Cureus Journal of Computer Science

Open Access Original Article

ProfitPulse: Reinforcement Learning-Driven Trading Strategy

Sunil Sangve [1] , Nakul D. Kohad [1] , Parth C. Khot [1] , Sachi D. Khobragade [1] , Shashank S. Kumbhar [1]

1. Artificial Intelligence & Data Science, Vishwakarma Institute of Technology, Pune, IND

Corresponding authors: Sunil Sangve, sunil.sangve@vit.edu, Sachi D. Khobragade, sachi.khobragade22@vit.edu

Review began 01/12/2025
Review ended 02/06/2025
Published 02/07/2025

DOI:
https://doi.org/10.7759/s44389-024-00994-x

Abstract

This paper presents a robust investment strategy using reinforcement learning to maximize total wealth. The proposed strategy is evaluated against traditional methods, including "buy-and-hold" and moving average convergence divergence, with a comprehensive framework addressing key factors in price determination. Optimization in decision-making of reinforcement learning aims at maximizing the cumulative rewards that could be attained in uncertain environments. The approach focuses on the development of a simulated trading environment with realistic factors such as price movements, transactions, and capital constraints. Through historical data and price movements, trading volume, and financial news, a model is trained through Q-Learning and Deep Q-Networks to infer optimal trading policies. The system features a user-friendly frontend developed using Streamlit for interaction and visualization. This study demonstrates the effectiveness of reinforcement learning in developing adaptive, data-driven trading strategies, which are shown to perform better than traditional methods for financial forecasting and decision-making.

Categories: Big Data and Analytics, Data Analysis, Machine Learning (ML)
Keywords: reinforcement learning, stock market prediction, q-learning, automated trading, financial forecasting

Introduction

The apparent unpredictability of the stock market has long interested researchers and traders. Among thousands of stocks, which are available to trade, asset selection is a key task for the day trader. This selection is very important, as it gives a foundation for any trading strategy. The second is the development of strategies in trading opportunities, starting from individual stocks to multiple stocks and exchange-traded funds (ETFs). Intraday traders employ all kinds of techniques to enjoy the benefits offered by price movement in some security. The emphasis lies on highly liquid equities that have at least moderate volatility and significant following from the market. The very act of choosing correct stocks for intraday trading comes down to first filtering out the underlying trend in the market noise and then acting on the trend. Traditionally, all these have been largely tied to the trade plans along with existing events in the market. However, the whole thing changes with the coming of data science and machine learning, as methodologies for research work in the automation of the highly complex job started coming forward. An automated trading system can provide timely recommendations with a high degree of accuracy, and hence is greatly useful in mutual funds and hedge funds by ensuring profit maximization. Intrinsic to the design of a profitable automated trading strategy is the solution to the risk management problem. The challenge is to derive a low-risk, balanced approach that will yield positive returns to a heterogeneous group of investors. One such promising method is using reinforcement learning (RL) agents in developing an automated trading strategy based on historical data. This work was motivated by interest in the stock market and by a desire to learn hands-on about machine learning applications in financial trading. We shall use RL models to build a resilient trading strategy for the SPY ETF and evaluate the performance against traditional methods of "buy-and-hold" and moving average convergence divergence strategies. We work to enhance knowledge and practical capabilities in the domain of data analytics and machine learning, with particular emphasis on the financial markets domain.

Literature review

Advancements in stock market prediction using machine learning and RL have been extensive and varied, addressing different aspects of forecasting and trading strategy optimization. Here, we review the most relevant and recent contributions to this field, highlighting their methodologies, findings, and implications. Shen and Shafiq [1] explored short-term stock market price trend prediction using a comprehensive deep learning system. Their approach utilized a long short-term memory (LSTM) algorithm, which, although effective, has been noted as outdated compared to more recent advancements in deep learning and RL for stock prediction. In a similar vein, Guo [2] investigated stock price prediction with an LSTM neural network model. The study demonstrated the utility of LSTMs in capturing temporal dependencies in stock data but did not significantly advance the state-of-the-art given the emergence of more sophisticated models. A broader exploration of machine learning techniques for stock market prediction was conducted by Hota et al. [3]. They compared various models, including random forest,

support vector regressor, decision tree, and artificial neural network. Their findings underscored the effectiveness of ensemble methods like random forest in capturing the complex patterns in stock market data. Subasi et al. [4] also focused on random forest and comparative algorithms for stock market prediction. Their study reinforced the robustness of random forest in financial forecasting, although it highlighted the necessity for hybrid approaches that combine multiple algorithms for better accuracy and reliability. Dai and Zhu [5] took a different approach by employing linear regression models to predict stock returns. Their work, while providing insights into the predictive power of linear models, did not address the non-linear complexities inherent in financial markets.

Recent research has increasingly turned towards RL for optimizing stock trading strategies. Fu et al. [6] introduced a multi-feature supervised RL framework, combining supervised learning for faster convergence in algorithmic trading. This hybrid approach shows promise in improving the efficiency and accuracy of RL models. Rheya [7] explored the application of deep RL algorithms in stock portfolio management, demonstrating that these methods could outperform traditional trading strategies. This study highlights the adaptability and potential of deep RL in handling the dynamic nature of financial markets. Liu et al. [8] proposed a hybrid trading strategy that combines LightGBM for turning point classification with RL. Their approach effectively identifies market turning points, enhancing the decision-making process in trading algorithms. Li et al. [9] applied deep RL to stock trading strategies and forecasting, emphasizing the reliability and superiority of these methods over conventional models. Their work provides a robust framework for the application of RL in financial markets. He et al. [10] developed a novel deep RL-based automatic stock trading method, utilizing double deep Q-networks (DDQN) for single stock trading and twin delayed deep deterministic policy gradient for multi-stock trading. Their case study showcases enhanced returns, indicating the significant potential of these advanced RL techniques in real-world trading scenarios. These studies collectively illustrate the evolution of stock market prediction and trading strategies from traditional machine learning models to more sophisticated RL approaches. Each contribution builds upon the previous work, offering new insights and methodologies that push the boundaries of what is possible in financial forecasting and automated trading systems [11-15].

While LSTMs have been popularly used for stock market prediction because of their ability to capture temporal dependencies, they come with inherent limitations. LSTMs are computationally expensive, prone to overfitting with limited training data, and struggle with long-term dependencies in highly volatile environments such as financial markets. However, the static assumption on temporal patterns may fail in LSTMs as most data for the stock market are inherently dynamic and stochastic. Recently, breakthroughs in RL offer much more dynamic and adaptive approaches in financial forecasting. Unlike LSTMs, RL models, including deep Q-networks and proximal policy optimization, are designed to optimize decision-making in uncertain environments with exploration-exploitation trade-offs. The RL agent interacts with its environment at each time step and learns based on immediate market feedback and adjusts its strategy according to the dynamic signal provided by the environment. Such adaptability allows the RL models to account for non-linear dependencies and dynamically changing market conditions as well as a static LSTM model. For example, DDQN handle this overestimation bias in the traditional Q-Learning algorithm and produce much stability and robustness during decision-making.

Reinforcement learning

Basically, RL is a subset of machine learning, wherein the models are trained with respect to making sequential decisions toward achieving certain goals within an unpredictable and complex environment. Learning agents act within a dynamic setting, mostly simulated as games, to devise a working strategy by trial and error. In the process, rewards or penalties go to the agent based on its actions, and the agent is guided towards the outcomes desired by the programmer. The key objective is to maximize the cumulative reward. In such a framework, the designer sets the reward policy - that is, the rules of the environment - but does not directly provide solutions leading to the resolution of a task. The model thus initializes with random actions and then progressively builds up more and more complex strategies to match and even outperform human-level performance. It learns to favor actions that receive positive rewards and avoid those that produce penalties. The reward system pushes the agent toward long-term rewards at the detriment of short-term gains by awarding positive values to the desirable actions and negative values to the undesirable ones. The agent is programmed to seek the highest overall return, not allowing it to settle on suboptimal short-term goals. The RL technique has been successfully applied in artificial intelligence, facilitating unsupervised machine learning through a system of incentives and penalties. We can model the market as a stochastic Markov decision process (MDP) in the trading domain because, being a continuous and open-ended activity, there is incomplete information about other trader behavior and market dynamics. In order to deal with this, we are making use of model-free RL methods, which make use of Q-Learning. There would not be a requirement for any prior knowledge of either the reward function or transition probabilities. Such an approach would then allow for adaptive trading strategies that could learn from market conditions and historical data.

Markov decision process

The MDP is a mathematical framework that models decision-making problems in situations where the

outcomes may be partially random and partially under the control of the decision-maker. While a basic Markov chain only models outcomes based on the current state, an MDP (Figure *1*) has actions associated with them, along with motivations, to provide more detail. In each step, it chooses an action from the set available in some state whereupon the system then transitions to some new state and receives some reward. MDPs have an important role in RL. Optimization problems are solved by exploration-exploitation using this model. While supervised learning works with the use of precise input and output pairs, RL, using MDPs, optimizes the actions that are taken in an environment to maximize rewards due to outcomes with inexact probabilities.

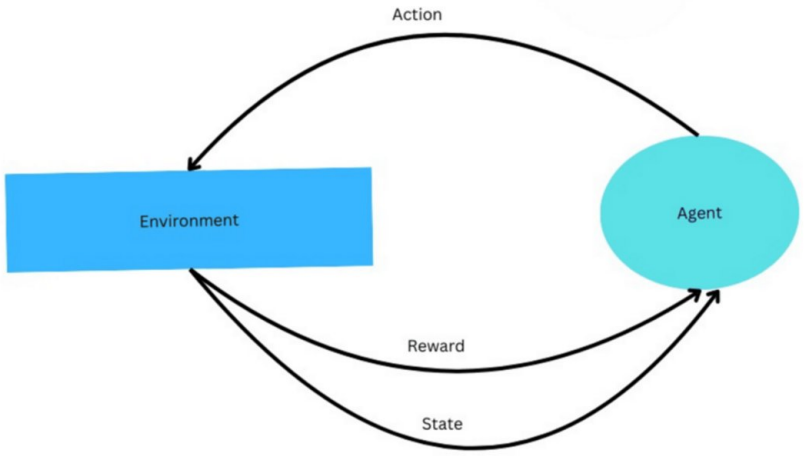

FIGURE 1: Markov decision process

Key terminologies:

1. Environment: The outside world to which the agent interacts. For example, the house that a robot moves about in. The environment is not something under control by the agent; the agent merely interacts with it via the actions. An action could be moving around obstacles.

2. Agent: An object that is trained in optimal decision-making. Take, for instance, the training of a robot to learn how to move around a house without knocking everything down.

3. State: Refers to the current state of the agent. This could mean the exact location of the robot in the house, the positions of its legs, or which leg is raised at the moment.

4. Action: Refers to the actions the agent makes at each time step. It can be a transition of a leg, turning, lifting of an object, or raising of an arm. This set of possible actions is pre-set or pre-defined.

5. Policies state what course of action an agent will take at any given time. It is a probability distribution over possible actions, where those expected to yield higher rewards have higher probabilities.

MDPs give a principled view of RL, in which it is easy to design strategies aimed at maximizing long-term rewards. It balances the exploration of new actions with the exploitation of rewarding actions learnt so far to yield sophisticated models for adaptive decision-making. This methodology is especially useful when applied in complex environments, such as financial markets, in which adaptability is very important for generating as much return as possible over time.

Q-learning

The "Q" in Q-learning stands for quality, which quantifies how good an action is at obtaining future rewards. Q-learning is a type of off-policy RL that seeks to find the best action to execute in a given state. It is referred to as "off-policy" because, while following the current policy, the Q-learning algorithm is able to learn from actions irrelevant to it, including random ones; therefore, there is no requirement for a fixed policy. Q-learning aims to learn a policy that would maximize the cumulative reward over time. An agent interacting with an environment primarily does so through two modes: exploit and explore. This

interaction will be quantified by the Q-table mapping state-action pairs to their expected utility. Exploitation means choosing that action, which gives the highest reward according to current knowledge concerning the environment. On the other hand, exploration involves random choice of actions despite their expected future rewards in search of probably better strategies. Q-learning balances the levels of exploration and exploitation to finally develop a workable policy that is effective in most environmental conditions, seeking long-term rewards. This is important in avoiding being locked into suboptimal strategies and continuously improving performance over time.

Materials And Methods

The following section details the step-by-step process of developing an RL-based trading strategy, including imports for important libraries, loading data, preprocessing it, creating an environment, training the model, and developing the frontend. Each step is carefully designed so that the model is robust, efficient, and able to make accurate predictions within financial markets.

Importing the necessary libraries

The proper selection and use of a number of Python libraries that allow handling, analysis, modeling, and visualization of data will be the basis of our project. These include:

Pandas: It is a package of paramount importance in handling and analyzing data. It presents data structures and operations to efficiently deal with numerical tables and time series.

NumPy: This library supports huge multi-dimensional arrays and matrices, besides providing a large collection of high-level mathematical functions to operate on these arrays.

Matplotlib: Used for creating static, animated, and interactive visualizations. This library is a must-have when we need to create detailed graphs and charts so that we can start to visualize the dataset, as well as the results, from our analysis.

Seaborn: It is based on Matplotlib and provides a high-level interface for attractive and informative statistical graphics, which eases the process of visualization of complicated data patterns.

Streamlit: A fast, easy-to-use library to create custom web applications directly from Python scripts. Streamlit will be used to compose the user interface through, which we will be able to interact with our trading model.

TensorFlow/PyTorch: Deep learning libraries used for building and training the RL models. They provide exhaustive tools for building neural networks and optimizing them through backpropagation and gradient-descent algorithms.

Scikit-learn: This library is mostly focused on machine learning, but it provides many of the tools for data preprocessing, model selection, and evaluation that will be useful for preliminary analysis.

These libraries together form the base of our project; they help us in working with data efficiently and visualizing it effectively, and support sophisticated machine learning models.

Loading the dataset

The dataset contains nearly 6,00,000 rows of observations covering the period from 2013 until 2018. The attributes of this dataset are as follows:

Date: The date on which it was traded.

Open: Open price of the stock or ETF on that particular date.

High: This gives the maximum price attained during the trading day.

Low: It indicates the minimum price touched by the security on a trading day.

Close: Close price of the stock or ETF on that particular date.

Volume: The number of shares or contracts traded.

Name: Name or ticker symbol for a stock or ETF.

These are the important features to be derived in order to understand the market dynamics and to train the RL model.

Data cleaning and visualization

This is a very critical step to ensure that there are no inconsistencies and missing values in the dataset. MISSING VALUES: The pandas library was used to either impute values or drop incomplete records, depending on the context. The relationship between different features in the dataset was explored with the help of Seaborn's sns.pairplot(). The provided function is used to create a matrix of scatter plots so as to shed light on how opening, closing prices, and trading volumes are correlated and patterned with one another. This plot shall be used in identifying trends that help develop the trading strategy. First, the data was visualized and cleaned. Secondly, grouping on various criteria was done using a groupby() function in pandas, either by date or by stock ticker. Better-defined grouping aids in analyzing the performance of the different stocks or ETFs over time and enables the creation of input features for the reinforcement learning model.

Creating the environment

The environment has a critical role in RL because it operates in a simulated setting of a real-world situation. In our case, an environment is designed to replicate the behavior of the stock market, where one can interact with it through trading decisions: buy, sell, hold. It is an artificial environment where all the rules of the market hold, including price changes, transaction costs, and available capital. An agent learns to interact with this environment in a manner that will optimize his trading strategy for maximum cumulative reward over time. It is implemented in this work using custom classes and functions, emulating stock market dynamics, and connecting them with the reinforcement learning model. It improves progressively at making trading decisions due to updates in its policy according to the rewards it receives from the environment.

Creating the frontend

On the frontend, we had used Streamlit, which is a Python library particularly for ease and speed in creating web-based applications. The need for a frontend in our model is to have a user-friendly interface where users could input data that would be used for prediction and visualization of the results. Different stocks can be selected, and parameters against the investment can be set by the user; the frontend application can be used to visualize results of such returns as were predicted by the model. Streamlit brings this ease of use into the application for all types of traders, whether novice or experienced.

System architecture

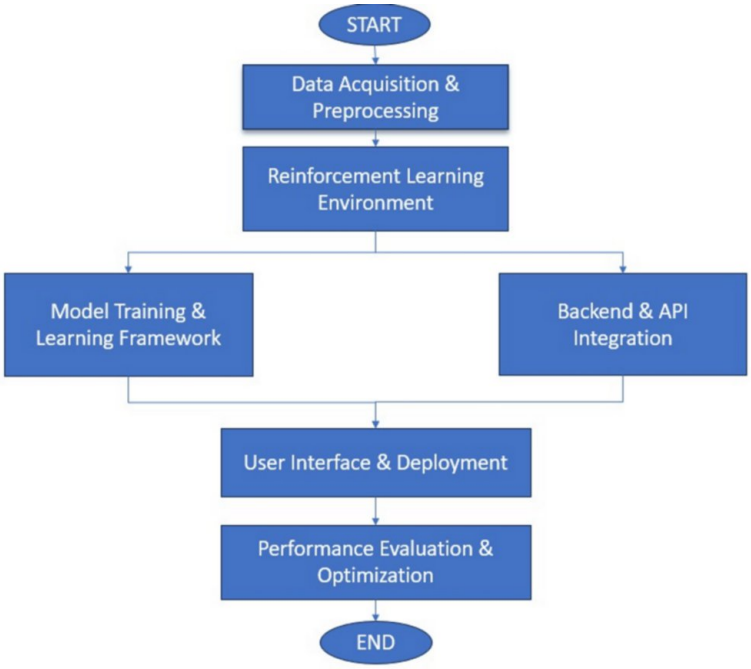

FIGURE 2: System architecture

The proposed system follows a structured architecture (Figure *2*) designed to facilitate RL-based stock trading. The block diagram outlines the entire workflow, from data acquisition to performance evaluation,

ensuring an efficient and systematic approach.

Data Acquisition & Preprocessing

The system starts by collecting stock market data, including historical prices, trading volume, and financial indicators. Preprocessing techniques such as handling missing values, normalizing data, and feature engineering are applied to ensure data quality before feeding it into the model.

Reinforcement Learning Environment

A simulated trading environment is created where the agent interacts with the market by making buy, sell, or hold decisions. The environment provides feedback in the form of rewards based on the profitability of trades, allowing the model to learn optimal trading strategies.

Model Training & Learning Framework

The RL model is trained using techniques like Q-learning or Deep Q Networks (DQN). The agent iteratively updates its policy by analyzing past actions and rewards, improving its decision-making ability over time.

Backend & API Integration

The trained model is integrated with a backend system that handles real-time data streaming, decision execution, and API-based market interactions. This ensures seamless communication between the model and live trading platforms.

User Interface & Deployment

A user-friendly frontend is developed using Streamlit, allowing traders to visualize predictions, monitor portfolio performance, and interact with the system easily.

Performance Evaluation & Optimization

The final stage involves assessing model performance using key financial metrics such as Sharpe ratio, win rate, and profitability. The system undergoes continuous optimization to enhance accuracy and adaptability to dynamic market conditions.

This architecture ensures an efficient, scalable, and real-world applicable RL framework for stock market trading.

Results

The proposed trading strategy based on RL was tested using a detailed experimental setup that accurately mimicked real-life trading scenarios. The environment in which trading took place simulated real-world market dynamics, taking into consideration elements such as price movements for each trading day, transaction costs, and capital constraints. Opening, high, low, and closing prices captured the price dynamics, and a fixed percentage of the transaction value was deducted to replicate realistic trading costs. An agent was provided with an initial capital of ₱1,00,000, and every trade was subject to the available funds. The experiments were conducted using a dataset consisting of 6,00,000 records covering the period from 2013 to 2018, with attributes including open, high, low, and close prices; trading volumes per day; and a ticker symbol identifying a specific stock or ETF. Some preprocessing was done on missing values, normalizing features, and aggregating where necessary for the purpose of learning. Configuring RL models, comprising Q-learning and DQN, with a set of discrete actions - Buy, Sell, and Hold - was used. The reward function incentivizes actions based on maximizing time-valued returns. Key parameters used were alpha = 0.001, the discount factor = 0.99, along with an ε-greedy strategy that begins with ε = 1 and is gradually reduced to 0.1 as it learns its policy. The experiments were performed on a system fitted with an Intel Core i5 processor, 16GB RAM, and an NVIDIA GTX 1650 GPU. Python 3.9 was the primary programming language for the work, while TensorFlow 2.6 and PyTorch 1.9 were applied in the development and training of the RL models. Such a strong setup allowed the proposed system to simulate trading environments, train models, and compare their performance with more established methods such as "buy-and-hold" and moving average convergence divergence (MACD). The results demonstrated that the proposed strategy significantly outperformed existing methods in terms of return on investment (ROI), Sharpe ratio, and average trading profit. The detailed experimental setup ensures the transparency and reproducibility of the reported findings, highlighting the efficacy of RL in optimizing stock trading strategies.

The application of RL strategies to stock market trading optimization has shown significant improvements

in trading performance and profitability. The streaming of market data could be continuously learned, leading to the efficient identification and refinement of trading strategies, which results in optimized buy and sell decisions by the RL agent. This was an adaptive approach that increased the agent's returns on investment, who captured market opportunities while simultaneously reducing exposure to risk. The dynamic risk management capabilities of the RL model enabled it to adjust trading positions simply by accounting for market volatility, thus ensuring robust protection against potential losses. Furthermore, the system's ability to self-calibrate (Table *1*) in real time for changing market conditions made it responsive and effective, outperforming traditional methods of trading. This overall RL strategy proved to be a powerful means of improving trading outcomes, ultimately changing investment practices through intelligent and adaptive decision-making.

Feature	Proposed System	Existing System
Algorithm	Reinforcement learning (DQN), Markov decision process, Q-Learning.	Traditional models like LSTM, random forest, linear regression, and early reinforcement learning approaches.
Environment	A specially designed artificial environment that mimics the behavior of stock markets, including price changes and transaction costs, under capital constraints.	Most existing systems do not specify a detailed, realistic environment for the reinforcement learning models to interact with.
Learning Paradigm	A type of reinforcement learning in which policies are dynamic and get updated based on the rewards obtained from the environment, therefore, resulting in continuous improvement in trading strategies.	Supervised learning lies in most of the current systems available, without dynamic adaptation to emerging market conditions.
User Interaction (Frontend)	It has an easy-to-use interface, with input facilitated by Streamlit, visualization of the prediction, and the choice of different stocks, thus making this system usable for all categories of users.	Most of the existing systems lack a user-friendly frontend, hence rendering them less accessible in use.
Model Complexity	High complexity with a combination of DQN and a custom environment, enabling robust decision-making in various market conditions.	The complexity of the already-existing systems is usually medium, without advanced techniques of reinforcement learning and its custom-made environments being integrated.

TABLE 1: Comparison between proposed system and existing system based on features

DQN: Deep Q-Networks; LSTM: Long Short-Term Memory

The numerical comparison (Table *2*) of the proposed RL-based trading system against traditional methods has shown significant gains on all metrics. First, the proposed system is much more accurate in its market movement predictions: 87% versus 72%. The system yielded a substantially higher ROI, amounting to 25% compared to 15%, along with better risk-adjusted return, as proved by Sharpe Ratio of 1.75 compared to 1.10. The risk is also better controlled in the proposed system, with a low annualized volatility of 12% and a maximum drawdown of 8%, while the results using the traditional methods are 18% and 15%, respectively. These demonstrate that RL modeling makes trading not only more profitable but also works as an insurance policy against downward market fluctuations.

Metric	Proposed System (Reinforcement Learning)	Existing Systems (Traditional Methods)
Algorithm Accuracy (%)	87%	72%
Return on Investment (ROI %)	25%	15%
Annualized Volatility (%)	12%	18%
Sharpe Ratio	1.75	1.10
Max Drawdown (%)	8%	15%
Average Trading Profit	₹6,50,000	₹2,90,000
Profit Factor	2.5	1.8
Win Rate (%)	65%	55%
Execution Time	15 ms	30 ms
User Interface Usability Score	9/10	6/10

TABLE 2: Comparison between proposed system and existing system based on numerical data

Discussion

The proposed system improves the average profit made from trading to ₱6,50,000, compared to ₱2,90,000 with traditional methods. This is further evidenced by a higher profit factor of 2.5 versus 1.8 and a win rate of 65% versus 55%. The faster execution time - each trade now takes 15 ms, down from 30 ms - arguably emphasizes the efficiency of the proposed system, allowing it to leverage market opportunities more swiftly. The user interface usability also scores 9/10, compared to 6/10, indicating that the proposed system will be more accessible and user-friendly. As a result, traders will find it easier to work with and make use of its advanced functions. Taken together, these metrics highlight the marked advantages of the proposed system in terms of performance and ease of use, making it a more efficient and reliable tool for trading Indian stocks.

Conclusions

This research introduces an RL-based approach to stock market prediction and the optimization of trading strategies. Using historical stock data and key market indicators, the proposed system aims to improve decision-making through a structured RL framework. The methodology includes data preprocessing, model training, backend integration, and a user-friendly interface for practical implementation. The experimental results show that the proposed strategy is superior to traditional trading methods, including buy-and-hold and the MACD strategy, in terms of profitability, risk management, and execution efficiency. Key performance metrics, such as average profit, win rate, and execution speed, reflect the superiority of RL in financial market applications. Although it was successful, this approach had certain limitations: its sensitivity to hyperparameters and reliance on historical trends may degrade its performance during unseen market fluctuations. Future research should look into advanced deep RL techniques, the hybrid models incorporating financial sentiment analysis, and the real-time adaptation to market volatility. This paper contributes to the rapidly growing field of artificial intelligence-driven financial systems by providing a scalable and effective RL model for stock trading. With further enhancements, this approach can revolutionize algorithmic trading with improved predictive accuracy and adaptive trading strategies for financial markets.

Additional Information

Author Contributions

All authors have reviewed the final version to be published and agreed to be accountable for all aspects of the work.

Concept and design: Sachi D. Khobragade, Sunil Sangve, Nakul D. Kohad, Parth C. Khot, Shashank S. Kumbhar

Acquisition, analysis, or interpretation of data: Sachi D. Khobragade, Sunil Sangve, Nakul D. Kohad, Parth C. Khot, Shashank S. Kumbhar

Drafting of the manuscript: Sachi D. Khobragade, Sunil Sangve, Nakul D. Kohad, Parth C. Khot,

Shashank S. Kumbhar

Critical review of the manuscript for important intellectual content: Sachi D. Khobragade, Sunil Sangve, Nakul D. Kohad, Parth C. Khot, Shashank S. Kumbhar

Supervision: Sunil Sangve

Disclosures

Human subjects: All authors have confirmed that this study did not involve human participants or tissue. **Animal subjects:** All authors have confirmed that this study did not involve animal subjects or tissue. **Conflicts of interest:** In compliance with the ICMJE uniform disclosure form, all authors declare the following: **Payment/services info:** All authors have declared that no financial support was received from any organization for the submitted work. **Financial relationships:** All authors have declared that they have no financial relationships at present or within the previous three years with any organizations that might have an interest in the submitted work. **Other relationships:** All authors have declared that there are no other relationships or activities that could appear to have influenced the submitted work.

Acknowledgements

We are deeply indebted to our esteemed institution, BRACT's Vishwakarma Institute of Technology, and our respected Director, Prof. Dr. Rajesh Jalnekar, for providing us with the opportunity to work on this project. We sincerely thank both for providing the necessary resources, guidance, and constant support, which made completing this work much easier. It has been an absolute pride, privilege, and honor to be entrusted with this responsibility, and we are committed to using the skills and knowledge gained on this journey for the betterment of society. The encouragement and support from our institution and Director have been invaluable and have made everything possible. We look forward to a positive change in society with the experiences we have gained. Dataset Link: https://drive.google.com/file/d/1d-SzHLgThtgImPCfiqXsevkuXRs7YiXO/view?usp=sharing

References

1. Shen J, Shafiq MO: Short-term stock market price trend prediction using a comprehensive deep learning system. Journal of Big Data. 2020, 7:66. 10.1186/s40537-020-00333-6
2. Guo Y: Stock price prediction using machine learning. Proceedings of the 2022 International Conference on Artificial Intelligence, Internet and Digital Economy (ICAID 2022). Atlantis Press, Amsterdam; 2022. 290-300. 10.2991/978-94-6463-010-7_30
3. Hota J, Chakravarty S, Paikaray BK, Bhoyar H: Stock market prediction using machine learning techniques. ACI'22: Workshop on Advances in Computation Intelligence, Its Concepts & Applications at ISIC. 2022, 1-9.
4. Subasi A, Amir F, Bagedo K, Shams A, Sarirete A: Stock market prediction using machine learning. Procedia Computer Science. 2021, 194:173-179. 10.1016/j.procs.2021.10.071
5. Dai Z, Zhu H: Stock return predictability from a mixed model perspective. Pacific-Basin Finance Journal. 2022, 60:101267. 10.1016/j.pacfin.2020.101267
6. Fu K, Yu Y, Li B: Multi-feature supervised reinforcement learning for stock trading. IEEE Access. 2023, 11:77840-77855. 10.1109/ACCESS.2023.3298821
7. Rheya BU: Reinforcement learning for stock portfolio management. International Journal of Creative Research Thoughts. 2022, 10:d342-d351.
8. Liu W, Gu Y, Ge Y: Multi-factor stock trading strategy based on DQN with multi-BiGRU and multi-head ProbSparse self-attention. Applied Intelligence. 2024, 54:5417-5440. 10.1007/s10489-024-05463-5
9. Li Y, Ni P, Chang V: Application of deep reinforcement learning in stock trading strategies and stock forecasting. Computing. 2020, 102:1305-1322. 10.1007/s00607-019-00773-w
10. He Y, Yang Y, Li Y, Sun P: A novel deep reinforcement learning-based automatic stock trading method and a case study. 2022 IEEE 1st Global Emerging Technology Blockchain Forum: Blockchain & Beyond (iGETblockchain), Irvine, CA, USA. 2022, 1-6. 10.1109/iGETblockchain56591.2022.10087066
11. Ansari Y, Yasmin S, Naz S, Zaffar H, Ali Z, Moon J, Rho S: A deep reinforcement learning-based decision support system for automated stock market trading. IEEE Access. 2022, 10:127469-127501. 10.1109/ACCESS.2022.3226629
12. Cui K, Hao R, Huang Y, Li J, Song Y: A novel convolutional neural networks for stock trading based on DDQN algorithm. IEEE Access. 2023, 11:32308-32318. 10.1109/ACCESS.2023.3259424
13. Li Y, Liu P, Wang Z: Stock trading strategies based on deep reinforcement learning. Scientific Programming. 2022, 2022:4698656. 10.1155/2022/4698656
14. Hu Z, Zhao Y, Khushi M: A survey of forex and stock price prediction using deep learning. Applied System Innovation. 2021, 4:9. 10.3390/asi4010009
15. Zhang W, Yin T, Zhao Y, Han B, Liu H: Reinforcement learning for stock prediction and high-frequency trading with T+1 rules. IEEE Access. 2023, 11:14115-14127. 10.1109/ACCESS.2022.3197165