Final Capstone Project:

Smart AI Stock Trading System

Group 6: Nathan Metheny, Javon Kitson, Adam Graves

University of San Diego

AAI-590: Capstone Project

Professor: Anna Marbut

April 15, 2024

GitHub: https://github.com/noface-0/AAI-590-01-Capstone/tree/main

Presentation: https://www.youtube.com/watch?v=smgWoTH2GzY

Introduction

This project introduces an advanced stock trading system utilizing AI-based algorithmic models. Algorithmic trading has significantly gained traction worldwide, with substantial growth noted in the U.S., where the market was valued at USD 14.42 billion in 2023 and is projected to reach USD 23.74 billion in the next five years (Mordor Intelligence, n.d.). The adoption of these systems has increased due to their efficiency, accuracy, and capability to process large volumes of data swiftly, gaining acceptance by regulatory bodies like the SEC and FINRA.

Contemporary algorithmic stock trading systems, such as TradeStation, rely on predictive models to forecast daily stock prices and refine these predictions down to specific moments within the trading day. The core functionality allows traders to execute orders based on specified limit prices, ensuring trades occur within predetermined cost boundaries. However, these systems often struggle to fully grasp and react to the multifaceted and interconnected nature of market dynamics due to their limited perspective, as they typically focus on individual stock patterns without fully considering the broader market's state space, which includes the interplay of various stocks and their collective influence on market behavior.

The aim of our research is to explore the performance of Deep Reinforcement Learning (DRL) within the context of financial markets. The project aims to deploy machine learning methods to construct an autonomous system that executes intelligent trading decisions. This requires the development of a model that can process and interpret the vast state and action spaces of the stock market to perform trades with the objective of optimizing financial returns. For these types of algorithms, a Deep Learning

(DL) model is more accurate than a standard Machine Learning (ML) model and performs well on unstructured data. However, it also requires a massive amount of training data and expensive hardware and software (Jakhar & Kaur, 2020).

The research will examine the configuration, training, and assessment of the system, comparing its performance with traditional trading strategies. A robust dataset from First Rate Data, consisting of 10,120 tickers and their relevant trading values, will be used to build the models. Alongside the technical, the study will investigate the conceptual aspects of reinforcement learning and the applicability to financial markets. This includes an analysis of the difficulties encountered when implementing DRL in a highly instable environment.

The model's reference behavior is designed to balance risk and reward efficiently, guiding the trading algorithm to make decisions that align with the expected risk-adjusted returns. This integration ensures that the system remains robust and responsive, capable of navigating market volatilities while adhering to the risk constraints. Our hypothesis is that this approach can adapt to market dynamics, make intelligent decisions, and produce an optimal portfolio to interact with.

Data Summary

Our dataset consisted of historical stock data from a paid licensed First Rate Data (firstratedata.com) (First Rate Data, 2023) and derived technical indicators for a diverse range of stocks spanning nearly two decades. The dataset included 35 variables, which were a combination of original stock price data and augmented

variables engineered to enhance the predictive capabilities of our Feedforward Neural Network and Deep Reinforcement Learning (DRL) models.

The variables included in the dataset are basic data fields required for stock trading, such as open, high, low, close, and volume, all of which are numeric values associated with a timestamp. In addition, we have augmented variables that were derived from the original stock price data and included numeric fields. These original variables, such as price and volume data, were directly related to our project goal of developing a DRL model for stock trading. They provided the foundation for the model to learn patterns and make trading decisions. The inclusion of augmented variables, like technical indicators, provided further signals of market movements, thereby improving the model's predictive capabilities.

We have another dataset that consists of client input used to build a client account. This data is used to calculate the risk tolerance assigned to the client. The calculations are not AI-related and are based on a combination of age, investing experience, and net worth. The risk tolerance is categorized into three levels, which have an impact on the trading portfolio.

We found significant correlations among the variables, particularly between price-related variables (e.g., open, high, low, close) and volume. Strong correlations were also observed between the original and augmented variables, as the latter were derived from the former. A representation of field correlations can be viewed in the heatmap in the visualization section (Figure 5).

Background Information

Stock trading has been a domain of significant interest for academic researchers, business entrepreneurs, and financial institutions. The goal of maximizing returns while minimizing risk has driven the development of various methods and technologies to predict market movements and make informed trading decisions. Traditionally, stock trading strategies have relied on fundamental analysis, technical analysis, and human expertise. Fundamental analysis is a method of evaluating a company through means of its financials and potential for growth. In contrast, technical analysis is an approach that relies on analyzing past market data to recognize patterns that could suggest future price movements. Human traders use a combination of these approaches, along with their experience and intuition, to make trading decisions.

However, these methods have inherent limitations in their ability to capture the complex dynamics of the stock market and adapt to evolving market conditions. Specifically, these techniques fail to consider the entirety of the state space, where all other stocks and their corresponding patterns should be taken into account. In recent years, significant progress has been made in solving challenging problems across various domains using deep reinforcement learning (DRL) (Henderson et al., 2018). DRL merges deep learning with reinforcement learning and allows an agent to learn optimal actions through continual interactions with a pre-defined, structured environment. In the context of stock trading, the agent (our DRL model) observes the state of the market (e.g., stock prices, technical indicators) and takes actions (e.g., buy, sell, hold) to maximize a reward signal (e.g., portfolio value, profit). The agent learns from its experiences of profit and loss and adjusts its strategy over time to improve its performance while profit is the goal.

Our project focuses on the application of DRL in stock trading, aiming to create an autonomous system that can learn from historical data and adapt to changing market conditions. Numerous academic articles discussing the use of DRL models to automate stock trading activity are available. For example, Yang et al. (2020) propose an ensemble strategy combining Proximal Policy Optimization (PPO), Advantage Actor Critic (A2C), and Deep Deterministic Policy Gradient (DDPG) in their research paper "Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy." This approach integrates the strengths of these three actor-critic-based algorithms, aiming to create a robust system that adapts to various market conditions.

Similar to their approach, we employ a combination of FNN, SAC, and PPO to train the stock trading component of our system. The actor network, implemented in the ActorSAC class, is responsible for selecting actions (trading decisions) based on the current state of the market. It takes the state as input and outputs the mean and log-standard deviation of a Gaussian distribution (Frisch et al., 2016). The action is then sampled from this distribution using reparameterization, allowing for the learning of a stochastic policy. The critic network, implemented in the CriticSAC class, estimates the Q-values of state-action pairs. It takes the state and action as input and outputs two Q-value estimates using separate neural networks. The use of two Q-value estimates helps to stabilize the learning process and mitigate overestimation bias.

In contrast to single-model systems, our approach is advanced and sophisticated, harnessing the collective intelligence of multiple models to enhance decision-making accuracy and adaptability within the dynamic landscape of the stock market. Current popular algorithmic stock trading systems, such as TradeStation, have

been based on the ability to predict the trading price of the stock on a day-to-day basis. As they advanced, they had the ability to go deeper into the prediction at a certain point of time, with the foundation being the ability to trade on the condition of the limit price entered.

Our research demonstrates that DRL provides significant alpha, as the stock trading strategies learned through this approach have resulted in returns that are substantially higher than those of the market average or a relevant benchmark. DRL models have also been successfully applied in the field of robotics. Haarnoja et al. (2018) discuss the valuable properties of the Soft Actor-Critic (SAC) algorithm in their research paper "Soft Actor Critic—Deep Reinforcement Learning with Real-World Robots," where they used models to train a robot to move, a 3-finger dexterous robotic hand to manipulate an object, and a 7-DoF Sawyer robot to stack Lego blocks.

Furthermore, Nan et al. (2021) incorporated additional external factors that are subject to frequent changes and often unable to be inferred solely from historical trends in their research paper "Sentiment and Knowledge-Based Algorithmic Trading with Deep Reinforcement Learning." To address this, they employed Partially Observable Markov Decision Processes (POMDP), taking into account events outside the realm of stock trading, such as the destruction of a trading data center, a scenario that actually occurred on September 11th.

In our architecture, we utilize a Genetic Agent (GA) to select a subset of stocks from a larger pool based on a predefined objective and in line with the strategy based on the client portfolio input. This ensures that the trading is within regulation requirements. The DRL models (SAC and PPO) in our project are responsible for

making trading decisions based on market conditions, and the FNN model is used as the underlying architecture for both the actor and critic networks in the SAC and PPO algorithms to predict future stock prices. This combination is well-suited to build a successful stock trading system.

Experimental Methods

Our research and implementations were built upon the foundation provided by the FinRL library (AI4Finance-Foundation, n.d.). FinRL is an open-source framework that facilitates the application of deep reinforcement learning in quantitative finance. By leveraging the FinRL library, we were able to efficiently implement and experiment with the PPO, while building the custom Genetic Algorithm (GA), Feedforward Network (FNN), Soft Actor-Critic (SAC), and Twin Delayed Deep Deterministic Policy Gradient (TD3) agents to fit into the broader architecture. The FinRL library provided a solid starting point for our research, offering a range of pre-built environments, agents, and evaluation metrics that accelerated our development process and allowed us to focus on the specific adaptations and optimizations required for our ensemble approach.

The project employs an ensemble approach, combining Deep Reinforcement Learning (DRL), a Feedforward Neural Network (FNN), and a Genetic Algorithm (GA) for stock trading, price prediction, and portfolio optimization. The DRL model makes trading decisions based on market conditions, while the FNN model predicts future stock prices, providing additional input to the DRL model. An individual's portfolio questionnaire input creates an additional dataset, from which a unique ID is identified, and a risk factor is calculated.

The risk factor, ranging from 0 to 30, is determined based on the individual's net worth, trading experience, age, and trading goals. The range is split into three categories: minimum drawdown (0-10), maximum return (11-20), and a combination of minimum drawdown and maximum return (21-30).

The Genetic Algorithm (GA) follows a standard evolutionary process encapsulated in the GeneticAlgorithm class. This class incorporates portfolio initialization, fitness evaluation of symbols calculating drawdowns, selection of parent orders, crossover to create children orders, mutation based on probabilities, output of trade results, and termination upon finding a satisfactory solution. The time_interval, start_date, and end_date variables align with the inputs used for FNN and DRL training. The GA returns the best individual portfolio found, along with its corresponding returns and drawdown. This constrained portfolio is then used for downstream training and trading.

The Feedforward Neural Network (FNN) model architecture is composed of three main components: an input layer that receives the initial data, multiple hidden layers that process the distribution, and an output layer that produces a final, singular prediction. Notable design choices include an input layer size determined by the number of input data features, a list of possible hidden layer structures [(32, 16), (64, 32), (128, 64), (256, 128)] for flexible hyperparameter selection, an output layer size of 1, dropout regularization with a default of 0.5, batch normalization, and a ReLU activation function.

FNN model training involves downloading historical stock data, dividing it into training and validation sets (typically 80/20), setting up the model architecture, and using Huber Loss (Huber F., 1964) as the loss function. Training utilizes the Adam

optimizer (Diederik P., Kingma, & Ba J. 2017) to adjust weights over 10 epochs with a batch size of 64, evaluating the model's performance on the validation set after each epoch. Optuna (Akiba T., Sano S., Yanase T., Ohta T., & Koyama M. 2019) is used for hyperparameter tuning and architectural adjustments, efficiently searching the hyperparameter space to find the best combination of learning rate and hidden layer sizes that minimize validation loss.

Two popular DRL algorithms, Soft Actor-Critic (SAC) and Proximal Policy Optimization (PPO), were experimented with for stock trading. The DRL agents were trained on the optimized portfolio returned by the GA, focusing their learning on a more promising subset of stocks. The SAC model's architecture uses multi-layer perceptrons (MLPs) with ReLU activation for both actor and critic networks. The actor network's output forms a Gaussian distribution for action decisions, while the critic network provides two Q-value estimates. The SAC algorithm's training process involves the agent interacting with the environment, gathering experiences, and updating the critic and actor networks using the collected experiences. Hyperparameters such as learning rate, discount factor, entropy coefficient, and network architecture were tuned to optimize performance.

PPO, an on-policy DRL algorithm (Schulman J., 2017), also consists of an actor network and a critic network. The actor network selects actions based on the current state, while the critic network estimates the value of each state (Chen & Xiao, 2023). PPO trains by having the agent interact with the environment, gather data, and update the actor and critic networks. The actor network updates aim to maximize future rewards without straying too far from the previous policy, while the critic network updates focus

on reducing the discrepancy between predicted and actual state values. Hyperparameters such as learning rate, discount factor, batch size, and clip range were adjusted to optimize PPO's performance.

Both SAC and PPO agents were trained using a rollout buffer to store collected experiences, interacting with the stock market environment for a specified number of iterations. The models were optimized by tuning various hyperparameters and experimenting with different reward scaling techniques and exploration strategies. The trained DRL agents were then evaluated on a separate test dataset to assess their ability to generate profitable trading strategies in unseen market conditions, using performance metrics such as cumulative returns and Sharpe ratio to compare the effectiveness of the SAC and PPO algorithms.

Results & Conclusion

The advanced stock trading system we developed, which leverages Deep Reinforcement Learning (DRL), Feedforward Neural Networks (FNN), and Genetic Algorithms (GA), has yielded promising results. By structuring the system into separate processing components and utilizing technical analysis, we designed a multiple model architecture capable of making intelligent trading decisions that balance risk and reward efficiently, based on the investor's profile input. The system's performance metrics suggest that this approach can effectively adapt to market dynamics and generate an optimal portfolio for interaction.

The system calculates a risk factor based on the investor's profile data (Figures 3 and 4), which the GA then uses to generate an appropriate portfolio. The DRL models,

Soft Actor-Critic (SAC) and Proximal Policy Optimization (PPO), were trained using these optimized portfolios, along with the Feedforward Neural Network's (FNN) future price prediction. This strategic combination allowed the system to focus on a subset of stocks, enhancing learning efficiency and trading performance. The SAC model, known for its sample-efficient learning, and the PPO model, recognized for balancing performance and stability, were instrumental in navigating the complex stock market environment. When tested against the validation data, the system generated a 52% return on investment above the initial value (Figure 6).

The assessment of our FNN model's performance demonstrated its ability to generalize effectively to new data, although there is room for improvement. The validation metrics, including Root Mean-Squared Error (RMSE) and R-Squared (R^2), indicate that the model successfully captures trends and patterns in stock price movements. The high performance can be primarily attributed to the immediate look-forward period, as the model only predicts the next timeframe. However, these results also highlight opportunities for further refinement to minimize prediction errors and improve accuracy, suggesting that with additional tuning, the model could achieve even more reliable predictions over longer, more future-oriented timeframes.

To further refine the system, we optimized the models by tuning various hyperparameters, such as the learning rate, discount factor, and network architectures (e.g., number of hidden layers and units). We also experimented with different reward scaling techniques and exploration strategies to improve the agents' performance, finding that a three-level reward scaling would best mitigate the risk factor related to the trade portfolio. The trained DRL agents were then evaluated on a separate test dataset

to assess their ability to generate profitable trading strategies in unseen market conditions, using performance metrics such as cumulative returns and Sharpe ratio to compare the effectiveness of the SAC and PPO algorithms.

When comparing the performance of the SAC and PPO algorithms, both models demonstrated strong trading strategies. The PPO model achieved a 52% return on investment above the initial value when tested against the validation data, whereas the SAC model generated an 89% return on investment.

The PPO algorithm exhibited higher stability and more consistent performance across learning, while the SAC model demonstrated increased speed in learning and adaptation in different environments. The Sharpe ratio was slightly higher for the PPO model, indicating a better balance between returns and risk.

Overall, the SAC and PPO algorithms proved to be highly effective in the advanced stock trading system. The PPO model is preferred for its stability, while the SAC model may be favored in situations requiring more extensive adaptation or when a higher risk appetite is acceptable. The selection between the two algorithms ultimately depends on the specific requirements of the trading system, such as the need for consistent performance or the ability to adapt to dynamic market conditions.

During the exploratory data analysis, we encountered some issues, such as missing data, which was primarily due to stocks being delisted and no longer traded. As DRL models learn from the entirety of the state space, the most practical way of handling missing data was to remove it, which was also the case for stocks listed later than the beginning timestamp. While this data holds training value, future work should focus on research aiming to extract this value. As Woodford M. and Xie Y. (2020)

suggest, "The most reasonable method to resolve is to backfill and forward fill with a monetary price of zero." Additional analysis was performed to check for duplicate values and format errors.

Our current setup operates within a paper trading framework, which closely mimics real-world market conditions but does not fully account for certain factors that can impact trading performance. It is crucial to acknowledge that while our system demonstrated advanced capabilities in navigating the complex stock market environment, the simulated nature of the paper trading environment has its limitations.

Factors such as the cost of executing trades and slippage are not entirely considered in our current setup. These can have a non-marginal impact on the system's overall performance in real-world trading scenarios. The promising results obtained in our paper trading environment may not directly translate to live trading, as the market inefficiencies and additional costs associated with real-world trading can significantly influence the outcome.

Regardless, the performance metrics obtained in our simulated environment serve as a valuable proof-of-concept, demonstrating the potential of integrating DRL, FNN, and GA in developing an advanced stock trading system. However, to gain a more accurate representation of the system's performance, future iterations of the model should carefully evaluate and incorporate these real-world factors.

The insights gained from this study provide a solid foundation for further research and development. By refining the model's architecture, incorporating additional market factors, and testing its performance in more realistic trading scenarios, we can bridge the gap between our paper trading results and real-world applications. This will enable

us to better assess the system's robustness and adaptability to live market conditions and make necessary adjustments to optimize its performance in practical trading environments.

Visualizations

Figure 1: High Level Diagram Flow:

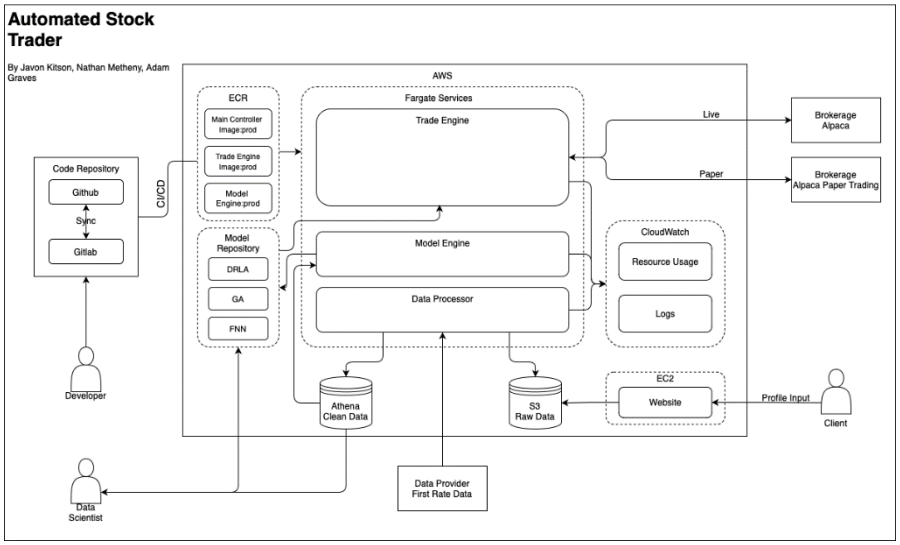


Figure 2: Diagram Of Trading Flow:

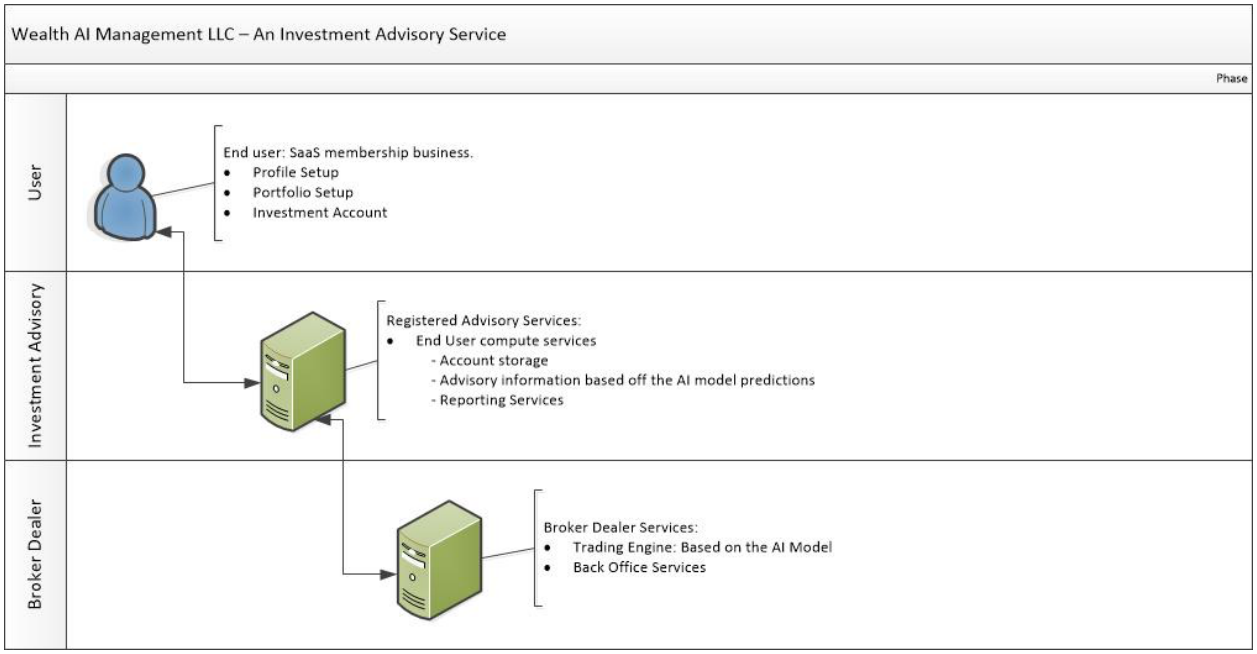
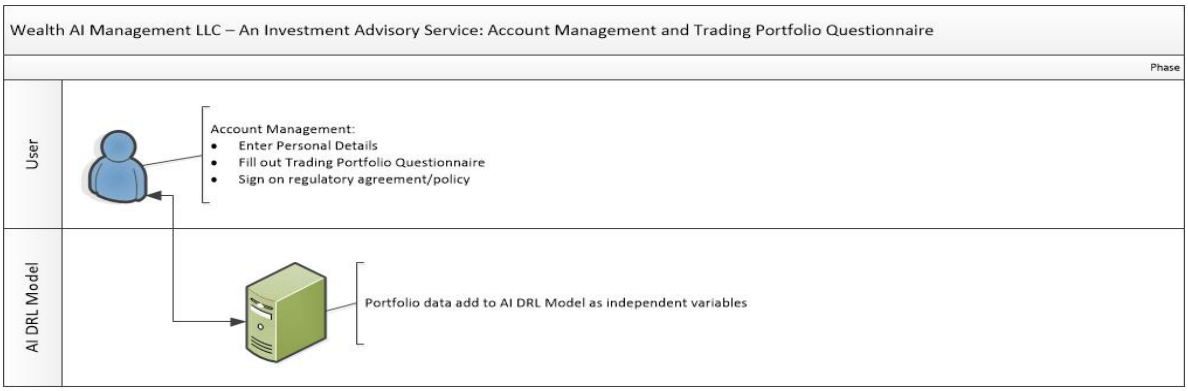


Figure 3: Profile Data



Portfolio Fields

- Name
- Social Security number or taxpayer identification number
- Address
- Telephone number
- E-mail address
- Date of birth
- Driver's license, passport information, or other government-issued identification
- Employment status and occupation
- Whether you are employed by a brokerage firm
- Annual income
- Other investments
- Financial situation and needs
- Tax status
- Investment experience and objectives
- Investment time horizon
- Liquidity needs and tolerance for risk
- Financial and trading record
- Net worth
- Trading experience
- Financial knowledge

Figure 4: Profile Form

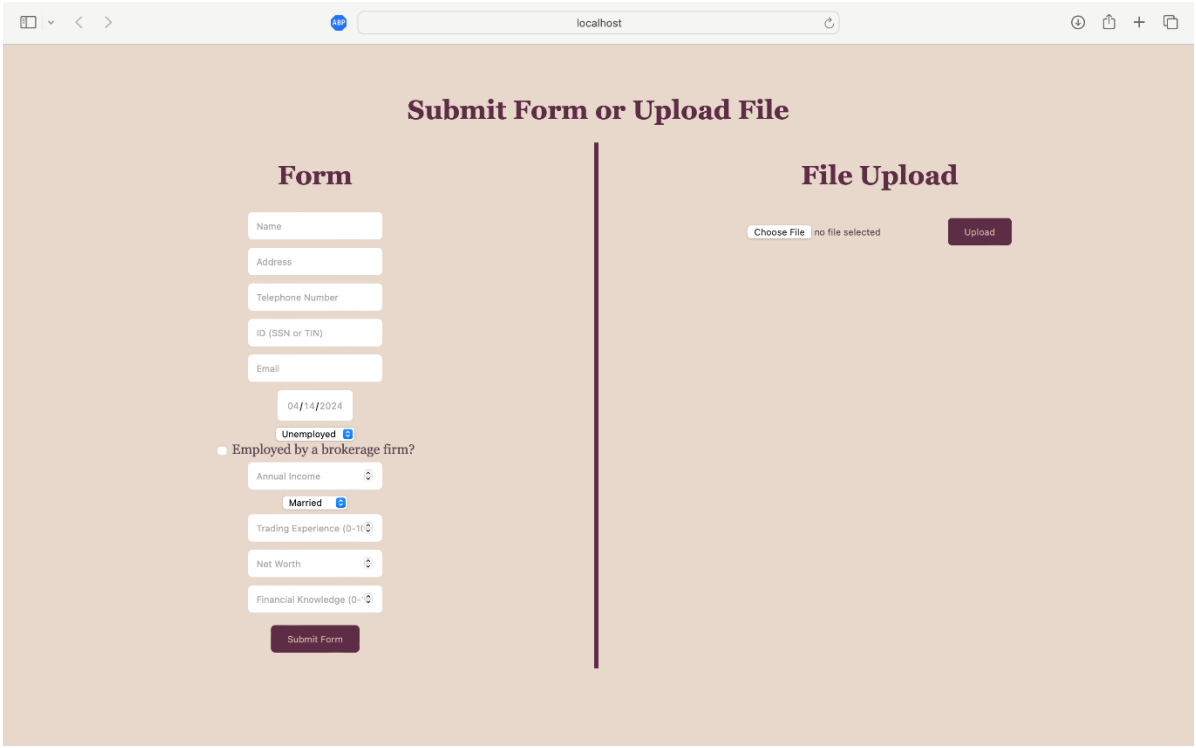


Figure 5: Heatmap Plot for Data Field Correlation

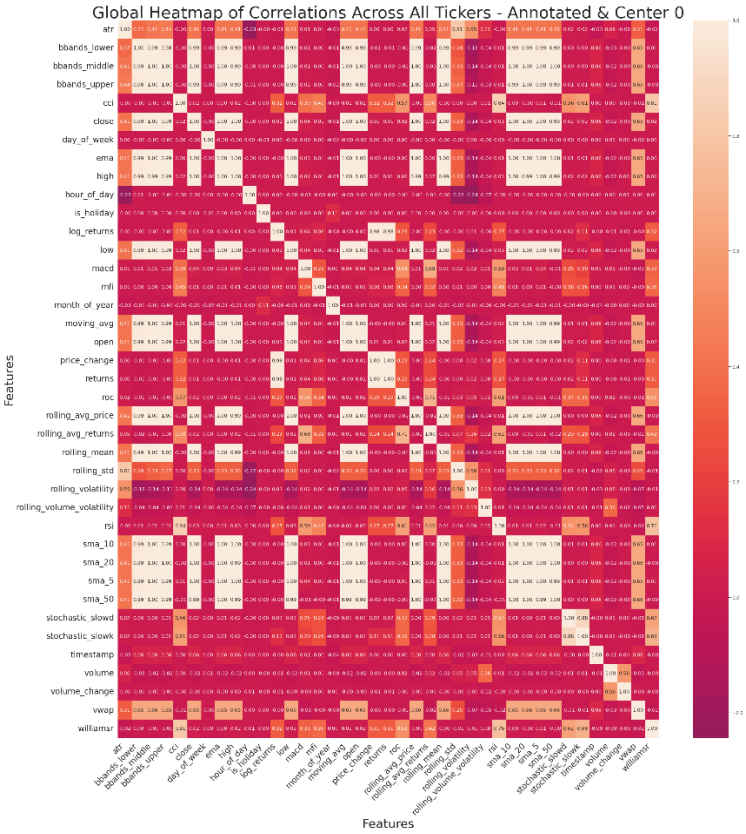


Figure 6: PPO Performance

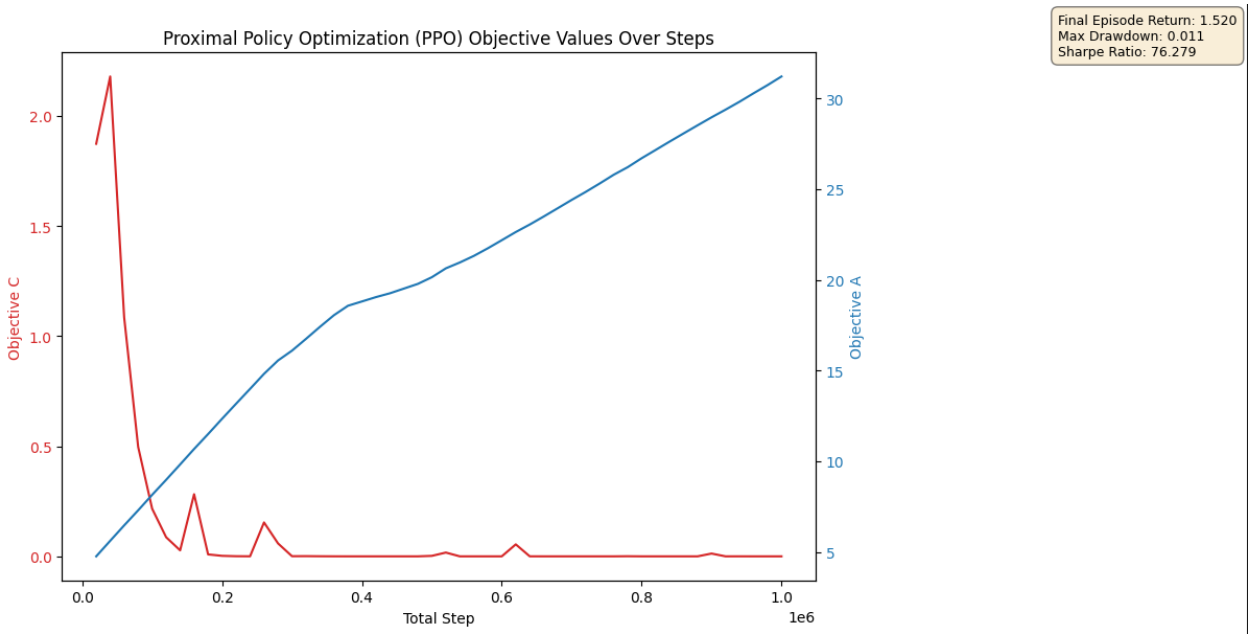
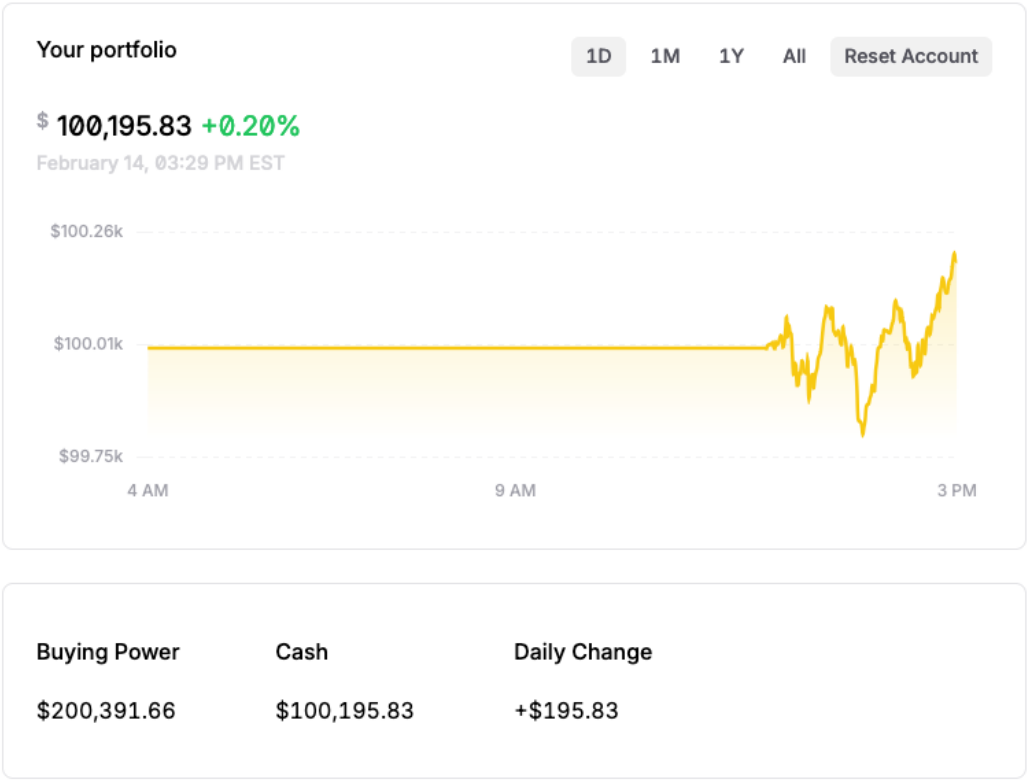


Figure 7: Trading Report per Portfolio



References

1. AI4Finance-Foundation. (n.d.). FinRL: A deep reinforcement learning library for automated stock trading in quantitative finance. GitHub. Retrieved [2024], from https://github.com/AI4Finance-Foundation/FinRL
2. Chen, Y., & Xiao, J. (2023). Target search and navigation in heterogeneous robot systems with deep reinforcement learning. arXiv preprint arXiv:2308.00331.
3. First Rate Data. (2023). Historical Stock Data [Data set]. Retrieved from https://www.firstratedata.com
4. Haarnoja T., Zhou A., Abbeel P., & Levine S., Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, Taken from: extension://efaidnbmnnnibpcajpcglclefindmkaj/https://proceedings.mlr.press/v80/haarnoja18b/haarnoja18b.pdf
5. Haarnoja T., Pong V., Hartikainen K., Zhou A., Dalal M., & Levine S.(2018), Actor Critic—Deep Reinforcement Learning with Real-World Robots, Taken from: https://bair.berkeley.edu/blog/2018/12/14/sac/
6. Heeswijk W., PhD, Proximal Policy Optimization (PPO) Explained, November 29, 2022.
7. Lin, C. C., & Marques, J. A. (2023). Stock market prediction using artificial intelligence: A systematic review of systematic reviews. https://doi.org/10.2139/ssrn.4341351

8. Nan, A., Perumal, A., & Zaiane, O. R. (2022). Sentiment and knowledge based algorithmic trading with deep reinforcement learning. Lecture Notes in Computer Science, 167-180. https://doi.org/10.1007/978-3-031-12423-5_13
9. Woodford, M., & Xie, Y. (2020). Fiscal and monetary stabilization policy at the zero lower bound: Consequences of limited foresight. https://doi.org/10.3386/w27521
10. Yang, H., Liu, X., Zhong, S., & Walid, A. (2020). Deep reinforcement learning for automated stock trading. Proceedings of the First ACM International Conference on AI in Finance. https://doi.org/10.1145/3383455.3422540
11. Yarats, D., & Kostrikov, I. (2020). Soft Actor-Critic (SAC) implementation in PyTorch. GitHub. https://github.com/denisyarats/pytorch_sac