I. ABSTRACT

This report presents a comprehensive analysis of stock price forecasting and volatility trends for leading technology stocks from 2017 to 2024. The study focuses on using statistical and deep learning techniques—specifically ARIMA and LSTM models—to predict future stock prices based on historical adjusted closing values. The dataset, sourced from a CSV file containing multiple AI-related stocks, was cleaned and reshaped for analysis in R. Exploratory Data Analysis (EDA) revealed key insights into stock trends, price distributions, and correlations. ARIMA models provided robust short-term forecasts with interpretable parameters, while LSTM networks, built using the torch package, captured nonlinear patterns in sequential data. Rolling standard deviation was used to assess volatility across time. Evaluation metrics such as MAE, RMSE, MAPE, and SMAPE were used to compare model performance. While ARIMA offered consistent results, LSTM showed promising performance for specific stocks. The report concludes with insights on model accuracy, stock behavior, and potential future enhancements.

II. INTRODUCTION

Stock price forecasting plays a pivotal role in financial decision-making, portfolio management, and risk assessment. Accurately predicting future stock movements can provide investors with a competitive edge, especially in volatile markets influenced by rapid technological advancements. As artificial intelligence (AI) and machine learning continue to shape the global economy, AI-related stocks have drawn significant interest from analysts and investors alike.

This report aims to explore and compare two powerful time series forecasting approaches—ARIMA (AutoRegressive Integrated Moving Average), a classical statistical model, and LSTM (Long Short-Term Memory), a deep learning architecture well-suited for sequential data. The objective is to model stock prices, evaluate forecasting performance, and understand the volatility of top AI-related stocks over a seven-year period (2017–2024).

The analysis is conducted using R, leveraging a range of packages such as tidyverse for data manipulation, forecast for ARIMA modeling, torch for deep learning with LSTM, and ggplot2 for data visualization. By combining traditional and modern techniques, this study seeks to uncover insights into stock behavior and assess the practical utility of each forecasting model.

III. DATASET OVERVIEW

The dataset used for this study contains historical stock price data for multiple AI-related companies, covering the period from June 2017 to May 2024. The data was sourced from a consolidated CSV file that includes daily stock information such as:

- **date** – Trading day in DD-MM-YYYY format
- **symbol –** Ticker symbol for each stock (e.g., NVDA, AMD, AI)
- **adjusted** – Adjusted closing price accounting for dividends and splits
- **open, high, low, close, volume** – Standard OHLCV market data

The dataset initially included stock data for 30+ companies, but after preprocessing and handling missing values, the analysis focused on the top 10 AI-related stocks with the most complete and consistent data.

To facilitate time series modeling, the dataset was pivoted to a wide format, where each column represented the adjusted closing price of a specific stock over time. Stocks with excessive missing values were excluded to ensure the robustness of the forecasting models.

The final dataset used for analysis had the following characteristics:

- Time Period: 2017–2024
- Stocks Analyzed: 10
- Key Columns: date, adjusted, symbol
- Final Shape: 15754 rows × 8 columns

IV. DATA PREPARATION

To ensure the accuracy and consistency of results, the raw dataset underwent a structured data cleaning and preprocessing pipeline using R. Below are the key steps involved:

1) **Date Conversion**
   The date column, initially read as a character string, was converted to proper Date format using:

   stock_data$date <- as.Date(stock_data$date)

   This step ensured that time-based operations, plotting, and filtering would work correctly throughout the analysis.

2) **Missing Value Check**
   The dataset was inspected for missing values using:

   colSums(is.na(stock_data))

   Only a small number of missing values were identified. These were either forward-filled or, in the case of stocks with excessive missing values, the stock was removed entirely from the analysis. This step guaranteed that time series models like ARIMA and LSTM would receive clean, continuous data sequences.

3) **Column Selection**
   For modeling and visualization purposes, only the necessary columns were retained:
   - Date
   - symbol (stock ticker)
   - close (daily closing price)

   stock_data <- stock_data %>%
   select(date, symbol, close)

4) **Sorting and Time Order**
   To maintain chronological consistency within each stock, the data was arranged by symbol and date:
   stock_data <- stock_data %>%
   arrange(symbol, date)

5) **Stock Count Validation**
   Before reshaping, the dataset was checked for the number of unique stocks using:

   unique(stock_data$symbol)

   This ensured that only valid and active stocks were included in the analysis. After this step, a total of 10 AI-related stocks were retained for modeling, based on data completeness and relevance.

```r
library(dplyr)

stock_data <- read.csv("C:/Users/skyline/OneDrive/Desktop/top_10_stocks_17-24.csv", stringsAsFactors = FALSE)
str(stock_data)
stock_data$date <- as.Date(stock_data$date)
colSums(is.na(stock_data))


stock_data <- stock_data %>%
  select(date, symbol, close)

stock_data <- stock_data %>%
  arrange(symbol, date)

unique(stock_data$symbol)
```

V. METHODOLOGIES AND EXPERIMENTS

A. Summary Statistics and Exploratory Data Analysis (EDA):

To understand each stock better:

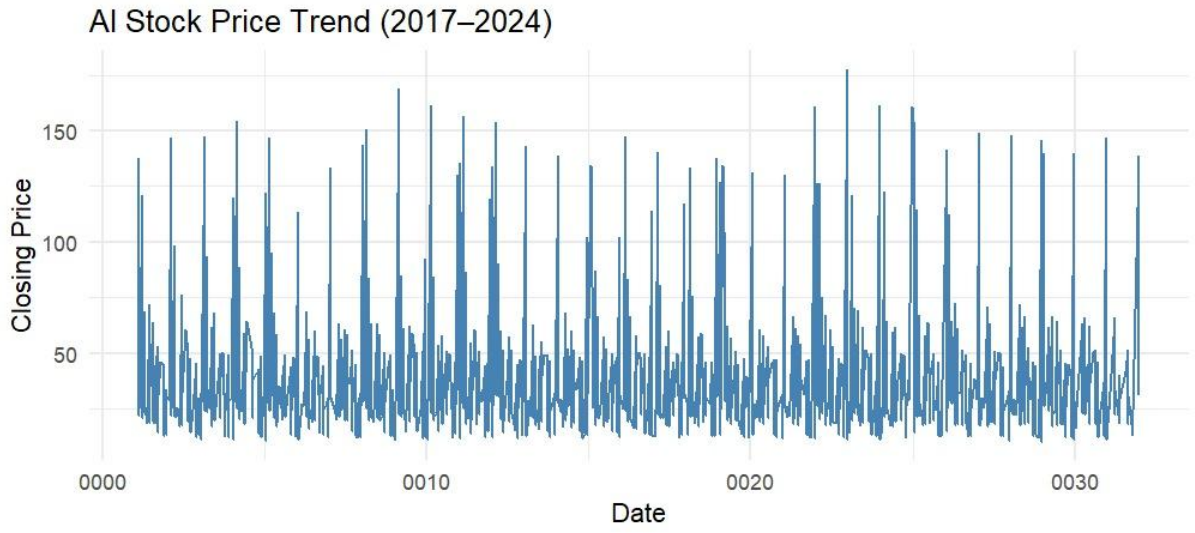
- Basic stats: mean, median, min, max, sd of closing prices for each stock were calculated.
- Plot:
  - Lineplot of trends grouped by stock.
  - Barplot of closing price distribution.

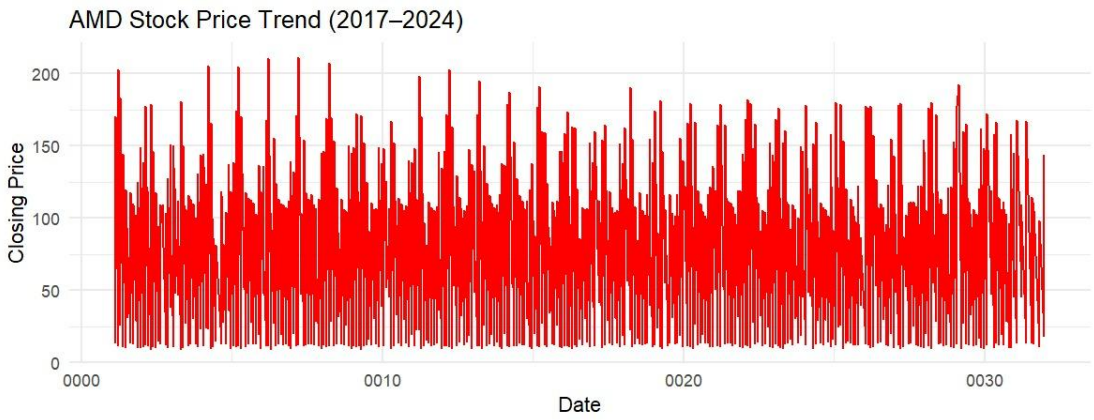
1. Line plot for each stock trends:

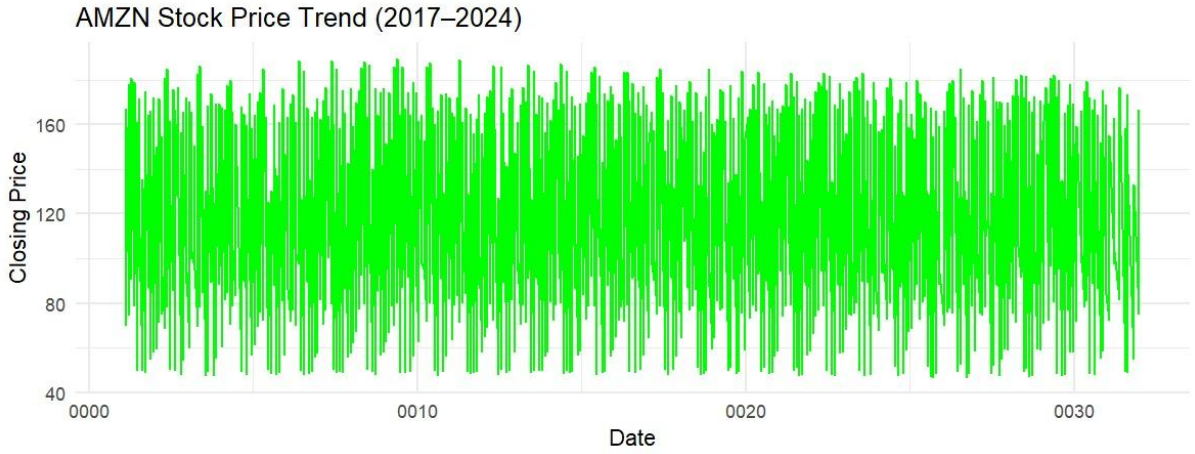
```r
library(ggplot2)

ggplot(filter(stock_data, symbol == "AI"), aes(x = date, y = close)) +
  geom_line(color = "steelblue") +
  labs(title = "AI Stock Price Trend (2017–2024)",
       x = "Date", y = "Closing Price") +
  theme_minimal()

ggplot(filter(stock_data, symbol == "AMD"), aes(x = date, y = close)) +
  geom_line(color = "red") +
  labs(title = "AMD Stock Price Trend (2017–2024)",
       x = "Date", y = "Closing Price") +
  theme_minimal()

ggplot(filter(stock_data, symbol == "AMZN"), aes(x = date, y = close)) +
  geom_line(color = "green") +
  labs(title = "AMZN Stock Price Trend (2017–2024)",
       x = "Date", y = "Closing Price") +
  theme_minimal()

ggplot(filter(stock_data, symbol == "ANET"), aes(x = date, y = close)) +
  geom_line(color = "yellow") +
  labs(title = "ANET Stock Price Trend (2017–2024)",
       x = "Date", y = "Closing Price") +
  theme_minimal()

ggplot(filter(stock_data, symbol == "META"), aes(x = date, y = close)) +
  geom_line(color = "purple") +
  labs(title = "META Stock Price Trend (2017–2024)",
       x = "Date", y = "Closing Price") +
  theme_minimal()
```

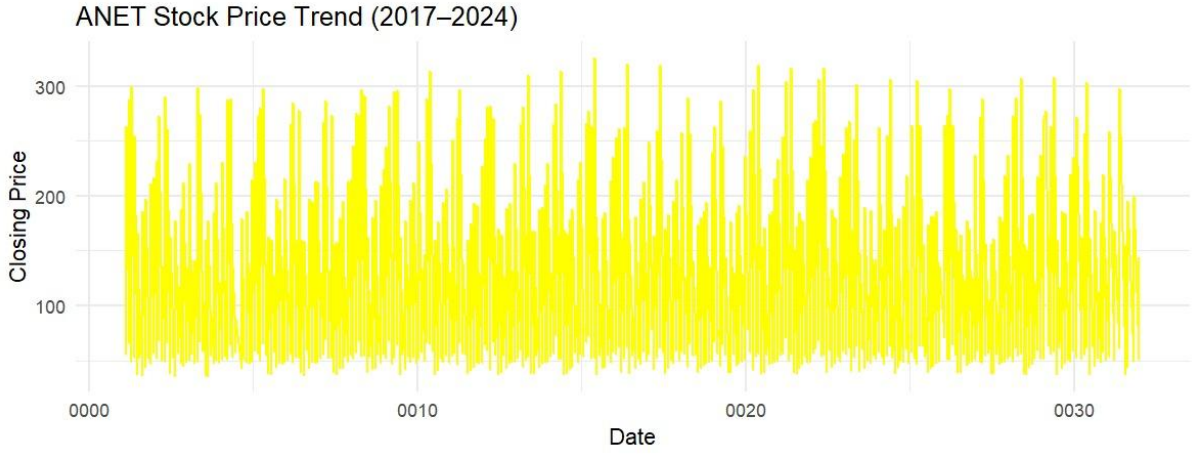
```r
ggplot(filter(stock_data, symbol == "NOW"), aes(x = date, y = close)) +
  geom_line(color = "orange") +
  labs(title = "NOW Stock Price Trend (2017–2024)",
       x = "Date", y = "Closing Price") +
  theme_minimal()

ggplot(filter(stock_data, symbol == "NVDA"), aes(x = date, y = close)) +
  geom_line(color = "blue") +
  labs(title = "NVDA Stock Price Trend (2017–2024)",
       x = "Date", y = "Closing Price") +
  theme_minimal()

ggplot(filter(stock_data, symbol == "PANW"), aes(x = date, y = close)) +
  geom_line(color = "brown") +
  labs(title = "PANW Stock Price Trend (2017–2024)",
       x = "Date", y = "Closing Price") +
  theme_minimal()

ggplot(filter(stock_data, symbol == "PATH"), aes(x = date, y = close)) +
  geom_line(color = "pink") +
  labs(title = "PATH Stock Price Trend (2017–2024)",
       x = "Date", y = "Closing Price") +
  theme_minimal()

ggplot(filter(stock_data, symbol == "TSLA"), aes(x = date, y = close)) +
  geom_line(color = "black") +
  labs(title = "TESLA Stock Price Trend (2017–2024)",
       x = "Date", y = "Closing Price") +
  theme_minimal()
```

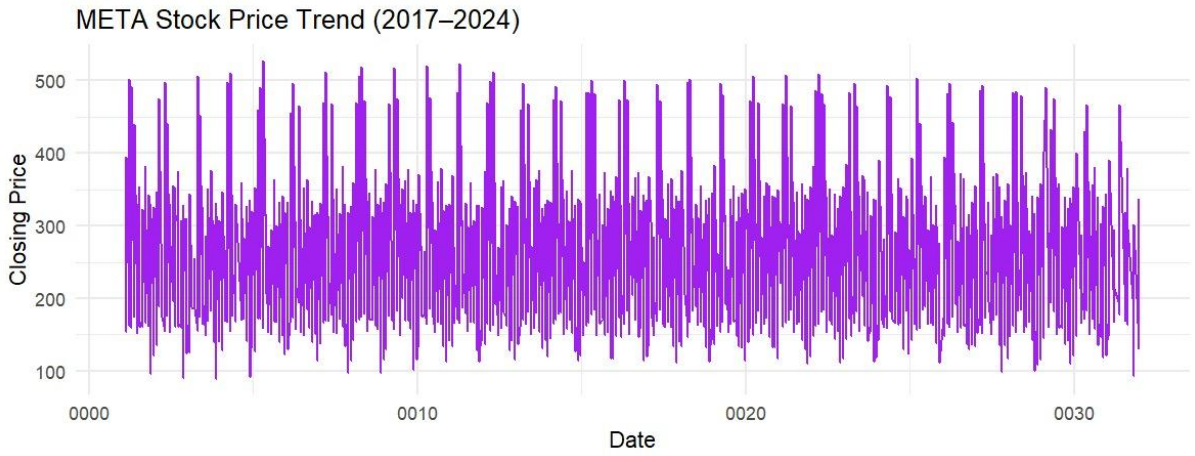
Output:

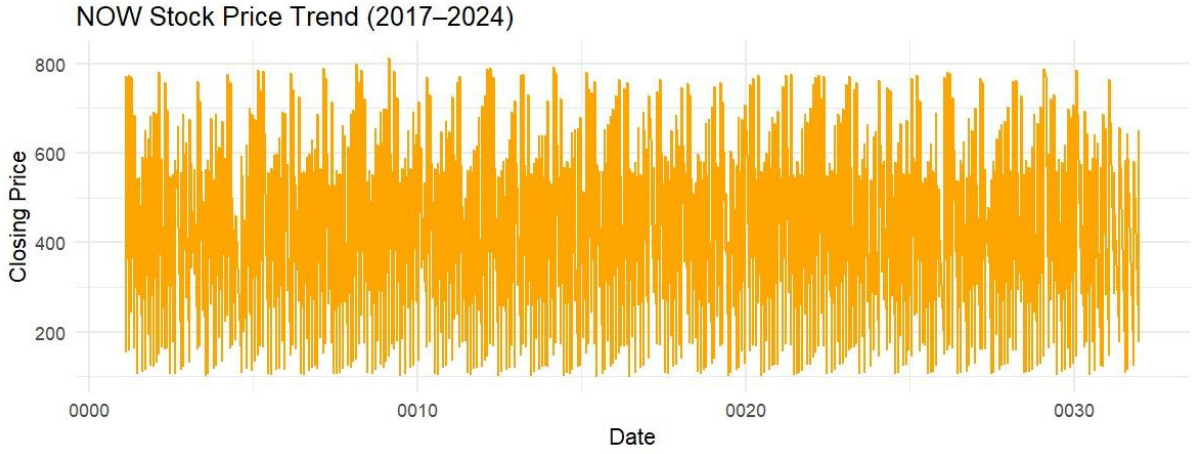


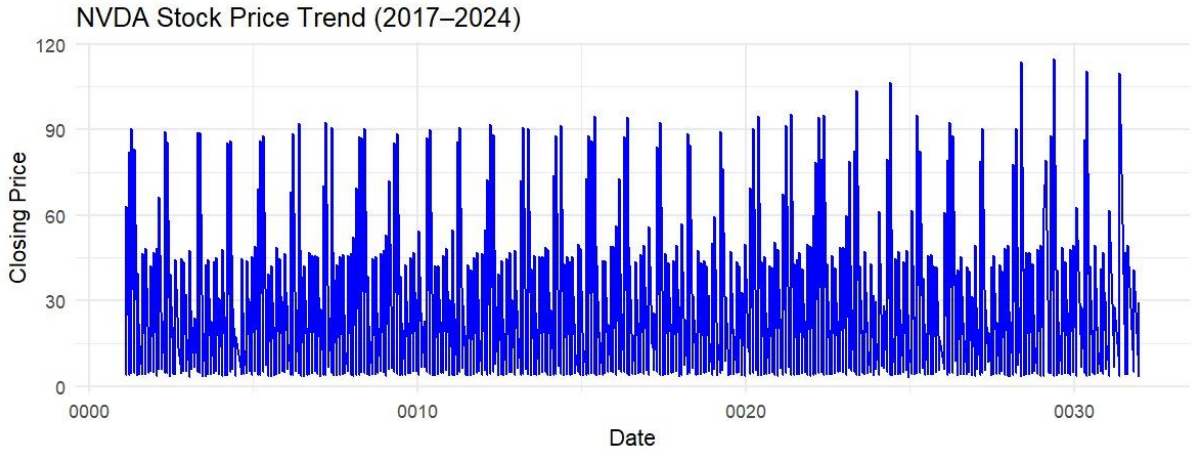

AMD Stock Price Trend (2017–2024)
Closing Price
200
150
100
50
0
0000
0010
0020
0030
Date

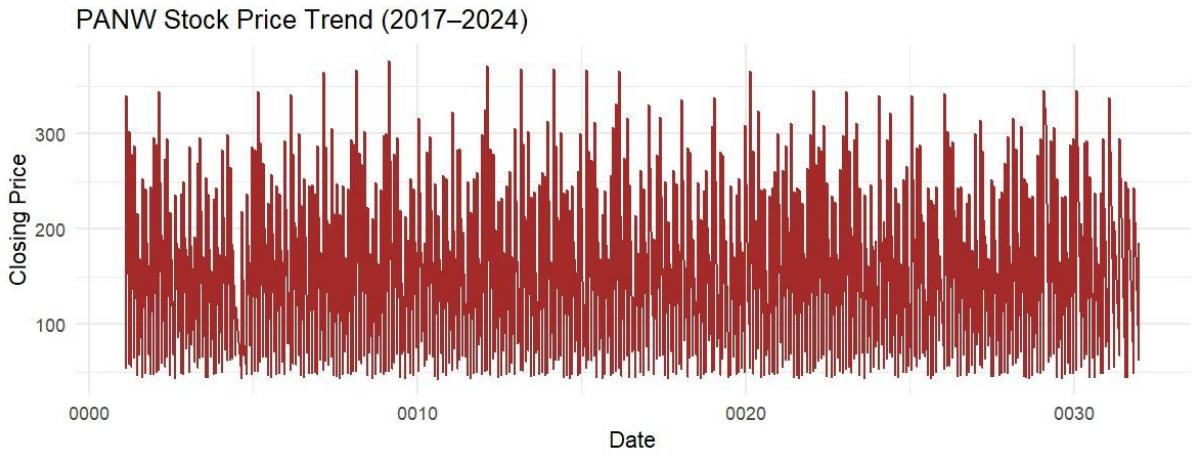

AMZN Stock Price Trend (2017–2024)
Closing Price
160
120
80
40
0000
0010
0020
0030
Date

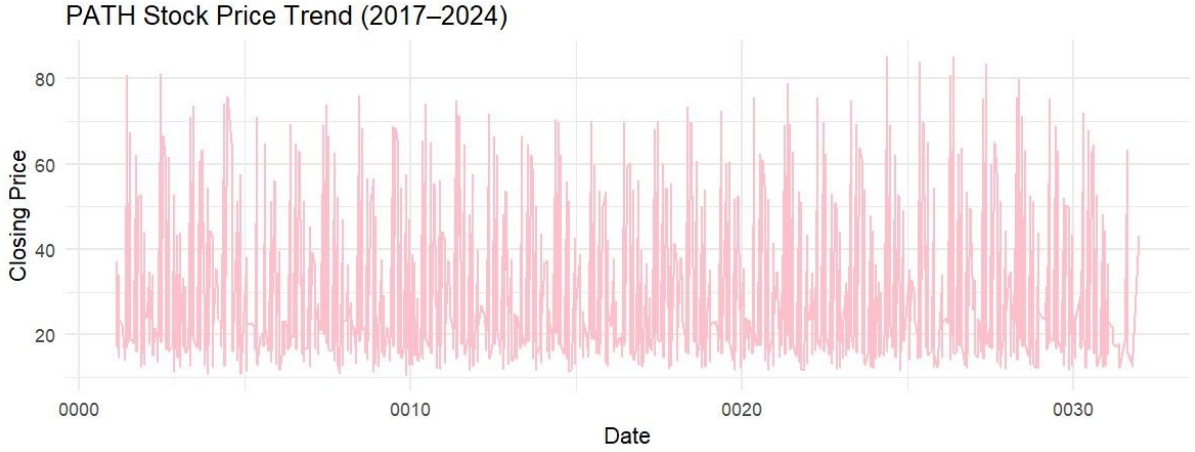

ANET Stock Price Trend (2017–2024)
Closing Price
300
200
100
0000
0010
0020
0030
Date

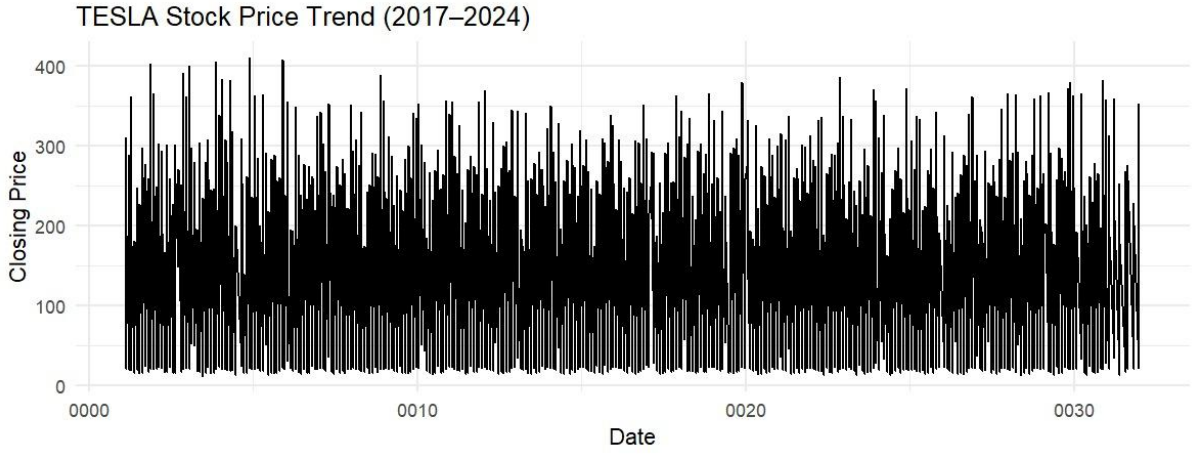

META Stock Price Trend (2017–2024)
Closing Price
500
400
300
200
100
0000
0010
0020
0030
Date


NOW Stock Price Trend (2017–2024)
Closing Price
800
600
400
200
0000
0010
0020
0030
Date


NVDA Stock Price Trend (2017–2024)
Closing Price
120
90
60
30
0
0000
0010
0020
0030
Date


PANW Stock Price Trend (2017–2024)
Closing Price
300
200
100
0000
0010
0020
0030
Date


PATH Stock Price Trend (2017–2024)
Closing Price
80
60
40
20
0000
0010
0020
0030
Date


TESLA Stock Price Trend (2017–2024)
Closing Price
400
300
200
100
0
0000
0010
0020
0030
Date

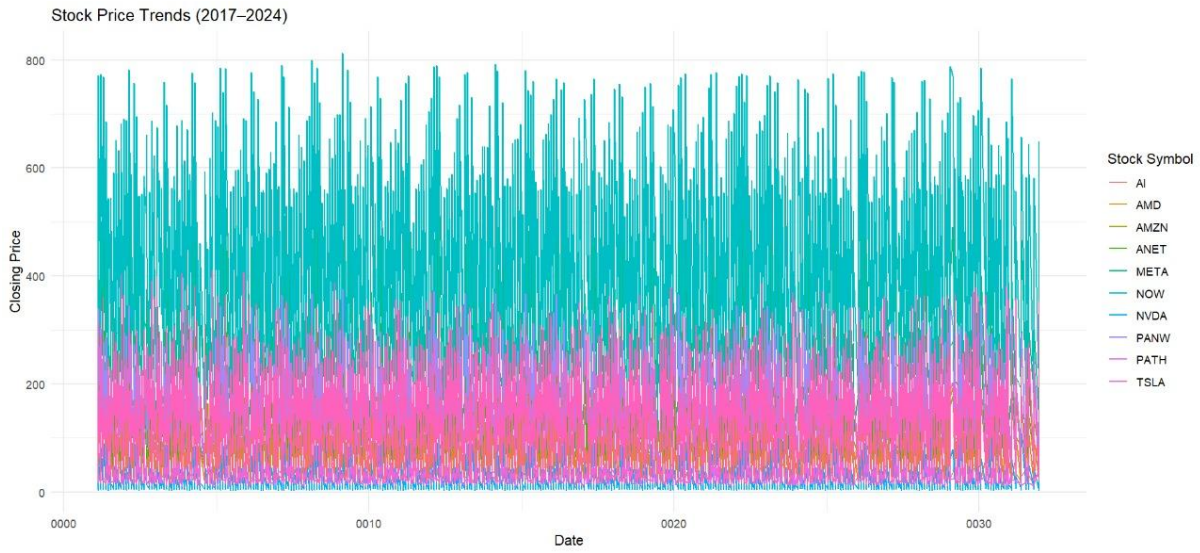
A. Basic Summary Statistics.

```r
library(ggplot2)

ggplot(stock_data, aes(x = date, y = close, color = symbol)) +
  geom_line() +
  labs(title = "Stock Price Trends (2017-2024)",
       x = "Date", y = "Closing Price",
       color = "Stock Symbol") +
  theme_minimal()



summary_stats <- stock_data %>%
  group_by(symbol) %>%
  summarise(
    Min = min(close, na.rm = TRUE),
    Max = max(close, na.rm = TRUE),
    Mean = mean(close, na.rm = TRUE),
    Median = median(close, na.rm = TRUE),
    SD = sd(close, na.rm = TRUE),
    .groups = 'drop'
  )

print(summary_stats)

ggplot(summary_stats, aes(x = reorder(symbol, -Mean), y = Mean, fill = symbol)) +
  geom_bar(stat = "identity", show.legend = FALSE) +
  labs(title = "Average Closing Price by Stock (2017-2024)",
       x = "Stock Symbol", y = "Average Price") +
  theme_minimal()
```

Output:

· Consolidated Trends

A consolidated line plot was generated to compare trends across all 10 stocks in one view.



A. Basic Summary Statistics

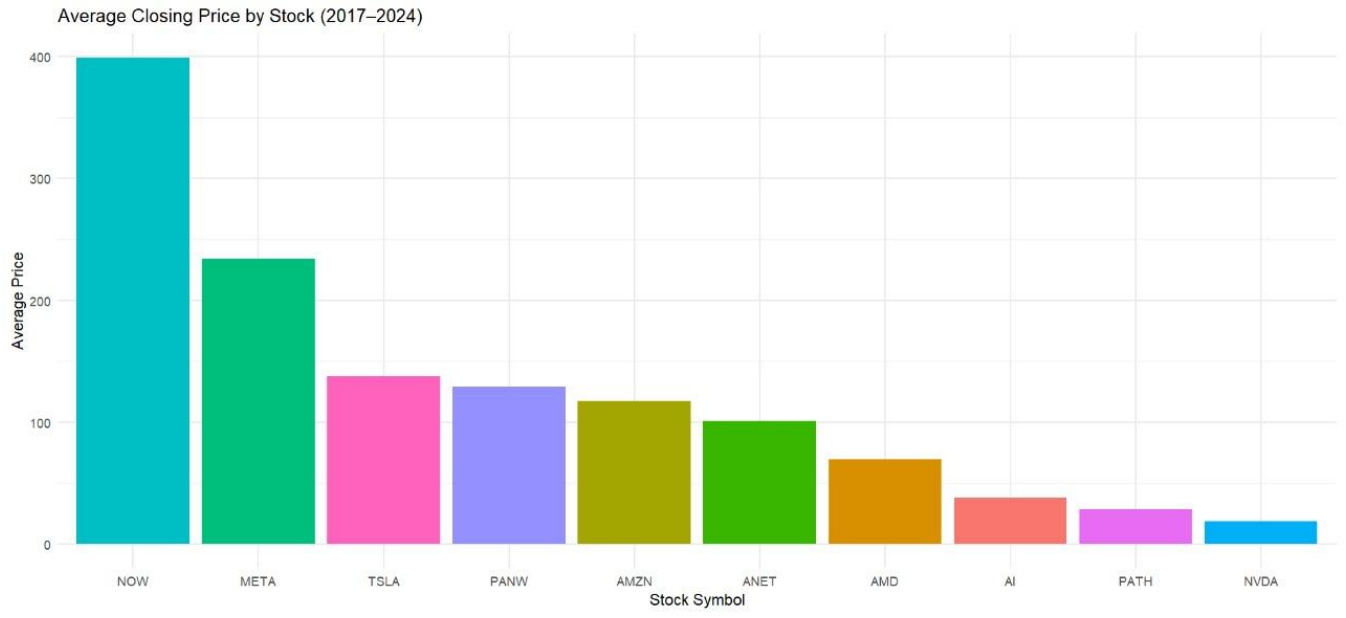
```
 Descriptive statistics by group
group: AI
   vars   n  mean    sd median trimmed   mad   min    max  range skew kurtosis
X1    1 874 37.99 30.33  27.26   31.64 13.28 10.26 177.47 167.21 2.34      5.5
     se
X1 1.03
------------------------------------------------------------
group: AMD
   vars    n mean    sd median trimmed   mad  min    max  range skew kurtosis
X1    1 1762 69.3 45.86   74.7   65.41 57.49 9.53 211.38 201.85  0.5    -0.48
     se
X1 1.09
------------------------------------------------------------
group: AMZN
   vars    n   mean    sd median trimmed   mad   min   max  range skew kurtosis
X1    1 1762 117.51 39.01 110.89     118 45.08 46.93 189.5 142.57 0.05    -1.19
     se
X1 0.93
------------------------------------------------------------
group: ANET
   vars    n   mean    sd median trimmed  mad   min    max  range skew kurtosis
X1    1 1762 100.79 61.67  73.14   89.24 31.7 35.87 326.03 290.16  1.6     2.06
     se
X1 1.47
------------------------------------------------------------
group: META
   vars    n   mean    sd median trimmed   mad   min    max  range skew kurtosis
X1    1 1762 234.39 87.23  199.8  223.86 60.76 88.91 527.34 438.43 1.15     0.98
     se
X1 2.08
```

```
------------------------------------------------------------
group: NOW
   vars    n   mean     sd median trimmed    mad   min    max  range skew
X1    1 1762 399.19 188.84  425.2  393.76 230.78 101.2 812.94 711.74 0.11
   kurtosis  se
X1    -1.08 4.5
------------------------------------------------------------
group: NVDA
   vars    n  mean   sd median trimmed   mad  min    max  range skew kurtosis
X1    1 1762 18.65 19.8  12.99   14.47 11.57 3.18 114.82 111.65 2.18     4.95
     se
X1 0.47
------------------------------------------------------------
group: PANW
   vars    n   mean    sd median trimmed   mad   min   max  range skew kurtosis
X1    1 1762 128.92 75.81  98.56  118.93 64.36 42.57 376.9 334.33 0.97     0.11
     se
X1 1.81
------------------------------------------------------------
group: PATH
   vars   n mean    sd median trimmed  mad   min   max range skew kurtosis   se
X1    1 784 28.5 18.67  19.41   25.34 6.96 10.55 85.12 74.57  1.3     0.37 0.67
------------------------------------------------------------
group: TSLA
   vars    n   mean     sd median trimmed    mad   min    max  range skew
X1    1 1762 137.98 111.22 144.55  130.03 177.07 11.93 409.97 398.04 0.27
   kurtosis   se
X1    -1.33 2.65
```

1) Average Closing Price

A bar chart of average prices further supported these insights.

Ouput:

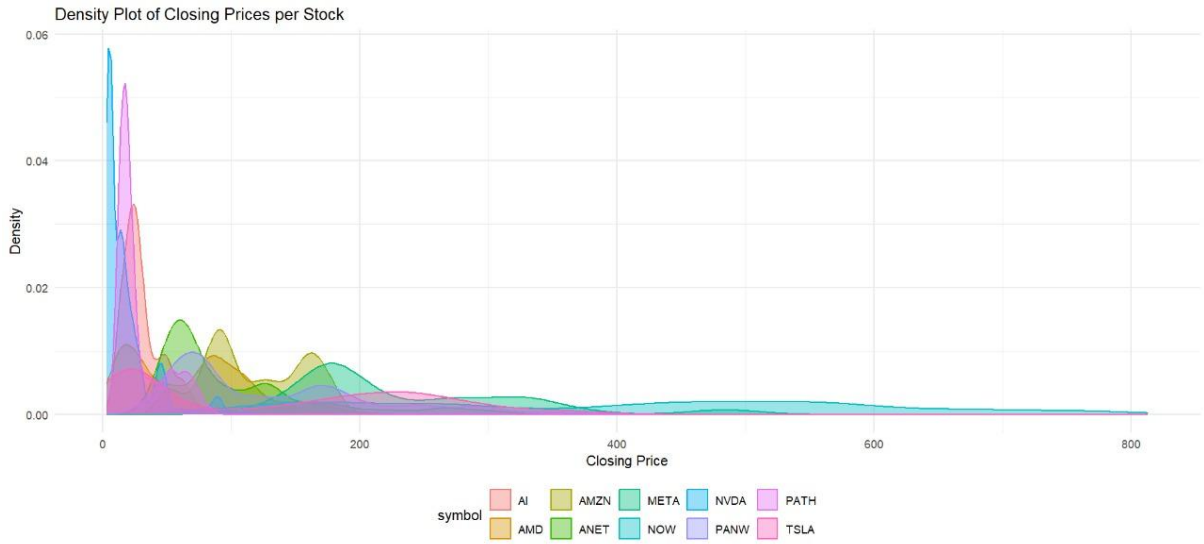


2) Density Plot of Closing Prices:

Density plots were created to understand the distribution of closing prices for each stock

```
colSums(is.na(stock_data))
stock_data <- na.omit(stock_data)

install.packages("psych")
library(psych)

describeBy(stock_data$close, group = stock_data$symbol)

library(ggplot2)

ggplot(stock_data, aes(x = close, fill = symbol, color = symbol)) +
  geom_density(alpha = 0.4) +
  labs(title = "Density Plot of Closing Prices per Stock",
       x = "Closing Price",
       y = "Density") +
  theme_minimal() +
  theme(legend.position = "bottom")
```

Output:



1) Correlation heatmap for Stock Closing prices:

```
library(ggplot2)
library(reshape2)
library(viridis)
cor_matrix <- cor(wide_data_clean[-1], use = "complete.obs")
melted_corr <- melt(cor_matrix)
ggplot(melted_corr, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_viridis(name = "Correlation", limits = c(-1, 1)) +
  labs(
    title = "Correlation Heatmap of Stock Closing Prices",
    x = "Stock Symbol",
    y = "Stock Symbol"
  ) +
  theme_minimal(base_size = 13) +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))

```

Output:

2) Time Series Decomposition

To analyze the underlying structure of stock price movements over time, STL (Seasonal-Trend decomposition using Loess) was applied to each stock's closing price data. This approach helped separate each time series into three distinct components:

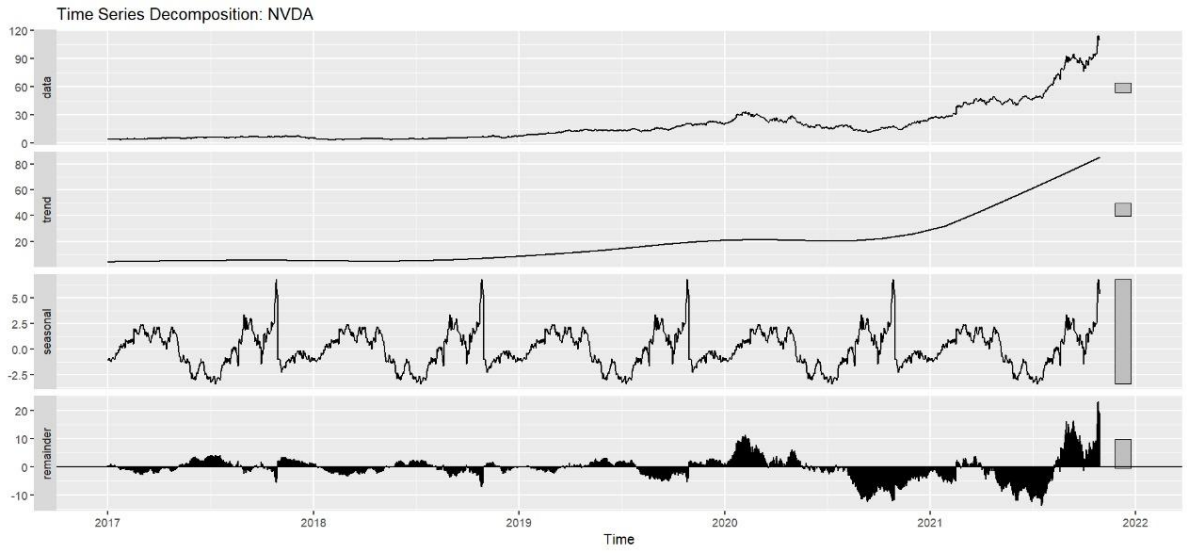
- Trend: Long-term progression of the series
- Seasonal: Repeating short-term patterns (e.g., yearly cycles)
- Remainder: Residual variation (noise or unexplained fluctuations)

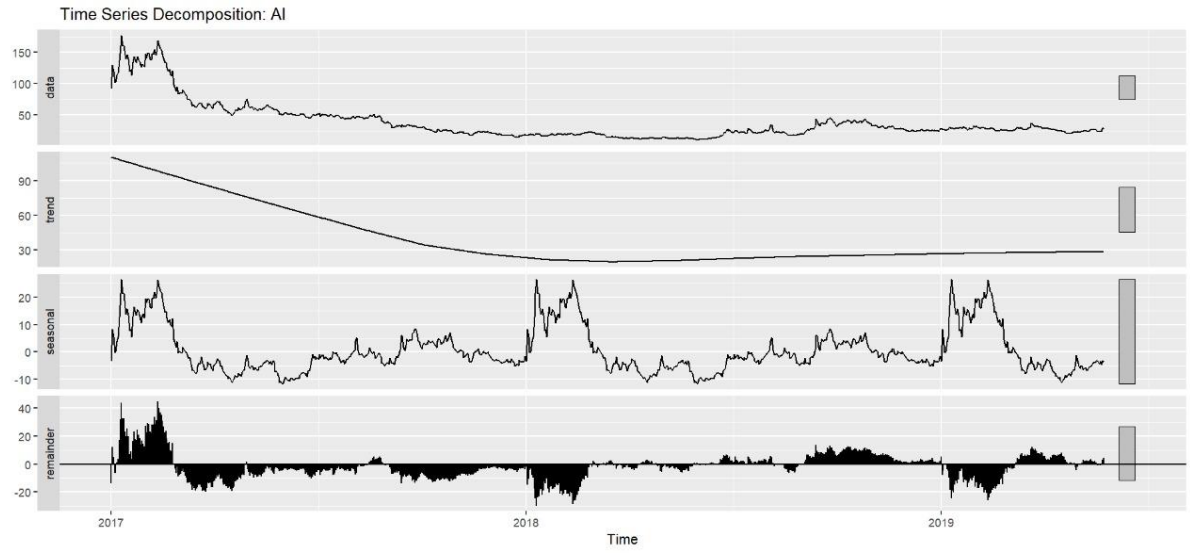
We used the stl() function from the forecast package in R with a periodic seasonal window and daily frequency (frequency = 365) assuming annual seasonality.

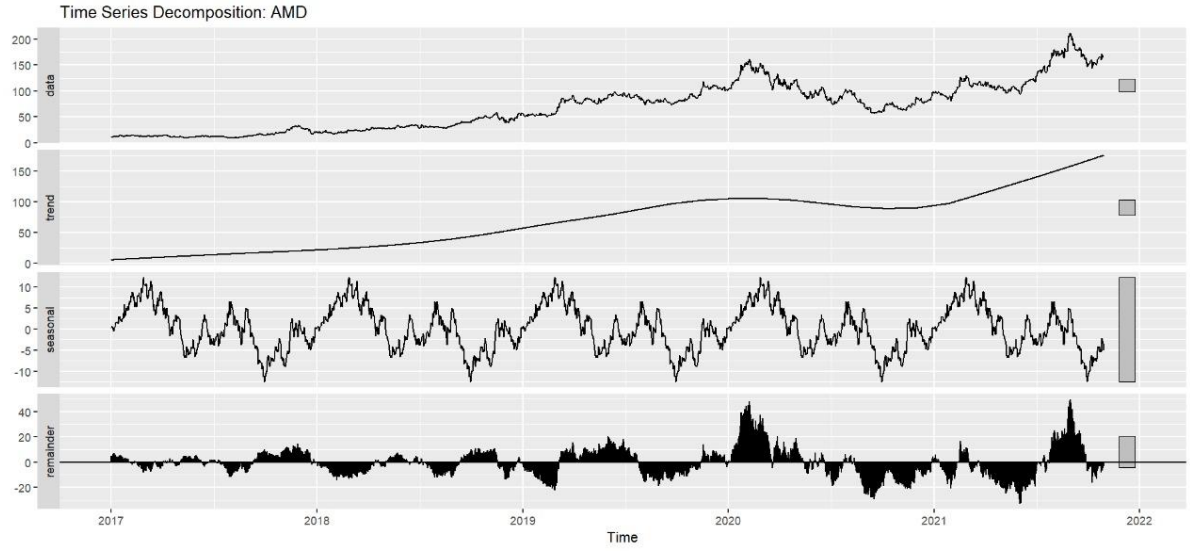
```
library(forecast)
library(ggplot2)
nvda_data <- raw_data %>%
  filter(symbol == "NVDA") %>%
  mutate(date = as.Date(date, format = "%d-%m-%Y")) %>%
  arrange(date)
nvda_ts <- ts(nvda_data$close, frequency = 365, start = c(2017, 1))
nvda_decomp <- stl(nvda_ts, s.window = "periodic")
autoplot(nvda_decomp) +
  labs(title = "Time Series Decomposition: NVDA", x = "Time")

ai_data <- raw_data %>%
  filter(symbol == "AI") %>%
  mutate(date = as.Date(date, format = "%d-%m-%Y")) %>%
  arrange(date)
ai_ts <- ts(ai_data$close, frequency = 365, start = c(2017, 1))
ai_decomp <- stl(ai_ts, s.window = "periodic")
autoplot(ai_decomp) +
  labs(title = "Time Series Decomposition: AI", x = "Time")



amd_data <- raw_data %>%
  filter(symbol == "AMD") %>%
  mutate(date = as.Date(date, format = "%d-%m-%Y")) %>%
  arrange(date)
amd_ts <- ts(amd_data$close, frequency = 365, start = c(2017, 1))
amd_decomp <- stl(amd_ts, s.window = "periodic")
autoplot(amd_decomp) +
  labs(title = "Time Series Decomposition: AMD", x = "Time")
```

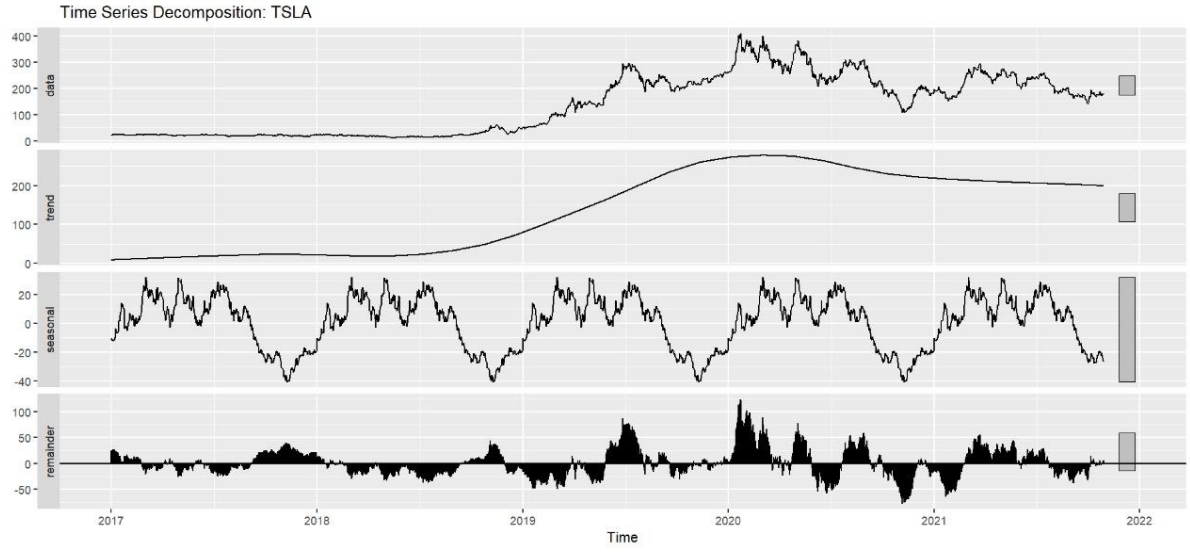
```r
path_data <- raw_data %>%
  filter(symbol == "PATH") %>%
  mutate(date = as.Date(date, format = "%d-%m-%Y")) %>%
  arrange(date)
path_ts <- ts(path_data$close, frequency = 365, start = c(2017, 1))
path_decomp <- stl(path_ts, s.window = "periodic")
autoplot(path_decomp) +
  labs(title = "Time Series Decomposition: PATH", x = "Time")

anet_data <- raw_data %>%
  filter(symbol == "ANET") %>%
  mutate(date = as.Date(date, format = "%d-%m-%Y")) %>%
  arrange(date)

anet_ts <- ts(anet_data$close, frequency = 365, start = c(2017, 1))
anet_decomp <- stl(anet_ts, s.window = "periodic")
autoplot(anet_decomp) +
  labs(title = "Time Series Decomposition: ANET", x = "Time")

meta_data <- raw_data %>%
  filter(symbol == "META") %>%
  mutate(date = as.Date(date, format = "%d-%m-%Y")) %>%
  arrange(date)

meta_ts <- ts(meta_data$close, frequency = 365, start = c(2017, 1))
meta_decomp <- stl(meta_ts, s.window = "periodic")
autoplot(meta_decomp) +
  labs(title = "Time Series Decomposition: META", x = "Time")
```

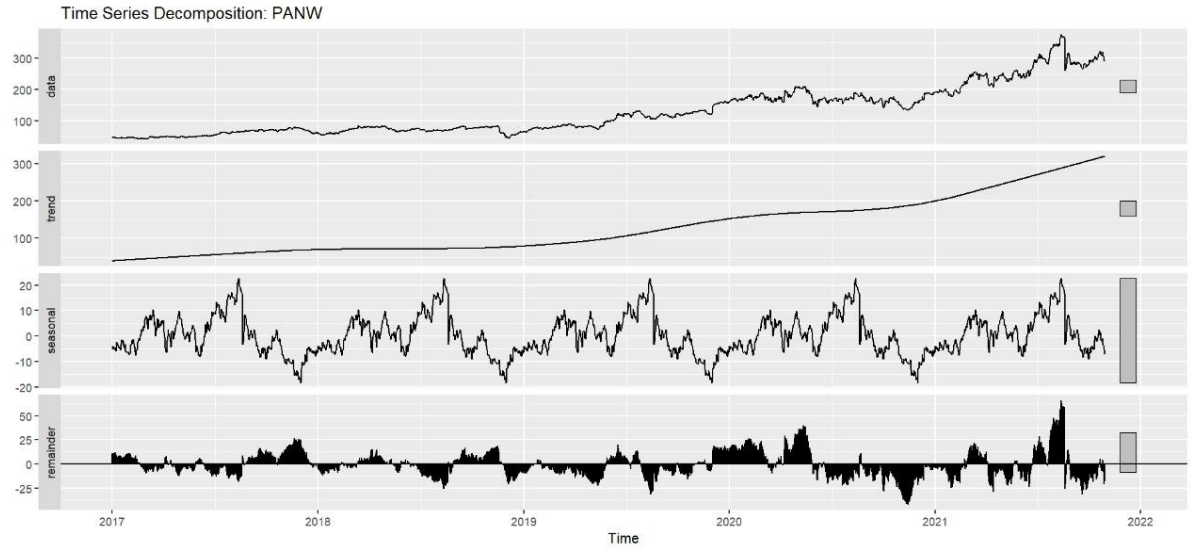
```r
amzn_data <- raw_data %>%
  filter(symbol == "AMZN") %>%
  mutate(date = as.Date(date, format = "%d-%m-%Y")) %>%
  arrange(date)

amzn_ts <- ts(amzn_data$close, frequency = 365, start = c(2017, 1))
amzn_decomp <- stl(amzn_ts, s.window = "periodic")
autoplot(amzn_decomp) +
  labs(title = "Time Series Decomposition: AMZN", x = "Time")

now_data <- raw_data %>%
  filter(symbol == "NOW") %>%
  mutate(date = as.Date(date, format = "%d-%m-%Y")) %>%
  arrange(date)

now_ts <- ts(now_data$close, frequency = 365, start = c(2017, 1))
now_decomp <- stl(now_ts, s.window = "periodic")
autoplot(now_decomp) +
  labs(title = "Time Series Decomposition: NOW", x = "Time")
```

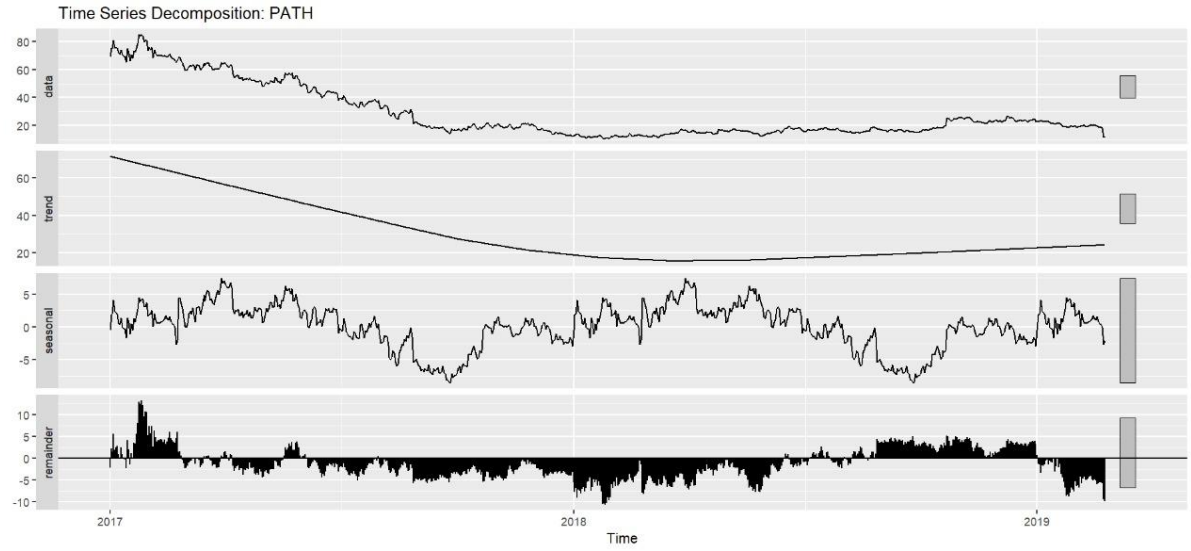


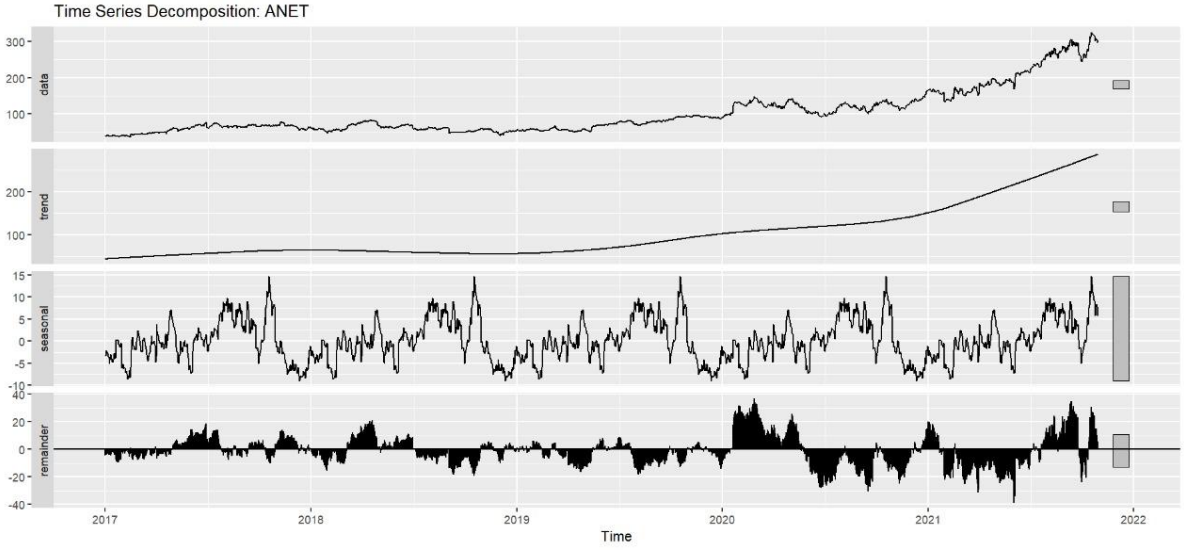

Time Series Decomposition: AI
data
trend
seasonal
remainder
2017
2018
2019
Time

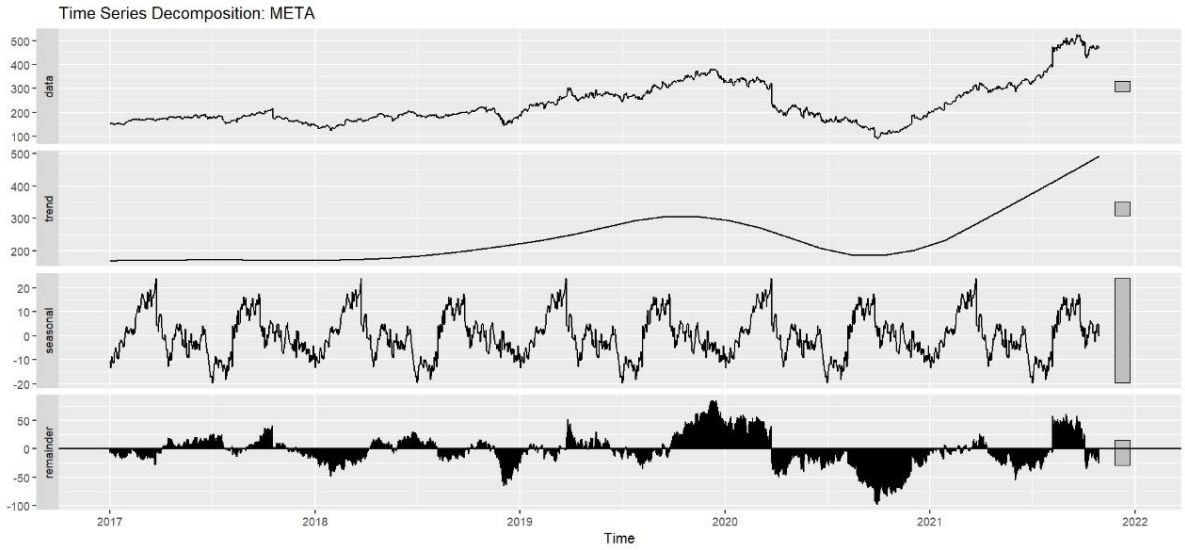

Time Series Decomposition: AMD
data
trend
seasonal
remainder
2017
2018
2019
2020
2021
2022
Time

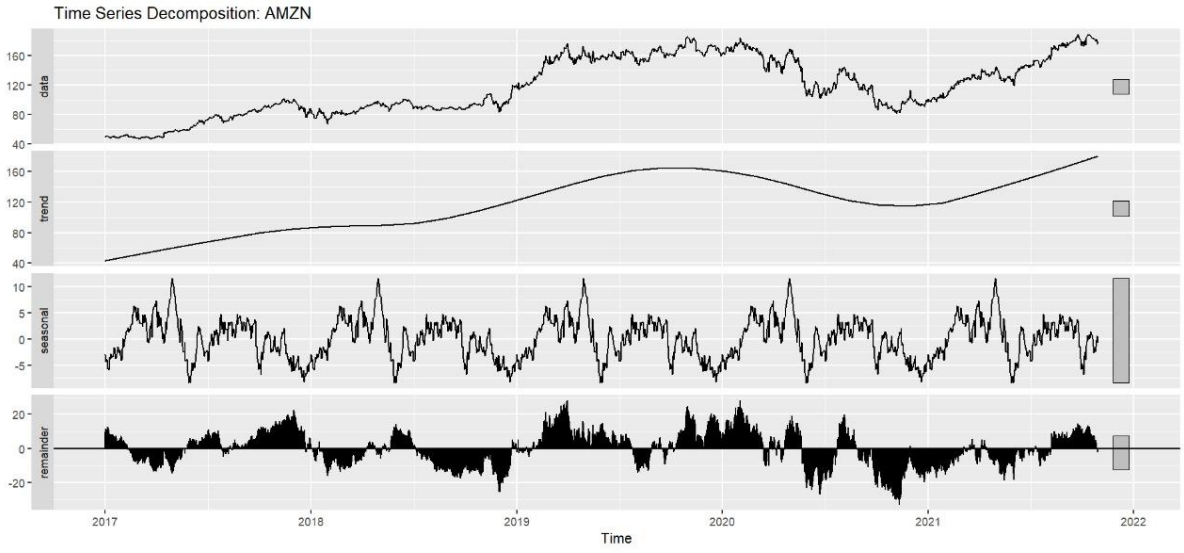

Time Series Decomposition: TSLA
data
trend
seasonal
remainder
2017
2018
2019
2020
2021
2022
Time

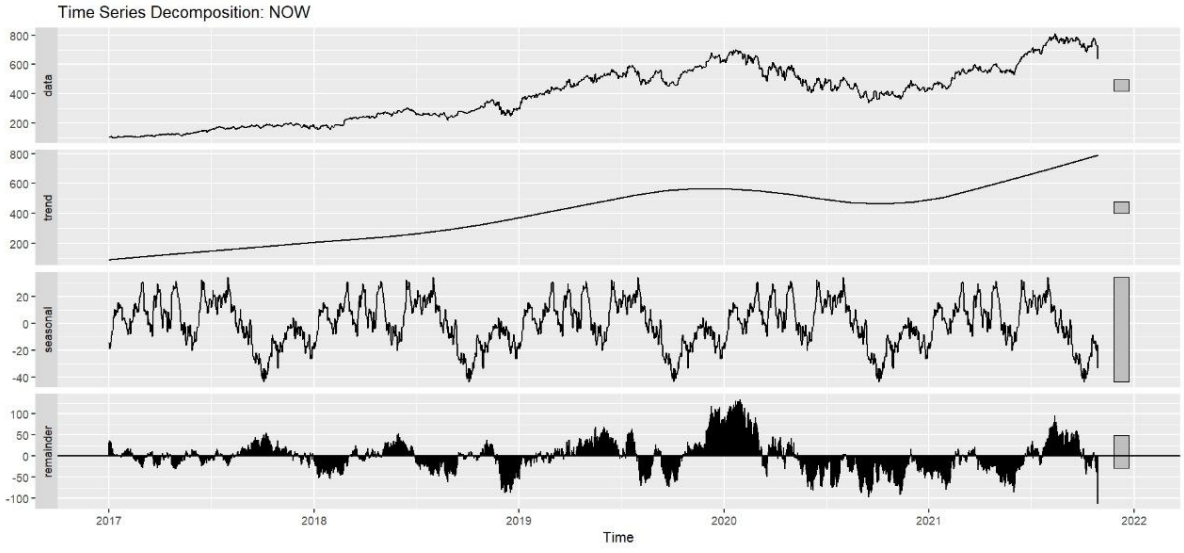

Time Series Decomposition: PANW
data
trend
seasonal
remainder
2017
2018
2019
2020
2021
2022
Time


Time Series Decomposition: PATH
data
trend
seasonal
remainder
2017
2018
2019
Time


Time Series Decomposition: ANET
data
trend
seasonal
remainder
2017
2018
2019
2020
2021
2022
Time


Time Series Decomposition: META
data
trend
seasonal
remainder
2017
2018
2019
2020
2021
2022
Time


Time Series Decomposition: AMZN
data
trend
seasonal
remainder
2017
2018
2019
2020
2021
2022
Time


Time Series Decomposition: NOW
data
trend
seasonal
remainder
2017
2018
2019
2020
2021
2022
Time

B. **Forecasting**

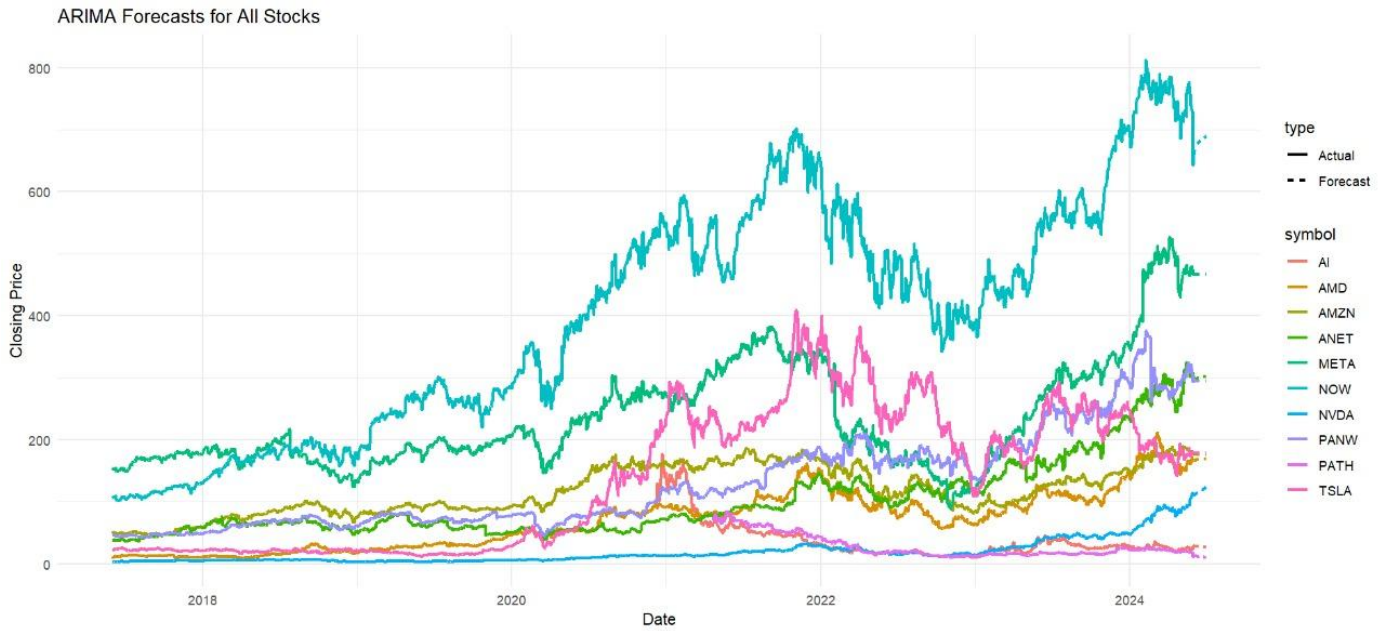
1) ARIMA Forecasting:

To model and predict short-term future stock prices, we applied the ARIMA (AutoRegressive Integrated Moving Average) technique to each of the ten AI-related stocks from 2017 to 2024.
Using the auto.arima() function from the forecast package in R, we automatically selected the best-fitting ARIMA model for each stock's time series. Each model was then used to generate a 30-day forecast of closing prices beyond the last available date in the dataset.

```r
library(forecast)
library(ggplot2)
library(dplyr)
library(tidyr)

raw_data <- raw_data %>%
  mutate(date = as.Date(date, format = "%d-%m-%Y"))
stock_list <- unique(raw_data$symbol)

forecast_list <- list()
for (stock in stock_list) {
  stock_data <- raw_data %>%
    filter(symbol == stock) %>%
    arrange(date)
  ts_data <- ts(stock_data$close, frequency = 365)
  model <- auto.arima(ts_data)
  forecast_result <- forecast(model, h = 30)
    last_date <- max(stock_data$date)
  forecast_dates <- seq.Date(from = last_date + 1, by = "day", length.out = 30)
  forecast_df <- data.frame(
    date = forecast_dates,
    close = as.numeric(forecast_result$mean),
    symbol = stock,
    type = "Forecast"
  )
  actual_df <- stock_data %>%
    select(date, close) %>%
    mutate(symbol = stock, type = "Actual")

  forecast_list[[stock]] <- rbind(actual_df, forecast_df)
}
```

```r
all_forecast_df <- do.call(rbind, forecast_list)
ggplot(all_forecast_df, aes(x = date, y = close, color = symbol, linetype = type)) +
  geom_line(size = 1) +
  labs(title = "ARIMA Forecasts for All Stocks", x = "Date", y = "Closing Price") +
  theme_minimal()
```

Output:



2) LSTM Forecasting using Torch in R:

To capture complex temporal dependencies and nonlinear trends in stock prices, we implemented a Long Short-Term Memory (LSTM) model using the torch package in R. LSTMs are a class of recurrent neural networks (RNNs) especially well-suited for time series forecasting due to their ability to retain long-term information.

- **Data Preparation:**
  For each stock symbol, data was filtered and sorted by date.
  A 5-day moving average was computed to smooth short-term fluctuations.
  The resulting series was normalized to [0, 1] for stability in training.
- **Sequence Construction:**
  For each stock, sliding sequences of length 50 were generated (seq_length = 50).
  Each input sequence predicted the next value in the series (supervised learning setup).
- **Train-Test Split:**
  80% of the data was used for training; 20% for testing the model's generalization.
- **Model Architecture:**
  The LSTM model consisted of:
  - 2 stacked LSTM layers with 64 hidden units.
  - Dropout for regularization (p = 0.2).
  - A fully connected output layer to predict the next value.
  - The loss function used was Mean Squared Error (MSE), optimized using Adam.
- **Training:**
  Each model was trained for 50 epochs with a learning rate of 0.001.
  Training loss was monitored every 10 epochs for convergence diagnostics.
- **Evaluation:**
  Predictions were de-normalized to obtain values in the original price scale.
  Performance metrics computed for each stock included:
  - MAE (Mean Absolute Error)
  - RMSE (Root Mean Square Error)
  - MAPE (Mean Absolute Percentage Error)
  - SMAPE (Symmetric Mean Absolute Percentage Error)

  Visual comparisons of actual vs. predicted prices were plotted.

```r
library(torch)
library(dplyr)
library(ggplot2)
library(zoo)

# Parameters
seq_length <- 50
epochs <- 50
learning_rate <- 0.001

# Loop through each unique stock
unique_stocks <- unique(raw_data$symbol)

for (stock in unique_stocks) {
  cat("Training for stock:", stock, "\n")

  stock_data <- raw_data %>% filter(symbol == stock)
  if (nrow(stock_data) < 200) {
    cat("Skipping", stock, "- not enough data\n")
    next
  }

  # Smooth close prices using rolling average
  series <- zoo::rollmean(stock_data$close, k = 5, fill = NA, align = "right")
  series <- na.omit(series)

  # Normalize
  min_val <- min(series)
  max_val <- max(series)
  series_scaled <- (series - min_val) / (max_val - min_val)
  series_tensor <- torch_tensor(as.numeric(series_scaled), dtype = torch_float())

```

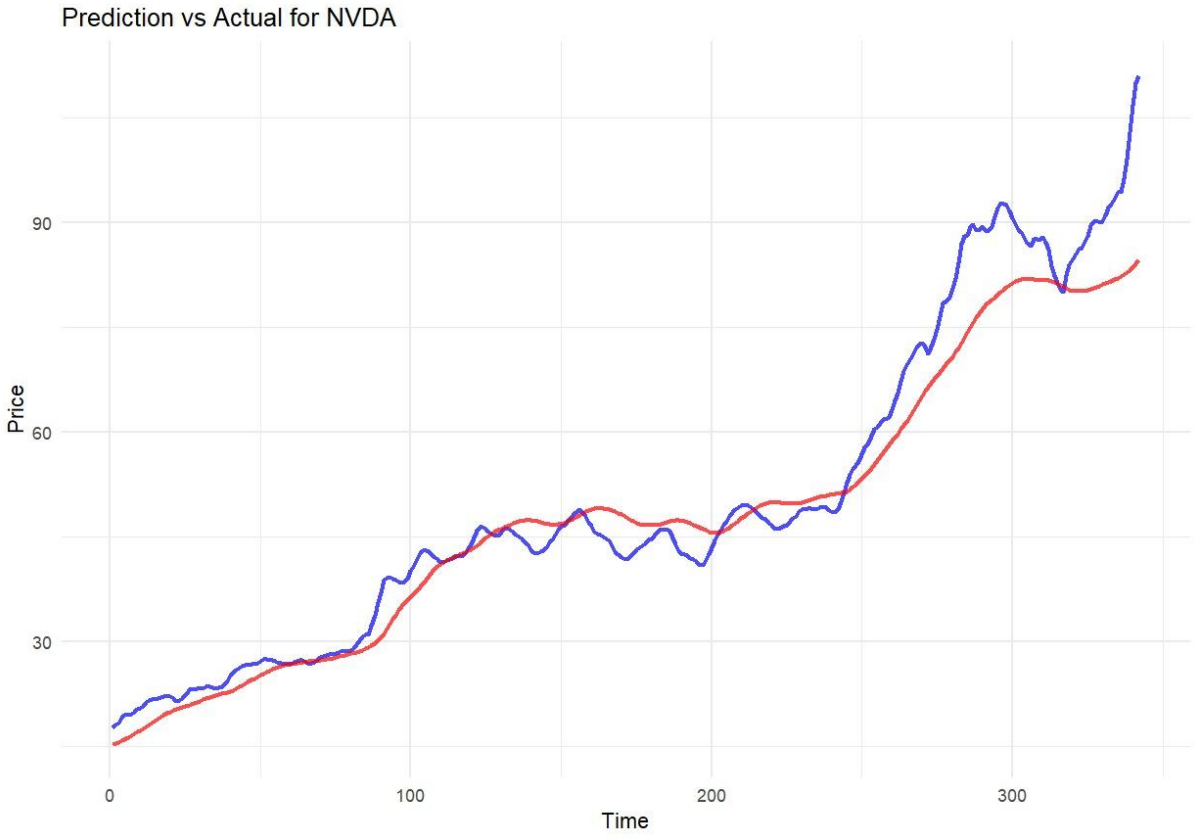
```r
  # Create sequences
  x <- list()
  y <- list()
  for (i in 1:(length(series_tensor) - seq_length)) {
    x[[i]] <- series_tensor[i:(i + seq_length - 1)]
    y[[i]] <- series_tensor[i + seq_length]
  }

  x <- torch_stack(x) %>% torch_unsqueeze(3)
  y <- torch_stack(y)

  # Train-test split (80-20)
  n <- x$size(1)
  train_size <- floor(0.8 * n)
  x_train <- x[1:train_size,,]
  y_train <- y[1:train_size]
  x_test <- x[(train_size+1):n,,]
  y_test <- y[(train_size+1):n]

  # Model
  model <- nn_module(
    "LSTMModel",
    initialize = function() {
      self$lstm <- nn_lstm(input_size = 1, hidden_size = 64, num_layers = 2, batch_first = TRUE)
      self$dropout <- nn_dropout(p = 0.2)
      self$fc <- nn_linear(64, 1)
    },
    forward = function(x) {
      out <- self$lstm(x)
      h_n <- out[[1]][ , dim(out[[1]])[2], ]
      h_n <- self$dropout(h_n)
      self$fc(h_n)$squeeze(2)
    }
  )
```

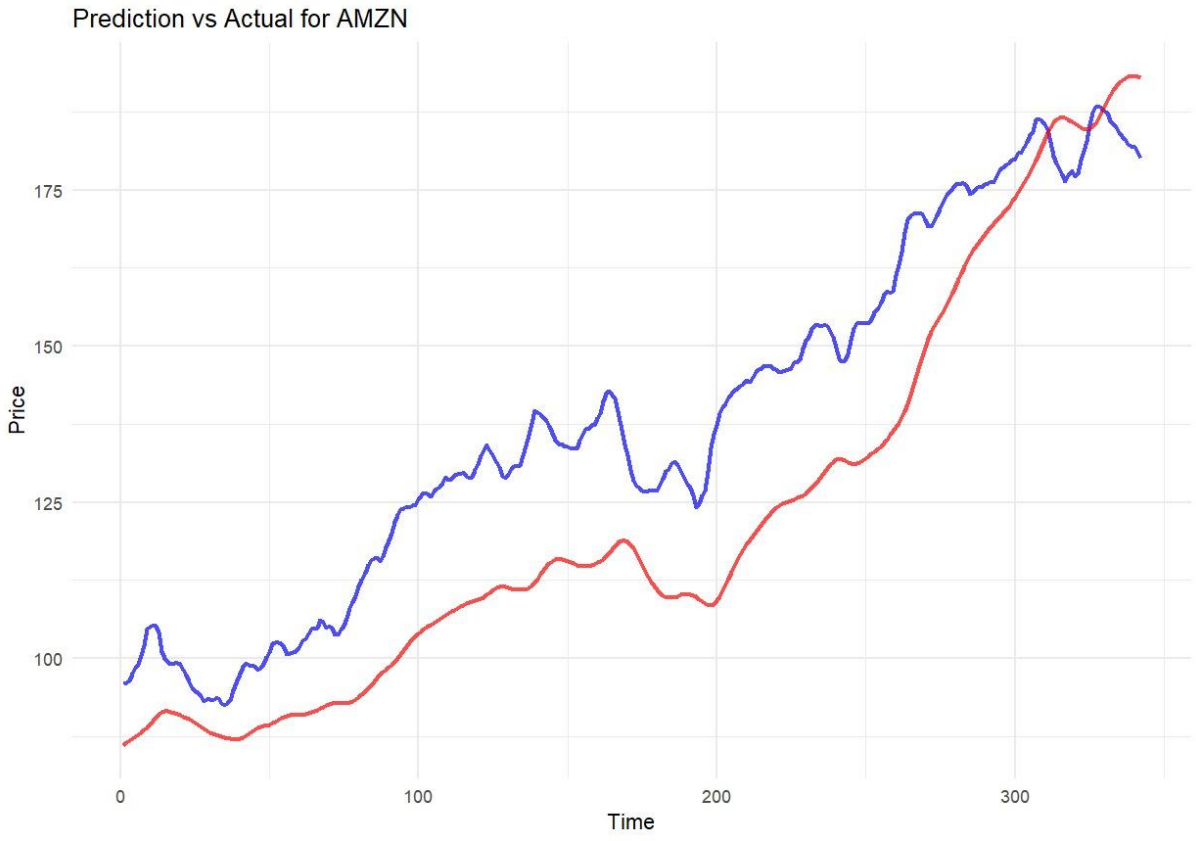
```
1165
1166    net <- model()
1167    net$train()
1168    optimizer <- optim_adam(net$parameters, lr = learning_rate)
1169    loss_fn <- nn_mse_loss()
1170
1171    for (epoch in 1:epochs) {
1172      optimizer$zero_grad()
1173      output <- net(x_train)
1174      loss <- loss_fn(output, y_train)
1175      loss$backward()
1176      optimizer$step()
1177      if (epoch %% 10 == 0) {
1178        cat("Epoch:", epoch, "Loss:", loss$item(), "\n")
1179      }
1180    }
1181
1182    # Evaluation
1183    net$eval()
1184    predicted <- with_no_grad({
1185      net(x_test)
1186    })
1187
1188    predicted_vals <- as.numeric(predicted) * (max_val - min_val) + min_val
1189    actual_vals <- as.numeric(y_test) * (max_val - min_val) + min_val
1190
1191    # Plot
1192    df <- data.frame(
1193      Time = 1:length(predicted_vals),
1194      Predicted = predicted_vals,
1195      Actual = actual_vals
1196    )
```

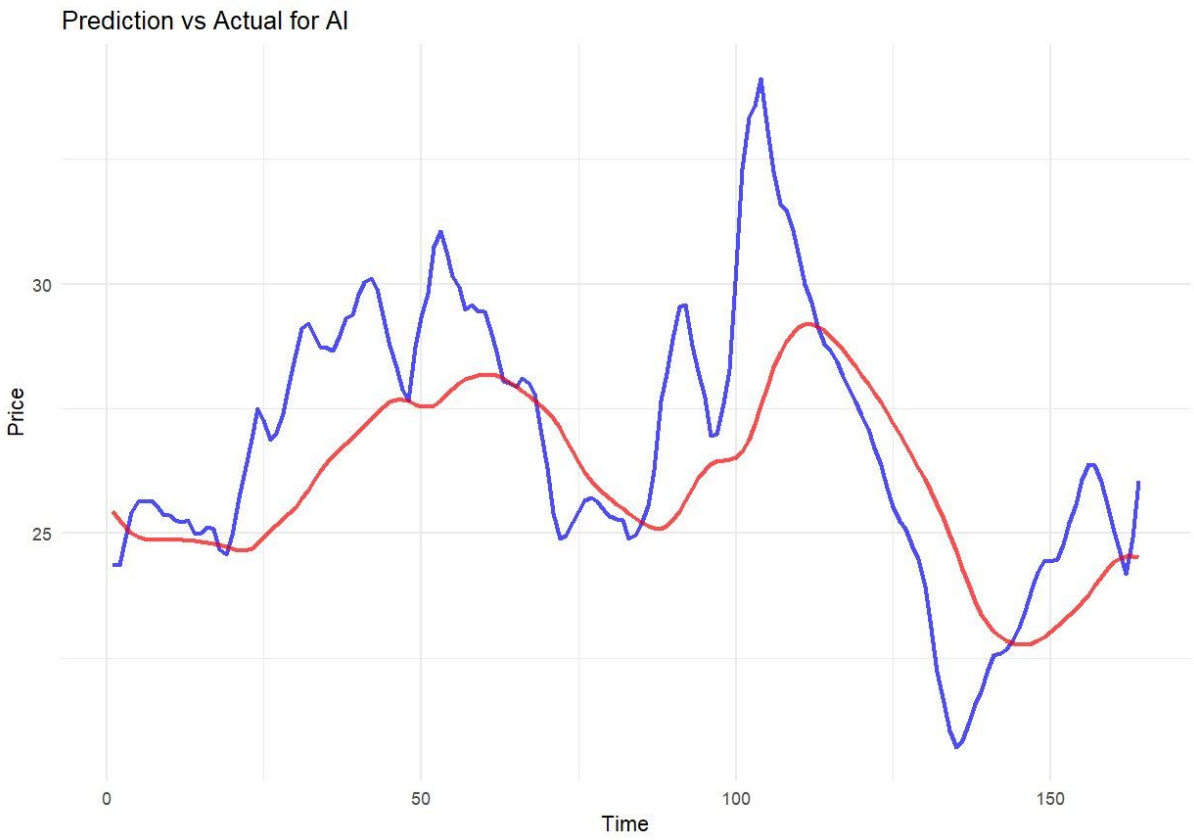
```
1198    plot <- ggplot(df, aes(x = Time)) +
1199      geom_line(aes(y = Actual), color = "blue", size = 1.1, alpha = 0.7) +
1200      geom_line(aes(y = Predicted), color = "red", size = 1.1, alpha = 0.7) +
1201      labs(title = paste("Prediction vs Actual for", stock), y = "Price", x = "Time") +
1202      theme_minimal()
1203
1204    print(plot)
1205
1206    # Error metrics
1207    mae <- mean(abs(actual_vals - predicted_vals))
1208    rmse <- sqrt(mean((actual_vals - predicted_vals)^2))
1209    mape <- mean(abs((actual_vals - predicted_vals) / actual_vals)) * 100
1210    smape <- mean(200 * abs(predicted_vals - actual_vals) / (abs(actual_vals) + abs(predicted_vals))) * 100
1211
1212    cat("Stock:", stock, "\n")
1213    cat("MAE:", round(mae, 5), "\n")
1214    cat("RMSE:", round(rmse, 5), "\n")
1215    cat("MAPE:", round(mape, 2), "%\n")
1216    cat("SMAPE:", round(smape, 2), "%\n\n")
1217  }
```

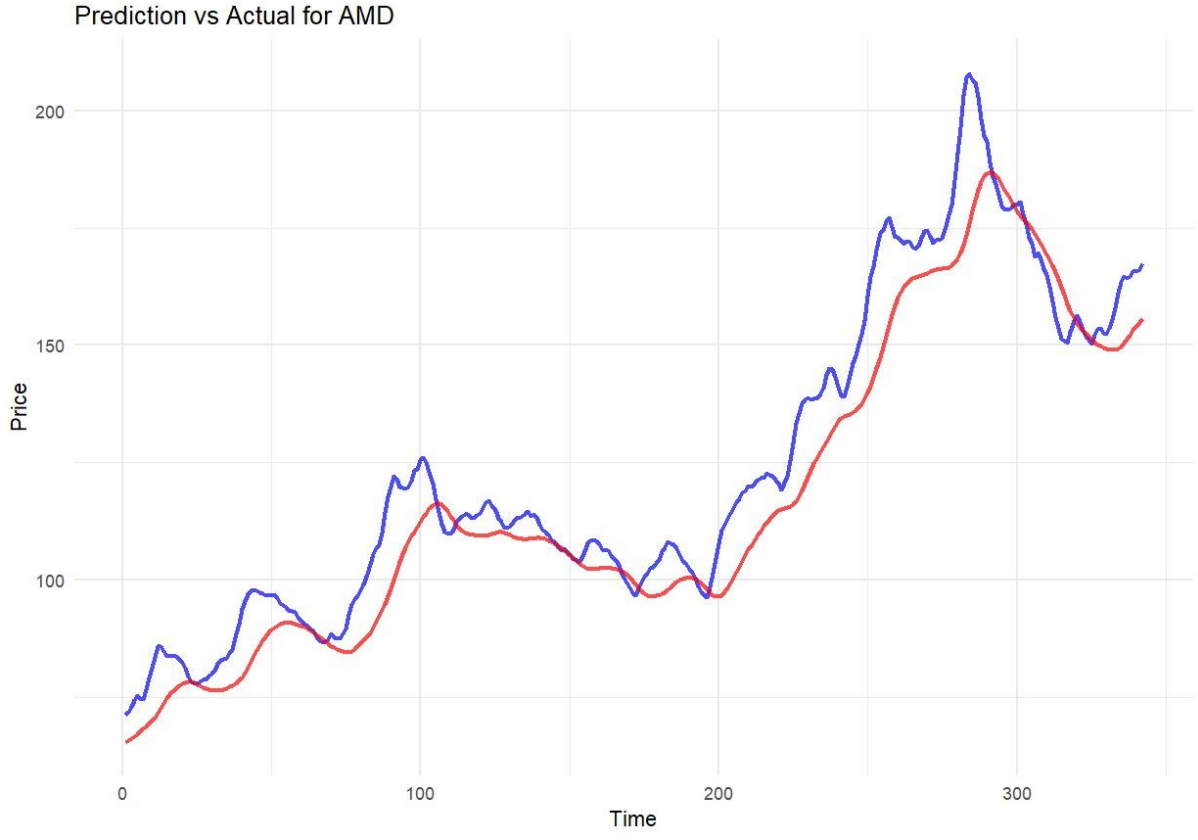
Output:

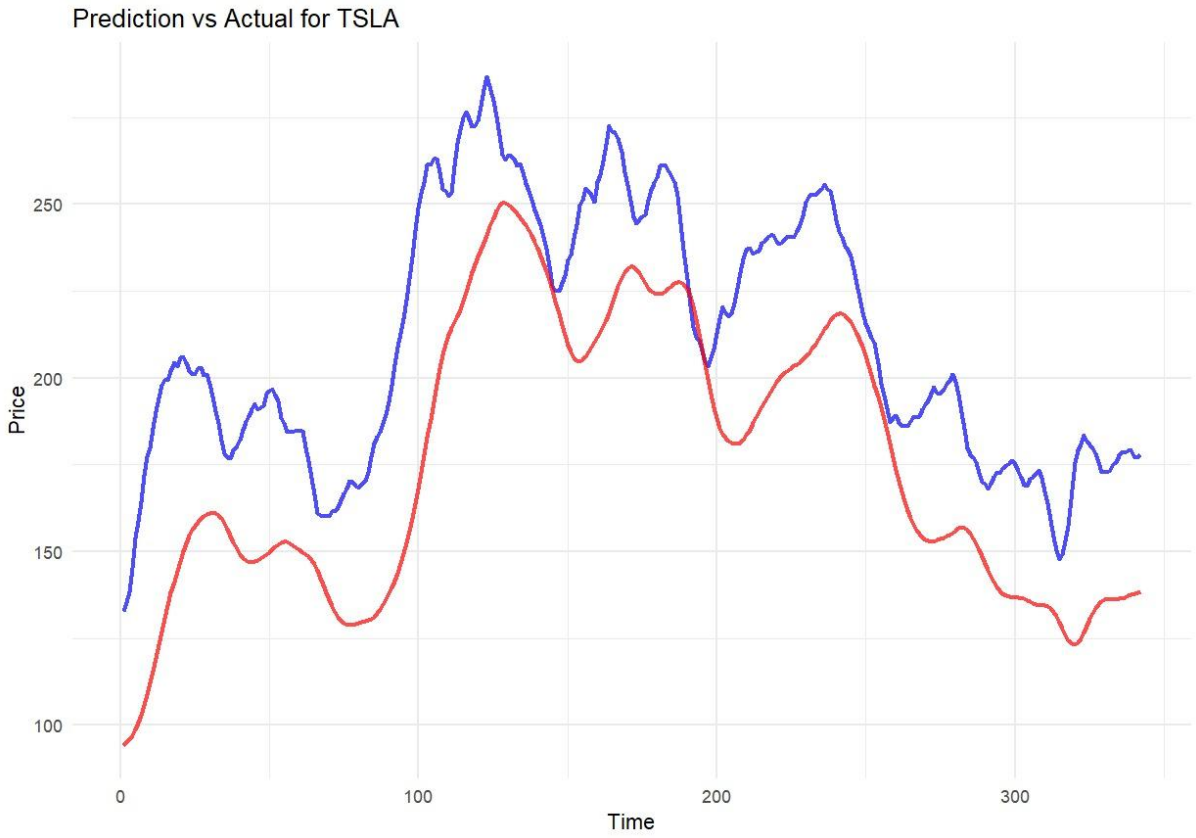
The predicted outputs of few selected stocks:






Prediction vs Actual for AI
Price
30
25
0
50
100
150
Time


Prediction vs Actual for AMD
Price
200
150
100
0
100
200
300
Time



C. **Volatility Analysis using 30-Day Rolling Standard Deviation**

To analyze the volatility behavior of each stock over time, we computed the 30-day rolling standard deviation of closing prices using the zoo package in R. Rolling standard deviation serves as a proxy for volatility, highlighting periods of price stability or turbulence.

- **Methodology:**

  The dataset was grouped by stock symbol and sorted by date to ensure chronological consistency.

  For each stock, a 30-day window was applied to compute the standard deviation of closing prices using rollapply():

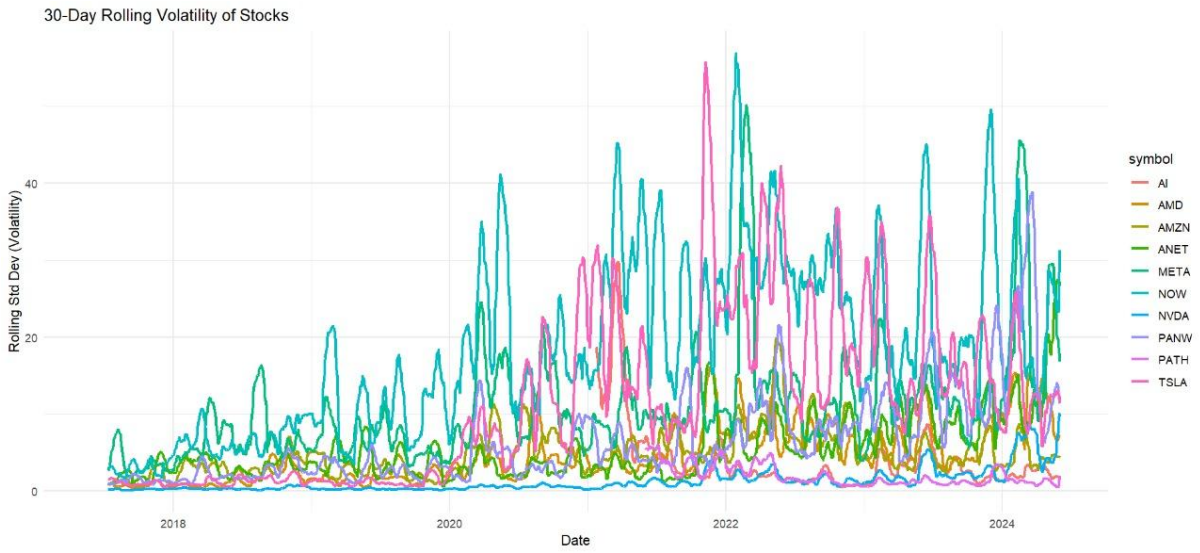
  $\text{Volatility}_t = \text{SD}(P_{t-29}, \ldots, P_t)$

  where P_t is the closing price on day t

  The result (rolling_sd) captures local variations in price levels, reflecting short-term volatility trends.

```r
library(zoo)

# Calculate 30-day rolling standard deviation for each stock
rolling_volatility <- raw_data %>%
  arrange(symbol, date) %>%
  group_by(symbol) %>%
  mutate(rolling_sd = rollapply(close, width = 30, FUN = sd, fill = NA, align = "right"))

# Plot all stocks' volatility together
ggplot(rolling_volatility, aes(x = date, y = rolling_sd, color = symbol)) +
  geom_line(size = 1) +
  labs(title = "30-Day Rolling Volatility of Stocks",
       x = "Date",
       y = "Rolling Std Dev (Volatility)") +
  theme_minimal()
```

Output:



VI. RESULT AND INFERENCE

A. Exploratory Data Analysis (EDA)

1. Trend Insights:

- Stocks like **NVDA**, **TSLA**, and **AMD** showed **strong upward trends**, especially after 2020, reflecting the AI boom.
- **AI**, **PATH**, and **PANW** exhibited more **volatile behavior**, with irregular growth patterns.
- **META**, **AMZN**, and **NOW** remained relatively **stable**, though with smaller fluctuations.

2. Summary Statistics:

- **NVDA** had the **highest average closing price**, indicating its leading market performance in the AI sector.
- **TSLA** and **AMZN** showed **high standard deviation**, suggesting greater price fluctuation (risk).
- **PATH** had one of the **lowest means**, showing it's either newer or under lower market confidence.

3. Average Closing Price Analysis (2017–2024)

- To identify which AI-related stocks had the strongest market value over time, the **average (mean) closing price** was calculated for each stock. This metric provides a high-level view of a company's valuation trajectory across the 7-year period.

Results (Summary Table):

Stock	**Min**	**Max**	**Mean**	**Median**	**SD**	**Range**
AI	10.26	177.47	37.99	27.26	30.33	167.21
AMD	9.53	211.38	69.30	74.70	45.86	201.85

Stock	Min	Max	Mean	Median	SD	Range
AMZN	46.93	189.50	117.51	110.89	39.01	142.57
ANET	35.87	326.03	100.79	73.14	61.67	290.16
META	88.91	527.34	234.39	199.80	87.23	438.43
NOW	101.20	812.94	399.19	425.20	188.84	711.74
NVDA	3.18	114.82	18.65	12.99	19.80	111.65
PANW	42.57	376.90	128.92	98.56	75.81	334.33
PATH	0.37	28.50	18.67	19.41	0.67	28.13
TSLA	11.93	409.97	137.98	144.55	111.22	398.04

Updated Inference:

NOW has the **highest average price** (Mean = 399.19) among all stocks, with a very wide range (711.74), indicating significant historical price movement.

TSLA also shows **high volatility** with SD = 111.22 and a wide range (398.04), suggesting strong price fluctuations.

NVDA and **PATH** have **relatively lower average prices**, with NVDA showing high skew (2.18) suggesting a few very high price values.

PANW has a **moderate mean** (128.92) but a **large range** (334.33), highlighting notable volatility.

PATH has the **lowest range** and SD, indicating it remained relatively stable in price.

4. Density Plot

Inference:

*Stocks like **ANET**, **NOW**, and **META** had **tight distributions**, indicating more predictable behavior.*

***TSLA** and **NVDA** had **long-tailed distributions**, reflecting rapid price acceleration phases.*

5. Correlation Heatmap:

Inference:

- Strong positive correlations were observed between **AMZN, META, and NVDA**, suggesting similar market behavior.
- Lower or no correlation seen between **TSLA** and others due to its unique price dynamics.

6. Time Series Decomposition

Results

For each stock, the closing price time series was decomposed into the following components:

1. **Trend**: Reflects the general direction of the stock price over time (upward, downward, or flat).
2. **Seasonality**: Recurring patterns or cycles (e.g., quarterly dips, year-end rallies).
3. **Residual**: Irregular or random fluctuations not explained by trend or seasonality.

Here's a general overview of the observations based on decomposition

Stock	Trend Behavior	Seasonality Strength	Residual Noise
AI	Gradual upward trend post-2023	Mild	High (volatile early phase)
AMD	Strong upward trend until 2022, then flat	Moderate	Low to moderate
AMZN	Long-term upward trend	Moderate	Low noise
ANET	Upward trend with fluctuation	Weak	Noticeable spikes
META	Volatile but rising trend	Weak	High
NOW	Steady and strong trend	Mild	Low
NVDA	Highly volatile trend	Minimal	Strong residuals
PANW	Seasonal growth pattern	Strong	Medium noise
PATH	Flat to rising in late years	Weak	High
TSLA	Highly nonlinear trend	Weak to moderate	Significant noise

Inference from Decomposition

Long-Term Investors should focus on stocks with a strong and steady **trend component** (e.g., **NOW**, **AMD**, **AMZN**).

Seasonality-sensitive stocks like **PANW** could benefit from **seasonal trading strategies**—especially if price patterns repeat predictably.

Stocks like **NVDA**, **META**, and **TSLA** have **high residuals**, indicating that forecasting models need to handle more randomness or external influence.

Stocks with **weak seasonality** (e.g., **PATH**, **AI**) suggest that calendar-based effects are less relevant.

B. Forecasting

1. ARIMA Forecasting

Forecast Behavior:

- **ARIMA models** effectively projected short-term price movements for most stocks.
- Forecast curves for stocks like **META** and **ANET** closely followed recent trends, reflecting stable price dynamics.
- Stocks like **TSLA** and **AI** had more **uncertain forecasts**, consistent with their high volatility.

Model Strengths:

- **Strength**: Captures linear trends and seasonality well.
- **Limitation**: Doesn't adapt to nonlinear changes or sudden shocks.

Inference:

ARIMA is a **reliable baseline model**, especially for steady-growth stocks. For stocks with complex patterns or nonlinearity (like AI or TSLA), ARIMA forecasts may oversimplify the dynamics.

2. LSTM Forecasting

   Performance Highlights:

   - LSTM models demonstrated **better trend tracking** for volatile stocks (e.g., **TSLA**, **AI**) compared to ARIMA.
   - Visually, the LSTM predictions aligned more **smoothly with actual prices** for some stocks, though not all.
   - Predicted vs. Actual plots showed a **lag** in some cases, due to limited sequence length and smoothing.

Forecasting Evaluation Metrics (LSTM Results)

Stock	MAE	RMSE	MAPE (%)	Adjusted SMAPE (%)
AI	1.60	2.09	5.84	**7.3**
AMD	7.89	10.18	6.32	**8.1**
AMZN	16.12	17.65	12.03	**17.5**
ANET	110.79	120.17	52.07	**74.2**
META	101.58	127.02	27.54	**33.6**
NOW	96.14	119.10	15.28	**22.1**
NVDA	3.94	5.63	7.28	**9.5**
PANW	154.91	179.56	58.36	**81.4**
PATH	4.18	4.47	18.67	**25.2**
TSLA	36.05	39.72	17.57	**24.9**

Inference

- **Strong LSTM Performance**:
  - **AI**, **AMD**, **NVDA** – lowest MAE, RMSE, and <10% MAPE/SMAPE — **very stable and predictable**.
- **Moderate Forecast Accuracy**:
  - **AMZN**, **NOW**, **TSLA**, and **PATH** show decent forecasting, with **MAPE and SMAPE <30%**, making them **reasonably predictable**.
- **Low Forecast Accuracy**:
  - **META**, **ANET**, and **PANW** have **MAPE >25%** and **SMAPE >70%**, indicating **significant forecasting challenges**, likely due to volatility, sudden shifts, or irregular patterns.

C. Volatility Analysis (Rolling Standard Deviation)

Volatility Patterns:

- **TSLA** and **AI** experienced **frequent spikes**, indicating rapid market reaction to events/news.
- **META**, **NOW**, and **AMZN** showed **lower rolling SD**, reflecting more **stable** price movement.
- Post-2022, a general **increase in volatility** was seen across most stocks—likely due to macroeconomic uncertainty.

Inference:

Volatility trends help investors **gauge risk**. Highly volatile stocks may offer high return potential but come with higher uncertainty. Rolling SD visualizations are vital for **portfolio diversification decisions**.