# A1. Background on you/your team

- Competition Name: MITSUI&CO. Commodity Prediction Challenge
- Team Name: honkai01031227
- Private Leaderboard Score: 0.490
- Private Leaderboard Place: 8th


- Name: Wong Ye Fei
- Location: Singapore
- Email: wongyefei@gmail.com


# A2. Background on you/your team

If part of a team, please answer these questions for each team member. For larger teams (3+), please give shorter responses.

What is your academic/professional background?
Year 2 undergraduate at Nanyang Technological University (NTU), majoring in Mathematical Sciences with a second major in Data Analytics

Did you have any prior experience that helped you succeed in this competition?
Prior experience in university research project on optimization on smooth manifolds

What made you decide to enter this competition?
To have more experience in coding

How much time did you spend on the competition?
few month

If part of a team, how did you decide to team up?
NA

If you competed as part of a team, who did what?
NA

# A3. Summary

4-6 sentences summarizing the most important aspects of your model and analysis, such as:

The final solution is a supervised learning pipeline using a Transformer encoder that treats each numeric feature as a token through a feature tokenizer and column embeddings, and predicts all 424 target values jointly. The most important signal comes from lagged features of historical targets, which are combined with the original tabular market features from train.csv to capture temporal dependencies. Inputs are standardized using StandardScaler fit on the training data to ensure stable optimization and reproducible inference. Model training is implemented in PyTorch, incorporating gradient clipping, a cosine learning-rate schedule with warmup, and early stopping based on validation loss. This design enables efficient batch inference, with the full training process completing in approximately 12 minutes on GPU, while maintaining strong predictive performance on the private leaderboard.

The code was developed with the assistance of ChatGPT Plus.

# A4. Features Selection / Engineering

**What were the most important features?**

The most important features were the original numerical input features provided in the dataset (train.ssv), together with lagged versions of the target variables. In particular, the 4-day lag (lag-4) of all target variables was used, consistent with the information available during evaluation.

In total, the model used 981 features, consisting of:

587 raw input features from train.csv, and

424 lag-4 target features, representing historical target values shifted by four days.

These lagged target features captured short-term temporal dependencies and were a key contributor to model performance.

**Did you make any important feature transformations?**

Yes, several important feature transformations were applied:

- Lag feature engineering:
  Historical values of the target variables were added as input features using time shifts (lag-4), enabling the model to learn temporal and autoregressive patterns.

- Missing value handling:
  Missing values were handled using causal forward-filling to ensure temporal validity, followed by filling any remaining missing values with zero.

- Feature scaling:
  All input features were standardized using StandardScaler from scikit-learn, with mean and standard deviation computed only on the training set. Target variables were kept in their original scale.

# A5. Training Method(s)

**What training methods did you use?**

We trained a Transformer-based tabular model using mini-batch gradient descent with the AdamW optimizer. The model was trained end-to-end using a masked Mean Squared Error (MSE) loss, which allows training on data with missing or invalid target values by computing the loss only over finite prediction–target pairs.

To improve training stability and efficiency, we used Automatic Mixed Precision (AMP) with bfloat16 on GPU, together with gradient scaling and gradient clipping. A cosine learning rate schedule with linear warmup was applied to stabilize early training and improve convergence. Early stopping based on validation loss was used to prevent overfitting, and the best-performing model checkpoint on the validation set was retained.

# A6. Interesting Findings

One interesting finding from this project is that the Transformer architecture is a highly powerful and flexible deep learning model. Even without extensive fine-tuning, the model was able to achieve strong performance, demonstrating its ability to capture complex temporal and cross-feature dependencies in the data.

Another notable observation is that the private leaderboard score increased consistently across different private score releases, improving from 0.271 → 0.306 → 0.337 → 0.490. This gradual improvement suggests that the Transformer model generalizes well and adapts effectively to different market regimes, reflecting its robustness in modeling the dynamic and non-stationary nature of real-world financial markets.

# A7. Model and Methods

The most important and only model used in this solution is a Transformer model. It directly consumes all raw numerical features and the constructed lag-day target features, learning temporal and cross-feature dependencies in an end-to-end manner without the use of additional models or ensembles.

# A8. Model Execution Time

The execution time of the submission notebook is approximately 12 minutes and 21 seconds on a Kaggle, based on the recorded runtime.

# A9. References

OpenAI. *ChatGPT (ChatGPT Plus)*. https://openai.com/chatgpt/