_test_split which is used to split the data into a training and a test set.

- **stable_baseline v.2.2.1**- This package contains the MLP algorithm that is used to traverse the reinforcement learning environment, as well as a DummyVecEnv which created a dummy vector, so that the data used in the MLP is of the correct type.

- **gym v.0.26.2**- This package contains a number of reinforcement learning environments to choose from when 'init'iating the action space and observation spaces.

- **matplotlib.pyplot v.3.8.0**- This is a package that is used to visualise the data. This library also works well with numpy and pandas.

- **math**- This package allows the user to use a number of mathematical functions.

Along with the packages described above, the framework of the project is based around the work done by Trading Tech AI , AI (2023) who describe how to use a simple algo-trading strategy with reinforcement learning and an MLP.

The three algorithms will henceforth be allocated the names:

- **Pure Reinforcement Rearning algorithm**- This algorithm will use deep reinforcement learning, with no technical indicators.

- **Pure Technical Indicator Algorithm**- This algorithm will combine the all-or-nothing technical indicator algorithm with the deep reinforcement learning algorithm.

- **Strategic Technical Indicator Algorithm**- This algorithm will use a threshold system to bin the values of the technical indicators, and enforce rules about how much of the stock can be bought or sold.

The use of PPO as an agent in the algorithms was chosen to reduce the need for hyper-parameter tuning. The algorithm is also considered stable and robust. This will afford the user to apply the same algorithms to

different stock options with little to no parameter tuning. As described in the introduction of this paper, this serves to generate a semi autonomous, stable and robust algorithm with a simplistic implementation that can help to democratise the stock market.

4.2 Data Sources and Preprocessing

The data used for this project was collected using the package yfinance. yfinance is an open source tool that allows the user to collect stock data directly from Yahoo. Data may be collected from other websites using API's or other means, but yfinance provides a simple and effective way to collect stock data for free and without having to generate an API key. The collected data should contain the following information about the stock:

- **Date**: The time stamp.
- **Open**: The price of the stock at the opening of the market.
- **High**: The highest price the stock reached during this time period.
- **Low**: The lowest price the stock fell to during this time period.
- **Close**: The price of the stock at the end of the time stamp.
- **Adj Close**: The price of the stock adjusted to reflect corporate actions.
- **Volume**: The number of stocks sold over the time stamp period.

Once the data has been collected, it should be inspected to insure the quality of data is high. The number of missing values in the data should be considered. If the data is missing too many values, it should be considered too volatile to use and another stock option may be chosen. If however, there are few missing values,less than 1% of the data, these values can be removed from the dataset, or they may be imputed. The stock data pertaining to apple, between the times 1994-07-17 and 2023-10-31, contained zero missing values, and no duplicated values. This implied that this data is of good quality and there is no need to impute or remove rows of data. The data should subsequently be considered for feature selection and engineering.

Since this paper is considering the effects of technical indicators on the algorithms, the first additional feature that should be added to this data set should be the MACD value of the stock. This computed using pandas built

in 'EWMA' function, that generates the Exponential Weighted Moving Average of the stock over a specified span. The relevant MACD values are then generated using the EWMA following the methodology described in chapter 2.3.3 of this paper. The result of these values can be plotted as follows:

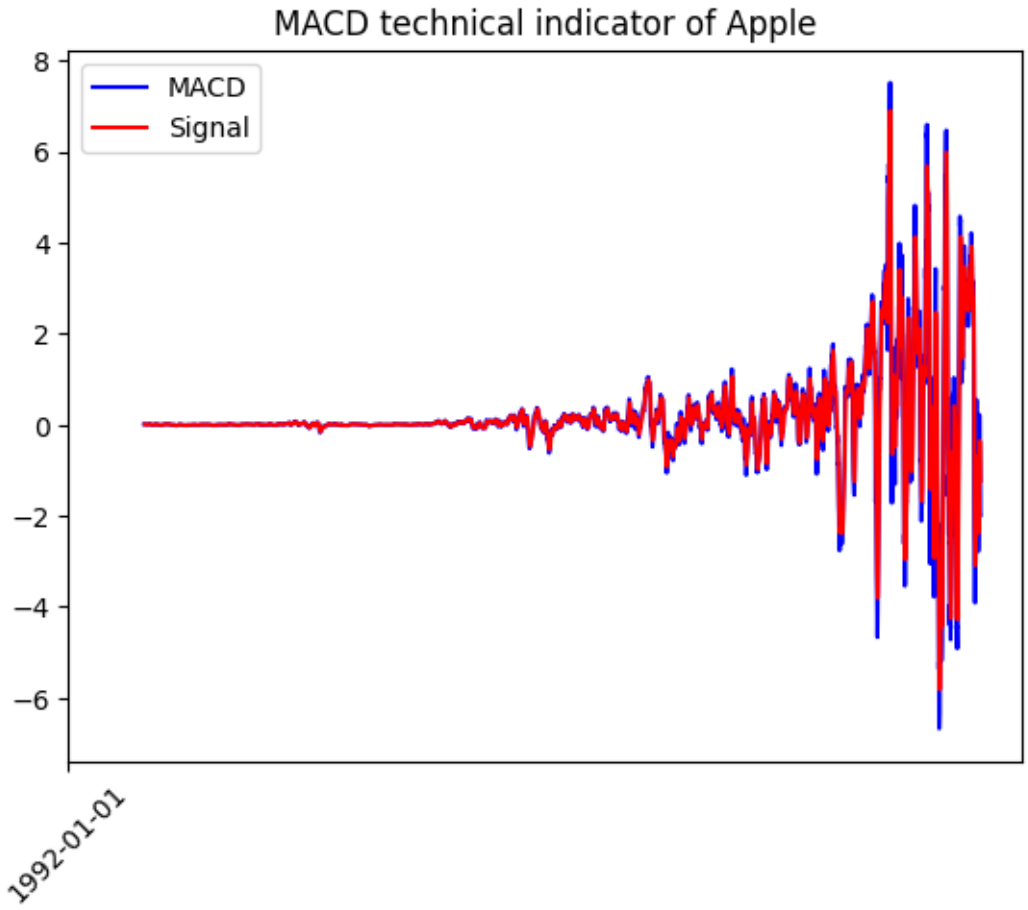

Figure 4.1: MACD for Apple dataset

Figure 4.1 illustrates the MACD calculations applied to the Apple weekly time series dataset. As evidenced by the graph, the signal line and the MACD line often overlap each other. When the MACD line is greater than the signal line- the blue line is the leader in the image, that implies that the MACD is greater than the signal line, indicating a bullish momentum in the stock. This is derived from the fact that the MACD line is the difference between the short term and long term EWMA. If this difference is greater than 0, it implies that the short term stocks are weighted more than the long term, implying an increase in the stock prices in the shorter term- and hence a Bullish momentum. The opposite indicates a bearish momentum in the stock.

The data pertaining to the MACD values of the stock should be appended to the dataframe that contains the information of the stock. This will be used in the Pure Technical Indicator algorithm. For the strategic technical indicator algorithm, the bins of the MACD values should be derived. The pseudo code describing the method for deriving these bins can be seen in Algorithm 1 (18):

This will generate 4 bins that aim to capture some of the information about the magnitude of the MACD

Algorithm 1 MACD Value Binning

Input: Column containing MACD values
Split the column into positive list P and negative list N
Determine the mean of P: μ_P
Determine the mean of N: μ_N
for each value n in N **do**
    if $n < \mu_N$ **then**
        Allocate n to bin 0
    else
        Allocate n to bin 1
    end if
end for
for each value p in P **do**
    if $p < \mu_P$ **then**
        Allocate p to bin 2
    else
        Allocate p to bin 3
    end if
end for

values. These can be used to add the additional constraint to the strategic technical indicator method. The binned MACD data should also be appended to the dataset.

The dataset contains values that relate to the volume and the adjusted close of the stock. The volume is a useful indicator that might provide insight into the decisions taken by big players in the stock market. With high volume indicating high sales- which could indicate the decisions taken by some of the hedge funds and major market leaders. Since the goal of this paper primarily revolves around the use of technical indicators, this column will be removed from the dataset. Additionally, the columns of open,high,low and adjusted close may prove to introduce noise into the environment and they will also be removed from the dataset.

After all of the feature engineering has been completed, the columns of the dataframe should contain only the information pertaining to the close, MACD and MACD bins. This will be the same information used for each of the algorithms. Scaling the data, particularly the price data make the action of buying stocks more complicated, as the price would have been adjusted to the mean of the price set. This will make calculating the number of available stock and thus the size of transaction harder to obtain. Thus, there will be no scaling of the data.

The final step in this process is to split the data into a training and a test set. This allows the algorithm to learn patterns and policies on historical data (training set). The policy that is found on the training set can then be applied to the new, unseen data (test set). The training-test split will be comprised of a 70-30 split. The data falls under the category of time series data. Thus, the split should not disrupt the order of the data and there is no reason to stratify the data. The training set will then be the first 70% of the data, and the test set will be

comprised of the remaining 30%.

MACD requires a large amount of data for sufficient application. If the user intends to invest in a stock option that is relatively new, or for which there is minimal data available, the use of MACD as the financial technical indicator is not advised.

4.3 Algorithm Implementation

The general structure of the algorithm was taken and modified from the article created by Trading AI , AI (2023). The environment is set up as a class, with a number of functions within it. The functions included are:

Function:	**Role**
init	This function is responsible for 'init'ialising the environment. It includes information about the data being passed into the class, the size of the action space and the size of the observation space.
reset	This function is used to (re)-set the values within the class when the environment is 'init'ialised. It includes information about the starting account balance, the current step and a number of other variables.
next observation	This function is used for returning the value of the data at the current step.
step	This action updates the step value, informing the environment about where it is in the data.
take action	This function decides the action for the algorithm to take- buy,sell or hold- at the current step.
get reward	This function generates the reward signal of the environment.

Table 4.1: Environment design

Each of the three algorithms used in this paper have their environment set up in the manner described in table 4.1. They are each given the same information and the same 'init'ial starting parameters. The most important starting parameters for each of the algorithms are as follows:

Starting parameter	**Value**:
Account balance	100000
Current step	0
Shares held	0
Net worth	Account balance
Max net worth	Net worth
Reward	0

Table 4.2: Starting values in the environment

As the environment moves from the current step to the next, it first observes what action is being taken. This is where the 'take action' function is called. This function will make the decision for what action the actor will make at this step. This action will be decided randomly using the gym environment that was used to generate the state space within the 'init' function, but the function will ensure that the randomly chosen action is viable.

If the account balance does not hold enough to buy the stock, it will not be able to. If the user holds no stocks, they cannot sell any stocks. Once the action has been taken, the step will be updated and the reward function will be called. The reward function will be updated to include information about the difference between the user's current net worth, which is calculated to be a combination of their current account balance and their current wealth in stock options, and their maximum net worth. If the difference is positive, the net worth has increased and this will relay positive growth in the reward function, the opposite is true if the signal is negative. The reward signal is also set up to include information about the algorithms previous action. If the actor buys a stock at a price that is lower than when they last sold a stock, a positive reward is generated. If the actor sells the stock at a higher price than when they last bought the stock, a positive reward is generated. A negative reward is generated for the opposite actions.

The environment for each of the three algorithms differ in the 'take action' function. This function helps the algorithm make its decisions on whether to buy, sell or hold a stock. In the case of buying a stock, the constraints pertaining to the action in each of the environments are described in the table below.

Pure reinforcement:	**Pure Technical Indicator**	**Strategic technical indicator**
The algorithm will not be able to buy with insufficient funds		
The algorithm will buy all available stocks when it has sufficient stocks to buy	The algorithm will buy all available stocks only if the MACD value is above 0	The algorithm will buy a percentage of the available stocks, dependant on the binned value the MACD value currently exhibits.

Table 4.3: Constraints for buying stocks in each of the algorithms

The constraints added to the action of selling are closely related to those of the action of buying described in table 4.3, they differ in that the algorithm will consider the number of available stocks, and in the technical indicator algorithms, will perform the action based on the MACD value being negative, or again according to the bins of the MACD. Once the environment is properly set up, the algorithm can be trained. The training data will be passed into the environment and the agent (PPO) will navigate the environment in order to generate and update the policy. Following this, the policy can be applied to the unseen, test data and the results can be recorded and evaluated.

4.4 Evaluation Metrics and Performance Analysis

Once the data has been correctly formatted and pre-processed, each of the the algorithms are applied to the training data set. The number of training epochs used may influence the probability of obtaining the optimal policy. The number of epochs is set to 1000, but this number can be adjusted for the processing power of the user's system. Each epoch will return a policy for the training episode.

The reward signal can be highly influenced by large changes in the user's maximum net worth, whilst the

relative state of the action taken by the algorithm will generate more consistent and smaller changes to the reward signal. Each epoch's policy is evaluated based on the total returned net worth of the user. This forces the algorithm to prioritise the total net gain of wealth, whilst still considering some immediate rewards. The policy that returns the highest net worth is thus called the optimal policy. This policy will be stored, and used to perform trading operations on the test set. The cumulative reward, as well as the choices of action at each stage in the series can then be plotted, this can give a visual representation of the effectiveness of the algorithm on the new data.

Each of the algorithms will be implemented and run as explained in section 4.2. With the specific constraints explained in table 4.3 being used in each of the respective 'take action' functions. The constraints that are used in the strategic technical indicator method are explained below. Depending on the desired action, the algorithm will either trade 20,40,60 or 80% of the stocks available to it. These percentages are determined as follows:

$$p2b = (MACD\_val + 1) * 0.2 \tag{4.1}$$

If the action is buy and

$$p2s = 0.8 - 0.2 * MACD\_val \tag{4.2}$$

If the action is sell.

The transformations described by 4.1 and 4.2 are shown in the table 4.4.

MACD bin	percentage stocks to buy	percentage stocks to sell
0	0.2	0.8
1	0.4	0.6
2	0.6	0.4
2	0.8	0.2

Table 4.4: Table showing the MACD bin and the corresponding percentage of stocks to trade.

Above, $p2s$ stands for percentage to sell, $p2b$ is percentage to buy. The $MACD\_val$ refers to the binned value of the MACD. This method serves to derive a method for using MACD indicators in a way other than the pure strategy- which has been described as being sub-optimal in Yingzi et al. (2009). The net worth of the individual after each of the test cases will be recorded and plotted. The optimal strategy will be derived from the comparison of the final net worth's generated by the algorithms. Each instance of buying and selling stocks

will also be recorded and plotted on top of the stock price trajectory. This will give an indication of the actions taken by the algorithm at specific times.

The net worth speaks directly to the profitability of the algorithm in question. If the algorithm has increased the user's net worth, the algorithm is profitable, otherwise not. The profitability of the algorithms thus form the basis of their comparison.

Since technical indicators are typically used as a means to mitigate risk in general trading environments, the latter algorithms serve to provide a comparative narrative between more risk adverse and risk prone algorithms. The pure and binned variations of applying the technical indicators in the algorithm provide the user with the ability to not only compare, but to adjust their risk parameters as they see fit. By adjusting equations 4.1 and 4.2, the user may choose how much stock to place in trusting the MACD indicator. Of course, the number of bins used for the MACD values may also be adjusted and then equations 4.1 and 4.2 should be updated accordingly. This serves to empower the user in governing their risk adversity functions in the environment. The application of these techniques in the algorithms will also provide insight into the effectiveness of their inclusion.

Each algorithm will be run on their own using the same PPO agent. The hyper-parameters of the PPO are not manually adjusted throughout the experiments. This ensures that the only differences in the algorithms are the constraints in the 'take action' function of the environment. This will provide increased insight into how the algorithms compare given their different constraints, and ensures that the differences in results are not generated by human interference.

The results of the algorithms will be compared by considering the final net worth of each algorithm, along with the graph displaying the net worth over time. The general trends of the graphs will be considered, as the final net worth may not encapsulate all of the most vital information. If the trend of the net worth over time seems to generally increase, the algorithm will be deemed healthy. Each test done on the algorithms will be repeated 100 times and the statistics pertaining to the 100 results will be collected for comparison.

4.5 Experiment Design and Validation

Each algorithm used will be designed and run using the same set of hyper-parameters in both the agent and the environment. The information pertaining to the stocks, along with the MACD and MACD bins will all be the same, this will prevent the results differing due to information asymmetry. The differences between the algorithms will be found only in the 'take action' section of the environment. In this way, the results will provide information about the effects of including the different technical indicator methods.

The algorithms in question are still subject to a number of randomised actions. This means that the optimal policy of one trained method may not exactly equal that of another, even if the number of training epochs is high. For this reason, the process of training and testing the algorithms will be repeated 100 times. In each of the 100 iterations, the resulting net worth generated by applying the the optimal policies to the test set are recorded.

The recorded net worth's are then plotted, and their summary statistics are generated. These statistics will give an idea of the general spread of the generated net worth's of the optimal policies. This will include information about highest returns, the standard deviation in the returns as well as the mean of the returns of the optimal policies. The plot of the graphs will illustrate these statistics, and will highlight the tendency for the algorithm to generate positive rewards, versus negative rewards. This information can be used to compare the algorithms in a more generalised manner. It provides information about which algorithms perform the best in terms of net worth growth, and can help capture information about the risk of applying the algorithms.

The policy that generates the highest net worth over the 100 iterations will be deemed the most optimal policy, and will have it's net worth over time recorded and plotted. The actions taken in this policy will also be plotted over the stock trend. This will provide insight into the general trends of the net worth over time, and elucidate the actions taken in order to produce this trend.

4.6 Implementation Challenges and Solutions

Throughout the creation of this algorithm, there have been a number of challenges that required attention. At the forefront of these challenges was the inclusion of a transaction cost into the environment. Before this was implemented, each of the three algorithms were producing promising results. However, once this 'minor' cost of 3% per stock sold was put into the algorithm, it started to fail drastically. To deal with this, the reward signal was looked at. The algorithm aimed to make as few transactions as possible- attempting to favour the more lucrative transactions. The signal was also updated to include information about the most recent traded stocks. If the last action was to buy stocks, then the cost of those bought stocks would be compared to the last time the stocks were sold and if they were bought at a lower price than when they were last sold, the algorithm would generate a positive reward to add to the signal. If the last action was sell, and the stocks were sold at a higher price than when they were last bought- the reward signal would again receive a positive boost. The opposite of these actions would incur a negative reward- offset so as not to completely discourage the exploration probability of the algorithm. Finally, the reward signal would have the transaction cost of the current transaction removed from it. This served to provide the algorithm with information about the losses incurred during trading.

Despite all of these changes to the reward signal, the algorithm still under-performed. For this reason, the transaction cost was removed form the algorithm at whole. Following this, a new cumulative transaction cost (ctc) was calculated throughout the algorithm. This cost can be viewed after the entire algorithm has been run. Due to the fact that the transaction cost is difficult to incorporate, it is advised that the transaction cost be either debited before the algorithm is run, with consideration of the expected return. Or, the algorithm can be modified to limit the amount of transactions taken. Once this limit has been reached, the transaction cost can be calculated as a percentage of the transactions to make.

Despite not being able to generate a reward signal that performed perfectly with the transaction cost in mind, the increase and decrease in the signal based on the recent stock sales proved highly beneficial, and thus the reward signal was updated to not only include information about the net worth of the user, but also to include information about the stock sales.

4.7 Ethical Considerations and Regulatory Compliance

The algorithms discussed in this model are all generated using random updates, with a reward signal that serves to promote good 'decision-making'. These 'decisions' should not be considered in the same paradigm as human decisions. They are based solely off of statistics and mathematics. For this reason, the algorithm should not be personified. As explained above, the stochastic nature of the algorithm implies an inherent lack of determinism. This means that one 'optimal policy' may not exactly equal another. This may be considered as being 'unfair' by the user, but it is impossible to truly say which of the policies will perform best on new data. Even after testing the values on the test set, which is made up of unseen data, a good result may not imply that said optimal policy is optimal for future cases.

That being said, the algorithms were all given the exact same starting parameters, and contain the same information. In order to achieve the most optimal policy, with regards to the test set, the algorithm may need to be run more than once, and the most optimal policy may be selected from the iterations. The algorithms used throughout this paper will remain unchanged and the results of applying each of them will be displayed and discussed below. The algorithms were generated with increased consideration about how to adequately mitigate risks in the trading environments, this was done in an attempt to diversify the options of the algorithm, allowing more players to use it and to generate wealth in a manor that they see fit.

The Algorithms were made with the consideration of the following compliance acts for algorithmic trading code both in the Eu and in the USA:

4.7.1 European Union

Markets in Financial Instruments Directive II (MiFID II):

MiFID II , European Securities and Markets Authority (ESMA) (n.d.*b*) imposes stringent requirements on firms engaging in algorithmic trading, including the implementation of robust systems and controls to manage risks associated with automated trading strategies. This includes pre-trade risk controls, continuous monitoring, and post-trade analysis to ensure market integrity and transparency.

Market Abuse Regulation (MAR):

MAR , European Securities and Markets Authority (ESMA) (n.d.*a*) addresses issues related to insider trading and market manipulation, mandating firms to report suspicious transactions and maintain high standards of market conduct. This regulation is crucial for preventing activities that could undermine market confidence and stability.

4.7.2 United States

Securities and Exchange Commission (SEC) Regulations:

- **Regulation NMS (National Market System):** , Securities and Exchange Commission (SEC) (n.d.*b*) Ensures that trading activities promote fair and efficient markets by maintaining price transparency and protecting investor interests.

- **Market Access Rule (Rule 15c3-5):** , Securities and Exchange Commission (SEC) (n.d.*a*) Requires brokers and dealers to implement risk management controls to prevent erroneous trades and ensure financial and operational risk management.

Commodity Futures Trading Commission (CFTC) Regulations:

Although Regulation AT (Automated Trading) , Commodity Futures Trading Commission (CFTC) (n.d.) is not fully implemented, its proposed guidelines emphasize the importance of risk controls, transparency, and registration for automated trading systems.

Financial Industry Regulatory Authority (FINRA):

- **Rule 5210:** , Financial Industry Regulatory Authority (FINRA) (n.d.*a*) Ensures that all trade reports are

accurate and not misleading.

- **Rule 5310:** , Financial Industry Regulatory Authority (FINRA) (n.d.*b*) Mandates best execution practices, ensuring that customer orders receive the most favorable terms reasonably available.

4.7.3 Compliance Measures Implemented

Risk Management: Pre-trade risk controls were implemented into the algorithms, from the simplest form of reward signal to more comprehensive forms such as the use of the technical indicators. After the completion of the algorithm, a number of post-trade analytics are made available in the form of plots for the user to consider.

Transparency and Reporting: Each trade made by the algorithm is stored in the optimal policy and the trade time and amount stored for the user to comply with relaying accurate reports to the respective regulatory authorities in order for those authorities to perform the necessary audits.

Market Conduct:

The algorithm has been designed to avoid practices that could be deemed manipulative or disruptive, in line with MAR and SEC guidelines.

Cybersecurity:

Since this algorithm is made for theoretical use at the moment, there has been no such endeavour to comply with this regulation as of yet.

Chapter 5

Experiments and Results

The experiments were set up as described in section 4.5. Each of the Algorithms were initiated with the same hyper-parameters and given the same initial information. The environment was initialised and trained using the same PPO algorithm for each of the agents- an optimal policy for the training data was found over 1000 epochs, and applied to the test data. This process was run 100 times for each of the methods and the data pertaining to the change in the user's net worth was collected and plotted. The descriptive statistics of the values were calculated and tabulated for ease of comparison. The most optimal policies for each of the 100 iterations were considered for plotting the change in net worth over time as well as the actions taken by the algorithms. This loop can be found at the bottom of the code- it will be commented out due to its computational expensiveness. Consideration about the user's computational limitations is required when running this section of the code.

5.1 Deep RL Method

	Net Worth
count	100.000000
mean	145222.017062
std	48611.771183
min	50368.574917
25%	109893.038835
50%	139357.666995
75%	169436.408326
max	283158.939123

Table 5.1: Descriptive statistics for the generated net worth of the Deep RL method over 100 trials

Table 5.1 above demonstrates the results of running the algorithm using the pure deep-RL method described in sections 2 and 4, over 100 iterations. The results depict the descriptive statistics of the net worths obtained from the algorithm after each of the iterations. A scatter plot of all of the values is available in the appendix of this paper A.1. From the data displayed in 5.1 and the image shown in A.1, a number of important results

can be obtained. Firstly- the data indicates that the maximum net worth obtained during this experiment was 283158.94. This indicates a profit of 183158.94 over the trading period. This is close to triple the amount of the starting value and by far the largest growth in net worth over all of the algorithms tested. The minimum value over the 100 iterations is shown to be 50368.57, which indicates a net worth loss of 49,631.43. This is the greatest loss generated by any of the algorithms. The standard deviation, too, was the highest shown standard deviation of any of the algorithms with a value of 48611.77. The trading portfolio for the highest output net worth is shown below along with the net worth over time of the same algorithm.

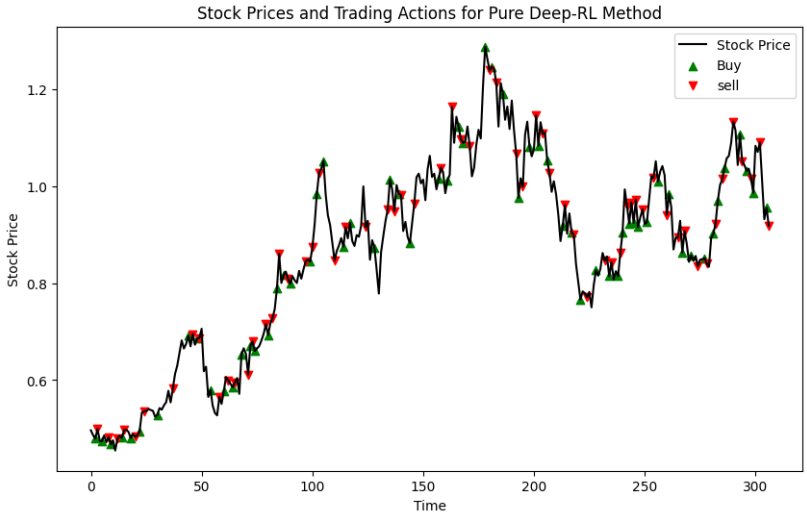

Figure 5.1: Graph indicating the actions of the highest yielding model using deep RL over 100 iterations

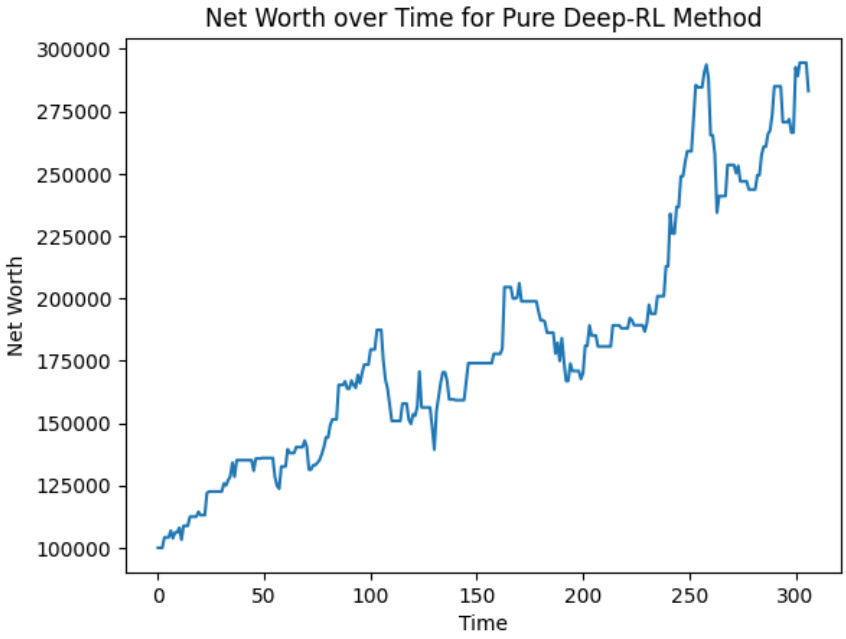

Figure 5.2: Graph showing the net worth over time of the highest yielding model using deep RL over 100 iterations

Figure 5.1 indicates that the model prioritised buying and selling stocks frequently in order to optimise trading. This model did not have the choice of selling or buying percentages of total available stocks- but worked by selling all or nothing as was described in chapter 4. Examining the graph of the net worth over time 5.2, the increase in wealth is healthy, and the profitability of this particular model is high, only losing value in small

areas, whilst never reducing the net worth below the starting value.

As shown, there are clear benefits to using the pure RL strategy. It should, however, be considered that the standard deviation of the model was the highest of any of the algorithms, together with the fact that the algorithm produced a loss in net worth of 17% of the time depicted in A.1. This indicates the highest number of negative returns of all of the algorithms. For these reasons, the algorithm may be considered to be one of high-risk high-reward.

5.2 Pure Technical Indicator Method

	Net Worth
count	100.000000
mean	121086.641636
std	22345.688511
min	80825.644306
25%	104242.132388
50%	117100.048827
75%	135096.996971
max	183134.104245

Table 5.2: Descriptive statistics for the generated net worth of the Pure Technical Indicator method over 100 trials

Table 5.2 above shows the results of applying the Pure Technical Indicator algorithm explained in section 4.4 of this paper. The results showed a reducuction in maximum net worth over the 100 iterations when compared to the deep RL algorithm. As indicated by figure A.2 in appendix A, the algorithm produced a negative return 15% of the time. This shows a 2% reduction in negative returns over all iterations. The algorithm also boasts the highest minimum net worth value in the experiments.

The standard deviation of the model was shown to decrease by a factor of 2.18x. This could imply that the variance of the outcome has been decreased, and that the algorithm, with the correct modifications, may prove effective in mitigating some of the underlying risks associated with this methodsof trading.

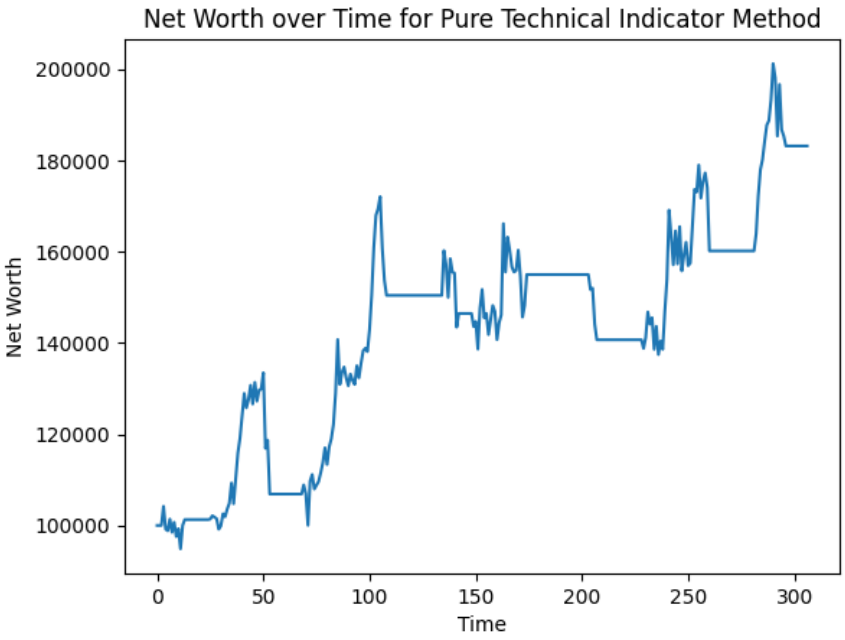

Figure 5.3: Graph showing the net worth over time of the highest yielding model using the Pure Technical Indicator method over 100 iterations

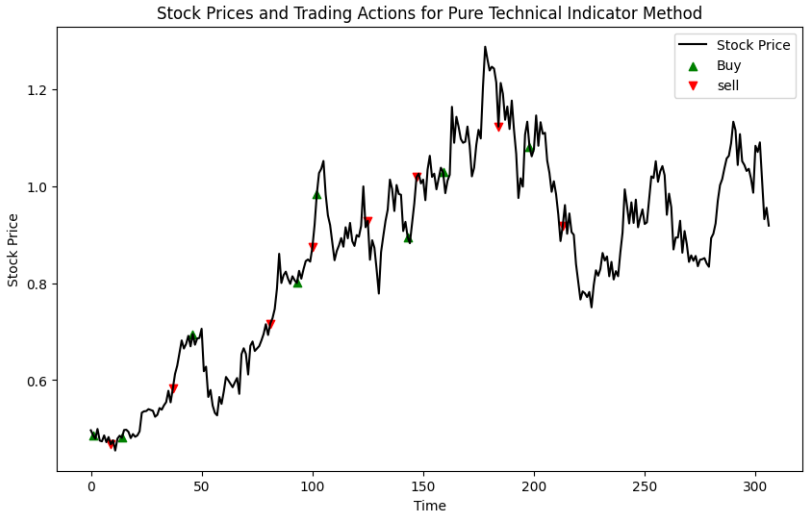

Figure 5.4: Graph indicating the actions of the highest yielding model using the Pure Technical Indicator method over 100 iterations

Figure 5.3 indicates a healthy growth in net worth over time. A difference in optimal policy decisions, however, can be seen in the graph of 5.4. The model prioritises holding both the assets for longer periods of time. This results in minimising the number of transactions made in the model. This is likely due to the inclusion of the additional constraints applied to the algorithm that were used to try and mitigate risky stock transactions in the environment.

The decrease in variance and reduced negative return percentage due to the inclusion of the technical indicators

implies an increase in risk mitigation. This implies that this algorithm can be considered as a moderate-to-high-risk, moderate-to-high-reward strategy. The risks associated with the algorithm are still valid, but the resulting poor performances are less detrimental than those of the previous method. This is indicated by the increased minimum value.

5.3 Strategic Technical Indicator Method

	Net Worth
count	100.000000
mean	119797.186193
std	19992.348160
min	73991.823949
25%	103560.357755
50%	117399.225861
75%	133962.173319
max	179646.056955

Table 5.3: Descriptive statistics for the generated net worth of the Strategic Technical Indicator method over 100 trials

Table 5.3 above shows the descriptive statistics of applying the same test procedure to the Strategic Technical Indicator model. The relationship between the stock bin and the percentage of the stock allocated to trade is based off of the considerations made in Yingzi et al. (2009), wherein Zhipeng et al. derived the first order approximation risk function. The first order approximation function assigns inverse proportionality between MACD value and the risk value. This method exhibited a further reduction in maximum net worth. The minimum value also dropped below the threshold of the Pure Technical Indicator algorithm.

The standard deviation, as seen in table 5.3 was reduced further, with the strategic TI algorithm generating the lowest standard deviation over all experiments. From figure A.3 in appendix A, the algorithm returned a negative result 14% of the times it was run. This is the lowest number of negative returns of all of the algorithms. These results could imply further risk mitigation generated by the algorithm.

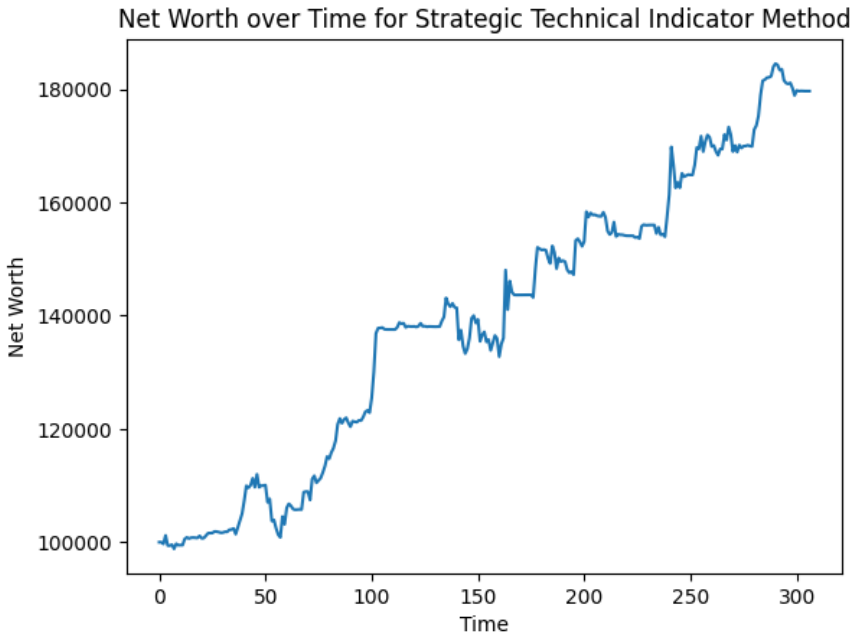

Figure 5.5: Graph showing the net worth over time of the highest yielding model using Strategic Technical Indicator method over 100 iterations

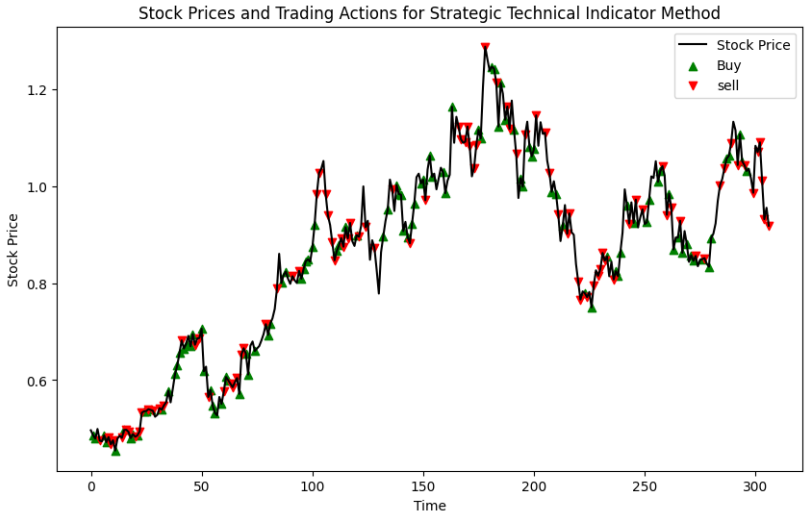

Figure 5.6: Graph indicating the actions of the highest yielding model using Strategic Technical Indicator method over 100 iterations

Figure 5.5 indicates a steady growth in net worth over time which is consistent with the previous methods. The trading patterns demonstrated by the strategic TI algorithm as depicted in figure 5.6 appear to be more similar to those of the deep RL method, (5.1) , than to those of the pure TI method as shown in figure 5.4. Indicating a preference for trading numerous stocks, whilst holding fewer assets for shorter periods.

Although this algorithm produced the lowest maximum return value of any of the algorithm, the reduced number of negative returns in net worth along with the reduced standard deviation in net worth over the 100

iterations provide compelling evidence that this variation of the code furnishes the user with a lower risk algorithm to use for what is still considered a good return on investment. For this reason, this algorithm may be labelled as a moderate-risk, moderate-reward algorithm.

Chapter 6

Discussion

Considering the results derived in Chapter 5 of this paper, there are numerous of aspects of the algorithms that can be considered successful and some that may prove to limit the algorithms performances. As shown by the results, the algorithms each proved their merit in use in the trading environment. With steady growth in net worth and generally positive returns, each of the algorithms exhibited the ability to generate policies on historical data that were useful for trading on unseen data.

The Pure Reinforcement Learning strategy performed the best in terms of maximum net worth generation, however also produced the lowest net worth in the 100 trials. The Pure Technical Indicator boasted the highest minimum net worth, and reduced the number of negative returns when compared to the Pure Reinforcement Learning method. However, it also produced a reduced maximum return over the 100 trials in comparison. The Strategic Technical Indicator method returned the lowest number of negative rewards; and lowest standard deviation of all the algorithms, however also produced the lowest maximum net worth of all of the algorithms.

6.1 Evaluation of Reinforcement Learning Algorithm

As stated above, the Reinforcement learning algorithm performed generally well as a stock trader. With the highest maximum net worth increase of any algorithm. The algorithm produced the highest standard deviation in net worth generation, and the highest number of negative returns, which indicates a comparably high volatility in returns.

One possible reason for the high volatility is that the algorithm has less constraints in its take action function. Despite containing the same information as the other algorithms about the stock trends, MACD values and

the MACD bins, there were no additional constraints applied to the actions that the algorithm can take. This allows the actor more freedom to make more stochastic decisions in the environment, which can result in an increase in the number of positive and/or negative actions taken.

In agreement with the article by , Liang et al. (2018), the algorithm showed the ability to capture the general sentiment and complex patterns of the stocks in order to generate actions that increase the users net worth over time.The higher volatility of the algorithm indicates the requirement for additional modifications to be made on the algorithm in order for it to be viable for the more risk adverse user, which aligned with the conclusions drawn in , Liang et al. (2018).

Examining graph 5.1, which indicates the actions taken by the algorithm overlaid on the stock projection chart it is evident that the algorithm did not always buy and/or sell at the most optimal times. There are a few instances for example, at times 150 and approximately time 200, where the algorithm bought at the trough, and at times, close to 100 and 150, where the algorithm sold at the peak of the stock values.

Although this behaviour would be considered optimal, it will be difficult to achieve consistently. The reward signal of the algorithm may be adjusted in a way that considers the gradient of the stocks trajectory in order to predict a turning point, for instance, receiving an increased positive reward for performing actions at these turning points. It is unclear whether or not this algorithm would out perform a seasons stock trader in this particular instance. That being said, considering the graphs 5.3 and 5.1, along with the time period of between 200 and 250, graph 5.1, illustrates that over this period, the stock showed a depreciating trend. Figure 5.3, however, maintains a relatively steady upward trajectory of net worth change over this period, selling at some of the minor peeks that exist. This illustrates this algorithm's ability to handle some of the complex pattern demonstrated by the stock's trajectory.

The PPO used in the algorithm proved itself useful in that it required no hyper-parameter tuning and was simple to implement, as stated in the paper by , Schulman et al. (2017). These factors make it a good option for individuals with limited knowledge of coding and financial techniques. There is a strong argument for the use of DDPG as an agent- given its continuous action space is useful for selling fractions of the total number of stocks available. However the hyper-parameter tuning makes this option more complicated for individuals with limited coding know-how. The use of DDPG as an agent, should be considered for future investigations.

This algorithm, despite being relatively risky in some senses, has shown its effectiveness through simplicity and general positive returns- it can be considered a useful tool for stock trading for potential investors. More investigation into the use of the algorithm is always encouraged, however if used correctly, this method can prove lucrative for the user. It should be noted that it is uncertain how exactly the algorithm will perform

in other stock environments, however from the tests that were run on the Apple stocks, the algorithm proved useful.

6.2 Impact of Technical Indicators

In section 5, we saw the results of applying each of the algorithms over 100 iterations. Compared to the Pure Reinforcement Learning method, the Pure Technical Indicator method generated a few concerning results. First and foremost, the maximum return of the algorithm was shown to decrease by a significant amount, from 283158.94 to 183134.10. This is a drop in over 100000. This implies that this algorithm, as stated in Yingzi et al. (2009), is not optimal. Graph 5.3 shows the net worth over time of the most optimal policy in the 100 iterations of the Pure Technical Indicator method.

Considering figure 5.3,with figure 5.4, which indicated the stock trajectory and the actions taken by the optimal policy, the net worth over time graph tends to mimic the trajectory of the stock prices in a number of areas, for example, at approximately times 50 and 100. This shows that the policy of this algorithm prioritised holding the risky asset (the stock) for longer periods. In contrast, there are periods where the net worth remains constant over periods, examples including, between times 50 and 100, 125 and 150. This implies that the algorithm has chosen to prioritise holding the risk free asset (the money stored on the account) and that the number of transactions is fewer than in the previous algorithm. This indicated that this algorithm prioritised minimal- possibly more valuable- transactions.

These relatively long periods of holding contrast the Pure Reinforcement Learning method, in which there still exist periods over which both the risky and risk free assets were held, however the duration of these periods is far shorter by comparison- particularly in holding the risk free assets. This implies that the Pure Technical Indicator algorithm is prone to holding on to assets rather than making many transactions over time- which could be a sign that the algorithm is using the MACD values in an attempt to reduce the number of risky transactions.

The inclusion of the Pure Technical Indicator constraints did not generate only negative impacts on the returns. As shown in table 5.2, the standard deviation of the returns of the algorithm over the iterations, when compared with the Pure Reinforcement Learning case was drastically reduced. The number of negative returns of the algorithm as shown in figure A.2, was also reduced from 17% to 15%. When compared with the Pure Reinforcement Learning strategy, the minimum value was also shown to increase over all of the iterations. This implies that the risks of using this version of the algorithm differ to those of the Pure Reinforcement Learning algorithm.

Before concluding that this algorithm is less successful than the previous algorithm in maximum net worth generation, it should be noted that the transaction fees of each of the algorithms were not taken into consideration. This means that depending on the kind of transaction fee charged for the trading of stocks, the results may vary. For example, if a set transaction fee was used by the trader, then it could be argued that the Pure Reinforcement Learning method proved more efficient, however, if the transaction fee was taken as a percentage of the trades made, then there may be an argument in favour of the Pure Technical Indicator method.

6.3 Effectiveness of Strategic Technical Indicators

When compared with the previous two methods, the Strategic Technical Indicator method displayed both advantages, as well as disadvantages in its results. Considering figure 5.5, the net worth over time generated by the optimal policy of the algorithm exhibits steady increase. The similarieties between figures 5.5 and 5.2 imply that the optimal policy found by this algorithm (akin to that of the Pure Reinforcement Learning algorithm) values numerous trades. Unlike the Pure Reinforcement Learning method, however, the maximum return of the algorithm was below even that of the Pure Technical Indicator method.

The minimum return, although larger than that of the Pure Reinforcement Learning strategy, was also smaller than the Pure Technical Indicator method, with many more transactions being made. This may have negative implications about the algorithm, however when the standard deviations of all of the algorithms are compared, the strategic method out-performed the other two methods quite convincingly. These statistics can be seen in tables, 5.1, 5.2 and 5.3.

This method was a curde assumption of how to correctly utilise the technical indicators, in an attempt to rectify the concerns of non optimal usage referred to in Yingzi et al. (2009). The use of binning may not have proved the best method for using the technical indicators, and thus more optimal methods of the use of these functions may still be derived. This does, however serve as an initial attempt in the right direction for using these values.

Despite the use case not necessarily being optimal, the use of this method was shown to help in lowering the variance of the results, as well as reducing the number of negative results. This implies that the apopted approach may have proved useful in mitigating some portion of the risks of applying this algorithm for stock trading.

6.4 Comparison of Algorithms

Each of the three algorithms has their own merits in use. The results are based off 100 iterations of the algorithms. Due to the stochastic nature of the algorithms, 100 iterations may not prove to be statistically

significant enough to provide decisive evidence to choose one over another. Given the information at hand, the user should consider what they primarily value in order to make a decision on what algorithm to choose.

The choice of algorithm should also depend on the type of transaction cost that the user would incur. Below is a table indicating the level of acceptable risk taken by the user, together with what we would suggest as the optimal algorithm to use given the type of transaction cost incurred by the user.

Risk	**Once-off transaction cost**	**Percent based transaction cost**
Lowest	Strategic	Pure TI
Medium	Strategic or Pure TI	Pure TI
Highest	Pure Reinforcement Learning	Pure TI

Table 6.1: Transaction Fees vs. Risk Levels

Table 6.1, which reflects our considerations on the algorithms,illuminates that the algorithm that appears most frequently is the Pure Technical Indicator algorithm. This is due to the fact that it emphasises minimal transactions and illustrates a useful risk-reward balance.

This does not necessarily mean that this algorithm is better or out performs the other algorithms. The decision about algorithmic risk-reward balance should be considered before the algorithm is implemented. According to this analysis, the algorithms tend to mitigate risk in order of increased constraints. The Pure Reinforcement Learning method is shown to demonstrate the highest risk, with the highest potential for growth, the Pure Technical Indicator method illustrated the second highest risk, with the second highest potential for growth, and the Strategic Technical Indicator method indicated the lowest risk with the lowest potential for growth.

In terms of transaction costs, the Pure Technical Indicator method performed the least amount of transactions, and would thus be the most likely to mitigate this cost if it were on a percentage scale. The Strategic Technical Indicator method performed a similar number of stock transactions as the Pure Reinforcement Learning method, however was given the option to sell a percentage of the total stocks available, which could implicit a lower transaction than the Pure Reinforcement Learning strategy.

It should not be overlooked that the Pure Reinforcement Learning strategy outperformed both the pure and Strategic Technical Indicator strategies in producing the maximum net worth over all of the iterations. This shows the merit in using this algorithm for those who are less risk adverse.

In conclusion, the algorithms each exhibit the ability to generate positive returns, when implemented correctly. The risk-reward balance is shifted by the implementation of the constraints in the environment. The Pure Reinforcement Learning algorithm incurs the highest risk, with the highest reward. The Strategic Technical Indicator algorithm incurs the lowest risk and the lowest reward, whilst the Pure Technical Indicator method

strikes a balance between these two algorithms. It is not correct to say that any algorithm is definitively better than another, and the choice of algorithm should consider the risk-reward payout.

6.5 Addressing Research Questions

Effectiveness of algo-trading:

Each of the algorithms exhibited the ability to produce positive growth in the user's net worth over a set period, failure to produce positive change in net worth over time occurred in a combined average of 15.3% of the test cases. This means that for a combined average of 84.7% of the time, the algorithms generated positive rewards. This indicates that the algorithms in question are useful for stock trading. These results were generated in each case with only historical data as training in the algorithms. This highlights abilities of the algorithms to learn complex patterns of the environment for trading. No hyper-parameter tuning is required for the algorithms, and minimal data pre-processing is required. The stability and robustness of these algorithms is thus exemplified.

Overall, the results indicate that the algorithms are effective in stock trading, providing a solid foundation for further development and application in real-world trading scenarios. whilst we did not compare the results to more traditional algorithms, the positive outcomes suggest that these algo-trading strategies are a viable option for enhancing trading performance.

Enhancement with technical indicators:

The integration of technical indicators in the algorithms proved useful in a number of ways. Despite reducing the possible maximum output, both algorithms that integrated constraints pertaining to the technical indicators showed promise in mitigating risks associated with this kind of trading. The reduction in volatility in returns also indicates the advantages of including these functions into the algorithms.

In summary, whilst the integration of technical indicators may slightly reduce the maximum potential output of the trading algorithms, however, their benefits in risk mitigation and performance stability make them a valuable addition. These enhancements help cater to the needs of more conservative investors, ensuring a more reliable and less volatile trading experience.

Risk mitigation and general performance:

Risk is a complex value to try and quantify in terms of raw figures. Both of the technical indicator methods saw a reduction in standard deviation when compared to the Pure Reinforcement Learning method. The

reduction in standard deviation suggests a reduction in volatility, which implies a lower level of risk in this context. In contrast to this, the reduction in maximum and mean net worth gain generated by the algorithms with technical indicators incorporated may, at first glance imply an increased risk to returns. However, risk assessment should incorporate information about the frequency and magnitude of the negative returns. The key benefits of using the technical indicator methods are evidenced by their ability to lower standard deviation and reduce the number of negative returns. For this reason, they are considered invaluable assets to incorporate into the algorithms. However, the balance between risk mitigation and reward is a concept that requires more consideration in future experiments.

6.6 Market Conditions, Algorithmic Design and Data Quality

The algorithms proposed in this paper were trained using one set of stock data. As a result, the trading policies derived on this data set are likely non-transferable to other, unseen stock data. To generate meaningful results on new data, the algorithms will have to be retrained on said data before being used.

As made evident by graphs 5.1, 5.4 and 5.6, when considering the net worth over time graphs of, 5.2,5.3, 5.5, it is important to highlight that even in areas of loss in stock value, the change in net worth was typically positive. It is unknown, however, how well these algorithms will fair when used on particularly complex stocks. Thus, the market conditions may play a role in the effectiveness of these algorithms, and should be considered when choosing a stock option to invest in. The onus is the user's, to make sure they properly train and test their algorithms before making use of them in the market.

For ease of comparison, each of the algorithms were trained and tested using the same hyper-parameters and the same reward signals. These parameters and signals may not be the ideal choice for each of the algorithms in question. It is therefore recommended that the user consider adjusting these values when applying their algorithm.

Examples include: changing the percentage of stock to trade when a particular bin is encountered in the take action function. This can affect the number of stocks being traded at a given time, which can effect the total change in net worth. Adjusting the reward signal to limit the number of transactions being made. This can help the code to prioritise more effective trading over more frequent trading and can affect the transaction cost generated.

The number of changes that can be made to the environment and the actor of this algorithm are numerous, however if the user is not comfortable with adjusting the algorithm to implement changes, the algorithms may be used as they are, with the caveat that they should be trained and tested on the chosen stock data, the results

considered before their use.

The quality of data in the use of this algorithm plays a vital role in generating useful results. The algorithm cannot be expected to learn meaningful patterns nor changes in stock behaviour without being afforded the opportunity to learn these behaviours in the form of data. If this algorithm is to be used for stock trading, it is recommended that the data pertaining to the stock be as close to complete and as clean as possible.

6.7 Limitations and Future Work

The results generated were done so over 100 iterations. This was done in an attempt to get a sample mean that is close enough to the approximate true mean of applying the algorithm at any given time. Although this is not a particularly large sum of iterations, it took 45 minutes per algorithm to run these functions.

In order to obtain the true optimal policy, or a better approximation of the true mean with regards to net worth returns, the policy may require hundreds, if not thousands, of iterations. This requires a vast amount of computational power, which should be considered. The process may possibly be sped up by using the computer's GPU, however this has not been tested as of yet.

There may also be some existing limitations in financial knowledge exhibited through out this paper. Despite studying some data driven models for investment, finance is a broad field with many different facets. There are a wide variety of functions which were not considered. These techniques and functions may be substituted into the algorithm in order to improve the general performance of the algorithm.

Ultimately, it is encouraged that the user studies financial trading techniques further in order to optimise their trading results. Further studies into these techniques may afford the user the ability to generate more effective constraints for use in the environment. This could optimise the risk-reward balance of the algorithm, increasing the productivity of the algorithms in question.

The Strategic Technical Indicator method was created in a crude attempt to placate the concern of non optimality expressed by Yingzi et al. (2009). This method is far from optimal itself, however with further research and dedication, this method may be optimised, either by using continuous values instead of binning, or by optimising the binned values to correspond to more optimal percentages for transactions.

Another area that requires further attention and future research is that of the reward signal. The reward signal governs how the algorithm as a whole makes decisions. If this signal is correctly modified, then the algorithm may be 'motivated' to perform more optimally.

The use of reinforcement learning paired with technical indicators in algorithmic trading is a multi-faceted field of study that requires further work and consideration before it can be deemed optimal. The work done in this paper is aimed at highlighting some of the major benefits and disadvantages of using some of these techniques. Whilst this paper may be useful for fundamentals of applying the algorithms in question, there are numerous algorithms and mathematical techniques to consider. These function and techniques could serve to optimise future iteration of this algorithm, further improving their utility.

Chapter 7

Conclusion

The stock trading sector in finance provides a multifaceted and dynamic environment characterized by highly complex patterns and trends that can be difficult to interpret even for seasoned stock traders. Formulating an efficient trading strategy can take years of experience and may require significant monetary and temporal investment. This serves as a barrier to entry to market access, separating those with the knowledge and means to obtain it, from those without. The aim of this paper was to explore the possibility of lowering this barrier to democratize the stock market by providing users with an algorithm that requires minimal understanding and intervention to run effectively. Furthermore, the algorithm should accommodate the needs of both risk-tolerant and risk-averse users according to their preferences.

By using reinforcement learning combined with supervised learning techniques and financial technical indicators, we developed and tested three algorithms. As detailed in Chapters 5 and 6, the results show promising outcomes with an overall net worth gain in the vast majority of cases. whilst there is a possibility of generating negative returns, these instances were limited to 17% in the riskiest algorithm and 14% in the most risk-free algorithm. This implies that with correct implementation, the algorithms in question will return positive results in the majority of cases.

Evidence suggests that the inclusion of financial technical indicators, as stated in Yingzhi et al. (2009), can reduce some of the risks related to stock trading. These technical indicators, such as MACD, proved to be particularly beneficial for risk-averse users. That being said, the use of these values may still not be optimal at this stage in the algorithms' development. Further research into how to more effectively utilize them, along with modifications to the reward signal of the algorithm, is encouraged to enhance their effectiveness.

To conclude, the use of deep reinforcement learning coupled with financial technical indicators demonstrated the ability to adapt to and navigate the complex patterns of the financial stock market in a way that produced a positive return in net worth over time. The algorithms developed do not require extensive hyper-parameter tuning nor data pre-processing, making them accessible to users with limited financial market knowledge. The inclusion of financial technical indicators mitigated some of the surrounding risks of applying the algorithm, underscoring their potential for democratizing the stock market.

Future research should focus on refining the algorithms, particularly in the use of financial technical indicators with an enhanced reward signal, to further improve performance and reduce risk.

Appendix A

The First Appendix

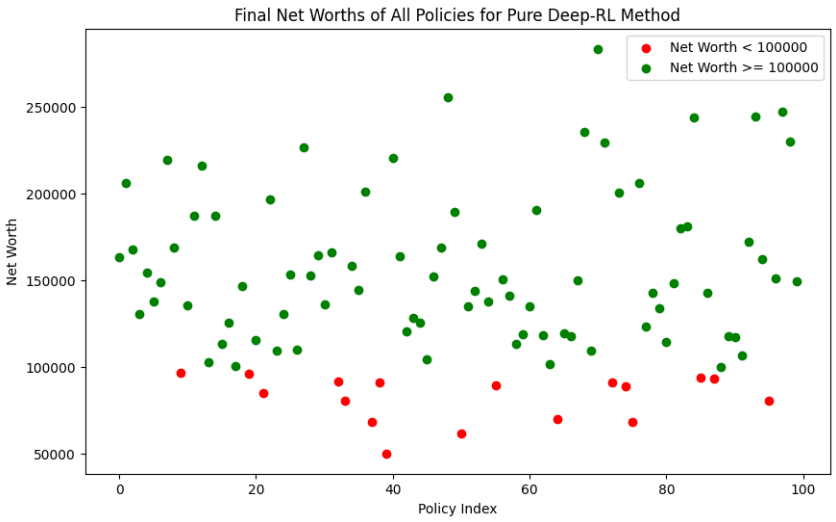

Figure A.1: Scatter plot showing the resulting netwoths of all of the 100 iterations of running the pure RL algorithm

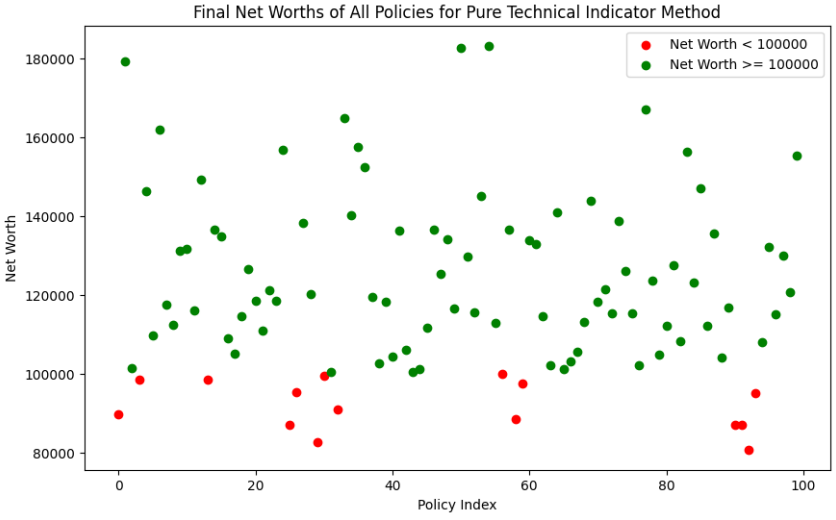

Figure A.2: Scatter plot showing the resulting netwoths of all of the 100 iterations of running the pure technical indicator algorithm

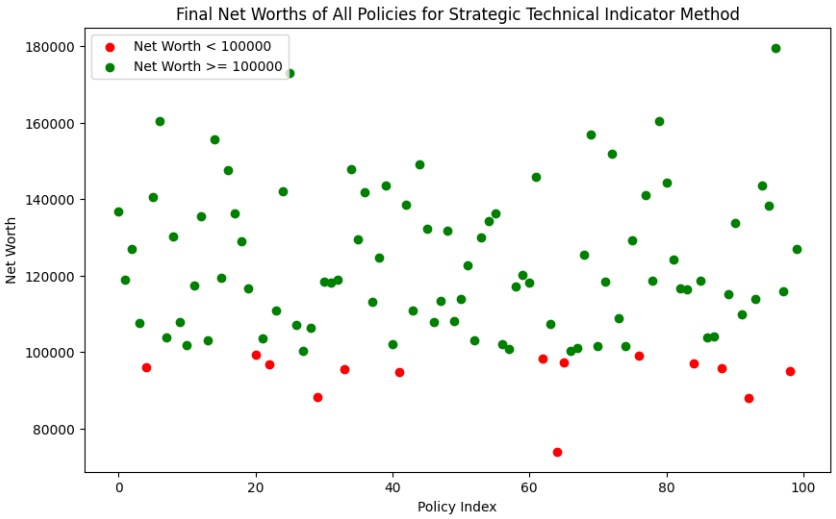

Figure A.3: Scatter plot showing the resulting netwoths of all of the 100 iterations of running the strategic technical indicator algorithm

Bibliography

AI, T. T. (2023), 'Building an algorithmic trading strategy using reinforcement learning.'. Accessed: 10/04/2024.
URL: *Member-only source*

Bellman, R. (1957), *Dynamic Programming*, Princeton University Press.

Bishop, C. M. (2006), *Pattern Recognition and Machine Learning*, Springer.

Commodity Futures Trading Commission (CFTC) (n.d.), 'Regulation at (automated trading)'. https://www.cftc.gov/PressRoom/SpeechesTestimony/opaquinterative.

Cowles, A. (1933), 'Can stock market forecasters forecast?', *Econometrica* **1**, 309–324.

European Securities and Markets Authority (ESMA) (n.d.*a*), 'Market abuse regulation (mar)'. https://www.esma.europa.eu/policy-rules/market-abuse-regulation.

European Securities and Markets Authority (ESMA) (n.d.*b*), 'Markets in financial instruments directive ii (mifid ii)'. https://www.esma.europa.eu/policy-rules/mifid-ii-and-mifir.

Fama, E. F. & Blume, M. (1966), 'Filter rules and stock market trading', *Journal of Business* **39**, 226–241.
URL: *https://www.journals.uchicago.edu/doi/10.1086/294131*

Financial Industry Regulatory Authority (FINRA) (n.d.*a*), 'Finra rule 5210'. https://www.finra.org/rules-guidance/rulebooks/finra-rules/5210.

Financial Industry Regulatory Authority (FINRA) (n.d.*b*), 'Finra rule 5310'. https://www.finra.org/rules-guidance/rulebooks/finra-rules/5310.

Kaelbling, L. P., Littman, M. L. & Moore, A. W. (1996), 'Reinforcement learning: A survey', *Journal of Artificial Intelligence Research* **4**, 237–285.

Liang, Z., Jiang, K., Chen, H., Zhu, J. & Li, Y. (2018), 'Deep reinforcement learning in portfolio management', *Likelihood Technology & Sun Yat-sen University* .

Lo, A., Mamaysky, H. & Wang, J. (2000), 'Foundations of technical analysis: computational algorithms, statistical inference, and empirical implementation', *Journal of Finance* **55**, 1705–1765.

Lo, A. W. (2010), 'Hedge funds: An analytic perspective'. http://www.jstor.org/stable/j.ctt7rq28.

McCulloch, W. S. & Pitts, W. (1943), 'A logical calculus of the ideas immanent in nervous activity'. Accessed: 27/03/2024.
URL: *https://www.cs.cmu.edu/ ./epxing/Class/10715/reading/McCulloch.and.Pitts.pdf*

Murphy, J. J. (1999), *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*, New York Institute of Finance.

Puterman, M. L. (1994), *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, John Wiley & Sons.

Rosenblatt, F. (1958), 'The perceptron: A probabilistic model for information storage and organization in the brain', *Psychological Review* **65**(6), 386–408.

Ross, S. M. (2014), *Introduction to Probability Models*, 11th edn, Academic Press.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986), 'Learning representations by back-propagating errors', *Nature* **323**, 533–536.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. *&* Klimov, O. (2017), 'Proximal policy optimization algorithms', *OpenAI* . {joschu, filip, prafulla, alec, oleg}@openai.com.

Securities and Exchange Commission (SEC) (n.d.*a*), 'Market access rule (rule 15c3-5)'. https://www.sec.gov/rules/final/2010/34-61379.pdf.

Securities and Exchange Commission (SEC) (n.d.*b*), 'Regulation nms (national market system)'. https://www.sec.gov/fast-answers/divisionsmarketregmrexchangesshtml.html.

Smigel, L. (2023), 'Algorithmic trading history: A brief summary'. Accessed: 20/03/2024.
URL: *https://analyzingalpha.com/algorithmic-trading-history*

Sutton, R. S. & Barto, A. G. (2018), *Reinforcement Learning: An Introduction*, 2nd edn, The MIT Press.

Thomas, P. S., Jordan, S. M., Chandak, Y., Nota, C. & Kostas, J. (2019), 'Classical policy gradient: Preserving bellman's principle of optimality'. Accessed: 27/03/2024.

Zhu, Y., & Zhou, G. (2009). Technical analysis: An asset allocation perspective on the use of moving averages. Journal of Financial Economics, 92(3), 519-544.