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

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, exemplified by platforms like 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.

Despite these advancements, such systems often grapple with the intricate dynamics of the stock market, struggling to adapt to its volatile nature. A significant challenge lies in 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. This oversight can hinder their ability to fully grasp and react to the multifaceted and interconnected nature of market dynamics.

Our objective is to demonstrate the application of Deep Reinforcement Learning (DRL) in the domain of stock trading. For these type of algorithms, a DL model is more accurate than an ML model and performs exceptionally well on unstructured data.

However, it also needs a massive amount of training data and expensive hardware and software (Jakhar & Kaur, 2020). By leveraging advanced machine learning techniques, we aim to develop a system capable of making informed trading decisions autonomously. This involves the creation of a model that can analyze historical stock data, understand market trends, and execute trades with the goal of maximizing returns. The demonstration will cover the setup, training, and evaluation of the DRL model, showcasing its potential to outperform traditional trading strategies.

In addition to the technical implementation, we will explore the theoretical foundations of reinforcement learning and its suitability for financial markets. This includes discussing the challenges of applying DRL in a highly volatile environment, such as stock trading, and the strategies used to mitigate these risks. Our demonstration aims to provide a comprehensive overview of how deep reinforcement learning can be utilized to innovate in the field of stock trading, offering insights into both its capabilities and limitations.

Implementation of a robust dataset from First Rate Data with a dataset of 10120 tickers including their relevant trading values will be used to build the models on.

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.

The 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) and derived technical indicators for a diverse range of stocks over 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 stock trading, being the open, high, low, close, volume all numeric values as related to a timestamp.

In addition we have augmented variables that were derived from the original stock price data and included numeric fields.

We have another dataset that is of client input that consists of the fields to build a client account. This data is used to calculate the risk tolerance the client will be assigned. The calculations are not AI related and are based off a combination of, Age, Investing experience, and net worth.

The original variables, such as price and volume data, were directly related to our project goal of developing a DRL model for stock trading. These variables provided the foundation for the model to learn patterns and make trading decisions. The augmented variables, such as technical indicators, offered additional insights into market trends, momentum, and potential reversal points, which enhanced the model's predictive learning. The risk tolerance is categorized into three levels which have the 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 4)

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. Our project focuses on the application of Deep Reinforcement Learning (DRL) in stock trading, aiming to create an autonomous system that can learn from historical data and adapt to changing market conditions.

Traditionally, stock trading strategies have relied on fundamental analysis, technical analysis, and human expertise. Fundamental analysis involves evaluating a company's financial health, market position, and growth prospects to determine its intrinsic value. Technical analysis, on the other hand, focuses on studying historical price and volume data to identify patterns and trends that may indicate future price movements. Human traders use a combination of these approaches, along with their experience and intuition, to make trading decisions.

Numerous academic articles discussing the use of DRL models to automate stock trading activity are available. One example is an academic research of an Electronic Trading System is the research paper: "Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy" by Yang H., Yang X., Zhong S., &

Walid A. (2020), in which they propose an ensemble strategy combining Proximal Policy Optimization (PPO), Advantage Actor Critic (A2C), and Deep Deterministic Policy Gradient (DDPG). This approach integrates the strengths of these three actor-critic-based algorithms, aiming to create a robust system that adapts to various market conditions. In a way like ours but we are using a combination of Feedforward Neural Network (FNN), Soft Actor-Critic (SAC) and Proximal Policy Optimization (PPO) for the training of the Stock Trading section. 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.

This is a great example of how there are multiple model for the same solutions. Both our systems are advanced, unlike single-model systems. This ensemble method leverages the collective intelligence of multiple models to improve decision-making accuracy and adaptability in the dynamic 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. The foundation of this was the ability to trade on the condition of the limit price entered.

However, these methods have limitations in capturing the complex dynamics of the stock market and adapting to changing market conditions. Specifically, these techniques fail to consider the entirety of the state space, where all other stocks and their corresponding patterns should be considered. We find that Deep Reinforcement Learning (DRL) provides significant alpha, being that the stock trading strategies learned have resulted in returns that are significantly higher than those of the market average or a relevant benchmark. “In recent years, significant progress has been made in solving challenging problems across various domains using deep reinforcement learning (RL)”. (Henderson H., Islam R., Bachman P., Pineau J., Precup D, & Meger D., February 2018Article No.: 392Pages 3207–3214).

DRL combines deep learning with reinforcement learning, enabling an agent to learn optimal actions through trial-and-error 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.

These DRL models are also used in the world of Robotics. These models have the ability to learn robust policies bringing robustness to hyperparameters, and effective performance in a variety of simulated environments. In a research paper “Soft Actor Critic—Deep Reinforcement Learning with Real-World Robots” by Haarnoja T., Pong V., Hartikainen K., Zhou A., Dalal M., & Levine S. (2018), they have the in deep discussion about the valuable properties of the Soft Actor-Critic (SAC) algorithm. In this they used models to train a robot to move, a 3-finger dexterous robotic hand to manipulate an object, and 7-DoF Sawyer robot to stack Lego blocks.

In the research paper: “Sentiment and Knowledge Based Algorithmic Trading with Deep Reinforcement Learning” by Nan A., Perumal A., & Zaiane O. (2021), they incorporated additional exterior factors that are prone to very frequent changes and often these changes cannot be inferred from the historical trend alone. For this they incorporated the Partially Observable Markov Decision Processes (POMDP). This will take into account activities outside the realm of trading stocks, such as, a trading data center getting destroyed, a scenario that was actual on September 11th.

In our architecture utilizing the Genetic Agent (GA) to select a subset of stocks from a larger pool of stocks based on a predefined objective, is in-line with the strategy based off the client portfolio input. This also 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. These are well suited models for this environment. 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

The approach taken for this project is an ensembled one employing a combination of Deep Reinforcement Learning (DRL), a Feedforward Neural Network (FNN), and a Genetic Algorithm (GA) for stock trading, price prediction, and portfolio optimization, respectively. The DRL model is responsible for making trading decisions based on market conditions, while the FNN model is used to predict future stock prices, which serves as additional input to the DRL model. An additional dataset for the individual is created from a portfolio questionnaire input. From this dataset a unique id is identified, and a risk factor is calculated. Finally, the GA is responsible for structuring the portfolio for an individual with optimal performance based on the trading objective.

Portfolio Data, is from an individual’s input, where the unique id is defined. A risk factor is calculated based on the data fields from the data entry. The logic for the calculation of the risk factor is based on net worth, trading experience, individual’s age, and trading goals. The values range 0-30, and split to three ranges:

- minimum drawdown: 0-10
- max return: 11-20
- Min drawdown + max return: 21-30

The Genetic Algorithm (GA) followed a standard evolutionary process which is encapsulated in the GeneticAlgorithm class that incorporates Initialization of portfolios, Fitness Evaluation of symbols calculating the drawdowns, Selection to serve as the parent order, Crossover to create children orders, Mutation based on probabilities, Output results of trades, and Termination after a satisfactory solution was found.

The time_interval, start_date, and end_date variables were all aligned with the inputs used for the FNN and DRL training. The GA returned the best individual portfolio found, along with its corresponding returns and drawdown. This constrained portfolio was then used for downstream training and trading.

The Feedforward Neural Network (FNN) model architecture consists of an input layer, multiple hidden layers, and an output layer. The notable architectural design choices include the Input layer size which gets determined by the number of features in the input data, a list of possible hidden layer structures being [(32, 16), (64, 32), (128, 64), (256, 128)] allowing for flexible hyperparameter selection based on best results, Output layer size set to 1, Dropout regularization with a default of 0.5, Batch normalization, and an Activation function of ReLU,

The training of the FNN model in stock trading involves several key steps. Initially, historical stock data is downloaded and divided into training and validation sets, typically with an 80/20 split. The model is then set up with its architecture and utilizes Huber Loss (Huber F., 1964) as the loss function to balance between Mean Absolute Error and Mean Squared Error. Training involves using the Adam optimizer (Diederik P., Kingma, & Ba J. 2017) to adjust weights over 10 epochs and a batch size of 64, where after each epoch, the model's performance is evaluated on the validation set to ensure it generalizes well on new data.

Hyperparameter tuning and architectural adjustments are performed using Optuna (Akiba T., Sano S., Yanase T., Ohta T., & Koyama M. 2019) to optimize the FNN model's performance. Optuna efficiently searches the hyperparameter space and finds the best combination of learning rate and hidden layer sizes that minimize the validation loss. The optimal hyperparameters are then used to train the final FNN model.

We experimented with two popular Deep Reinforcement Learning (DRL) algorithms for stock trading; Soft Actor-Critic (SAC) and Proximal Policy Optimization (PPO). Both algorithms are designed to learn optimal trading strategies by interacting with the market environment and adapting to changing market conditions. After obtaining the optimized portfolio from the GA stage, the selected subset of stocks was used to train the DRL agents (SAC and PPO) to learn optimal trading strategies. The DRL agents were trained using the same experimental setup and architectural choices as previously described.

The key difference here is that the DRL agents were trained specifically on the optimized portfolio returned by the GA. This focused the agents' learning on a more promising subset of stocks, potentially leading to better trading performance and less computationally expensive training.

In the Soft Actor-Critic (SAC) model's architecture, key design features include the use of multi-layer perceptrons (MLPs) with ReLU activation for both actor and critic networks. The actor network's output divides into mean and log-standard deviation, forming a Gaussian distribution for action decisions. Meanwhile, the critic network provides two Q-value estimates, enhancing stability and minimizing overestimation bias.

In the training process of the SAC algorithm, the agent interacts with its environment to gather experiences, which include the states, actions, rewards, and subsequent states, over a defined horizon length. The collected experiences are used to update the critic network by minimizing the mean squared error between the predicted Q-values and the target Q-values.

The actor network is updated using the critic network's Q-values to maximize the expected future rewards, while the target networks for the critic are updated using a soft update mechanism to stabilize learning.

Hyperparameters such as the learning rate set to 0.0001, discount factor gamma set to 0.985, and the entropy coefficient alpha of 0.2, setting the target networks to updated gradually with tau set to 0.5, with two hidden layers [256, 256], were tuned to optimize the performance of the SAC algorithm.

Proximal Policy Optimization (PPO) is an on-policy DRL algorithm (Schulman J., 2017) that has shown impressive performance in various domains, including stock trading. The PPO architecture also consists of an actor network and a critic network.

The actor network in PPO is responsible for selecting actions based on the current state. It takes the state as input and outputs the mean and log-standard deviation of a Gaussian distribution, similar to SAC. The action is then sampled from this distribution. The critic network in PPO estimates the value of each state. It takes the state as input and outputs a single value estimate.

The PPO algorithm trains through a specific process where the agent first interacts with its environment to gather data (like states, actions, and rewards) over a certain period. This data then informs updates to the actor and critic networks within the algorithm. The actor network updates aim to maximize future rewards without straying too far from the previously established policy. In contrast, the critic network updates focus on reducing the discrepancy between predicted and actual state values, using mean squared error as the measure.

Hyperparameters such as the learning rate set at 0.00025, discount factor gamma set to 0.01, and batch size set to 64, and the clip range for the PPO objective can be adjusted to optimize the performance of the PPO algorithm.

In our experiments, we trained both the SAC and PPO agents using a rollout buffer to store the collected experiences. The agents interact with the stock market environment for a specified number of iterations, and the networks are updated using the collected experiences.

We optimized the models by tuning various hyperparameters, such as the learning rate, discount factor, epochs, batch sizes, 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. 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. The performance metrics, such as cumulative returns and Sharpe ratio, were used to compare the effectiveness of the SAC and PPO algorithms.

Results & Conclusion

The structuring of our advanced stock trading system into separate processing processes, leveraging Deep Reinforcement Learning (DRL), Feedforward Neural Network (FNN), and Genetic Algorithm (GA), yielded promising outcomes. Utilizing technical analysis and the system's architecture we designed the system as a multiple model architecture to make intelligent trading decisions, balancing risk based on the investor’s profile input and measured reward efficiently, which was reflected in the performance metrics we observed. The hypothesis is that this approach can adapt to market dynamics, make intelligent decisions, and produce an optimal portfolio to interact with.

With the profile input data the system calculates a risk factor and based on that the GA will generate the appropriate portfolio. The DRL models, Soft Actor-Critic (SAC) and Proximal Policy Optimization (PPO), were trained with optimized portfolios generated by the GA. This strategic combination allowed the system to focus on a subset of stocks, enhancing the 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.

The PPO model ran against the test data returned a value of 1.52, meaning it generated a 55% return on investment above the initial value. (Figure 5)

The assessment of our FNN model's performance demonstrated that the model is capable of generalizing effectively to new data, though there is potential for enhancement. The validation metrics, including Mean Absolute Error (MAE) and Mean Squared Error (MSE), indicate that the model successfully captures trends and patterns in stock price movements. However, these results also highlight opportunities for further refinement to minimize prediction errors and improve accuracy. This suggests that with additional tuning, the model could achieve even more reliable predictions. The system achieved an improved portfolio performance, as evidenced by the significant enhancement in the key financial metrics compared to baseline models. The integration of DRL and GA in our approach facilitated a nuanced understanding of the market dynamics, enabling the algorithm to adapt to the volatile nature of the stock market effectively.

Despite the advanced capabilities of our system, it exhibited a higher validation loss than anticipated, which could be attributed to the complex and unpredictable nature of financial markets. The discrepancies between the predicted and actual stock prices underscore the challenges in modeling such dynamic systems. This observation suggests a potential overfitting to the training data or an underestimation of the market's complexity in the model's current configuration.

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. We found that scaling to a three-level reward would fit best for mitigating 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. The performance metrics, such as cumulative returns and Sharpe ratio, were used to compare the effectiveness of the SAC and PPO algorithms.

In conclusion, our project demonstrates the potential of integrating DRL, FNN, and GA in creating an advanced stock trading system capable of making informed and efficient trading decisions and adhering to compliance requirements. The observed results underscore the system's proficiency in handling the intricacies of stock market trading while adhering to constraints based on the profile data. Future work will focus on addressing the identified shortcomings through advanced optimization techniques and expanding the model's capabilities to encompass a broader spectrum of financial instruments and market conditions.

During the exploratory data analysis, we encountered some issues, such as missing data. In most instances, the missing data was because of the stock being delisted and no longer traded. Due to DRL models learning from the entirety of the state space, the most practical way of handling missing data was to remove. This was also the case for stocks being listed later than the beginning timestamp. While this data holds training value, future work should be focused on research aiming to solve for extracting this value. “The most reasonable method to resolve is to backfill and forward fill with a monetary price of zero.” (Woodford M., Xie Y., 2020). Additional analysis was performed in checking for duplicate values, and format errors.

For further enhancements we are looking at better tailoring the portfolio of investments to the client’s investment strategies. This can be accomplished by expanding the risk tolerance classes and a new model training.

References

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

Yarats, D., & Kostrikov, I. (2020). Soft Actor-Critic (SAC) implementation in PyTorch. GitHub. https://github.com/denisyarats/pytorch_sac

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

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

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/

Heeswijk W., PhD, Proximal Policy Optimization (PPO) Explained, November 29, 2022.

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

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

Visualization

Figure 1: High Level Diagram Flow:

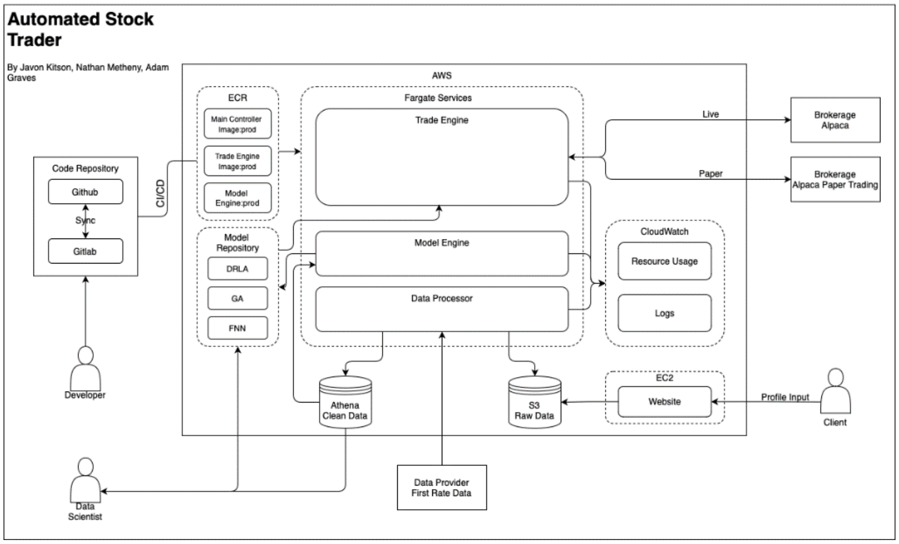
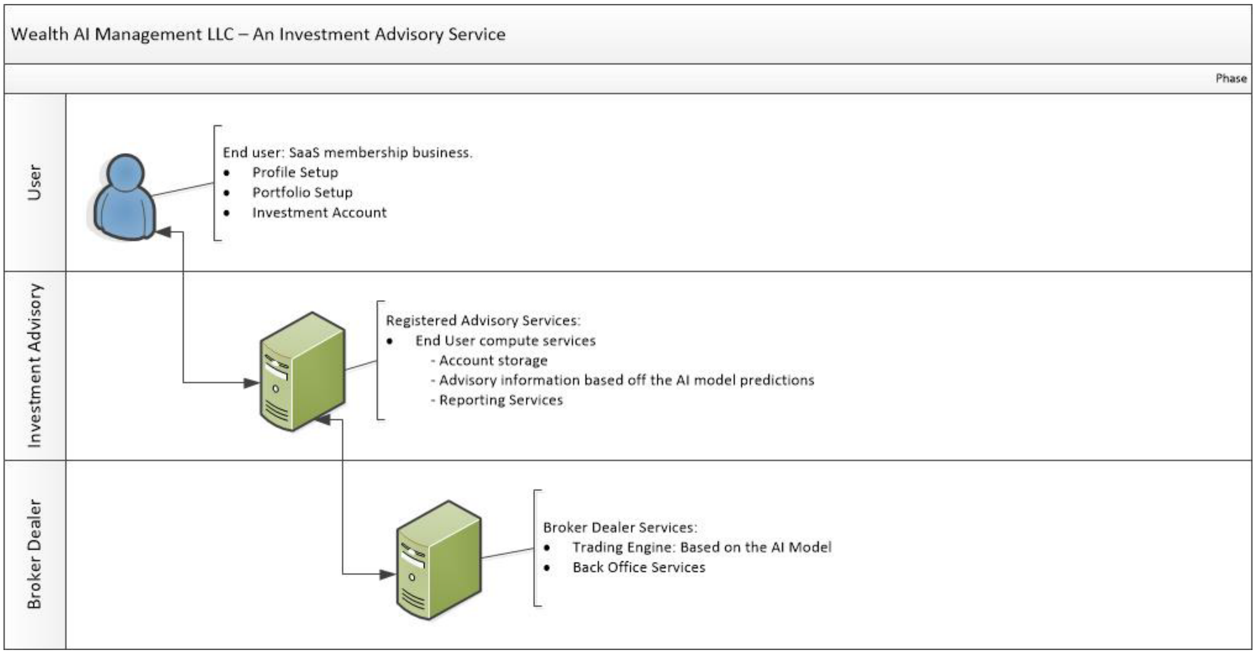


Figure 2: Diagram Of Trading Flow:



Wealth AI Management LLC – An Investment Advisory Service

End user: SaaS membership business.

- Profile Setup
- Portfolio Setup
- Investment Account

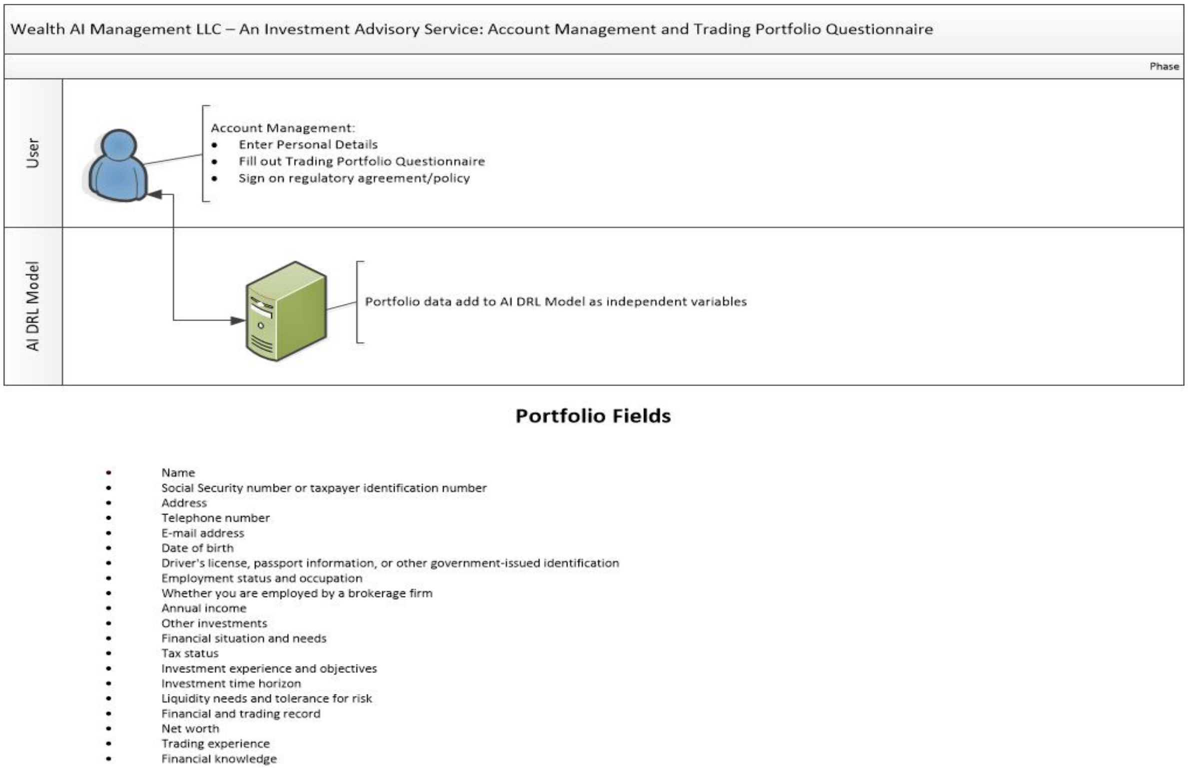
Registered Advisory Services:

- End User compute services
  - Account storage
  - Advisory information based off the AI model predictions
  - Reporting Services

Broker Dealer Services:

- Trading Engine: Based on the AI Model
- Back Office Services

Figure 3: Profile Data



Wealth AI Management LLC – An Investment Advisory Service: Account Management and Trading Portfolio Questionnaire

Account Management:

- Enter Personal Details
- Fill out Trading Portfolio Questionnaire
- Sign on regulatory agreement/policy

Portfolio data add to AI DRL Model as independent variables

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: Heatmap Plot for Data Field Correlation

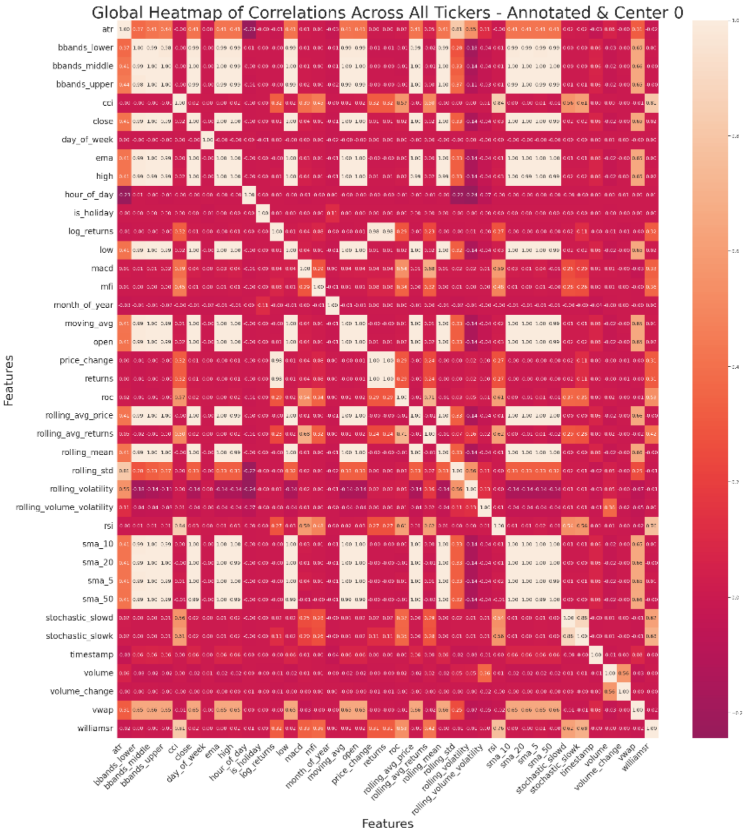
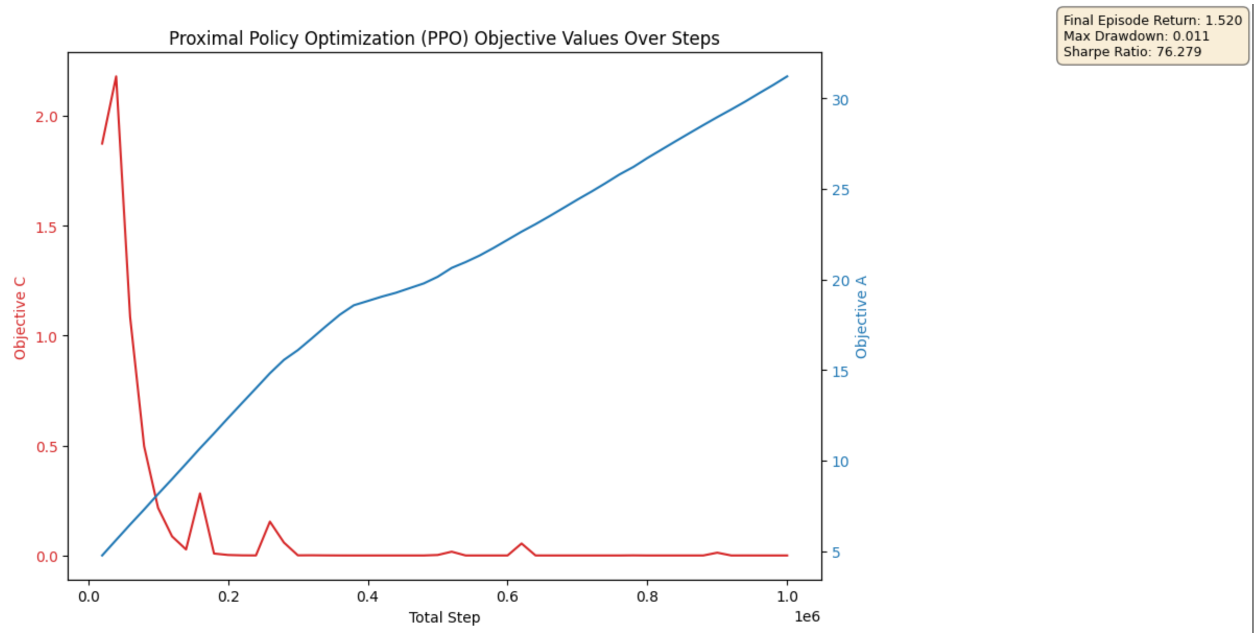
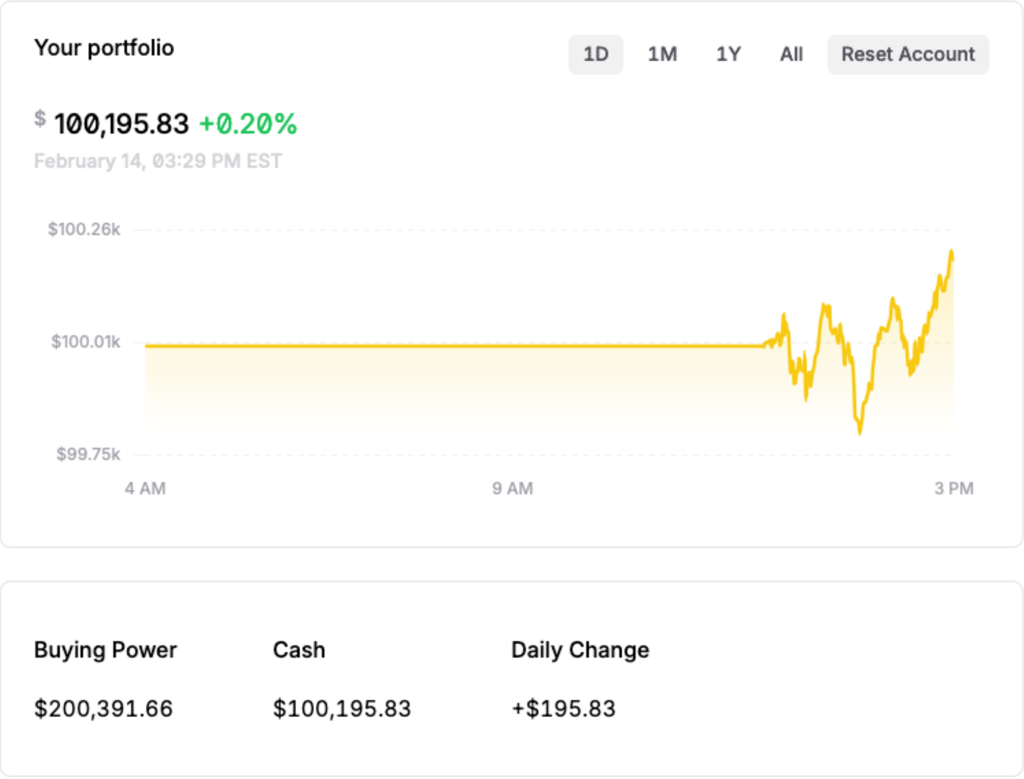


Figure 5: PPO Score



Final Episode Return: 1.520
Max Drawdown: 0.011
Sharpe Ratio: 76.279

Figure 6: Trading Report per Portfolio



Your portfolio

$ 100,195.83 +0.20%

February 14, 03:29 PM EST

Buying Power	Cash	Daily Change
$200,391.66	$100,195.83	+$195.83