A ROBUST SFT-GRPO PIPELINE FOR HULL TACTICAL MARKET PREDICTION

ABSTRACT

Financial market prediction is characterized by high noise, non-stationarity, and evolving distributions. In this course project for the Kaggle Hull Tactical Market Prediction competition, we propose a two-stage end-to-end framework: **Supervised Fine-Tuning (SFT)** followed by **Generalized Reinforcement Policy Optimization (GRPO)**. Unlike traditional approaches that rely solely on regression loss, we decouple signal generation from decision-making. First, we perform rigorous feature engineering to select stable predictive factors. Second, we conducted a comparative analysis between Transformer-based (PatchTST) and MLP-based (N-HiTS) models, ultimately selecting N-HiTS for its superior capability to capture strong negative correlations in key factors. We generate "Rotten Apple" OOS signals to simulate realistic errors. Finally, we train a Policy Network (MLP) using Reinforcement Learning to optimize a comprehensive reward function ($PnL - Risk - Turnover$). Our approach achieved a public leaderboard score of **0.431**.

1 INTRODUCTION

Predicting stock market returns is a notoriously difficult task due to the low signal-to-noise ratio and the dynamic nature of financial data (Hull et al., 2017). The Hull Tactical Market Prediction competition challenges participants to forecast future returns using a provided set of proprietary features. A key challenge in such competitions is the disconnect between the surrogate loss function used in training (e.g., Mean Squared Error) and the ultimate financial objective (e.g., Profit and Loss, Sharpe Ratio).

To address this, we developed a pipeline that separates *signal generation* from *decision making*. We treat the base model's predictions not as the final answer, but as a probabilistic feature state for a Reinforcement Learning (RL) agent. This allows the agent to learn a policy that adapts to the base model's uncertainty, balancing risk and return dynamically.

Our contributions are as follows:

- A systematic feature selection process based on Information Coefficient (IC), Information Ratio (IR), and group return monotonicity.
- A data-driven model selection process where we identified that simple MLPs (N-HiTS) outperform complex Transformers (PatchTST) due to the presence of strong linear factors.
- A "Rotten Apple" data generation mechanism that produces unbiased Out-of-Sample (OOS) signals for RL training, preventing look-ahead bias.
- A GRPO-based Policy Network that optimizes a composite reward function of PnL, risk, and turnover.

2 DATA EXPLORATION AND FEATURE ENGINEERING

The dataset consists of anonymized financial features. Given the "black box" nature of the features, statistical analysis was crucial to identify those with genuine predictive power.

2.1 EVALUATION METRICS

We evaluated features using three key metrics:

1. **Information Coefficient (IC) & Rank IC**: The Pearson/Spearman correlation between the feature value and the forward return over a rolling window. A stable, high IC indicates predictive power.
2. **Information Ratio (IR)**: Defined as $IR = Mean(IC)/Std(IC)$, measuring the stability of the factor's predictive power.
3. **Grouped Returns**: We divided time periods into deciles based on feature values. A monotonic relationship between feature deciles and returns indicates strong discriminative ability.

2.2 ANALYSIS AND SELECTION

Our analysis revealed the presence of "Super Factors". Specifically, factors `E2` and `E3` exhibited a Rank IC of approximately **-0.15** with a negative Information Ratio (IR) exceeding **-2.3**. In quantitative finance, a Rank IC of 0.05 is typically considered strong; -0.15 indicates an exceptionally strong and stable negative linear relationship with returns.

As shown in Figure 1, the Cumulative IC for feature `E2` approximates a straight line with a negative slope, confirming this stable negative correlation. Similarly, Figure 1 demonstrates that `E3` exhibits excellent monotonicity in grouped returns.

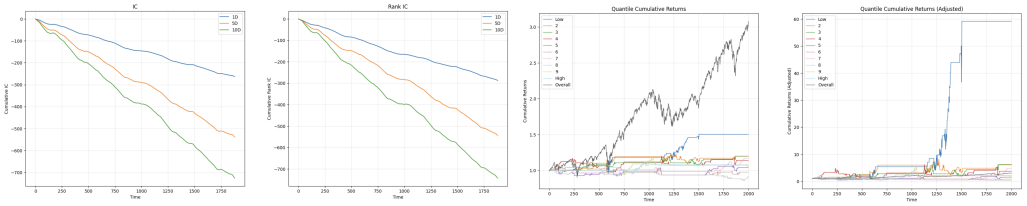

Figure 1: Left: Cumulative IC and Rank IC for feature E2. Right: Grouped cumulative returns for feature E3.

Initial screening selected 21 promising features. To reduce redundancy and multicollinearity, we computed the correlation matrix (Figure 2, Left). For pairs with correlation > 0.7, we retained the one with higher IR.

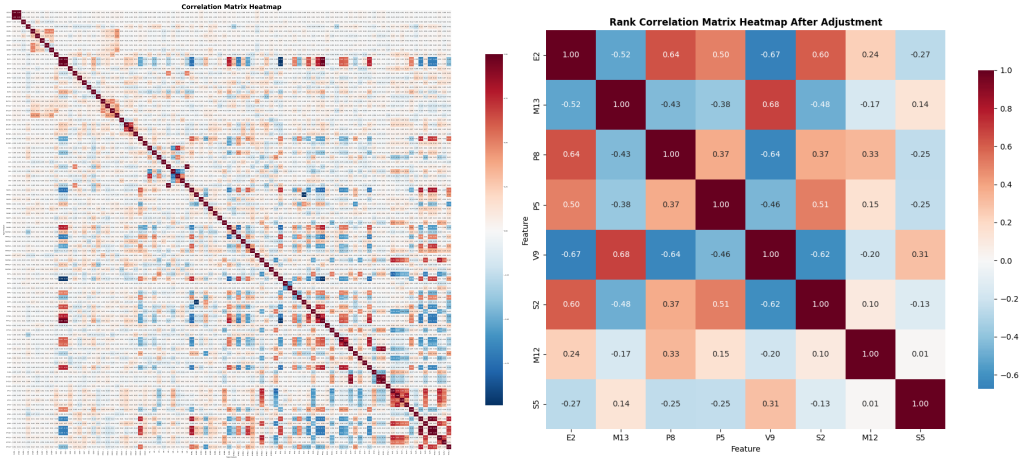

Figure 2: Left: Correlation matrix of initial features. Right: Correlation matrix of the final selected 8 features.

The final selected set consists of 8 features: `["E2", "M13", "P8", "P5", "V9", "S2", "M12", "S5"]`. These features serve as the input for both our SFT models and the RL Policy Network. We use a `FeatureStore` to ensure consistent `StandardScaler` transformation between training and inference.

3 METHODOLOGY

Our architecture follows a "Sim-to-Real" philosophy, ensuring the training process mimics the uncertainty of the online inference environment.

3.1 STAGE 1: SUPERVISED FINE-TUNING (SFT)

3.1.1 MODEL SELECTION: THE PATCHTST VS. N-HITS DILEMMA

In the initial phase, we experimented with two state-of-the-art time-series architectures: **PatchTST** (Transformer-based) (Nie et al., 2022) and **N-HiTS** (MLP-based) (Challu et al., 2023). Our empirical results (Table 1) strongly favored N-HiTS.

Table 1: Performance Comparison on 10 Test Samples (Local Validation)

Model	MSE	IC (Corr)	Hit Rate
PatchTST	0.00007963	-0.2466	50.00%
N-HiTS	**0.00006249**	**0.0611**	**70.00%**
Ensemble	0.00005537	-0.1082	60.00%

Analysis of Failure (PatchTST): PatchTST employs a Channel Independence (CI) strategy, treating each variable as an isolated univariate series. However, our data analysis showed that returns are driven by strong **cross-sectional factors** (E2, E3). PatchTST fails to model the explicit relationship $y \approx w \cdot E2 + b$ because it only looks at the history of $E2$ to predict $E2$, rather than using $E2$ to predict y. Consequently, it achieved a Hit Rate of only 50% (random guess) and a negative IC, indicating overfitting to noise.

Analysis of Success (N-HiTS): N-HiTS, despite being a time-series model, effectively functions as a stacked MLP that can perform multivariate regression. It successfully captured the strong linear signal from the "Super Factors" E2 and E3. The high Hit Rate (70%) and positive IC confirm that N-HiTS learned the correct directional relationship, leveraging the strong alpha present in the selected features. Therefore, we adopted an "All-in N-HiTS" strategy.

3.1.2 THE "ROTTEN APPLE" MECHANISM

To train the downstream RL agent effectively, we cannot feed it "perfect" predictions trained on the full dataset (which would cause look-ahead bias). Instead, we need signals that reflect the model's true out-of-sample errors. We implemented a **Rolling Cross-Validation** mechanism, which we term "Rotten Apples":

$$\hat{y}_t = f_{\theta_{t-k}}(\boldsymbol{x}_t) \tag{1}$$

where the model f used to predict at time t is trained only on data up to $t-k$ (where k is the forecast horizon). We generate these OOS predictions for the entire training history. This provides the RL agent with a realistic tuple $(\hat{y}_t, y_t, \text{Vol}_t)$ for every time step t.

3.2 STAGE 2: GENERALIZED REINFORCEMENT POLICY OPTIMIZATION (GRPO)

We formulate the trading problem as a simplified Reinforcement Learning task. Since the environment (the market) does not react to our actions (assuming no market impact), we treat each day as a **single-step trajectory**.

3.2.1 STATE SPACE AND POLICY NETWORK

The agent observes a state $s_t \in \mathbb{R}^{11}$ consisting of:

- **Signal**: $\hat{y}_t$ (prediction from N-HiTS).
- **Market State**: Rolling Volatility (std dev of returns) and Rolling Trend (mean of returns).
- **Features**: The 8 selected raw financial features.

Crucially, we explicitly include the raw features (especially E2, E3) in the state. This allows the Policy Network to "see" the strong factors directly, enabling it to override the N-HiTS signal if necessary or learn a non-linear risk adjustment based on factor regimes.

The Policy Network (Actor) is an MLP that maps state s_t to the parameters of a Beta distribution, constrained to be positive via Softplus:

$$\alpha, \beta = \text{Softplus}(\text{MLP}(s_t)) + 1 \tag{2}$$

The action $a_t \in [0, 2]$ (representing position size) is sampled from this distribution:

$$a_t \sim 2 \cdot \text{Beta}(\alpha, \beta) \tag{3}$$

During training, sampling encourages exploration. During inference, we use the expectation $\mathbb{E}[a_t] = 2 \cdot \frac{\alpha}{\alpha+\beta}$ for deterministic execution.

3.2.2 REWARD FUNCTION AND OPTIMIZATION

We maximize a composite reward function R_t designed to balance profit, risk, and transaction costs:

$$R_t = \underbrace{\text{PnL}_t}_{\text{Profit}} - \underbrace{\lambda_{risk} \cdot (a_t \cdot \sigma_t)^2}_{\text{Risk Penalty}} - \underbrace{\lambda_{turnover} \cdot |a_t - a_{t-1}| \cdot \text{Cost}}_{\text{Transaction Cost}} \tag{4}$$

where $\text{PnL}_t = a_t \cdot y_t \cdot \text{ScaleFactor}$. The scale factor (e.g., 100) is crucial to prevent gradient vanishing given the small magnitude of daily returns.

We optimize the policy using **Generalized Reinforcement Policy Optimization (GRPO)**, which simplifies PPO (Schulman et al., 2017) by eliminating the value function critic and using the batch mean reward as the baseline for advantage estimation:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(a_i|s_i) \cdot (R_i - \bar{R}_{batch}) \tag{5}$$

4 EVALUATION

4.1 EXPERIMENTAL SETUP

The submission was evaluated using the Kaggle Evaluation API, which imposes strict constraints on inference time and internet access.

- **Environment:** Kaggle Notebook (Offline). We utilized a custom offline package installation mechanism for `neuralforecast` (Garza et al., 2022) using pre-downloaded wheel files.
- **Hyperparameters:** The Policy Network was trained for 50 epochs with a batch size of 128 and a learning rate of 1e-3 with StepLR scheduling.
- **Cold Start Strategy:** A momentum-based heuristic ($\tanh(return \times 50)$) is used when the history window (< 60 days) is insufficient for the N-HiTS model to form a prediction window.

4.2 RESULTS

Local Validation: Our best policy achieved a significantly lower loss on the validation set compared to the baseline SFT signals. Qualitatively, the agent learned to reduce position sizes ($a_t \to 1.0$) during periods of high volatility, demonstrating successful risk aversion learning. This behavior was not explicitly programmed but emerged from the λ_{risk} term in the reward function.

Kaggle Leaderboard Performance: We successfully submitted our agent to the Kaggle competition. The inference pipeline functioned correctly without runtime errors, verifying the robustness of our engineering implementation.

Figure 3: Snapshot of our Public Leaderboard entry. We achieved a score of **0.431**, ranking 2923 at the time of submission.

As shown in Figure 3, we achieved a public score of 0.431. While this score indicates room for improvement compared to the top leaderboard positions, it validates that our end-to-end pipeline—from feature engineering to RL inference—is functional and capable of generating positive signals.

5 CONCLUSION

In this project, we successfully implemented a sophisticated SFT-GRPO trading pipeline. By decoupling signal generation from portfolio decision-making, we created a flexible system capable of optimizing for complex financial objectives beyond simple MSE.

Key takeaways include:

1. **Factor Analysis Drives Architecture:** The discovery of super-strong negative linear factors (E2/E3 with Rank IC -0.15) directly informed our decision to abandon PatchTST in favor of N-HiTS, which could better model these relationships.
2. **Honest Validation:** Generating "Rotten Apple" OOS signals is essential. Training an RL agent on "perfect" in-sample SFT predictions leads to severe overfitting and failure in online deployment.
3. **Infrastructure Matters:** Robust handling of online inference constraints (cold start, package management, memory management) is as important as the theoretical model itself.

Future work will focus on refining the reward function hyperparameters to better capture market regimes and experimenting with TimesNet for potentially better multi-period forecasting.

REFERENCES

Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Ramirez, Merense Garza, and Max Mergenthaler. N-hits: Neural hierarchical interpolation for time series forecasting. *arXiv preprint arXiv:2201.12886*, 2023.

Merense Garza, Cristian Challu, and Kin G Olivares. Neuralforecast: User friendly deep learning for time series forecasting. *arXiv preprint arXiv:2202.12415*, 2022.

Blair Hull, Xiao Qiao, and M Bakato. Return predictability and market-timing: A one-month serial correlation anomaly. *Available at SSRN 3050254*, 2017.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

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