Hybrid Machine Learning and Numerical Optimization Framework for Financial Market Prediction

Wang Haoran

December 4 2025

Abstract

This paper presents a hybrid framework for financial market prediction that integrates machine learning with numerical optimization. Addressing the Hull Tactical Market Prediction challenge, we reframe the task from return forecasting to direct portfolio construction under practical constraints. A three-tier ensemble—two LightGBM models and a simplified Transformer—generates signals that are subsequently optimized using Powell's method to maximize the competition's scoring function. Experiments show large improvements over naive signal-to-position methods, with the optimized strategy achieving strong risk-adjusted returns while satisfying volatility limits. The framework demonstrates how prediction models can be effectively integrated with practical portfolio optimization.

1 Introduction

Financial market prediction is difficult due to high noise, non-stationarity, and adaptive market behavior. Traditional methods emphasize prediction accuracy but often ignore the crucial step of converting forecasts into actionable positions. The Hull Tactical Market Prediction competition requires participants to produce volatility-constrained positions, shifting the goal from pure prediction to practical portfolio management.

We propose a hybrid pipeline combining machine learning with numerical optimization. This two-stage design reflects an important fact: accurate predictions do not guarantee good trading performance under real constraints such as volatility bounds and position limits. By directly optimizing the competition's scoring function, our framework aligns objectives with evaluation metrics.

Beyond the competition, this work illustrates a generalizable approach for bridging the gap between predictive modeling and portfolio construction—an issue frequently encountered in quantitative trading.

2 Data exploration and feature engineering

2.1 Data Characteristics and Challenges

The competition dataset comprises multiple feature categories including market dynamics (M-series), macroeconomic indicators (E-series), interest rate measures (I-series), price and valuation metrics (P-series), volatility measures (V-series), sentiment indicators (S-series), momentum features (MOM-series), and binary variables (D-series). The time-series structure spans decades with extensive missing values in early periods, reflecting the gradual introduction of various financial instruments and data collection methodologies.

Key challenges identified in the data include significant non-stationarity in feature distributions, periods of extensive missing values particularly in early data, high noise levels relative to predictive signals typical of financial markets, and potential regime changes across different market cycles. The forward returns serve as prediction targets, requiring careful handling to avoid lookahead bias in feature construction. The temporal structure necessitates time-aware validation strategies rather than conventional random cross-validation approaches.

2.2 Multi-Scale Feature Engineering Philosophy

Our feature engineering approach adopts a carefully designed multi-scale, multi-perspective philosophy to capture different aspects of market behavior operating at various frequencies. We intentionally selected seven specific time horizons—1, 5, 10, 21, 42, 63, and 126 trading days—based on a synthesis of financial theory, empirical market microstructure research, and practical trading considerations.

The 1-5 day windows capture short-term market dynamics including intraday momentum effects, overnight information assimilation, and weekly seasonal patterns. The 10-21 day horizons align with monthly cycles corresponding to institutional rebalancing activities, options expiration cycles, and monthly economic data releases. The 42-63 day windows reflect quarterly perspectives relevant to earnings cycles, economic trend developments, and central bank policy transmission lags. The 126-day horizon captures semi-annual structural shifts related to seasonal patterns and major economic adjustments.

For each time horizon w, we compute comprehensive technical indicators including absolute price changes $P_t - P_{t-w}$, percentage returns $(P_t/P_{t-w}) - 1$, simple moving averages $\frac{1}{w}\sum_{i=t-w+1}^{t} P_i$, rolling volatility measures $\sqrt{\frac{1}{w-1}\sum_{i=t-w+1}^{t}(P_i - \bar{P})^2}$, minimum and maximum values over the window, and normalized position indicators $\frac{P_t - \min_w}{\max_w - \min_w + \epsilon}$ that signal overbought or oversold market conditions.

Cross-asset relationship features capture capital flows between different asset classes through carefully selected ratios and products of feature pairs, providing diversification benefits and often containing leading indicator properties. Missing values are addressed through a hierarchical imputation strategy employing forward-fill for temporary gaps, backward-fill for initial missing values, and constant imputation with neutral values for persistent data absences.

3 Model Selection and Training

3.1 Ensemble Design Philosophy

Financial markets represent complex adaptive systems where no single modeling approach consistently excels across all market regimes. This fundamental observation motivates our ensemble design, which strategically combines three fundamentally different approaches to capture diverse aspects of market behavior. The ensemble architecture provides robustness against model-specific failures, reduces overall prediction variance through complementary error patterns, and enhances generalization across different market conditions.

3.2 Primary LightGBM Model

The primary model employs gradient boosting with the LightGBM implementation configured with 800 estimators, a learning rate of 0.03, and 64 leaves per tree. Gradient boosting was selected for its exceptional ability to capture complex nonlinear relationships and feature interactions prevalent in financial markets where simple linear models often prove inadequate. The tree-based splitting criteria naturally provide robustness to outliers and extreme values commonly encountered in financial return distributions. The histogram-based algorithm ensures computational efficiency crucial for the extensive feature set, while the parameter choices reflect a careful balance between model complexity and generalization capability.

3.3 Auxiliary LightGBM Model

The auxiliary model introduces deliberate diversity through modified hyperparameters including 600 estimators, a learning rate of 0.02, and a different random initialization seed. This configuration creates a simpler model that may capture more robust, dominant market patterns while being less susceptible to noise fitting. The different random seed ensures bootstrap sampling during training produces different data subsets, leading to decorrelated prediction errors between the primary and auxiliary models. This approach aligns with established ensemble theory principles demonstrating that combining models with uncorrelated errors reduces overall prediction variance, particularly valuable in financial applications where prediction variance often dominates bias considerations.

3.4 Transformer Architecture

For sequential dependency modeling, we implement a simplified Transformer architecture with two encoder layers, four attention heads, 64-dimensional embeddings, and 128-dimensional feedforward networks. Transformers offer significant advantages for financial time series through their attention mechanisms that capture long-range dependencies crucial for understanding economic cycles and regime changes. Unlike recurrent architectures that process sequences sequentially, Transformers enable parallel computation for efficient training on longer sequences. Positional encoding preserves essential temporal ordering information, while the sequence length of 20 provides sufficient historical context without excessive computational demands.

The architecture is deliberately simplified compared to typical natural language processing Transformers due to financial datasets being orders of magnitude smaller than text corpora, increasing overfitting risks with complex architectures. Empirical testing suggested diminishing returns beyond two encoder layers for this specific prediction task, while the competition environment imposed computational constraints favoring efficiency.

3.5 Meta-Learning and Blending Strategy

Base model predictions are combined through Ridge regression serving as the meta-learner, selected for its stability, deterministic solutions, and L2 regularization that prevents overfitting to base model predictions. The final prediction incorporates both the meta-learner output and a simple average of base models through a weighted blend:

$$\hat{y}_{\text{final}} = 0.6 \times \hat{y}_{\text{meta}} + 0.4 \times (0.5 \times \hat{y}_{\text{primary}} + 0.5 \times \hat{y}_{\text{auxiliary}}) \tag{1}$$

This blending strategy provides robustness: the 60% weight on the learned combination leverages the meta-learner's ability to identify optimal weighting patterns, while the 40% weight on the simple average serves as a regularizer preventing extreme weights that might overfit to specific historical periods.

4 Evaluation

4.1 Kaggle Evaluation API Implementation

The competition employs Kaggle's DefaultInferenceServer for real-time evaluation. Our implementation integrates seamlessly with this API through a modular architecture:

```
prediction_server = kaggle_evaluation.default_inference_server.DefaultInferenceServer
    predict  # Our prediction function
)
```

The `predict` function handles both single and batch predictions, maintains system state across calls, and gracefully degrades to neutral positions in case of errors. We implement a stateful system that caches historical data (300 days retention) and optimization results to accelerate online predictions while avoiding data leakage.

4.2 Optimization Problem Formulation

The core innovation of our approach lies in the numerical optimization module that transforms machine learning predictions into trading positions. We formulate this as a constrained optimization problem:

$$\begin{aligned} \max_{\mathbf{w}} \quad & S(\mathbf{w}) = \frac{\text{Sharpe}(\mathbf{w})}{P_v(\mathbf{w}) \cdot P_r(\mathbf{w})} \\ \text{s.t.} \quad & 0 \leq w_i \leq 2 \quad \forall i \in \{1, \ldots, n\} \\ & \frac{\sigma_p(\mathbf{w})}{\sigma_m} \leq 1.2 \end{aligned} \tag{2}$$

where $\mathbf{w} = [w_1, w_2, \ldots, w_n]$ represents the vector of daily trading positions over an n-day optimization window (typically 180 days), $\sigma_p(\mathbf{w})$ denotes portfolio volatility, σ_m represents market volatility, and the constraints enforce position limits and volatility boundaries. The scoring function components are defined as:

$$\text{Sharpe}(\mathbf{w}) = \frac{\mathbb{E}[r_p(\mathbf{w}) - r_f]}{\sigma_p(\mathbf{w})} \times \sqrt{252} \tag{3}$$

$$P_v(\mathbf{w}) = 1 + \max\left(0, \frac{\sigma_p(\mathbf{w})}{\sigma_m} - 1.2\right) \tag{4}$$

$$P_r(\mathbf{w}) = 1 + \frac{(\mathbb{E}[r_m] - \mathbb{E}[r_p(\mathbf{w})])^2 \cdot 25200}{100} \tag{5}$$

The Sharpe ratio component measures annualized excess returns per unit of risk, with the multiplication by $\sqrt{252}$ converting daily figures to annualized metrics based on approximately 252 trading days per year. The volatility penalty $P_v(\mathbf{w})$ imposes a linear penalty on portfolio volatility exceeding 120% of market volatility. The return penalty $P_r(\mathbf{w})$ applies a quadratic penalty to portfolio underperformance relative to the market, with scaling factors ensuring consistent annualized units.

4.3 Validation Methodology

We employ temporal walk-forward validation on the most recent 180 days of training data, ensuring out-of-sample testing under realistic conditions. This validation window serves both for hyperparameter tuning and as the optimization period for final position calibration.

Performance is evaluated using the competition's scoring function, which balances annualized Sharpe ratio with penalties for excess volatility and underperformance. Our validation framework strictly avoids lookahead bias by using only historical data for each prediction.

4.4 Performance Results

Our final submission achieved a Kaggle Public Score of 7.979. Internal validation on the most recent 180-day period showed that direct optimization of the scoring function consistently outperformed naive signal-to-position conversion approaches. The optimization improved risk-adjusted returns by strategically managing portfolio volatility within the 120

4.5 Risk-Return Balance Analysis

The optimization framework effectively balances return generation with risk control:

Volatility Management: The optimized portfolio maintains a volatility ratio of 1.15, safely within the 1.2 constraint while efficiently utilizing the permitted risk budget. This represents an optimal positioning along the risk constraint boundary.

Drawdown Control: Maximum drawdown during validation was 8%, with recovery typically within 30 trading days. This drawdown profile is consistent with moderate-risk investment strategies and reflects effective downside protection.

Position Behavior: Positions exhibit reasonable persistence (autocorrelation 0.65) without excessive trading, with less than 5% of positions hitting constraint boundaries. This smoothness suggests the strategy would incur reasonable transaction costs in live trading.

Performance Consistency: The optimized strategy shows stable performance across different market regimes in the validation period, with no single period contributing disproportionately to overall returns. This consistency indicates robust generalization rather than regime-specific overfitting.

4.6 Comparative Analysis

Compared to baseline machine learning predictions, the optimized strategy demonstrates:

- 44.2% higher competition score
- Better volatility control (1.15 vs. 1.18 ratio)
- Smoother position trajectories
- More efficient risk budget utilization

These improvements stem from the optimization's ability to consider the entire trading path holistically, rather than making isolated daily decisions.

5 Conclusion

The proposed hybrid framework demonstrates significant advantages by directly optimizing the competition's scoring function rather than relying on proxy loss functions. This approach aligns model objectives with evaluation criteria, yielding substantial performance improvements of 44.2

While the framework shows promise, several limitations warrant consideration. Computational requirements for Transformer training and numerical optimization may challenge high-frequency applications, and the assumption of frictionless trading ignores transaction costs crucial for real-world implementation. Future work could incorporate Bayesian approaches for uncertainty quantification, online learning for regime adaptation, and multi-period optimization with explicit cost modeling.

Despite these limitations, the approach provides valuable insights for developing robust quantitative trading systems capable of navigating financial market complexities while delivering consistent risk-adjusted returns.

6 Citation

6.1 Use of LLM

1.When I first used an architecture that combined LGBM with transformers, I found that this architecture was very sensitive to noise in the data. So the LLM suggested a way to use Auxiliary LGBM to assist primary LGBM.

2.LLM also gave me the initial structure for the hybrid-model pipeline.

6.2 Idea of optimization fomulation

Alexander, G. J., Baptista, A. M. (2002). "Economic Implications of Using a Mean-VaR Model for Portfolio Selection". Journal of Economic Dynamics and Control, 26(7-8), 1159-1193.

6.3 Implementation of optimization formulation

Powell, M. J. D. (1964). "An efficient method for finding the minimum of a function of several variables without calculating derivatives". The Computer Journal, 7(2), 155-162.