Model Summary

A1. Background on you/your team

- **Competition Name:** MITSUI & CO. Commodity Prediction Challenge
- **Team Name:** Poidenko Artem
- **Private Leaderboard Score:** 0.514
- **Private Leaderboard Place:** 6

Team Members:

- **Name:** Poidenko Artem
- **Location:** Kyiv, Ukraine
- **Email:** poidenko.artem@lll.kpi.ua

A2. Background on you/your team

What is your academic/professional background?

My name is Artem Poidenko. I am a 4th-year student at the Institute for Applied System Analysis, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute".

Did you have any prior experience that helped you succeed in this competition?

I had no prior experience; this was my first Kaggle competition.

What made you decide to enter this competition?

During the first semester of my 4th year, I took an elective course called "Conflict-Controlled Systems." The professor introduced the class to Kaggle and this competition. He challenged us: anyone who could match his performance and defend their code would receive a high grade for the course. Out of 50 students, I was the only one to accept the challenge. Additionally, the professor inspired me with his own story—he had previously placed in a competition, which directly helped him land his first job in IT.

How much time did you spend on the competition?

I treated the competition like a university research project. I worked independently for about 1-2 days a week. Additionally, I had online check-ins with my professor on Saturdays to review my results and receive feedback. I made my first submission on September 20th.

A3. Summary

Our solution employs a recursive feature forecasting approach using a Long Short-Term Memory (LSTM) network. Instead of predicting the target returns directly, we trained the model to forecast the underlying 424 commodity features for the next 5 days. These feature forecasts are then mathematically transformed into the required target pairs (log returns and spreads). To handle concept drift and short-term market volatility, we implemented a post-processing lag correction step, blending the model's output with the recent trend observed in the label_lags provided by the API.

A4. Features Selection / Engineering

- Feature Selection: We utilized all 424 continuous numerical features provided in the dataset without manual selection, relying on the LSTM's ability to extract relevant temporal patterns.
- Preprocessing:
  - Imputation: Missing values were filled using the global mean of the training set.
  - Scaling: All features were standardized using a StandardScaler (Z-score normalization) to ensure stable training.
  - Windowing: Data was transformed into rolling windows of 50 days to capture medium-term temporal dependencies.
- Important Interactions: The model implicitly learns interactions between "pair" components (e.g., Commodity A vs. Commodity B) by training on the joint history of all features simultaneously.

A5. Training Method(s)

- Architecture: We used a Feature-Forecasting LSTM (Many-to-One) with:
  - Input Dimension: 424
  - Hidden Dimension: 256
  - Output Dimension: 424 (Predicting the next day's feature state)
- Training Strategy:
  - Loss Function: Mean Squared Error (MSE) on feature values.

    - Optimizer: Adam (lr=1e-3).
    - Epochs: 25 epochs with early stopping based on a chronologically split validation set (last 15%).
- Inference: Recursive inference (rolling the window forward with predicted values) was used to generate the 5-day forecast horizon required for the target calculation.

A6. Interesting findings

- **Key "Trick" (Lag Correction):** The most significant performance boost came from the **Label Lag Correction**. We noticed that pure model predictions sometimes drifted from the immediate market regime. By calculating the mean of the label_lags (ground truth from recent past batches) and blending it with our model predictions (α=0.7), we significantly reduced error during volatile periods.
- **Recursive vs. Direct:** Forecasting the *features* first and then deriving the math targets proved more stable than trying to predict the noisy log-returns directly.

A7. Simple Features and Methods

- Simplified Model: A simple "Persistence" model that predicts the next day's return will be the average of the last known label_lags (essentially setting our blending alpha to 0.0) captures a significant portion of the performance baseline.
- Subset of Features: Focusing only on the columns involved in the target definitions (the specific commodity pairs) rather than all 424 features would retain ~90% of performance but ignores cross-commodity correlations.

A8. Model Execution Time

- Training Time: ~2-3 minutes on a standard CPU, or less then 1 minute on GPU P100.
- Prediction Time: < 500ms per batch (very fast due to simple LSTM matrix operations).

A9. References

Main references and resources:

- Competition Resources:

    - Official Kaggle competition documentation (Data description, API usage).
    - Kaggle discussion boards for troubleshooting and submission format clarification.
- Large Language Models (LLMs) for code generation, debugging, and documentation assistance:
    - Google Gemini Pro
    - ChatGPT
    - DeepSeek
- Academic Guidance:
    - Mentorship and feedback from the professor of the "Conflict-Controlled Systems" course at National Technical University of Ukraine "Igor Sikorsky KPI".
- Community & Research:
    - Analysis of public Kaggle Notebooks (Kernels) to understand baseline approaches and feature engineering techniques.
- Official Documentation:
    - PyTorch Documentation: https://pytorch.org/docs/ (for LSTM implementation)
    - Scikit-learn Documentation: https://scikit-learn.org/stable/ (for preprocessing)
    - Pandas & Polars Documentation for data manipulation.