Reinforcement Learning for Stock Recommendation

Ruoheng Du
rd3165, MS in Data Science, Columbia University
This is a project report for the EECS6892 course at Columbia University.

May 2025

1. Introduction

Stock recommendation is an important problem at the intersection of finance and machine learning. Unlike portfolio optimization, which aims to allocate capital across multiple assets, stock recommendation focuses on identifying a subset of stocks that are likely to align with investor objectives, such as return maximization and risk preference. Traditional recommendation approaches in this area have largely relied on collaborative filtering, content-based filtering, or supervised ranking models. More recently, reinforcement learning (RL) has emerged as a promising tool in financial applications due to its strength in sequential decision-making and dynamic adaptation to non-stationary stock market environments.

A growing body of literature has explored the use of RL for tasks such as algorithmic trading and asset allocation, however, the application of RL to personalized stock recommendation with the goal of adapting the recommendations to individual preferences, remains relatively underexplored. Moreover, the evaluation of such systems typically centers on return-based metrics, with less attention to individual specific style such as preference alignment.

This report provides a survey of existing methods for stock recommendation, with an emphasis on how reinforcement learning compares to more traditional approaches. This study classifies existing methods into three broad categories:

1. Rule-based or collaborative filtering methods that use predefined user profiles;
2. Feature-driven supervised learning models that leverage market and user signals;
3. Reinforcement learning-based approaches that learn optimal recommendation policies.

To supplement the result, this study implements a small-scale experimental comparison between one method from each category. Using simulated user profiles and real stock feature data, this study examines their relative performance in terms of average return, Sharpe ratio, and an alignment score to measure style consistency. While the experimental results are illustrative rather than conclusive, they help validate some of the observations and trade-offs discussed in the literature.

2. Related Work

This section reviews the three research streams that are closest to this project and then explains how this study follows naturally from them.

2.1. RL in General Recommendation Systems

Research in recommendation systems applies reinforcement learning (RL) mainly in three ways.

Contextual or Bandit-style Methods. Algorithms such as LinUCB [7] make a separate recommendation at each step based on the current user's context. They use confidence scores to decide whether to try new options or stick with known good ones. These methods respond quickly to feedback but focus only on short-term rewards, not long-term user satisfaction.

Value-based RL. Q-learning or DQN variants learn to assign a score to every possible action in a given situation. The agent then picks the action with the highest score. For example, SlateQ [5] extends this idea to recommend a whole set of items at once. This approach works well for ranking or displaying multiple items together, but it can become very slow when there are too many options, because the model needs to score all possible combinations.

Policy Gradient and Actor-critic Methods. These methods directly learn a recommendation policy by optimizing long-term user engagement. For example, Zhao et al. proposed an actor-critic model that generates page-wise recommendations based on user response signals [23], while later work extended this to list-wise recommendation using sequential decision modeling [24]. To address data sparsity and cold-start problems, Wang et al. incorporated knowledge graphs into the learning process, improving generalization in sparse environments [16].

All of these studies optimize media-centric rewards (clicks, watch-time) and do not deal with price or risk in stock markets. This gap motivates applying RL methods to domains like stock recommendation, where actions have delayed consequences and user preferences are dynamic and risk-sensitive.

2.2. RL for Trading and Portfolio Management

Reinforcement learning has been widely applied to trading tasks. Early work by Neuneier [9] used Q-learning for multi-period portfolio optimization under transaction costs. Later, O et al. [11] introduced a meta policy RL system that achieved better returns on the Korean stock market. As markets are often non-stationary, actor–critic methods like PPO and A2C have become more popular. Yang et al.[18] showed that combining these methods improves Sharpe ratios on U.S. equities. Other studies incorporate alternative data: Yang et al.[19] modeled news sentiment as hidden rewards, while Benhenda [1] proposed a CVaR-constrained PPO agent enhanced with risk signals from large language models. A recent survey [14] confirms that actor–critic RL has become the dominant framework in portfolio management, especially when combined with market features and risk control mechanisms.

2.3. RL for Stock Recommendation

Methods like collaborative filtering (CF) have been widely used. For example, Nourahmadi et al. [10] applied a CF approach based on stock co-movement and achieved strong results in the Tehran Stock Exchange. But unlike RL, these models rely on static relationships and don't handle user preferences or market shifts well. So, some recent studies treat stock selection as a recommendation problem using reinforcement learning. Shen et al.[13] built a graph-based RL agent that moves through a stock–industry network and outperformed a basic buy-and-hold benchmark. Zha et al.[22] used hierarchical RL to separate stock picking from portfolio rebalancing, which led to better returns in the Chinese market. Multi-agent systems are also common: Khonsha et al.[6] combined several agents—each focusing on a different type of input like technical indicators or news—while Yu et al.[21] created an ensemble of RL agents to adapt to changing market conditions. To support this kind of research, tools like FinRL-Meta [8] offer open-source environments and benchmarks for training and testing RL agents in financial settings.

2.4. Summary and Project Scope

Prior work shows that RL can enhance recommendation or trading decisions, yet comparisons are hard because studies use different data and metrics. To address these issues, this project benchmarks four models of increasing complexity on the same daily stock data:

1. a static rule-based selector,
2. a feature-engineered supervised model,
3. a single-agent RL recommender, and
4. a multi-agent RL recommender (one per user style).

The goal is to compare these models under identical conditions and assess how much value reinforcement learning actually adds over simpler methods—both in terms of return and user-style alignment.

3. Methods

The section introduces the environment design, user simulation, feature construction, and the models evaluated. Training procedures, reward structure, and evaluation metrics are also described.

3.1. Problem Formulation

This study models the interaction between a recommender and a user as a Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$. At each trading day t:

- the state $s_t \in \mathcal{S}$ contains current market statistics for the candidate stock, along with a scalar indicating the user's risk preference (see Section 3.3);
- the action $a_t \in \mathcal{A}$ is a discrete recommendation selected from a universe of 50 stocks;
- the environment transitions deterministically to s_{t+1} using the next-day market data and a probabilistic user style change;
- the reward $r_t$is defined as the one-day return of the recommended stock, scaled by the user's risk preference.

The agent's objective is to learn a policy $\pi_\theta(a|s)$ maximizing the expected discounted cumulative reward:

$$J(\theta) = \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t r_t \right],$$

where $\gamma = 0.99$ and T is the length of the test period.

3.2. Synthetic Environment and User Simulation

Stock Pool and Market Data. This study collects daily OHLCV data (which refers to daily opening price, daily highest price, daily lowest price, closing price, and volume) for 50 S&P-500 constituents from 1 Jan 2020 to 1 Apr 2025.

User Generation. This study simulates 10,000 synthetic users, each beginning with one of four investment styles:

- Conservative – seeks low volatility;
- Value – prefers low price-to-earnings stocks;
- Growth – favors high momentum;
- Trader – targets short-term swings.

Each user generates a trajectory of 30–60 steps, taking one action (buy or sell) per day. With 20% probability per step, the user may switch to a different style, requiring the agent to adapt to changing preferences.

Mapping Style to Risk Preference. Each discrete style is mapped to a scalar $p \in [0.8, 1.5]$, which is included in the state and used to scale the reward. This penalizes recommendations that mismatch the user's current risk preference.

Style	p
Conservative	0.8
Value	1.0
Growth	1.2
Trader	1.5

3.3. Feature Vector

At each time step, the environment returns a 9-dimensional feature vector for each stock–user pair:

$$o_t = [R_t, \sigma_{5,t}, \Delta_{3,t}, \mathrm{MA}_{10,t}, \sigma_{10,t}, V_t, \mathrm{Price}_t, p_t] \in \mathbb{R}^9,$$

where

- R_t – one-day simple return,
- $\sigma_{5,t}$ – 5-day volatility,
- $\Delta_{3,t}$ – 3-day price change (momentum),
- $\mathrm{MA}_{10,t}$ – 10-day moving average,
- $\sigma_{10,t}$ – 10-day volatility,
- V_t – trading volume,
- Price_t – closing price,
- p_t – current risk-preference scalar.

3.4. Recommendation Models

1. Static Top-K: Always recommends the $K = 5$ most frequently bought stocks in the training data. No user information is used.
2. Feature-based Random Forest: A supervised baseline predicting user actions using the 9-dimensional feature vector. Random-Forest (RF) [2] is an ensemble of decision trees trained on bootstrap samples. Averaging across many uncorrelated trees reduces variance and captures non-linear interactions, making RF well suited for stock data. Here, ground-truth actions are extracted from the synthetic logs, and the model uses 100 trees and default Gini impurity criterion.
3. Single-Agent PPO: A single PPO agent [12] is trained across all users to learn a unified policy. Proximal Policy Optimization (PPO) is a policy gradient method that uses a clipped objective to prevent large policy updates, helping to stabilize training under high-variance rewards, such as in financial tasks. The agent uses a two-layer MLP with 64 tanh units each. Training runs 100,000 steps with learning rate 3×10^{-4}.
4. Multi-Agent PPO: Four separate PPO agents are trained, one per user style. Each agent uses the same architecture and training settings as in the single-agent setup.

3.5. Reward Shaping and Training Details

At each step t, the agent receives a reward

$$r_t = \alpha \cdot p_t \cdot (\mathrm{Price}_{t+1} - \mathrm{Price}_t) + \beta \cdot \{\text{alignment}\},$$

where p_t is the user's risk preference scalar, and the alignment indicator equals 1 if the recommended stock matches the user's preference, and 0 otherwise. Coefficients α and β control the trade-off between financial return and personalization.

The dataset is split chronologically: data before 1 Jan 2025 are used for training; the first quarter of 2025 is reserved for testing. No action masking is applied—the agent must learn to infer which stocks align with each user's profile.

3.6. Evaluation Metrics

This study uses the following metrics, averaged across all users:

- Average Daily Return: Mean of r_t over the test episode.
- Sharpe Ratio: Mean divided by standard deviation of daily returns (risk-free rate assumed 0).
- Preference-Alignment Score: The proportion of test steps where the recommended stock matches the user's preference.

In the multi-agent PPO setup, each user is evaluated using the agent corresponding to their current style at each time step.

4. Results and Discussion

This section first presents empirical results for the four candidate models. Performance is evaluated using three metrics: average daily return, Sharpe ratio, and preference alignment. This study then interprets the findings in relation to prior literature, highlighting the strengths and limitations.

4.1. Empirical Results

Table 1 reports the performance of all four models across three evaluation metrics. Single-agent PPO achieves the highest return and Sharpe ratio, while the feature-based random forest achieves perfect alignment. Multi-agent PPO offers a strong balance between the stock return performance and user preference satisfaction.

Table 1: Test performance of four recommendation methods. Boldface highlights the best result per column.

Method	**Avg Return**	**Sharpe**	**Alignment**
Static Top-K	-80.47	-0.38	0.25
Feature-driven RF	0.12	0.01	**1.00**
Single-Agent PPO	**68.93**	**1.21**	0.30
Multi-Agent PPO	65.69	1.19	0.69

4.2. Insights and Interpretation

Static Top-K CF This non-learning baseline recommends the five most frequently purchased stocks from the training period, regardless of user preferences or market context. While straightforward, it performs poorly in the test period. A likely reason is that stock popularity does not imply future performance—especially in volatile markets. Empirically, we observe that several top-ranked stocks suffered significant drawdowns beginning in late 2022, meaning that past high purchase frequency actually led to poor returns in 2025. As a result, this method produces the lowest average return and Sharpe ratio, highlighting the risk of relying on static model in a shifting market environment.

Feature-driven Random Forest. This model leverages current market indicators and user risk preference to predict user actions. Functionally, it behaves like a contextual bandit, which makes decisions based solely on immediate features without considering long-term rewards. This leads to a perfect alignment score (1.00), since it directly conditions on user preference, but yields a Sharpe ratio near zero. The model tends to overreact to short-term noise and lacks the ability to recognize patterns that unfold over time. This is different from PPO, which optimizes the expected cumulative return and trades off short-term losses for future gains. So, models like RF struggle to adapt to cases when users change style mid-trajectory.

Single-Agent PPO. This model learns a shared policy across all users and investment styles. It achieves strong returns and a Sharpe ratio of 1.21, outperforming both static and feature-based baseline models. However, its preference alignment score drops to 0.30. Since the agent must accommodate four distinct styles with varying risk preferences, the learned policy tends to converge toward a generic strategy that may ignore individual user styles in favor of globally safe actions.

Multi-Agent PPO. By training a separate PPO agent for each investment style, this approach resolves the policy conflict in the single-agent setup. Each agent can specialize—e.g., the trader agent learns to exploit high-momentum, high-volatility stocks, while the conservative agent focuses on low-risk, stable performers. This specialization leads to a substantially higher alignment score (0.69 vs. 0.30) without sacrificing Sharpe or return. Although the approach requires four parallel training runs, the overall computational cost remains manageable.

Overall Comparison. The results reveal a clear performance hierarchy: static / rule-based models $<$ contextual / feature-based models $<$ single-agent PPO $\approx$ multi-agent PPO with style specialization. PPO-based methods consistently outperform others in terms of return and Sharpe ratio. Moreover, introducing style-specific agents significantly improves personalization, as reflected in the higher alignment score, without compromising stock return performance. These findings underscore the importance of combining sequential decision-making with user-aware modeling in personalized stock recommendation systems.

4.3. Experimental Support for Prior Literature

The empirical results reinforce several well-established findings in the literature, while also emphasizing the personalization perspective.

Contextual or Bandit-style Methods Contextual bandit methods are known for fast adaptation but limited temporal reasoning. The near-zero Sharpe ratio of our Random Forest baseline reflects this limitation: high short-term accuracy does not translate into consistent long-term returns, especially in volatile settings where user preferences shift over time.

Policy Gradient and Actor-critic Methods These are widely recognized as the most stable class of RL algorithms under noisy rewards, especially with PPO's clipped surrogate objective [12]. The single-agent PPO achieves a Sharpe ratio of approximately 1.2, validating its robustness in a personalized stock recommendation setting. When extending PPO to a multi-agent setup (one agent per user style), the algorithm remains stable and significantly improves alignment without sacrificing return.

Static Popularity-based Methods Static models, such as theTop-K model used in this study, again prove unreliable. A market shift after 2022 results in large drawdowns and a negative Sharpe ratio. This highlights how fixed, non-adaptive strategies can fail in dynamic market conditions.

In summary, the experiment confirm that clipped-objective PPO is the most robust baseline for stock recommendation, and further shows that style-specific policy partitioning provides meaningful personalization gains without compromising financial performance.

4.4. Limitations and Open Directions

While the study results demonstrate the promise of reinforcement learning in personalized stock recommendation, several limitations remain.

Sample inefficiency and hyperparameter sensitivity. Training even lightweight PPO agents required over 10^5 steps per style. Extending the time horizon or expanding the stock universe quickly increases computational cost. Prior benchmarks [15] show that many deep RL algorithms struggle with out-of-sample robustness, especially when trained on a single market condition. Model-based RL and offline pretraining on large replay buffers may help reduce interaction costs and improve stability, but remain underexplored in personalized financial recommendation.

Limited risk control and drawdown sensitivity. The reward in this study is based on daily return scaled by user risk preference, but does not explicitly penalize volatility or tail risk. Risk-sensitive objectives such as CVaR-PPO [1] have shown success in portfolio tasks, but have not yet been adapted to preference-aware settings. Conditioning risk control on user style is a promising next step.

Lack of interpretability and explainability. While multi-agent PPO improves alignment, its learned policies remain opaque. Users and compliance teams cannot easily trace why a volatile stock is recommended one day but dropped the next. Explainable RL techniques [3] that generate post-hoc rationales for each decision could be paired with preference embeddings to improve transparency and trust.

Scalability of multi-agent systems. Training a separate PPO agent for each investment style improves alignment, but becomes impractical when moving toward finer-grained personalization, such as per-user agents. As the number of user types grows, the training cost increases linearly, and policy generalization becomes difficult due to limited data per subgroup. Moreover, user preferences often change gradually rather than suddenly, suggesting that hard partitioning may be suboptimal. Recent work in personalized recommendation provides several promising directions. The Deep Adaptive Interest Network (DAIN)[4] dynamically tracks evolving user preferences with shared context-aware modules, reducing the need for one-policy-per-user setups. Meanwhile, meta-learning strategies like LiMAML[17] and PADM [20] personalize deep recommenders via lightweight per-user adaptation, enabling fast online updates without retraining full models. Integrating these ideas into a multi-agent RL framework—e.g., using a meta-learner to generate agent-specific embeddings or warm-start each policy—could maintain specialization while significantly improving scalability.

By addressing these limitations—improving sample efficiency, incorporating risk-aware objectives, enhancing interpretability, and scaling multi-agent training—future work can make personalized stock recommendation systems more practical, resilient, and user-aligned in the face of real-world market complexity.

5. Conclusion

This study investigates personalized stock recommendation system through reinforcement learning (RL), combining a literature survey with a controlled set of experiments. This study formulates stock recommendation as a preference-aware decision problem, where users follow different investment styles that may change over time. Four models are compared: a static rule-based model, a supervised feature-based random forest, a single-agent PPO, and a multi-agent PPO with one policy per style. Using synthetic user data, this study confirms that:

- RL-based agents clearly outperform static rule-based and featured-based baselines in Sharpe ratio and average return;
- Multi-agent PPO achieves about 2.3 times higher alignment with user styles than the single-agent version, while maintaining similar return.

These results suggest that RL is a promising approach for optimizing return under user uncertainty. Splitting policies by user style helps resolve conflicts in multi-user environments and improves personalization without hurting performance. In future work, it would be valuable to test robustness under changing market conditions, explore risk-aware reward designs, and investigate explainable RL methods to increase transparency.

References

[1] Mostapha Benhenda. Finrl-deepseek: Llm-infused risk-sensitive reinforcement learning for trading agents. *arXiv preprint arXiv:2502.07393*, 2025.

[2] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] Mao Guan and Xiao-Yang Liu. Explainable deep reinforcement learning for portfolio management: An empirical approach. In *Proceedings of the 2nd ACM International Conference on AI in Finance (ICAIF)*, pages 1–9. ACM, 2021.

[4] Shuaishuai Huang, Haowei Yang, You Yao, Xueting Lin, and Yuming Tu. Deep adaptive interest network: Personalized recommendation with context-aware learning. *arXiv preprint arXiv:2409.02425*, 2024.

[5] Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutilier. Slateq: A tractable decomposition for reinforcement learning with slate actions. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*. IJCAI, 2019.

[6] Sajad Khonsha, M. A. Sarram, and R. Sheikhpour. A robust concurrent multi-agent deep reinforcement learning based stock recommender system. *Journal of Electrical and Computer Engineering Innovations (JECEI)*, 13(1):225–240, 2025.

[7] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, pages 661–670. ACM, 2010.

[8] Xiao-Yang Liu, Ziyi Xia, Jingyang Rui, Jiechao Gao, Hongyang Yang, Ming Zhu, Christina Dan Wang, Zhaoran Wang, and Jian Guo. Finrl-meta: Market environments and benchmarks for data-driven financial reinforcement learning. In *36th Conference on Neural Information Processing Systems (NeurIPS 2022)*, 2022.

[9] Ralph Neuneier. Enhancing q-learning for optimal asset allocation. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1997.

[10] Marziyeh Nourahmadi, Ali Rahimi, and Hojjatollah Sadeqi. Designing a stock recommender system using the collaborative filtering algorithm for the tehran stock exchange. *Financial Research Journal*, 26(2):318–346, 2024.

[11] Jangmin O, Jongwoo Lee, Jae Won Lee, and Byoung-Tak Zhang. Adaptive stock trading with dynamic asset allocation using reinforcement learning. *Information Sciences*, 176(15):2121–2147, 2006.

[12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[13] Yifei Shen, Tian Liu, Wenke Liu, Ruiqing Xu, Zhuo Li, and Jia Wang. Deep reinforcement learning for stock recommendation. In *Journal of Physics: Conference Series*, volume 2050, page 012012. IOP Publishing, 2021.

[14] Vinay Singh, Shiuann-Shuoh Chen, Minal Singhania, Brijesh Nanavati, Arpan Kumar Kar, and Agam Gupta. How are reinforcement learning and deep learning algorithms used for big data based decision making in financial industries–a review and research agenda. *International Journal of Information Management Data Insights*, 2:100094, 2022.

[15] Marc Velay, Bich-Liên Doan, Arpad Rimmel, Fabrice Popineau, and Fabrice Daniel. Benchmarking robustness of deep reinforcement learning approaches to online portfolio management. *arXiv preprint arXiv:2306.10950*, 2023.

[16] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. Knowledge graph convolutional networks for recommender systems. In *Proceedings of the 2019 World Wide Web Conference (WWW)*, pages 3307–3313. ACM, 2019.

[17] Ruofan Wang, Prakruthi Prabhakar, Gaurav Srivastava, Tianqi Wang, Zeinab S Jalali, Varun Bharill, Yunbo Ouyang, Aastha Nigam, Divya Venugopalan, Aman Gupta, et al. Limaml: Personalization of deep recommender models via meta learning. *arXiv preprint arXiv:2403.00803*, 2024.

[18] Hongyang Yang, Xiao-Yang Liu, Shan Zhong, and Anwar Walid. Deep reinforcement learning for automated stock trading: An ensemble strategy. In *Proceedings of the ACM International Conference on AI in Finance (ICAIF)*, pages 1–8. ACM, 2020.

[19] Steve Y. Yang, Yangyang Yu, and Saud Almahdi. An investor sentiment reward-based trading system using gaussian inverse reinforcement learning algorithm. *Expert Systems with Applications*, 114:388–401, 2018.

[20] Runsheng Yu, Yu Gong, Xu He, Yu Zhu, Qingwen Liu, Wenwu Ou, and Bo An. Personalized adaptive meta learning for cold-start user preference prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4461–4469, 2021.

[21] Xiaoming Yu, Wenjun Wu, Xingchuang Liao, and Yong Han. Dynamic stock-decision ensemble strategy based on deep reinforcement learning. *Applied Intelligence*, 53:2452–2470, 2023.

[22] Lijun Zha, Le Dai, Tong Xu, and Di Wu. A hierarchical reinforcement learning framework for stock selection and portfolio. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2022.

[23] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. Deep reinforcement learning for page-wise recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys)*, pages 95–103. ACM, 2018.

[24] Xiangyu Zhao, Liang Zhang, Long Xia, Zhuoye Ding, Dawei Yin, and Jiliang Tang. Deep reinforcement learning for list-wise recommendations. In *Proceedings of the 1st Workshop on Deep Reinforcement Learning for Knowledge Discovery (DLRKD)*, Anchorage, AK, USA, 2019. ACM.

A. User Style Simulation and Transition Dynamics

This study simulates dynamic user preferences over four investment styles: conservative, value, growth, and trader. Each user follows a discrete Markov process with a 20% probability of switching styles at each time step. The empirical transition matrix is shown in Figure 1.

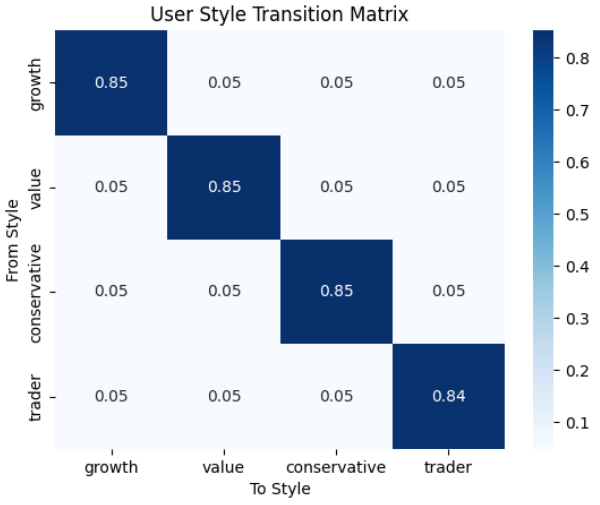

Figure 1: User style transition probability matrix.

As shown, over 80% of users remain in their current style at each step, with uniform probability of switching to one of the three alternative styles.

B. Stock Universe and Price Dynamics

The environment includes 50 U.S. equities across multiple S&P 500 sectors. Figure 2 shows the price dynamics of all stocks from Jan 2020 to Apr 2025.

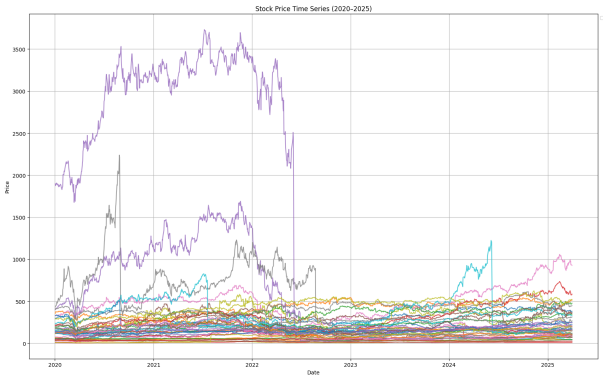

Figure 2: Raw price evolution of the stocks (2020–2025).

Notably, the universe exhibits high variance in stock behavior, with a few highly volatile assets dominating the upper range while the majority remain relatively stable within specific price range.

C. Feature Engineering

Each user–stock interaction at time t is represented as a 9-dimensional feature vector:

$$o_t = [R_t, \sigma_{5,t}, \Delta_{3,t}, \mathrm{MA}_{10,t}, \sigma_{10,t}, V_t, \mathrm{Price}_t, p_t] \in \mathbb{R}^9$$

where each component is defined as follows:

- One-day return:

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}}$$

- 5-day rolling volatility:

$$\sigma_{5,t} = \sqrt{\frac{1}{5}\sum_{i=t-4}^{t}(R_i - \bar{R}_5)^2} \quad \text{with } \bar{R}_5 = \frac{1}{5}\sum_{i=t-4}^{t} R_i$$

- 3-day momentum:

$$\Delta_{3,t} = P_t - P_{t-3}$$

- 10-day moving average:

$$\mathrm{MA}_{10,t} = \frac{1}{10}\sum_{i=t-9}^{t} P_i$$

- 10-day volatility:

$$\sigma_{10,t} = \sqrt{\frac{1}{10}\sum_{i=t-9}^{t}(R_i - \bar{R}_{10})^2} \quad \text{with } \bar{R}_{10} = \frac{1}{10}\sum_{i=t-9}^{t} R_i$$

- V_t: number of shares traded by the user at time t
- Price_t: closing price of the selected stock
- Risk preference $p_t \in [0.8, 1.5]$: a scalar indicating the user's risk tolerance level

All features are standardized and missing values are filled using forward/backward filling with fallback to zeros.

D. Simulator Interface Pseudocode

```
Initialize environment with:
    - User behavior dataframe (style trajectory)
    - Stock universe (50 symbols)
    - Style-to-stock mapping

For each time step t:
    1. Construct state s_t:
        - Market features of candidate stocks
        - User risk preference scalar p_t
    2. Agent selects action a_t in stock universe
    3. Environment computes:
        - Daily return:
          r_price = p_t * (Price_{t+1} - Price_t)
        - Alignment bonus:
          r_align = 1 if a_t match style else 0
        - Final reward:
          r_t = alpha * r_price + beta * r_align
    4. Sample new user style with prob 0.2
       (optional transition)
    5. Advance to next market day - state s_{t+1}
    6. Check episode end (trajectory ends)

Return (s_{t+1}, r_t,
        terminated=False, truncated=done,
        info={"alignment": r_align})
```

E. Additional Results by Style

Table 2: Evaluation by user style using Multi-Agent PPO.

Style	Avg Return	Sharpe	Alignment
Growth	68.36	1.14	0.96
Value	96.33	1.49	0.83
Conservative	37.03	0.93	0.55
Trader	59.77	1.12	0.42

Table 2 reports the performance of the multi-agent PPO framework evaluated separately for each user style on the test

set. While all styles benefit from personalized learning, their returns and alignment vary substantially.

Users with value and growth preferences achieve the highest average returns and Sharpe ratios, reflecting more stable or upward-trending stock pools associated with these styles. In particular, the value-oriented agent achieves a Sharpe ratio of 1.49, the highest among all styles.

The conservative group obtains lower average return and alignment scores, likely due to more limited price movements and stricter alignment constraints. Similarly, the trader style exhibits relatively high volatility, resulting in strong returns but weaker alignment (0.42), suggesting the agent sometimes explores outside the preferred stock pool in search of high-yield opportunities.

F. Ablation Study: Risk Preference Encoding

To understand the impact of risk preference representation on model performance, we conduct an ablation study comparing two variants of the RL agent:

- Scalar risk encoding: a single continuous value $p_t \in [0.8, 1.5]$ representing the user's current risk tolerance, appended directly to the state vector.
- One-hot risk encoding: a 4-dimensional binary vector encoding the user's current investment style (growth, value, conservative, trader) as a one-hot indicator.

Table 3: Comparison of risk preference representations in Single-Agent PPO.

Encoding	**Avg Return**	**Sharpe**	**Alignment**
One-hot	63.96	1.15	0.29
Scalar	68.93	1.21	0.30

As shown in Table 3, using scalar risk encoding yields slightly higher average returns and Sharpe ratio compared to one-hot encoding. However, alignment scores are nearly identical across both methods.

This suggests that the scalar risk representation may provide a smoother input signal that facilitates better generalization, while the discrete one-hot encoding, although interpretable, may limit the model's ability to interpolate between user styles.