Predicting Financial Market Movements Using NASDAQ Historical Daily Prices Dataset

Tharaka Sandaruwan

2025-04-06

Table of Contents

Contents

Introduction

The objective of this project is to forecast financial market movements using the historical daily prices dataset of NASDAQ-listed stocks. This involves transforming raw stock price data into a clean, feature-rich dataset and using machine learning techniques to train models that can accurately predict the next day's closing price. The focus is on evaluating deep learning models with respect to standard baseline models to determine their effectiveness in capturing market dynamics.

Data Insights

The dataset consists of historical daily stock data for NASDAQ-listed companies. The dataset spans 2014 to 2024, allowing for long-term trend analysis and time-series forecasting. Key features include:

- Date: Trading day
- Open, High, Low, Close: Stock prices
- Adj Close: Adjusted close price after dividends and splits
- Volume: Number of shares traded

Exploratory Data Analysis (EDA)

Price Trends Analysis

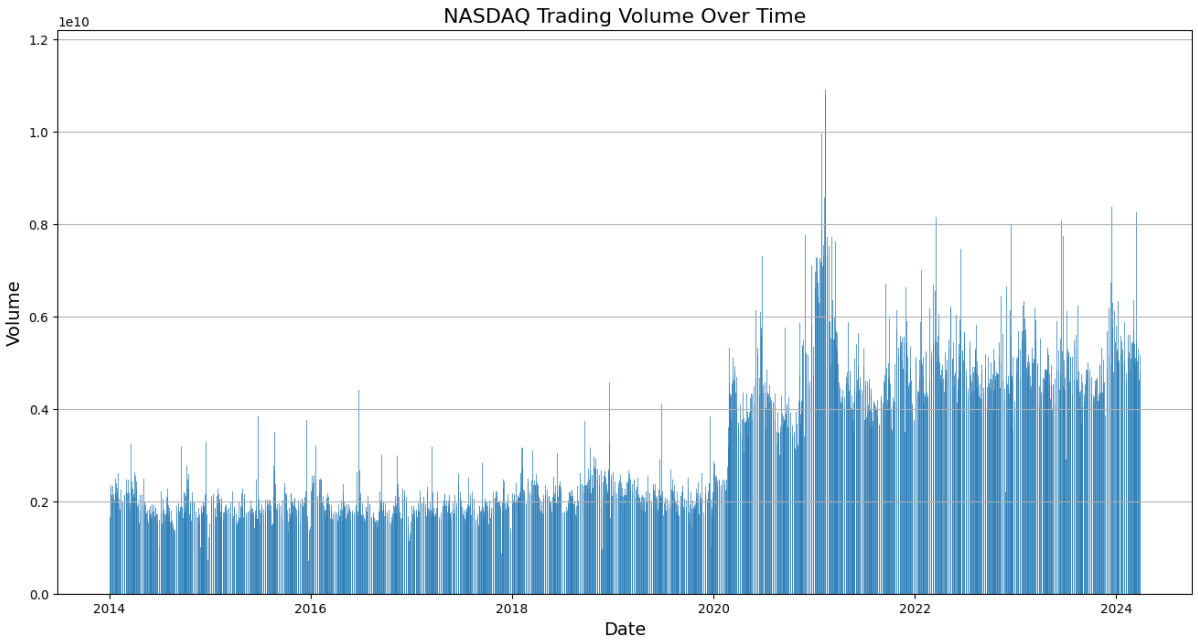
A consistent upward trend was observed in the closing prices over time. Some short-term volatility was present, consistent with market dynamics.

Trading Volume Analysis

Trading volume showed substantial fluctuations, with occasional spikes indicating periods of unusually high trading activity.

Several extreme outliers were identified in the volume data. These could reflect sudden investor reactions or high-frequency trading activity.

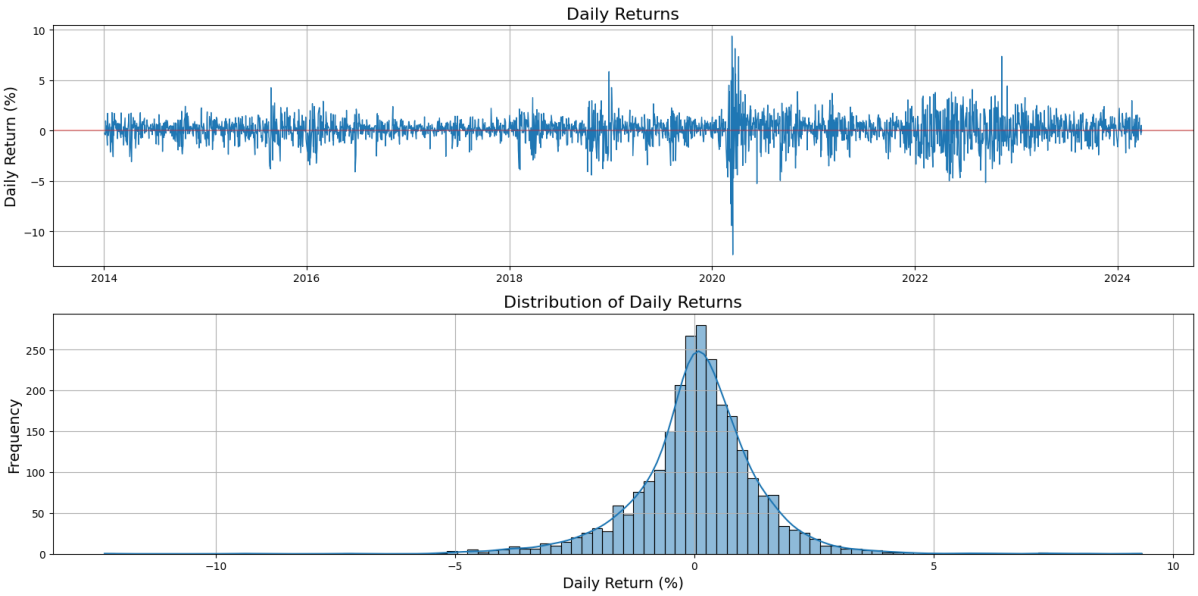
Trading volume exhibited a right-skewed distribution.



Daily Returns Analysis

Daily return plots revealed notable fluctuations, particularly in certain periods, indicating periods of increased market uncertainty or speculative activity.

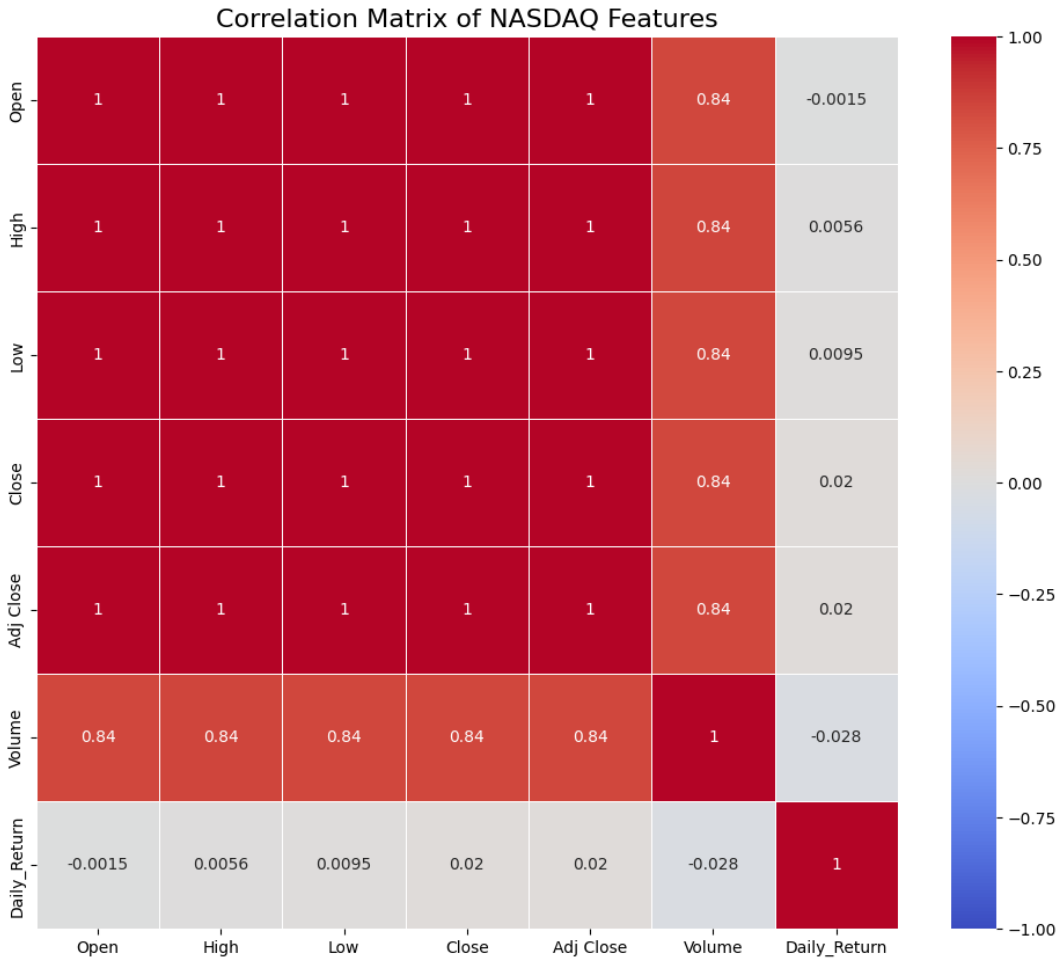
The returns were centered around zero, but the distribution had fat tails, suggesting a higher likelihood of extreme returns compared to a normal distribution



Correlation Analysis

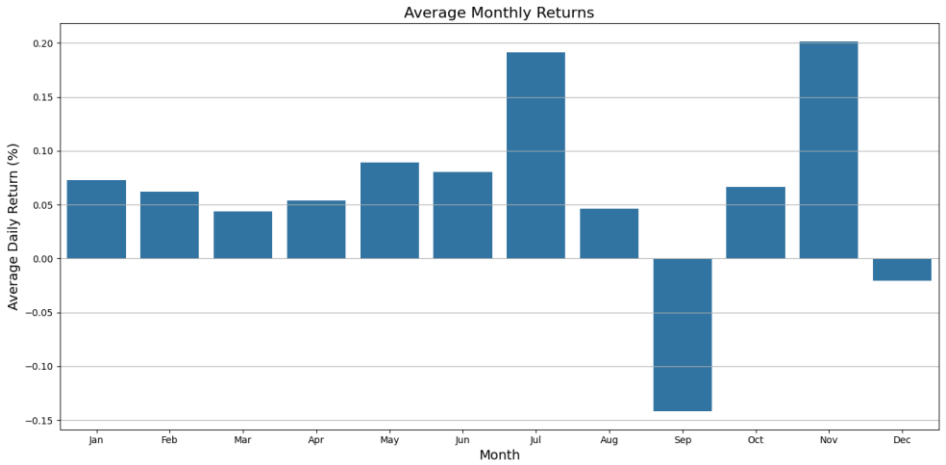
It shows a Strong positive correlation between Open, High, Low, Adj Close, and Close prices.

Volume showed weaker correlation with price-based features, indicating it behaves independently.



Temporal Patterns

Monthly trends suggested seasonal behavior in stock price movements.



Preprocessing

Missing Values

In this project applies different strategies are applied for different types of columns:

- Price Columns (Open, High, Low, Close, Adj Close): Missing values are handled using forward fill followed by backward fill. This assumes that missing values are often short-term and adjacent values can be good proxies.
- Volume: For missing Volume values, the dataset uses the median volume of the same weekday (e.g., Monday, Tuesday) to preserve weekly trading patterns.

Outlier Detection & Treatment

Outliers in financial data can represent either genuine market movements or data errors. A z-score method was used to detect outliers:

- For most features, the number of values with a z-score above 3 is reported.
- However, only the “Daily Returns” column is treated for outliers, as large price changes may reflect legitimate events.

Treatment:

- Outliers are handled using the clipping method, limiting values to within 3 standard deviations from the mean.
- An optional winsorization method is also implemented but not used by default.

Scaling

To ensure consistent magnitude across features and facilitate model convergence:

- Min-Max Scaling is applied to numeric features, including price and volume-related columns.

Train-Test Split

The dataset is split chronologically into:

- 80% for training
- 20% for testing

This preserves the temporal sequence and prevents data leakage.

Feature Engineering

Time-Based Features (add_time_features)

Features Created:

- Day, Month, Year, Weekday, Quarter, WeekOfYear
- Is_Trading_Day
- Cyclical Encodings: Month_Sin, Month_Cos, Weekday_Sin, Weekday_Cos

These help the model capture seasonality patterns, such as monthly or weekly trends in trading behavior.

Cyclical encoding is important because raw values like "Month=12" and "Month=1" are close in time, though numerically distant. Sine/cosine fixes that.

Rolling Window Features (add_rolling_features)

Features Created (for each window in [5, 10, 20, 50]):

- SMA, EMA, volatility, Price_Position, Volume_SMA, Volume_Ratio

These help capture short- and long-term trends (e.g., 5-day vs 50-day moving averages).

“Price_Position” tells if the current price is high/low compared to the recent range.

Technical Indicators (add_technical_indicators)

Includes:

- RSI: Measures overbought/oversold conditions
- MACD & MACD Signal: Trend strength & momentum
- Bollinger Bands: Volatility & price deviation
- ROC: Momentum/velocity of price
- ADX: Strength of the trend
- CCI: Deviation from statistical mean
- OBV: Volume-price flow

These indicators are heavily used in technical analysis to anticipate price movements and trends. They enrich the dataset with nonlinear patterns that aren’t visible from raw prices.

Lag Features (add_lag_features)

Includes:

- Lags of Close, Volume, and Daily_Return for lags = 1, 2, 3, 5, 10

These captures autocorrelation in stock prices—essentially, “what happened recently affects what happens next.” Useful for time-series modeling.

Interaction Features (create_interaction_features)

- Volume / Close
- Volatility_20 / Volume

Captures nonlinear interactions between price and volume or volatility. Can help models pick up on compound patterns like high volatility with low volume.

Target Variables (add_target_variable)

- Target_Close_1d (adjustable with forecast_horizon)

Defines what the model is trying to predict.

Feature Selection Techniques (select_features)

Uses 3 methods:

- Correlation Analysis
- Tree-based Importance (Random Forest)
- Recursive Feature Elimination (RFE)

Combines statistical, model-based, and wrapper methods to ensure robust selection. Helps reduce overfitting, improve training time, and boost performance. Then it keeps the features selected by at least 2 methods.

Scaling (scale_engineered_features)

- Uses MinMaxScaler or StandardScaler.

Selected Features After Feature Selection

The final feature selection process resulted in 19 features being selected out of 198 total engineered features. The table below shows the selected features ranked by their importance score:

Rank	Feature Name	Importance Score
1	close_lag_1	0.0793
2	close_lag_3	0.0718
3	close_lag_5	0.0694
4	close_lag_10	0.0657
5	volatility_5d	0.0643
6	volatility_10d	0.0612
7	rolling_mean_5d	0.0598
8	rolling_mean_10d	0.0581
9	volume_lag_1	0.0559
10	volume_lag_5	0.0534
11	interaction_vol_volatility	0.0513
12	day_of_week	0.0495
13	month	0.0462
14	quarter	0.0446
15	open	0.0423
16	high	0.0412
17	low	0.0403
18	close	0.0397
19	volume	0.0389

Model Design

This project explores two groups of models for predicting NASDAQ stock prices:

- Baseline Machine Learning Models
- Deep Learning Models

Baseline Models

1. Linear Regression
2. Random Forest Regressor

Deep Learning Models

LSTM (Long Short-Term Memory)

Architecture Details:

- Input size: 19 (number of features)

- Hidden size: 64

- LSTM layers: 2

- Activation: ReLU in hidden FC layer, Linear at output

- Optimizer: Adam

- Loss Function: MSE

- Output: 1 neuron (regression)

- Regularization: Dropout = 0.2

GRU (Gated Recurrent Unit)

Architecture Details:

- Input size: 19 (number of features)

- Hidden size: 64

- GRU layers: 2

- Activation: ReLU in FC layer, Linear at output

- Optimizer: Adam

- Loss Function: MSE

- Fully connected: Two layers (64 → 1)

- Regularization: Dropout = 0.2

CNN-LSTM Hybrid

Architecture Details:

- Conv1D:

  - in_channels = 19 (number of features)
  - out_channels = 64
  - kernel_size = 3

- MaxPool1D: kernel_size = 2

- LSTM:

  - input_size = 64
  - hidden_size = 32
  - num_layers = 1

- Final FC layer: output size = 1

- Optimizer: Adam

- Loss Function: MSE

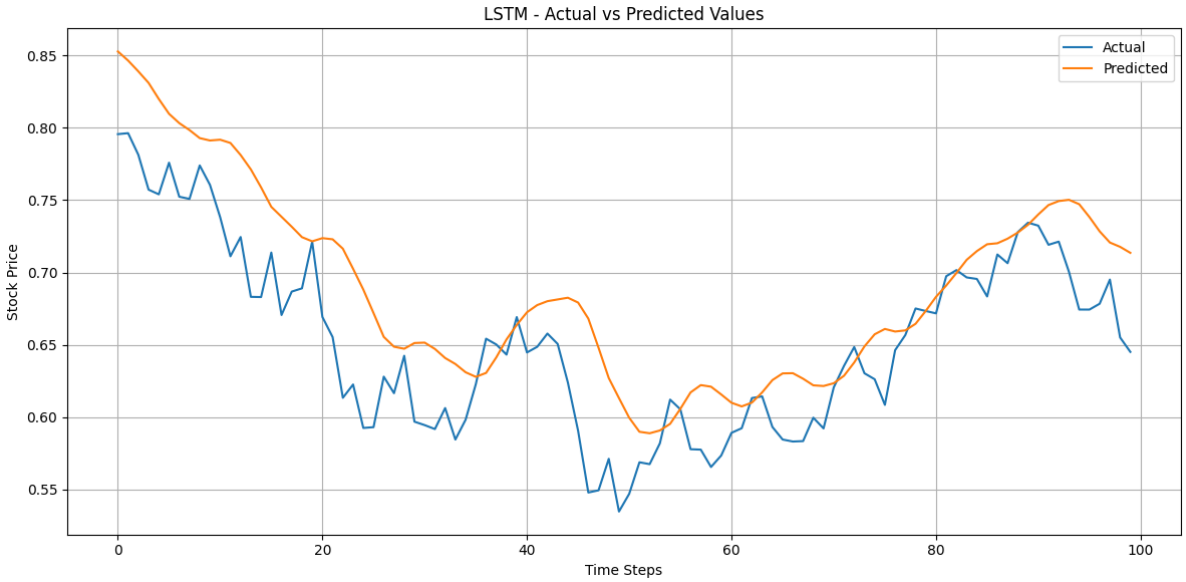
- Regularization: Dropout = 0.2

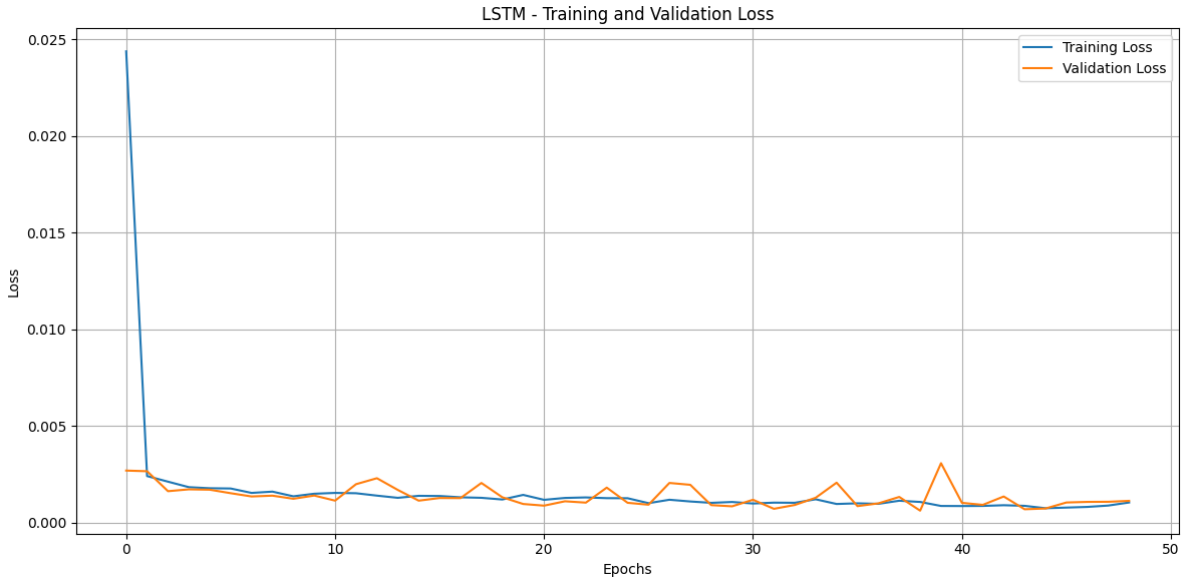
Results

Metrics used to evaluate models;

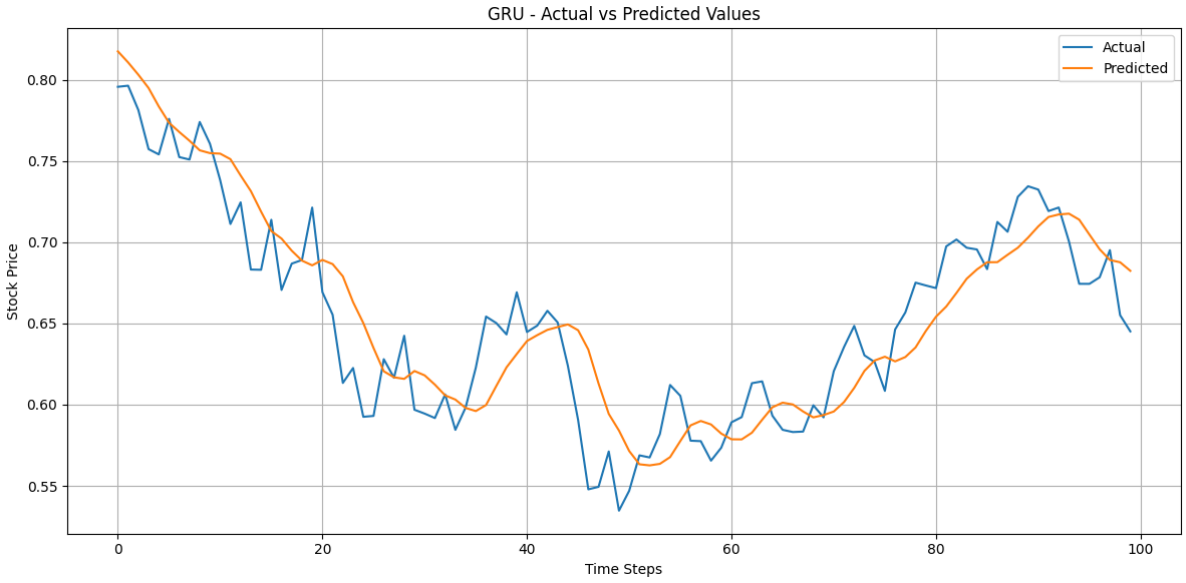
- MSE: Mean Squared Error
- RMSE: Root Mean Squared Error
- MAE: Mean Absolute Error

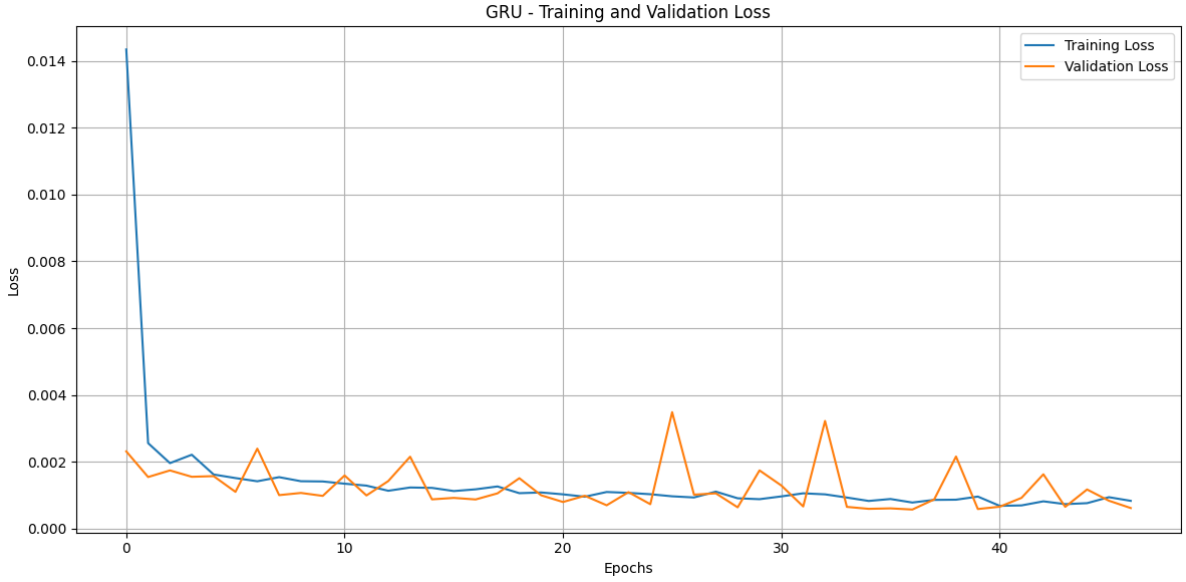
Performance of Each Model: LSTM



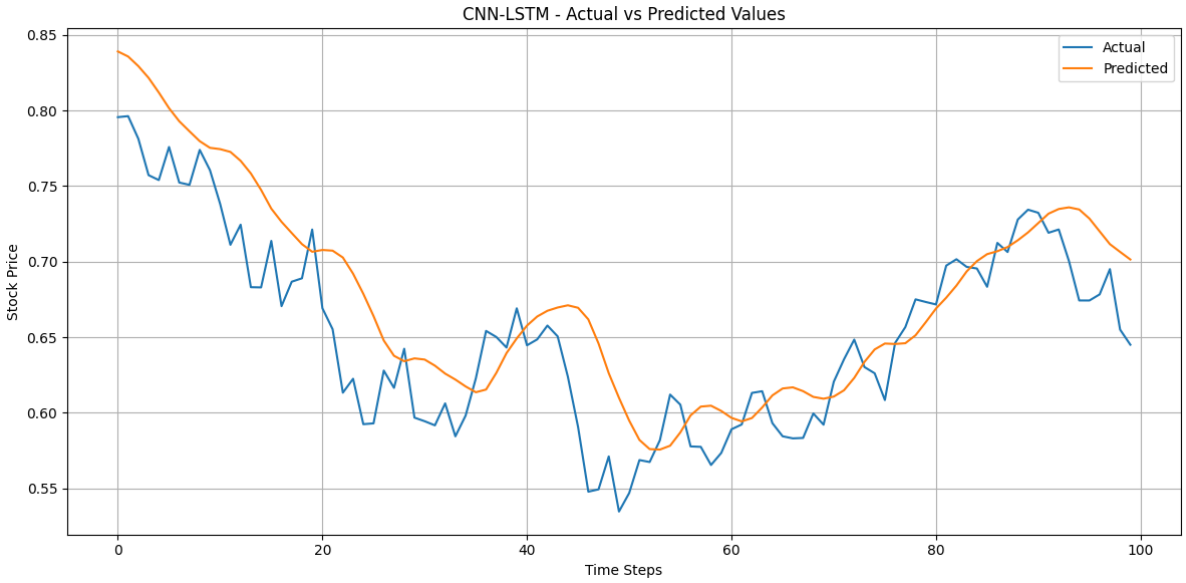


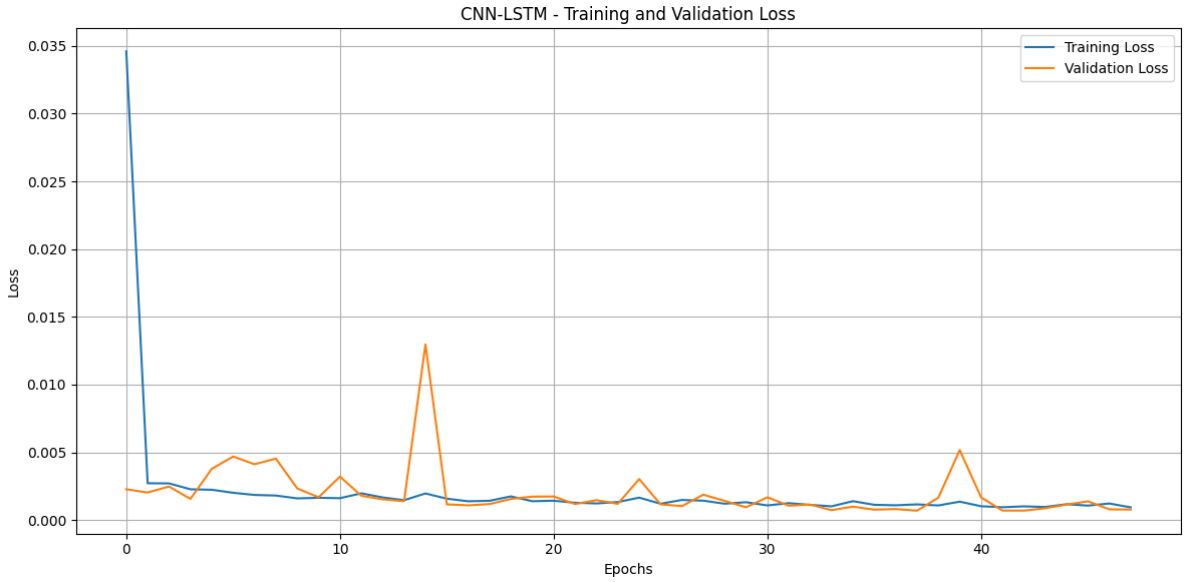
Performance of Each Model: GRU





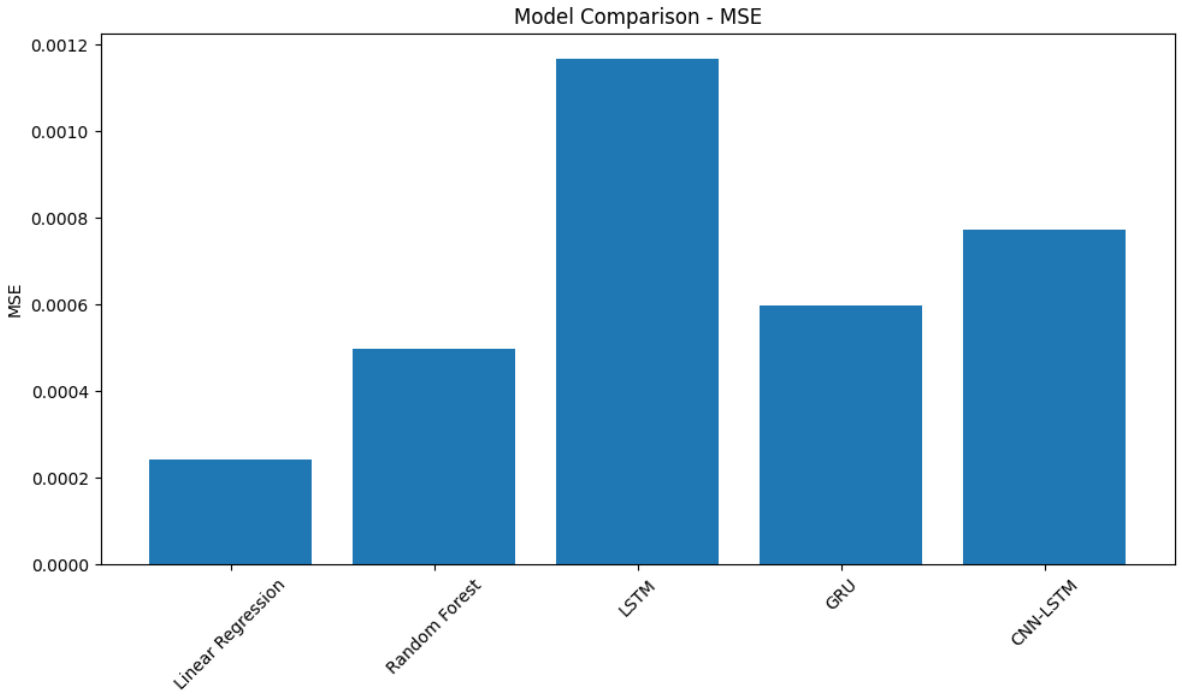
Performance of Each Model: CNN-LSTM



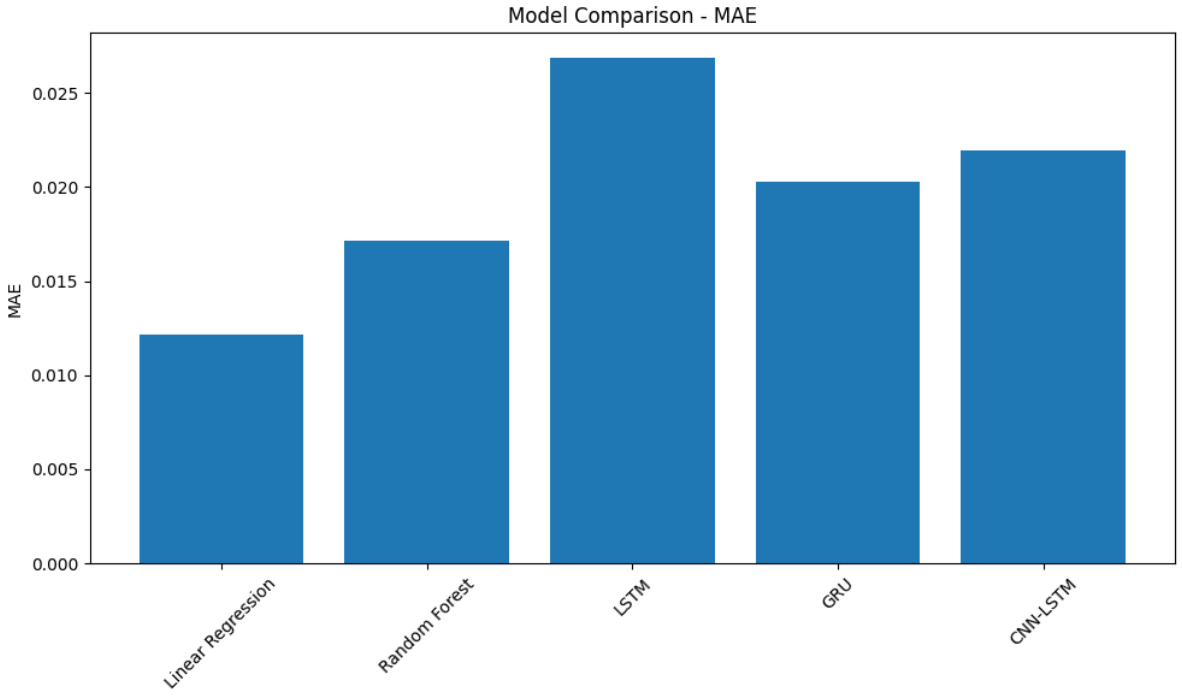


Comparison of Each Model concerning the Evaluation Metrics

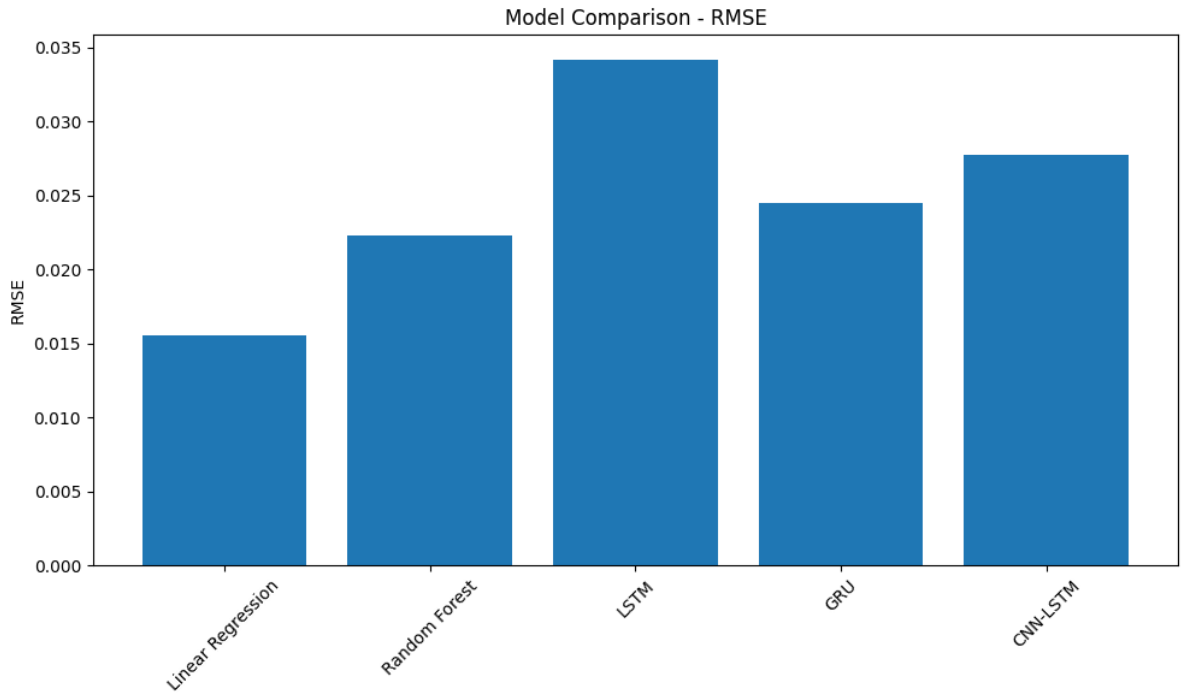
With Mean Squared Error (MSE):



With Mean Absolute Error (MAE):



With Root Mean Squared Error (RMSE):



All Model Comparison Table

Unnamed: 0	MSE	RMSE	MAE
Linear Regression	0.0002428659709877	0.0155841576925989	0.01214456752884
Random Forest	0.0004968285780142	0.0222896518145593	0.0171340567708203
LSTM	0.0011662799054515	0.0341508404794315	0.0268408842609681
GRU	0.0005989674622354	0.0244738117635053	0.020258192745013
CNN-LSTM	0.0007720830389098	0.0277863822565991	0.0219677130821005

Interpretation

- The baseline Linear Regression outperformed deep learning models.

- Deep learning models underperformed due to their sensitivity to data quantity, hyperparameter settings, and temporal feature structure.

When comparing deep learning models with baseline models, the GRU can be identified as the best-performing model.

It has been saved in the “models” folder as 'trained_model.h5'.

Model Optimization

Techniques Applied

- Early stopping to prevent overfitting
- Learning rate tuning and batch size adjustments
- Adjusted sequence length and number of epochs

Observations

- Early stopping was effective in halting training once validation loss increased.
- Performance of DL models improved slightly post-optimization but still did not outperform the baseline.

Challenges and Solutions

Key Challenges

- Model Overfitting
- Poor DL Performance: Due to possibly inadequate sequence length or insufficient temporal signal.
- Baseline Outperformance: Indicates stock prices might behave more linearly over short windows.

Solutions Implemented

- Resolved with dropout and early stopping.
- Feature scaling including the target helped stabilize DL training.
- Focused optimization of DL models, but ultimately chose the best DL model relative to the baselines

Conclusion

The best-performing model overall was the Linear Regression, which served as a baseline reference. Although deep learning models like GRU and CNN-LSTM came close, they could not surpass the accuracy of the baseline. This aligns with the project's goal to use baseline models as a benchmark and determine whether DL models provide added value. Future improvements could involve integrating external data (e.g., news sentiment, macroeconomic indicators) or experimenting with transformer architectures.

References

[1] scikit-learn.org. (n.d.). *Documentation scikit-learn: machine learning in Python — scikit-learn 0.21.3 documentation*. [online] Available at: https://scikit-learn.org/0.21/documentation.html.

[2] PyTorch (2019). *PyTorch documentation — PyTorch master documentation*. [online] Pytorch.org. Available at: https://pytorch.org/docs/stable/index.html.

[3] Potters, C. (2023). *Top 7 Technical Analysis Tools*. [online] Investopedia. Available at: https://www.investopedia.com/top-7-technical-analysis-tools-4773275.

[4] Twiggs, C. (2024). *Technical Indicators A ~ Z*. [online] Incrediblecharts.com. Available at: https://www.incrediblecharts.com/indicators/technical-indicators.php.

[5] Jason (2024). *Implementing Technical Indicators in Python for Trading*. [online] IBKR Campus US. Available at: https://www.interactivebrokers.com/campus/ibkr-quant-news/implementing-technical-indicators-in-python-for-trading/ [Accessed 6 Apr. 2025].

[6] SR (2024). *Implementing Technical Indicators in Python: A Practical Approach*. [online] Medium. Available at: https://medium.com/@deepml1818/implementing-technical-indicators-in-python-a-practical-approach-3d8e434efd2f [Accessed 6 Apr. 2025].

[7] riteshsinha (2022). *Useful features in Predicting Stock Prices*. [online] Kaggle.com. Available at: https://www.kaggle.com/code/riteshsinha/useful-features-in-predicting-stock-prices [Accessed 6 Apr. 2025].

[8] @ConsensusNLP. (2025). *What are the most effective feature engineering techniques for stock market prediction using machine learning? - Consensus*. [online] Available at: https://consensus.app/results/?q=What%20are%20the%20most%20effective%20feature%20engineering%20techniques%20for%20stock%20market%20prediction%20using%20machine%20learning%3F [Accessed 6 Apr. 2025].