# Stock Market Prediction Using Deep Reinforcement Learning

**A PROJECT REPORT**

Submitted by

**ELDHOSE ABRAHAM TOC18CS019**

**to**

## APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

in partial fulfillment of the requirements for the award of the degree of

**Bachelor of Technology**
in
**COMPUTER SCIENCE AND ENGINEERING**


**JUNE 2022**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**Toc H INSTITUTE OF SCIENCE & TECHNOLOGY**
**Arakkunnam P.O, Ernakulam District, Kerala – 682 313**

# Toc H INSTITUTE OF SCIENCE & TECHNOLOGY
## Arakkunnam PO, Ernakulam District, Kerala – 682 313

## DEPARTMENT OF
## COMPUTER SCIENCE AND ENGINEERING

# CERTIFICATE

This is to certify that the project report entitled **'STOCK MARKET PREDICTION USING DEEP REINFORCEMENT LEARNING'** submitted by **ELDHOSE ABRAHAM (Reg. No.:TOC18CS019)** to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering, is a bonafide record of the project work carried out by him under our guidance and supervision, during the academic year 2021-'22. This report in any form has not been submitted to any other University or Institute for any purpose.

Asst. Prof Rinu Rose George
**Internal Supervisor**

Asst. Prof Elsaba Jacob
**Project Coordinator**

Asso. Prof Dr. Sreela Sreedhar
**Head of the Department**

Prof. Dr. Preethi Thekkath
**Head of the Institution**

## VISION OF CSE DEPARTMENT

To acquire global excellence in the field of Computer Science and Engineering, nurturing in professionals, technical competence, innovative skills, professional ethics and social commitment.

## MISSION OF CSE DEPARTMENT

- To equip students with a strong foundation in the area of Computer Science and Engineering using effective teaching -learning practices.
- To provide state-of-the-art infrastructure to suit academic, industry and research needs at the global level.
- To engage students and faculty in interdisciplinary research that promotes innovative ideas for sustainable development.
- To incorporate skill enhancement programmes for students and faculty to cope with the contemporary developments in technology.
- To inculcate effective communication skills, professional ethics and social commitment among professionals through value added programs.

## PROGRAM EDUCATIONAL OBJECTIVES (PEO)

Graduates of Computer Science & Engineering will

1. Evolve as globally competent computer professionals, researchers and entrepreneurs possessing collaborative and leadership skills, for developing innovative solutions in multidisciplinary domains.
2. Excel as socially committed computer engineers having mutual respect, effective communication skills, high ethical values and empathy for the needs of society.
3. Involve in lifelong learning to foster the sustainable development in the emerging areas of technology.

## PROGRAM SPECIFIC OUTCOMES (PSOs)

Students of the Computer Science and Engineering program will:

**PSO1**: **Professional Skills:** Attain the ability to design and develop hardware and software based systems, evaluate and recognize potential risks and provide creative solutions.

**PSO2**: **Successful Career and Entrepreneurship:** Gain knowledge in diverse areas of Computer Science and experience an environment conducive in cultivating skills for successful career, entrepreneurship and higher studies.

## PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

# COURSE OUTCOMES

**Course Code: CS 492**

**Course Name: PROJECT** (Semester 8)

| CO | Outcome | Level |
|---|---|---|
| CO1 | Develop a project in the computer science field or in multidisciplinary domains, preserving ethical values. | L6 |
| CO2 | Choose innovative technologies and modern tools which are appropriate for the project development. | L5 |
| CO3 | Develop the capability to manage projects as an individual or as a member/leader in a team. | L3 |
| CO4 | Develop effective communication and technical report writing skills adhering to international standards. | L3 |
| CO5 | Propose solutions to real life problems through the knowledge gained | L6 |

# DECLARATION

I undersigned hereby declare that the project design report '**STOCK MARKET PREDICTION USING DEEP REINFORCEMENT LEARNING',** submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of **Asst. Prof: Ms. Rinu Rose George**. This submission represents my ideas in my own words and where ideas or words of others have been included,I have adequately and accurately cited and referenced the original sources. I also declare that i have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place: Arakkunnam

Date: 27-06-2022

Eldhose Abraham

# ACKNOWLEDGEMENT

It is my proud privilege and duty to acknowledge the kind of help and guidance received from several people in preparation of this report. It would not have been possible to prepare this report in this form without their valuable help, cooperation, and guidance. First and foremost, I wish to record my sincere gratitude to the Management of this college and to my beloved Principal, **Prof. (Dr.) Preethi Thekkath, Principal**, Toc-H Institute of Science and Technology, Arakkunnam, for her constant support and encouragement in the preparation of this report and for making available library and laboratory facilities needed to prepare this report. My sincere thanks to **Asso. Prof. (Dr.) Georgina Binoy Joseph, Dean Academics (UG Studies)** for her valuable support and encouragement in the preparation of this report. Our sincere thanks to **Asso. Prof. (Dr.) Sreela Sreedhar**, Head of the Department of Computer Science & Engineering, for her valuable suggestions and guidance throughout the period of this report. My sincere thanks to project guide **Asst. Prof. Rinu Rose George** for her valuable suggestions and guidance. Their contributions and support in preparing this report are greatly acknowledged. I wish to thank my parents for financing our studies in this college as well as for constantly encouraging us to learn to engineer. Their personal sacrifice in providing this opportunity to learn engineering is gratefully acknowledged.

# ABSTRACT

The system that has been proposed is to develop a real-time Stock Market Prediction Model. The main aim of the project is to give predictions that help a stock trader in buying and selling Equity stocks and Futures/Options with Target and Stoploss estimation. It is an application which is designed to deliver the best user benefits. In this application we are analyzing market mood index as well as the company's historical stock data. This is done with the help of data analysis in which dynamic data is fetched and stored into a CSV file. The prediction is done using Machine Learning. In particular, we are using the Reinforcement learning method to maximize the reward and minimize the risk of loss by continuous self-learning. Thus we are acquiring high accuracy and speed.

***Keywords****: Open price,Close price,Volume,Volatility,Index.*

*Mapping with POs and PSOs:*

| ***PO1*** | ***PO2*** | ***PO3*** | ***PO4*** | ***PO5*** | ***PO6*** | ***PO7*** | ***PO8*** | ***PO9*** | ***PO10*** | ***P011*** | ***PO12*** | ***PSO1*** | ***PSO2*** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ✓ | ✓ | ✓ | | ✓ | | | ✓ | | | ✓ | ✓ | | |

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| DQN | Deep Q Learning |
| A2C | Actor 2 Critic |
| MDP | Markov Decision Process |
| NSE | National Stock Exchange |
| PPO | Percentage Price Oscillator |

# INTRODUCTION

# 1. INTRODUCTION

The task of stock prediction has always been a challenging problem for statistics experts and finance. The main reason behind this prediction is buying stocks that are likely to increase in price and then selling stocks that are probably to fall. Generally, there are two ways for stock market prediction. Fundamental analysis is one of them and relies on a company's technique and fundamental information like market position, expenses and annual growth rates. The second one is the technical analysis method, which concentrates on previous stock prices and values. This analysis uses historical charts and patterns to predict future prices. Stock markets were normally predicted by financial experts in the past. However, data scientists have started solving prediction problems with the progress of learning techniques. Also, computer scientists have begun using machine learning methods to improve the performance of prediction models and enhance the accuracy of predictions. Employing deep learning was the next phase in improving prediction models with better performance. Stock market prediction is full of challenges, and data scientists usually confront some problems when they try to develop a predictive model. Complexity and nonlinearity are two main challenges caused by the instability of the stock market and the correlation between investment psychology and market behavior.

## 1.1 BACKGROUND

What is the Stock Market?

A stock market is a public market where you can buy and sell shares for publicly listed companies. The stocks, also known as equities, represent ownership in the company. The stock exchange is the mediator that allows the buying and selling of shares.

Importance of Stock Market

- Stock markets help companies to raise capital.
- It helps generate personal wealth.
- Stock markets serve as an indicator of the state of the economy.
- It is a widely used source for people to invest money in companies with high growth potential.

Stock is an unpredictable curve that has been in the picture ever since. Its essence had been ever long living and indulging. It had grown its popularity with respect to time.People are more fascinating and interested on the same then before times.Organization had created it as a better source of revenue generation rather than investing and taking a loan approval from the bank It's way efficient and less hectic from the firm point of view.Stock is unpredictable and its been the same from the start. Its way of escalating and de-escalating has been a phenomenon and experiencing the same is the best integral part of it. It has its upper hand and flexibility with the changes that has the chance of uprising as well as crashing the whole market. Its easily defined in a few words but making an essence and understanding the same is way more hectic and time consuming.Simpler it sounds complex are its phenomenon and integrating the same. Its has its whole different sets of dependencies and integration from different agents which fluctuate the same in the market. Finding an accurate and exact value out of the same is still unaligned and no particular model of the same is seen in the market value.Finding the closest and getting an accurate proximate value out of such unpredictability is a problem in itself. Merging of the data getting the best prediction to increase the efficiency alongside considering the different aspects of the

moderator is tough and we took the same in consideration and implemented with every aspect to generate the best out of the same and get a result that can be better interrupted and the efficiency remains the same with the value of different aspects of creating an impact of reducing the risk and influencing the same over the time period to gain the most out of it.This is totally based on Machine Learning Algorithm to proceed and provide an effective result. Getting the data and processing it and generating a forecast for three days is the problem statement that we worked on.

# LITERATURE SURVEY

# 2. LITERATURE SURVEY

## 2.1. RELATED WORK

2.1.1.Srinath Ravikumar, Prasad Saraf ,Prediction of Stock Prices using Machine Learning (Regression,Classification) Algorithms,2020 International Conference for Emerging Technology (INCET) Belgaum, India. Jun 5-7, 2020,8 pages.

2.1.2.Sukhman Singh,Tarun Kumar Madan,Jitendra Kumar,Ashutosh Kumar Singh,Stock Market Forecasting using Machine Learning: Today and Tomorrow,2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT),2019,8 pages.

2.1.3.Hiransha M , Gopalakrishnan E.A , Vijay Krishna Menon, Soman K.P, NSE Stock Market Prediction Using Deep-Learning Models, International Conference on Computational Intelligence and Data Science (ICCIDS 2018), 12 pages.

2.1.4.Amritha Sharma R, Debjyoti Guha, Hitesh Agarwal, Kothiya Meetkumar Harshadbhai,Stock Market Prediction and Investment using Deep Reinforcement Learning- a Continuous Training Pipeline,International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249-8958 (Online), Volume-10 Issue-2, December 2020,8 pages

2.1.5. Hind Bourezk, Amine Raji, Nawfal Acha, Hafid Barka,Analyzing Moroccan Stock Market using Machine Learning and Sentiment Analysis, 2020 IEEE International Conference on Machine Learning,2020,8 pages

## 2.2. EXISTING SYSTEM

### 2.2.1.Prediction of Stock Prices using Machine Learning (Regression,Classification) Algorithms

1. The raw data used is from Yahoo finance.

2. The attributes used for feature extraction are ‘date’ and ‘closing price’ of a stock.

3. Features used to predict the momentum of stock price of a particular company are ‘stock momentum’, ‘index volatility’, ‘sector momentum’. These features are scaled.

4. The dataset is split into training and test data sets.

5. The training dataset is used for model training and test dataset is used for prediction .The significance of a feature is determined using the R^2 values 6. The values of the test data are predicted and the results are evaluated. The result is given on the basis of accuracy, confusion matrix and time required for the model used.

Before proceeding to the machine learning model, it is necessary to understand the technical terms related to finance and stock market in general.

A. Stock Market Index:Sometimes referred to as an index, it gives a measure of the relative value of the group of stocks within the index in numerical terms. If the stocks within an index change their values, the value of the index gets affected.

B. Outstanding Shares: They refer to a company's stock currently held by all its shareholders, including share blocks held by institutional investors and restricted shares owned by the company’s officers and insiders.

C. Market Capitalization:It refers to the total dollar market value of a company's outstanding shares. Usually referred to as ‘market cap’, it is calculated by multiplying a company's outstanding shares by the current market price of one share.

D. S&P 500 :It is an American stock market index with market capitalizations of 500 large companies that have common stock listed in NASDAQ, NYSE, etc

Disadvantages

- Since supervised ML methods are used,prediction is not accurate.
- The model is a time-consuming process
- Error prone

### 2.2.2.Stock Market Prediction Using Machine Learning

The research work was done by V Kranthi Sai Reddy Student, ECM, Sreenidhi Institute of Science and Technology, Hyderabad, India. In the finance world stock trading is one of the most important activities. Stock market prediction is an act of trying to determine the future value of a stock or other financial instrument traded on a financial exchange. This paper explains the prediction of a stock using Machine Learning. The technical and fundamental or the time series analysis is used by most of the stockbrokers while making the stock predictions. The programming language used to predict the stock market using machine learning is Python. In this paper we propose a Machine Learning (ML) approach that will be trained from the available stock data and gain intelligence and then use the acquired knowledge for an accurate prediction. In this context this study uses a machine learning technique called Support Vector Machine (SVM) to predict stock prices for the large and small capitalizations and in the three different markets, employing prices with both daily and up-to-the-minute frequencies.

Disadvantages

- Not suitable for large data sets.
- SVM does not perform very well when the data set has more noise i.e. target classes are overlapping.
- In cases when the number of features for each data point exceeds the number of training data samples, the SVM will underperform.

### 2.2.3.Stock Price Correlation Coefficient Prediction with ARIMA LSTM Hybrid Model

The research work was done by Hyeong Kyu Choi, B.A Student Dept. of Business Administration Korea University Seoul, Korea. Predicting the price correlation of two assets for future time periods is important in portfolio optimization. We apply LSTM recurrent neural networks (RNN) in predicting the stock price correlation coefficient of two individual stocks. RNN's are competent in understanding temporal dependencies. The use of LSTM cells further enhances its long-term predictive properties. To encompass both linearity and nonlinearity in the model, we adopt the ARIMA model as well. The ARIMA model filters linear tendencies in the data and passes on the residual value to the LSTM model. The ARIMA-LSTM hybrid model is tested against other traditional predictive

financial models such as the full historical model, constant correlation model, single-index model and the multi-group model. In our empirical study, the predictive ability of the ARIMA-LSTM model turned out superior to all other financial models by a significant scale. Our work implies that it is worth considering the ARIMA LSTM model to forecast correlation coefficient for portfolio optimization.

Disadvantage

- The model falls short in predicting values at the more extreme ends of the scales.
- They can be effective in capturing seasonality and the overall trend, but they can fall short in forecasting values that fall significantly outside the norm.

### 2.2.4.An innovative neural network approach for stock market prediction

The research work was done by Xiongwen Pang, Yanqiang Zhou, Pan Wang, Weiwei Lin. To develop an innovative neural network approach to achieve better stock market predictions. Data was obtained from the live stock market for real-time and off-line analysis and results of visualizations and analytics to demonstrate the Internet of Multimedia of Things for stock analysis. To study the influence of market characteristics on stock prices, traditional neural network algorithms may incorrectly predict the stock market, since the initial weight of the random selection problem can be easily prone to incorrect predictions. 9 Based on the development of word vectors in deep learning, we demonstrate the concept of "stock vector." The input is no longer a single index or single stock index, but multi-stock high-dimensional historical data. We propose the deep long short-term memory neural network (LSTM) with embedded layers and the long short-term memory neural network with automatic encoder to predict the stock market. In these two models, we use the embedded layer and the automatic encoder, respectively, to vectorize the data, in a bid to forecast the stock via a long short-term memory neural network. The experimental results show that the deep LSTM with embedded layers is better. Specifically, the accuracy of two models is 57.2 and 56.9%, respectively, for the Shanghai A-shares composite index. Furthermore, they are 52.4 and 52.5%, respectively, for individual stocks. We demonstrate research contributions in IMMT for neural network-based financial analysis.

Disadvantage

- It takes longer to train.
- Easy to overfit.
- Sensitive to different random weight initializations.
- Linear layers require large amounts of memory bandwidth to be computed

**2.2.5.Analyzing Moroccan Stock Market using Machine Learning and Sentiment Analysis**

Sentiment analysis or opinion mining is one of the most important points in big data research . It describes various computational techniques focused on discovering, extracting and distilling the human emotions, feelings or opinions from textual information within the web content towards certain entities [9]. Broadly, there is two types of methods for sentiment analysis: machine learning method and lexicon-based methods.Machine learning methods can be further divided into supervised and unsupervised approaches but it often rely on supervised classification approaches, where sentiment detection is framed as a binary (positive or negative). This approach requires labeled data to train classifiers. For supervised approaches, we need two sets of annotated data, one each for training and testing. A training set is used by an automatic classifier to learn the differentiating characteristics of documents, and a test set is used to validate the performance of the automatic classifier. Some of the most applied classifiers for supervised learning are Decision Tree (DT), SVM, Neural Network (NN), Na¨ıve Bayes, and Maximum Entropy (ME). An advantage of machine learning is its ability to be defined in detail for specific contexts. On the contrary, a disadvantage is its low applicability to new data due to costly text tagging or unavailability of sufficient examples from which the machine learning program can learn.

Disadvantage

- Delay in news availability.
- Fake news
- Investors intuition regarding news
- Sudden fluctuation without the effect of analyzing news

# PROBLEM IDENTIFICATION & OBJECTIVES OF THE PROJECT

# 3. PROBLEM IDENTIFICATION AND OBJECTIVES OF THE PROJECT

## 3.1 PROBLEM STATEMENT & OBJECTIVES

Time Series forecasting & modeling plays an important role in data analysis. Time series analysis is a specialized branch of statistics used extensively in fields such as Econometrics & Operation Research. Time Series is being widely used in analytics & data science. Stock prices are volatile in nature and price depends on various factors. The main aim of this project is to predict stock prices using Deep Reinforcement Learning.

## 3.2 SYSTEM STUDY

### 3.2.1 Drawbacks

- Since supervised ML methods are used,prediction is not accurate.
- The model is time consuming
- Error prone
- LSTMs take longer to train.
- LSTMs are easy to overfit.
- LSTMs are sensitive to different random weight initializations.
- Linear layers require large amounts of memory bandwidth to be computed

### 3.2.2 Proposed system

- The proposed system is to develop a real-time Stock Market Prediction Web Application.
- The main aim of the project is to give predictions that help a stock trader in buying and selling Equity stocks and Futures/Options with Target and Stoploss estimation.

- It is an application which is designed to deliver the best user benefits.
- In this application we are analyzing market mood index as well as the companies' historical stock data.

## 3.3 FEASIBILITY ANALYSIS

- Availability of good financial data such as news, tweets, expert's opinions and so forth
- Stock trading environment for reinforcement learning that best simulates "live" trading.
- Computational resources

## 3.4 SCOPE AND APPLICATIONS

The system that has been proposed is to develop a real-time Stock Market Prediction Web Application. The main aim of the project is to give predictions that help a stock trader in buying and selling Equity stocks and Futures/Options with Target and Stoploss estimation. It is an application which is designed to deliver the best user benefits. In this application we are analyzing market mood index as well as the companies' historical stock data. This is done with the help of data analysis in which dynamic data is fetched and stored into a CSV file. The prediction is done using Machine Learning. In particular , we are using the Reinforcement learning method to maximize the reward and minimize the risk of loss by continuous self learning. Thus we are acquiring high accuracy and speed.

### 3.4.1 Prominent features of the Project:

#### 3.4.1.1 Analyzing stock data.

We need to provide data of a particular company, and its Monthly Sales / Profit report with Months High and Low points of its Stock.

#### 3.4.1.2 Analyzing the factors.

We have to obtain the data in the same period for the following factors.

1. Demand and Supply: We will obtain the previous data entered.

2. Corporate results: Companies declare their performance results and profit at the end of each quarter.

3. Popularity: If any news about a company is about to come and is it bad or good.

We have to analyze the variations in the stock value of the companies with respect to these factors using the DRL algorithm.

# PROBLEM ANALYSIS & DESIGN

# 4. PROBLEM ANALYSIS & DESIGN

## 4.1 HIGH LEVEL DESIGN

### 4.1.1 BLOCK DIAGRAM

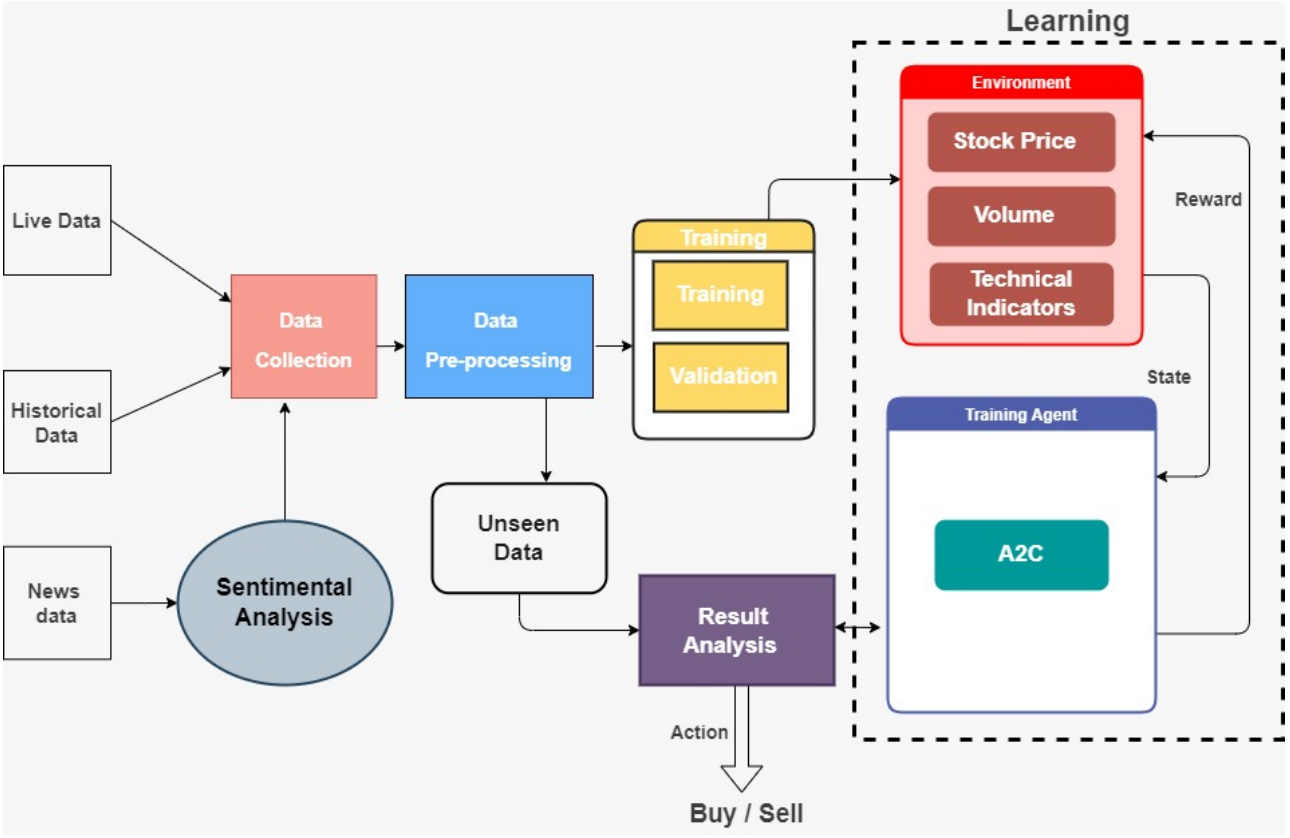



**Fig 4.1 Block diagram**

## 4.2 MODULE DESCRIPTION

### 4.2.1 Data Collection

Live Data Collection Using NSE Tools Library : nsetools is a library for collecting real time data from National Stock Exchange (India). It can be used in various types of projects which require fetching live quotes for a given stock or index or building large data sets for further data analytics.

```
pip install nsetools
```

TradingView Chart data/Yahoo Finance: This provides Historical Data with 1 min, 5 min,30 min,1 hr,4 hr.. Interval Historical data for Training Purposes.

MoneyControl API**:** To fetch real time news for sentimental analysis data.

#### 4.2.1.1 Types of Data

Market Data Market data are the temporal historical price-related numerical data of financial markets. Analysts and traders use the data to analyze the historical trend and the latest stock prices in the market. They reflect the information needed for the understanding of market behavior. Technical indicators have been widely used for SMP due to their summative representation of trends in time series data. Some studies considered different types of technical indicators, e.g, trend indicators, momentum indicators, volatility indicators and volume indicators . Textual Data Textual data is used to analyze the effect of sentiments on the stock market. Public sentiments have been proven to affect the market considerably. The textual data has many sources, such as financial news websites, general news, and social platforms.

### 4.2.2 Data preprocessing

The companies with ticker names and the sector they belong to are fetched. Using the ticker values of a company, the historical stock data of a company is acquired from Yahoo finance.The data is present in a raw format and is not feasible for analysis. The data contains the highest value, lowest value, opening value, closing value and volume of traded

stocks for a given particular date. Out of these, the date and closing value of the stock are of utmost importance to us. Using the closing value of a stock we calculate two more parameters – 'Momentum' and 'Volatility' for each company, sector and index. For each company in the dataset, the corresponding stock momentum, sector momentum and index momentum are considered. The volatility of the company, sector and index are also considered. This is done for each and every company. Now that the dataset is clean, it can be fed to a machine learning model.

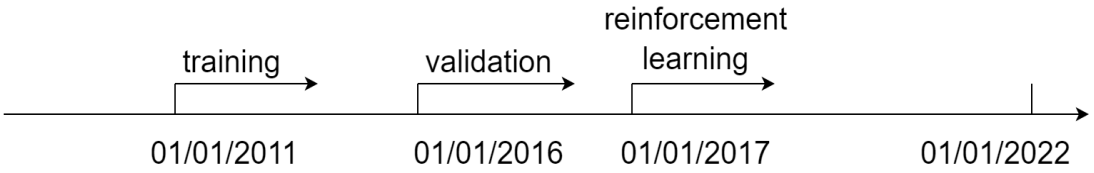
Feature Selection Feature selection is a crucial step in textual data processing. Most of the studies on SMP have used basic feature extraction techniques such as Bag of Words, where the text is broken into words and each word is converted into a numeric feature. The feature selection depends on the number of occurrences of a word. Another feature selection method is Word2Vec, It is a word embedding technique based on a multi-layer perceptron. This technique takes into consideration the order and co-occurrence of words, and hence retains the context. Order Reduction The feature selection process for the textual data leads to an increase in the number of features. High dimensional features are extremely difficult to process, and leads to the poor efficiency of most of the learning algorithms .Principal Component Analysis (PCA), is used to select the most relevant features, reducing the dimensionality of the features. In this, the daily direction of the S&P 500 index is predicted using 60 features. Feature Representation is one of the important factors for the efficient training of machine learning algorithms. Once the number of required features is determined, the input data is converted to a numeric representation so that machine learning algorithms can readily process it. Boolean representation is one of the most basic techniques of feature representation, in which the presence and absence of the feature (word) are represented by 1 and 0, respectively, for Bag of words. Another technique, Term Frequency-Inverse Document Frequency (TF-IDF).Generally, the text pre-processing phase is considered to be a crucial phase, and may significantly impact the model's accuracy.

### 4.2.3 Training

Data is divided into two : training and validation**,** for tuning of parameters. Finally, we test our agent's performance on trading data.

Before feeding the data into the training model, we need to split the entire dataset into training and test sets. The Machine Learning A2C model will be trained on the data present in the training set and tested upon on the test set for accuracy and backpropagation.

For this, we will be using the TimeSeriesSplit class of the sci-kit-learn library. We set the number of splits as 10, which denotes that 10% of the data will be used as the test set, and 90% of the data will be used for training the A2C model. The advantage of using this Time Series split is that the split time series data samples are observed at fixed time intervals.




## 4.3 USE-CASE DIAGRAM

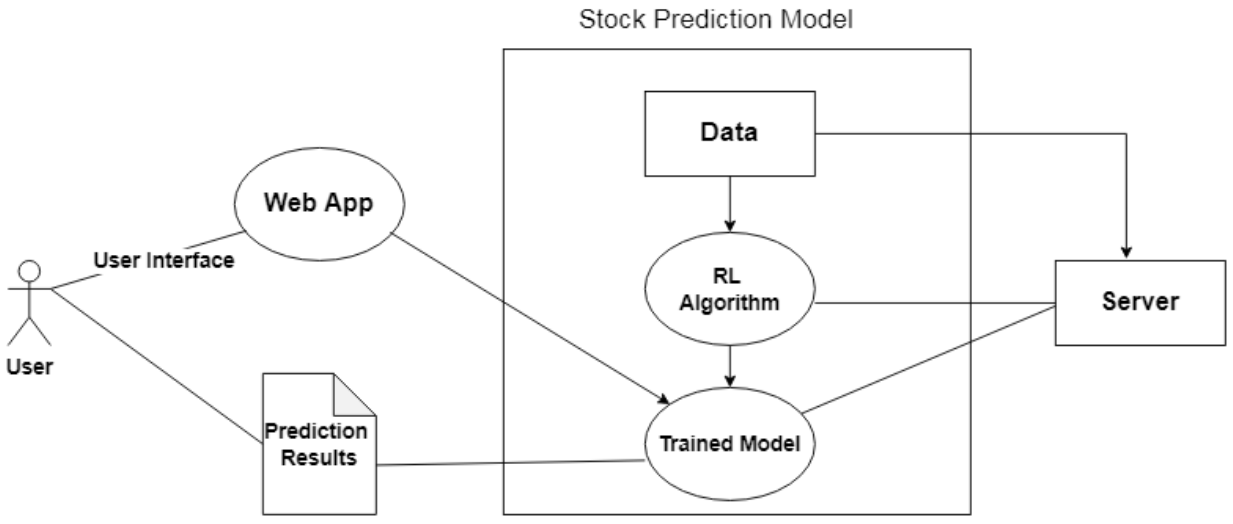



**Fig 4.2 Use-Case Diagram**

## 4.4 DATABASE DESIGN

### 4.4.1 Live Data Feed

| FIELD | TYPE |
|---|---|
| Time | int |
| Open | int |
| Close | int |
| High | int |
| Low | int |
| Volume | int |

**Table 4.1 Live Data Feed**

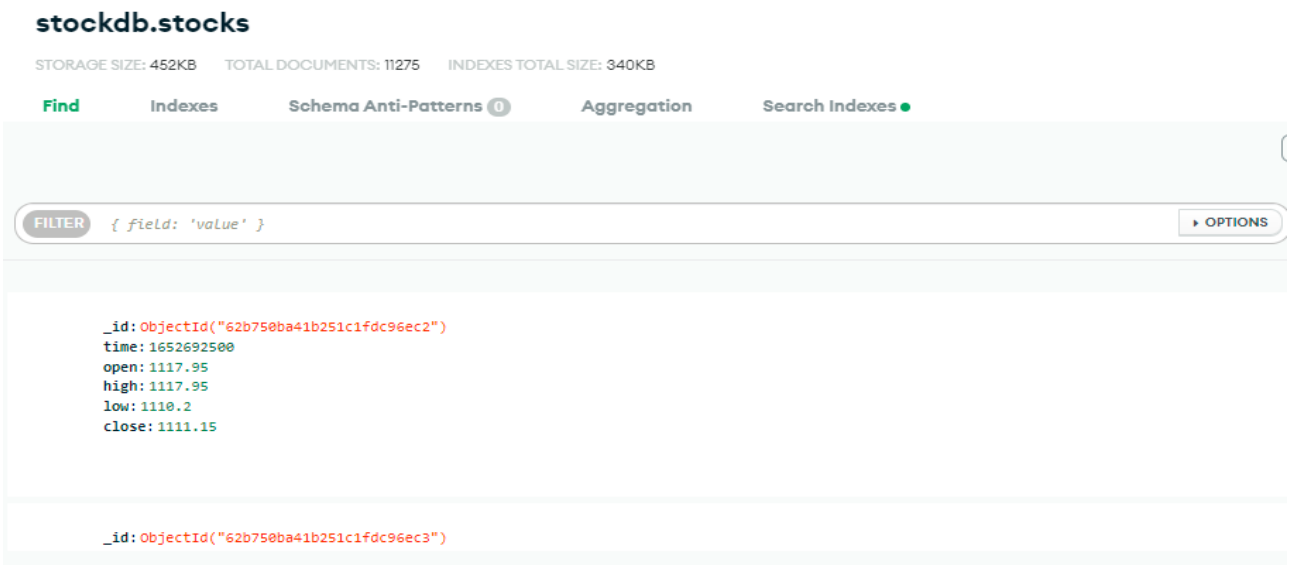



**Fig 4.3 Database**

## 4.5 ALGORITHMS / TECHNOLOGIES USED

### 4.5.1 A2C Algorithm

A2C is a typical actor-critic algorithm which we use as a component in the ensemble method. A2C was introduced to improve the policy gradient updates. A2C utilizes an advantage function to reduce the variance of the policy gradient. Instead of only estimating the value function, the critical network estimates the advantage function. Thus, the evaluation of an action not only depends on how good the action is, but also considers how much better it can be. This reduces the high variance of the policy networks and makes the model more robust.A2C uses copies of the same agent working in parallel to update gradients with different data samples. Each agent works independently to interact with the same environment. After all of the parallel agents finish calculating their gradients, A2C uses a coordinator to pass the average gradients over all the agents to a global network. So that the global network can update the actor and the critic network. The presence of a global network increases the diversity of training data. The synchronized gradient update is more cost-effective, faster and works better with large batch sizes. A2C is a great model for stock trading because of its stability.

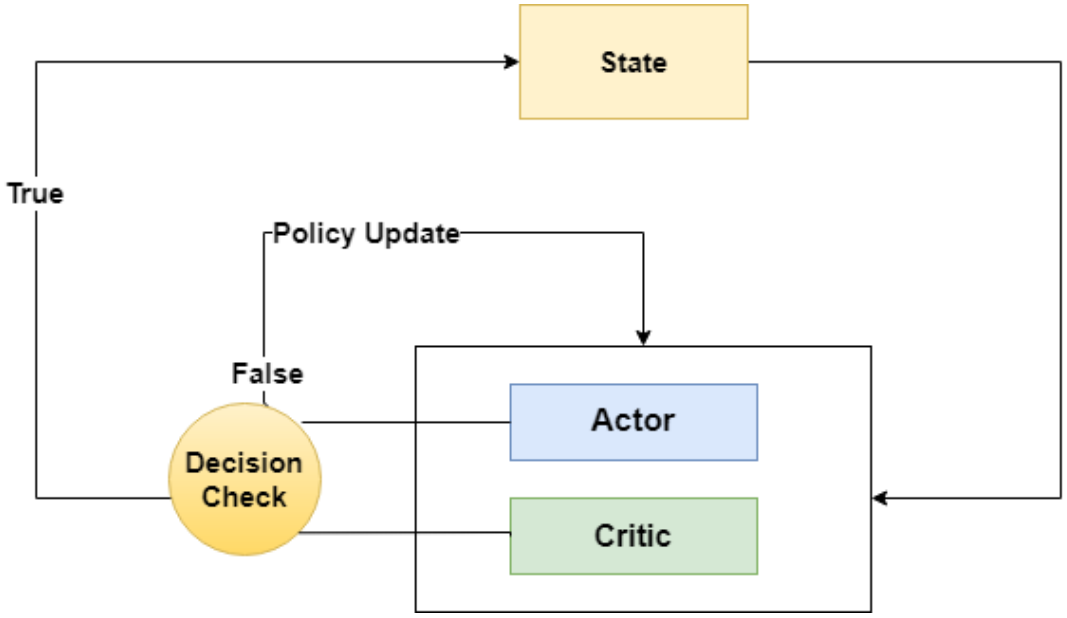



**Fig 4.4 A2C Graph**

# IMPLEMENTATION AND RESULTS

# 5. IMPLEMENTATION AND RESULTS

## 5.1 SYSTEM REQUIREMENTS

### 5.1.1 Hardware Requirements:

• RAM: 4 GB

• Storage: 500 GB

• CPU: 2 GHz or faster

• Architecture: 32-bit or 64-bit

### 5.1.2 Software Requirements:

• Python 3.5 in Google Colab is used for data pre-processing, model training and prediction

• Operating System: windows 7 and above or Linux based OS or MAC OS

## 5.2 IMPLEMENTATION PLAN

**5.2.1 Critic-only approach:** the critic-only learning approach, which is the most common, solves a discrete action space problem using, for example, Q-learning, Deep Q-learning (DQN) and its improvements, and trains an agent on a single stock or asset. The idea of the critic-only approach is to use a Q-value function to learn the optimal action-selection policy that maximizes the expected future reward given the current state. Instead of calculating a state-action value table, DQN minimizes the mean squared error between the target Q-values, and uses a neural network to perform function approximation. The major limitation of the critic-only approach is that it only works with discrete and finite state and action spaces, which is not practical for a large portfolio of stocks, since the prices are of course continuous.

- Q-learning: is a value-based Reinforcement Learning algorithm that is used to find the optimal action-selection policy using a Q function.
- DQN: In deep Q-learning, we use a neural network to approximate the Q-value function. The state is given as the input and the Q-value of allowed actions is the predicted output.

**5.2.2 Actor-only approach:** The idea here is that the agent directly learns the optimal policy itself. Instead of having a neural network to learn the Q-value, the neural network learns the policy. The policy is a probability distribution that is essentially a strategy for a given state, namely the likelihood to take an allowed action. The actor-only approach can handle continuous action space environments.

- Policy Gradient: aims to maximize the expected total rewards by directly learning the optimal policy itself.

The actor-critic approach has been recently applied in finance. The idea is to simultaneously update the actor network that represents the policy, and the critic network that represents the value function. The critic estimates the value function, while the actor updates the policy probability distribution guided by the critic with policy gradients. Over time, the actor learns to take better actions and the critic gets better at evaluating those actions. The actor-critic approach has proven to be able to learn and adapt to large and complex environments, and has been used to play popular video games, such as Doom. Thus, the actor-critic approach fits well in trading with a large stock portfolio.

- A2C**:** A2C is a typical actor-critic algorithm. A2C uses copies of the same agent working in parallel to update gradients with different data samples. Each agent works independently to interact with the same environment.
- PPO: PPO is introduced to control the policy gradient update and ensure that the new policy will not be too different from the previous one.

**5.2.3 Techniques**

The technique that we are using is Markov Decision Process and Deep Q Learning.

Markov Decision Process: The mathematical approach for mapping a solution in reinforcement Learning is recon as a Markov Decision Process.Markov decision processes (mdps) model decision making in discrete, stochastic, sequential environments. The essence of the model is that a decision maker, or agent, inhabits an environment, which changes state randomly in response to action choices made by the decision maker. The

state of the environment affects the immediate reward obtained by the agent, as well as the probabilities of future state transitions. The agent's objective is to select actions to maximize a long-term measure of total reward. Efficient algorithms for mdps based on dynamic programming and linear programming, and more recently on compact representations, enable large planning problems from artificial intelligence, operations research, economics, robotics, and the behavioral sciences to be modeled and solved.

Components of markov Decision Process:

- Agent
- Environment
- Actions
- States
- Reward

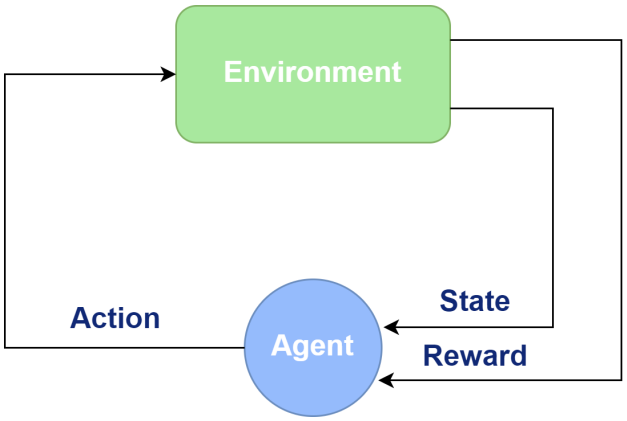



**Fig 5.1 MDP**

DQN:In deep Q-learning, we use a neural network to approximate the Q-value function. The state is given as the input and the Q-value of allowed actions is the predicted output. A DQN, or Deep Q-Network, approximates a state-value function in a Q-Learning framework with a neural network. In the Atari Games case, they take in several frames of the game as input and output state values for each action as an output.

It is usually used in conjunction with Experience Replay, for storing the episode steps in memory for off-policy learning, where samples are drawn from the replay memory at random. Additionally, the Q-Network is usually optimized towards a frozen target network that is periodically updated with the latest weights every k steps (where k is a

hyperparameter). The latter makes training more stable by preventing short-term oscillations from a moving target. The former tackles autocorrelation that would occur from on-line learning, and having a replay memory makes the problem more like a supervised learning problem.

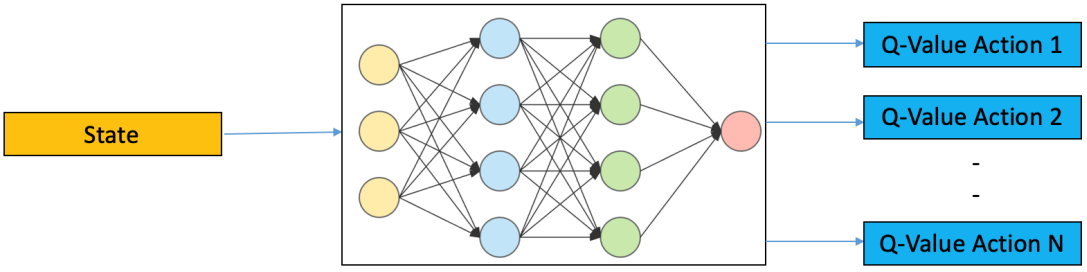



**Fig 5.2 DQN**

### 5.2.4 A2C

A2C is a policy gradient algorithm and it is part of the on-policy family. That means that we are learning the value function for one policy while following it, or in other words, we can't learn the value function by following another policy. We will be using another policy if we were using experience replay for example, because by learning from too old data, we use information generated by a policy (ie. the network) slightly different to the current state.

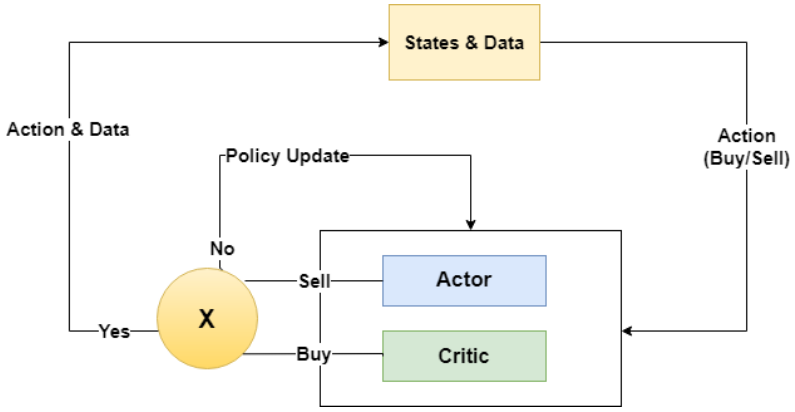



**Fig 5.3 A2C Working**

A2C, or Advantage Actor Critic, is a synchronous version of the A3C policy gradient method. As an alternative to the asynchronous implementation of A3C, A2C is a

synchronous, deterministic implementation that waits for each actor to finish its segment of experience before updating, averaging over all of the actors. This more effectively uses GPUs due to larger batch sizes.

Why does it matter to know that it's on-policy?Because that tells us how we can update the network. We can collect experiences still, but only process them immediately, the process looks like:

1. Interact with the environment and collect state transitions.
2. After n-steps, or end of the episode, calculate updates and apply them.
3. Throw away the data.

This works because the data we are going to use was generated following the same policy we are updating.

Uses the advantage function to reduce the variance of the policy gradient.Instead of only estimates the value function, the critic network estimates the advantage function.Uses copies of the same agent working in parallel to update gradients with different data samples.A coordinator to pass the average gradients over all the agents to a global network.

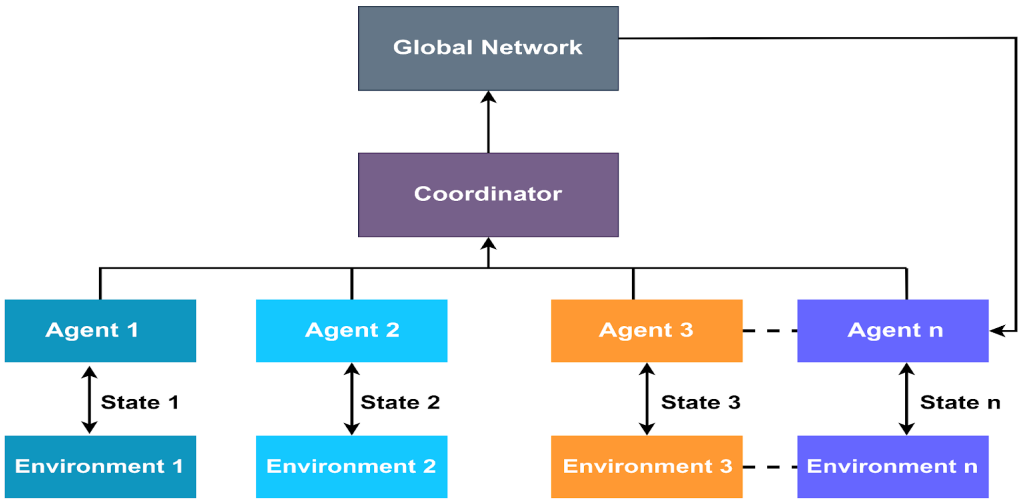



**Fig 5.4 A2C**

### 5.2.5 Sentimental Analysis

Sentiment analysis is using NLP (Natural Language Processing) techniques to determine whether data is positive, neutral or negative. News sentiment analysis helps brands and businesses monitor sentiment in article headlines and content.By monitoring news, you can extract references and mentions about your brand or business from the media. In News sentiment analysis, you analyze a news article and the content to identify a sentiment – positive, negative or neutral. This helps you understand how the public feels about your company.Sentiment analysis is a process of analyzing the sentiment of peoples from various sources where they can freely express their feelings and opinion like social media, stock market related blogs, etc. Such sources influence the other peoples to make decisions accordingly. It uses Natural Language Processing (NLP) to divide the sentiment into three categories like positive, negative and neutral. If the sentiment is positive then stock price may increases, if it is negative price may decrease and if it is neutral it maybe neither increase nor decrease.We have a stock news data taken from various twitters handles post regarding news of stock trends.We got the accuracy of approx. 76% using LSTM model. We can further increase the accuracy by finding the best hyper parameter for this model and this can be done by hyper parameter tuning, you can read more about this from here.

MoneyControl API: To fetch real time news for sentimental analysis data.

## 5.3 TESTING

System testing is normally carried out in a planned manner according to the system test plan document. The system test plan identifies all testing-related activities that must be performed, specifies the schedule of testing, and allocates resources. It also lists all the test cases and the expected outputs for each test case. Here the modules are integrated in a planned manner.

Each module must conform to a list of basic tests. This list is generic enough to apply to all modules, but rigid enough to provide a high-level start to the testing process in an accurate manner. The testing will be divided into the following sections:

1. Unit Testing
2. Integration Testing
3. Functional Testing

### 5.3.1 Unit Testing

Unit testing is the responsibility of the developer. Each module must be tested against a basic list of tests. This verifies that each module satisfies the minimum requirements for it to be accepted into the branch. Considering that each module may be significantly different, this list is only a starting place and each developer is required to expand the test list to fully verify the integrity of each module. Before development of a module, each developer is required to provide a list of basic test cases that are specific to their module. These test cases should encompass the main aspects of the module and test them thoroughly. Upon completion of development, the developer is required to create a separate list of test cases that will further verify the integrity of the module. This ensures that the module is thoroughly tested and all results are documented for the code reviewers.

Basic Unit Testing List

- Input validation and handling implemented
- Output format and standards correct
- Coding and documentation standards used
- Error handling and data handling implemented
- Does it satisfy all of the given requirements for the module

### 5.3.2 Integration Testing

Integration testing requires that the associated modules be completed and unit tested. Integration testing will verify the integration of the various modules. This requires a review of the unit testing documentation, the long-term requirements, and the standards that apply to the modules. With these documents in hand, the testers can begin to test the integration of the modules. This is done by starting with a predefined list of generic integration test cases and then defining further test cases based on the specific module functionality.

Basic Integration Testing List:

- Does the output of module A correctly match the inputs of module B.
- Do both modules conform to integration standards.

3Do both modules have the required documentation.

- Does module A call the correct functions in module B for the required functionality.
- Is the output of the combined system within the requirements specified.
- Is the input of the system a correct mapping to the output of the integrated system.
- Does the integration create any new bugs or unintended behavior.

### 5.3.3 Function Testing

Functional testing will be broken down into two sections; the preliminary functional test design early in the development process and the execution of the designed functional tests when the bulk of the system has been completed. This tests the entire system and its specifications. The design of the functional testing will focus on the actual functionality of the system. That is, they will test the inputs and outputs of the system to verify that they conform to the system specifications. Designing the functional tests early in the development process allows for early detection of specification problems that might not be discovered until the actual execution of the functional testing. The second section, the execution of the functional tests, requires the completion of the designed functional tests. Each test is executed and the result is either a pass or fail. If all functional tests pass, the system is said to pass the functional testing.

### 5.3.4 Test Cases

A test case in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not**.**

| Test Case ID | Description | Input | Expected Output | Actual Output | Pass/Fail/ Not Executed | Comment |
|---|---|---|---|---|---|---|
| 01 | Market Open | 1.Time | Success | Success | Pass | Login Successful |
| 02 | Live Data Connection | 1.Request | Retrieve Live Data | Live Data Retrieved | Pass | Data Fetch Successful |
| 03 | Model | 1.Time, 2.Price 3.Technical indicators | Predict future price and Sell or Buy stock | Sell or Buy stock and Predicted future price | Pass | Prediction Successful |

**Table 5.1 Test Cases**

## 5.4 EXPERIMENTAL RESULTS & DISCUSSION

Result Analysis is done by comparing the unseen data. Accurate results are given for action which is the prediction.Variation are taken back for learning. Thus we continue training our agent while in the trading stage, since this will help the agent to better adapt to the market dynamics.

Total Reward 0.600000 ~ Total profit 0.981852

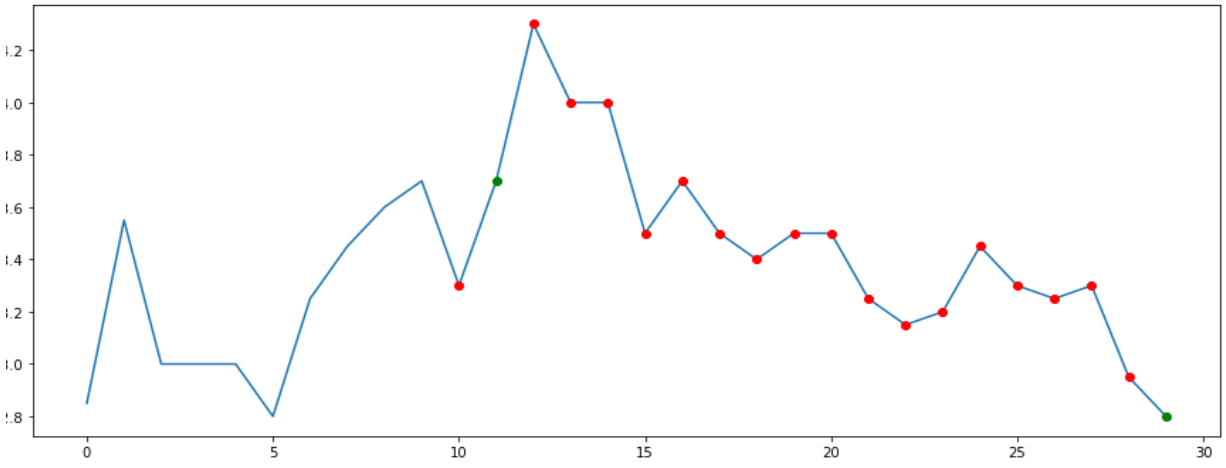



**Fig 5.4 Result Analysis**


**Fig 5.6 UI Dashboard**


**Fig 5.7 A2C UI**

# CONCLUSION

# 6. CONCLUSION

As the stock market is too uncertain, the investors must invest their money after assessing the affecting factors such as public reviews, historical data and news events. Many researchers have tried to devise prediction models using machine learning algorithms to predict the accurate prices of stocks using various tools and techniques, but have yet not been able to come up with the best possible solution. Our paper overall summarizes a few of the machine learning techniques that have been used by the research scholars to forecast the stock market trend and prices using machine learning and artificial intelligence algorithms, keeping in mind the extensive detailing, features and parameters involved. Even after analyzing the major affecting factors and incorporating the social reviews related to stock, the accurate prediction of stock price is not possible. There are some techniques that have been able to get a really close approximation.The method that we covered here is deep reinforcement learning.Though each model has its own advantage and disadvantage over others based on their evaluating factors and the datasets used for their experiments. Some models work better with historical data and others with sentiment data. Based on the literature analysis, among all the models discussed, the fusion algorithms predicted results with higher precision. They incorporate the important features of the individual techniques that they are composed of and as a result the computational time was also improved as compared to other prediction models.

# FUTURE WORKS SUGGESTED

# 7. FUTURE WORKS SUGGESTED

Future scope of this project will involve adding more parameters and factors like the financial ratios, multiple instances, etc. The more the parameters are taken into account the more will be the accuracy. The algorithms can also be applied for analyzing the contents of public comments and thus determine patterns/relationships between the customer and the corporate employee. The use of traditional algorithms and data mining techniques can also help predict the corporation performance structure as a whole.The performance of the system can be increased by adding more factors and parameters and can make the model more accurate.Applying some data augmentation techniques to the already existing data sets in orders to grow it in size and make it viable for a deep learning project,it help to train the model better.In the future, we can extend this application for predicting cryptocurrency trading.

.

# REFERENCES

# 8.REFERENCES

[1] Y.Li,W.Zhengand Z.Zheng,"Deep Robust Reinforcement Learning for Practical Algorithmic Trading," in IEEE Access, vol. 7, pp. 108014- 108022, 2019, doi: 10.1109/ACCESS.2019.2932789.

[2] K. Hiba Sadia, Aditya Sharma, Adarrsh Paul, SarmisthaPadhi, Saurav Sanyal, "Stock Market Prediction Using Machine Learning Algorithms", April 2019.https://www.ijeat.org/wpcontent/uploads/papers/v8i4/D6321048419.pdf

[3] I. Kumar, K. Dogra, C. Utreja and P. Yadav, "A Comparative Study of Supervised Machine Learning Algorithms for Stock Market Trend Prediction," 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, 2018, pp. 1003-1007, doi: 10.1109/ICICCT.2018.8473214.

[4]I. Parmar et al., "Stock Market Prediction Using Machine Learning," 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC), Jalandhar,India,2018,pp.574-576,doi: 10.1109/ICSCCC.2018.8703332.

[5]. Emerson, Sophie and Kennedy, Ruairí and O'Shea, Luke and O'Brien, John, Trends and Applications of Machine Learning in Quantitative Finance (May 30, 2019). 8th International Conference 6. on Economics and Finance Research (ICEFR 2019).

[6] M. R. Vargas, B. S. L. P. de Lima and A. G. Evsukoff, "Deep learning for stock market prediction from financial news articles," 2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), Annecy,2017,pp.60-65,doi: 10.1109/CIVEMSA.2017.7995302.

[7]. D. Shah, H. Isah and F. Zulkernine, "Predicting the Effects of News Sentiments on the Stock Market," 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 2018, pp. 4705-4708, doi: 10.1109/BigData.2018.8621884.

[8]. Pang, X., Zhou, Y., Wang, P. et al. An innovative neural network approach for stock market prediction. J Supercomput76,2098– 2118(2020). https://doi.org/10.1007/s11227-017-2228-y

[9]. Meng, T.L.; Khushi, M. Reinforcement Learning in Financial Markets. Data 2019, 4, 110. 12. Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep learning for event-driven stock prediction. In Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI'15). AAAI Press, 2327–2333.

[10]. A. Sachdeva, G. Jethwani, C. Manjunath, M. Balamurugan and A. V. N. Krishna, "An Effective Time Series Analysis for Equity Market Prediction Using Deep Learning Model," 2019 International Conference on Data Science and Communication (IconDSC), Bangalore, India,2019,pp.1-5,doi: 10.1109/IconDSC.2019.8817035.

[11] Arora, Naman and M, Parimala, Financial Analysis: Stock Market Prediction Using Deep Learning Algorithms (February 24, 2019). Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur - India, February 26-28,2019.SSRN: https://ssrn.com/abstract=3358252

[12]. C. T. Chen, A. Chen and S. Huang, "Cloning Strategies from Trading Records using Agent-based Reinforcement Learning Algorithm," 2018 IEEE International Conference on Agents (ICA), Singapore, 2018, pp. 34-37, doi: 10.1109/AGENTS.2018.8460078.

[13]. ZhuoranXiong, Xiao-Yang Liu, Shan Zhong, Hongyang (Bruce) Yang, and Anwar Walid, "Practical Deep Reinforcement Learning Approach for Stock Trading", 2018, arXiv:1811.07522v2.

[15]. Wonsup Shin, Seok-Jun Bu, Sung-Bae Cho, "Automatic Financial Trading Agent for Low-risk Portfolio Management using Deep Reinforcement Learning", 2013, arXiv:1909.03278v1 24. Azhikodan A.R., Bhat A.G.K., Jadhav M.V. (2019) Stock Trading Bot Using Deep Reinforcement Learning. In: Saini H., Sayal R., Govardhan A., Buyya R. (eds) Innovations in Computer Science and Engineering. Lecture Notes in Networks and Systems, vol 32. Springer, Singapore, doi: https://doi.org/10.1007/978-981-10-8201- 6_5

[16]. Gran, PetterKowalik; Holm, August Jacob Kjellevold; Søgård, StianGropen, "A DeepReinforcementLearningApproachtoStockTrading",2019,http://hdl.handle.net/11250/2622891

# APPENDIX

# APPENDIX - CODE SNIPPET

## Live Data

```
class LiveData:
    def __init__(self):
        self.data=[]
        self.cred={
            "APP_NAME": "5P51035409",
            "APP_SOURCE": "10077",
            "USER_ID": "F36rW3WT6ei",
            "PASSWORD": "7ehs8tC3EZb",
            "USER_KEY": "SiOqz4ExATxvpwQ0jXsPl2kFj0006DQ8",
            "ENCRYPTION_KEY":
"yhjovTPjGtCY7qwNRr1LJUHKlZE5gkKn"
        }
               self.client.login()
        scrip_list=[
            { "Exch":"N","ExchType":"C","ScripCode":3499}
            ]

self.feed_data=self.client.Request_Feed('mf','s',scrip_list)
        self.client.connect(self.feed_data)

    def callWS(self):
        def on_message(ws, message):
            live_datas = json.loads(message)
            self.data = (live_datas[0])
            ws.close()

        self.client.receive_data(on_message)
        return self.data


    #
    # def getPrice(self):
    #     def on_message(ws,message):
    #         obj = json.loads(message)
    #         time=int(ts)
    #         # time = int(obj[0]["TickDt"][6:16])
    #         # print(time, time % 60)
    #         if self.starting_ == True:
    #             open = int(obj[0]["LastRate"])
```

```
    #             high = open
    #             low = open
    #             self.starting_ == False
    #             print("mod")
    #         if (time % 60 == 0):
    #             open = int(obj[0]["LastRate"])
    #             high = open
    #             low = open
    #         close = int(obj[0]["LastRate"])
    #         if (high < close):
    #             high = close
    #         if (low > close):
    #             low = close
    #         # collection.insert_one({"time": time, "open":
open, "high": high, "low": low, "close": close})
    #         print(high, low, open, close)
```

**Main**

```
import asyncio
import json
import websockets
import time
import random

from nse_tool_live import Live_data
from live_data_5p import LiveData

live_data=LiveData()
# print(nse_t_price.fetch())



async def handler(websocket):
   start_=True
   while True:
    nse_t_price=Live_data()
    ntp=nse_t_price.fetch()
    fivePdata = live_data.callWS()
    fprice = fivePdata["LastRate"]
    time_t = int(fivePdata["TickDt"][6:16])
```

```
    ts = int(time.time())
    if(start_==True):
        ts2=ts
        open = fprice
        high = open
        low = open
        start_=False
    if(ts%60==0):
        ts2=ts
        open = fprice
        high = open
        low = open
    close = fprice
    if (high < close):
        high = close
    if (low > close):
        low = close


chartdata={"time":time_t,"open":open,"high":high,"low":low,"c
lose":close}
    print(chartdata)
    ntp_dict={'price':ntp}

main={'time':ts2,"nse_t":ntp_dict,"chartdata":chartdata,"full
":fivePdata}
    await websocket.send(json.dumps(main))
    await asyncio.sleep(1)




async def main():
    async with websockets.serve(handler, "", 8001):
        await asyncio.Future()  # run forever


if __name__ == "__main__":
    asyncio.run(main())
```

## Model

```
import asyncio
import json
import websockets
import time
import random
import numpy as np
from stable_baselines3 import A2C
from nse_tool_live import Live_data
from Five import LiveData
import pandas as pd

live_data=LiveData()


def process_data(price_lst):
    p_len = len(price_lst)
    prices = price_lst[p_len - 10:]
    print(prices)
    diff = np.insert(np.diff(prices), 0, 0)
    signal_features = np.column_stack((prices, diff))
    return signal_features

def model_loading_AWT(obs):
    model = A2C.load("A2C_without_features.h5")
    action, _states = model.predict(obs)
    print(action)
    action_list = [0.002, 0.005, 0.008, 0.011, 0.014, 0.017,
0.02, -0.002, -0.005,
                   -0.008, -0.011, -0.014, -0.017, -0.02]
    price_inc = action_list[action]
    if (price_inc > 0):
        state = "Buy"
    else:
        state = "Sell"
    return price_inc, state

def model_loading_AT(obs):
    model = A2C.load("A2C_without_features.h5")
    action, _states = model.predict(obs)
    print(action)
```

```
    action_list = [0.002, 0.005, 0.008, 0.011, 0.014, 0.017,
0.02, -0.002, -0.005,
                   -0.008, -0.011, -0.014, -0.017, -0.02]
    price_inc = action_list[action]
    if (price_inc > 0):
        state = "Buy"
    else:
        state = "Sell"
    return price_inc, state

def model_loading_A2CD(obs):
    model = A2C.load("A2C_diff_reward.h5")
    action, _states = model.predict(obs)
    print(action)
    action_list = [0.002, 0.005, 0.008, 0.011, 0.014, 0.017,
0.02, -0.002, -0.005,
                   -0.008, -0.011, -0.014, -0.017, -0.02]
    price_inc = action_list[action]
    if (price_inc > 0):
        state = "Buy"
    else:
        state = "Sell"
    return state



async def handler(websocket):
   tqty=0
   live=[]
   counter = 0
   start_=True
   while True:
    nse_t_price=Live_data()
    ntp=nse_t_price.fetch()
    fivePdata = live_data.callWS()
    fprice = fivePdata["LastRate"]
    lqty=fivePdata["LastQty"]
    tqty=tqty+lqty
    if(counter%2==0):
        live.append(fprice)
        print("added",len(live))
        if(len(live)>15):
```

```
            print("staeted")
            observations=process_data(live)
            aWT_p,aWT_s=model_loading_AWT(observations)
            a2c_T_p, a2c_T_s = model_loading_AT(observations)
            a2cd_s = model_loading_A2CD(observations)
            awt_f = {'price':fprice+fprice*aWT_p, "state":
aWT_s}
            a2c_t={'price': fprice+fprice*a2c_T_p, "state":
a2c_T_s}
            a2cd = {"state": a2cd_s}
            main = {'A2C_W_T':
awt_f,'A2CT':a2c_t,'A2CD':a2cd}
            await websocket.send(json.dumps(main))
    print(counter)
    counter=counter+1


    await asyncio.sleep(.1)



async def main():
    async with websockets.serve(handler, "", 8002):
        await asyncio.Future()  # run forever


if __name__ == "__main__":
    asyncio.run(main())
```

## Stock

```
import numpy as np

from trading_env import TradingEnv, Actions, Positions


class StocksEnv(TradingEnv):

    def __init__(self, df, window_size, frame_bound):
        assert len(frame_bound) == 2

        self.frame_bound = frame_bound
        super().__init__(df, window_size)

        self.trade_fee_bid_percent = 0.01  # unit
        self.trade_fee_ask_percent = 0.005  # unit


    def _process_data(self):
        prices = self.df.loc[:, 'Close'].to_numpy()

        prices[self.frame_bound[0] - self.window_size]  #
validate index (TODO: Improve validation)
        prices =
prices[self.frame_bound[0]-self.window_size:self.frame_bound[
1]]

        diff = np.insert(np.diff(prices), 0, 0)
        signal_features = np.column_stack((prices, diff))
        return prices, signal_features


    def _calculate_reward(self, action):
        step_reward = 10
        current_price = self.prices[self._current_tick]
        increment_price = current_price *
np.array(self.action_list)[action]
        _future_price = current_price + increment_price

        self._prediction_tick =
self.prices[self._current_tick + 10]
```

```
        if self._prediction_tick == _future_price:
            step_reward = step_reward + 15
        elif abs(self._prediction_tick - _future_price) <
self._prediction_tick * 0.0035:
            step_reward = step_reward + 7
        elif abs(self._prediction_tick - _future_price) >=
self._prediction_tick * 0.0035 and abs(
                self._prediction_tick - _future_price) <
self._prediction_tick * 0.0065:
            step_reward = step_reward + 5
        elif abs(self._prediction_tick - _future_price) >=
self._prediction_tick * 0.0065 and abs(
                self._prediction_tick - _future_price) <
self._prediction_tick * 0.01:
            step_reward = step_reward + 4
        elif abs(self._prediction_tick - _future_price) >=
self._prediction_tick * 0.01 and abs(
                self._prediction_tick - _future_price) <
self._prediction_tick * 0.015:
            step_reward = step_reward + 2
        elif abs(self._prediction_tick - _future_price) >=
self._prediction_tick * 0.015 and abs(
                self._prediction_tick - _future_price) <
self._prediction_tick * 0.02:
            step_reward = step_reward + 1
        elif abs(self._prediction_tick - _future_price) >=
self._prediction_tick * 0.02 and abs(
                self._prediction_tick - _future_price) <
self._prediction_tick * 0.03:
            step_reward = step_reward - 3
        elif abs(self._prediction_tick - _future_price) >=
self._prediction_tick * 0.03 and abs(
                self._prediction_tick - _future_price) <
self._prediction_tick * 0.04:
            step_reward = step_reward - 5
        elif abs(self._prediction_tick - _future_price) >=
self._prediction_tick * 0.04 and abs(
                self._prediction_tick - _future_price) <
self._prediction_tick * 0.06:
            step_reward = step_reward - 7
        elif abs(self._prediction_tick - _future_price) >=
self._prediction_tick * 0.06 and abs(
```

```
                self._prediction_tick - _future_price) <
self._prediction_tick * 0.1:
            step_reward = step_reward - 10
        elif abs(self._prediction_tick - _future_price) >=
self._prediction_tick * 0.1 and abs(
                self._prediction_tick - _future_price) <
self._prediction_tick * 0.3:
            step_reward = step_reward - 15
        else:
            step_reward = step_reward - 20

        return step_reward, _future_price

    def _update_profit(self, action):
        trade = False
        if ((np.array(self.action_list)[action] >= 0 and
self._position == Positions.Short) or
                (np.array(self.action_list)[action] <= 0 and
self._position == Positions.Long)):
            trade = True

        if trade or self._done:
            current_price = self.prices[self._current_tick]
            last_trade_price =
self.prices[self._last_trade_tick]

            if self._position == Positions.Long:
                shares = (self._total_profit * (1 -
self.trade_fee_ask_percent)) / last_trade_price
                self._total_profit = (shares * (1 -
self.trade_fee_bid_percent)) * current_price

    def max_possible_profit(self):
        current_tick = self._start_tick
        last_trade_tick = current_tick - 1
        profit = 1.

        while current_tick <= self._end_tick:
            position = None
            if self.prices[current_tick] <
self.prices[current_tick - 1]:
                while (current_tick <= self._end_tick and
```

```
                        self.prices[current_tick] <
self.prices[current_tick - 1]):
                    current_tick += 1
                position = Positions.Short
            else:
                while (current_tick <= self._end_tick and
                       self.prices[current_tick] >=
self.prices[current_tick - 1]):
                    current_tick += 1
                position = Positions.Long

            if position == Positions.Long:
                current_price = self.prices[current_tick - 1]
                last_trade_price =
self.prices[last_trade_tick]
                shares = profit / last_trade_price
                profit = shares * current_price
            last_trade_tick = current_tick - 1

        return profit
```

## Trading

```
import gym
from gym import spaces
from gym.utils import seeding
import numpy as np
from enum import Enum
import matplotlib.pyplot as plt


class Actions(Enum):
    inc_002 = 0
    inc_004 = 1
    inc_006 = 2
    inc_008 = 3
    inc_01 = 4
    inc_012 = 5
    inc_014 = 6
    inc_016 = 7
```

```
    inc_018 = 8
    inc_020 = 9
    dec_002 = 10
    dec_004 = 11
    dec_006 = 12
    dec_008 = 13


class Positions(Enum):
    Long = 1
    Short = 0

    def opposite(self):
        return Positions.Short if self == Positions.Long else
Positions.Long


class TradingEnv(gym.Env):
    print("Trading Env!")
    metadata = {'render.modes': ['human']}

    def __init__(self, df, window_size):
        assert df.ndim == 2
        self.seed()
        self.df = df
        self.window_size = window_size
        self.prices, self.signal_features=
self._process_data()
        self.shape = (window_size,
self.signal_features.shape[1])
        print(self.signal_features.shape[1])

        self.action_space = spaces.Discrete(len(Actions))
        print(self.action_space)
        self.observation_space = spaces.Box(low=-np.inf,
high=np.inf, shape=self.shape, dtype=np.float32)
        self._start_tick = self.window_size
        self._end_tick = len(self.prices) - 11
        self._done = None
        self._current_tick = None
        self._last_trade_tick = None
        self._position = None
```

```
        self._position_history = None
        self._total_reward = None
        self._total_profit = None
        self._first_rendering = None
        self.history = None
        self.future_pr = None
        self.action_list = [0.002, 0.005, 0.008, 0.011,
0.014, 0.017, 0.02, -0.002, -0.005,
                            -0.008, -0.011, -0.014, -0.017,
-0.02]

    def seed(self, seed=None):
        self.np_random, seed = seeding.np_random(seed)
        return [seed]

    def reset(self):
        self._done = False
        self._current_tick = self._start_tick
        self._last_trade_tick = self._current_tick - 1
        self._position = Positions.Short
        self._position_history = (self.window_size * [None])
+ [self._position]
        self._total_reward = 0.
        self._total_profit = 1.  # unit
        self._first_rendering = True
        self.history = {}
        observation=self._get_observation()
        return observation

    def step(self, action):
        self._done = False
        self._current_tick += 1

        if self._current_tick == self._end_tick or
self._current_tick > self._end_tick:
            self._done = True

        step_reward, self.future_pr =
self._calculate_reward(action)
        self.future_prices_obs()
        self._total_reward += step_reward
```

```
        self._update_profit(action)

        trade = False
        if ((np.array(self.action_list)[action]>= 0 and
self._position == Positions.Short) or
            (np.array(self.action_list)[action] <= 0 and
self._position == Positions.Long)):
            trade = True

        if trade:
            self._position = self._position.opposite()
            self._last_trade_tick = self._current_tick

        self._position_history.append(self._position)
        observation = self._get_observation()
        info = dict(
            total_reward=self._total_reward,
            total_profit=self._total_profit,
            position=self._position.value
        )
        self._update_history(info)

        return observation, step_reward, self._done, info

    def future_prices_obs(self):
        return self.prices[self._current_tick],self.future_pr

    def _get_observation(self):
        return self.signal_features[(self._current_tick -
self.window_size + 1):self._current_tick + 1]

    def _update_history(self, info):
        if not self.history:
            self.history = {key: [] for key in info.keys()}

        for key, value in info.items():
            self.history[key].append(value)

    def close(self):
        plt.close()

    def save_rendering(self, filepath):
```

```
        plt.savefig(filepath)

    def pause_rendering(self):
        plt.show()

    def _process_data(self):
        raise NotImplementedError

    def _calculate_reward(self, action):
        raise NotImplementedError

    def _update_profit(self, action):
        raise NotImplementedError

    def max_possible_profit(self):  # trade fees are ignored
        raise NotImplementedError

    def render(self, mode='human'):

        def _plot_position(position, tick):
            color = None
            if position == Positions.Short:
                color = 'red'
            elif position == Positions.Long:
                color = 'green'
            if color:
                plt.scatter(tick, self.prices[tick],
color=color)

        if self._first_rendering:
            self._first_rendering = False
            plt.cla()
            plt.plot(self.prices)
            start_position =
self._position_history[self._start_tick]
            _plot_position(start_position, self._start_tick)

        _plot_position(self._position, self._current_tick)

        plt.suptitle(
            "Total Reward: %.6f" % self._total_reward + ' ~ '
+
```

```
            "Total Profit: %.6f" % self._total_profit
        )

        plt.pause(0.01)

    def render_all(self, mode='human'):
        window_ticks = np.arange(len(self._position_history))
        plt.plot(self.prices)

        short_ticks = []
        long_ticks = []
        for i, tick in enumerate(window_ticks):
            if self._position_history[i] == Positions.Short:
                short_ticks.append(tick)
            elif self._position_history[i] == Positions.Long:
                long_ticks.append(tick)

        plt.plot(short_ticks, self.prices[short_ticks], 'ro')
        plt.plot(long_ticks, self.prices[long_ticks], 'go')

        plt.suptitle(
            "Total Reward: %.6f" % self._total_reward + ' ~ '
+
            "Total Profit: %.6f" % self._total_profit
        )
```