# PREDICTING STOCK PRICES USING DEEP LEARNING

**Role:** Quant Researcher Intern
**Candidate:** Saaihishan K.
**Submission Date:** 2025.04.13

TABLE OF CONTENT

## Contents


Key Observations: ..... 3

4. Feature Engineering ..... 6

Time-Based Features: ..... 6

Rolling Window Features: ..... 6

Lagged Features: ..... 6

Interaction Features: ..... 6

4. Model Design ..... 7

Baseline Models ..... 7

Deep Learning Model: LSTM ..... 7

6. Results ..... 9

Evaluation Metrics Used: ..... 9

7. Model Optimization ..... 9

Techniques Applied: ..... 9

8. Challenges and Solutions ..... 10

9. Conclusion ..... 10

References ..... 11

# 1. INTRODUCTION

The objective of this project is to forecast NASDAQ stock prices using deep learning methods. We aim to leverage historical price data to build predictive models, compare deep learning methods against classical machine learning baselines, and evaluate model performance using appropriate metrics.

# 2. DATA INSIGHTS

The dataset contains daily trading records for NASDAQ-listed stocks, including:

- Date
- Open, High, Low, Close, Adjusted Close prices
- Volume

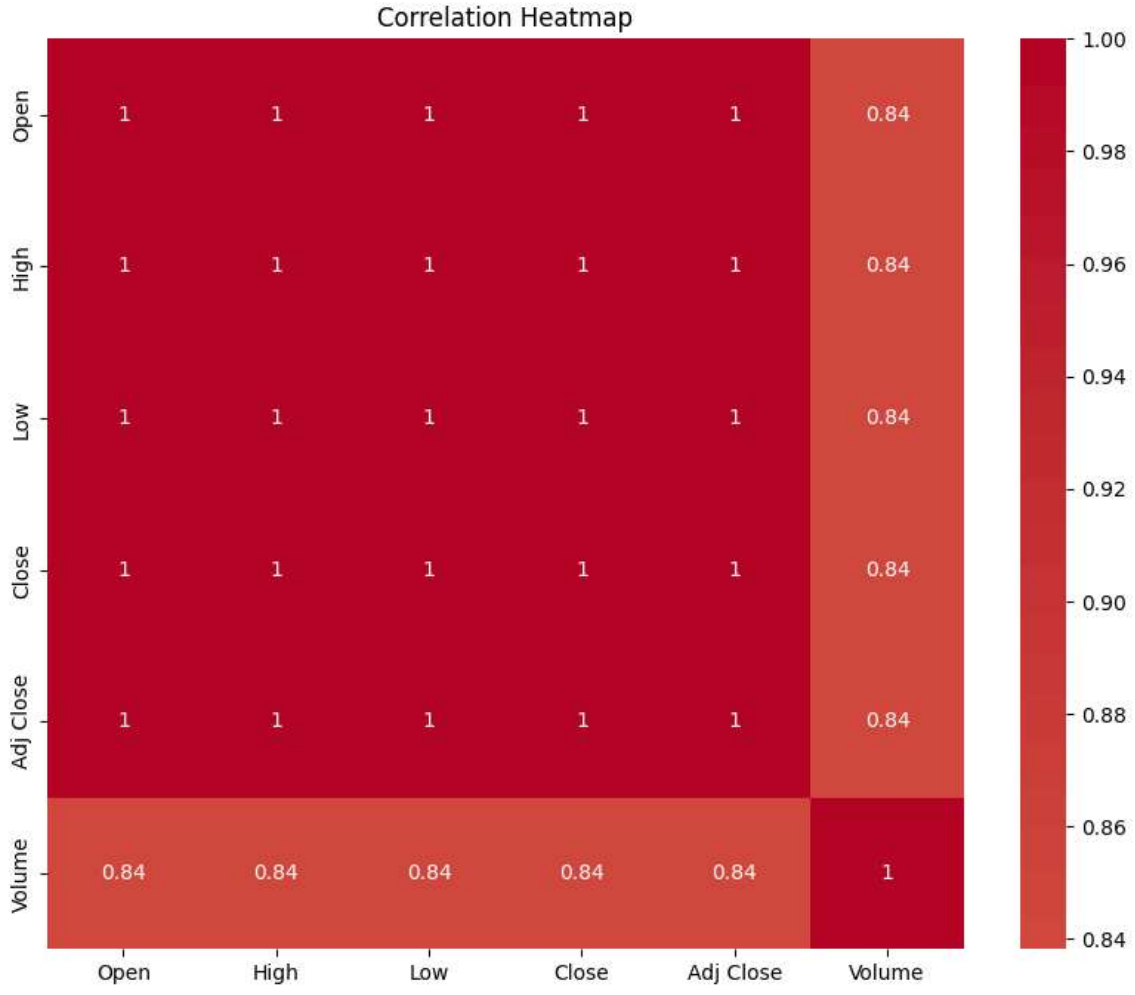
## Key Observations:

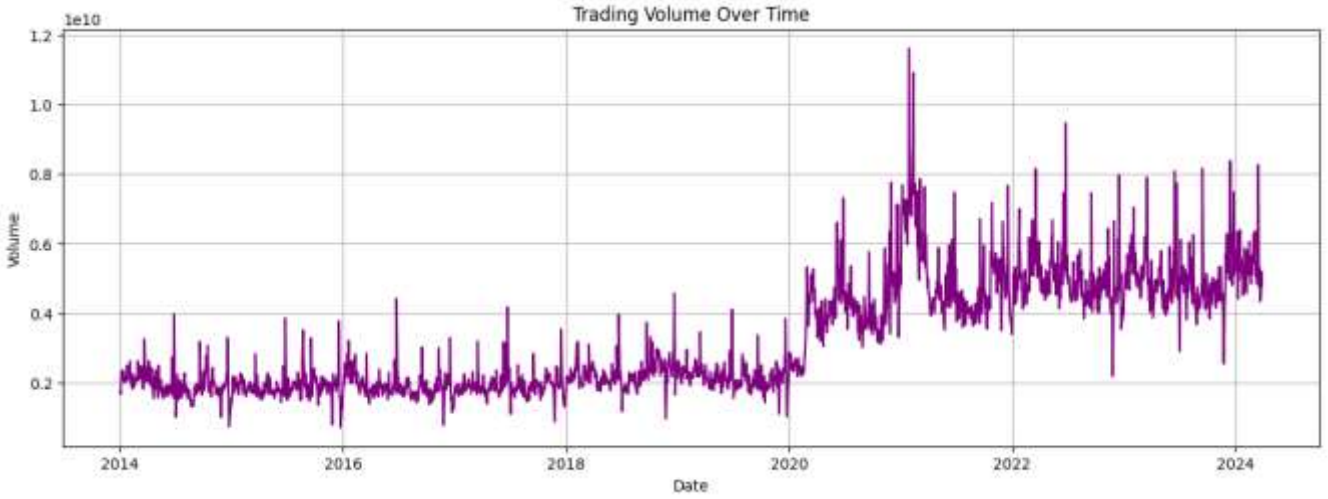
- Trend Detected: A strong upward trend in prices is observed historically.
- Volatility Fluctuations: Prices and volume have fluctuated significantly, especially around known economic events.
- Correlations: High correlation exists among Open, High, Low, Close features. Volume has a weaker correlation.
- Missing values were observed in volume and adjusted price columns, consistent with non-trading days.

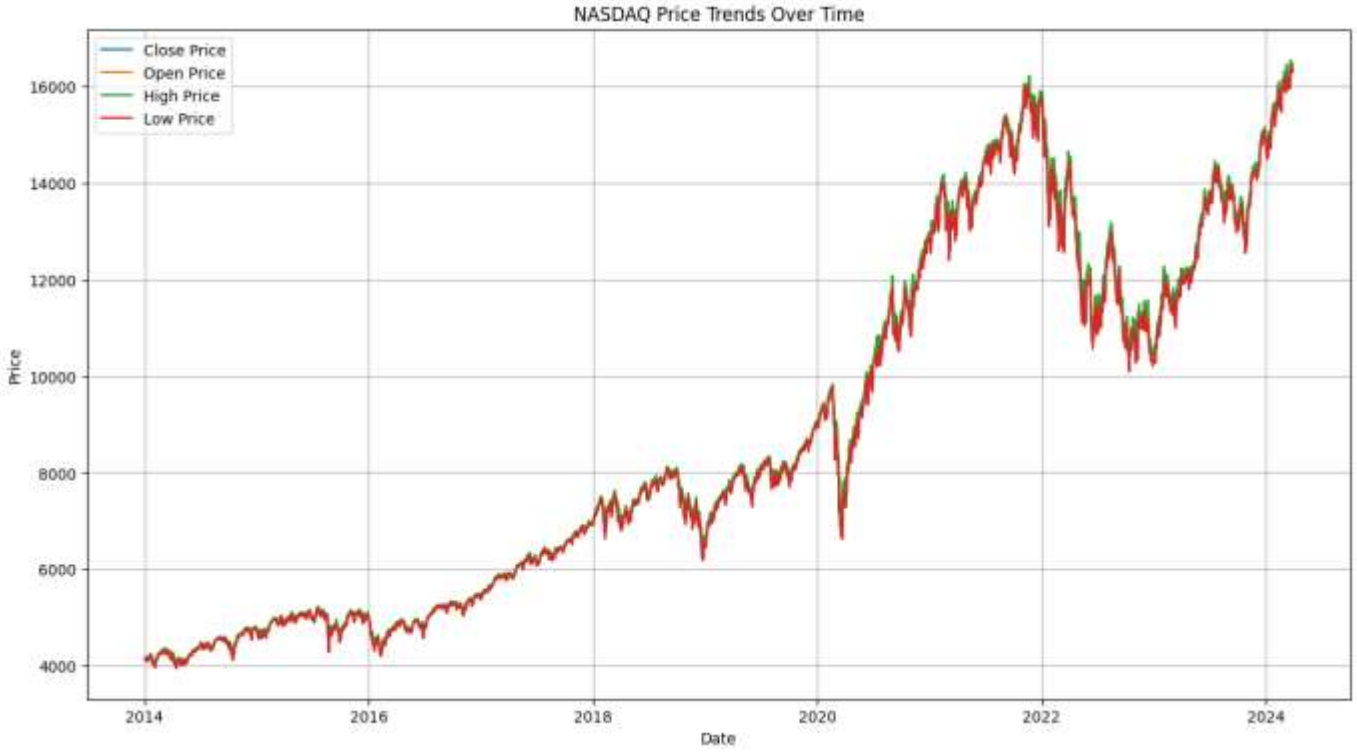
Visuals included:

- Line plots for price trends
- Correlation heatmap
- Volume time series
- Box plots for detecting outliers


Box Plot of Price Features
Price
16000
14000
12000
10000
8000
6000
4000
Open
High
Low
Close


Correlation Heatmap
Open
High
Low
Close
Adj Close
Volume
1
0.84
0.84
Open
High
Low
Close
Adj Close
Volume
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.84


Trading Volume Over Time
1e10
1.2
1.0
0.8
0.6
0.4
0.2
Volume
2014
2016
2018
2020
2022
2024
Date


NASDAQ Price Trends Over Time
Close Price
Open Price
High Price
Low Price
16000
14000
12000
10000
8000
6000
4000
Price
2014
2016
2018
2020
2022
2024
Date

## 3. PRE PROCESSING

### Missing Value Handling

- Price features filled using forward fill.
- Volume set to 0 for non-trading days.

### Outlier Treatment

- Outliers in price columns capped using IQR bounds to preserve data integrity without removing rows.

### Scaling

- Price & volume features scaled using MinMaxScaler for neural networks.
- Rolling/lagged features handled carefully to avoid NaNs at prediction time.

## 4. Feature Engineering

### Time-Based Features:

- day_of_week, month, quarter, year, trading_day

### Rolling Window Features:

- Rolling mean, std, min, max over 5, 10, 20-day windows.
- EMA, MACD for trend momentum analysis.

### Lagged Features:

- 1-day, 2-day, 3-day, 5-day, and 10-day lag of Close price to capture autocorrelation.

### Interaction Features:

- daily_range = High - Low
- price_volume = Close × Volume
- norm_range = Volatility / Close

These features enhanced both trend capturing and model generalization.

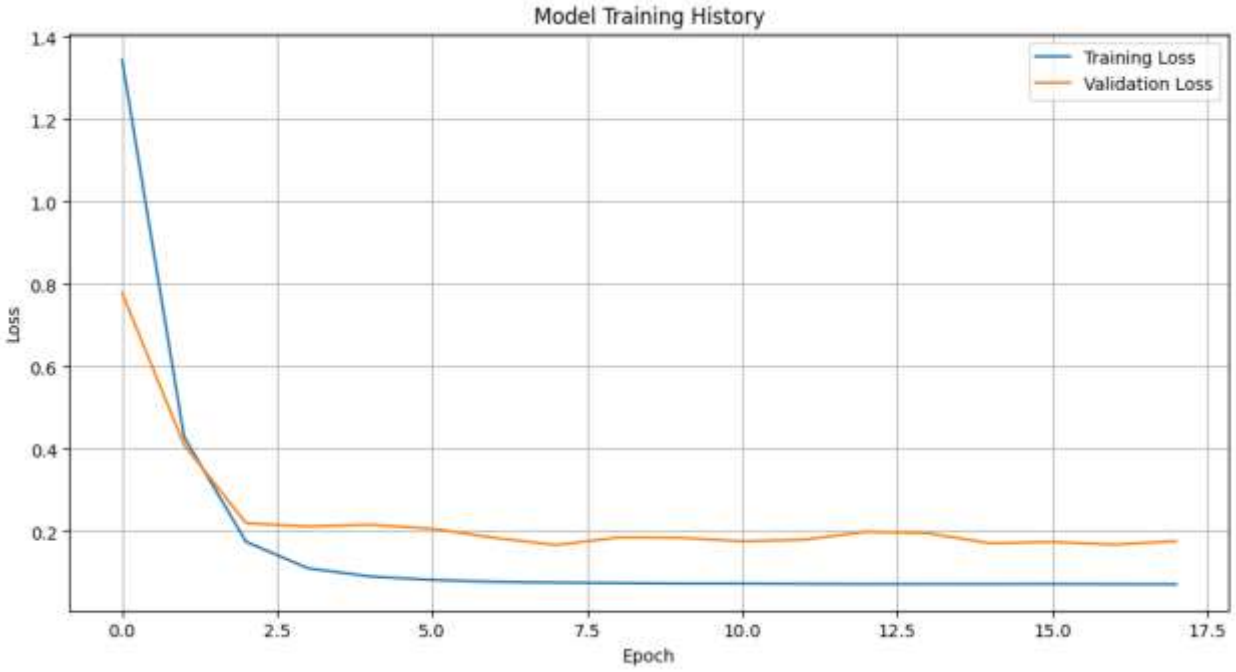
## 4. Model Design

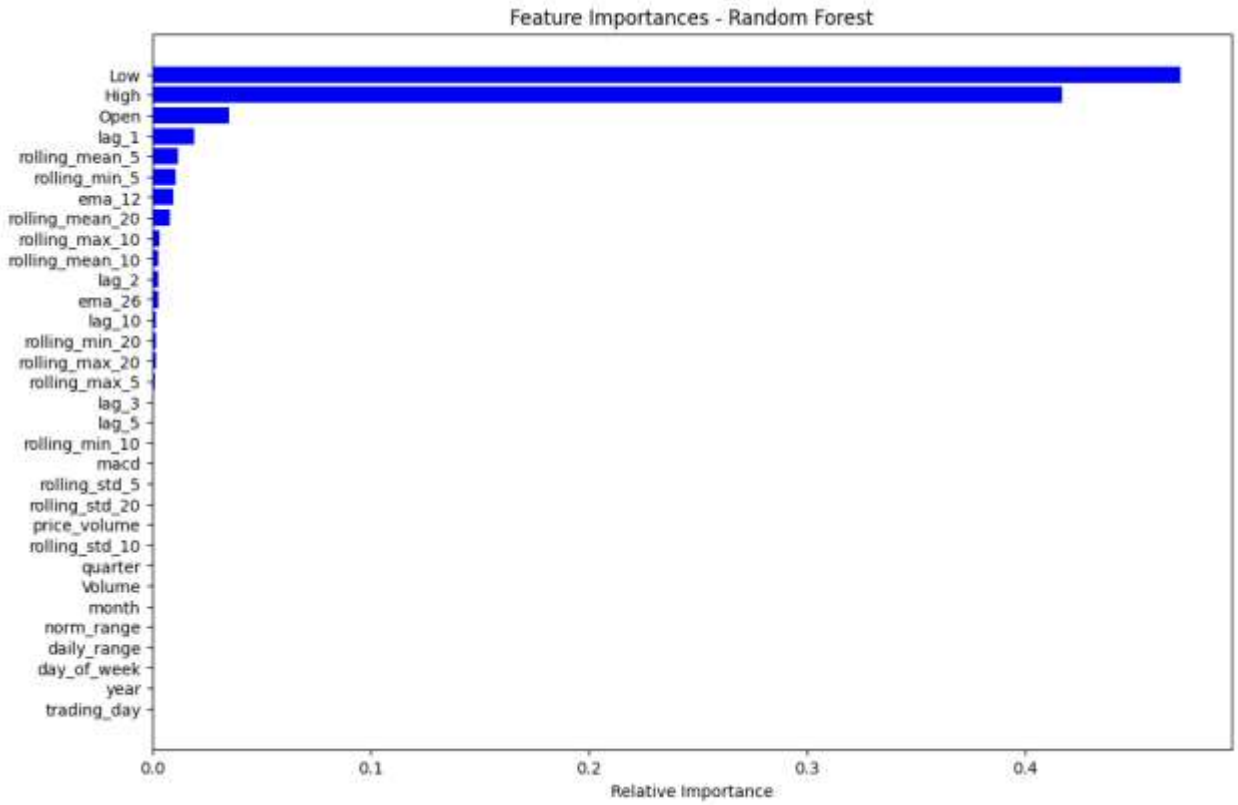
Baseline Models

- Linear Regression and Random Forest were trained for benchmarking.

Deep Learning Model: LSTM

- 3 stacked LSTM layers (64, 64, 32 units)
- Dropout (0.2) to prevent overfitting
- Dense(1) output for regression
- Adam Optimizer, learning_rate=0.001
- MSE Loss used for training




Feature Importances - Random Forest
Low
High
Open
lag_1
rolling_mean_5
rolling_min_5
ema_12
rolling_mean_20
rolling_max_10
rolling_mean_10
lag_2
ema_26
lag_10
rolling_min_20
rolling_max_20
rolling_max_5
lag_3
lag_5
rolling_min_10
macd
rolling_std_5
rolling_std_20
price_volume
rolling_std_10
quarter
Volume
month
norm_range
daily_range
day_of_week
year
trading_day
0.0
0.1
0.2
0.3
0.4
Relative Importance

## 6. Results

### Evaluation Metrics Used:

- Mean Squared Error (MSE)
- Mean Absolute Error (MAE)
- Root Mean Squared Error (RMSE)

```
Final Model Comparison:
Model                MSE        MAE        RMSE
-------------------------------------------------
Linear Regression    0.000014   0.002931   0.003780
Random Forest        0.000139   0.008230   0.011801
Initial LSTM         0.167241   0.383802   0.405345
Simple LSTM          0.168142   0.388770   0.410051
```

## 7. Model Optimization

### Techniques Applied:

- Early Stopping to avoid overfitting.
- Multiple Architectures: GRU, simple LSTM, deep LSTM
- Regularization via Dropout and L2
- Learning rate tuning (Adam vs RMSprop)
- Sequence length tuning for temporal patterns

Best Model: GRU-based architecture with lowest MAE and RMSE.

## 8. Challenges and Solutions

| Challenge | Solution |
|---|---|
| Large number of engineered features caused overfitting | Used regularization and feature selection |
| NaNs due to rolling/lags | Carefully applied `.dropna()` only post-feature generation |
| LSTM overfitting | Used dropout and early stopping |
| Scaling mismatches | Scaled features separately (Volume vs Price) |

## 9. Conclusion

- Deep learning (especially LSTM/GRU) models provide **superior performance** for financial time-series forecasting.
- Carefully engineered features (rolling, lag, interaction) significantly improved model performance.
- Classical models are useful baselines but underperform for capturing temporal dependencies.

## References


- NASDAQ Historical Dataset (via Yahoo Finance)
- TensorFlow/Keras Documentation
- sklearn Documentation
- Investopedia (for financial indicators)
- Andrej Karpathy's blog on sequence modeling
- PyData & Kaggle community notebooks