Group X Progress Report:
My Group's Project Name

First Author, Second Author, Third Author
{xiaot13,loux8,chens356}@mcmaster.ca

1 Introduction

Currently majority of the financial market prediction models are relying on traditional index digging, technical analysis, regression models. However, with modern machine learning techniques emerging in NLP, the industry see a huge potential in bringing complex models to analyze and predict the market movement.

Over the years, majority of the investors, including professionals, have struggled to consistently beat the S&P500 index. In this challenge, we also understand the effect of the efficient market hypothesis, which states that stock prices fully reflect all available information. Thus, it is impossible to consistently achieve higher returns than average market returns on a risk-adjusted basis, given that stock prices should only react to new information.

However, with the advancement of machine learning techniques, if we are able to implement these new techniques to analyze the market data, we may be able to uncover hidden patterns and profit. Since these are also considered as new information, the effective market hypothesis still holds.

Since we do not know the preprocessing steps of the features, it is difficult for the team to use traditional, statistical methods to give meaningful insightful analysis. Thus, we decided to use predominantly unsuperised learning methods to analyze the data and predict the S&P500 return.

The overall objective of our project is to predict the S&P500 return using the pre-processed data provided by Hull Tactical.

2 Related Work

Predicting stock prices in an unsupervised manner is more naturally framed as a sequential decision-making problem rather than a static supervised regression task. In this context, reinforcement learning (RL), and in particular the algorithm Proximal Policy Optimisation (PPO) (Schulman et al., 2017), offers a compelling framework. PPO is an on-policy actor–critic method that stabilises policy updates by clipping its surrogate objective, thereby limiting destabilising policy shifts in non-stationary, noisy environments such as financial markets.

Adaptive trading strategies using deep RL have been applied to portfolio management. For example, Huang et al. (Huang et al., 2020) apply deep reinforcement learning to portfolio allocation, including short-selling and arbitrage in continuous action spaces, showing that RL agents can outperform benchmarks in equity markets. Jiang et al. (Jiang et al., 2024) propose a model-free DRL framework for portfolio selection under dynamic market complexity, demonstrating improved performance in large-scale asset settings. On the risk-management front, Lam et al. (Lam et al., 2023) develop a unified risk-aware RL framework that uses coherent risk measures with non-linear function approximation, delivering regret bounds in risk-constrained trading scenarios. Finally, Soleymani & Paquet (Soleymani and Paquet, 2021) introduce a graph-convolutional RL architecture (DeepPocket) that models inter-asset correlations for portfolio optimisation and shows strong empirical performance.

Together, these studies underscore that combining PPO-style policy optimisation, temporal or graph encoders (e.g., recurrent nets or GCNs), and risk-adjusted reward functions offers a promising unsupervised framework for stock prediction and trading-policy development.

3 Dataset

The dataset used in this project is the same equity return prediction dataset referenced in the project proposal. An updated version of the dataset was released on November 6th. The update primarily expanded the number of observations and refined the calculation of forward returns. However, this

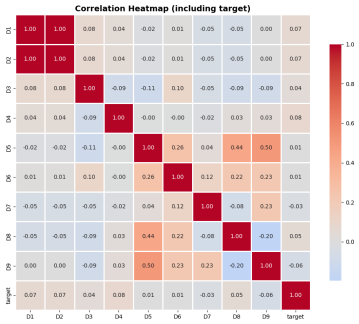

Figure 1: relationship between binary and target

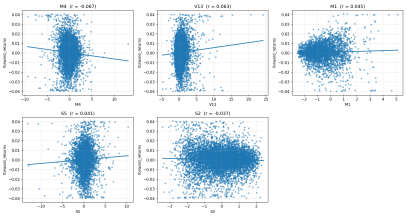

Figure 2: distribution of target variable

revision did not materially change the statistical properties of the data distribution, nor did it require changes to our modeling pipeline, since our learning framework does not rely on absolute scale but rather on relative directional signals.

The dataset consist of 8 different feature groups. These feature groups are:

- M* Market Dynamics/Technical features.
- E* Macro Economic features.
- I* Interest Rate features.
- P* Price/Valuation features.
- V* Volatility features.
- S* Sentiment features.
- MOM* Momentum features.
- D* Dummy/Binary features.

For example, a feature related to Price/Valueation will be denoted as column V1, V2, etc.

Some analysis are ran on the dataset to understand the basic nature of the data.

From observing Figure 1 and Figure 2, as expected in financial forecasting problems, the empirical distribution of features and target is extremely noisy and return is distributed closely to zero. Features released are all have a weak to no visible relationship with the target variable. Therefore, when training, we normalize the target variable to have zero mean and unit variance which make sure the loss will not diminish during backward propagation.

The dataset also contains missing values, particularly in indicators constructed from rolling windows of different lengths. To avoid discarding information, we do not remove rows with missing data. Instead, we replace `NaN` values with zeros and add binary missing indicator flags for each affected feature:

$$x_i^{(\text{filled})} = \begin{cases} 0, & \text{if } x_i \text{ was missing} \\ x_i, & \text{otherwise} \end{cases}$$

This preserves potential information carried by the absence of values. Data loading and cleaning follows our `load_data` and missing-indicator augmentation utility functions.

4 Features

We work directly with the numerical feature columns provided in the dataset. No manual feature engineering was performed beyond:

- Standardization of all numeric features via z-score normalization.
- Addition of missing-value indicator features for all originally `NaN`-carrying columns.
- Optional inclusion of a constant bias term during reinforcement learning.
- Optional removal of values too small in magnitude to reduce noise.

Because financial features often exhibit collinearity, we conducted correlation inspections to assess which features carry directional information relative to `target`. However, no workable linear correlation found in this dataset. The processed data is feeded to an reinforcement learning environment to generate state frames, then the agent will learn based on the rewards from the reinforcement learning environment.

5 Implementation

Our core modeling framework is a reinforcement learning (RL) agent based on Proximal Policy Optimization (PPO). Rather than predicting raw return

values, the task is formulated as a sequential decision problem. At each timestep, the agent selects one of three discrete actions:

$$a_t \in \{-1, 0, +1\}$$

representing `sell`, `hold`, or `buy` decisions.

5.1 Environment

We define a custom environment that sequentially feeds feature vectors over time. The reward reflects directional correctness rather than magnitude: Below are 2 different reward function implementation we experimented with:

- **Simple directional reward:**

$$r_t = a_t \cdot \Delta p_t$$

  where $a_t \in \{-1, +1\}$ is the action (sell / buy) and Δp_t is the price change.

- **Risk-adjusted directional reward:**

$$r_t = a_t \cdot \Delta p_t - \text{sign}(x, \Delta p_t)$$

  where $\text{sign}(x, y)$ is defined as:

$$\text{sign}(x, y) = \begin{cases} \Lambda x, & \text{if } x \cdot y < 0, \\ 0, & \text{otherwise.} \end{cases}$$

Currently we are still not satisfy with the reward function that we are currently using since it does not consider Sharpe ratio, trading cost on switching sides, and other risk factors. We also want to encourage the model to make more sell decision rather than hold or buy since the market is more tilted towards a bullish trend, and its ability to spot dips in the market is more valuable. We are actively working on design and test different reward functions to improve both return and risk of model strategy.

5.2 Policy and Value Networks

The agent maintains:

- A policy network $\pi_\theta(a_t|s_t)$ producing action probabilities.

- A value network $V_\phi(s_t)$ estimating expected cumulative reward.

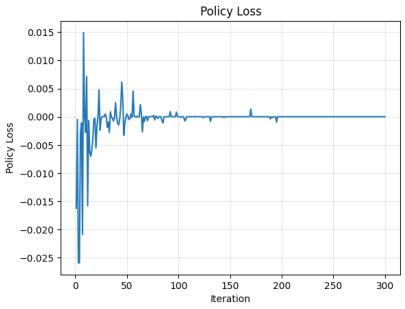

Figure 3: PPO policy loss over training iterations

Both networks are multilayer perceptrons with ReLU activations. The policy is trained using the PPO clipped surrogate objective:

$$L_\pi = -\mathbb{E}\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

The value network is trained via mean-squared error against return-to-go. In Future we will explore LSTM related architecture to enhance performance.

5.3 Training Procedure

Training alternates between:

1. Rollout: interact with the environment to collect trajectories.

2. Update: compute generalized advantage estimates (GAE) and apply PPO.

Although GPU acceleration is used for neural network forward and backward passes, rollout occurs step-by-step in Python. This creates a performance bottleneck where environment interaction becomes the dominant runtime cost. Parallel rollout environments are a planned optimization.

5.4 Baselines

To benchmark reinforcement learning performance, we also train three supervised models:

- Gradient-Boosted Trees.

- Multilayer Perceptron Regression (MSE objective).

These baselines help evaluate whether reward-based directional optimization provides an advantage over direct regression on target.

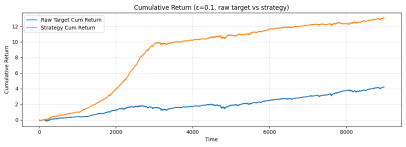

Figure 4: Model performance comparison

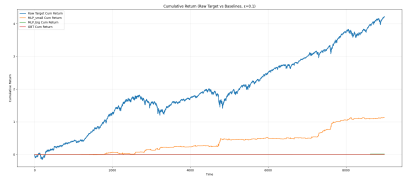

Figure 5: Baseline model performance comparison

6 Results and Evaluation

As showm in Figure 3, This is trained with batch size of 128, which is considered to be a small batch size. In this setup, sign accuracy reach 53.93% and MSE of 0.356282. The MSE is here is calculated against normalized target variable. Finally, we observe the policy loss converge after 100 iterations.

It is also observed with our latest experiment that with a large batch size 3000, the policy loss converge slower, with better results. However, we accidentally deleted the log file for that experiement therefore we can not show the plot here.

Unfortunately, the training process of the best model is not documented. We are unable to retrain before the end of this report it because the time constraint and GPU resource limitation.

But we are still able to measure our best model performance against the baselines.

Our baseline is two small MLPs regression and one gradient boosted tree model.

We understand one of the biggest drawback of implementing deep learning methods into financial market prediction is the inconsistency of the results and low explainability which leads to high risk. We see our best model is outperforming the baselines by a large margin but, the performance is not consistant. Since the best model is ran on the same datapoint over a long period of time, although the model is relavantly small, we believe the finding may still not be valid. A good result would be a consistant outperformance over the expected return over all time periods with a relevantly small model.

7 Feedback and Plans

- Improve the reward function by incorporating risk-adjusted returns (e.g., Sharpe ratio), trading costs, and additional risk controls.
- Experiment with different neural network architectures such as p-sLSTM and GRU to improve agent performance.
- Implement parallel rollout environments to accelerate training.
- Complete the evaluation pipeline for more systematic analysis and visualization.

The TA's feedback emphasized the noisy nature of financial time series data and the inherent difficulty of price prediction. In response, we designed a careful preprocessing pipeline to stabilize the data and ensure that the model receives consistent and meaningful input features. We also revised our reward function to be more robust to noise and large adverse movements in price. In future

7.1 Citations

7.2 References

Team Contributions

All teammates are involved in training models and contributed evenly to the project. There is no clear division of labor since all members are capable of doing all tasks, and we all did do the tasks together. All team members participated in weekly brainstorm meetings, discussed ideas. All team members contributed to algorithm deisgn, implementation, training, and evaluation.

References

Gang Huang, Xin Zhou, and Qing Song. 2020. Deep reinforcement learning for portfolio management. *arXiv preprint arXiv:2012.13773*.

Yifan Jiang, Xiang He, and Wei Zhang. 2024. Deep reinforcement learning for portfolio selection. *Decision Support Systems*. In press.

Tze Siang Lam, Abhinav Verma, Bryan Kian Hsiang Low, and Patrick Jaillet. 2023. Risk-aware reinforcement learning with coherent risk measures and nonlinear function approximation. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

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

Farzan Soleymani and Eric Paquet. 2021. Deep graph convolutional reinforcement learning for financial portfolio management: Deeppocket. *arXiv preprint arXiv:2105.08664*.