"Deep Q-Network (DQN) trader: Reinforcement learning for Automated trading"

॥वसुधैव कुटुम्बकम्॥

THESIS SUBMITTED TO

Symbiosis Institute of Geoinformatics

FOR PARTIAL FULFILLMENT OF THE M. Sc. DATA SCIENCE AND SPATIAL ANALYTICS DEGREE

By

Avesh Kumar Bhati

PRN: 22070243005

(Batch 2022-2024)

Symbiosis Institute of Geoinformatics
Symbiosis International (Deemed University)
5th Floor, Atur Centre, Gokhale Cross Road
Model Colony, Pune – 411016

CERTIFICATE

Certified that this thesis titled "**Deep Q-Network (DQN) Trader: Reinforcement learning for automated trading**" report is a bonafide work done by Mr. Avesh Kumar Bhati (22070243005) at Symbiosis Institute of Geoinformatics, under our supervision.

Supervisor, Internal

Name: Dr. Vidya Patkar

Symbiosis Institute of Geoinformatics

ACKNOWLEDGEMENT

In today's world of competition there is a race of existence in which only those succeed who have the will to come forward and accept challenges. Project acts as a bridge between theoretical concepts and the practical working. Keeping this in mind I started this project.

First of all, I would like to thank the Almighty without his grace this project could not have become a reality.

Next are my parents, whom have constantly supported me and encouraged me to complete this project.

I would also like to express my special thanks of gratitude to my internal guide "Vidya Patkar" for her able guidance and support in completing my project.

I would also like to thank Dr. T.P. Singh and Dr. Navendu Chaudhary for helping us with the project and internal guide allocation.

I would also like to thank all my teachers and data science friends associated with Symbiosis Institute of Geoinformatics who have always been helpful and encouraged me throughout the year.

To all those who have directly or indirectly contributed to this project, I am truly grateful for your involvement. Your Support has been instrumental in its success.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF EQUATION

LIST OF TABLES

ABBREVIATION LIST

1. **API** - Application Programming Interface
2. **LSTM** - Long Short-Term Memory
3. **MSE** - Mean Squared Error
4. **ReLU** - Rectified Linear Unit
5. **DQN** - Deep Q-Network
6. **NSEI** - Nifty (National Stock Exchange Fifty)
7. **AMZN** - Amazon
8. **GOOGL** - Google
9. **TSLA** - Tesla
10. **APPL** - Apple
11. **Rs**. - Indian Rupees (currency symbol)
12. **AI** - Artificial Intelligence

PREFACE

In the ever-evolving landscape of financial markets, the ability to make informed and accurate investment decisions is crucial. With advancements in artificial intelligence and machine learning, new opportunities have emerged to harness the power of technology in the world of trading. One such ground-breaking development is the Deep Q-Network (DQN) trader.

This project report explores the application of DQN, a reinforcement learning algorithm, in the domain of trading. DQN combines the concepts of deep neural networks and Q-learning to create an intelligent agent capable of learning and making decisions in complex environments. By training the agent on historical market data, it can learn patterns, adapt to changing market conditions, and optimize trading strategies to maximize returns.

The primary objective of this project is to investigate the performance and effectiveness of the DQN trader in real-world financial markets. By designing and implementing a trading system based on DQN, we aim to assess its ability to generate profitable trading decisions while considering risk management and market dynamics.

Throughout this report, we delve into the theoretical foundations of DQN, exploring its underlying principles and the key components that enable it to learn and improve over time. We discuss the challenges and considerations specific to implementing DQN in the context of trading, including data pre-processing, feature selection, and reward function design.

Furthermore, we present the methodology employed in this project, outlining the data collection process, the architecture of the DQN model, and the training and evaluation procedures. We carefully consider the choice of hyperparameters and provide a detailed analysis of the performance metrics used to assess the effectiveness of the DQN trader.

Importantly, this project report aims to be a comprehensive resource for both traders and researchers interested in exploring the application of DQN in financial markets. We provide a critical analysis of the results obtained, highlighting the strengths and limitations of the DQN trader, and offer insights into potential avenues for further research and improvement.

We would like to acknowledge the support and guidance of our project supervisor, who provided valuable input and helped shape the direction of this research. Additionally, we extend our gratitude to the open-source community for providing access to the necessary tools and libraries used in the implementation of the DQN trader.

It is our hope that this project report contributes to the growing body of knowledge on the use of reinforcement learning techniques in trading. By shedding light on the capabilities and challenges of the DQN trader, we aim to inspire further exploration and innovation in this exciting field.

Avesh Kumar Bhati

Introduction

In the ever-changing and highly desirable world of stock markets, traders strive to maximize their potential profits. To achieve this goal, researchers and professionals have sought to automate the trading process using advanced techniques like Data Science and Machine Learning. One such promising approach is the DQN Trader, a unique model that leverages the power of reinforcement learning to generate automated trading strategies based on historical market data. (Shah, 2021)

The DQN Trader takes its inspiration from the terms commonly used in the stock market: "bear" and "bull." A bear run signifies a market downturn, while a bull run represents a long-term rise in market prices. In the realm of intraday trading, where traders buy and sell financial instruments within the same trading day, these terms hold great significance. Intraday trading involves closing all positions before the market closes for the day, making it a type of securities speculation.

The significance of historical market data and current events cannot be ignored when formulating trading plans. While human traders have traditionally relied on their expertise and intuition, the advent of Data Science and Machine Learning has opened up opportunities for automating this laborious process. By harnessing the capabilities of these technologies, an automated trading technique can offer better estimates and timely suggestions, particularly beneficial for mutual funds and hedge funds seeking maximum profits.

However, creating an effective automated trading system is not without challenges. There is always a certain level of risk associated with achieving consistently profitable returns. It requires considerable effort to design a balanced and low-risk strategy that can benefit a wide range of individuals in the stock market.

Enter the DQN Trader, a cutting-edge solution that employs reinforcement learning agents to develop automated trading strategies based on historical data. The Deep Q-Network (DQN), a deep reinforcement learning algorithm, lies at the core of this model. By combining deep learning techniques with reinforcement learning principles, the DQN Trader aims to optimize trading decisions and maximize profitability. (Shah, 2021)

The DQN Trader's training process involves iterative interactions with the market environment. Through trial and error, the model learns from past market data, identifies patterns, and seeks to make informed trading decisions. It optimizes a reward function to maximize profits or minimize losses over time, ultimately refining its decision-making capabilities.

To create a successful DQN Trader system, various aspects must be carefully considered. This includes selecting appropriate input features, designing an optimal network architecture, defining an effective reward function, and addressing transaction costs and market dynamics. These considerations contribute to developing a robust and efficient trading system.

PROBLEM STATEMENT:

In recent years the technical analysis attracts a lot of attention due to a simple fact that we have enough information just by looking to the historical stock market, which is public and well-organized, compared to the fundamental analysis where we need to analyze unstructured dataset.

Compared to the supervised learning techniques and at a certain level, un- supervised learning algorithms, are widely used in stock price prediction, to the best of our knowledge the reinforcement learning for stock price prediction has not yet received enough support as it should be. The main issue of supervised learning algorithms is that they are not adequate to deal with time-delayed reward . In other words, supervised learning algorithms focus only on the accuracy of the prediction at the moment without considering the delayed penalty or reward. Furthermore, most supervised machine learning algorithms can only provide action recommendation on particular stocks, using reinforcement learning can lead us directly to the decision making step, i.e. to decide how to buy, hold or sell any stock.

So, in this report we will discuss about the Machine learning techniques that were earlier used for predicting the Stock price.

PROPOSING:

In order to solve this issue, we suggest an agent model that will automatically make the neces sary decisions regarding whether to buy, hold, or sell stocks.

BACKGROUND OF THE PROJECT:

The complicated and volatile financial markets have always attracted academics and traders looking for lucrative investment opportunities. Traditional trading methods are labour-intensive, subject to human bias, and primarily reliant on human intuition, analysis, and manual decision-making. Automated trading systems, which can process enormous volumes of data and make wise trading decisions in real-time thanks to advances in AI and machine learning, have become a potential alternative.

In the field of AI, Deep Q-Networks (DQNs) have attracted a lot of interest, notably in the area of reinforcement learning. DQNs are neural network architectures that incorporate reinforcement learning algorithms and deep learning techniques. This enables the model to learn the best course of action by interacting with the environment and receiving feedback in the form of incentives or Penalties.

OBJECTIVES:

- Automate Trading Strategy: The DQN Trader's main goal is to automate the procedure of coming up with trading strategies. By using a model that can make trading decisions automatically based on historical market data, it seeks to replace manual decision-making.

- Maximise Profitability: By making the best trading decisions possible, the DQN Trader aims to maximise profitability. The programme attempts to find patterns and trends that could result in winning trades by using reinforcement learning techniques to learn from historical data.

- Reduce Risk: Reducing risk is another goal of the DQN Trader. The model seeks to generate trading decisions that strike a balance between prospective gains and potential losses by including a reward function that takes into account both profitability and risk management.

- Learn from previous Market Data: The DQN Trader builds its trading methods by studying previous market data. The model seeks to extract useful information that can direct its decision-making process by examining previous price movements, trends, and other pertinent factors.

- Enhance Trading Decisions: The DQN Trader seeks to enhance its trading decisions through the iterative training process. The model aims to enhance performance over time by assessing and changing its tactics in response to input in the form of rewards.

- Offer a Low-Risk Trading Approach: The DQN Trader strives to create a well-balanced, low-risk trading approach that may be utilised by a variety of traders. Profitability is a goal, but it's also important to reduce risk. The approach aims to balance minimising possible losses with profit maximisation.

Expected Outcomes:

- Creation of a fully functional trading system with the ability to conduct transactions on its own based on forecasts provided by the DQN model is the project's main objective.

- Enhanced trading performance compared to traditional tactics is what the project intends to achieve by utilising AI and reinforcement learning approaches, with the end goal being consistent profitability.

- The DQN-based system should be able to modify its trading strategy in response to changing market conditions so that it can successfully seize opportunities and reduce risks.

LITERATURE REVIEW

Automated stock trading, also known as algorithmic trading or algo-trading, is the use of computer programs to execute trading decisions based on predefined rules and parameters. The use of algorithms and artificial intelligence (AI) in stock trading has become increasingly popular in recent years due to its ability to analyse large amounts of data and make rapid decisions based on market trends and patterns. In this literature review, we will examine several studies that have explored the use of automated stock trading.

The combined findings from various papers indicates that deep reinforcement learning and machine learning techniques offers promising results in automated stock trading. These approaches have the potential to improve trading performance metrics such as cumulative return, Sharpe ratio, and adaptability to changing market conditions and can create more opportunities in the same field. (Hongyang Yang1, 2021)

Ensemble strategies using deep reinforcement learning have been shown to outperform baseline strategies in terms of cumulative return, Sharpe ratio, and maximum drawdown. The FinRL library, based on deep reinforcement learning, has also demonstrated significant outperformance compared to baseline strategies.

The application of deep reinforcement learning in the Chinese stock market has led to improved trading performance, while incorporating stock-specific news analysis has resulted in automated trading systems that outperform baseline strategies.

Furthermore, synchronous deep reinforcement learning models and Q-learning agents have shown higher returns and adaptability to changing market conditions. Machine learning techniques, including decision trees, performance weighted random forests, and logical clustering algorithms, have also exhibited superior performance compared to traditional trading strategies.

Additionally, the integration of optimization methods, portfolio optimization, and sentiment analysis has significantly improved the performance of automated trading systems. Studies have also explored the role of materiality and high-frequency trading in the development of automated trading systems.

Although these combined findings suggest the potential benefits of using deep reinforcement learning and machine learning in automated stock trading, it is important to note that further research and live trading environment testing are necessary to evaluate the practical effectiveness and potential risks associated with these approaches.

Sr. No.	**Author(s), Journal**	**Title**	**Year Published**	**DATA SAMPLES**	**Purpose**	**Findings**
1	H. Yang, X. Y. Liu, S. Zhong, and A. Walid	Deep reinforcement learning for automated stock trading: An ensemble strategy	2020	S&P Histroical	The purpose of the paper is to propose a new approach to automated stock trading using deep reinforcement learning and an ensemble strategy. The authors aim to address the challenges associated with stock trading, such as the dynamic and complex nature of the stock market and the difficulty of accurately predicting stock prices.	The findings of the paper indicate that the proposed ensemble strategy for automated stock trading using deep reinforcement learning outperforms several baseline strategie
2	X. Y. Liu, H. Yang, Q. Chen, R. Zhang, L. Yang, and H. Li	A deep reinforcement learning library for automated stock trading in quantitative finance	2020	The paper does not describe specific data samples collected. Instead, the authors use three popular datasets	The purpose of the paper is to present a deep reinforcement learning (DRL) library called FinRL, designed for automated stock trading in quantitative finance.	

3	L. Chen and Q. Gao	Application of deep reinforcement learning on automated stock trading	2019	The paper does not describe specific data samples collected. Instead, the authors use historical stock data	The purpose of the paper is to apply deep reinforcement learning (DRL) to automated stock trading and investigate its performance in the Chinese stock market..	The paper presents a DRL-based automated trading system that is trained on historical stock data from the Chinese stock market.
4	J. Zou, H. Cao, L. Liu, Y. Lin, E. Abbasnejad, and H. Zhang	Astock: A new dataset and automated stock trading based on stock-specific news analyzing model	2022	The Astock dataset is created by collecting stock-specific news and stock price data for 1696 listed companies in the Chinese stock market	The purpose of the paper is to introduce a new dataset called Astock, which contains stock-specific news and stock price data for the Chinese stock market. specific news to predict future stock price movements and make profitable trades.	The paper presents an automated stock trading system that uses a stock-specific news analyzing model to predict future stock price movements and make trades.

5	Ramy AbdelKawy, Walaa M. Abdelmoez, and Ahmed Shoukry	A synchronous deep reinforcement learning model for automated multi-stock trading	2021	The paper uses historical stock price data for a number of companies, including Apple, Microsoft.	The purpose of this paper is to propose a synchronous deep reinforcement learning model for automated multi-stock trading. The authors aim to develop a model that can learn to trade multiple stocks	The authors develop a synchronous deep reinforcement learning model that can simultaneously trade multiple stocks.
6	Bin Huang, Yong Huan, Lin Da Xu, Lina Zheng	Automated trading systems statistical and machine learning methods and hardware implementation: a survey	2019	The paper does not involve any data collection or analysis.	The purpose of this paper is to provide a comprehensive survey of automated trading systems, including the statistical and machine learning methods used, and the hardware implementations that enable high-speed trading.	The authors provide an overview of the different types of automated trading systems, including rule-based systems.
7	Jayant B. Chakole, Manish S. Kolhe, Gajanan D. Mahapurush, and Nitin S. Mahalle	A Q-learning agent for automated trading in equity stock markets	2021	The paper uses historical stock price data for a number of companies	The purpose of this paper is to develop and evaluate a Q-learning agent for automated trading in equity stock markets.	The authors develop a Q-learning agent that can learn to make profitable.

8	Mingyu Kong and Jaewoo So	Empirical Analysis of Automated Stock Trading Using Deep Reinforceme nt Learning	2023	The paper uses historical stock price data for a number of companies , including Apple, Amazon, Facebook, and Google.	The purpose of this paper is to empirically analyze the effectiveness of using deep reinforcement learning for automated stock trading.	The authors develop a deep reinforceme nt learning-based trading agent that can learn to make profitable trades in the stock market. The agent uses a combinatio n of technical indicators and market sentiment data to make trading decisions.
9	Adam Booth, Enrico Gerding, and Frank McGroarty	Automated trading with performance weighted random forests and seasonality	2014	The paper uses historical stock price data for a number of companies Apple.	The purpose of this paper is to develop and evaluate a machine learning approach for automated trading that accounts for seasonality in data.	The authors develop a machine learning-based trading system. conditions,

10	Mohammed Alsulmi and Nada Al-Shahrani	Machine Learning-Based Decision-Making for Stock Trading: Case Study for Automated Trading in Saudi Stock Exchange	2022	The paper uses historical stock price data for companies listed in the Saudi Stock Exchange.	The purpose of this paper is to develop and evaluate a machine learning approach for automated trading in the Saudi Stock Exchange. The authors aim to investigate whether their approach can achieve higher returns and outperform traditional trading strategies.	The authors develop a machine learning-based trading system that uses a decision tree algorithm to predict stock prices.
11	Aleksandra Rakićević, Vladimir Simeunović, Branislav Petrović, and Siniša Milić	An automated system for stock market trading based on logical clustering	2018	The paper uses historical stock price data from the New York Stock Exchange (NYSE) for the period between 2013 and 2017.	The purpose of this paper is to develop an automated trading system that uses logical clustering algorithms to identify patterns in stock market data and make trading decisions based on those patterns.	The authors develop an automated trading system that uses logical clustering algorithms to group stocks based on their similarities

12	Andrea Bigiotti and Alfredo Navarra	Optimizing automated trading systems	2019	The paper does not collect any data samples but instead focuses on reviewing and analyzing existing literature on automated trading systems and optimization methods.	The purpose of this paper is to explore various methods for optimizing automated trading systems, with a focus on machine learning techniques and portfolio optimization	The authors review and compare various machine learning techniques for predicting stock prices and optimizing trading strategies, including support vector machines, artificial neural networks.
13	Donald MacKenzie	A material political economy: Automated trading desk and price prediction in high-frequency trading	2017	The paper does not collect any data samples but instead focuses on a case study of the development.	The purpose of this paper is to explore the role of materiality in high-frequency trading and the development of automated trading systems, with a focus on the Automated Trading Desk (ATD) and its efforts to predict stock prices	The author argues that the success of the ATD and other high-frequency trading firms is due in part to their ability.

14	Y Ansari, S Yasmin, S Naz, H Zaffar, Z Ali, J Moon	A Deep Reinforcement Learning-Based Decision Support System for Automated Stock Market Trading	2022	Not explicitly stated	To develop a decision support system for automated stock market trading using deep reinforcement learning	The authors demonstrated the effectiveness of their proposed system by conducting experiments on real-world stock market data and comparing the results.
15	Q. Huang, J. Yang, X. Feng, W. Li, and K. Li	Automated trading point forecasting based on bicluster mining and fuzzy inference	2019	The authors used historical data of six real stocks from the Chinese stock market to validate the proposed method.	The paper presents a novel approach for point forecasting in automated trading using bicluster mining and fuzzy inference. The aim is to accurately predict trading points and achieve higher returns.	The experimental results show that the proposed method outperforms traditional time series forecasting methods and other state-of-the-art machine learning models.

16	B. Taylor, M. Kim, A. Choi	Automated stock trading algorithm using neural networks	2014	Not explicitly stated	The purpose of this paper is to propose an automated stock trading algorithm using neural networks. The paper aims to evaluate the effectiveness of the algorithm in generating profit in the stock market.	The paper presents a neural network-based automated trading algorithm that uses technical indicators as input features to predict stock prices. e of the algorithm.
17	S Bajpai	Application of deep reinforcement learning for Indian stock trading automation	2021	Not specified.	To apply deep reinforcement learning to automate stock trading in the Indian stock market.	The study found that the proposed deep reinforcement learning algorithm outperformed a traditional buy and hold strategy.

18	TR Silva, AW Li, EO Pamplona	Automated trading system for stock index using LSTM neural networks and risk management	2020	Not specified	To develop an automated trading system for stock index using LSTM neural networks and risk management	The proposed system achieved better returns compared to a buy-and-hold strategy, and also outperformed a baseline LSTM-

TABLE 1: Summary of Literature review

METHODOLOGY

We use a common approach in validating time-series data, which is called the walk-forward validation. In this experimental scenario, the semantic linking between the observation at time t and t +1 is taken into account to compose the same bunch of training, validation and test sets. This is different with respect to common cross-validation approaches like the leave-one-out cross validation or the k-fold cross validation, where data are randomly sampled in different folds, no matter when they were acquired. Such an approach is quite biased when applied to time series prediction, as features from late past and early future can be mixed in the same fold of data when using that strategy. The walk forward validation better fits this scenario, since the considered folds are temporally split and processed as training, validation and testing data.

A normal Flowchart for the methodology is represented below:

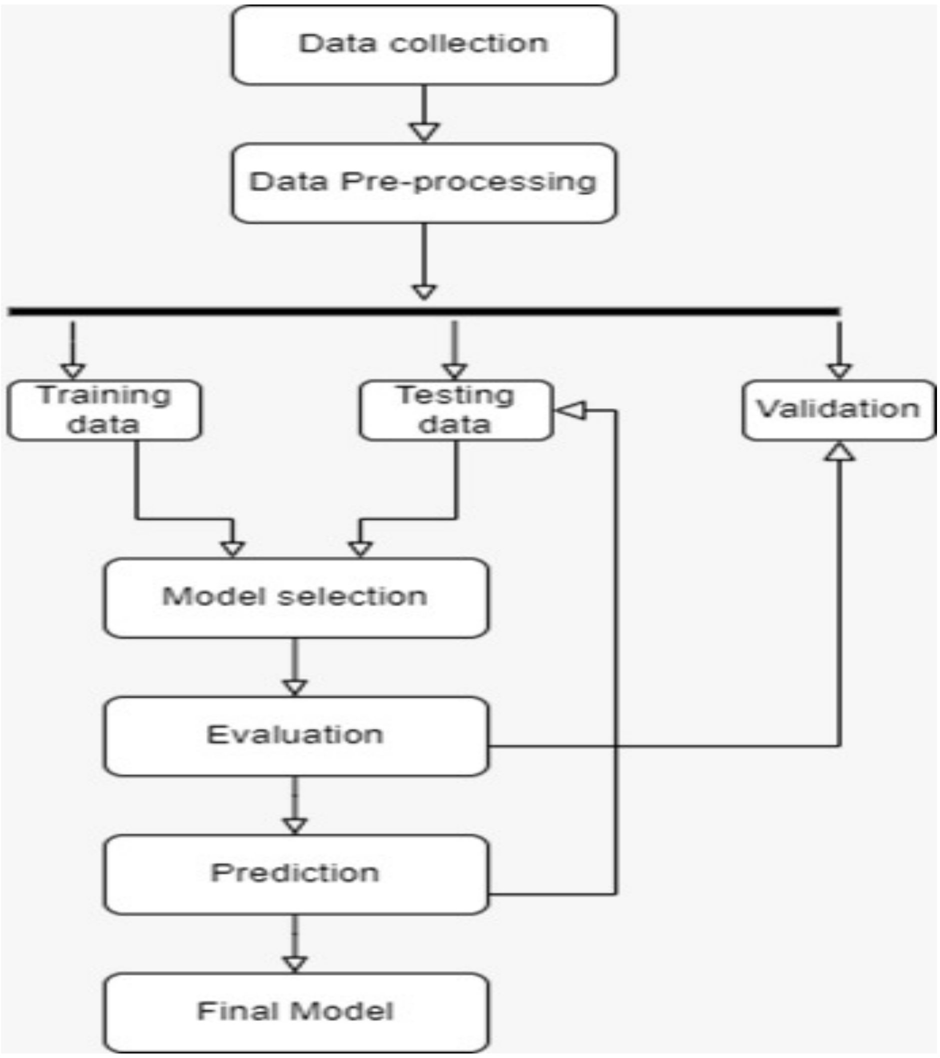


Figure 1 – Methodology Flowchart

- **Data Collection**

The data collection method used for this project is secondary data collection, which involves collecting data from existing sources. In this case, the dataset that is used for the study of Automated Stock Trading is collected from the different financial databases or API's like Yahoo finance , Up stocks .The dataset consist of different columns like Date, Open, High, Low, Close, Adj Close, Volume.

The dataset is a historical data containing from 2018 to 2023 of different Companies like Googl, Amazon, Tesla. Also there is a dataset of Nifty from the year 2018 to 2023.

Since the data was collected from the financial databases so we can definitely trust the data source and assumed it to be the cleaned data and up to dated.

The different columns in the dataset represents:

- Date: The date on which the stock was traded.
- Open: The price at which the stock opened for trading on that day.
- High: The highest price the stock traded for during the day.
- Low: The lowest price the stock traded for during the day.
- Close: The price at which the stock closed for trading on that day.
- Adj Close: The adjusted closing price takes into account any corporate actions that occurred during the trading day, such as stock splits or dividends. It is the closing price adjusted for these actions and is considered to be a more accurate reflection of the stock's true value.
- Volume: The number of shares of the stock that were traded on that day.

Data Pre-Processing:

Since the data was collected from the historical stock data so the data we got is almost cleaned for all the datasets. We had 4-5 datasets of different stocks like 'APPL', 'AMAZN', 'GOOGL', 'TSLA' and 'NIFTY'.

So the first step we did was of merging all the datasets of all the companies into one data frame.

Then we looked for the null values in the dataset but since the dataset was from financial databases so there were no null values however there were 2-3 incorrect values so we used the mean of the column in those places.

After that we checked the duplicate values in the dataset but there were no duplicate values present in the dataset so we can move on the next step which was to look at the datatypes of all the columns, The 'Date' column should be of datetime type. All other columns have the correct data types.
After the merging all the datasets we added another column named 'Company' so at the end we had 1313 entries with 8 columns for the final dataset.

For DQN Trader the Data was already preprocessed so we have not done anything specific. We just simply used the data directly.

MODEL BUILDING

STOCK PRICE PREDICTION (USING SUPERVISED ALGORITHM)

We have used three Machine Learning Models for predicting the Stock Price:

1. LSTM
2. Linear Regression
3. XGBOOST

LSTM:

The report presents an LSTM (Long Short-Term Memory) model for stock price prediction on multiple companies. The study begins with data pre-processing, where the dataset is divided into training and test sets for each company. The data is then scaled using the MinMaxScaler to normalize the values between 0 and 1.
The LSTM model architecture comprises two LSTM layers with 13 units each, along with dropout layers for regularization. The model is trained using the Adam optimizer and mean squared error (MSE) loss. The training process includes 30 epochs, a batch size of 10, and a validation split of 20% to ensure optimal model performance.
Following training, the model predicts stock prices for the test data. The predicted values are transformed back to their original scale using the inverse transform method of the MinMaxScaler.
(Kong, 2021)

So now to solve our main problem which was the complexity of predicting The evaluation of the model's performance is measured using the R-squared score, which assesses how well the predicted values align with the actual values.
To visualize the results, the paper includes plots that compare the actual stock prices and the predicted prices for each company. The plots are presented in separate subplots, with the company name clearly indicated.

LINEAR REGRESSION:
Linear is not one of the famous Techniques in predicting Stock Prices as it may not capture the complexities and non-linear relationships in stock market data.
However Linear regression is a simple and widely-used technique for modelling the relationship between dependent and independent variables.
It assumes a linear relationship between the input features (previous stock prices) and the target variable (next stock price).
The code prepares the data by iterating over each unique company in the dataset.
Historical stock prices (represented by 'Adj Close') are extracted for each company.

XGBOOST:
Stock price prediction is a challenging task that requires sophisticated machine learning techniques. XGBoost (Extreme Gradient Boosting) has emerged as a powerful algorithm for accurate and robust predictions in various domains, including finance.
XGBoost proves to be a valuable tool for stock price prediction in the provided code. By utilizing its ensemble learning approach and advanced techniques, XGBoost demonstrates its capability to capture nonlinear relationships and handle challenges in financial data however we found that the model is giving different Accuracy for different company stock price.
We uses different hyperparameters during our model building like max_depth,n_estimators, seed, gamma, learning rate .

- gamma [default=0, alias: min_split_loss] parameter
  The minimal loss reduction needed to partition the tree's leaf node farther. The algorithm becomes increasingly conservative as gamma increases.
- max_depth [default=6]
  Increasing this amount will complicate the model and increase the likelihood of overfitting. No depth cap is indicated by 0, or 0. Be aware that when training a deep

tree, XGBoost rapidly eats memory. No value can be zero when using the precise tree approach.

- seed (int) - Seed for creating the folds.

DQN TRADER

DQN Trader is a term used to describe a trading algorithm or system that uses a Deep Q-Network (DQN) to make trading choices. DQN is a reinforcement learning method that blends Q-learning, a type of value-based reinforcement learning, with deep neural networks. (Foy, 2021)

The typical context in which DQN Trader functions is one in which it gets historical market data and strives to learn a trading strategy that will maximise its cumulative profit over time. The DQN model produces actions, such purchasing, selling, or keeping particular financial assets, based on previous price and volume data as input.

Notably, designing a successful DQN Trader necessitates careful consideration of a number of variables, including the selection of the state representation, incentive design, hyperparameter tuning, and appropriate handling of transaction costs and market constraints.

Overall, DQN Trader leverages the power of deep reinforcement learning to make trading decisions based on historical market data, aiming to learn an optimal trading policy that maximizes long-term returns.

Stock prices we are proposing DQN trader where we will create an agent which will take all the decisions of buying, hold and sell and to maximize our profits in our portfolio.

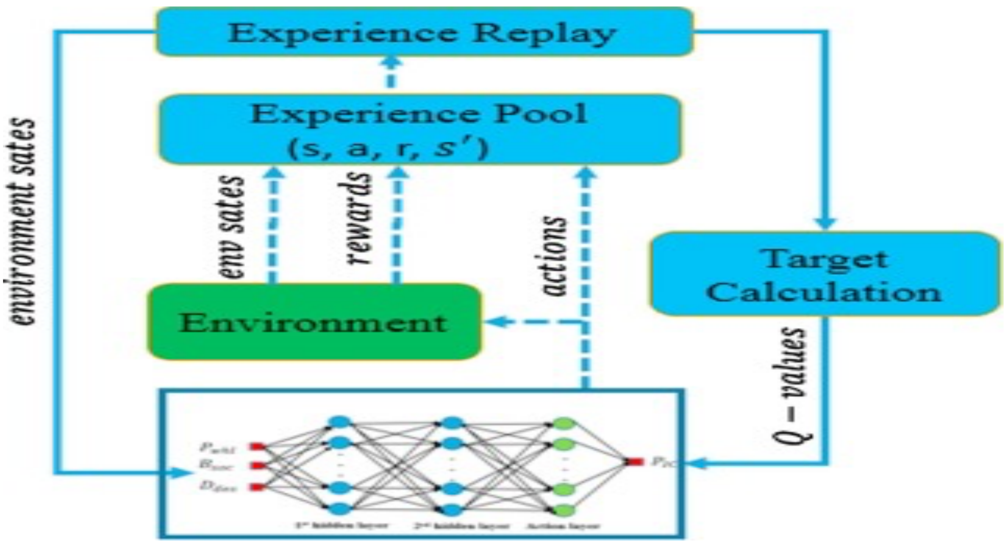


FIGURE 2 Working of DQN Trader

AGENT CLASS:

Due to its capacity to learn the best decision-making strategies in dynamic and uncertain contexts, reinforcement learning has attracted considerable attention in the field of financial trading. The implementation of a reinforcement learning agent that learns to make trading decisions based on the condition of the market is the main topic of this research. By using lessons from the past and continually enhancing its decision-making skills, the agent seeks to maximise its revenues over a certain time period.

Agent Class and Initialization:

The agent is created using the Python class "Agent." The state space size, the maximum number of time steps, the initial amount of money, and an optional pre-trained model are all initialised. These parameters provide the necessary context for the agent to operate effectively.

(Yawei Li, 2022)

Model Creation:

The Keras library's neural network model is used by the agent. The model has three dense layers, the first two of which add nonlinearity using the rectified linear unit (ReLU) activation function. To generate Q-values for each potential action, the output layer uses a linear activation function. The Adam optimizer and mean squared error (MSE) loss function used in the model's construction make it possible to train and improve the agent's decision-making skills quickly.

Action Selection:

The agent's act() method chooses the appropriate course of action based on the status of the market. When the agent is in training mode, it randomly selects an action with a probability set by the exploration rate (epsilon) to investigate the surroundings. As an alternative, the agent chooses the action with the highest Q-value predicted by the neural network model if it is in evaluation mode or if exploration is not picked at random.

Experience Replay:

The agent's learning process heavily relies on experience replay. Using a sample from its memory, the agent can learn from a group of prior encounters by using the expReplay() method. The target Q-value is determined using the Bellman equation, taking into account both the discounted maximum Q-value of the subsequent state and the immediate reward.

After that, the neural network model is trained to reduce the mean squared error between the target and predicted Q-values, which helps the agent's decision-making abilities to converge. Bellman Equation:

$$\boxed{v_{\pi}(s) = \mathbb{E}_{\pi}\left[R_{t+1} + \gamma v_{\pi}(s_{t+1}) \mid s_t = s \right]}$$

Bellman equation

(Taylan Kabbani, 2022)

Equation 1Bellman Equation

Where eqn of s represents a State value function.

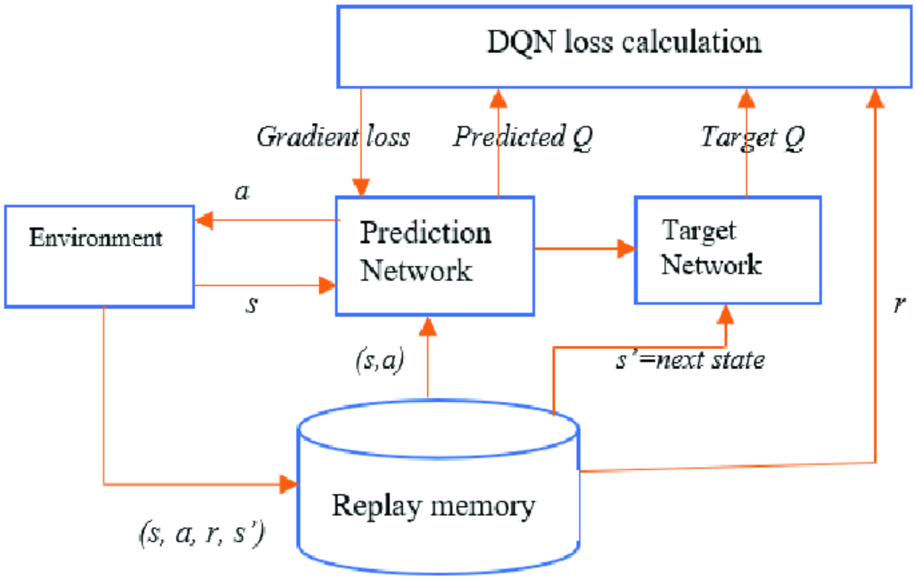


Fig2.1. Loss Calculation of DQN Trader

This is the loss calculation of DQN Trader for Improving the next state.

Epsilon Decay:

The agent gradually lowers its exploration rate (epsilon) over time in order to strike a balance between exploration and exploitation. The epsilon value is multiplied by a decay factor (epsilon_decay) after each experience replay step to make sure that the agent gradually relies more on its learnt policy and makes use of the training-related information.

FUNCTIONS OF DQN TRADER

- BUY(Agent, Price)
- SELL
- HOLD

Buy(agent, price):

The agent can carry out a buy action in the trading environment thanks to this capability. It requires two inputs: the agent object and the most recent stock price. Based on the agent's remaining budget (money) and the remaining time steps (max_t - transactions), the function determines how much money should be invested. It adds the matching stock quantity to the agent's inventory and deducts the computed amount from the money attribute of the agent. If a purchase is successful, the function returns 0, otherwise it returns -1 if it is unsuccessful because of financial limitations or exceeding the maximum transaction limit.

- formatPrice(n)- This function formats the numeric number n that is supplied as a string that represents a currency value. If the value is negative, denoting a loss, it includes a prefix of "Rs." and a negative sign. The function makes sure that prices are formatted consistently and readable.
- sigmoid(x)- The input value x is subjected to the sigmoid function, which employs the sigmoid activation function. The chance of a favourable outcome is represented by a value between 0 and 1 that is returned. The trading agent uses sigmoid to normalise input data and make sure that values fall within an appropriate range for processing.

Sell(agent, price):

The sale function allows the agent to execute a sell action in the trading environment. It requires two inputs: the agent object and the most recent stock price. The function determines the total value of the stocks based on the current price and adds it to the agent's money attribute if the agent's inventory is not empty (indicating stocks are owned). The function also determines the reward by comparing the agent's final cash balance with the higher of the original budget or the money_before attribute. The reward is a representation of the gain or loss from the sale activity. The function also refreshes the money_before attribute and resets the agent's transactions and inventory. The calculated reward is returned.

Get_state(agent, data):

This function uses the incoming data and the agent object to prepare the state representation for the agent. It accepts as inputs an array of data along with the agent object. The function normalises the data values by applying feature scaling using the StandardScaler from the scikit-learn module. The sigmoid function is then applied element-by-element to guarantee that all values are between 0 and 1.
(Salvatore Carta, 2021)

The function adds two further aspects to the data: the agent's current transaction progress (transactions / max_t) and the amount of money now in the agent's possession relative to the starting budget (money / (money_before + 1). The function returns a NumPy array that represents the final state.

$$
\begin{array}{cc}
 & \textbf{Action} \\
\textbf{State} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array}
\end{array}
$$

$$
R = \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
\begin{bmatrix}
-1 & -1 & -1 & -1 & 0 & -1 \\
-1 & -1 & -1 & 0 & -1 & 100 \\
-1 & -1 & -1 & 0 & -1 & -1 \\
-1 & 0 & 0 & -1 & 0 & -1 \\
0 & -1 & -1 & 0 & -1 & 100 \\
-1 & 0 & -1 & -1 & 0 & 100
\end{bmatrix}
$$

$$
Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}
$$

(Yang, 2020)

Equation 2 Q Learning

The above picture is taken from Analytics Vidhya Website which describes how Q learning tells the agent which Action to take.

MODEL TRAINING

So we have divided the model training into three parts namely Initialization, Training loop and Time Step loop

Initialization:

- agent.is_eval is set to False, indicating that the agent is in training mode.
- episode_count is set to 12, representing the number of training episodes.
- l is set to the length of the data.
- agent.max_t is set to 5, indicating the maximum number of time steps per episode.
- batch_size is set to 32, representing the size of the mini-batches used for experience replay.

Training Loop:

- The loop iterates over each episode.
- The current episode number is printed.
- The initial state is obtained using the get_state function, which takes a window of data as input.
- agent.money is set to Money (presumably a predefined starting capital).
- A dictionary code is defined to map actions to corresponding codes ('b', 'r', 'g').
- Lists decisions and actions are initialized to store the agent's decisions and action counts, respectively.
- agent.inventory and agent.transactions are set to 0, representing the current inventory and transaction count of the agent.

Time Step Loop:

- The loop iterates over each time step within the window size to l-1.
- The agent chooses an action using the act method, which returns the chosen action and the corresponding action code.
- The chosen action code is appended to the decisions list.
- Based on the chosen action, the agent updates the actions list and calculates the reward.
- If the action is 0, the reward is based on the negative count of previous 'buy' actions.
- If the action is 1 ('buy'), the buy function is called to perform a buying action and calculate the reward.

- If the action is 2 ('sell'), the sell function is called to perform a selling action and calculate the reward.
- The next state is obtained using the get_state function for the next window of data.
- The variable done is set to True if it is the last time step, indicating the end of the episode.
- The (state, action, reward, next_state, done) tuple is appended to the agent's memory for experience replay.
- If the episode is done, the agent's final profit is calculated and printed.
- If the agent's memory exceeds the batch size, the agent performs experience replay using the expReplay method.
- The current time step, agent's money, chosen action, and action code are printed.

TESTING

The model was tested on various stock companies like Apple, Tesla, Nifty ,Google, Amazon.
The model was given the Money and Max transaction as the inputs .
The agent is in evaluation mode, as evidenced by the value of agent.is_eval being set to True.
The 'Close' column from the stock data was transformed to a NumPy array.
The get_state function, which accepts a window of stock data as input, is used to determine the initial state.
The judgements made by the agent at each time step are recorded in a dictionary called decisions.

RESULTS AND DISCUSSION

COMPARISON OF MODELS FOR 3 COMPANIES:

MODELS	AMZN	NSEI	GOOG
LSTM	90.8	97.2	96.6
LINEAR REGRESSION	96.36	98.71	97.99
XGBOOST	95.51	97.90	96.96

Table 2 - Model Accuracy Comparison

The table compares the performance of different models for predicting stock prices for three companies: Amazon (AMZN), National Stock Exchange of India (NSEI), and Google (GOOG). The models evaluated are LSTM (Long Short-Term Memory), Linear Regression, and XGBoost (Extreme Gradient Boosting).

The results indicate the following:

LSTM: The LSTM model achieved an accuracy of 90.8% for predicting AMZN stock prices, 97.20% for NSEI, and 96.60% for GOOG. It performed relatively well for predicting AMZN and GOOG stocks but had better accuracy for NSEI.

Linear Regression: The Linear Regression model yielded an accuracy of 96.36% for AMZN, 98.71% for NSEI, and 97.99% for GOOG. This model performed consistently well across all three companies, with relatively high accuracy for NSEI.

XGBoost: The XGBoost model achieved an accuracy of 95.51% for AMZN, 97.90% for NSEI, and 96.96% for GOOG. It performed well for predicting AMZN and GOOG stocks but had slight better accuracy for NSEI, similar to the LSTM model.

Overall, the Linear Regression model demonstrated the highest accuracy for predicting stock prices, especially for NSEI. It outperformed both LSTM and XGBoost models in terms of accuracy across the evaluated companies.

LSTM:

```
3/3 [==============================] - 0s 3ms/step
0.7876576526756704
```

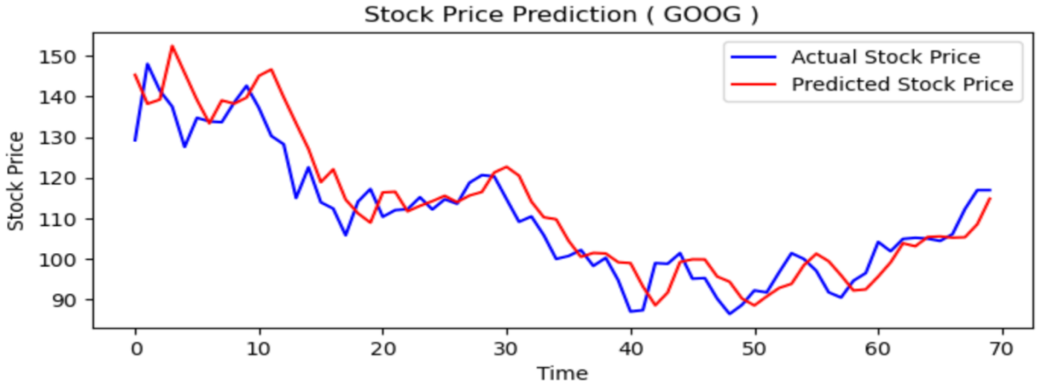


Figure 1 - Lstm (Goog)

```
3/3 [==============================] - 0s 1ms/step
0.7760423678712113
```

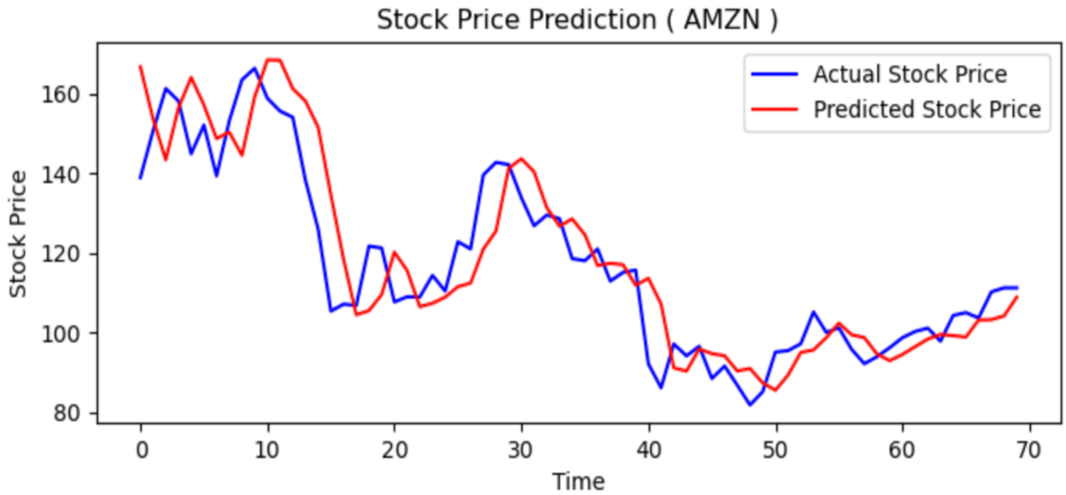


Figure 4 - Lstm(Amazon)

```
3/3 [==============================] - 0s 1ms/step
0.6338755280413229
```

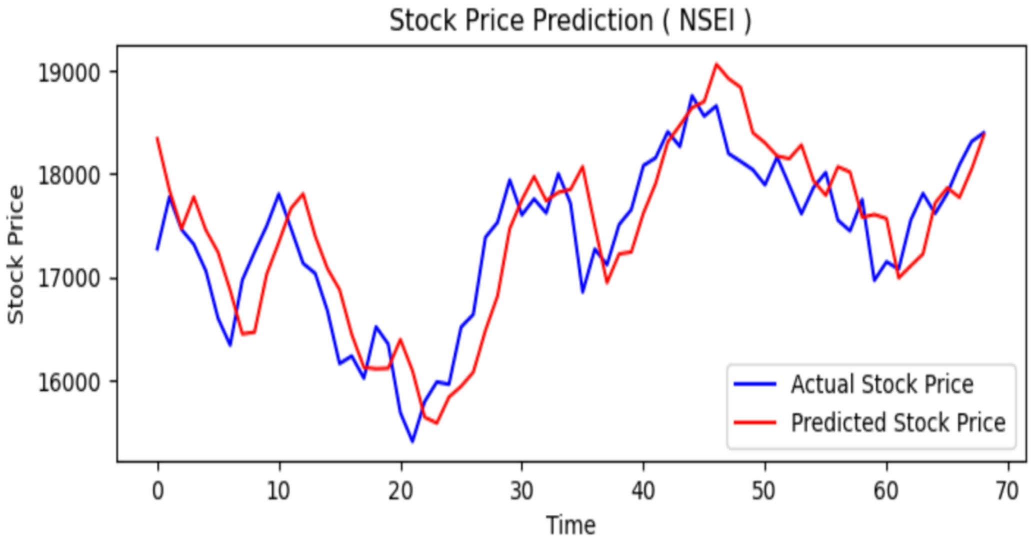


Figure 5 – Lstm (Nifty)

LINEAR REGRESSION:

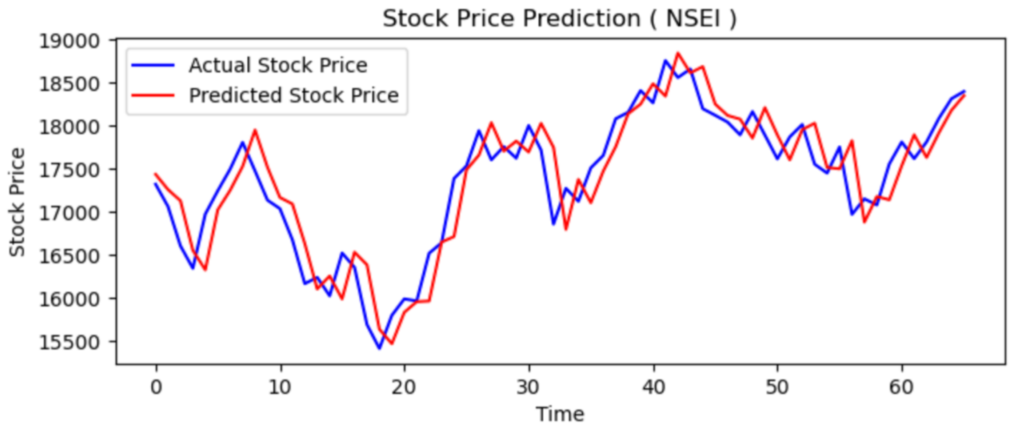


Figure 6 -Linear Regression(Nifty)

```
0.8661198051948157
```

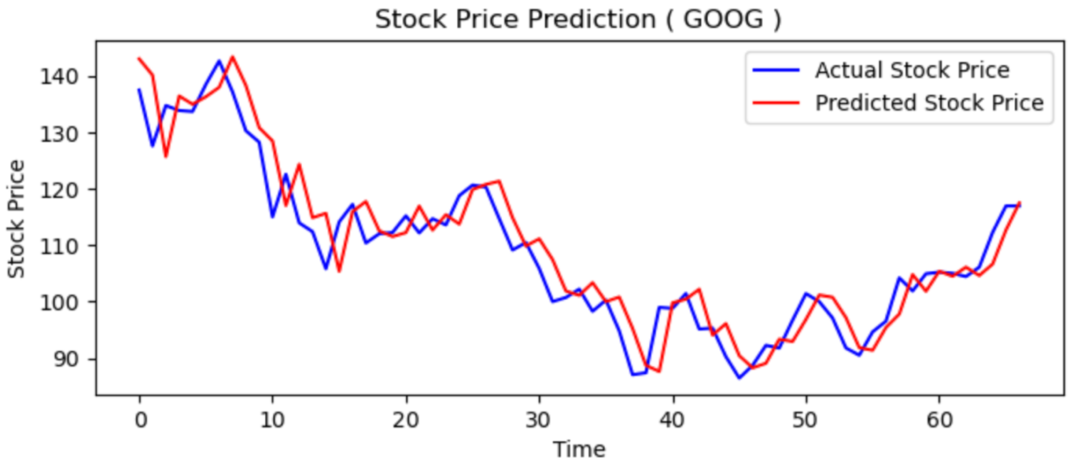


Figure 7 - Linear Regression(GOOG)

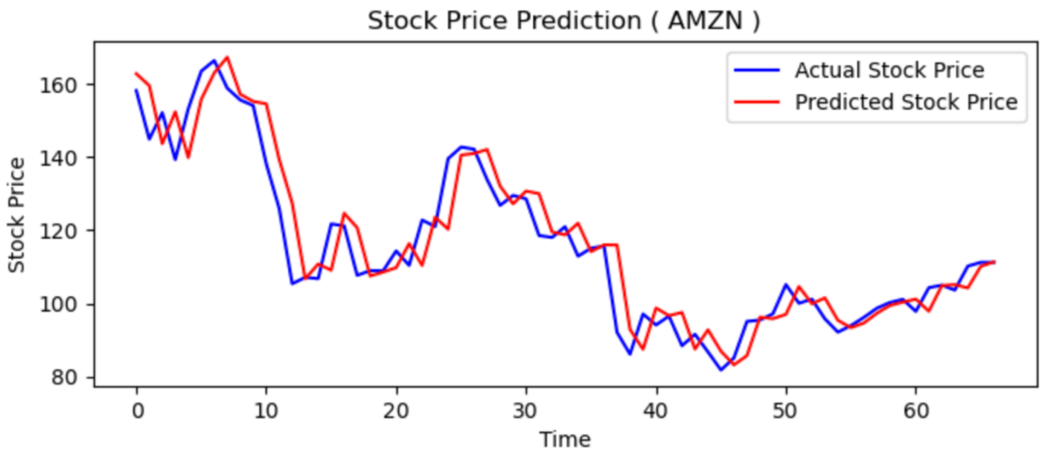


Figure 8 - Linear Regression(Amazon)

XGBOOST:

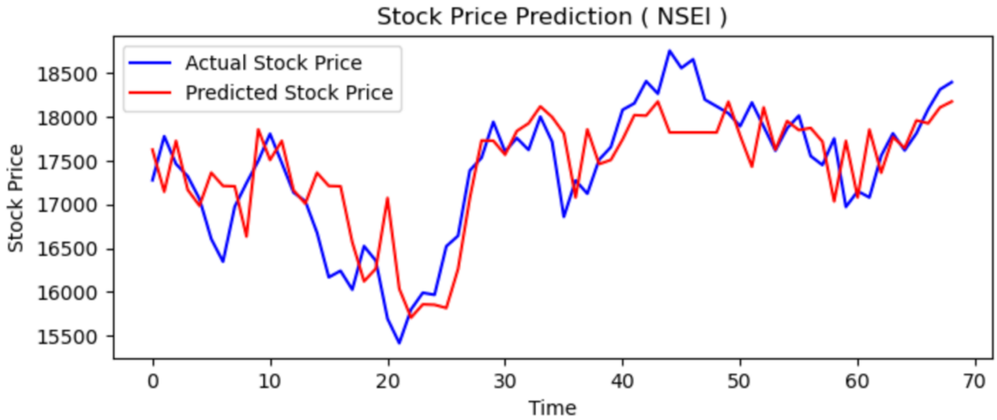


Figure 9 - XGBOOST(Nifty)

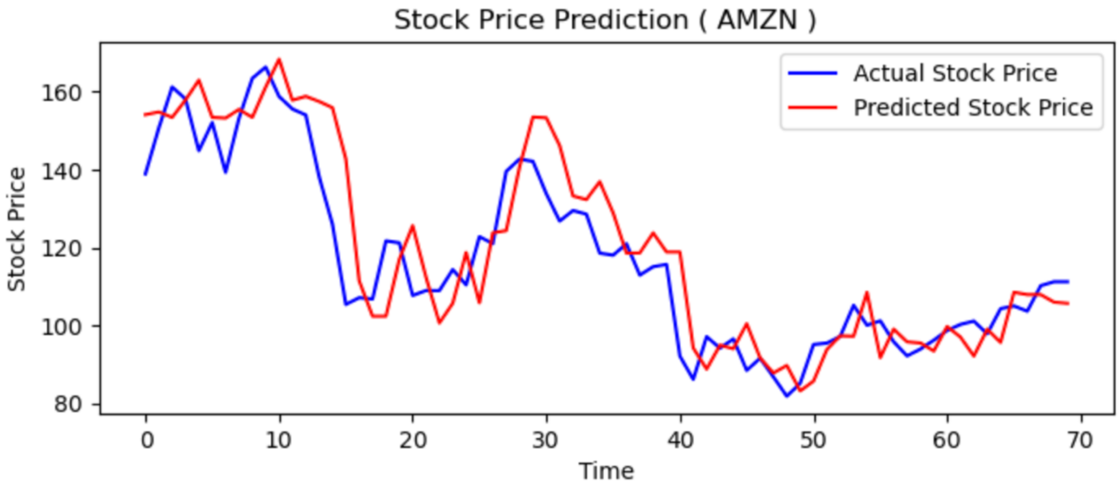


Figure 10 - XGBOOST(Amazon)

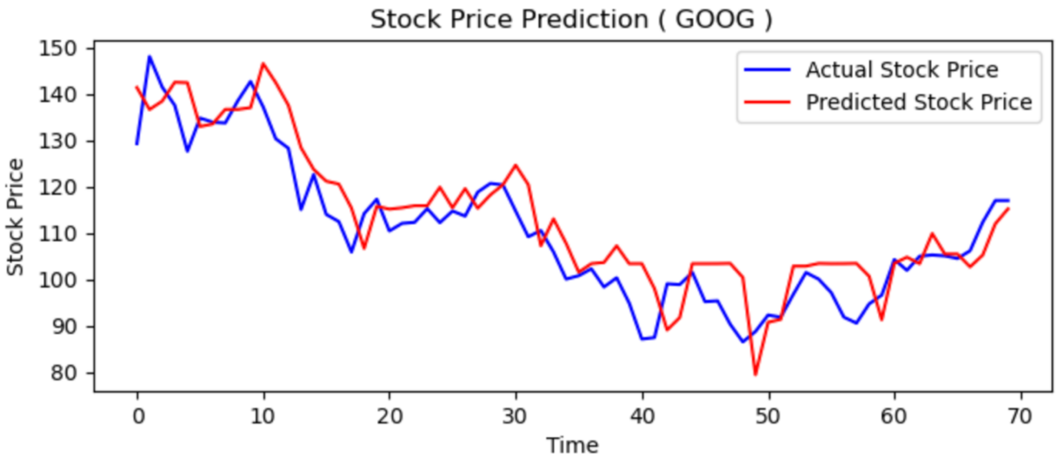
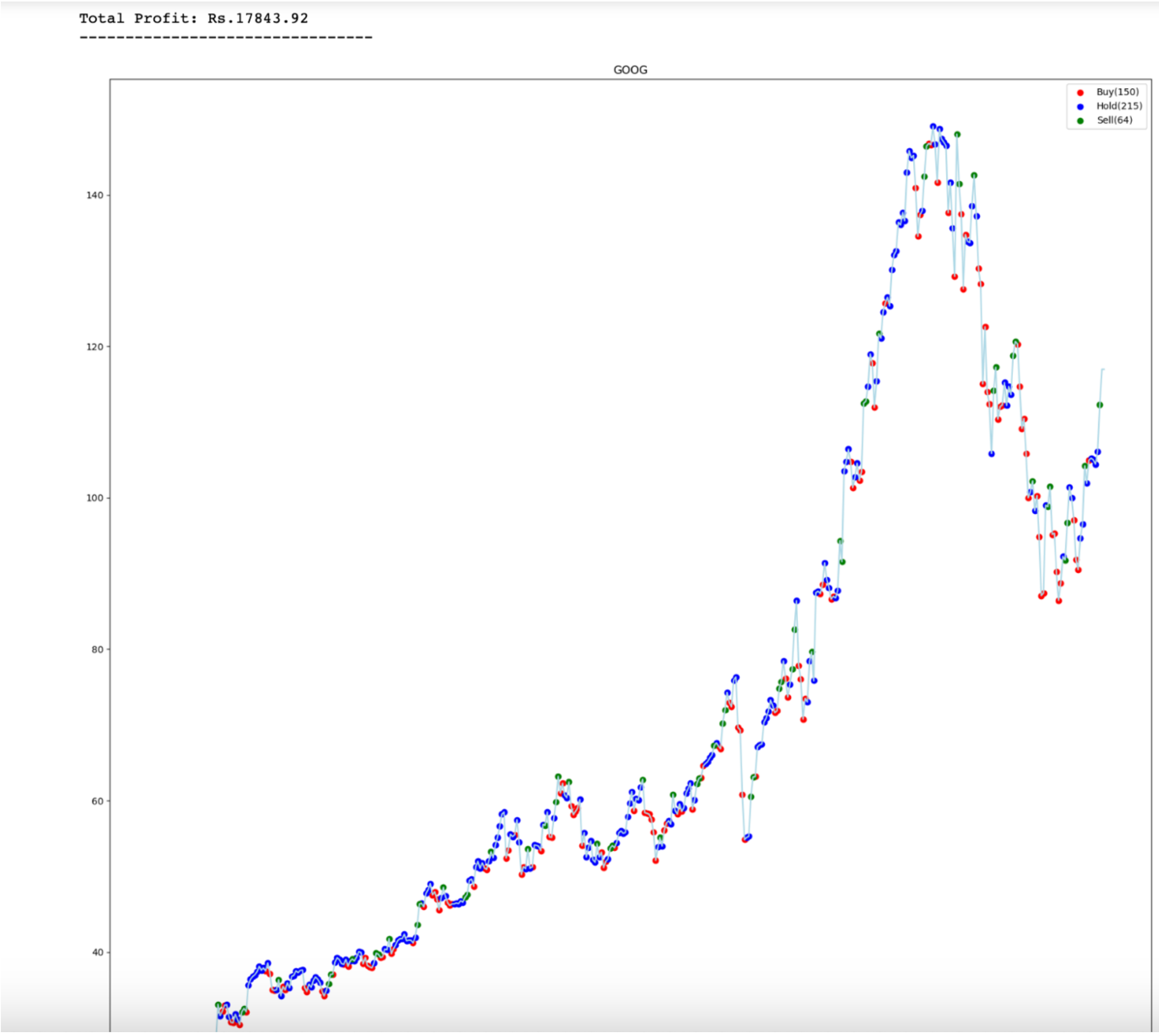


Figure 11 XGBOOST(Goog)



DQN TRADER

As mentioned above, the results that we got were extremely good as the model gave an average profit of 28% combining all stocks.

Figure 12 DQN(Goog)

```
--------------------------------
Total Profit: Rs.26042.81
--------------------------------
```

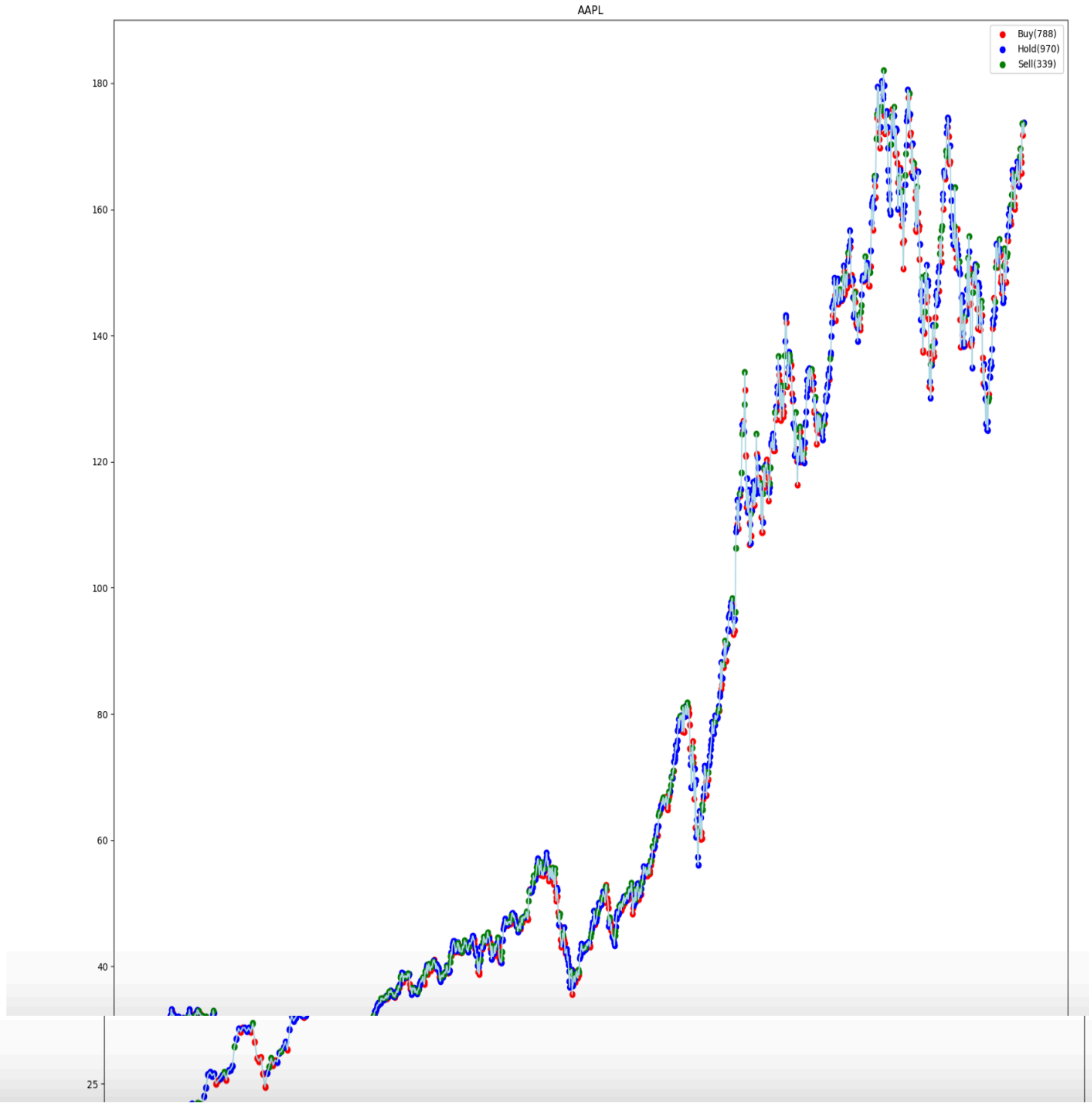


Figure 14 - DQN(Apple)

- The green dot represents the Sell
- Red represents the buy

- Blue represents the Hold

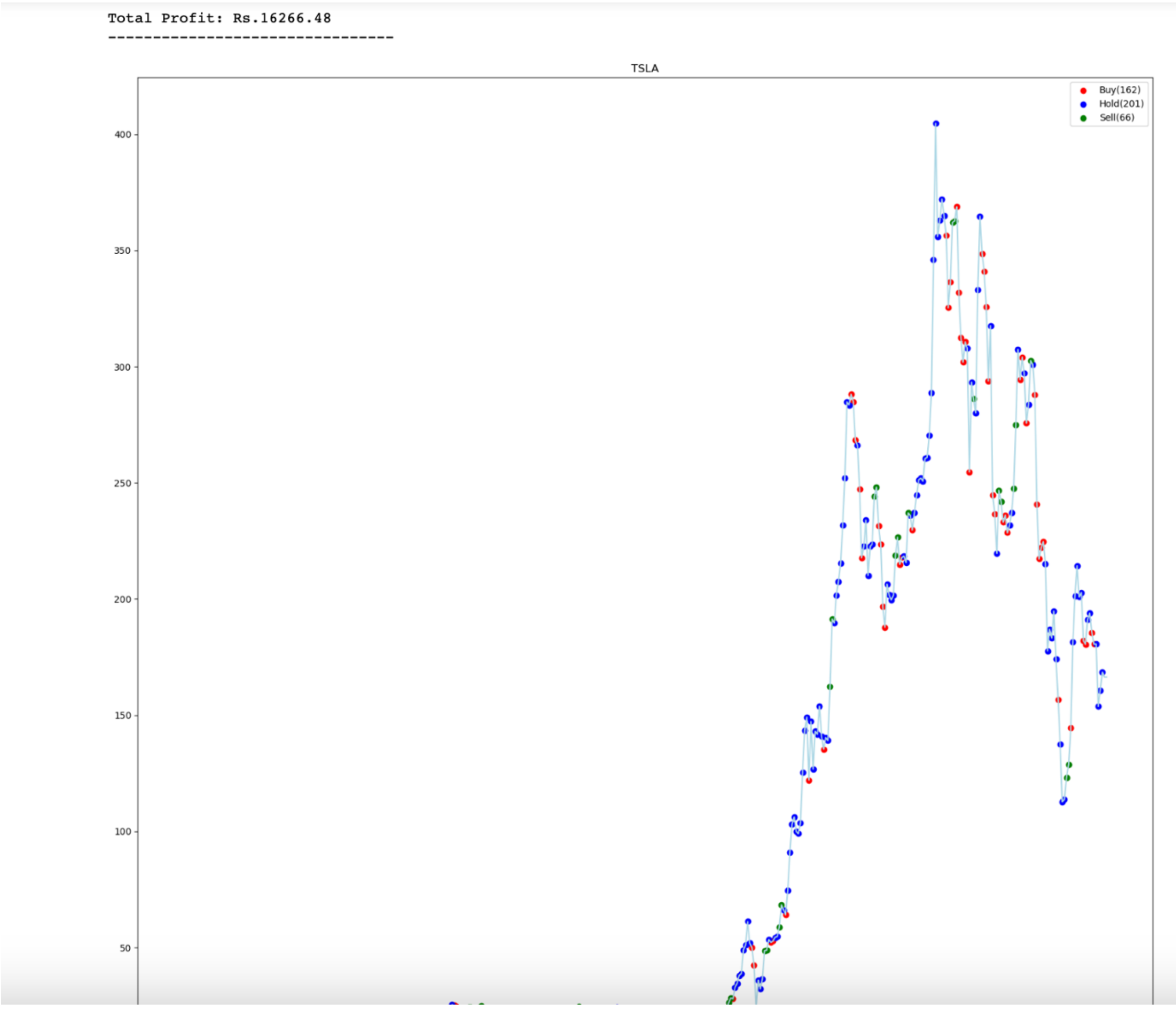


Figure 15 - DQN(Tesla)

The model was also tested on different values where we changed the Amount of money and total Transaction and it gave an average of 22.65% profit on all the Stocks.

Total Profit:- 17665

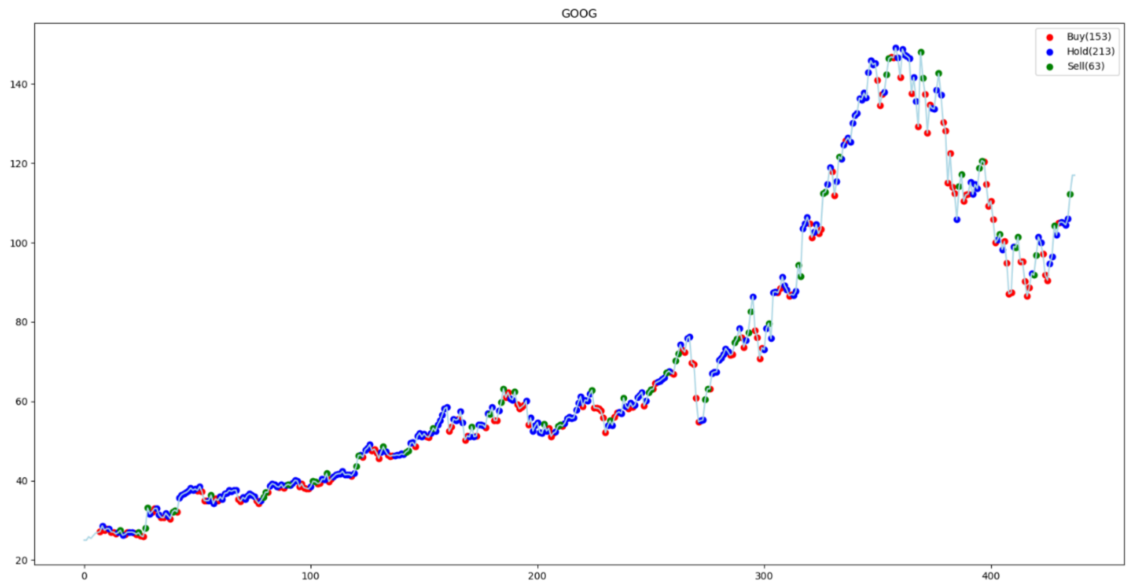


Figure 16 - DQN(New Value)

During the evaluation period, the DQN model generated a profit of about 28%. This shows that the trading technique the model used was able to produce positive returns. Multiple equities were included in the evaluation, and the model consistently showed how to make lucrative trading selections.

Discussion:
A striking outcome, showing that the DQN model has the potential to be a useful tool for trading on financial markets, is the obtained profit of 28%. However, it's crucial to take into account a number of considerations when interpreting these findings.

The performance of the model inside the particular financial market and time frame of its evaluation must first be carefully examined. Market conditions, volatility, and economic events are just a few of the many variables that affect the financial markets. It is necessary to assess the model's robustness in various situations because the model's performance may vary depending on the state of the market.
Evaluation of the evaluation methodology's shortcomings is also crucial. Due to the use of historical data, the evaluation may not have accurately reflected current market conditions. Due to delays in the availability of data, the execution of orders, and market liquidity, the model's performance may differ when used in real-world trading scenarios. As a result, the findings should be carefully evaluated before being tried again in a real-world trading setting.

The quantity and quality of training data also play a role in the trading strategy's performance. It is crucial to make sure that the training data is indicative of the desired trading environment and captures a wide range of market situations. Additional information about the model's generalization ability can be gained by analyzing its performance on other datasets.

The performance of the DQN model is also greatly influenced by the choice of hyperparameters. The model's profitability may be increased by fine-tuning the hyperparameters, including learning rate, discount factor, exploration rate, and neural network design. Sensitivity analysis on these parameters can give important information about how they affect the effectiveness of the model.

It's also crucial to evaluate how the DQN model performs in comparison to benchmarks and alternative trading methods. This enables a thorough evaluation of the model's performance and potential superiority over conventional trading strategies or market indices. These comparisons offer helpful context for evaluating the model's performance in relation to current approaches.

CONCLUSION

The DQN model, in conclusion, showed encouraging results by turning a profit of about 28% over the evaluation time. Although these results show the model's potential, it is important to take into account the unique market situation, assess its robustness, and address any potential shortcomings. To fully comprehend the model's performance and its feasibility as a trading tool, further improvements are required, such as hyperparameter fine-tuning, testing in real-world trading scenarios, and comparisons with competing methods.

Future work in this area will focus on exploring advanced model architectures, ensemble methods, transfer learning methods, and optimizing the trade-off between exploration and exploitation. Important areas of focus will include incorporating risk management techniques, portfolio optimization methods, and real-time trading implementation.

REFERENCES:

Shah, E. (2021, January 16). *Automated Stock Trading*. Retrieved from Analytics VIdhya: https://www.analyticsvidhya.com/blog/2021/01/bear-run-or-bull-run-can-reinforcement-learning-help-in-automated-trading/

Yang, B. (2020, August 25). *Deep Reinforcement Learning for Automated Stock Trading*. Retrieved from Towards Data Science: https://towardsdatascience.com/deep-reinforcement-learning-for-automated-stock-trading-f1dad0126a02

Taylan Kabbani, E. D. (2022, July 05). *Deep Reinforcement Learning Approach for Trading Automation in The Stock Market*. Retrieved from arxiv.org: https://arxiv.org/abs/2208.07165

Foy, P. (2021). *Deep Reinforcement Learning for Trading: Strategy Development & AutoML*. Retrieved from mlq.ai: https://www.mlq.ai/deep-reinforcement-learning-trading-strategies-automl/

Hongyang Yang1, X.-Y. L. (2021, April). *Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy*. Retrieved from https://damoracapital.com/wp-content/uploads/2021/04/Deep-reinforcement-learning-for-Automated-Stock-trading-Ensemble-Strategy-ID3690996.pdf

Kong, M. (2021, Jan 03). *first_pagesettingsOrder Article Reprints Open AccessArticle Empirical Analysis of Automated Stock Trading Using Deep Reinforcement Learning*. Retrieved from mdpi.com: https://www.mdpi.com/2076-3417/13/1/633

Salvatore Carta, A. F. (2021, Feb). *Multi-DQN: An ensemble of Deep Q-learning agents for stock market forecasting*. Retrieved from sciencedirect.com: https://www.sciencedirect.com/science/article/abs/pii/S0957417420306321

Yawei Li, 1. L. (2022, March 1). *Stock Trading Strategies Based on Deep Reinforcement Learning*. Retrieved from hindawi.com: https://www.hindawi.com/journals/sp/2022/4698656/

Undertaking from the PG student while submitting his/her final dissertation to his respective institute

Ref. No. _________

I , the following student

Sr. No.	Sequence of students names on a dissertation	Students name	Name of the Institute & Place	Email & Mobile
1.	First Author	Avesh Kumar Bhati	SIG	Email:22070243005@sig.ac.in Mobile:7302436125

Note: Put additional rows in case of more number of students

hereby give an undertaking that the dissertation" Deep **Q-Network Trader: Reinforcement Learning for Automated Trading**" been checked for its Similarity Index/Plagiarism through Turnitin software tool; and that the document has been prepared by me and it is my original work and free of any plagiarism. It was found that:

1.	The Similarity Index (SI) was: *(Note: SI range: 0 to 10%; if SI is >10%, then authors cannot communicate ms; **attachment of SI report is mandatory**)*	7%
2.	The ethical clearance for research work conducted obtained from: *(Note: Name the consent obtaining body; if 'not appliable' then write so)*	NA
3.	The source of funding for research was: *(Note: Name the funding agency; or write 'self' if no funding source is involved)*	Self
4.	Conflict of interest: *(Note: Tick √ whichever is applicable)*	No
5.	The material (adopted text, tables, figures, graphs, etc.) as has been obtained from other sources, has been duly acknowledged in the manuscript: *(Note: Tick √ whichever is applicable)*	Yes

In case if any of the above-furnished information is found false at any point in time, then the University authorities can take action as deemed fit against all of us.

Avesh Kumar Bhati

Full Name &
Signature of the student

Dr. Vidya Patkar

Name &
Signature of SIU Guide/Mentor

Date: 19 July 2023

Endorsement by
Academic Integrity Committee (AIC)

Place: Pune

Turnitin Originality Report

Processed on: 19-Jul-2023 12:08 IST

ID: 2125058353

Word Count: 5469

Submitted: 4

Project Report By Avesh Kumar Bhati

Similarity Index	Similarity by Source	
7%	Internet Sources:	4%
	Publications:	3%
	Student Papers:	3%

1% match (student papers from 25-Apr-2023)
Submitted to University of Central Florida on 2023-04-25

1% match (Salvatore Carta, Anselmo Ferreira, Alessandro Sebastian Podda, Diego Reforgiato Recupero, Antonio Sanna. "Multi-DQN: an Ensemble of Deep Q-Learning Agents for Stock Market Forecasting", Expert Systems with Applications, 2020)
Salvatore Carta, Anselmo Ferreira, Alessandro Sebastian Podda, Diego Reforgiato Recupero, Antonio Sanna. "Multi-DQN: an Ensemble of Deep Q-Learning Agents for Stock Market Forecasting", Expert Systems with Applications, 2020

1% match (Internet from 24-Mar-2021)
https://www.ic.unicamp.br/~ra023169/publications/jp14.pdf

1% match (student papers from 01-May-2023)
Submitted to University of Bolton on 2023-05-01

< 1% match ("Intelligent Systems and Applications", Springer Science and Business Media LLC, 2020)
"Intelligent Systems and Applications", Springer Science and Business Media LLC, 2020

< 1% match (Internet from 08-Jul-2023)
https://link.springer.com/chapter/10.1007/978-3-031-36402-0_45?code=1af21d27-2c41-477a-b1ea-8209838d71a1&error=cookies_not_supported

< 1% match (Internet from 12-May-2023)
https://www.mdpi.com/2076-3417/13/1/633/htm

< 1% match (student papers from 19-Dec-2022)
Submitted to Birla Institute of Technology and Science Pilani on 2022-12-19

< 1% match (Internet from 27-May-2023)
https://fatcat.wiki/release/3u3wyd57gzh57liyumcknbaehy