Time Series Models: A Comparative Analysis of Time Series Transformer and LSTM

1.Introduction

I conducted a comparative analysis of stock price prediction using two time series models: the Time Series Transformer from Hugging Face and a Long Short-Term Memory (LSTM) network as a baseline. Both models were employed to forecast future stock prices based on historical data. The models were evaluated using the Symmetric Mean Absolute Percentage Error (sMAPE) metric across four financial indicators: high, low, close, and volume prices. The results demonstrate that the Time Series Transformer outperforms the baseline LSTM model in all evaluated metrics.

2.Methodology

I remove entries whose features are all zero values, as they belongs to the none-tradable time. I believe this is a solid strategy for the following reasons:

- **Eliminate Noise:**

Non-tradable times often contain periods where no transactions happen, resulting in flat or constant values (zeros) that don't contribute any meaningful information to the model. Keeping these entries can

confuse the model, as it tries to learn relationships from these irrelevant periods, introducing noise.

- **Improves Signal-to-Noise Ratio:**

By removing non-tradable periods, I increase the proportion of meaningful data points, allowing the model to focus more on learning real trends and patterns from active trading periods.

- **Better Time-Series Feature Engineering:**

Many time series features like volatility, momentum, or rolling averages are influenced by the continuity of data. Including non-tradable time may artificially flatten or skew these features, leading to inaccurate feature representations. Removing them helps keep these calculations precise.

I used the following models:

1. Time Series Transformer

- **Framework**: Hugging Face Transformers library: https://huggingface.co/docs/transformers/model_doc/time_series_transformer
- **Architecture**: Utilizes self-attention mechanisms to capture long-range dependencies.
- **Implementation**: Configured to process the time series data with appropriate positional embeddings.

2. Long Short-Term Memory (LSTM) Network

- **Role**: Serves as a baseline model for comparison.
- **Architecture**: A recurrent neural network capable of learning order dependence in sequence prediction problems.
- **Implementation**: Configured with layers suitable for capturing temporal patterns in the data.

3.Training and Predicting

I've divided the dataset into a training set and a validation set to ensure that my model can generalize well to unseen data. I stop training when the validation loss increases consistently for 4 epochs (early stopping).

Due to memory constraints on my device, I was unable to generate predictions for all time points in the dataset.

To handle this, I've devised a strategy where I only generate predictions at specific intervals of time points. Specifically, I used every 34th time point, which includes 30 historical time steps and 4 lagged steps.

I believe this approach can still provide meaningful insights, especially when I am interested in periodic or spaced-out forecasts.

Furthermore, during training I didn't group the data by ID, because I used the batch training, which already ensures that the model is exposed to different segments of the data in each iteration. Each batch provides a small window into the overall sequence, and the Transformer is capable of learning from this window using time-based features. Grouping by ID would not necessarily enhance the model's ability to

learn in this context. Furthermore, Positional encodings, a key aspect of Transformer models, allow the model to encode the relative position of each data point in the sequence. These encodings give the model an understanding of the sequence's order without needing to explicitly group data by ID. As a result, the model inherently understands where each batch fits within the larger sequence of time steps.

4.Results

Model	**high**	**low**	**close**	**volume**
Time series transformer	**148.5016**	**144.2690**	**146.6783**	**177.8588**
LSTM(baseline model)	174.5938	199.6960	164.8433	188.6728

5.Training Performance

During training, both models achieved sMAPE scores between 10 and 20 for the high, low, and close prices, indicating effective learning of these features. However, for the volume feature, both models exhibited significantly higher sMAPE scores exceeding 170, even during the training phase. This suggests that the models struggled to learn and generalize patterns associated with trading volume.

6.Discussion

Performance on Price Features

The Time Series Transformer model achieved lower sMAPE scores across all price-related financial indicators compared to the LSTM model, indicating higher prediction accuracy:

- High Prices: The transformer reduced the sMAPE by approximately 26.0922 points compared to the LSTM.
- Low Prices: An improvement of about 55.427 points was observed.
- Close Prices: The transformer outperformed the LSTM by around 18.165 points.

These improvements demonstrate the transformer model's superior ability to capture temporal dependencies and complex patterns in stock price data.

Challenges in Predicting Volume

Despite reasonable performance on price features, both models struggled with predicting the volume feature. Both models had sMAPE scores over 170 for volume during training and testing, indicating poor predictive performance.

Possible Reasons:

- Trading volume can be highly volatile and subject to sudden spikes or drops due to external factors not captured in historical price data.

- Volume may be influenced by news events, market sentiment, or institutional trading activities, which are not reflected in the models' input features.
- The architectures used may not be well-suited to capture the complexities inherent in volume data.

7.Conclusion

The comparative analysis demonstrates that the Hugging Face Time Series Transformer model outperforms the traditional LSTM baseline in predicting stock prices (high, low, and close), effectively capturing temporal patterns in the data. However, both models struggled significantly with predicting the volume feature, even during training, indicating that volume prediction poses inherent challenges not addressed by these models.