This is the report of the project we did in the official course “Reinforcement Learning" in Chennai Mathematical Institute; instructed by Prof. K. V. Subrahmanyam.

Automated Stock Market Investment Using Reinforcement Learning

Ankan Kar[1], Ishan Banerjee[2], Bijayan Ray[2], Praveen Kumar[3], Shreyam Banerjee[3]

[1] Computer Science, Chennai Mathematical Institute
[2] Mathematics and Computer Science, Chennai Mathematical Institute
[3] Data Science, Chennai Mathematical Institute

Abstract:
This report explores several approaches to developing an automated method for stock trading, with a primary focus on the application of Deep Q-Learning (DQN) and comparisons with other methods such as the Z-score algorithm, Q-Learning. The report also presents a comparative study of different approaches to defining state space and rewards in reinforcement learning models.

Index Terms: Z-score, Q-learning, Deep Q-learning, DQN, Trading, Single stock, Multi-stock.

1 Introduction

The integration of artificial intelligence (AI) and financial markets has revolutionized automated stock market investment, with reinforcement learning (RL) at the forefront of this transformation. RL involves training algorithms to make sequential decisions by rewarding desired actions and penalizing undesired ones, mimicking human learning but with the ability to process vast amounts of data and respond to market changes at superhuman speeds. Key RL methods such as Q-Learning, Deep Q-Networks (DQNs), Policy Gradient methods, Actor-Critic models, and Proximal Policy Optimization (PPO) have shown significant promise in this domain. Q-Learning focuses on learning the value of actions, while DQNs enhance this by using neural networks to handle complex state spaces. Policy Gradient methods optimize actions directly, and Actor-Critic models combine value-based and policy-based approaches for stable learning. PPO further refines policy updates to balance exploration and exploitation. In [4] we can get an idea about reinforcement the quantitative trading. These advanced techniques enable the development of adaptive, intelligent, and potentially more profitable trading strategies, marking a significant advancement in the field of automated stock market investment.

2 Topics We Focused On

In this report, we delve into various aspects of automated stock market investment using reinforcement learning, focusing on the following topics: To assess the potential of deterministic trading strategies, we began with the Z-score algorithm, which measures how far a data point is from the mean, in terms of standard deviations. This method provided a benchmark for evaluating the profitability of straightforward statistical strategies. We then implemented Q-Learning to evaluate its performance in dynamic trading environments. Q-Learning allowed us to create a model that could learn from past experiences and improve its decision-making process over time. Finally, we advanced to Deep Q-Learning, which leverages neural networks to handle complex state spaces and improve the accuracy of action-value predictions. This progression from simple statistical methods to advanced reinforcement learning techniques illustrated the increasing potential for optimizing automated stock market investments.

3 Data Availability

For our analysis and implementation of reinforcement learning models, we utilized historical stock price data from Yahoo Finance. Specifically, we focused on the stock prices of three major companies: Google (GOOGL), Tesla (TSLA), and Tata Consultancy Services (TCS). The data included daily open, high, low, close prices, adjusted close prices, and trading volume. By leveraging this data, we ensured that our reinforcement learning algorithms were trained on reliable and widely-used datasets, facilitating accurate and relevant investment strategy development. Our dataset was spanned in the time period “2014-01-01" to “2024-01-01". We used this time span of data for all the models we used. In Figure 8 to 3 we have the plot for Open and Close Values for the three datasets.

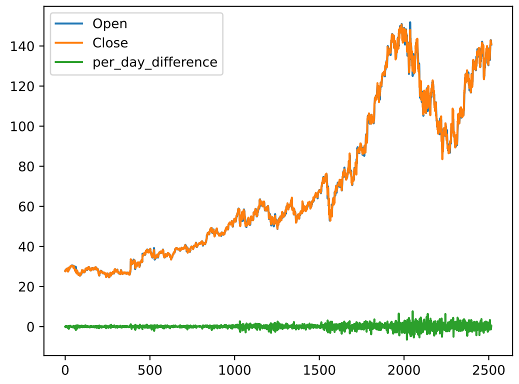


Figure 1. Google

4 Trading using Z-score

The Z-score is a statistical measure that quantifies the number of standard deviations a data point is from the mean of a dataset. In the context of trading, the Z-score is used to identify whether a stock is overbought or oversold.

The Z-score is calculated using the formula:

$$Z = \frac{X - \mu}{\sigma}$$

where X is the current stock price, μ is the mean of the stock prices

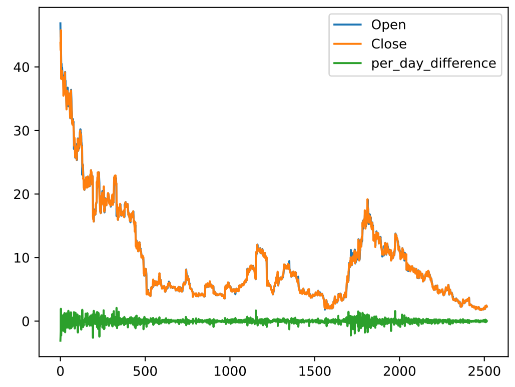


Figure 2. TCS

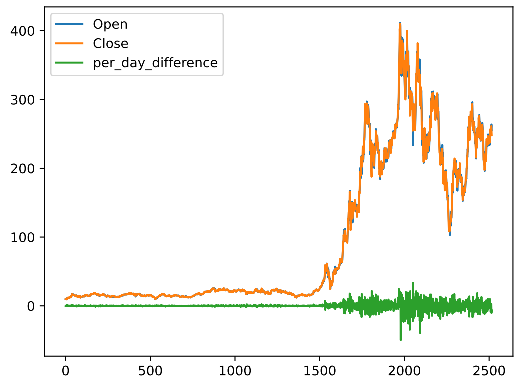


Figure 3. Tesla

over a specified period, and σ is the standard deviation of the stock prices over the same period.

A high positive Z-score indicates that the stock price is significantly above the mean, suggesting it may be overbought. Conversely, a low negative Z-score indicates that the stock price is significantly below the mean, suggesting it may be oversold. Traders use these signals to make buy or sell decisions, aiming to capitalize on the reversion to the mean.

Several studies have explored the effectiveness of the Z-score in trading strategies. In Gatev, Goetzmann, and Rouwenhorst (2006), the research focused on pairs trading, using the normalized methods, similar to Z-score to identify pairs of stocks that have historically moved together. The study concluded that this method could generate significant returns, particularly in periods of high volatility (*Gatev et al.*, 2006). In [1] we can see the relevance of ratios in Z-score Model for Predicting Bankruptcy where the study was done on Nifty.

These studies highlight the versatility and effectiveness of the Z-score in various trading strategies, demonstrating its potential to improve trading decisions and outcomes. Based on these idea we developed the following simple Z-score Algorithm for single stock trading.

Algorithm 1 Stock Trading Algorithm using Z-score

```
Input: mov_zscore, $df$
Output: money, stocks
1  begin
2      for $i \leftarrow n$ to $length(df)$ do
3          if mov_zscore[i] > 1 then
4              while stocks > 0 do
5                  money += $df$['Close'][$i$]  stocks -= 1
6          if mov_zscore[i] > 0.5 then
7              if stocks > 0 then
8                  money += $df$['Close'][$i$]  stocks -= 1
9          if mov_zscore[i] < -0.5 then
10             if money > $df$['Close'][$i$] then
11                 money -= $df$['Close'][$i$]  stocks += 1
12         if mov_zscore[i] < -1 then
13             while money > $df$['Close'][$i$] do
14                 money -= $df$['Close'][$i$]  stocks += 1
15     return money, stocks
```

Here the mov_zscore[i] stores the z-score based on the sample size taken from i – window-size to i and df is the dataset.

5 Trading using Q-Learning

Stock prediction using a simple version of Q learning isn't the best but it certainly does perform better than random. In [2] we get an idea of automated trading in equity stock markets using Q-Learning. In our approach we defined the state space as the tuple (nst, t, m) where nst represents the number of stocks we have at time step t of an episode and m represents the balance money we are left with. Action at any time step is to choose whether to buy, sell or hold a stock. Rewards are the profits made in each step. The basic form of Q-Learning takes the following form as pseudocode.

$$Q(s,a) = Q(s,a) + \alpha(R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

We took the values $\alpha = 0.1$ and $\gamma = 0.9$.
A major setback of this approach is the consideration of a discrete state space. Indeed considering the state space we considered we are missing out on a number of crucial factors like the current stock value.

6 Trading using Deep Q-Learning

The main idea of the learning using Deep Q learning is to gain experience, revisit them, learn and proceed. For a stable approach, usually two neural networks are used for the purpose. In [3] we can see applications of Deep Reinforcement Learning in developing stock market trading strategies. One of them we will call the Target network and the other Current network. We store the experience in a memory and sample the experiences uniformly at random. The Current network is trained on the basis of the values predicted by the Target network and the final weights are copied to the Target Network. .

$$Q(s,a) = Q(s,a) + \alpha(R(s,a) + \gamma \max_{a'} Q_{target}(s',a') - Q(s,a))$$

6.1 A variant of DQN for single stock trading

This approach does not particularly use Deep Q learning but is a simple variant of it. Instead of having two neural networks, we are working with a single one. In this approach, we take the experience gathered from an episode. Using the same network, we make the model learn on the experience. In other words, the target and the current network are the same. The state space we considered are the tuples containing the Opening, Closing, Highest and Lowest Price of the stock for the current day. The rewards considered are the total worth of stocks presently in hand along with the balance. Our action in this case is same as before.

6.2 DQN on a single stock

In this approach we have the conventional method of using two neural networks. Additionally its a more stable approach as the target network remains fixed and only get the copies of the current network weights after a certain memory has exceeded. In this approach our action includes buying a stock, holding the stocks and selling away all the stocks. The state space includes a ninety days window which records the number of change in the stock price (ith index says the difference in the stock price between the ith and the $(i-1)$th day.) along with the total worth of stocks we have presently. In memory, we will have stored the prices at which the stocks are bought and the reward will be the total profit earned on selling all the stocks. It is possible to calculate as we already have the stock prices at which we had bought them.

6.3 Multi-Stock Trading

This was done by modifying the state space of DQN on single stock. We simple consider the new state space consisting of lists $stocks$, $current\ prices$ and the variable $balance$ where $stocks$ contains the list of stocks owned, that is, its ith entry is the number of stocks owned of ith time. Similarly the ith entry of $current\ prices$ is the price of the ith stock at the current time and the $balance$ variable that stores the amount of money in hand as cash. With this the networth at a time becomes $balance + stocks \cdot current\ prices$ (here dot product means sum of element-wise product). We train the neural network with this modified state space using Deep Q learning algorithm and the let the model predict the stock market. Some contrasting changes that we have made while training this model based on intuition, compared the previous versions are:

1. We have changed the reward function to networth of the user. So the neural net tries to optimize its parameters in order to increase the networth of the user at any time instant which intuitively makes sense.
2. We have set a transaction limit of 100 stocks. That means if we have the action as sell, and we have more than 100 stocks we don't sell more than 100 stocks. And if we have enough money to buy more than 100 stock, we still don't buy more than 100 stocks when the action is. This intuitively makes sense because, based on a probabilistic prediction we want to bound the changes and ensure trading versatility, that is trading on different stocks and not focusing too much on just one.

7 Observations

The observations we got from the three methods discussed is described below:

7.1 Z-score Method

The observations for net income for this method can be seen in Figure 4 to 6.

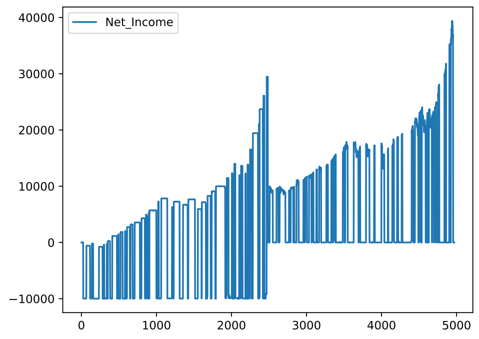


Figure 4. Google

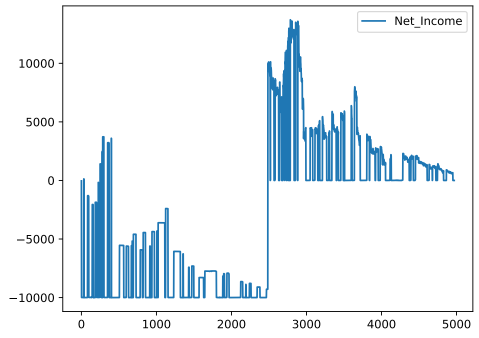


Figure 5. TCS

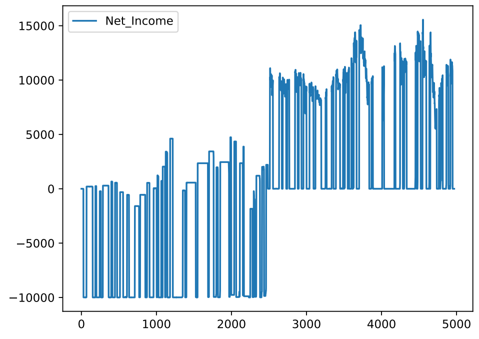


Figure 6. Tesla

We observed that the Z-score method ultimately yields a profit over a 5-year testing period; however, the profit is relatively minimal.

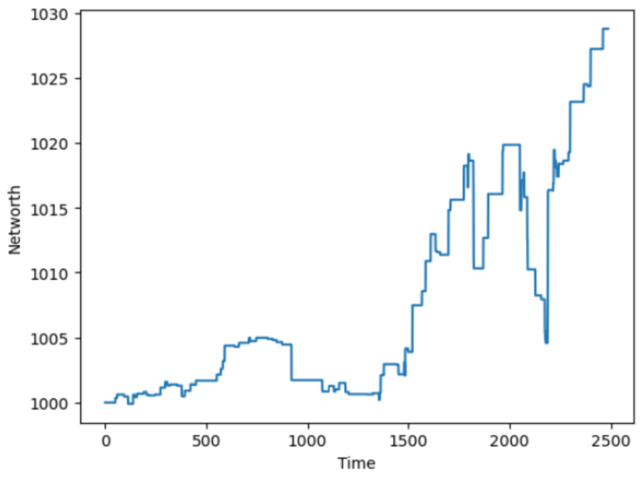


Figure 7. Variant of DQN for single stock

7.2 *Q-Learning*

Using Q-learning the observations we get were not great as it hardly made any risks of buying a stock and hence the net profit nearly remained 0. The learning curve where the reward was taking as the profit, approached zero on training since it refrained from taking any risk.

7.3 *Deep Q-Learning*

7.3.1 Variant of DQN for Single Stock: The different variant of DQN for a a single stock made a profit of roughly $30 dollars as can be seen in the figure 7.

7.3.2 Single Stock DQN: Here we enlist a few observations of how our DQN model performed on single stock trading. The observations for loss and reward on each of the three datasets is given in Figure 8 to 10.

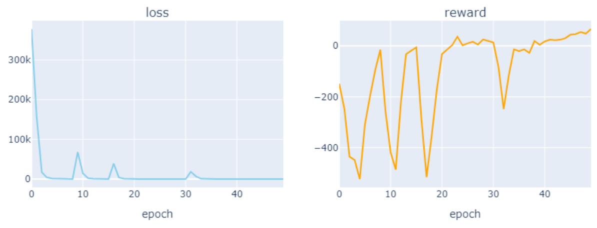


Figure 8. Google

The actions taken, rewards and profits are given in Figure 11 to 13.

We can see that the loss and the reward converged well enough over and gave a net profit, although less for a 5 year span.

7.3.3 Multi-Stock DQN: Here we enlist a few observations of how our DQN model performed on multi-stock trading. We have included to performances here:

Run 1: We have initial investment of $1000 which when traded with

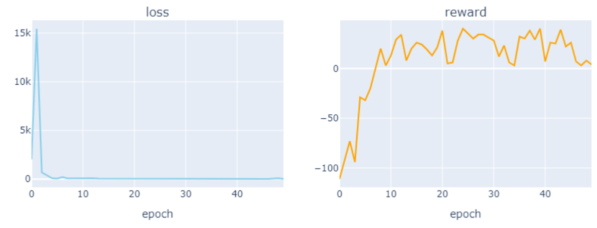


Figure 9. TCS

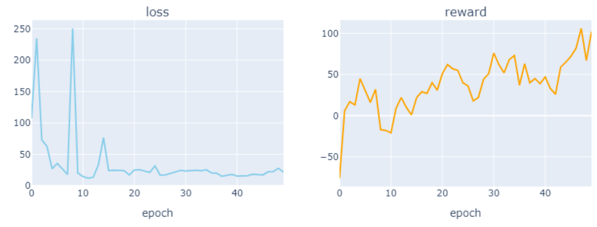


Figure 10. Tesla

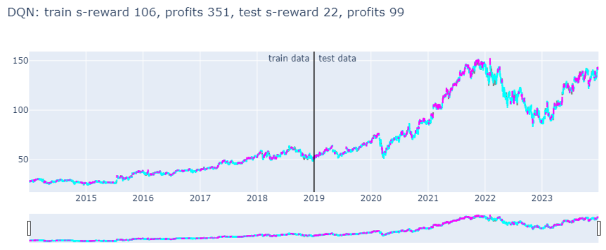


Figure 11. Google

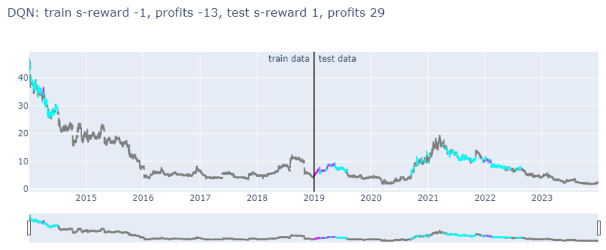


Figure 12. TCS

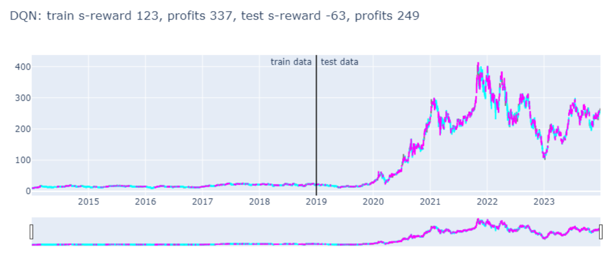


Figure 13. Tesla

DQN model (trained on 5 years ("2014-01-01" to "2019-01-01") data with 10 episodes and batch-size 10) for last 5 years ("2019-01-01" to "2024-01-01"), it made a profit of $2351.20 dollars. Refer to the Figure 14.

Run 2: We have initial investment of $1000 which when traded with DQN model (trained on 5 years ("2014-01-01" to "2019-01-01") data with 15 episodes and batch-size 20) for last 5 years ("2019-01-01" to "2024-01-01"), it made a profit of $2311.07 dollars. Refer to the Figure 15.

Which more or less is giving more than 200% profit in both the cases over 5 years time span.

8 Comparison of the Methods

The Z-score method performed well for chaotic stock data or when the stock price was increasing, as seen in figures 4 and 6. However, when the stock started to fall monotonically, this method failed to retain stocks and suffered losses, as illustrated in figure 5. Next, the Q-learning method did not appreciably improve over the Z-score method. Even though it did not incur high losses, it did not generate much profit either. It took minimal risks and rarely executed transactions, resulting in negligible profits throughout the time span.

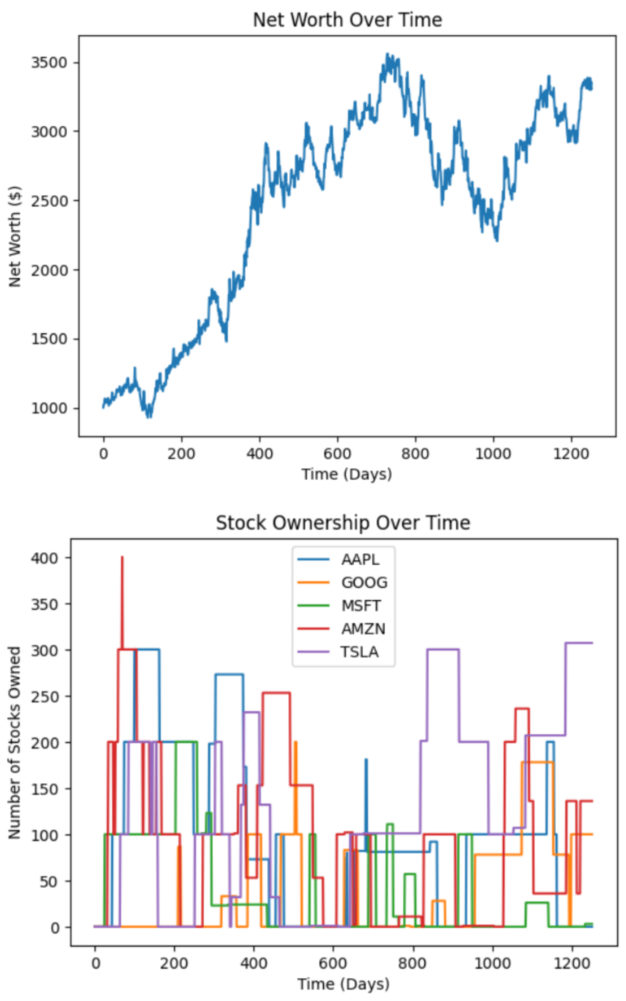
Incorporating neural networks into Q-learning improved stock prediction. The Deep Q-learning (DQN) method performed better than the previous methods, as shown in figures 8, 9, 10, 11, 12, and 13. We then changed the action space in the DQN method to predict the multi-stock trade system. The DQN method for multi-stock performed quite well, achieving more than 200% profit over five years, at least in our cases, as seen in figures 14 and 15.

9 Further Improvements

- Hyperparameter tuning can be used to improve the training of the model.
- Better DQN algorithm could have been used as in the duelling architecture, Prioritized Experience Replay, etc.
- ARIMA model could have also been used for time series prediction.
- News reports could have been added to the action space to take into consideration the environmental factors affecting the stock market.

10 Conclusion

We started by considering the general version of moving average, the Z-score method to predict the stock market. However it was performing poorly for monotonic decreasing data as can be seen in the TCS dataset. To improve the results, we considered using the Q-learning. Although we were getting profit by training our model using Q-learning, but it was very less. We sought to take help of neural nets and implemented the Deep Q-learning algorithm to predict the market. It performed substantially better than the Q-learning and Z-score method. Till then we were only considering trading a single stock. Finally we changed the action space to train the model using Deep Q-learning algorithm for multiple stocks. This model performed quite well compared to all the other models we tried. It gave more than 200% profit over 5 years as can be seen in figure 14 and figure 15.

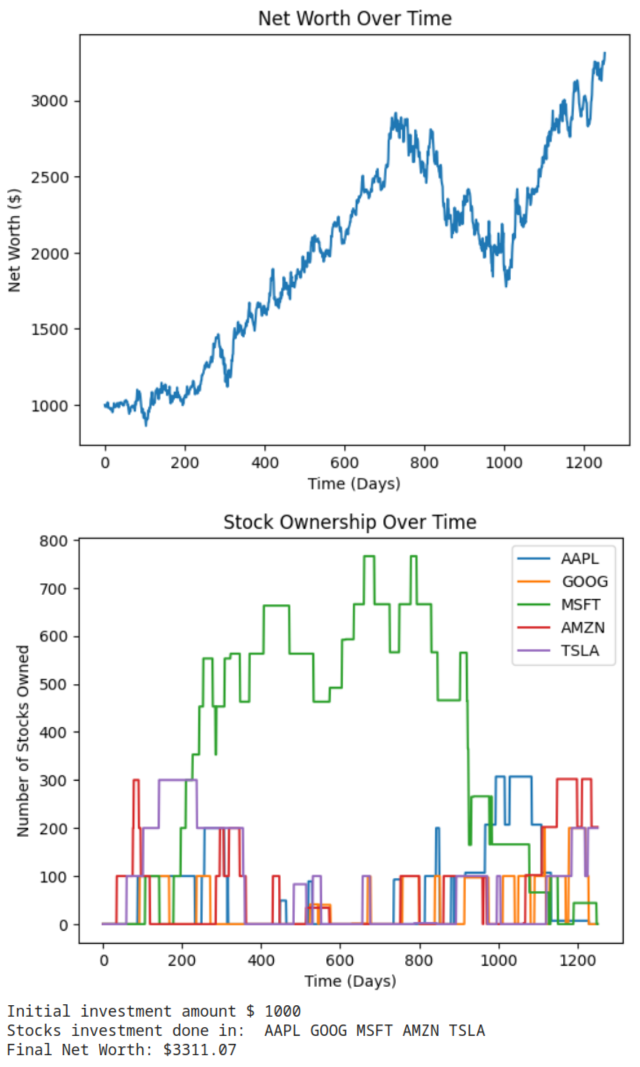


```
Initial investment amount $ 1000
Stocks investment done in:  AAPL GOOG MSFT AMZN TSLA
Final Net Worth: $3351.20
```

Figure 14. Multi-stock trading (run 1): DQN model trained on 10 episodes with batch-size of 10.

11 Author Contribution

- **Ankan Kar:** Conceptualization of Z-score Algorithm, Deep Q-Learning for Single and Multi Stock, Implementations of Z-score Algorithm, Deep Q-Learning for Single and Multi Stock, Validation, Visualization, Report Writing



```
Initial investment amount $ 1000
Stocks investment done in:  AAPL GOOG MSFT AMZN TSLA
Final Net Worth: $3311.07
```

Figure 15. Multi-stock trading (run 2): DQN model trained on 15 episodes with batch-size of 20.

- **Ishan Banerjee:** Conceptualization of Q-Learning, Variant of DQN, Deep Q-Learning for Single and Multi Stock, Implementations of Q-Learning, Deep Q-Learning for Single and Multi Stock, Validation, Visualization, Report Writing
- **Bijayan Ray:** Conceptualization of Z-score Algorithm, Deep Q-Learning for Single and Multi Stock, Implementations of Z-score Algorithm, Deep Q-Learning for Single and Multi Stock, Validation, Visualization, Report Writing
- **Praveen Kumar:** Conceptualization of Protfolio Management (as in presentation), Implementation of DQN, Visualization, Report Writing
- **Shreyam Banerjee:** Implenetation of Q-Learning and DQN, Validation, Visulaization, Report Writing.

12 Acknowledgement

We are thankful to Prof. K.V. Subrahmanyam for his unwavering guidance and support throughout our project. His enthusiasm for exploring new ideas greatly contributed to our progress and success.

References

[1] Prakash Basanna et al. "Relevance of Ratios in Z-score Model for Predicting Bankruptcy: Study of Nifty PSES". In: *International Journal of Mechanical Engineering* 7 (2022).

[2] Jagdish Bhagwan Chakole et al. "A Q-learning agent for automated trading in equity stock markets". In: *Expert Systems with Applications* 163 (2021), p. 113761.

[3] Yuming Li, Pin Ni, and Victor Chang. "Application of deep reinforcement learning in stock trading strategies and stock forecasting". In: *Computing* 102 (2020), pp. 1305–1322.

[4] Shuo Sun, Rundong Wang, and Bo An. "Reinforcement learning for quantitative trading". In: *ACM Transactions on Intelligent Systems and Technology* 14.3 (2023), pp. 1–29.