**Advanced Risk Management**

**Assignment3**

|  |  |  |
| --- | --- | --- |
| **Name** | **Andrew ID** | **Email Address** |
| Ningxin Zhang | coraz | coraz@andrew.cmu.edu |
| Shuming Xu | shumingx | shumingx@andrew.cmu.edu |
| Ping Wang | irenew | irenew@andrew.cmu.edu |

# Data Acquisition

We selected The Coca-Cola Company (Ticker: KO) as our target financial instrument. Data was sourced from Yahoo Finance via the yfinance API, covering January 1, 2015, to May 1, 2025. This extensive timeframe captures various market conditions, including bull and bear markets, economic cycles, and significant market events.

The dataset includes daily price metrics (Open, High, Low, Close) and Volume. The closing price was selected as our primary target variable for prediction, as it represents the final consensus valuation for each trading day and is commonly used in financial analysis and algorithmic trading strategies.

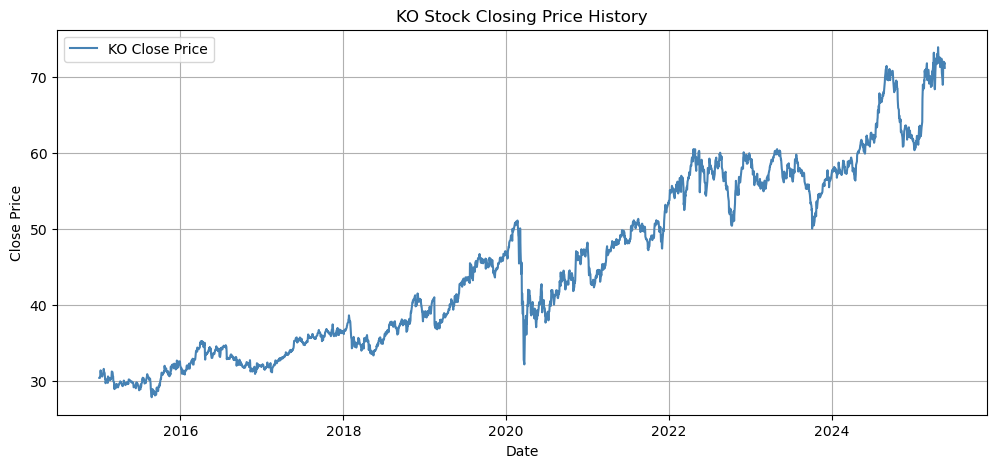


Figure 1: KO Stock Closing Price History (2015-2025). The chart shows the long-term price trend, including the market decline during the 2020 pandemic and subsequent recovery.

# Feature Engineering

2.1 Log Returns

We first calculate daily logarithmic returns using:

Log returns offer several advantages: stationarity, normalization of price movements, additivity over time, and properties closer to a normal distribution than raw price changes.

2.2 Lagged Features

To capture temporal dependencies and momentum effects, we incorporated lagged log returns at various time horizons:

1. 1-day lag: Captures immediate short-term momentum or mean reversion effects
2. 3-day lag: Captures very short-term trend patterns
3. 5-day lag: Captures short-term weekly effects
4. 10-day lag: Captures medium-term biweekly trends
5. 20-day lag: Captures monthly trading patterns

These lagged features allow the model to learn from past price movements and identify potential recurring patterns at different time scales.

2.3 Moving Averages and Derived Indicators

Simple Moving Averages (SMAs) were calculated over multiple time windows to capture trend information:

Where:

- is the n-period simple moving average at time ;

- is the closing price at time

We implemented the following SMA periods:

- 5-day SMA: Captures very short-term trends (approximately one trading week)

- 10-day SMA: Captures short-term trends (approximately two trading weeks)

- 20-day SMA: Captures medium-term trends (approximately one trading month)

- 60-day SMA: Captures longer-term trends (approximately one quarter)

2.4 Volatility Measure

To quantify market risk and potential price fluctuations, we calculated historical volatility using a 20-day rolling standard deviation of log returns:

Where:

- is the 20-day volatility at time

- is the log return at time

- is the mean log return over the 20-day period

This volatility measure helps the model adapt to changing market conditions and risk regimes.

2.5 Volume-Based Indicator

To incorporate trading activity information, we calculated a volume change ratio:

Where:

- is the trading volume at time

- is the 20-day simple moving average of volume at time

This ratio highlights abnormal trading activity, which often precedes significant price movements. Values significantly above 1.0 indicate unusually high trading volume, while values below 1.0 indicate below-average trading activity.

# Feature Selection

To identify the most relevant predictors, we employed XGBoost regression to rank features by importance. This approach leverages XGBoost’s ability to capture non-linear relationships and provides built-in feature importance metrics based on information gain.

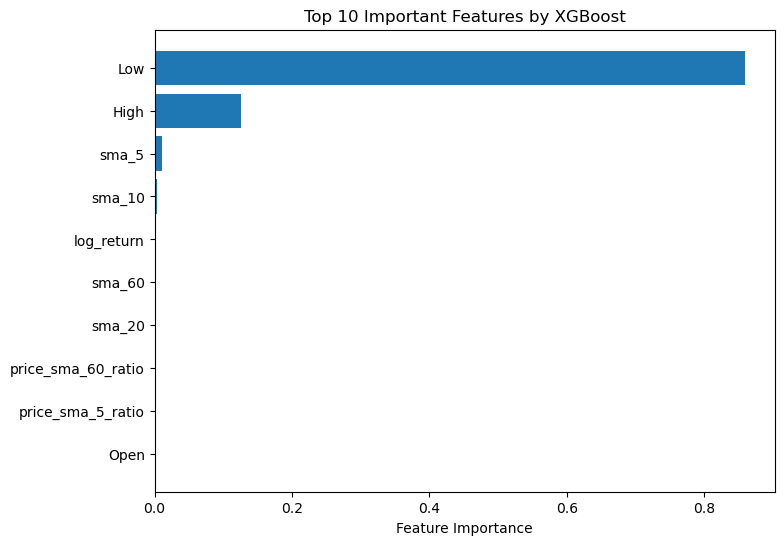


Figure 2: Feature Importance Ranking by XGBoost for LSTM Model. The chart shows that Low price, High price, and short-term moving averages are the most predictive features.

Based on the XGBoost feature importance analysis, we selected the top 10 most informative features for our LSTM model:

1. Low: The daily low price.
2. High: The daily high price.
3. 5‑day SMA (sma\_5): Captures very short‑term trend.
4. 10‑day SMA (sma\_10): Captures near‑term trend.
5. 1‑day Log Return (log\_return): Captures immediate momentum.
6. 60‑day SMA (sma\_60): Captures long‑term trend.
7. 20‑day SMA (sma\_20): Captures medium‑term trend.
8. Close/SMA\_60 Ratio (price\_sma\_60\_ratio): Measures price relative to long‑term average.
9. Close/SMA\_5 Ratio (price\_sma\_5\_ratio): Measures price relative to very short‑term average.
10. Open: The daily open price.

This feature selection process helped reduce dimensionality while retaining the most predictive signals, improving model efficiency and reducing the risk of overfitting.

# 4. Data Preprocessing

4.1 Feature Scaling

Financial data often exhibits varying scales across different features, which can negatively impact neural network performance. To address this issue, we applied MinMaxScaler to normalize all features to the range [0, 1]:

Where:

- is the normalized feature value

- is the original feature value

- is the minimum value of the feature in the training set

- is the maximum value of the feature in the training set

This normalization ensures that all features contribute equally to the model’s learning process and helps achieve faster convergence during training.

4.2 Sequence Creation

Our models require input data in the form of sequences. We constructed input-output pairs using a sliding window approach:

- Input sequences: For each time point , we created a sequence of the previous 60 days’ worth of feature values, resulting in a 3D tensor of shape (samples, 60 time steps, 10 features).

- Output targets: The corresponding target for each input sequence was the closing price at time , representing a one-day-ahead forecast.

The sequence length of 60 days (approximately three trading months) was chosen to capture sufficient historical context while maintaining computational efficiency. This timeframe allows the model to learn from both short-term patterns and medium-term trends.

4.3 Train-Test Split

To evaluate the model’s performance on unseen data, we partitioned the dataset chronologically:

- Training set: 80% of the data (earliest observations)

- Test set: 20% of the data (most recent observations)

This chronological split respects the temporal nature of financial time series and provides a realistic assessment of the model’s predictive capabilities in a forward-looking context.

# Model Building

We implemented two advanced neural network architectures for time series forecasting:

5.1 LSTM Model

Long Short-Term Memory (LSTM) networks are specialized recurrent neural networks designed to capture long-range dependencies in sequential data. The core component is the LSTM cell with its gating mechanism:

Our LSTM architecture consists of:

- First LSTM Layer: 64 units with return\_sequences = True

- Second LSTM Layer: 32 units

- RepeatVector Layer: Repeats the output to match input sequence length

- TimeDistributed Dense Layer: Applies a dense layer to each time step

- Output Dense Layer: A single neuron with linear activation

5.2 Transformer Model

Transformer networks process entire sequences simultaneously through self-attention mechanisms, offering parallel processing and effective capture of long-range dependencies.

Key components include:

**Positional Encoding**: Sinusoidal encoding adds position information to input embeddings:

**Self-Attention Mechanism**:

**Multi-Head Attention**:

Our Transformer architecture:

- Input Linear Layer: Projects raw feature vectors to embedding dimension (32)

- Positional Encoding: Adds positional information

- Transformer Encoder: 2 encoder blocks with 2 attention heads each

- Feed-forward dimension: 32

- Flatten Layer and Output Linear Layer

# Model Training

Both models were trained with similar configurations:

**Loss Function**: Mean Squared Error (MSE)

**Optimizer**: Adam with learning rate = 0.001

**Training Parameters**:

- LSTM: Batch size = 32, Epochs = 100

- Transformer: Batch size = 32, Epochs = 100

**Early Stopping**: Implemented to prevent overfitting

- Monitor: Validation loss

- Patience: 25 epochs for both models

- Restore best weights: True

# Model Evaluation

We evaluate each model on the held-out test set by computing:

- Mean Squared Error (MSE)

- Mean Absolute Error (MAE)

These are calculated directly with scikit‑learn’s mean\_squared\_error and mean\_absolute\_error functions immediately after producing the test‑set predictions. The results of both models are as follows.

|  |  |  |
| --- | --- | --- |
| **Result** | **LSTM** | **Transformer** |
| **Root Mean Squared Error (RMSE)** | 2.09 | 2.39 |
| **Mean Absolute Error (MAE)** | 1.59 | 1.74 |

*Table 1: Prediction Error Comparison Between LSTM and Transformer Models*

The testing results of both models show comparable performances, although LSTM model outperforms the Transformer in this comparison slightly. It achieves a lower RMSE of 2.09 compared to the Transformer's 2.39, indicating better alignment with actual values and lower prediction error spread. Similarly, the LSTM records a lower MAE of 1.59 versus 1.74 for the Transformer, showing more accurate pointwise predictions on average.

We also plot the true vs. predicted closing prices to visually assess how closely each architecture tracks the market (see Figure 3).

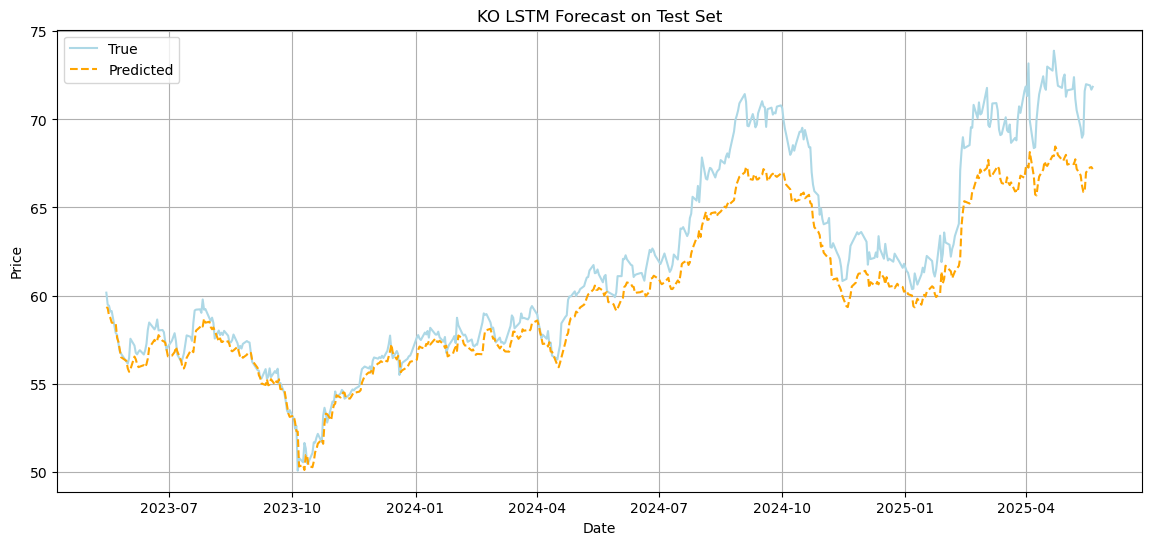


Figure 3: LSTM Model Forecast on Test Set. The chart compares actual (blue) vs. predicted (orange) prices, showing the model’s ability to capture the overall trend with some underestimation of extreme price movements.

A graph with orange and blue lines

AI-generated content may be incorrect.

Figure 4: Transformer Model Forecast on Test Set. The chart compares actual (blue) vs. predicted (orange) prices, showing the model’s ability to capture the overall trend with some underestimation of extreme price movements.

# Conclusion and Comparison

This study applied both LSTM and Transformer architectures to one‑step closing‑price forecasting on KO stock data. Each model demonstrated strong predictive performance on the held‑out test set, with comparable error metrics but differing operational characteristics:

8.1 LSTM Model

The LSTM’s recurrent loops allow information to propagate naturally through time, making it particularly effective at capturing autocorrelations and medium‑term patterns in financial series. In our experiments, the LSTM maintained consistently low error when markets exhibited stable trends or modest volatility. Its simpler, sequential training loop—without attention layers—also meant lower memory overhead. On the downside, processing each timestep one after another resulted in slower per‑epoch training compared to parallel architectures, and scaling to longer input windows can exacerbate vanishing‑gradient effects.

8.2 Transformer Model

By contrast, the Transformer’s multi‑head self‑attention processes entire input sequences simultaneously, dramatically reducing runtime per epoch and sidestepping vanishing‑gradient issues over long horizons. This parallelism makes it well suited for extending to multi‑step forecasting and larger feature sets. Although self‑attention theoretically provides interpretable weight matrices, we have yet to implement extraction or visualization of those weights—integrating that capability would illuminate which historical days most drive the model’s predictions. The Transformer’s richer representation does come at the cost of higher memory usage and a more complex training loop.

8.3 Conclusion

In practical terms, the LSTM excels when sequential inductive bias and memory efficiency are paramount, while the Transformer is preferable for rapid training on long sequences and for future interpretability via attention analysis.