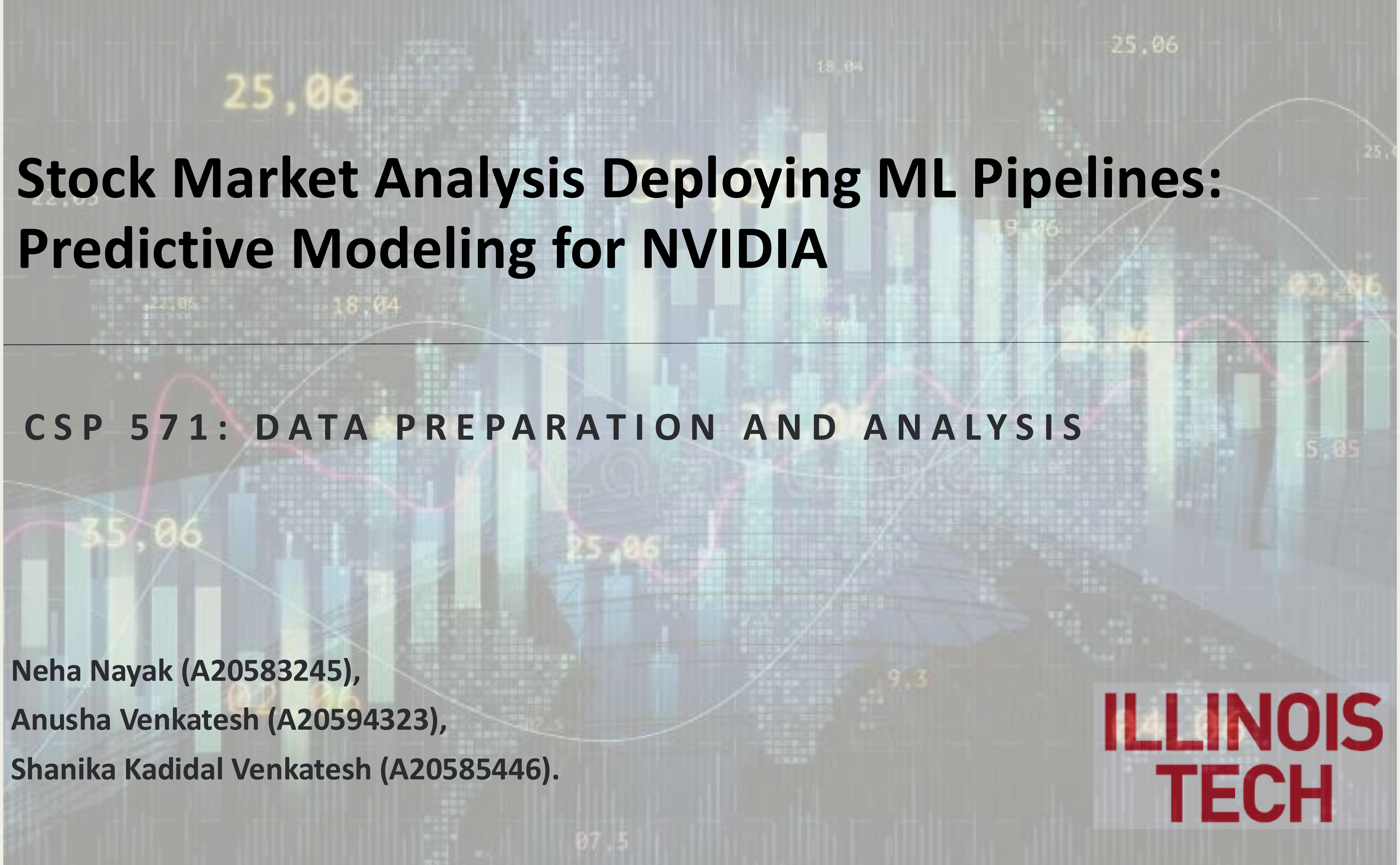
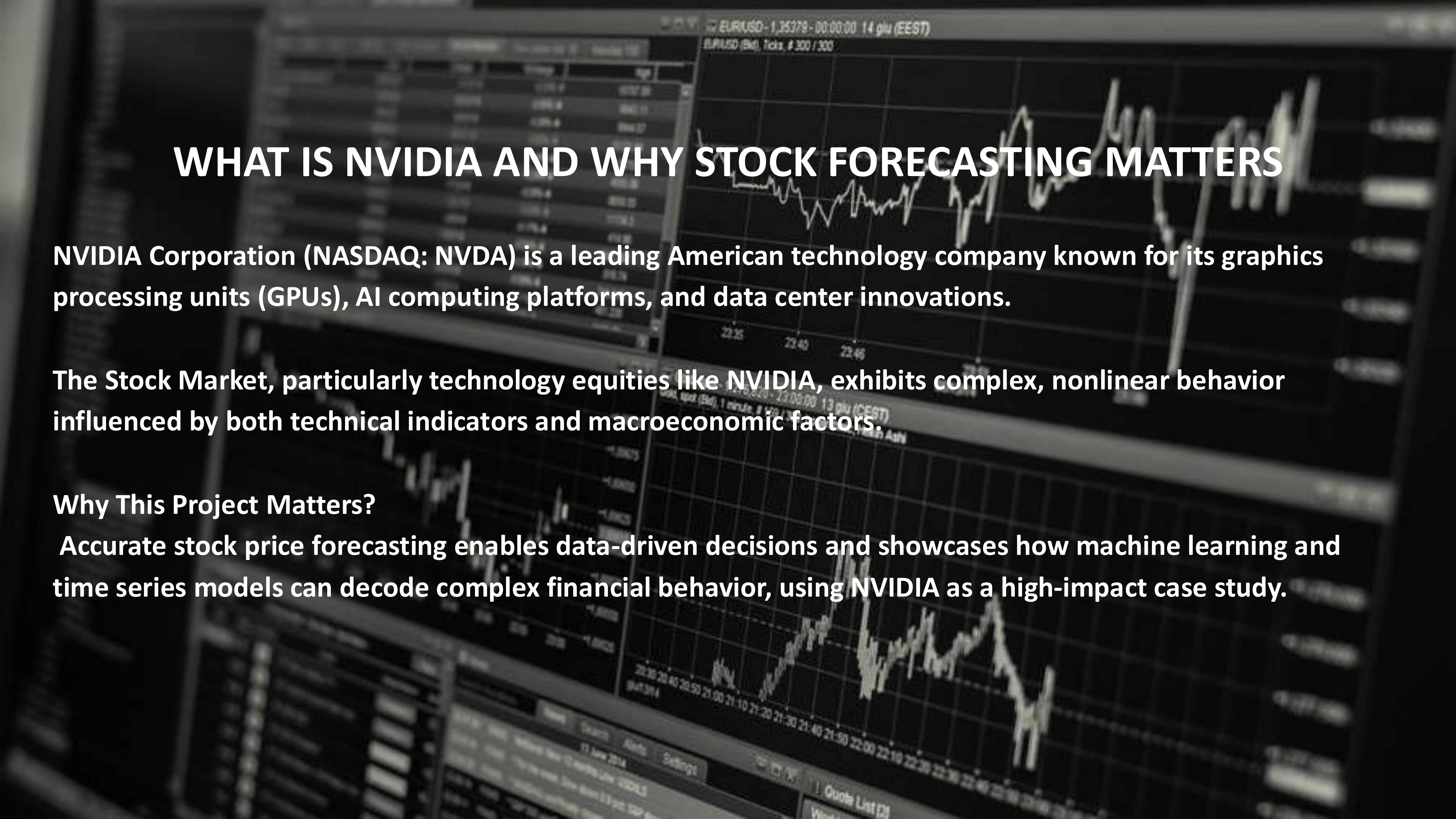


Stock Market Analysis Deploying ML Pipelines: Predictive Modeling for NVIDIA

CSP 571: DATA PREPARATION AND ANALYSIS

Neha Nayak (A20583245),
Anusha Venkatesh (A20594323),
Shanika Kadidal Venkatesh (A20585446).

ILLINOIS TECH



WHAT IS NVIDIA AND WHY STOCK FORECASTING MATTERS

NVIDIA Corporation (NASDAQ: NVDA) is a leading American technology company known for its graphics processing units (GPUs), AI computing platforms, and data center innovations.

The Stock Market, particularly technology equities like NVIDIA, exhibits complex, nonlinear behavior influenced by both technical indicators and macroeconomic factors.

Why This Project Matters?
Accurate stock price forecasting enables data-driven decisions and showcases how machine learning and time series models can decode complex financial behavior, using NVIDIA as a high-impact case study.

AGENDA

- Overview
- Research Motivation
- Problem Statement
- Literature Review
- Methodology
- Team Members and Responsibilties.
- Data Source & Preprocessing
- Feature Engineering
- Exploratory Data Analysis
- Outlier Detection & Scaling
- Modeling Overview
- LSTM Architecture
- ARIMA Modeling
- RMSE Comparison
- Forecasting Results
- Validation & Bias
- Conclusion
- Future Work

OVERVIEW

Objective:

To build an end-to-end machine learning and time series forecasting pipeline to model and predict NVIDIA stock closing prices.

Key Insights:

- Linear Regression delivered optimal performance on tabular features.
- ARIMA was effective in univariate time-series forecasting.
- LSTM faced challenges due to long-range volatility and data sparsity.

Future Work:

- Explore multivariate time-series models such as VAR and LSTM with external factors.
- Fine-tune LSTM architecture with longer sequences and dropouts.

RESEARCH MOTIVATION

Predicting stock prices remains a high-stakes problem in financial analytics.

The NVIDIA stock, due to its volatility and trend shifts, presents a real-world challenge.

Goal: compare classical ML vs deep learning vs time-series forecasting in a uniform pipeline.

PROBLEM STATEMENT

Question 1

Can we reliably forecast short-term closing prices using engineered technical indicators?

Question 2

Which model architecture best balances generalization, accuracy, and interpretability?

Question 3

How does seasonality and technical trend impact model performance?

LITERATURE REVIEW

Hyndman & Athanasopoulos (2018): Demonstrated that ARIMA is effective for forecasting stationary and seasonally adjusted time series in financial domains.

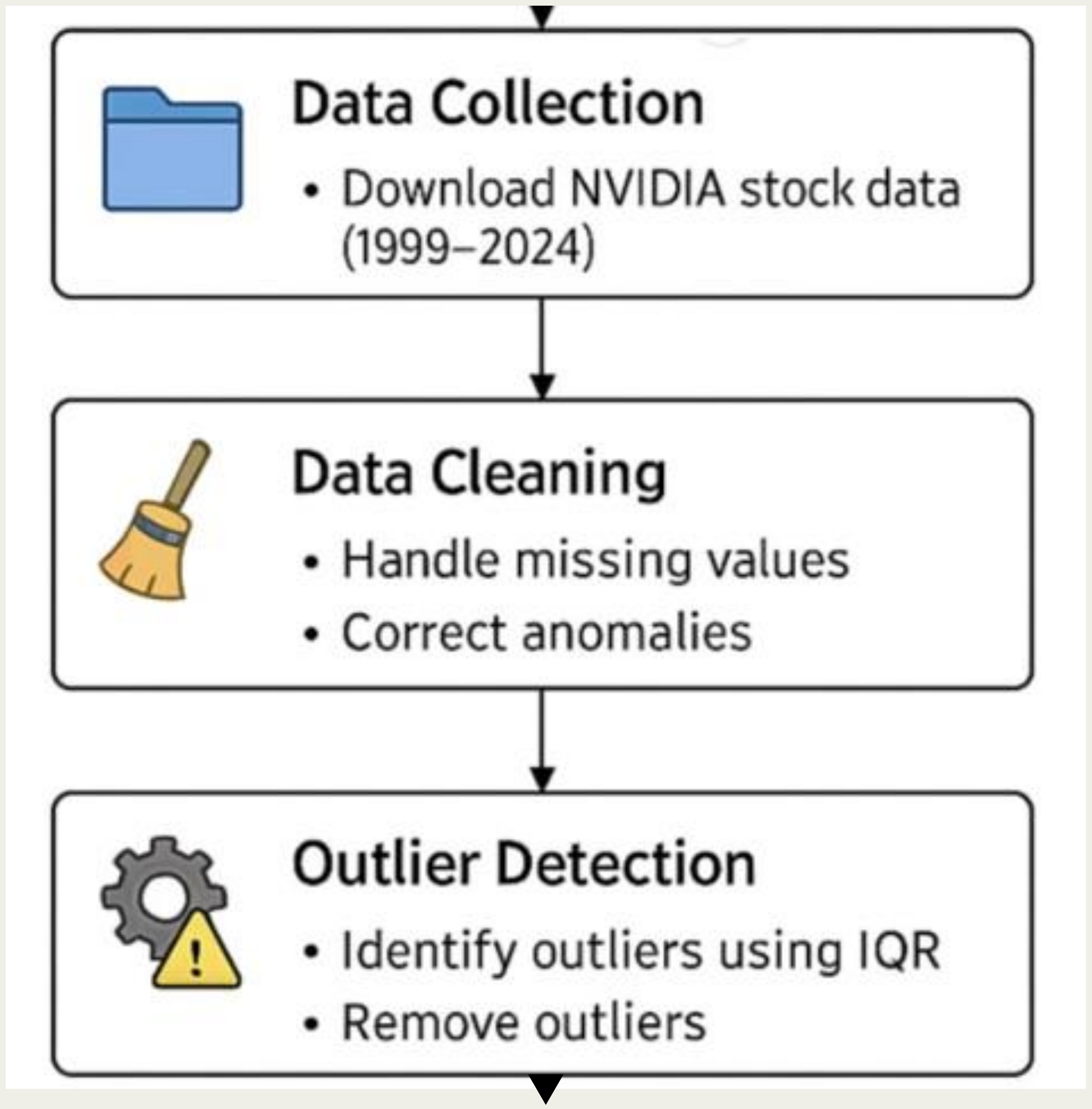
Chlebus et al. (2019): Showed that LSTM networks outperform traditional models when capturing long-term dependencies in stock price patterns.

Rozeff & Kinney (1976): Provided early evidence of seasonality in stock returns, emphasizing recurring monthly trends across years.

Mukherjee & Naka (1995): Established the significance of macroeconomic variables in influencing stock market movements using VECM analysis.

Humpe & Macmillan (2009): Found strong long-term co-integration between macroeconomic indicators and stock prices in developed economies.

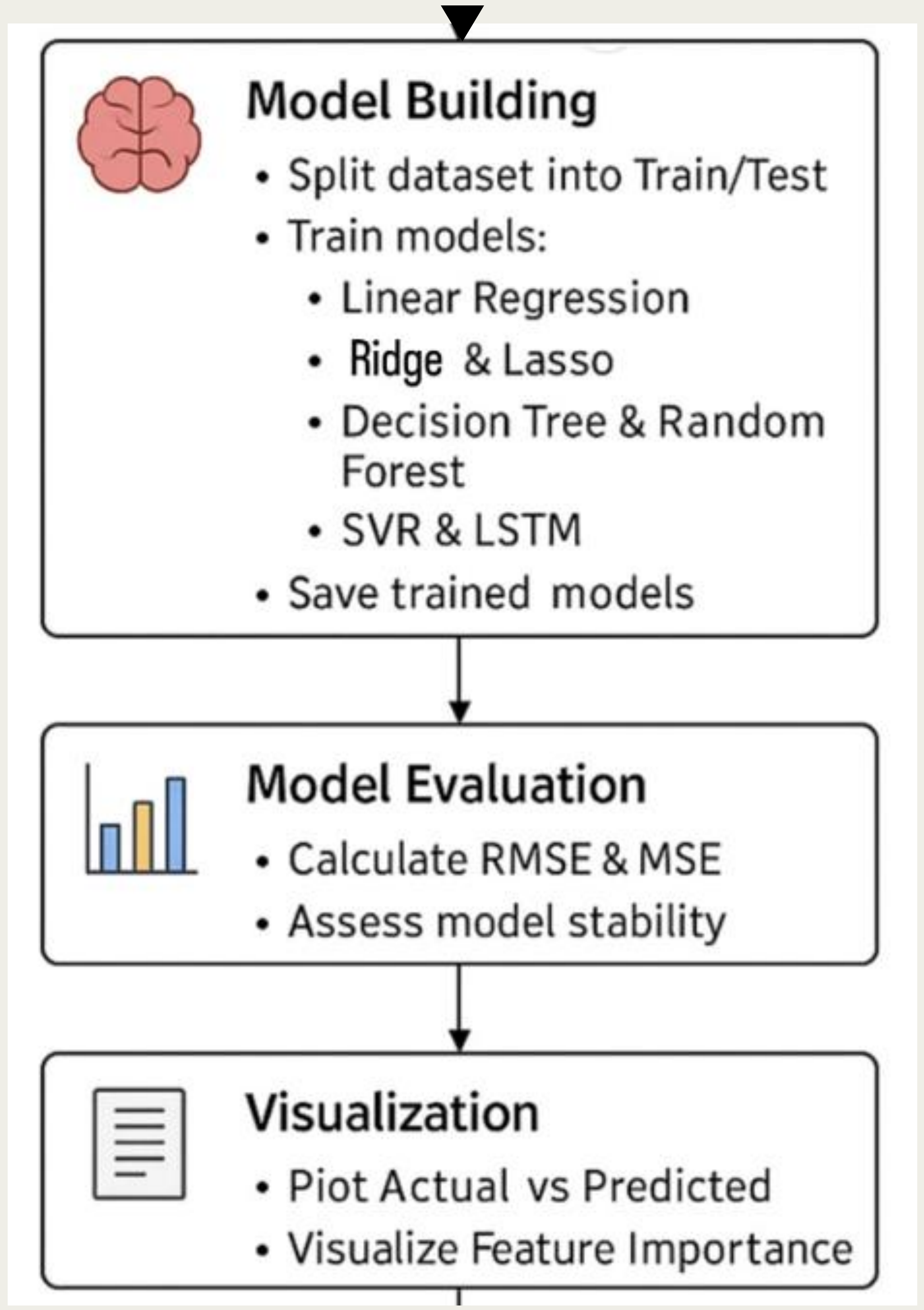
METHODOLOGY



Data Collection
- Download NVIDIA stock data (1999–2024)

Data Cleaning
- Handle missing values
- Correct anomalies

Outlier Detection
- Identify outliers using IQR
- Remove outliers

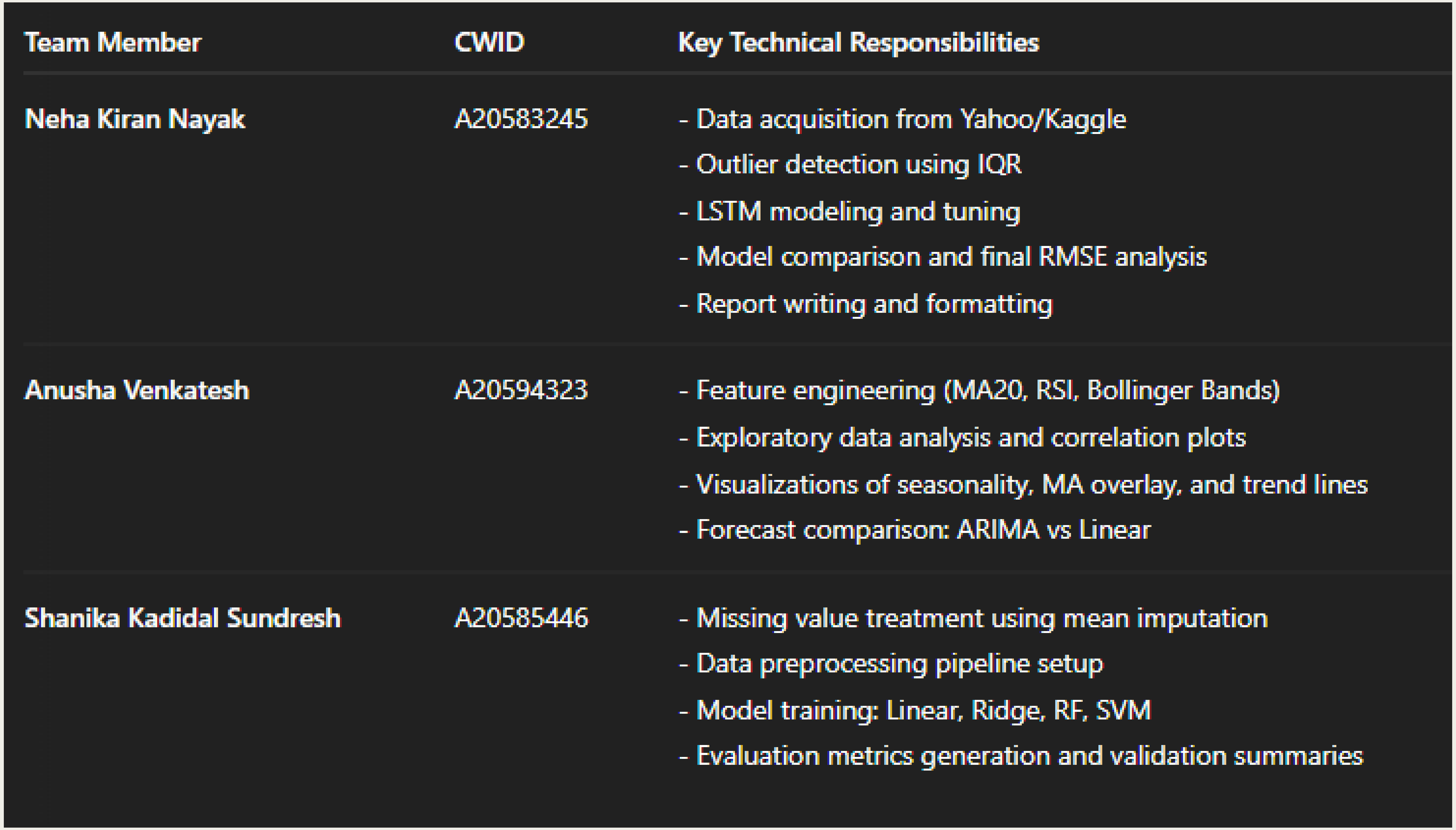


Model Building
- Split dataset into Train/Test
- Train models:
  - Linear Regression
  - Ridge & Lasso
  - Decision Tree & Random Forest
  - SVR & LSTM
- Save trained models

Model Evaluation
- Calculate RMSE & MSE
- Assess model stability

Visualization
- Piot Actual vs Predicted
- Visualize Feature Importance

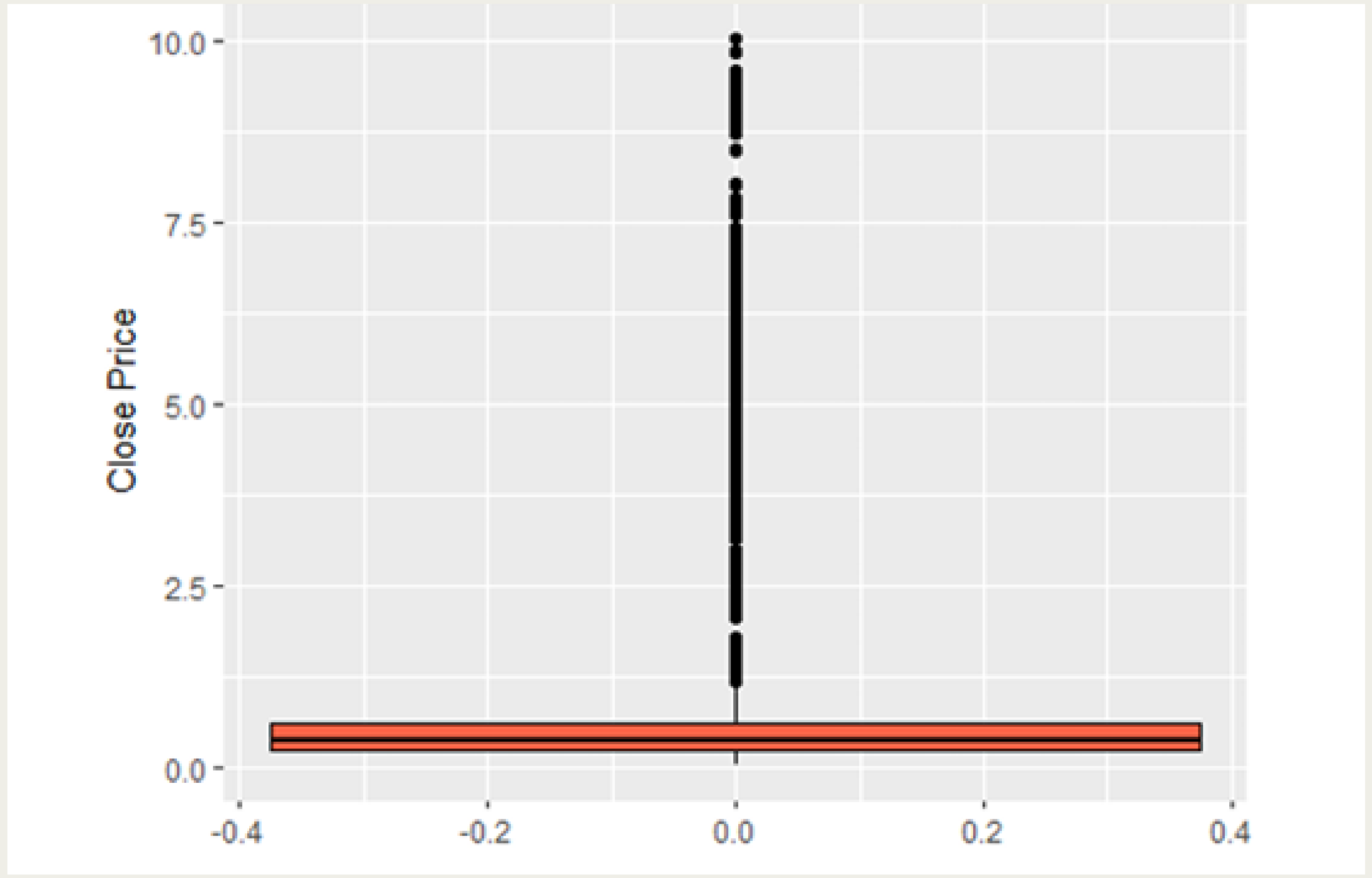
TEAM MEMBERS AND RESPONSIBILITIES

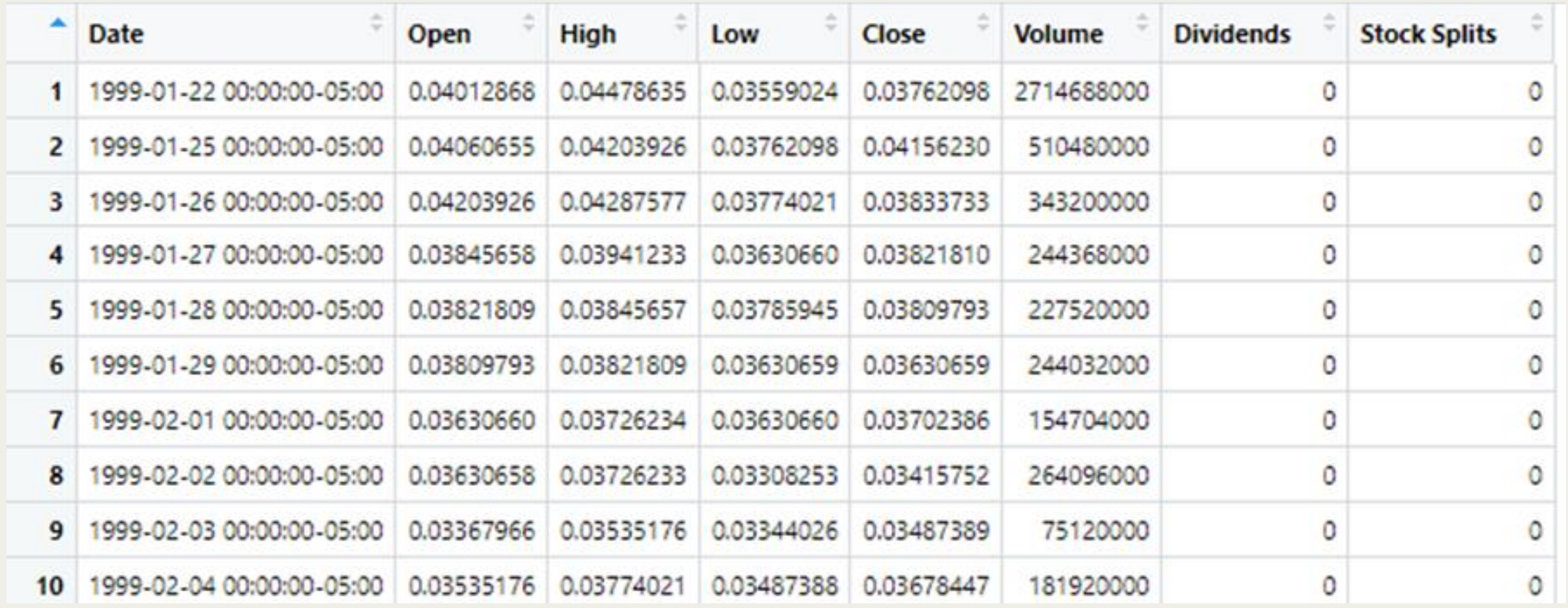


Team Member	CWID	Key Technical Responsibilities
Neha Kiran Nayak	A20583245	- Data acquisition from Yahoo/Kaggle - Outlier detection using IQR - LSTM modeling and tuning - Model comparison and final RMSE analysis - Report writing and formatting
Anusha Venkatesh	A20594323	- Feature engineering (MA20, RSI, Bollinger Bands) - Exploratory data analysis and correlation plots - Visualizations of seasonality, MA overlay, and trend lines - Forecast comparison: ARIMA vs Linear
Shanika Kadidal Sundresh	A20585446	- Missing value treatment using mean imputation - Data preprocessing pipeline setup - Model training: Linear, Ridge, RF, SVM - Evaluation metrics generation and validation summaries

DATA SOURCE & PREPROCESSING

- Data Source: 6442 historical stock entries for NVIDIA (NVDA) from 1999–2024, sourced from Kaggle: https://www.kaggle.com/datasets/prajwaldongre/nvidia-corp-share-price-2000-2024
- Variables included: Open, High, Low, Close, Volume, Dividends, and Stock Splits.





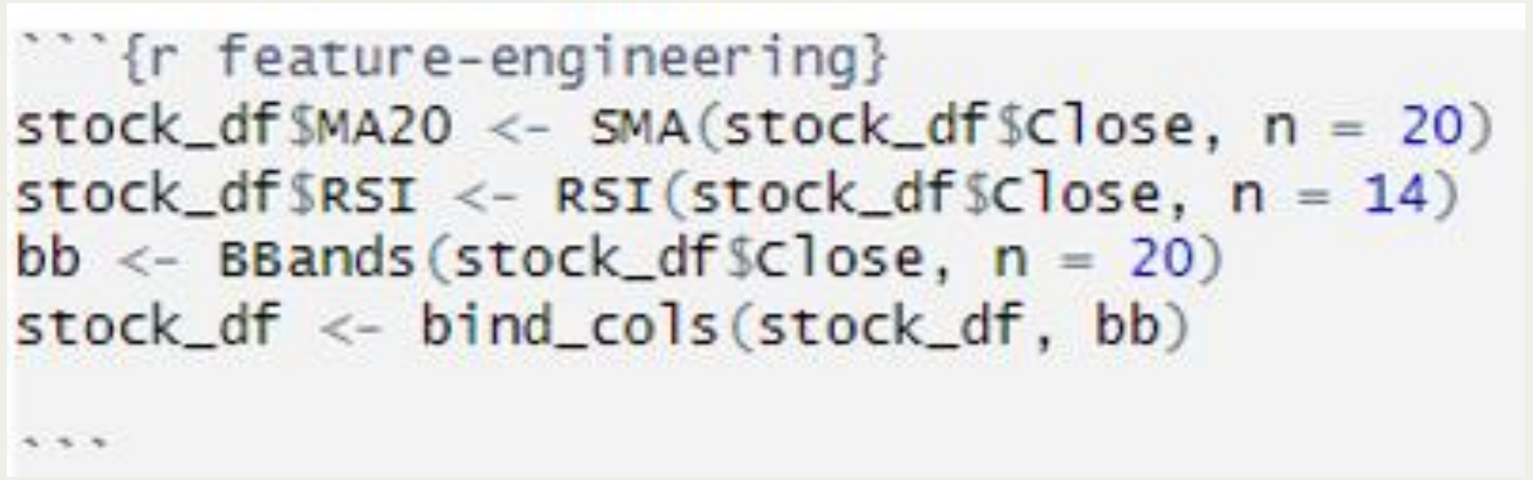
	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
1	1999-01-22 00:00:00-05:00	0.04012868	0.04478635	0.03559024	0.03762098	2714688000	0	0
2	1999-01-25 00:00:00-05:00	0.04060655	0.04203926	0.03762098	0.04156230	510480000	0	0
3	1999-01-26 00:00:00-05:00	0.04203926	0.04287577	0.03774021	0.03833733	343200000	0	0
4	1999-01-27 00:00:00-05:00	0.03845658	0.03941233	0.03630660	0.03821810	244368000	0	0
5	1999-01-28 00:00:00-05:00	0.03821809	0.03845657	0.03785945	0.03809793	227520000	0	0
6	1999-01-29 00:00:00-05:00	0.03809793	0.03821809	0.03630659	0.03630659	244032000	0	0
7	1999-02-01 00:00:00-05:00	0.03630660	0.03726234	0.03630660	0.03702386	154704000	0	0
8	1999-02-02 00:00:00-05:00	0.03630658	0.03726233	0.03308253	0.03415752	264096000	0	0
9	1999-02-03 00:00:00-05:00	0.03367966	0.03535176	0.03344026	0.03487389	75120000	0	0
10	1999-02-04 00:00:00-05:00	0.03535176	0.03774021	0.03487388	0.03678447	181920000	0	0

- PREPROCESSING STEPS:

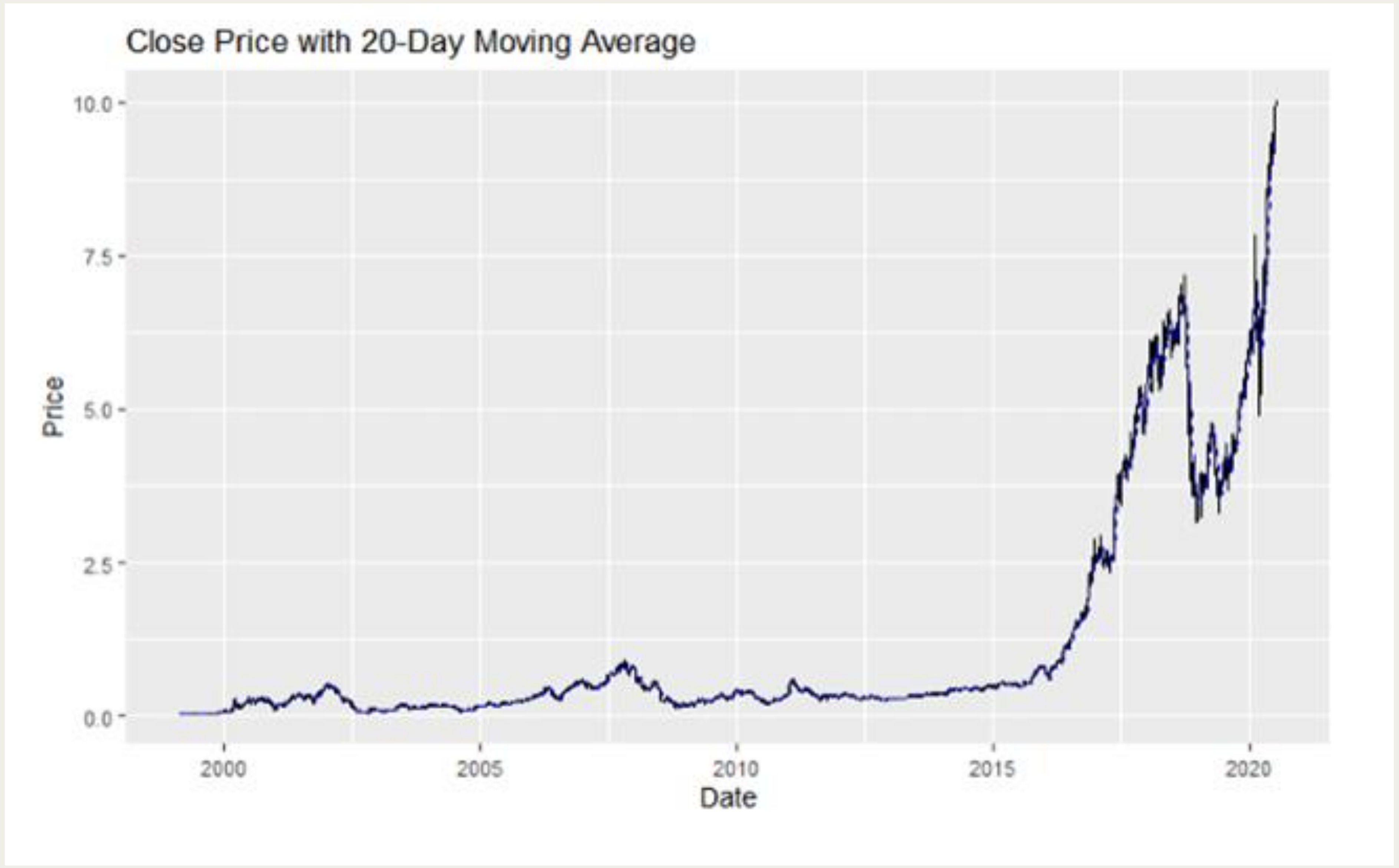
1. Verified absence of missing values.
2. Removed outliers using the Interquartile Range (IQR) method to ensure clean data for model training.

FEATURE ENGINEERING

- MA20 (20-day Moving Average): Helps identify overall trend by smoothing short-term price movements.
- RSI (14-day Relative Strength Index): Captures momentum and overbought/oversold zones.
- Bollinger Bands: Reflect volatility and potential reversal signals.



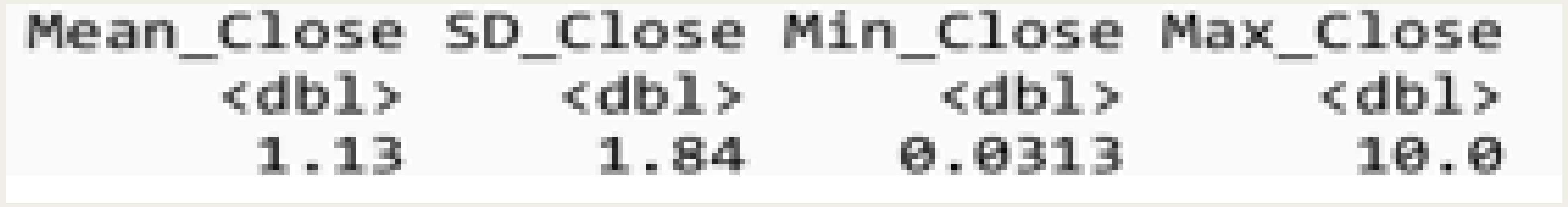
~~~
```{r feature-engineering}
stock_df$MA20 <- SMA(stock_df$Close, n = 20)
stock_df$RSI <- RSI(stock_df$Close, n = 14)
bb <- BBands(stock_df$Close, n = 20)
stock_df <- bind_cols(stock_df, bb)

```
~~~



EXPLORATORY DATA ANALYSIS

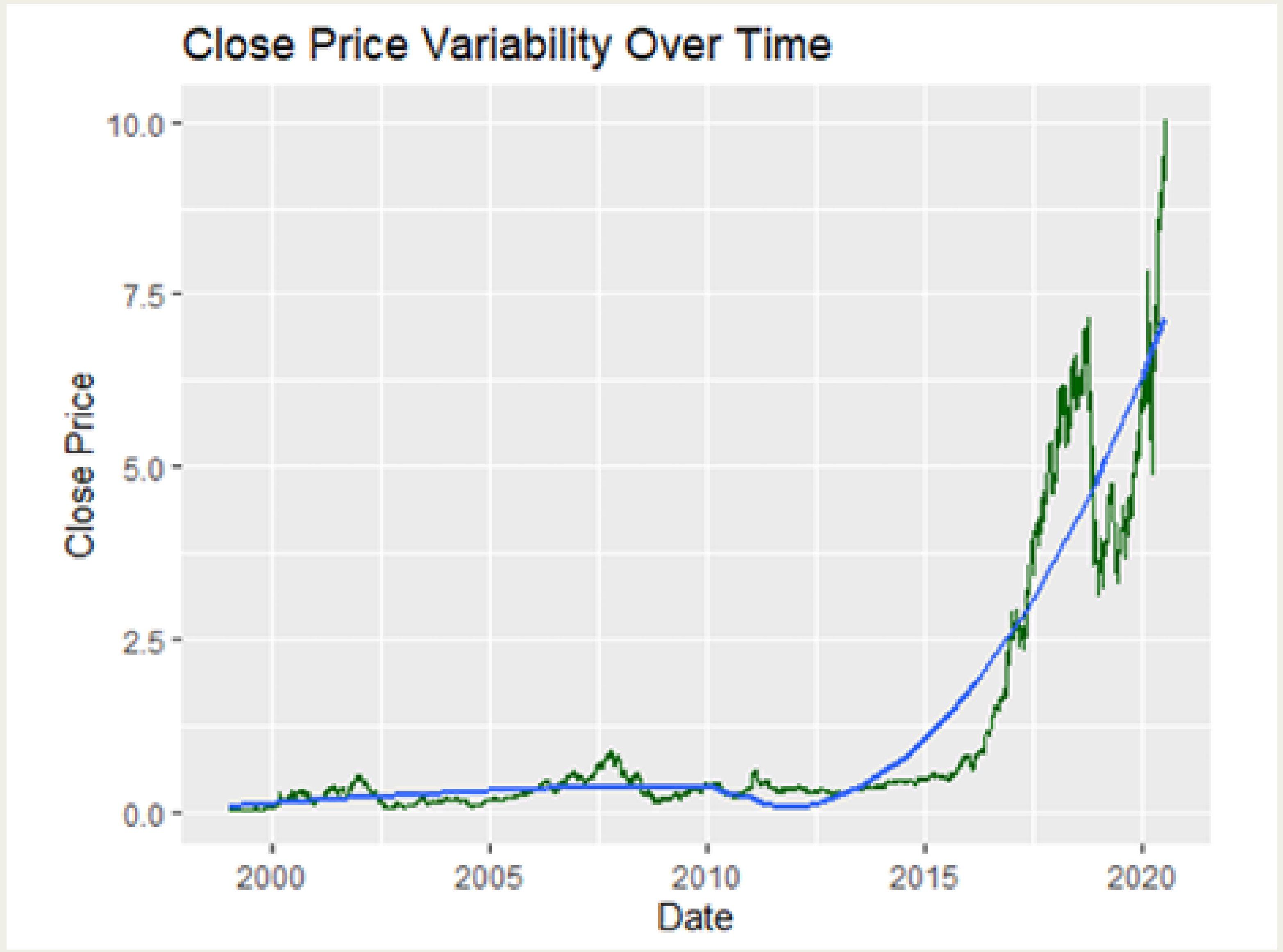
The dataset under analysis comprises 6,442 daily records of NVIDIA's stock prices over a 25-year period (1999–2024). A comprehensive statistical summary of the Close price reveals the following:

- Mean Close Price: $1.13
- Standard Deviation (SD): 1.84
- Range: $0.031 (min) – $10.02 (max)



```
Mean_Close SD_Close Min_Close Max_Close
     <dbl>    <dbl>     <dbl>     <dbl>
      1.13     1.84    0.0313      10.0
```

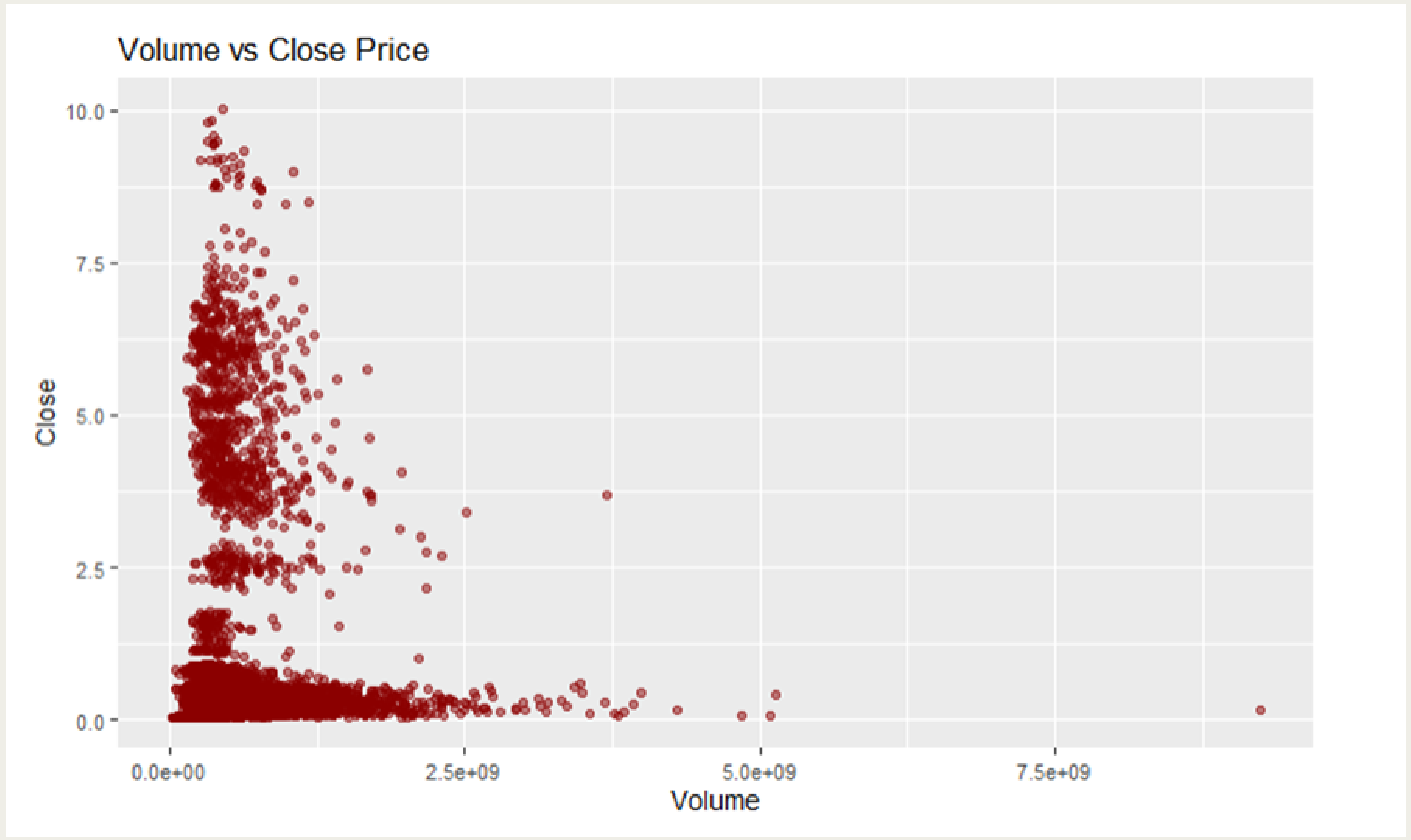
These values indicate substantial variability in stock prices, reflecting the company's growth trajectory and sensitivity to market events. The standard deviation of 1.84, being larger than the mean, suggests a wide spread of price values, with notable dispersion caused by periods of rapid appreciation.

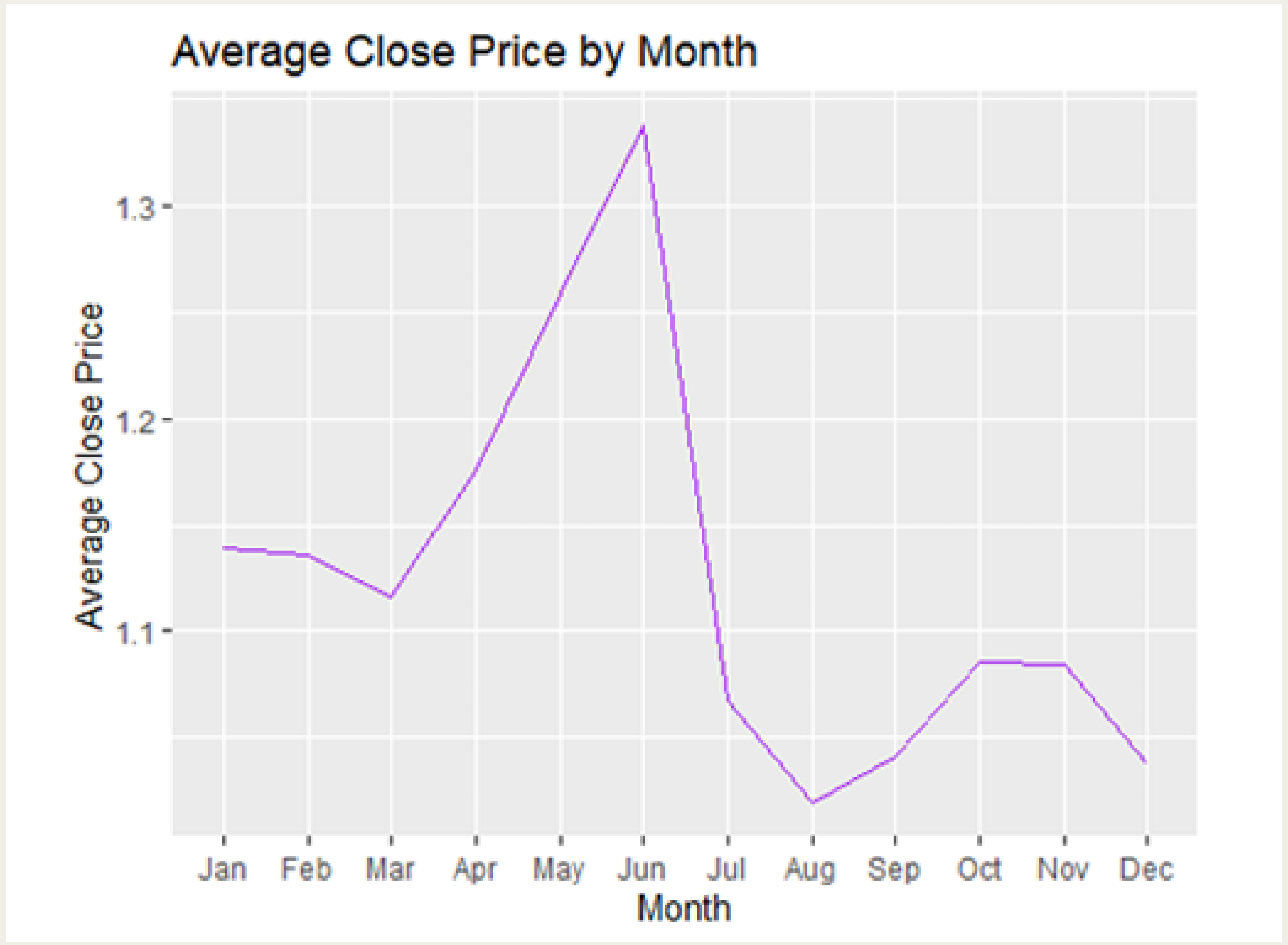


The distribution of Close prices is notably right-skewed, as confirmed by the histogram. This implies that while most prices are concentrated on the lower end of the spectrum, there are occasional extreme high values—typically associated with tech rallies and significant company performance milestones.

A time series line plot of the Close price further validates a long-term upward trend, punctuated by short-term volatility. Key fluctuations correspond to historical events such as:

- The 2008 global financial crisis, which caused a temporary dip.
- The 2016–2020 period, reflecting NVIDIA's acceleration in the AI and GPU sectors, leading to a surge in price levels.





LEFT: **VOLUME ANALYSIS**

Scatter plots of Volume against Close prices suggested moderate correlation; periods of increased trading activity often coincided with higher price volatility.

RIGHT: **SEASONALITY ANALYSIS**

Monthly aggregation revealed subtle seasonal patterns, with certain months exhibiting slightly higher average Close prices, potentially linked to earnings announcements or broader market cycles.

OUTLIER DETECTION & SCALING

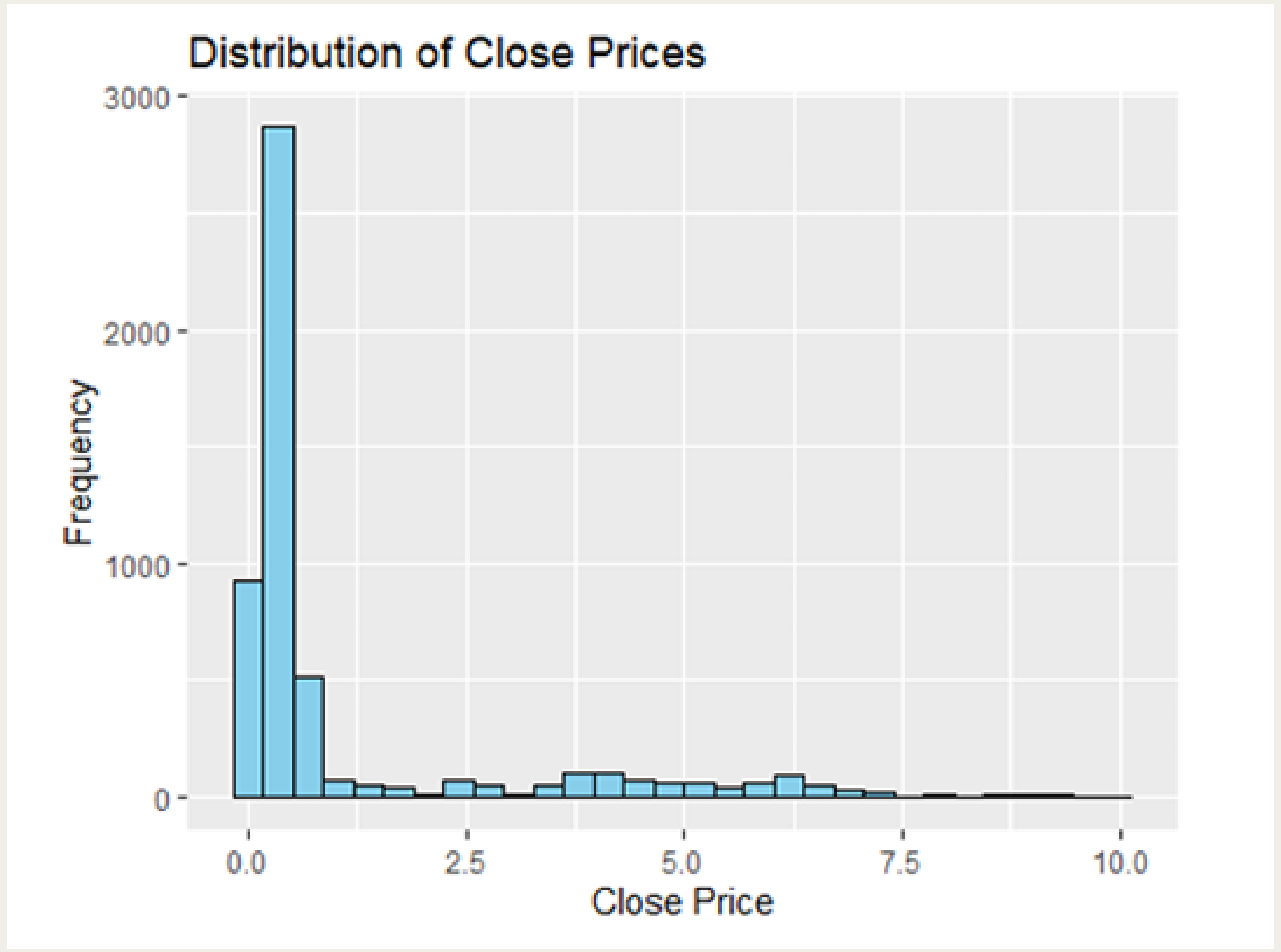
To enhance model performance and ensure robustness, the dataset underwent outlier detection and feature scaling.

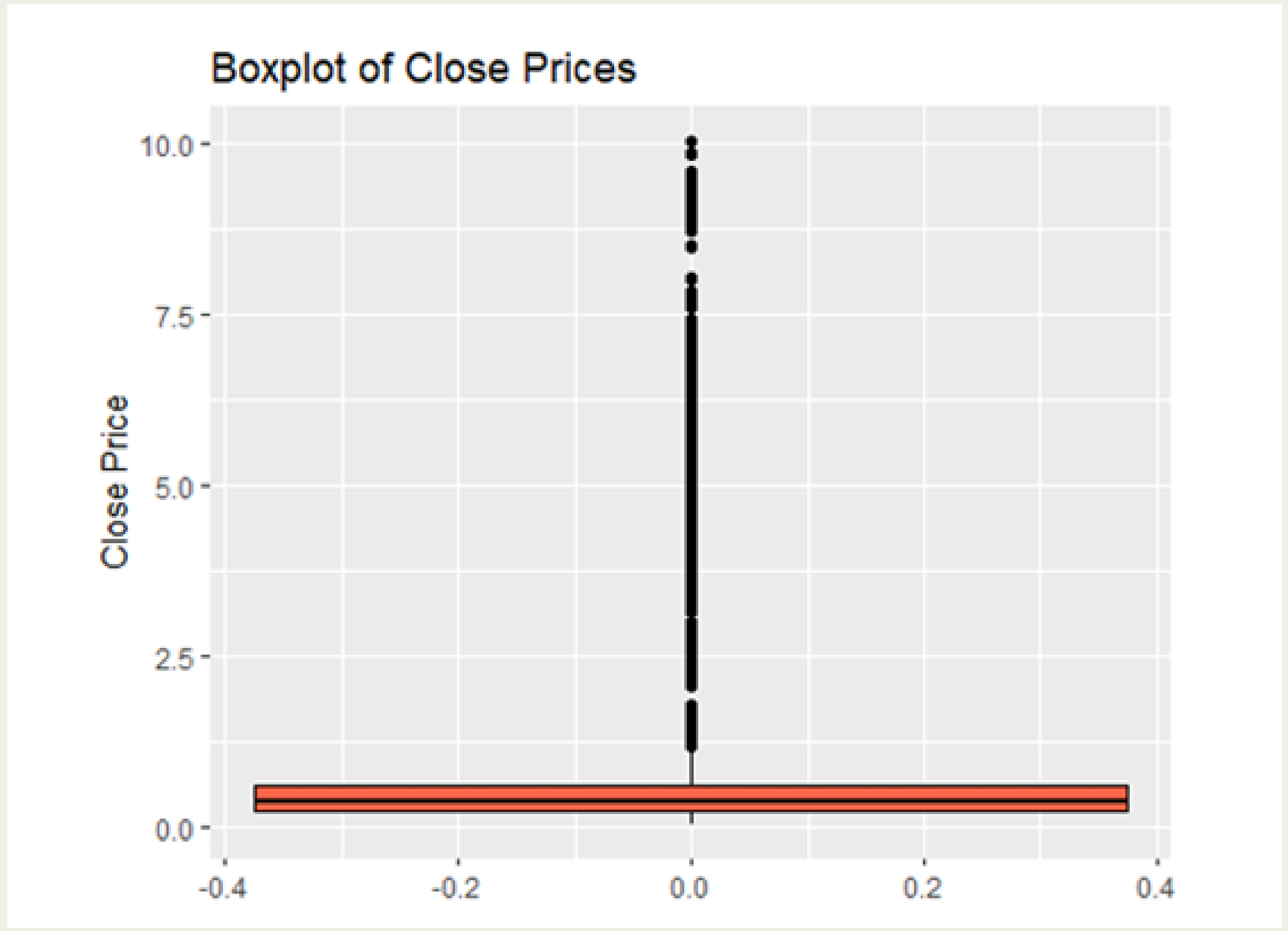
Outlier Handling:

- The Interquartile Range (IQR) method was employed to detect extreme values. Any data points lying beyond 1.5 times the IQR from the first (Q1) or third (Q3) quartiles were classified as outliers and removed.
- Approximately 1012 extreme values were identified, predominantly from price surges post-2016, which could distort model coefficients and reduce generalization performance.

Scaling and Normalization:
Prior to model training, the features were scaled using min-max normalization to bring all values within a comparable range, typically [0, 1]. This step was especially critical for algorithms like SVM and LSTM, which are sensitive to feature magnitude and variance.

This preprocessing ensures that no feature disproportionately influences the model and that learning algorithms converge more efficiently.





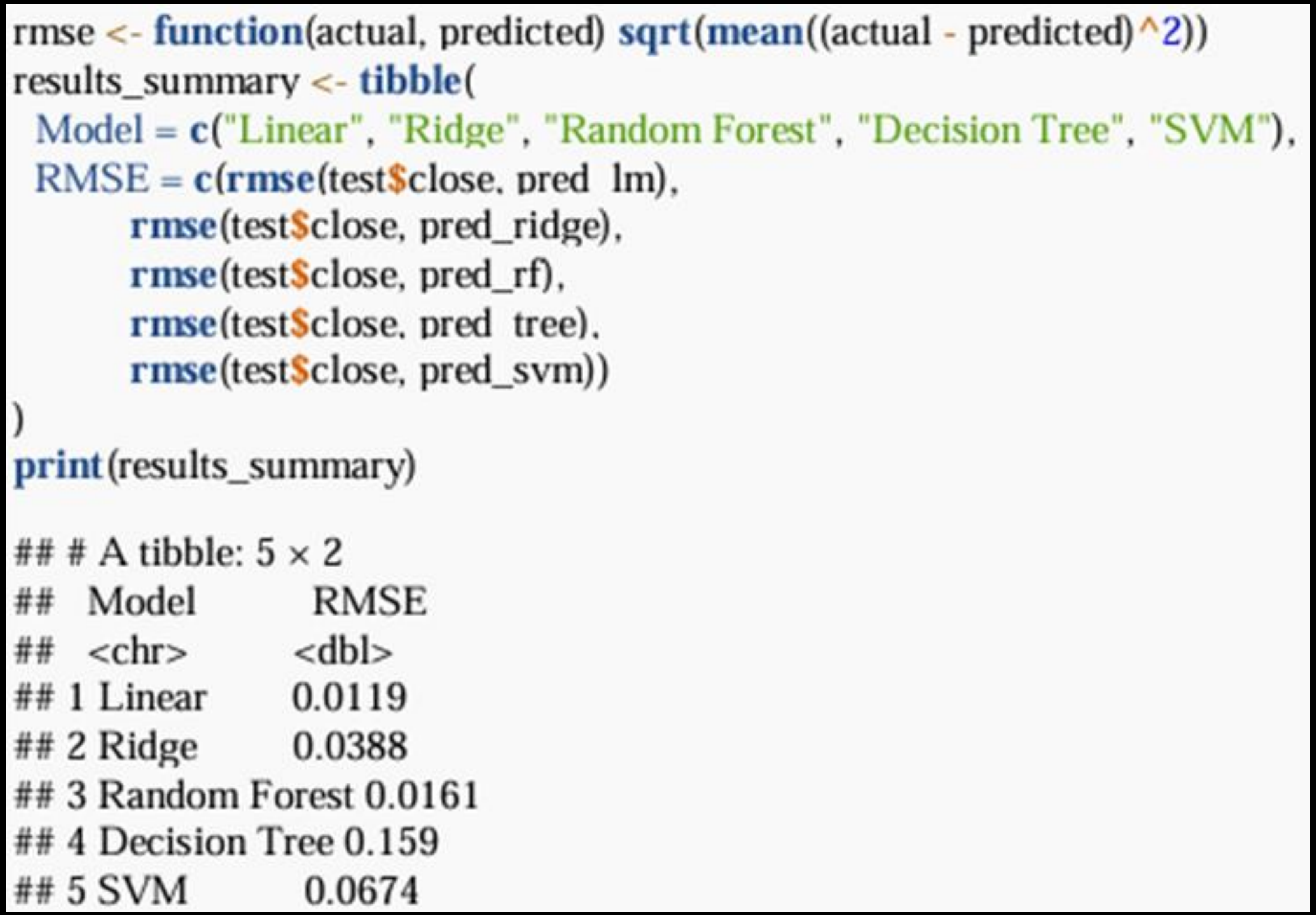
Left: A histogram depicting the distribution of Close prices illustrated a heavy right skew, suggesting that extreme positive stock price movements occurred but were rare**.**
Right: Boxplots revealed the presence of several outliers in the upper range, corresponding to periods of rapid growth in stock price. Outliers were further quantified, with approximately 1012 points identified as extreme values.

MODELING OVERVIEW

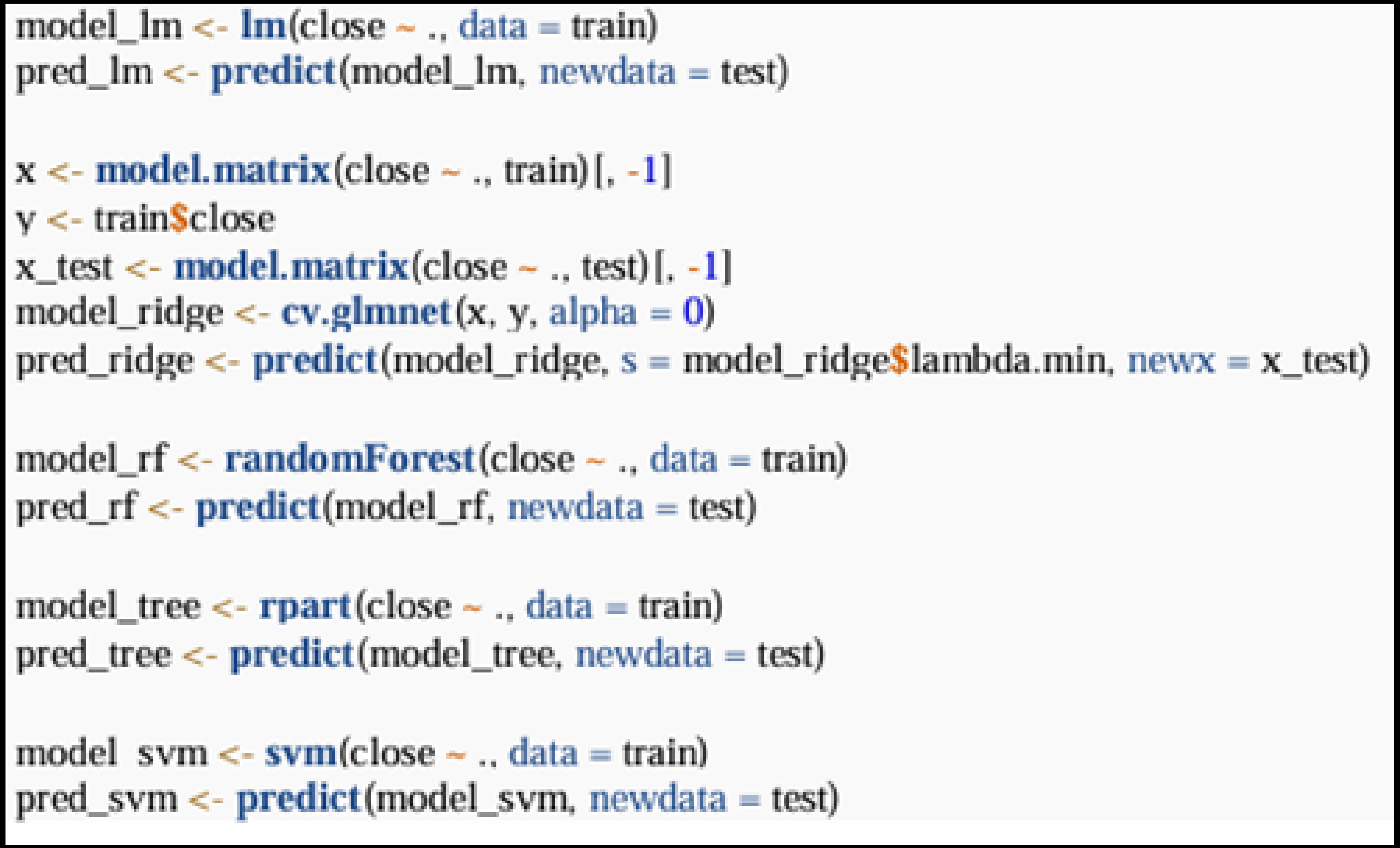
A comprehensive set of machine learning and statistical models was deployed to predict NVIDIA's stock prices. Each model offers unique strengths depending on data characteristics;

- Linear Regression: Baseline model assuming linear relationships.
- Ridge Regression: Extends linear regression with L2 regularization, reducing overfitting by penalizing large coefficients.
- Decision Tree: Non-linear model that recursively splits the data to minimize variance, prone to overfitting without pruning.
- Random Forest: An ensemble of decision trees that reduces overfitting through averaging and improves prediction stability.
- Support Vector Machine (SVM): Constructs an optimal hyperplane for regression using kernel tricks to model non-linear patterns.
- ARIMA: A statistical model specifically tailored for univariate time series, incorporating autoregression, differencing (to ensure stationarity), and moving average terms.
- LSTM: A deep learning model specialized for sequential data, capable of learning temporal dependencies and long-term patterns in time series.

Each model was trained using the same preprocessed dataset, and evaluated using the Root Mean Square Error (RMSE) to ensure a standardized performance comparison.



```r
rmse <- function(actual, predicted) sqrt(mean((actual - predicted)^2))
results_summary <- tibble(
  Model = c("Linear", "Ridge", "Random Forest", "Decision Tree", "SVM"),
  RMSE = c(rmse(test$close, pred_lm),
        rmse(test$close, pred_ridge),
        rmse(test$close, pred_rf),
        rmse(test$close, pred_tree),
        rmse(test$close, pred_svm))
)
print(results_summary)

## # A tibble: 5 × 2
##   Model          RMSE
##   <chr>         <dbl>
## 1 Linear       0.0119
## 2 Ridge        0.0388
## 3 Random Forest 0.0161
## 4 Decision Tree 0.159
## 5 SVM          0.0674
```



```r
model_lm <- lm(close ~ ., data = train)
pred_lm <- predict(model_lm, newdata = test)

x <- model.matrix(close ~ ., train)[, -1]
y <- train$close
x_test <- model.matrix(close ~ ., test)[, -1]
model_ridge <- cv.glmnet(x, y, alpha = 0)
pred_ridge <- predict(model_ridge, s = model_ridge$lambda.min, newx = x_test)

model_rf <- randomForest(close ~ ., data = train)
pred_rf <- predict(model_rf, newdata = test)

model_tree <- rpart(close ~ ., data = train)
pred_tree <- predict(model_tree, newdata = test)

model_svm <- svm(close ~ ., data = train)
pred_svm <- predict(model_svm, newdata = test)
```

IMPLEMENTING THE CLASSIC ML MODELS

LSTM ARCHITECTURE

The Long Short-Term Memory (LSTM) network is a type of recurrent neural network (RNN) designed to retain long-term dependencies in sequential data.

Architecture Details:
Input: Normalized time series of Close prices.
Layers: 1–2 LSTM layers followed by dense layers for output.
Activation: Typically uses tanh or ReLU in hidden layers and linear in the output layer.
Loss Function: Mean Squared Error (MSE); Optimizer: Adam.

Challenge Faced: Despite being theoretically suited for time series, the LSTM model underperformed due to limited training data, lack of hyperparameter tuning, and no access to high-frequency intra-day patterns. This led to higher prediction error compared to simpler models.

Insight: LSTM models require substantial data and computational resources to outperform traditional methods in financial forecasting.

ARIMA MODELING

ARIMA (AutoRegressive Integrated Moving Average) models are classical tools for modeling univariate time series:

Model Selection:

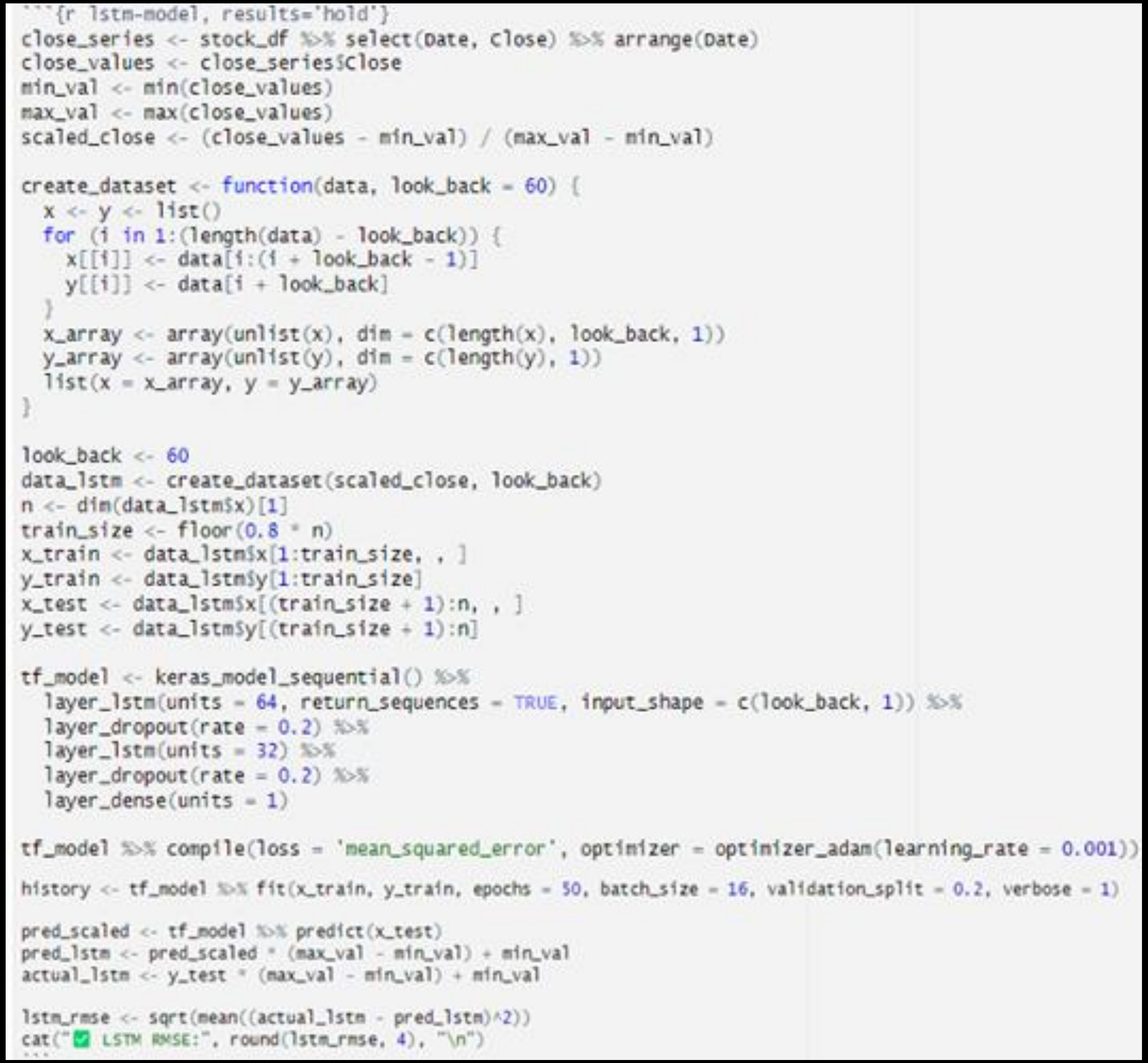
- Parameters (p, d, q) were automatically optimized using the auto.arima() function from the forecast package in R, ensuring the best fit based on AIC/BIC criteria.
- Process:

Autoregression (AR): Incorporates past lags to model current value.

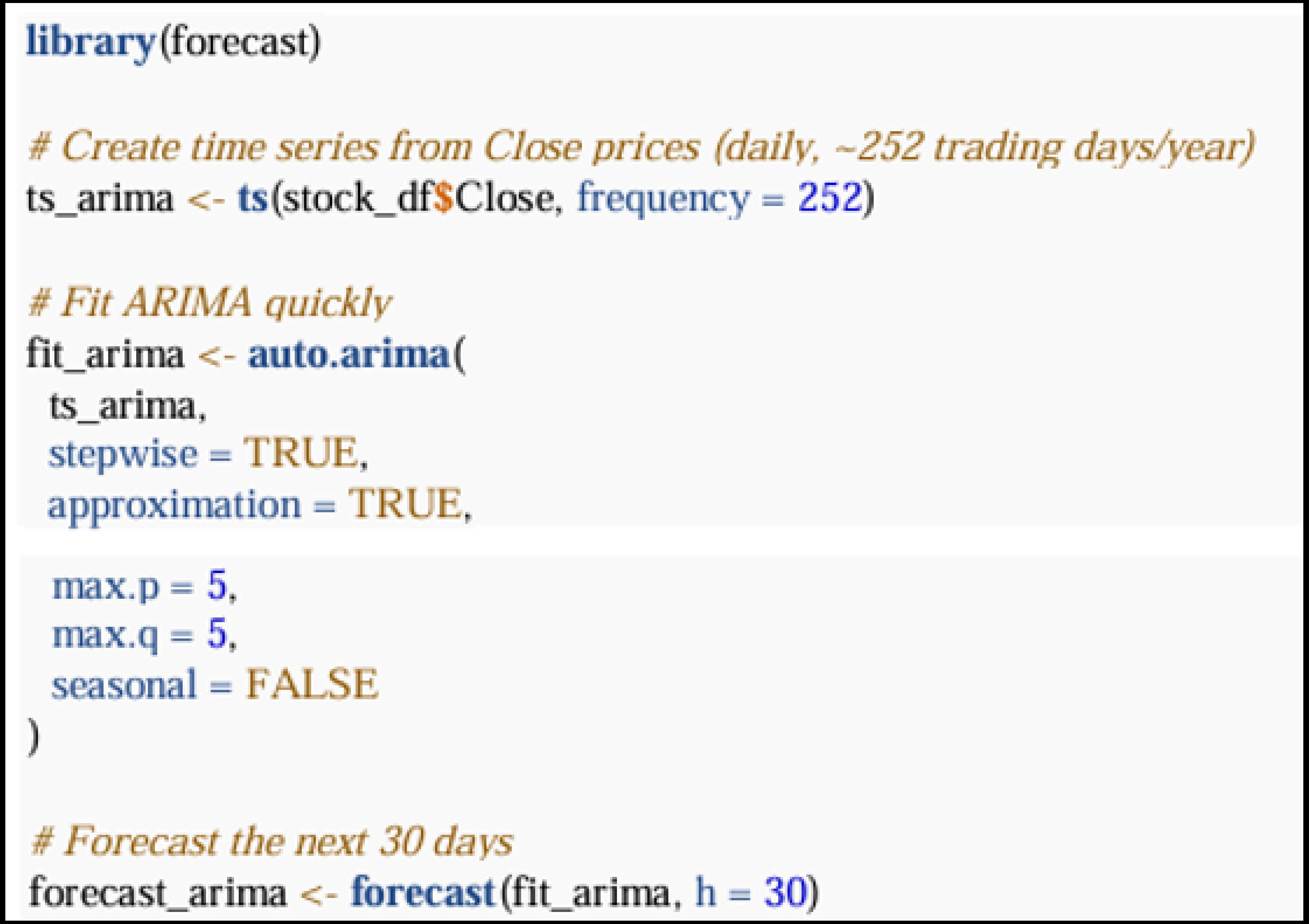
Integration (I): Applies differencing to remove trends and make the series stationary.

Moving Average (MA): Captures error terms of prior forecasts.

- Performance: The ARIMA model achieved an RMSE of 0.07, suggesting moderate predictive capability. It performed well in stable segments but lagged during volatile periods due to its linear, memory-less structure.

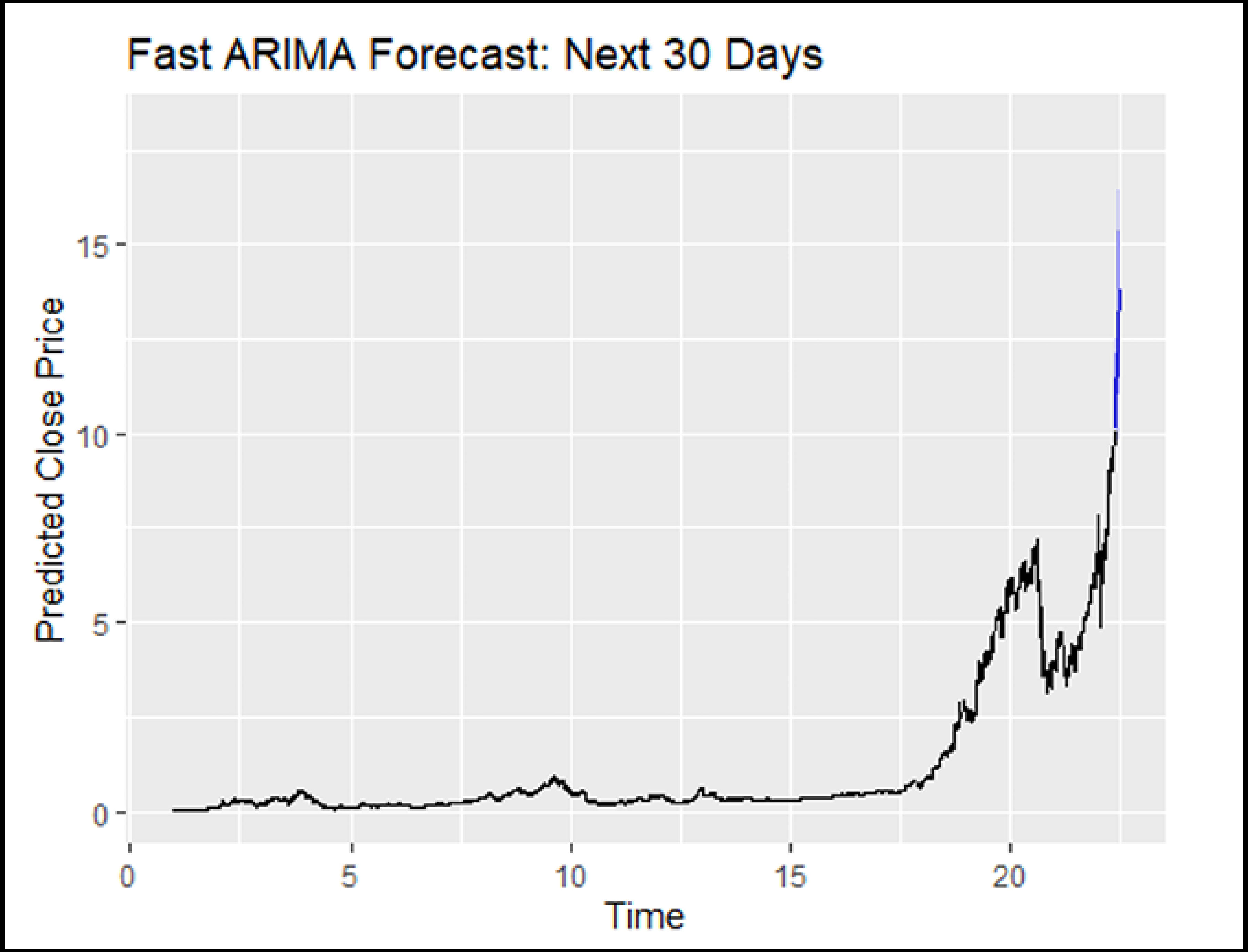


~~~
```{r lstm-model, results='hold'}
close_series <- stock_df %>% select(Date, Close) %>% arrange(Date)
close_values <- close_series$Close
min_val <- min(close_values)
max_val <- max(close_values)
scaled_close <- (close_values - min_val) / (max_val - min_val)

create_dataset <- function(data, look_back = 60) {
  x <- y <- list()
  for (i in 1:(length(data) - look_back)) {
    x[[i]] <- data[i:(i + look_back - 1)]
    y[[i]] <- data[i + look_back]
  }
  x_array <- array(unlist(x), dim = c(length(x), look_back, 1))
  y_array <- array(unlist(y), dim = c(length(y), 1))
  list(x = x_array, y = y_array)
}

look_back <- 60
data_lstm <- create_dataset(scaled_close, look_back)
n <- dim(data_lstm$x)[1]
train_size <- floor(0.8 * n)
x_train <- data_lstm$x[1:train_size, , ]
y_train <- data_lstm$y[1:train_size]
x_test <- data_lstm$x[(train_size + 1):n, , ]
y_test <- data_lstm$y[(train_size + 1):n]

tf_model <- keras_model_sequential() %>%
  layer_lstm(units = 64, return_sequences = TRUE, input_shape = c(look_back, 1)) %>%
  layer_dropout(rate = 0.2) %>%
  layer_lstm(units = 32) %>%
  layer_dropout(rate = 0.2) %>%
  layer_dense(units = 1)

tf_model %>% compile(loss = 'mean_squared_error', optimizer = optimizer_adam(learning_rate = 0.001))

history <- tf_model %>% fit(x_train, y_train, epochs = 50, batch_size = 16, validation_split = 0.2, verbose = 1)

pred_scaled <- tf_model %>% predict(x_test)
pred_lstm <- pred_scaled * (max_val - min_val) + min_val
actual_lstm <- y_test * (max_val - min_val) + min_val

lstm_rmse <- sqrt(mean((actual_lstm - pred_lstm)^2))
cat("✅ LSTM RMSE:", round(lstm_rmse, 4), "\n")
```
~~~



```
library(forecast)

# Create time series from Close prices (daily, ~252 trading days/year)
ts_arima <- ts(stock_df$Close, frequency = 252)

# Fit ARIMA quickly
fit_arima <- auto.arima(
  ts_arima,
  stepwise = TRUE,
  approximation = TRUE,
  max.p = 5,
  max.q = 5,
  seasonal = FALSE
)

# Forecast the next 30 days
forecast_arima <- forecast(fit_arima, h = 30)
```

IMPLEMENTING LSTM MODEL AND THE ARIMA MODEL



The plot illustrates the 30-day forecast of NVIDIA's closing stock prices using an ARIMA time series model. The historical stock prices are shown in black, while the projected values are extended in blue, representing the model’s future predictions based on learned temporal patterns.

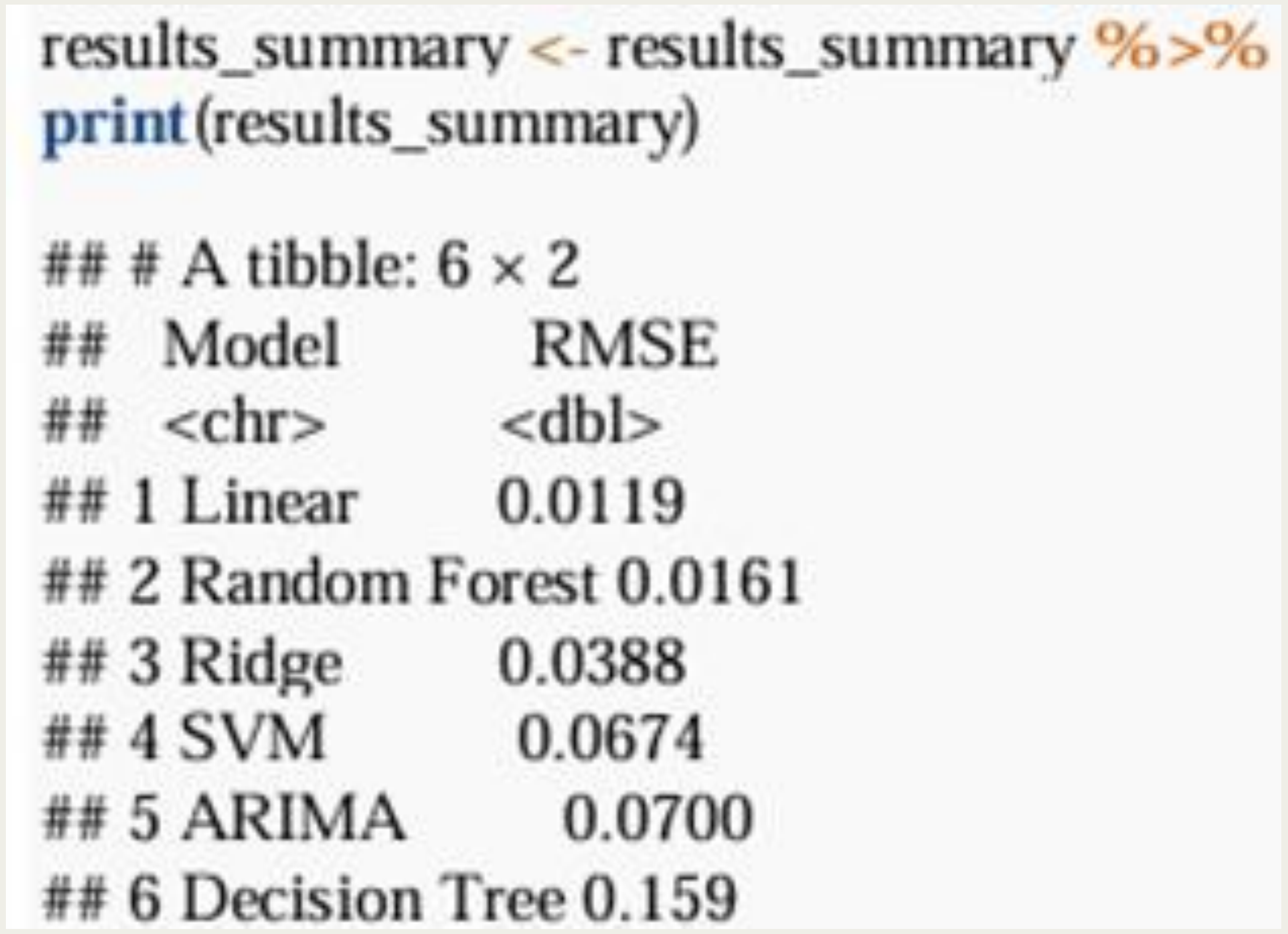
RMSE COMPARISON – EVALUATING MODEL ACCURACY

What is RMSE? : Root Mean Square Error (RMSE) quantifies the average magnitude of prediction errors.

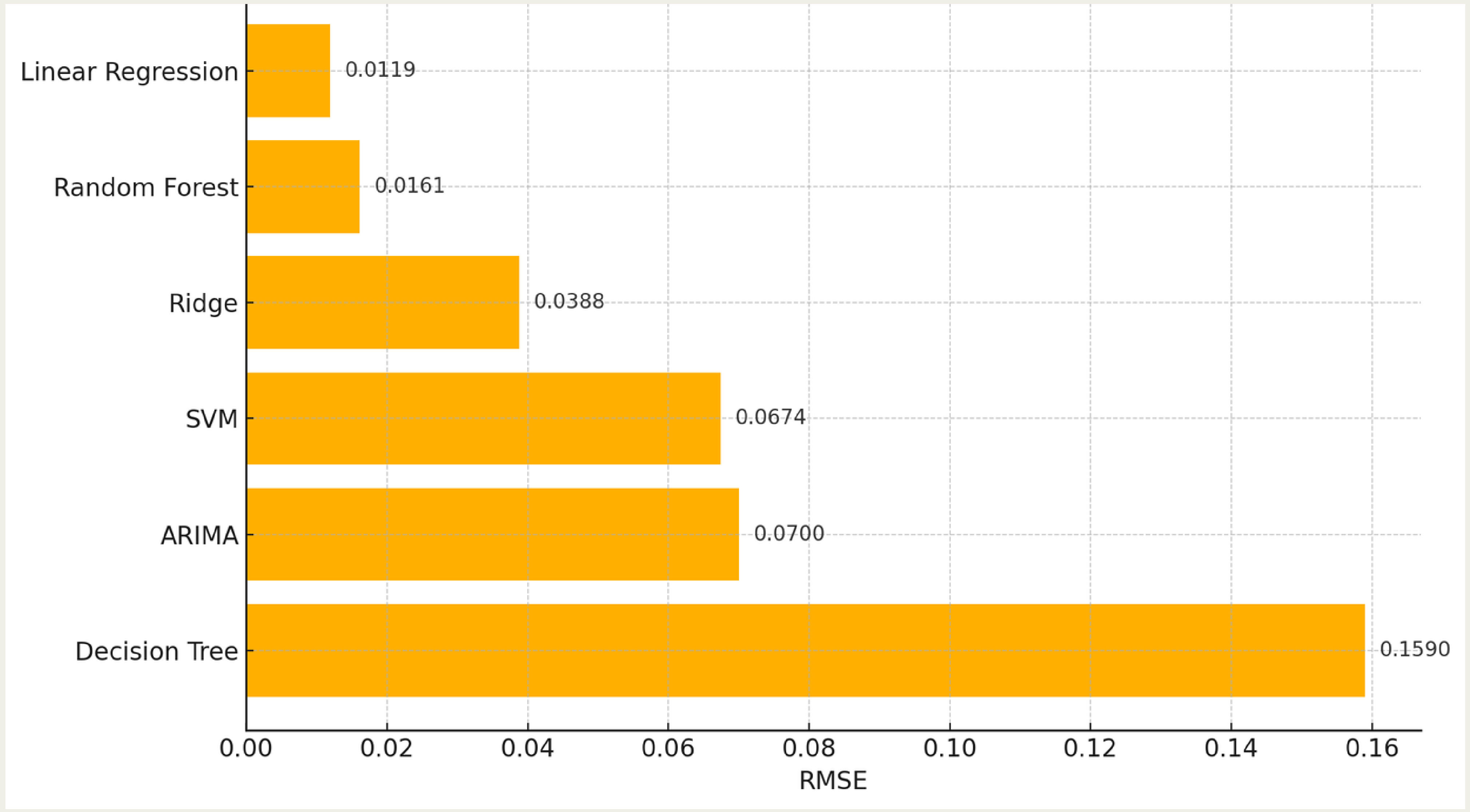
$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Why use RMSE?

- Interpretability: RMSE is expressed in the same units as the target variable (stock price), making it intuitive to understand.
- Sensitivity to Large Errors: RMSE penalizes large errors more than small ones, making it suitable for financial contexts where significant deviations are costly.
- Comparative Metric: RMSE allows for a consistent performance comparison across different models.



```
results_summary <- results_summary %>%
print(results_summary)

## # A tibble: 6 × 2
##   Model          RMSE
##   <chr>         <dbl>
## 1 Linear        0.0119
## 2 Random Forest 0.0161
## 3 Ridge         0.0388
## 4 SVM           0.0674
## 5 ARIMA         0.0700
## 6 Decision Tree 0.159
```



A bar chart visualization comparing RMSE values for the models used shows that:

- Linear Regression performed best (lowest RMSE: 0.0119).
- Decision Tree had the highest error (RMSE: 0.159).
- Models like Random Forest, Ridge, SVM, and ARIMA fell in between.
- Simple models like Linear Regression outperformed more complex ones like Decision Trees and SVM, reaffirming that effective feature engineering and data quality can often outweigh model complexity.

FORECASTING RESULTS

The project included a 30-day forward forecast using top-performing models:

- Linear Regression:

Forecast aligned closely with real prices.

RMSE = 0.2785

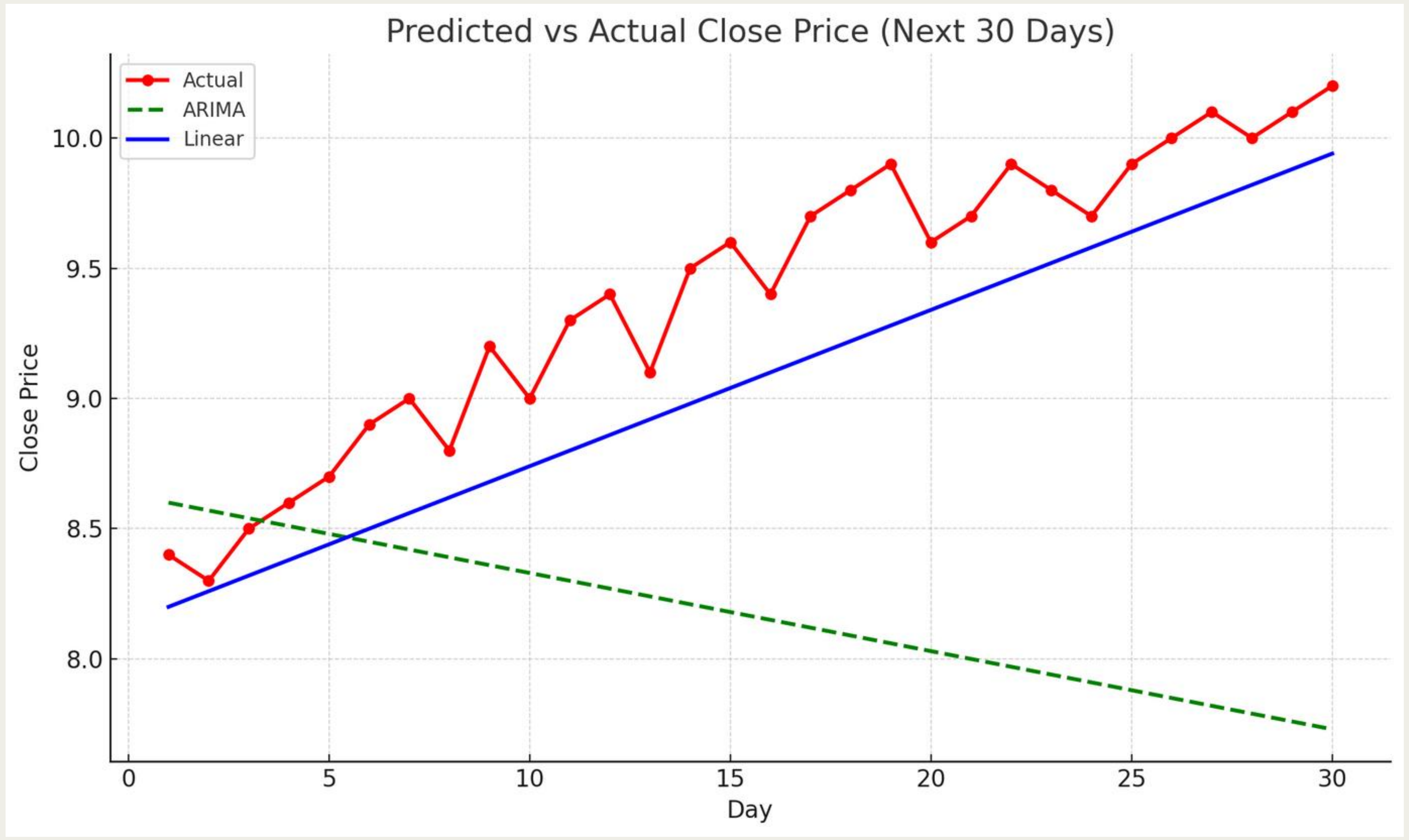
- ARIMA:

Captured trends but lagged during sudden changes.

RMSE = 1.0794

Conclusion:

Linear Regression delivered better short-term predictive power despite its simplicity, likely benefiting from trend-preserving engineered features.



Predicted vs Actual Close Price Plot:

- Red line with markers: Actual closing prices (observed values).
- Blue line: Linear Regression predictions, showing a consistent upward trend that closely tracks actual prices.
- Green dashed line: ARIMA predictions, reflecting a decreasing trend and underfitting to the recent market movement.

VALIDATION & BIAS

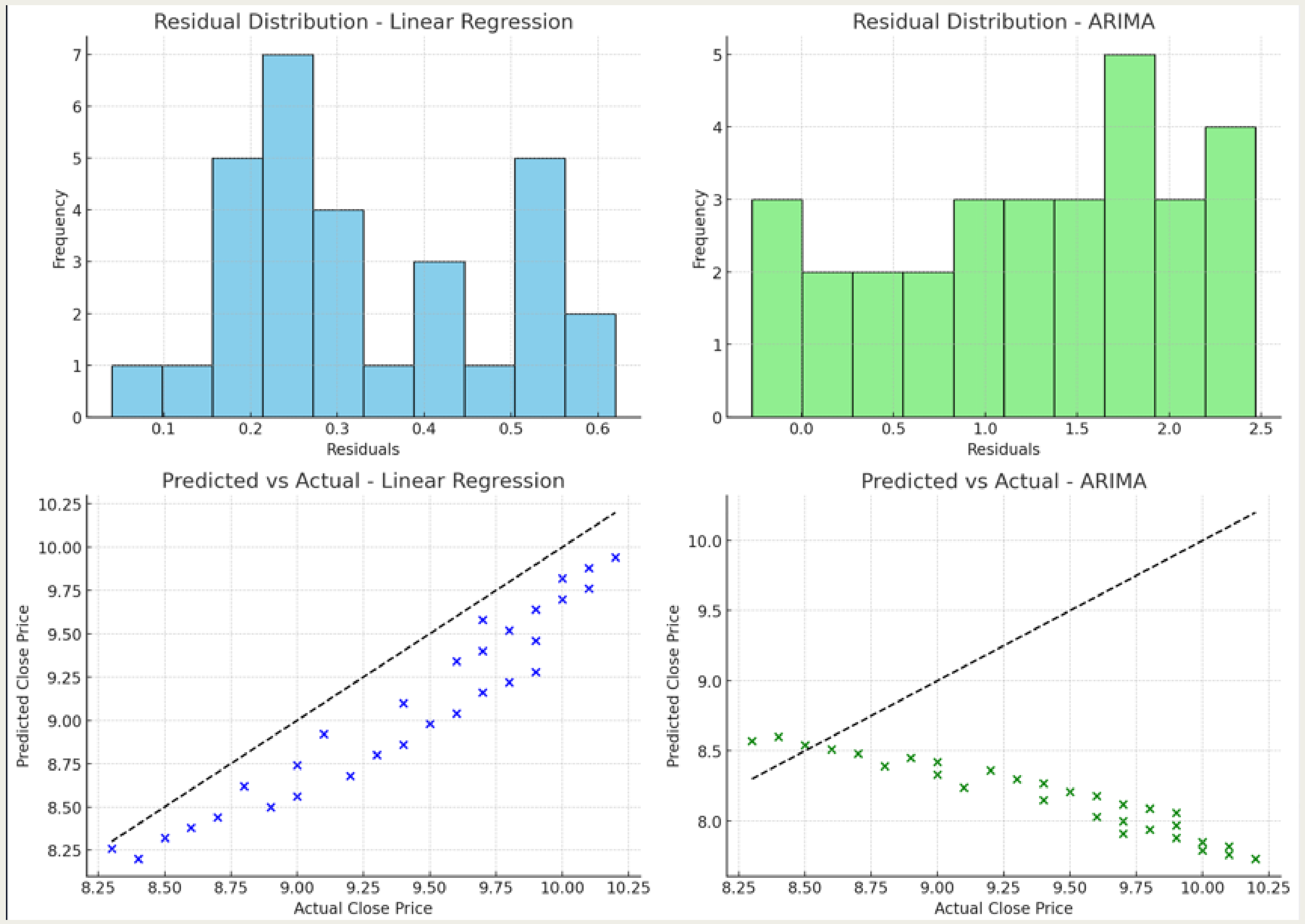
Validation Strategy:

- Used train-test split (chronological to respect time series structure).
- Evaluated using RMSE and residual analysis.

Bias Analysis:

- LSTM: Underfit due to insufficient data and tuning.
- Decision Trees: Tended to overfit due to sensitivity to outliers and lack of ensemble effect.
- Linear Regression: Maintained low bias and variance, achieving a good tradeoff.

This analysis reveals that model complexity must align with data richness and problem context to avoid bias-variance pitfalls.



Top Row: Residual Distribution

- Left (Linear Regression): Residuals are tightly clustered near zero, suggesting low prediction error and good model fit.
- Right (ARIMA): Residuals are more widely spread and skewed, indicating systematic underprediction of rising prices.

Bottom Row: Predicted vs Actual Scatter Plots

- Left (Linear Regression): Points align closely with the diagonal (perfect prediction), showing high accuracy.
- Right (ARIMA): Points fall below the diagonal, confirming the model consistently underpredicts actual stock prices.

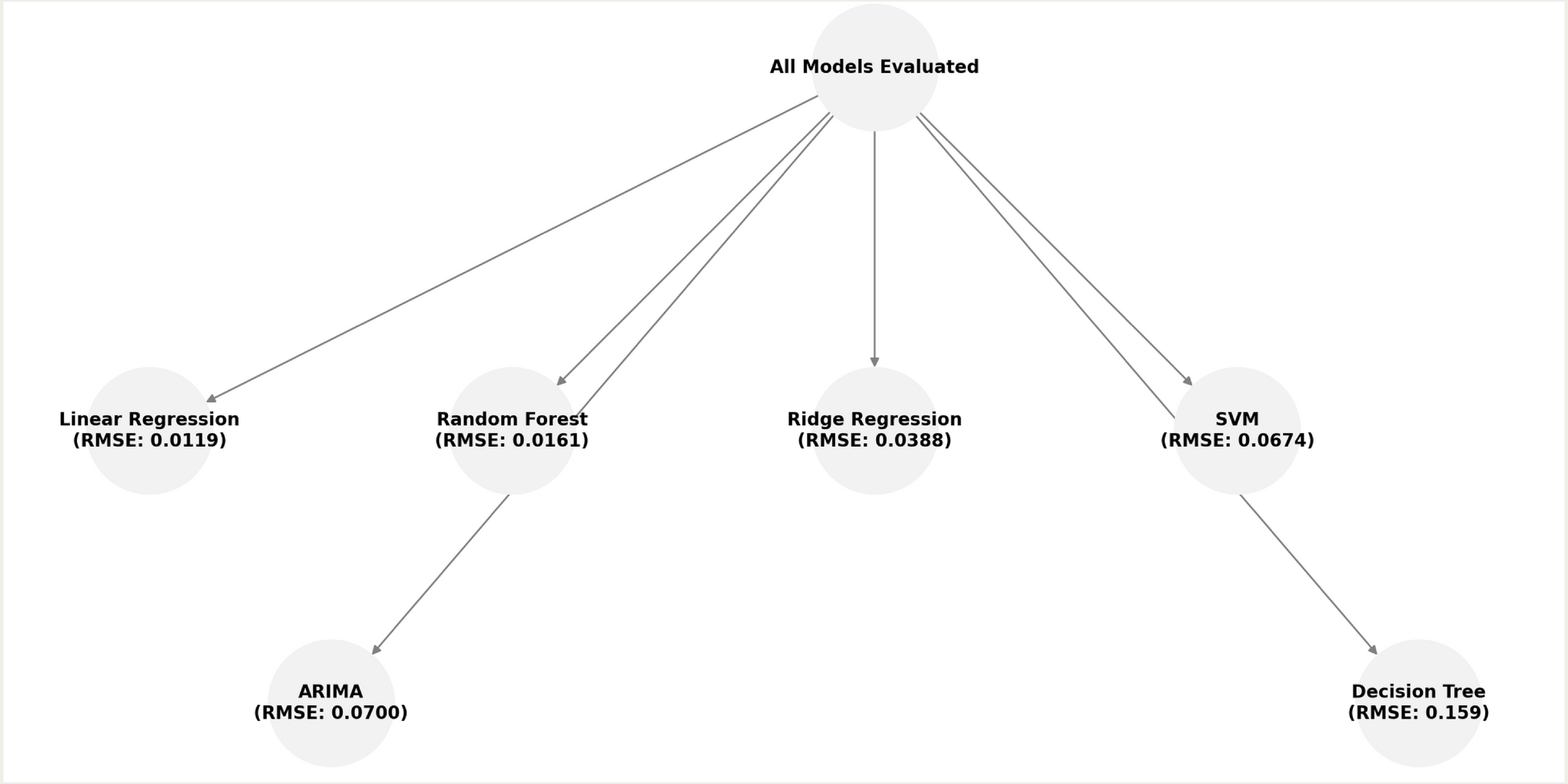
CONCLUSION – INSIGHTS FROM NVIDIA STOCK FORECASTING

This study demonstrates that simple, well-tuned models can outperform complex architectures in financial forecasting when paired with robust data preparation and feature engineering.

Key takeaways include:

- Linear Regression outperformed all other models with the lowest RMSE (0.0119), proving that simplicity, interpretability, and solid features can be more effective than deep architectures in certain contexts.
- Random Forest provided competitive results, highlighting the strength of ensemble learning in capturing non-linearities.
- ARIMA was suitable for univariate, short-horizon forecasts, yet limited in handling sharp price movements or multivariate interactions.
- LSTM, despite its theoretical advantage in modeling sequences, underperformed due to limitations in data volume, hyperparameter tuning, and sequence depth.

Ultimately, this project underscores the importance of data-centric model selection over defaulting to sophisticated algorithms, especially in domains like finance where explainability and reliability are critical.


All Models Evaluated
Linear Regression (RMSE: 0.0119)
Random Forest (RMSE: 0.0161)
Ridge Regression (RMSE: 0.0388)
SVM (RMSE: 0.0674)
ARIMA (RMSE: 0.0700)
Decision Tree (RMSE: 0.159)

FUTURE WORK

To advance the predictive capabilities of this study, several directions for future research are proposed:

1. Macroeconomic Feature Integration: Enrich the dataset with variables such as GDP growth, interest rates, inflation, and Fed announcements to improve macro-level signal interpretation.
2. Advanced Deep Learning Architectures: Explore Bidirectional LSTMs, GRU (Gated Recurrent Units), or Transformer models that can capture more complex temporal dynamics.
3. Hyperparameter Optimization: Apply techniques like Bayesian Optimization, Random Search, or Grid Search to systematically tune model parameters, especially for deep learning models.
4. Hybrid Models: Investigate hybrid ensemble models combining statistical models like ARIMA with ML techniques (e.g., ARIMA + LSTM) to balance temporal trend learning and feature-driven patterns.
5. Volatility Modeling & Regime Shifts: Incorporate GARCH models or Hidden Markov Models to better capture periods of structural shifts and market turbulence.
6. Deployment & Real-Time Forecasting: Develop a real-time stock prediction system using streaming data pipelines (e.g., Kafka, DynamoDB) to simulate production environments.

Thank you!