Market Prediction via Timeseries Models

Team7
Chung-Ang University
Seoul, Korea

Abstract

Stock market, a highly complex timeseries data that reflects real world company values. In this report, we report results of *Hull Tactical - Market Prediction* competition from kaggle. We do extensive experiments on the S&P500, NASDAQ100 (QQQ) and BTC dataset with 10 models. We perform feature augmentation and ablation studies to investigate the impact of features on market prediction. For reproducibility, we have released the code and datasets at https://github.com/HNUI/MachineLearningProject_TeamProj4_Team7.

1 Introduction

The stock market already gained interest couple decades ago and recently gained more interest due to increasing monopoly structure. Yet, it is believed that asset prices fully reflect all available information, thus acknowledged hard to *'beat the market'*. Although it is impossible to build a model to do a head to head comparison with the market due to data scarcity, we instead investigate how modern machine learning and deep learning models perform at predicting the market.

The core objective of this paper is to obtain a deeper understanding of how features impact the model's capacity on market prediction. The paper configuration is as follows.

- **Task Environment**: We summarize the environment of the kaggle competition. Explanation of features along with augmented features will be included.
- **Experiments**: We introduce 10 baseline models, 1 naive-model, 5 machine learning models and 4 deep learning models. We later perform feature ablation to see the impacting role of features in trianing.
- **Conclusion**: We briefly summarize the results and our understanding of this particular task.

2 Task Environment

The task environment is a univariate financial time–series prediction and allocation setup based on the Hull Tactical Market Prediction framework. At each decision time t, the agent observes only information that is known at or before t, and must output a continuous portfolio position $p_t \in [0, 2]$. The realized reward is computed one period ahead using excess market returns. To prevent look–ahead leakage, every feature is constructed from lagged or backward–looking quantities.

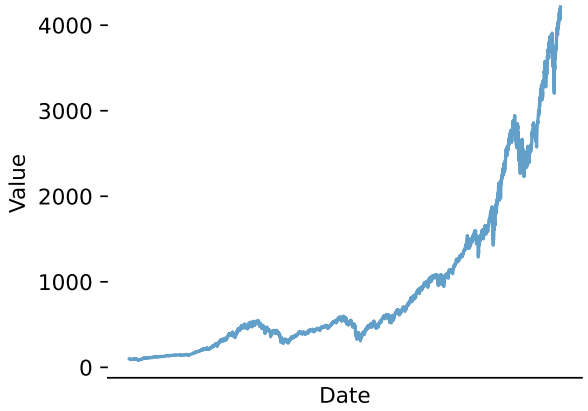

Figure 1: Value of S&P500

2.1 Observations

For each date t, the model receives a vector of features $x_t \in \mathbb{R}^d$ consisting of:

- **Lagged returns.** The one–day return is defined as

$$r_t = \text{forward\_returns}_{t-1},$$

  ensuring that future market movements are never used as input.
- **Momentum features.** Momentum feature was missing from the official competition data, thus we derive it from the given information we had. Multi–horizon momentum signals are computed using lagged cumulative returns. For a window k,

MLP '25, Seoul, Korea

the k–day momentum is

$$\mathrm{MOM}_k(t) = \prod_{i=0}^{k-1} (1 + r_{t-i}) - 1,$$

implemented numerically as a rolling sum of $\log(1+r_t)$ for stability. This includes horizons of 1 and 5 days. Excess–return momentum uses

$$r_t^{\mathrm{ex}} = r_t - \mathrm{risk\_free\_rate}_{t-1}.$$

- **Volatility and drawdown.** Short–term volatility is computed from rolling log–return variance

$$\mathrm{vol}_{21}(t) = \mathrm{Std}\big(\log P_{t-i}\big)_{i=0}^{20}.$$

- **Original dataset features.** All provided feature groups (D*, M*, E*, I*, P*, V*, S*, and lagged_* fields) are included except variables containing forward–looking information such as `forward_returns` and `market_forward_excess_returns`.

All missing values in backward–looking indicators are imputed using exponential weighted moving averages to maintain strict temporal causality.

2.2 Action Space

At time t, the model outputs a single continuous action

$$p_t \in [0, 2],$$

interpreted as the fraction of capital allocated to the market index. A value of 1 corresponds to a neutral allocation, $p_t = 0$ to a full risk–free allocation, and $p_t = 2$ to a leveraged long position.

2.3 Transition and Reward

Given the next–period forward return fret_t and risk–free rate rf_t, the realized strategy return is

$$R_t^{\mathrm{strat}} = \mathrm{rf}_t(1 - p_t) + p_t\,\mathrm{fret}_t.$$

The excess strategy return is $R_t^{\mathrm{strat}} - \mathrm{rf}_t$. Performance is evaluated using a volatility–adjusted Sharpe metric with penalties that enforce a maximum volatility ratio of 1.2 relative to the market.

2.4 Data Splitting and Evaluation

The dataset is ordered chronologically and split into training and test partitions from the train.csv file. This was inevitable due to the scarcity of test.csv file provided in the competition. All normalization parameters (e.g., the standardization mean and variance) are fitted exclusively on the training segment. Model selection uses walk–forward cross–validation: for each fold k, the model trains on dates 1 through t_k and validates on $[t_k, t_k + h]$, where h is a fixed hold–out horizon. This procedure ensures that every validation window simulates true forward–in–time prediction.

Final evaluation uses the official volatility–capped modified Sharpe score, computed separately on the training and held–out test segments.

3 Experiments

3.1 Experimental Setup

All experiments are conducted on the official `train.csv` file provided in the Hull Tactical dataset. The data are sorted chronologically by `date_id`, and all feature construction is strictly causal: each feature at time t depends only on information available at or before t. Forward–looking variables such as `forward_returns` and `market_forward_excess_returns` are excluded from the feature set and used only for constructing lagged targets and performance evaluation.

	#train	#test	#train feature
S&P500	5413	3608	101
QQQ	930	618	60
BTC	930	618	54

Table 1: Dataset statistics

The full dataset is partitioned into training and test segments using a fixed chronological split. Let T denote the total number of samples. The first $0.6T$ observations form the training set, and the remaining $0.4T$ are reserved for held–out testing. A StandardScaler is fitted only on the training features and applied to all subsequent data, ensuring no information leaks from the test region.

Model selection is performed using walk–forward cross–validation. For K folds, the k-th training window consists of all dates up to a cutoff t_k, and the validation window covers $[t_k, t_k + h]$, where h is a fixed evaluation horizon determined by a 20% holdout fraction. This emulates a realistic forward-in-time prediction setting and prevents access to future information during training.

Sequence models (LSTM[1], GRU[1], Temporal Convolutional Networks[2], and Transformers[4]) operate

on fixed-length windows of size ℓ constructed from the scaled feature matrix. For each model, only the most recent ℓ observations are used to predict the next-period excess market return. Training uses AdamW[3] with mean squared error loss over 500 epochs unless otherwise specified.

Performance is evaluated using the competition's modified Sharpe score with volatility enforcement. Given predicted positions $p_t \in [0, 2]$, strategy returns are computed as

$$R_t^{\text{strat}} = \text{rf}_t(1 - p_t) + p_t \, \text{fret}_t,$$

and a post-hoc volatility cap rescales any sequence of positions whose realized volatility exceeds 1.2 times the market's volatility. All reported metrics follow this final volatility–adjusted score.

3.2 Baseline Models

We benchmark a compact but diverse set of regression models spanning linear, tree–based, and sequence architectures. All models share the same causal feature set, chronological preprocessing, walk–forward training, and volatility–capped evaluation. We fixed hidden and batch size of sequential models to 128 in order to limit the model capacity to a certain bound.

Naive (1.0) Benchmark. We include a constant-position baseline defined by $p_t = 1.0$ for all t. This strategy holds a full unleveraged exposure to the market at every date, producing strategy returns

$$R_t^{\text{strat}} = \text{fret}_t,$$

where fret_t denotes the one-period forward market return. The naive strategy therefore corresponds to a simple buy-and-hold allocation under the dataset's return convention. Its volatility-capped modified Sharpe score provides a reference point against which all learned models are compared.

Ordinary Least Squares (OLS).. A linear model

$$\hat{y}_t = w^\top x_t + b,$$

serving as a minimal unregularized baseline.

Elastic Net. A penalized linear regression with mixed ℓ_1–ℓ_2 regularization,

$$\min_{w,b} \sum_i (y_i - w^\top x_i - b)^2 + \lambda_1 \|w\|_1 + \lambda_2 \|w\|_2^2,$$

and 10-fold CV for selecting λ.

Random Forest. A bagged ensemble of shallow regression trees (`max_depth=2`, `n_estimators=1000`) that captures nonlinear thresholds without temporal modeling.

LightGBM.. A gradient boosting model with leaf–wise growth and histogram splits. We use small tree depth and many boosting rounds, providing a strong nonlinear baseline.

XGBoost. A level–wise gradient boosting model using second–order loss expansion and shrinkage. Trained with shallow trees and `n_estimators=20000`.

LSTM.. A recurrent sequence model processing windows of length ℓ with hidden size 128 and dropout 0.8, capturing temporal dependencies.

GRU.. A lighter recurrent architecture using the same sequence length and hidden size as the LSTM, offering comparable temporal modeling with fewer parameters.

Temporal Convolutional Network (TCN).. A dilated 1-D convolution model with receptive fields expanding with depth. We use five levels, channel width 32, and kernel size 5.

Transformer (TX).. A self-attention sequence model with multi-head attention (`heads=16`) and feed-forward width 64, capturing both local and global temporal structure.

All baselines operate under identical data access, feature preprocessing, and walk–forward evaluation, ensuring that performance differences reflect model capacity rather than training protocol. Detailed hyper parameter settings are available at out code base.

3.3 Results

Table 2 summarizes model performance under the volatility–capped sharp metric. **Bold** indicates best, underline indicates second best. The naive baseline achieves a score of 0.8049, and most linear or tree–based models (Enet, OLS, RF, LGBM, XGB) remain close to this level, indicating limited gains from shallow nonlinear structure. Recurrent models (GRU, LSTM) offer moderate improvements, reflecting the benefit of temporal modeling in return prediction.

The TCN exhibits the strongest performance with a score of **0.8406**, substantially outperforming all other models. This suggests that dilated convolutions provide

	Metric
Naive(1.0)	0.8049 ± 0.0000
Enet	0.8050 ± 0.0000
Ols	0.8025 ± 0.0000
GRU	0.8094 ± 0.0016
LGBM	0.8053 ± 0.0000
LSTM	0.8108 ± 0.0011
RF	0.8051 ± 0.0002
TCN	**0.8406** ± 0.0264
TX	0.8075 ± 0.0025
XGB	0.8033 ± 0.0000

Table 2: Sharp score with out momentum features(S&P500)

a more effective inductive bias for capturing multi–scale temporal patterns than either recurrent models or boosted trees. Transformers (TX) outperform linear models but fall short of the TCN, likely due to the small data regime and the absence of extensive regularization.

Overall, sequence models consistently outperform static models, and the TCN in particular emerges as the most effective architecture for this forecasting task.

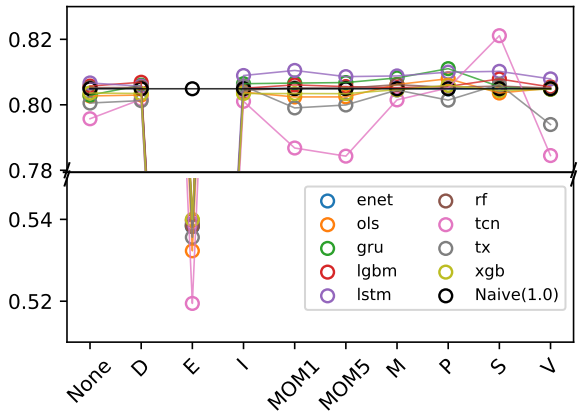

Figure 2: Feature engineering

Figure 2 shows the ablation of feature dropping. We can see a significant drop in performance when feature E* is all dropped from the training data, which indicates the importance of market dynamics/technical features. Other features didn't seem to have much impact on model performance. TCN fluctuated severely compared to other models.

4 Conclusion

This report examined a broad set of forecasting models under a unified, causally valid walk–forward evaluation protocol. Using identical feature engineering, chronological training windows, and a volatility–capped sharp metric, we compared linear models, tree–based ensembles, and modern sequence architectures.

Our findings show that static models—including OLS, Elastic Net, Random Forest, LightGBM, and XGBoost—provide only modest improvement over the naive baseline. These approaches capture limited nonlinear structure but fail to model temporal dependencies that are essential in financial return prediction.

Sequence models consistently outperform static alternatives. GRU and LSTM networks yield noticeable gains, demonstrating that gated recurrent architectures benefit from sequential information. The strongest performance comes from the Temporal Convolutional Network (TCN), which achieves the highest sharp score among all baselines.

Overall, our results highlight that temporal modeling capacity is crucial for achieving meaningful improvements in this prediction task. Future work may incorporate richer features, alternative volatility normalization schemes, or hybrid architectures to further enhance predictive stability and performance.

References

[1] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[2] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. 2017. Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 156–165.

[3] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

A QQQ

Table 3 reports the sharp scores for QQQ. The naive baseline attains a score of 1.1994, and several classical models—Enet, OLS, LGBM, RF, XGB—cluster tightly

	Metric
Naive(1.0)	1.1994 ± 0.0000
Enet	1.1993 ± 0.0000
Ols	1.2010 ± 0.0000
GRU	1.1955 ± 0.0212
LGBM	1.2011 ± 0.0000
LSTM	**1.2054** ± 0.0041
RF	1.2025 ± 0.0000
TCN	1.1334 ± 0.0736
TX	1.1979 ± 0.0106
XGB	1.2038 ± 0.0000

Table 3: Sharp score (QQQ)

	Metric
Naive(1.0)	1.0434 ± 0.0000
Enet	1.0483 ± 0.0000
Ols	1.0585 ± 0.0000
GRU	1.0727 ± 0.0146
LGBM	1.0616 ± 0.0000
LSTM	1.0817 ± 0.0130
RF	1.0534 ± 0.0005
TCN	1.0241 ± 0.0377
TX	**1.0959** ± 0.0142
XGB	1.0584 ± 0.0000

Table 4: Sharp score (BTC)

around this level, indicating that shallow linear or tree–based methods extract only limited additional signal under the walk–forward protocol. Recurrent models show meaningful improvements: the GRU reaches 1.1955, while the LSTM yields a stronger 1.2054, suggesting that sequence modeling contributes positively to stability and predictive sharpness.

The transformer (TX) performs comparably to linear models, with a score of 1.1979, but does not surpass the recurrent baselines. The TCN registers the lowest score at 1.1334, reflecting instability or overfitting on this particular asset. Overall, LSTM delivers the strongest performance in the QQQ experiment, outperforming both classical and convolutional models.

B BTC

Table 4 summarizes performance for BTC, which exhibits a higher noise regime and stronger nonlinearity than QQQ. Linear models again provide modest gains over the naive baseline, with OLS (1.0585) and Enet (1.0483) improving only slightly. Tree–based methods perform similarly, with LGBM and XGB achieving scores near 1.06. The RF baseline offers marginal improvement but remains close to the naive baseline.

Sequence models show substantially stronger performance. The GRU attains 1.0727, while the LSTM improves further to 1.0817, demonstrating the usefulness of temporal dependency modeling in volatile markets. The transformer achieves the highest score overall (1.0959), indicating that global self–attention captures temporal structure more effectively in the BTC series than either RNNs or convolutional models.

The TCN again underperforms, with a score of 1.0241, suggesting sensitivity to the irregular dynamics of BTC or insufficient robustness under the chosen hyperparameters. In contrast to QQQ, where LSTM dominated, the BTC results highlight the transformer as the most effective architecture.

Summary. Across both assets, traditional linear and tree–based models provide limited improvement over the naive baseline, while temporal models—especially LSTM and TX—offer consistent gains. The best-performing architecture varies by asset: LSTM is strongest for QQQ, whereas the transformer excels for BTC, reflecting differences in temporal complexity across the underlying markets.

Figure 3: Leaderboard