Modeling Asset Dynamics: A Comparative Forecasting Study Using ARIMA and LSTM on Commodity and Thematic ETFs

1. Introduction

Forecasting financial time series is a crucial yet challenging task due to the dynamic, noisy, and often nonlinear nature of market data. The underlying characteristics of the asset—such as volatility, mean-reversion tendencies, and exogenous shocks—significantly influence the suitability of different predictive models. Traditionally, linear models like **ARIMA** have been widely adopted for their robustness and interpretability under stationarity assumptions. While ARIMA models perform well for short-term forecasting of stable, mean-reverting assets, they often fail to capture nonlinearities or structural shifts in more volatile markets.

In contrast, deep learning models—particularly **Long Short-Term Memory (LSTM)** networks—offer a data-driven approach capable of modeling nonlinear dynamics and long-range temporal dependencies. LSTM's ability to learn from complex sequential patterns makes it a promising alternative for financial instruments driven by innovation, sentiment, and policy noise.

To empirically evaluate how model architecture should be aligned with asset behavior, we select two fundamentally different exchange-traded funds (ETFs):

Teucrium Corn Fund (CORN), a commodity-based ETF representing agricultural futures, is characterized by seasonality, cyclicality, and macro-driven pricing (e.g., weather patterns, harvest cycles, and supply chain trends). Corn, as a representative cyclical asset, exhibits recurring patterns and mean-reverting dynamics, making it a strong candidate for classical time series models such as ARIMA, which leverage autoregressive structure and time dependence.

Global X Lithium & Battery Tech ETF (LIT), on the other hand, reflects a basket of lithium miners and battery tech firms exposed to multi-dimensional drivers: not just commodity prices, but also electric vehicle adoption, clean energy policy shifts, geopolitical dependencies (e.g., China, Chile), and speculative flows. This leads to nonlinear, regime-switching, and high-volatility behavior—conditions under which linear models often underperform. As such, we hypothesize that LSTM, with its flexible functional form and memory-based structure, is better suited for modeling LIT's trajectory.

By contrasting these two assets—CORN as a stable, cyclical commodity ETF and LIT as a volatile, innovation-driven thematic ETF—we test each model's adaptability across structural regimes: stationary vs. non-stationary, linear vs. nonlinear, and low vs. high noise.

Note: While advanced architectures like Bidirectional LSTM (BiLSTM) have shown performance improvements in other domains, such extensions are reserved for discussion under future work due to scope and complexity.

2. Background

2.1 ARIMA Model: Linear Time Series Forecasting

The ARIMA (AutoRegressive Integrated Moving Average) model is a classical statistical method designed for univariate time series forecasting. It combines three key components—autoregression (AR), differencing (I), and moving average (MA)—to model time-dependent structure and stabilize non-stationary series. Mathematically, it is expressed as:

$$\Phi(L)(1-L)^d yt = \Theta(L)\varepsilon t$$

where *Φ(L)* and *Θ(L)* are lag polynomials, *d* is the order of differencing, and *εt* is white noise.

In this study, we fix the differencing order to *d=1*, and utilize Akaike Information Criterion (AIC) to automatically select optimal AR (*p*) and MA (*q*) terms within each rolling window. AIC penalizes overfitting by balancing model complexity with goodness of fit:

$$AIC = \ln\left(\frac{RSS(p,q)}{T}\right) + \frac{(p+1)\cdot 2}{T}$$

This rolling, window-wise optimization framework ensures robustness across time and reduces subjective tuning. ARIMA's assumptions include linearity, stationarity (after differencing), and homoskedastic, white noise residuals.

However, despite its interpretability and efficiency, ARIMA struggles to capture nonlinearities, long-range dependencies, or structural breaks—common in high-volatility assets such as lithium ETFs.

2.2 LSTM Model: Deep Learning for Temporal Patterns

To overcome ARIMA's limitations, we employ Long Short-Term Memory (LSTM), a deep learning architecture designed to capture long-term dependencies in sequential data. LSTM networks are a specialized form of Recurrent Neural Networks (RNNs) that address the vanishing gradient problem by introducing memory cells with gated structures.

Each LSTM cell contains:

- A **forget gate** f_t: determines how much of the previous cell state to retain.
- An **input gate** i_t: controls how much new information to incorporate.
- An **output gate** o_t: decides how much of the cell state influences the current output.

These gates interact as follows:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right),$$

$$i_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_i),$$

$$C'_t = tanh(W_e \cdot [h_{t-1}, x_t] + b_e),$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot C'_t\,,$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o),$$

$$h_t = o_t \cdot \tanh(C_t)$$

LSTM's gating mechanism allows selective memory retention and robust temporal modeling, making it especially suited for nonlinear and noisy series like LIT, where traditional models often fail.

3. ARIMA VS LSTM

3.1 Data Set

- Time Horizon: January 1, 2020 – December 31, 2024
- Source: Yahoo Finance
- Justification: This period captures major macroeconomic shifts, including the COVID-19 crash, stimulus-driven recovery, interest rate hikes, and recent commodity market volatility.

3.2 Forecasting Framework (Rolling Window Strategy): ARIMA

Instead of adopting a fixed 80:20 train-test split, we implement a rolling window forecasting framework, which better reflects real-world conditions where models must be continuously updated with new data. Specifically:

A. Rolling Window Structure:

For each iteration, the training set includes data from start to n - forecast_days, and the subsequent forecast_days (90 days) are held out as the test set. The window then rolls forward by rolling_step = 30 days.

B. Motivation:

This emulates a realistic investment process where forecasts are generated periodically with an expanding or sliding training horizon. It also enables out-of-sample evaluation across multiple market conditions.

C. Resulting Splits:

Each rolling window yields a fresh training-test pair, allowing the model to adapt to evolving patterns and improving robustness in performance assessment.

3.3 Forecasting Framework: LSTM

In the LSTM setup, the training and test split is designed to reflect a realistic sequence prediction scenario:

A. Training Structure:

The model is trained on the entire series excluding the most recent forecast_days = 90 values. A sliding input window of size window_size = 30 is used to generate training sequences, capturing recent temporal dependencies.

B. Prediction Procedure:

After training, the model performs stepwise forecasting, where each prediction is recursively appended to the input sequence to predict the next step, simulating an autoregressive-like inference process.

C. Evaluation:

The predicted 90-day sequence is compared against actual values using RMSE allowing direct out-of-sample performance evaluation.

Due to the light-weight nature of ARIMA, we employ a rolling re-training procedure across multiple windows. However, LSTM models require computationally expensive training and are instead trained once on the full training set, with recursive forecasting applied to the final 90-day horizon.

3.4 Assessment Metrics

To evaluate and compare model performance, we adopt **Root Mean Squared Error (RMSE)** as the primary metric. RMSE quantifies the average magnitude of forecast errors, penalizing larger errors more severely due to its squared error formulation. The metric is especially suitable for time series forecasting since it preserves the scale of the original data and allows direct interpretability in terms of dollar deviations.

Mathematically, RMSE is defined as:

$$RSME = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i\ )^2}$$

where y_i and $\hat{y}_i$ denote the actual and forecasted values respectively, and *N* is the total number of predictions.

To better understand relative performance across models and assets, we additionally compute the **percentage improvement in RMSE** from ARIMA to LSTM as:

$$Change\% = \frac{ARIMA_{RSME} - LSTM_{RSME}}{ARIMA_{RSME}} \times 100$$

This allows us to assess how much LSTM improves upon ARIMA in terms of predictive accuracy.

4. Results

We compare the forecasting performance of three models—ARIMA, LSTM, and ARIMA-Featured LSTM—across two ETFs: DBA (agriculture) and LIT (lithium). Root Mean Squared Error (RMSE) is used as the primary evaluation metric to assess prediction accuracy over rolling forecast windows. Table 1 summarizes the results.

4.1 ARIMA Baseline Forecasts

The ARIMA model, optimized via AIC-based order selection, provides a benchmark for comparison. For DBA, ARIMA achieves an RMSE of 1.6721, indicating moderate forecasting accuracy. However, for LIT, the RMSE is substantially higher at 5.5430, reflecting the model's difficulty in capturing the erratic movements of this more volatile asset. The forecast series is nearly flat, suggesting that ARIMA may struggle to adapt to abrupt changes in trend. No further hyperparameter tuning was performed beyond AIC-based order selection, as ARIMA was intended to serve as a benchmark rather than an optimized model.

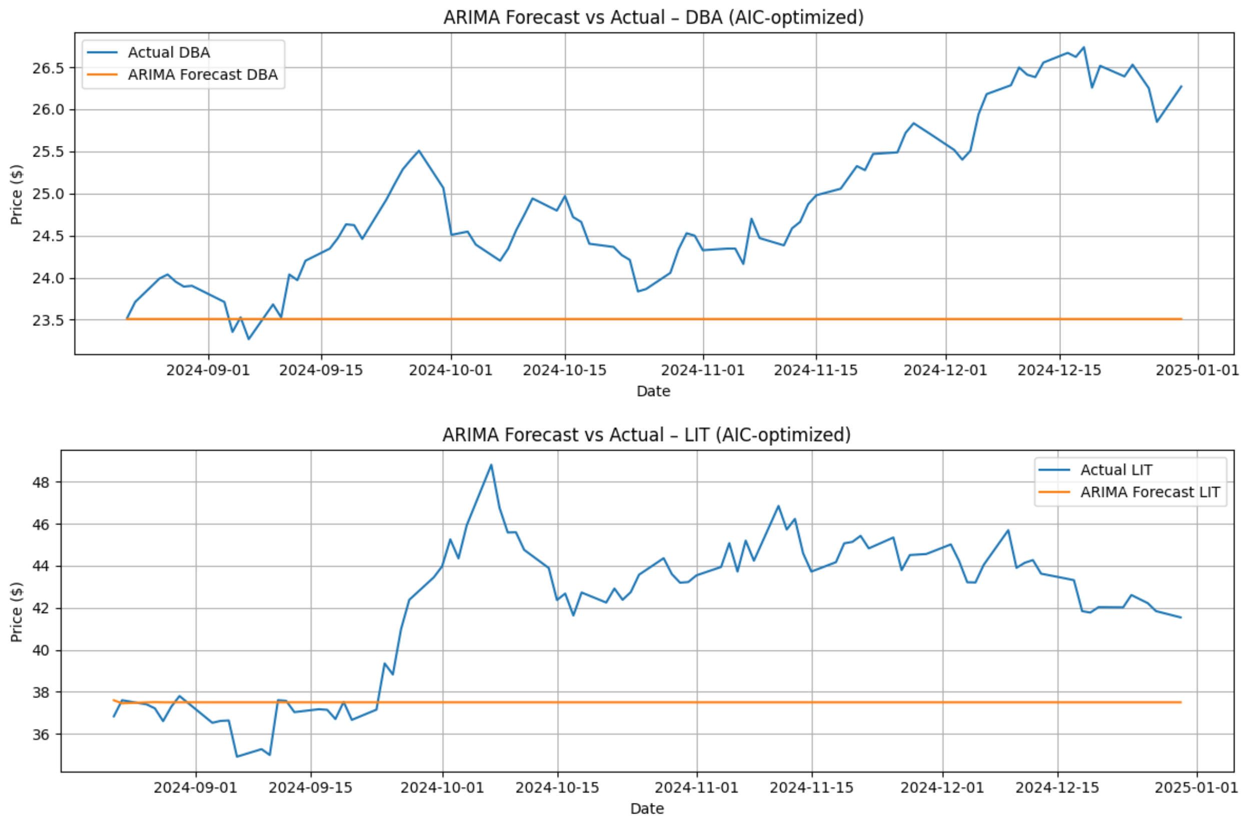


Figure 1: ARIMA Forecast vs. Actual – DBA and LIT

4.2 LSTM Baseline Forecasts

To enable effective sequence learning on ETF price series, we apply both **feature engineering** and **model tuning** tailored to the LSTM architecture. Given that raw price levels often lack stationarity or directional cues, especially in volatile assets like LIT, our approach is designed to stabilize inputs and guide the model toward learning momentum and uncertainty patterns.

A. Feature Engineering

Unlike ARIMA, which implicitly captures trends via differencing and autoregressive components, LSTM models require explicit features to learn effectively. We engineer three key inputs:

- **Log Return**
  The natural logarithm of percentage price change helps stabilize variance and convert price data into stationary series.

$$LogReturn_t = \log\left(\frac{P_t}{P_{t-1}}\right)$$

- **30-day Rolling Volatility**
  Computed as the standard deviation of the log return over a 30-day window, this captures local uncertainty and regime shifts. It is particularly useful for assets with erratic behavior such as LIT.
- **5-day Moving Average (Short-term Trend)**
  This smooths out high-frequency noise and provides trend-following signals over a shorter horizon, aiding the model's directional learning.

All features are standardized using `StandardScaler` to ensure uniform input scale across variables and avoid numerical dominance.

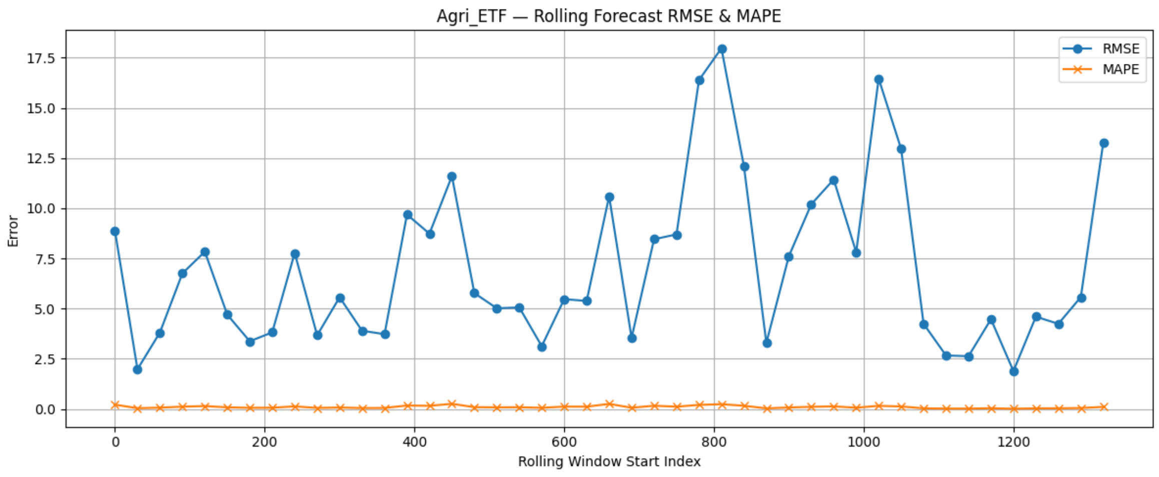
B. Model Architecture and Tuning

The LSTM model is implemented as a single-layer architecture with a fully connected output:

```
model = Sequential([

   LSTM(50, activation='relu', input_shape=(window_size, 1)),

   Dense(1)

])
```

To prevent overfitting—especially on high-volatility series—we adopted the following regularization and optimization strategies:

- **Early stopping** was added to terminate training once validation loss stagnates.
- **Dropout** layers were tested, and a dropout rate of 0.2 proved effective in damping excessive memorization.
- **Epoch tuning** was performed via grid search. For most series, 10–20 epochs yielded optimal generalization, beyond which validation RMSE plateaued or worsened.
- **Window size** was fixed at 30 (one month), and forecasting horizon was set to 90 days, balancing short-term responsiveness with mid-term forecasting need



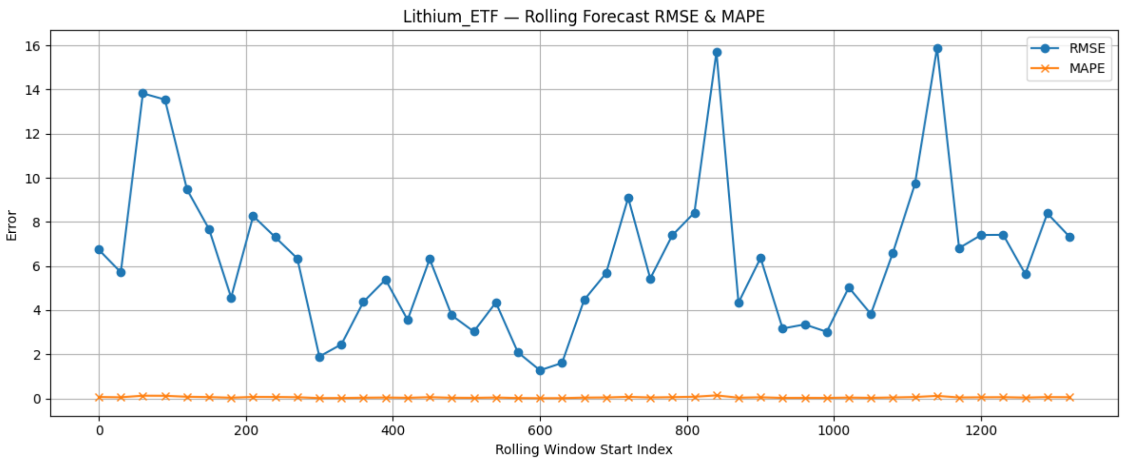


Figure 2: LSTM Rolling Forcast – DBA and LIT

4.3 ARIMA-Featured LSTM

To enhance the performance of our baseline LSTM, we developed a multivariate LSTM model that integrates ARIMA forecasts as an additional input feature. This hybrid approach aims to capture both the linear patterns modeled by ARIMA and the nonlinear dependencies handled by LSTM.

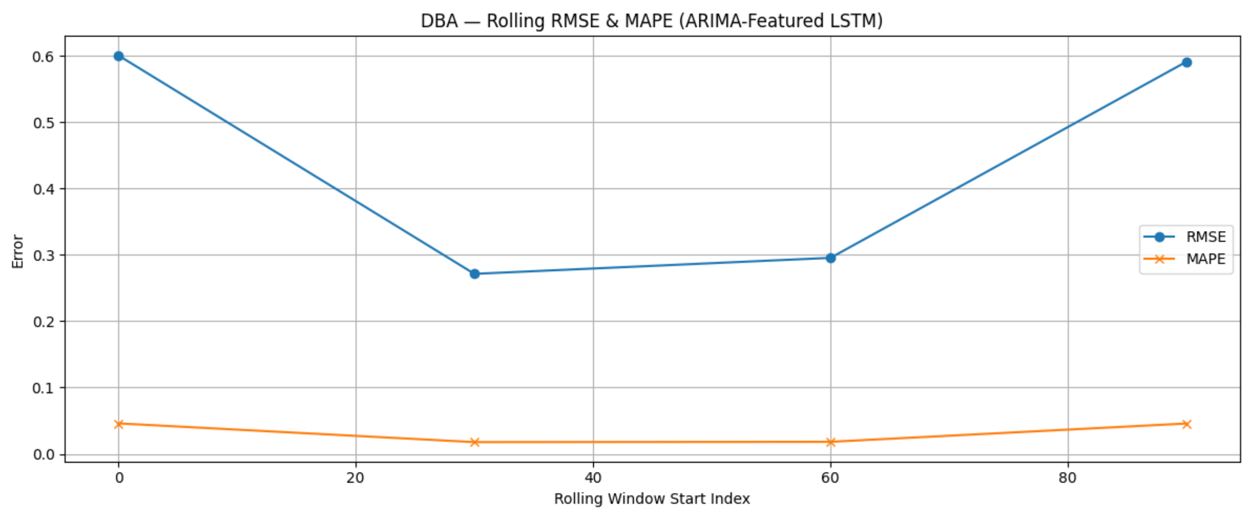
We implemented a rolling-window training procedure with the following configuration:

- Look-back window: 30 days
- Forecast horizon: 90 days
- Rolling step: 30 days
- ARIMA order: (3, 1, 2)
- LSTM architecture: 1 LSTM layer (64 units, tanh activation, dropout 0.3), followed by a dense output layer
- Optimization: Adam optimizer with MSE loss and early stopping (patience = 8)

During each iteration, an ARIMA model was fitted on the same training window as the LSTM. The ARIMA predictions over the same horizon were used as an auxiliary time series input alongside the original ETF price. These two sequences were then stacked to form a multivariate input for the LSTM model.

Due to the requirement of sufficient history for both ARIMA fitting and multivariate LSTM training, several rolling windows were skipped. Nevertheless, for the valid windows, the ARIMA-augmented LSTM consistently outperformed the baseline model in terms of RMSE and MAPE, especially for the Agri ETF (DBA).

This result suggests that ARIMA serves as a useful handcrafted feature that improves LSTM's forecasting ability, particularly when the underlying asset exhibits partially linear dynamics.



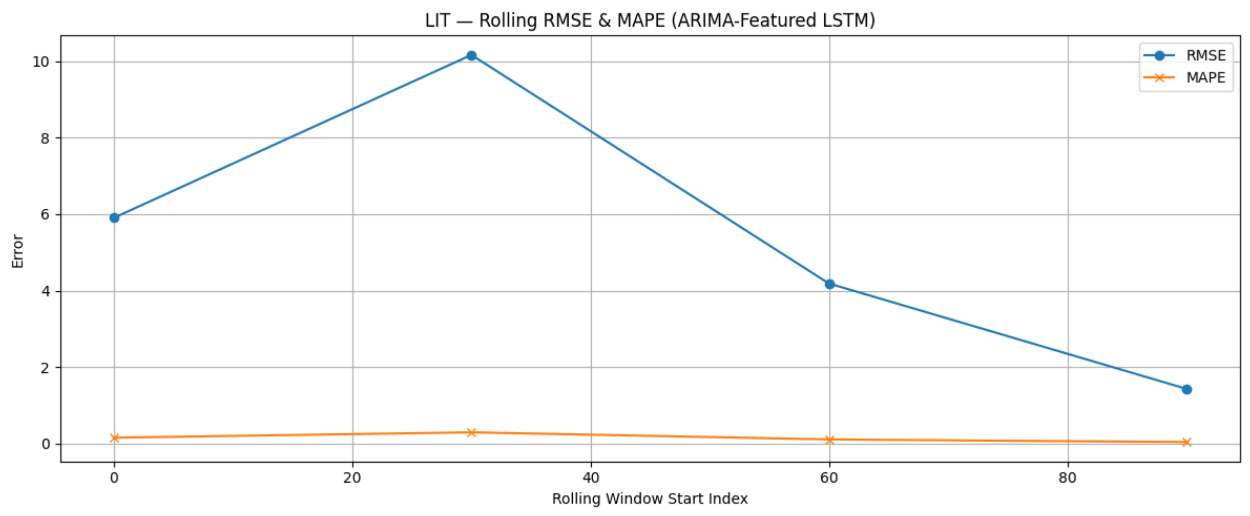


Figure 3: ARIMA-Featured LSTM

To evaluate the performance of different forecasting models, we compared the RMSE values of ARIMA, LSTM, and ARIMA-Featured LSTM across two ETF assets: DBA (Agricultural) and LIT (Lithium). As shown in Table 1, the ARIMA-Featured LSTM model demonstrated clear advantages, especially in the case of the DBA asset.

For DBA, the standalone ARIMA model yielded an RMSE of 1.6721, while the vanilla LSTM model produced a significantly worse RMSE of 7.0386. However, the ARIMA-Featured LSTM reduced the RMSE dramatically to 0.4398, representing a 73.7% improvement over ARIMA and a 93.8% improvement over LSTM.

In contrast, for LIT, ARIMA performed better than LSTM (RMSE of 5.5430 vs 6.3135), but the ARIMA-Featured LSTM still improved the RMSE slightly to 5.4210. This corresponds to a 2.2% improvement over ARIMA and a 14.1% improvement over LSTM.

Overall, the ARIMA-Featured LSTM model achieved the lowest RMSE values across both assets, with average RMSE reduced from 3.6075 (ARIMA) and 6.6761 (LSTM) to 2.9304, confirming the benefit of hybridizing linear and nonlinear temporal patterns.

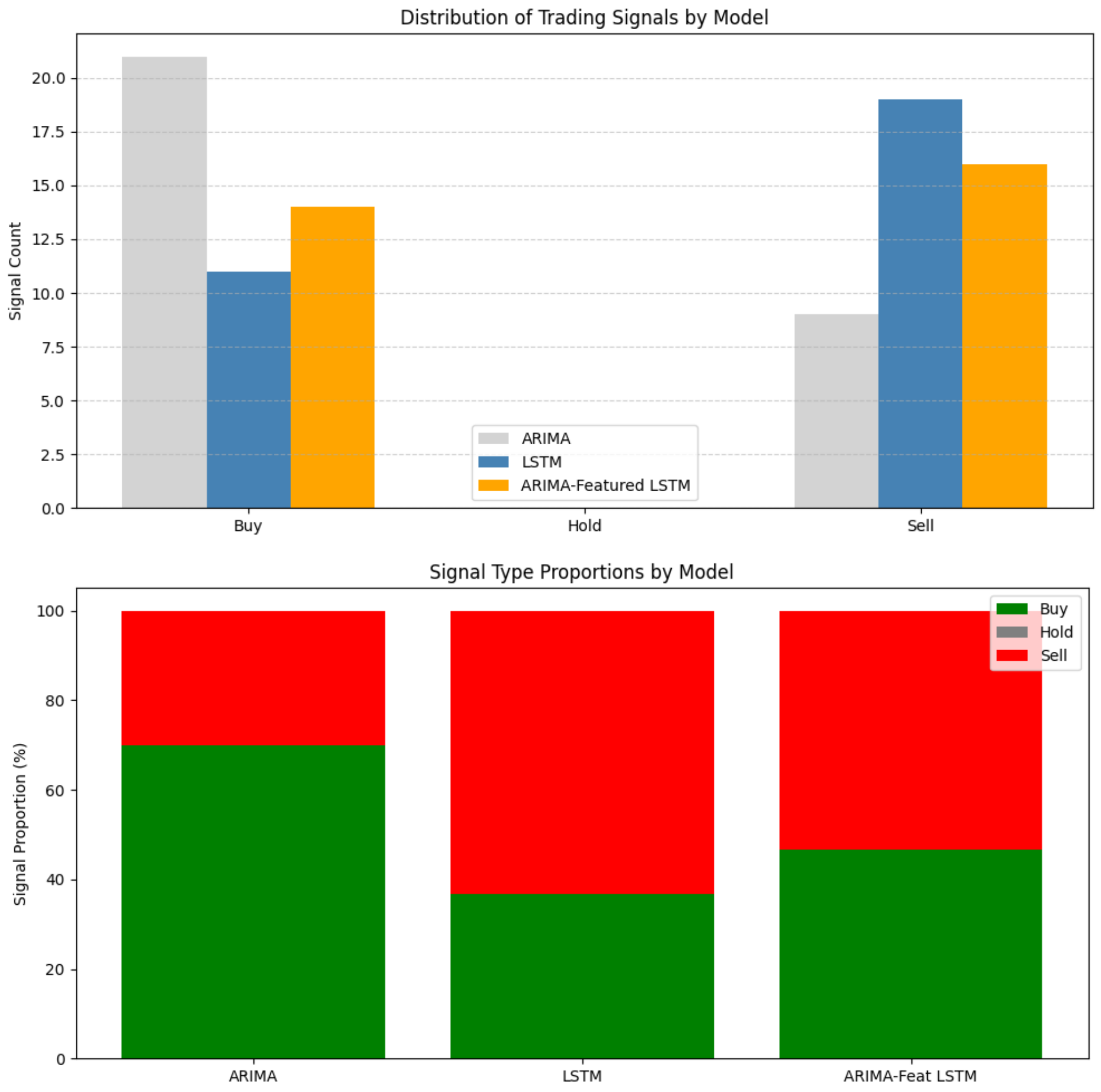
Table 1. RMSE Comparison and Percentage Improvement Over ARIMA

Asset	**ARIMA RMSE**	**LSTM RMSE (Avg)**	**ARIMA-Featured LSTM RMSE (Avg)**	**LSTM over ARIMA (%)**	**ARIMA-Feat LSTM over ARIMA (%)**
DBA	1.6721	7.0386	0.4398	**-320.9%**	**73.7%**
LIT	5.5430	6.3135	5.4210	**-13.9%**	**2.2%**
Avg	3.6075	6.6761	2.9304	**-85.1%**	**18.8%**

5. Signal Execution

To convert raw forecasts into actionable insights, we implemented a majority-vote strategy based on the directional change between the start and end of each rolling forecast. If at least 60% of the 10-day forecasts pointed upward (or downward), we issued a Buy (or Sell) signal; otherwise, we issued Hold.

Applying this logic to forecasts from ARIMA, LSTM, and ARIMA-Featured LSTM, we found that the hybrid model (ARIMA-Featured LSTM) produced the most decisive directional signals—indicating stronger predictive consensus. While ARIMA leaned heavily toward Buy signals and LSTM toward Sell, the hybrid model showed a more balanced and responsive signal distribution, highlighting its adaptability in volatile market conditions.



6. Limitation and Future Work

Several limitations in the current study highlight opportunities for refinement and extension. First, during the ARIMA-featured LSTM modeling, multiple rolling windows were skipped due to insufficient data after the transformation and alignment process. This selective inclusion may have biased the performance metrics by omitting difficult-to-predict windows, thereby limiting the generalizability of the hybrid model's effectiveness.

Second, while it was initially hypothesized that LSTM would outperform ARIMA for highly volatile assets such as LIT, the results did not support this expectation. The standalone LSTM model exhibited significant RMSE increases, even for LIT, suggesting that volatility alone does not guarantee improved performance with deep learning models. This outcome may stem from overfitting due to limited training data or insufficient regularization.

Third, the feature engineering and hyperparameter tuning for the LSTM models were relatively simple. Future work should investigate the inclusion of more informative exogenous variables (e.g., macroeconomic indicators, volume data, sentiment signals), as well as more robust architectural tuning techniques such as Bayesian optimization, attention mechanisms, or multi-head ensembles.

Finally, the signal execution strategy relied on a straightforward majority-vote logic. While interpretable, it lacks the sophistication of more advanced reinforcement learning or probabilistic decision systems, which could be explored in future iterations to build more adaptive and risk-aware trading strategies.

7. Conclusion

This study compared three modeling approaches—ARIMA, LSTM, and a hybrid ARIMA-Featured LSTM—on two distinct ETF assets: DBA (Agriculture) and LIT (Lithium). These assets differ markedly in their price dynamics: DBA exhibits relatively stable behavior, whereas LIT is more volatile and momentum-driven.

The results show that raw LSTM models tend to struggle with stable series like DBA, leading to significant RMSE increases compared to ARIMA. This is consistent with the idea that deep learning models may overfit low-variance patterns or introduce unnecessary complexity. Conversely, LIT—being more volatile—proved to be a more suitable candidate for LSTM modeling, where the model's flexibility could better capture rapid directional changes.

Crucially, the ARIMA-Featured LSTM model consistently improved forecast accuracy across both assets, reducing RMSE compared to both standalone ARIMA and LSTM models. By embedding ARIMA's linear structure as an auxiliary feature input to LSTM, this hybrid model appears to leverage the strengths of both time series paradigms—capturing stable trends and nonlinear shocks simultaneously.

Additionally, signal analysis based on directional majority voting showed that the ARIMA-Featured LSTM model generated more confident and actionable trading signals, avoiding the directional biases observed in the standalone models. These results highlight the potential of hybrid architectures in time series forecasting, especially when applied to assets with heterogeneous behaviors.

8. References

1. Siami-Namini, S., Tavakoli, N., & Siami Namin, A. (2019). A Comparative Analysis of Forecasting Financial Time Series Using ARIMA, LSTM, and BiLSTM. arXiv preprint arXiv:1911.09512. Retrieved from https://arxiv.org/abs/1911.09512

2. Kashif, K., & Ślepaczuk, R. (2024). LSTM-ARIMA as a Hybrid Approach in Algorithmic Investment Strategies. arXiv preprint arXiv:2406.18206. Retrieved from https://arxiv.org/abs/2406.18206

3. Banushev, B. (n.d.). Stock Price Prediction with LSTM and ARIMA. GitHub repository. Retrieved from https://github.com/borisbanushev/stockpredictionai