Stock Market Complete Project

Sneha Baddi Jituri

2024-12-14

```
#install.packages("rmarkdown")
#install.packages("tinytex")
tinytex::install_tinytex(force = TRUE)
```

```
## tlmgr install tlgpg
```

```
## tlmgr update --self
```

```
## tlmgr install tlgpg
```

```
## tlmgr --repository http://www.preining.info/tlgpg/ install tlgpg
```

```
## tlmgr option repository "https://mirrors.ibiblio.org/pub/mirrors/CTAN/systems/texlive/tlnet"
```

```
## tlmgr update --list
```

```
tinytex::is_tinytex()
```

```
## [1] TRUE
```

```
tinytex::tinytex_root()
```

```
## [1] "C:\\Users\\Kapil\\AppData\\Roaming\\TinyTeX"
```

```
#install.packages("webshot")
webshot::install_phantomjs()
```

```
## It seems that the version of ‘phantomjs‘ installed is greater than or equal to the requested version
```

Stock Market Prediction Analysis

Defining the Project Scope:

Develop and evaluate machine learning models to predict stock prices and returns for major tech companies using technical indicators, market data, and macroeconomic factors, focusing on both individual stocks and portfolio-level analysis.

Problem Statement

Stock market prediction is complex due to multiple influencing factors, making it difficult for investors to make informed decisions without comprehensive analysis of technical indicators, market trends, and macroeconomic factors.

Project Objectives

- Analyze historical stock data and create predictive models
- Evaluate impact of technical and macroeconomic indicators
- Develop portfolio-level prediction strategy
- Create actionable insights for investment decisions

Hypothesis

- H1: Predicting the Close Prices for Stocks
- H2: Predicting the Returns for Stocks

Risks & Limitations

1. Market volatility and unpredictable events can affect model accuracy
2. Past performance may not indicate future results
3. Macroeconomic factors can have delayed or unexpected impacts

Data Sources:

- Yahoo Finance: Stock price data for AAPL, AMZN, MSFT, NVDA
- FRED Database: Macroeconomic indicators (Interest rates, Inflation, GDP)
- Market Indices: S&P500 and NASDAQ data
- Event Timeline: Major events like COVID-19, Russia-Ukraine War

Data Preparation

```r
#install.packages(c("quantmod", "tidyquant", "dplyr", "xts", "corrplot", "tidyr", "randomForest"))
library(quantmod)
```

Installing and loading all the packages

```
## Warning: package 'quantmod' was built under R version 4.4.2

## Loading required package: xts

## Warning: package 'xts' was built under R version 4.4.2

## Loading required package: zoo

##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: TTR

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(tidyquant)
```

```
## Warning: package 'tidyquant' was built under R version 4.4.2

## Warning: package 'PerformanceAnalytics' was built under R version 4.4.2

## -- Attaching core tidyquant packages ------------------------ tidyquant 1.0.9 --
## v PerformanceAnalytics 2.0.4

## -- Conflicts ------------------------------------------ tidyquant_conflicts() --
## x zoo::as.Date()                 masks base::as.Date()
## x zoo::as.Date.numeric()         masks base::as.Date.numeric()
## x PerformanceAnalytics::legend() masks graphics::legend()
## x quantmod::summary()            masks base::summary()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.4.2

##
## ######################### Warning from 'xts' package ##########################
## #                                                                             #
## # The dplyr lag() function breaks how base R's lag() function is supposed to  #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or       #
## # source() into this session won't work correctly.                            #
## #                                                                             #
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop           #
## # dplyr from breaking base R's lag() function.                                #
## #                                                                             #
## # Code in packages is not affected. It's protected by R's namespace mechanism #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                             #
## ###############################################################################
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:xts':
##
##     first, last
```

```
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(xts)
library(ggplot2)
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.4.2
```

```
## corrplot 0.95 loaded
```

```r
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.4.2
```

```r
library(plotly)
```

```
##
## Attaching package: 'plotly'
##
## The following object is masked from 'package:ggplot2':
##
##     last_plot
##
## The following object is masked from 'package:stats':
##
##     filter
##
## The following object is masked from 'package:graphics':
##
##     layout
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.4.2
```

```
## Loading required package: lattice
```

```r
library(zoo)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.4.2
```

```
## randomForest 4.7-1.2
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:ggplot2':
##
##     margin
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(TTR)
library(xgboost)
```

```
## Warning: package 'xgboost' was built under R version 4.4.2
```

```
##
## Attaching package: 'xgboost'
##
## The following object is masked from 'package:plotly':
##
##     slice
##
## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:randomForest':
##
##     combine
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

Gathering/Fetching the data

```r
# Extract stock data for AAPL, AMZN, MSFT, NVDA
tickers <- c("AAPL", "AMZN", "MSFT", "NVDA")

# Create an empty list to store stock data
stocks_data <- list()
```

```r
# Set date range
start_date <- "2010-01-01"
end_date <- Sys.Date()

# Fetch stock data for each ticker and clean missing values
for (ticker in tickers) {
    # Extract data
    stock_data <- getSymbols(ticker, src = "yahoo", from = start_date, to = end_date, auto.assign = FALS

    # Remove missing values using na.omit (complete cases)
    stock_data <- na.omit(stock_data)

    # Assign cleaned data back to the list
    stocks_data[[ticker]] <- stock_data
}

# Extract daily Close prices
stock_prices_daily <- merge(
    Cl(stocks_data[["AAPL"]]),
    Cl(stocks_data[["AMZN"]]),
    Cl(stocks_data[["MSFT"]]),
    Cl(stocks_data[["NVDA"]])
)

# Rename columns
colnames(stock_prices_daily) <- c("AAPL_Close", "AMZN_Close", "MSFT_Close", "NVDA_Close")

# Display the head of the cleaned daily data
head(stock_prices_daily)
```

Extracting Stock data for Apple, Amazon, Microsoft and NVIDIA

```
##            AAPL_Close AMZN_Close MSFT_Close NVDA_Close
## 2010-01-04   7.643214     6.6950      30.95    0.46225
## 2010-01-05   7.656429     6.7345      30.96    0.46900
## 2010-01-06   7.534643     6.6125      30.77    0.47200
## 2010-01-07   7.520714     6.5000      30.45    0.46275
## 2010-01-08   7.570714     6.6760      30.66    0.46375
## 2010-01-11   7.503929     6.5155      30.27    0.45725
```

```r
# Extract Macroeconomic Indicators

# 1. Interest Rates (10-Year Treasury Constant Maturity Rate)
interest_rates <- getSymbols("DGS10", src = "FRED", from = start_date, to = end_date, auto.assign = FALS
interest_rates <- na.locf(interest_rates)  # Forward fill
colnames(interest_rates) <- "Interest_Rates"

# 2.# Inflation (Consumer Price Index for All Urban Consumers)
inflation <- getSymbols("CPIAUCSL", src = "FRED", from = start_date, to = end_date, auto.assign = FALSE)
inflation <- na.locf(inflation)  # Forward fill
```

```
colnames(inflation) <- "Inflation"

# 3.# GDP (Real Gross Domestic Product) - Quarterly data, needs filling for daily use
gdp <- getSymbols("GDPC1", src = "FRED", from = start_date, to = end_date, auto.assign = FALSE)
gdp <- na.locf(gdp)  # Forward fill
colnames(gdp) <- "GDP"
```

Extracting data for Macroeconomic Indicators (Interest Rates, Inflation and GDP)

```
# Extract Industry or Sector Indexes
# Get S&P 500 Index data
sp500_data <- getSymbols("^GSPC", src = "yahoo", from = start_date, to = end_date, auto.assign = FALSE)
sp500_close <- Cl(sp500_data)
sp500_close <- na.locf(sp500_close)  # Forward fill missing values
colnames(sp500_close) <- "SP500_Close"

# NASDAQ Composite Index
nasdaq_data <- getSymbols("^IXIC", src = "yahoo", from = start_date, to = end_date, auto.assign = FALSE
nasdaq_close <- Cl(nasdaq_data)
nasdaq_close <- na.locf(nasdaq_close)  # Forward fill missing values
colnames(nasdaq_close) <- "NASDAQ_Close"
```

Extracting data for Market Indices S&P500 and NASDAQ

```
# Define Geopolitical Events and Add Them to the Dataset
# Dummy variables for geopolitical events, set to 0 by default
geopolitical_events <- data.frame(Date = index(sp500_close),

Arab_Spring = ifelse(index(sp500_close) >= "2010-01-01" & index(sp500_close) <= "2011-12-31", 1, 0),

European_Sovereign_Debt_Crisis = ifelse(index(sp500_close) >= "2010-01-01" & index(sp500_close) <= "201

Taper_Tantrum = ifelse(index(sp500_close) >= "2013-05-22" & index(sp500_close) <= "2013-09-05", 1, 0),

Brexit = ifelse(index(sp500_close) >= "2016-06-23" & index(sp500_close) <= "2016-06-24", 1, 0),

US_China_Trade_War = ifelse(index(sp500_close) >= "2018-07-06" & index(sp500_close) <= "2020-01-15", 1,

COVID_19_Pandemic = ifelse(index(sp500_close) >= "2020-03-11" & index(sp500_close) <= "2021-10-31", 1,

Russia_Ukraine_War = ifelse(index(sp500_close) >= "2022-02-24", 1, 0))
```

Adding the Geopolitical Major Events

```r
# Convert all data into xts (time series) objects to ensure consistency in time format
sp500_close <- xts(sp500_close, order.by = index(sp500_close))
nasdaq_close <- xts(nasdaq_close, order.by = index(nasdaq_close))
interest_rates <- xts(interest_rates, order.by = index(interest_rates))
inflation <- xts(inflation, order.by = index(inflation))
gdp <- xts(gdp, order.by = index(gdp))

# Ensure the geopolitical events are in xts format with the same index as the stock data
geopolitical_events_xts <- xts(geopolitical_events[, -1], order.by = geopolitical_events$Date)
```

Converting the data to Time Series to ensure consistency in the Data Set

```r
# After checking the data, we can attempt merging again
merged_data_daily <- merge(
  stock_prices_daily,
  sp500_close,
  nasdaq_close,
  interest_rates,
  inflation,
  gdp,
  geopolitical_events_xts,
  all = TRUE  # Ensure we keep all rows, even if data is missing in some columns
)

# Print the date range and number of rows in the merged data
cat("\nMerged Data Date Range:\n")
```

Merging all the data and filling the missing values

```
##
## Merged Data Date Range:
```

```r
print(range(index(merged_data_daily)))
```

```
## [1] "2010-01-01" "2025-05-19"
```

```r
cat("\nNumber of Rows in Merged Data: ", nrow(merged_data_daily), "\n")
```

```
##
## Number of Rows in Merged Data:  4064
```

```r
# Fill missing values using Last Observation Carried Forward (locf)
merged_data_daily <- na.locf(merged_data_daily, fromLast = FALSE)  # Fill forward
merged_data_daily <- na.locf(merged_data_daily, fromLast = TRUE)   # Fill backward for any leading NAs

# **Debugging Step**: Check if there are any remaining missing values
cat("\nRemaining Missing Values After locf:\n")
```

```
##
## Remaining Missing Values After locf:
```

```r
print(sum(is.na(merged_data_daily)))
```

```
## [1] 0
```

```r
# Convert the 'xts' object to a data frame
merged_data_df <- data.frame(Date = index(merged_data_daily), coredata(merged_data_daily))

# Save the cleaned dataset with the date column to a CSV file
write.csv(merged_data_df, file = "cleaned_merged_daily_stock_data.csv", row.names = FALSE)

cat("The cleaned dataset with the date column has been saved as 'cleaned_merged_daily_stock_data.csv'.")
```

```
## The cleaned dataset with the date column has been saved as 'cleaned_merged_daily_stock_data.csv'.
```

```r
summary(merged_data_daily)
```

```
##      Index              AAPL_Close        AMZN_Close        MSFT_Close
##  Min.   :2010-01-01   Min.   :  6.859   Min.   :  5.431   Min.   : 23.01
##  1st Qu.:2013-11-06   1st Qu.: 20.903   1st Qu.: 15.206   1st Qu.: 35.90
##  Median :2017-09-09   Median : 38.855   Median : 49.754   Median : 74.12
##  Mean   :2017-09-09   Mean   : 73.544   Mean   : 74.750   Mean   :142.35
##  3rd Qu.:2021-07-14   3rd Qu.:134.900   3rd Qu.:127.760   3rd Qu.:244.38
##  Max.   :2025-05-19   Max.   :259.020   Max.   :242.060   Max.   :467.56
##    NVDA_Close        SP500_Close    NASDAQ_Close   Interest_Rates
##  Min.   :  0.2220   Min.   :1023   Min.   : 2092   Min.   :0.520
##  1st Qu.:  0.4488   1st Qu.:1763   1st Qu.: 3932   1st Qu.:1.840
##  Median :  3.7287   Median :2473   Median : 6411   Median :2.400
##  Mean   : 16.6917   Mean   :2804   Mean   : 7897   Mean   :2.547
##  3rd Qu.: 14.8309   3rd Qu.:3914   3rd Qu.:11802   3rd Qu.:3.120
##  Max.   :149.4300   Max.   :6144   Max.   :20174   Max.   :4.980
##    Inflation          GDP          Arab_Spring
##  Min.   :217.2   Min.   :16583   Min.   :0.0000
##  1st Qu.:234.1   1st Qu.:17954   1st Qu.:0.0000
##  Median :246.4   Median :19507   Median :0.0000
##  Mean   :255.8   Mean   :19803   Mean   :0.1299
##  3rd Qu.:272.0   3rd Qu.:21571   3rd Qu.:0.0000
##  Max.   :320.3   Max.   :23542   Max.   :1.0000
##  European_Sovereign_Debt_Crisis Taper_Tantrum         Brexit
##  Min.   :0.0000                 Min.   :0.00000   Min.   :0.0000000
##  1st Qu.:0.0000                 1st Qu.:0.00000   1st Qu.:0.0000000
##  Median :0.0000                 Median :0.00000   Median :0.0000000
##  Mean   :0.1951                 Mean   :0.01944   Mean   :0.0004921
##  3rd Qu.:0.0000                 3rd Qu.:0.00000   3rd Qu.:0.0000000
##  Max.   :1.0000                 Max.   :1.00000   Max.   :1.0000000
##  US_China_Trade_War COVID_19_Pandemic Russia_Ukraine_War
##  Min.   :0.00000    Min.   :0.0000    Min.   :0.0000
##  1st Qu.:0.00000    1st Qu.:0.0000    1st Qu.:0.0000
##  Median :0.00000    Median :0.0000    Median :0.0000
##  Mean   :0.09941    Mean   :0.1063    Mean   :0.2101
##  3rd Qu.:0.00000    3rd Qu.:0.0000    3rd Qu.:0.0000
##  Max.   :1.00000    Max.   :1.0000    Max.   :1.0000
```

Exploratory Data Analysis:

The goal of EDA is to understand the structure, patterns, and key relationships in the data set before diving into modeling.

```
# 1. Summary Statistics
cat("\nSummary Statistics:\n")
```

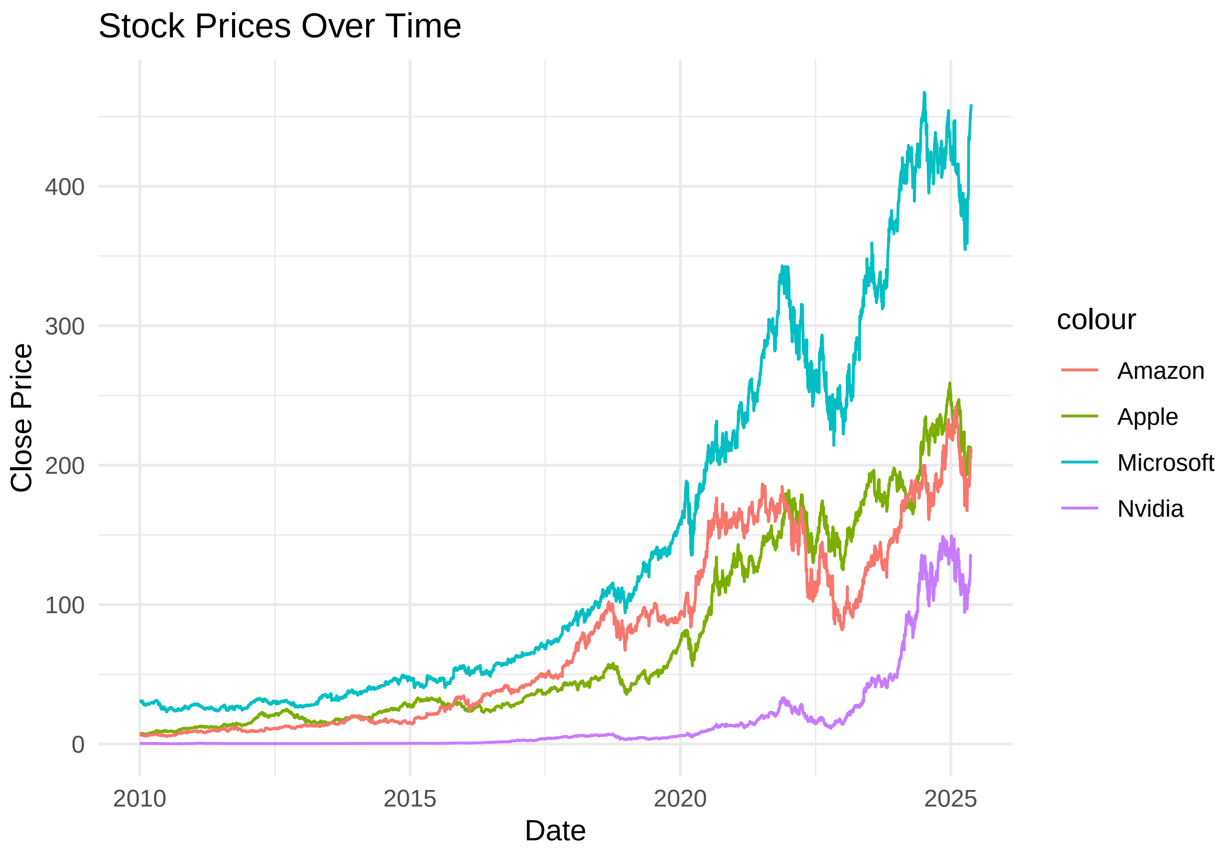
```
##
## Summary Statistics:
```

```
summary(merged_data_df)
```

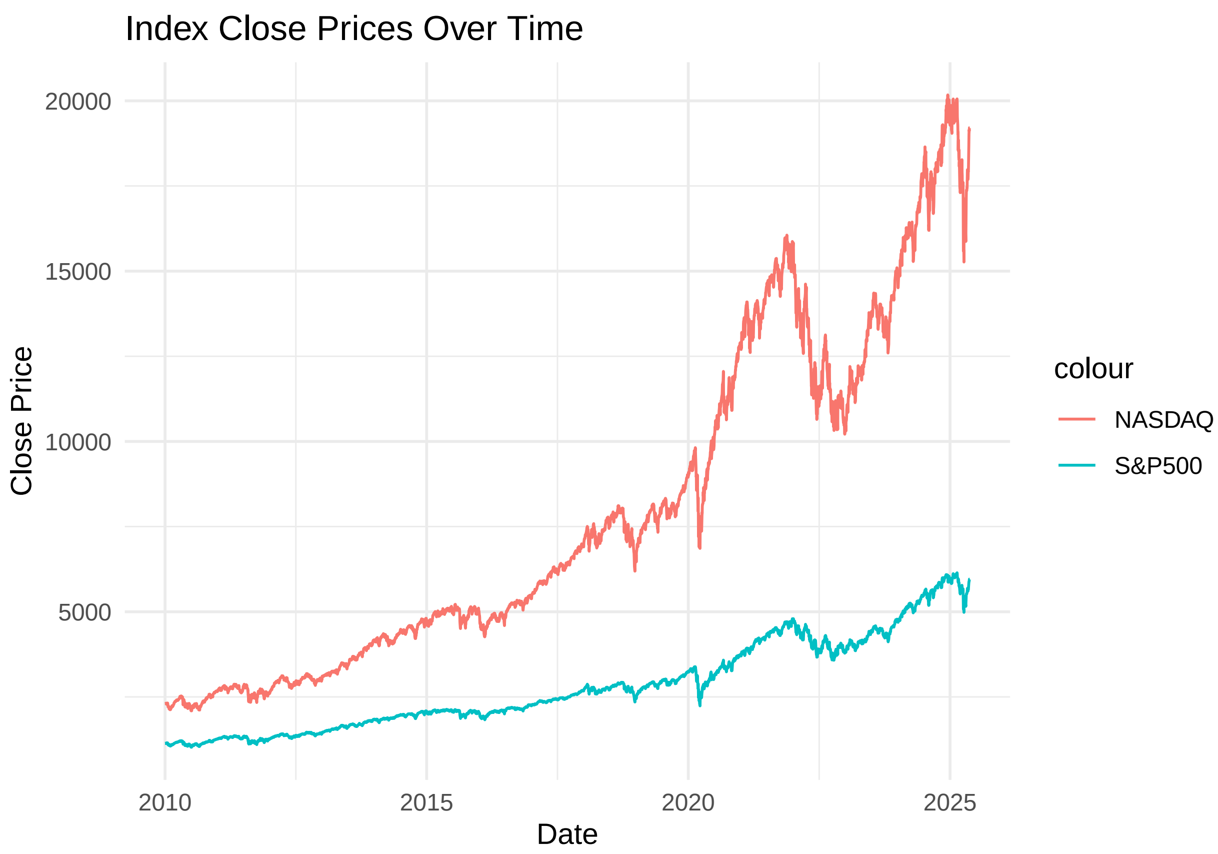
```
##       Date              AAPL_Close        AMZN_Close        MSFT_Close
##  Min.   :2010-01-01   Min.   :  6.859   Min.   :  5.431   Min.   : 23.01
##  1st Qu.:2013-11-06   1st Qu.: 20.903   1st Qu.: 15.206   1st Qu.: 35.90
##  Median :2017-09-09   Median : 38.855   Median : 49.754   Median : 74.12
##  Mean   :2017-09-09   Mean   : 73.544   Mean   : 74.750   Mean   :142.35
##  3rd Qu.:2021-07-14   3rd Qu.:134.900   3rd Qu.:127.760   3rd Qu.:244.38
##  Max.   :2025-05-19   Max.   :259.020   Max.   :242.060   Max.   :467.56
##    NVDA_Close         SP500_Close    NASDAQ_Close    Interest_Rates
##  Min.   :  0.2220   Min.   :1023   Min.   : 2092   Min.   :0.520
##  1st Qu.:  0.4488   1st Qu.:1763   1st Qu.: 3932   1st Qu.:1.840
##  Median :  3.7287   Median :2473   Median : 6411   Median :2.400
##  Mean   : 16.6917   Mean   :2804   Mean   : 7897   Mean   :2.547
##  3rd Qu.: 14.8309   3rd Qu.:3914   3rd Qu.:11802   3rd Qu.:3.120
##  Max.   :149.4300   Max.   :6144   Max.   :20174   Max.   :4.980
##    Inflation          GDP          Arab_Spring
##  Min.   :217.2   Min.   :16583   Min.   :0.0000
##  1st Qu.:234.1   1st Qu.:17954   1st Qu.:0.0000
##  Median :246.4   Median :19507   Median :0.0000
##  Mean   :255.8   Mean   :19803   Mean   :0.1299
##  3rd Qu.:272.0   3rd Qu.:21571   3rd Qu.:0.0000
##  Max.   :320.3   Max.   :23542   Max.   :1.0000
##  European_Sovereign_Debt_Crisis Taper_Tantrum          Brexit
##  Min.   :0.0000                 Min.   :0.00000   Min.   :0.0000000
##  1st Qu.:0.0000                 1st Qu.:0.00000   1st Qu.:0.0000000
##  Median :0.0000                 Median :0.00000   Median :0.0000000
##  Mean   :0.1951                 Mean   :0.01944   Mean   :0.0004921
##  3rd Qu.:0.0000                 3rd Qu.:0.00000   3rd Qu.:0.0000000
##  Max.   :1.0000                 Max.   :1.00000   Max.   :1.0000000
##  US_China_Trade_War COVID_19_Pandemic Russia_Ukraine_War
##  Min.   :0.00000    Min.   :0.0000    Min.   :0.0000
##  1st Qu.:0.00000    1st Qu.:0.0000    1st Qu.:0.0000
##  Median :0.00000    Median :0.0000    Median :0.0000
##  Mean   :0.09941    Mean   :0.1063    Mean   :0.2101
##  3rd Qu.:0.00000    3rd Qu.:0.0000    3rd Qu.:0.0000
##  Max.   :1.00000    Max.   :1.0000    Max.   :1.0000
```

```
# 2. Visualizations

# Line Plot: Stock Prices
```

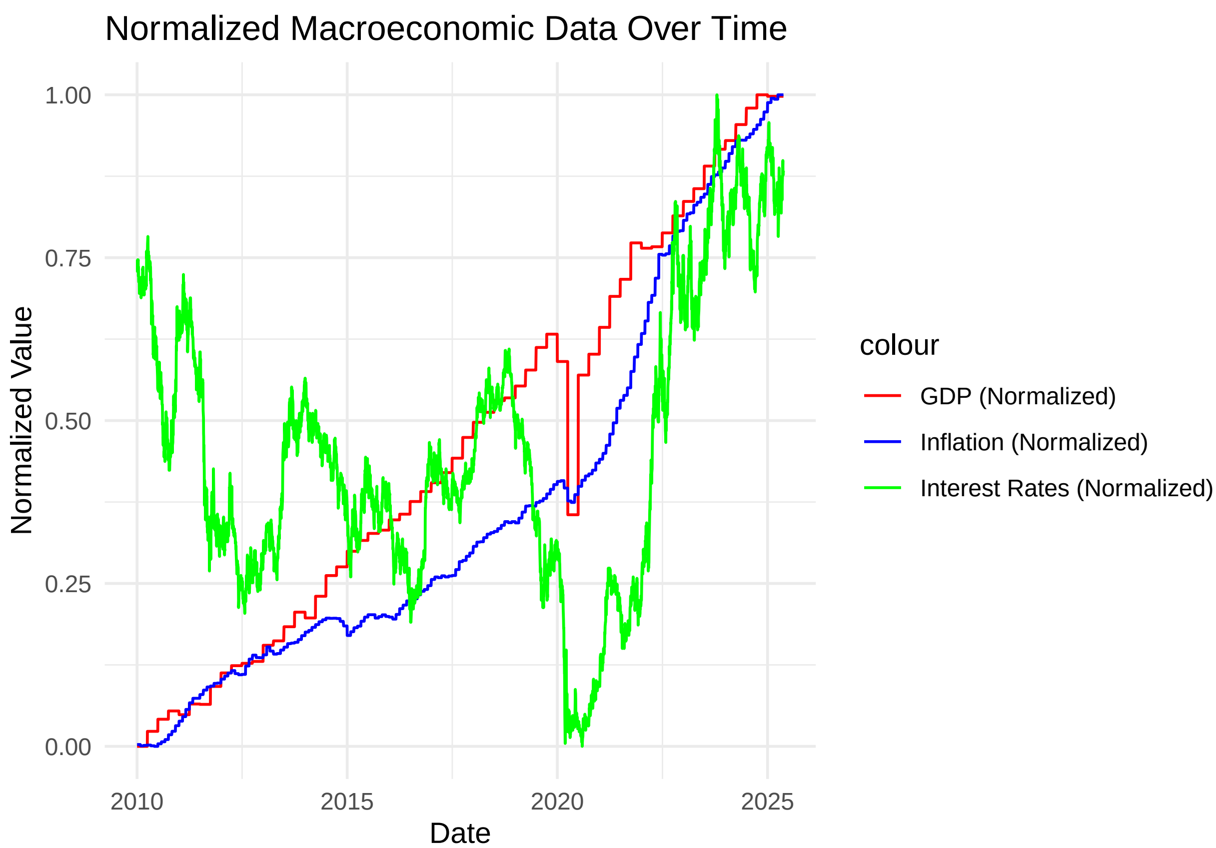
```r
ggplot(merged_data_df, aes(x = Date)) +
  geom_line(aes(y = AAPL_Close, color = "Apple")) +
  geom_line(aes(y = AMZN_Close, color = "Amazon")) +
  geom_line(aes(y = MSFT_Close, color = "Microsoft")) +
  geom_line(aes(y = NVDA_Close, color = "Nvidia")) +
  labs(title = "Stock Prices Over Time", x = "Date", y = "Close Price") +
  theme_minimal()
```




```r
# Line Plot: Market Indices Close Prices
ggplot(merged_data_df, aes(x = Date)) +
  geom_line(aes(y = SP500_Close, color = "S&P500")) +
  geom_line(aes(y = NASDAQ_Close, color = "NASDAQ")) +
  labs(title = "Index Close Prices Over Time", x = "Date", y = "Close Price") +
  theme_minimal()
```

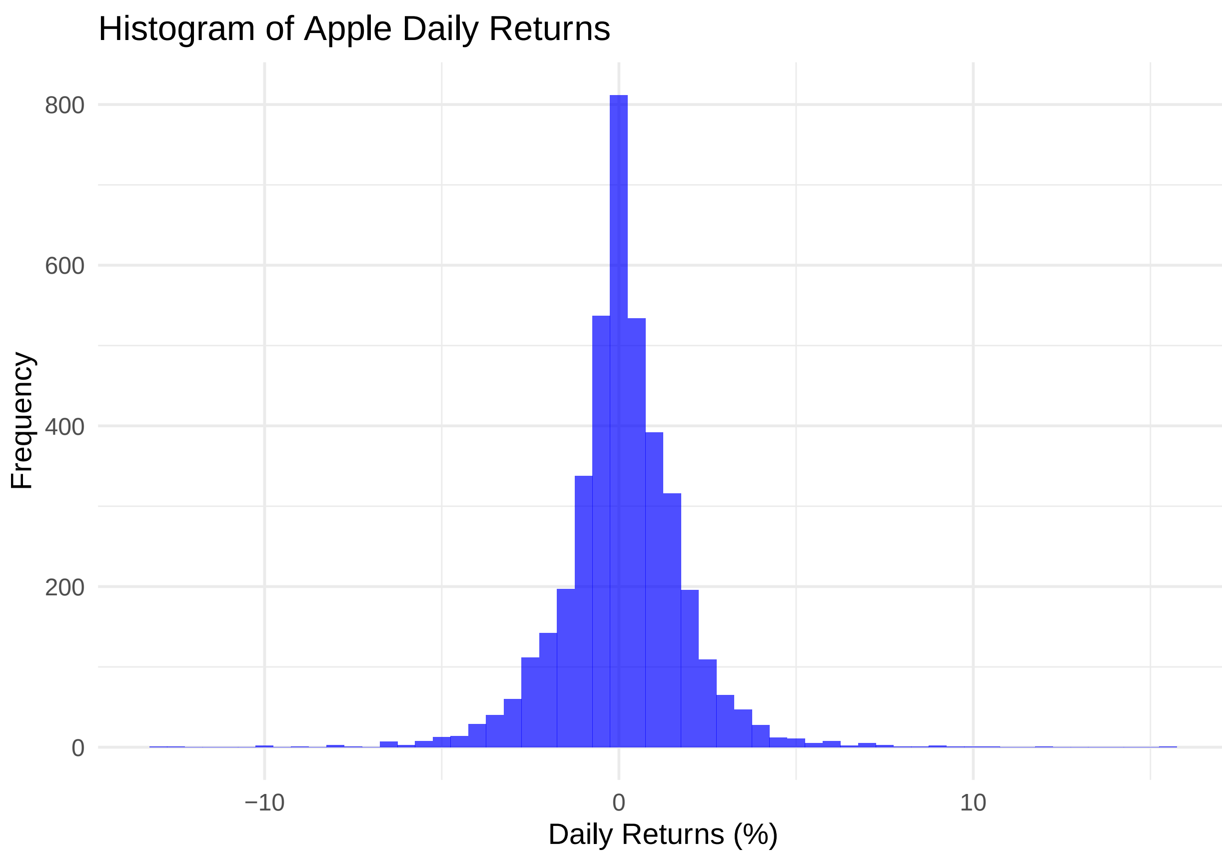



```r
# Line Plot: Macroeconomic data
# Normalize the data
merged_data_df$GDP_Norm <- (merged_data_df$GDP - min(merged_data_df$GDP)) /
                           (max(merged_data_df$GDP) - min(merged_data_df$GDP))
merged_data_df$Inflation_Norm <- (merged_data_df$Inflation - min(merged_data_df$Inflation)) /
                                 (max(merged_data_df$Inflation) - min(merged_data_df$Inflation))
merged_data_df$Interest_Rates_Norm <- (merged_data_df$Interest_Rates - min(merged_data_df$Interest_Rate
                                      (max(merged_data_df$Interest_Rates) - min(merged_data_df$Interest

# Plot normalized data
ggplot(merged_data_df, aes(x = Date)) +
  geom_line(aes(y = GDP_Norm, color = "GDP (Normalized)")) +
  geom_line(aes(y = Inflation_Norm, color = "Inflation (Normalized)")) +
  geom_line(aes(y = Interest_Rates_Norm, color = "Interest Rates (Normalized)")) +
  labs(title = "Normalized Macroeconomic Data Over Time", x = "Date", y = "Normalized Value") +
  scale_color_manual(values = c("red", "blue", "green")) +
  theme_minimal()
```

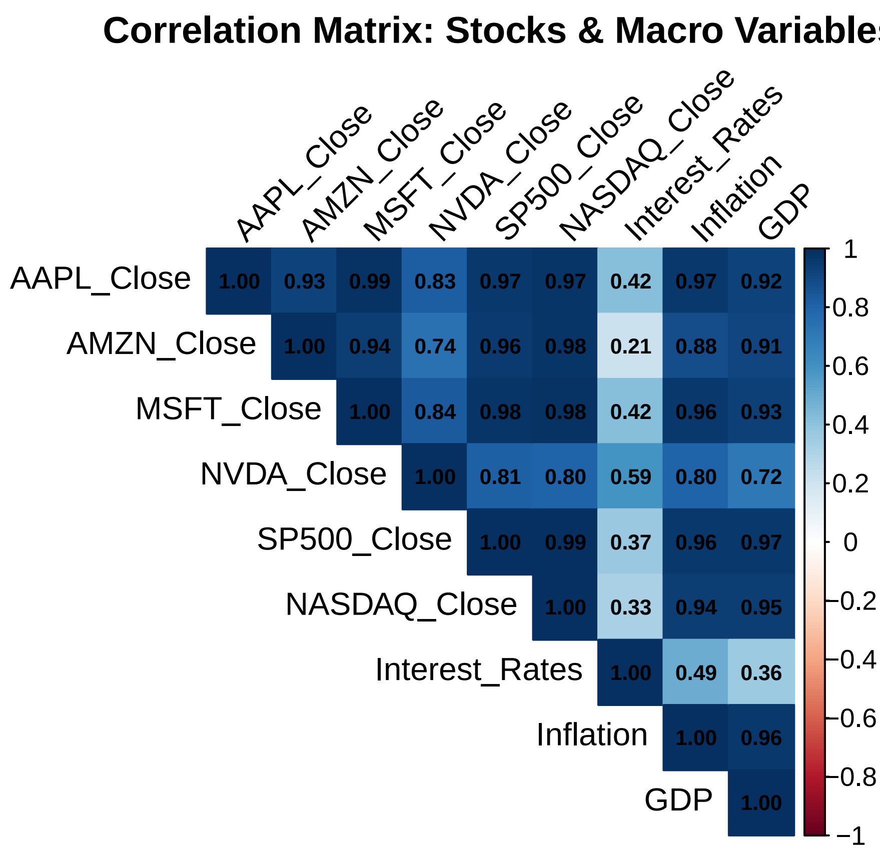



```r
# Histogram: Distribution of Daily Returns for Apple
merged_data_df <- merged_data_df %>%
  mutate(AAPL_Returns = (AAPL_Close / lag(AAPL_Close) - 1) * 100)

ggplot(merged_data_df, aes(x = AAPL_Returns)) +
  geom_histogram(binwidth = 0.5, fill = "blue", alpha = 0.7) +
  labs(title = "Histogram of Apple Daily Returns", x = "Daily Returns (%)", y = "Frequency") +
  theme_minimal()
```

```
## Warning: Removed 1 row containing non-finite outside the scale range
## (‘stat_bin()‘).
```



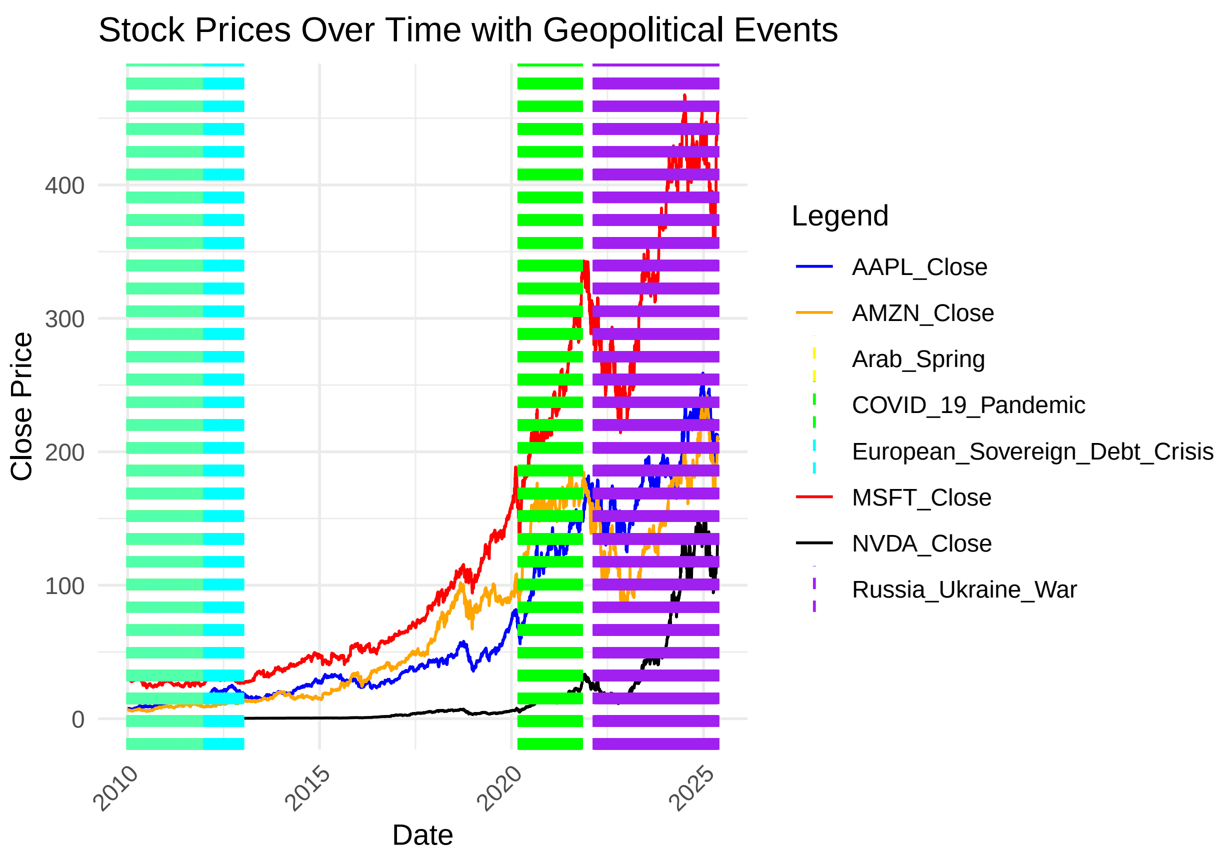

```r
# 3. Enhanced Correlation Matrix
correlation_matrix <- merged_data_df %>%
  select(AAPL_Close, AMZN_Close, MSFT_Close, NVDA_Close, SP500_Close, NASDAQ_Close,
         Interest_Rates, Inflation, GDP) %>%
  cor(use = "complete.obs")

corrplot(correlation_matrix,
         method = "color",
         type = "upper",
         addCoef.col = "black",
         number.cex = 0.7,
         tl.col = "black",
         tl.srt = 45,
         title = "Correlation Matrix: Stocks & Macro Variables",
         mar = c(0,0,2,0))
```




```r
# Step 4: Overlay Events on Stock Price Time-Series
# Reshape data to long format for easier handling of events
event_columns <- c("Arab_Spring", "European_Sovereign_Debt_Crisis",
                   "COVID_19_Pandemic", "Russia_Ukraine_War")

event_data <- merged_data_df %>%
  select(Date, all_of(event_columns)) %>%
  pivot_longer(cols = all_of(event_columns), names_to = "Event", values_to = "Occurred") %>%
  filter(Occurred == 1)  # Only keep rows where the event occurred

# Main plot with events and legend
ggplot(merged_data_df, aes(x = Date)) +
  # Plot stock prices
  geom_line(aes(y = AAPL_Close, color = "AAPL_Close")) +
  geom_line(aes(y = AMZN_Close, color = "AMZN_Close")) +
  geom_line(aes(y = MSFT_Close, color = "MSFT_Close")) +
  geom_line(aes(y = NVDA_Close, color = "NVDA_Close")) +

  # Add vertical lines for events using event_data
  geom_vline(data = event_data, aes(xintercept = as.numeric(Date), color = Event),
             linetype = "dashed", size = 0.5) +

  # Add labels and title
  labs(title = "Stock Prices Over Time with Geopolitical Events",
       x = "Date", y = "Close Price", color = "Legend") +
```

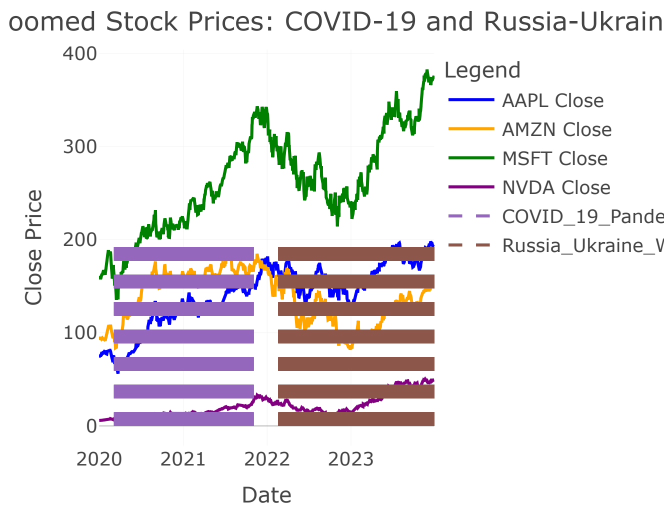
```r
  # Theme customization
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +

  # Customize stock line colors
  scale_color_manual(values = c("AAPL_Close" = "blue", "AMZN_Close" = "orange",
                                "MSFT_Close" = "red", "NVDA_Close" = "black",
                                 "Arab_Spring" = "yellow",
                                 "COVID_19_Pandemic" = "green",
                                 "Russia_Ukraine_War" = "purple",
                                 "European_Sovereign_Debt_Crisis" = "cyan"
                                 ))
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```




```r
#library(plotly)
# Define time windows for the events
covid_period <- as.Date(c("2020-01-01", "2021-12-31"))  # COVID-19 timeframe
```

```r
russia_ukraine_period <- as.Date(c("2022-01-01", "2023-12-31"))  # Russia-Ukraine War timeframe

# Filter data for these periods
zoomed_data <- merged_data_df %>%
  filter((Date >= covid_period[1] & Date <= covid_period[2]) |
         (Date >= russia_ukraine_period[1] & Date <= russia_ukraine_period[2]))

# Filter events for only COVID-19 and Russia-Ukraine War
selected_event_data_zoomed <- zoomed_data %>%
  select(Date, COVID_19_Pandemic, Russia_Ukraine_War) %>%
  pivot_longer(cols = c(COVID_19_Pandemic, Russia_Ukraine_War),
               names_to = "Event", values_to = "Occurred") %>%
  filter(Occurred == 1)

# Create the Plotly plot
plot_zoomed <- plot_ly() %>%
  # Add stock price lines
  add_lines(data = zoomed_data, x = ~Date, y = ~AAPL_Close, name = "AAPL Close",
            line = list(color = "blue")) %>%
  add_lines(data = zoomed_data, x = ~Date, y = ~AMZN_Close, name = "AMZN Close",
            line = list(color = "orange")) %>%
  add_lines(data = zoomed_data, x = ~Date, y = ~MSFT_Close, name = "MSFT Close",
            line = list(color = "green")) %>%
  add_lines(data = zoomed_data, x = ~Date, y = ~NVDA_Close, name = "NVDA Close",
            line = list(color = "purple")) %>%

  # Add vertical lines for COVID-19 and Russia-Ukraine War
  add_segments(data = selected_event_data_zoomed, x = ~Date, xend = ~Date, y = 0,
               yend = max(zoomed_data$AAPL_Close, na.rm = TRUE),
               name = ~Event, line = list(dash = "dash"), hoverinfo = "text",
               text = ~paste("Event:", Event)) %>%

  # Layout and zoomed-in axis limits
  layout(
    title = "Zoomed Stock Prices: COVID-19 and Russia-Ukraine War",
    xaxis = list(title = "Date", range = c(min(zoomed_data$Date), max(zoomed_data$Date))),
    yaxis = list(title = "Close Price"),
    legend = list(title = list(text = "Legend")),
    hovermode = "x unified"
  )

# Display the plot
plot_zoomed
```



Zoomed-in plot to see the variations in stocks closely

Data Preparation for Predictive Modeling

This step ensures that the data is formatted and ready for predictive analysis.Following are steps involved:

- Feature Engineering: We added useful features like daily returns, moving averages, volatility, and lagged prices to improve the predictive model.
- Train-Test Split: Splitting ensures we have separate data set for training and testing, reducing the risk of over fitting.
- Scaling: Min-Max Scaling converts all features to the same scale (between 0 and 1), which is crucial for machine learning models.

Features and Model Building for Stocks Close Price

- Moving Averages: Shows trend direction and strength
- Volatility: Measures price variability, Shows market uncertainty, Indicates potential risk

```
# Generalized code Close price of all stocks
# Calculating the returns for stocks and indexes.
merged_data_df <- merged_data_df %>%
  mutate(AAPL_Returns = (AAPL_Close / lag(AAPL_Close) - 1) * 100,
         AMZN_Returns = (AMZN_Close / lag(AMZN_Close) - 1) * 100,
         MSFT_Returns = (MSFT_Close / lag(MSFT_Close) - 1) * 100,
         NVDA_Returns = (NVDA_Close / lag(NVDA_Close) - 1) * 100,
         SP500_Returns = (SP500_Close / lag(SP500_Close) - 1) * 100,
         NASDAQ_Returns = (NASDAQ_Close / lag(NASDAQ_Close) - 1) * 100)

# Calculating the Moving averages( 7-day and 30-day) and volatility for Sector Index closing prices
merged_data_df <- merged_data_df %>%
  mutate( SP500_MA7 = zoo::rollmean(SP500_Close, 7, fill = NA, align = "right"),
          SP500_MA30 = zoo::rollmean(SP500_Close, 30, fill = NA, align = "right"),
          NASDAQ_MA7 = zoo::rollmean(NASDAQ_Close, 7, fill = NA, align = "right"),
          NASDAQ_MA30 = zoo::rollmean(NASDAQ_Close, 30, fill = NA, align = "right"),
          SP500_Volatility = zoo::rollapply(SP500_Returns, 7, sd, fill = NA, align = "right"),
          NASDAQ_Volatility = zoo::rollapply(NASDAQ_Returns, 7, sd, fill = NA, align = "right"))


# Function to create Technical indicators (Moving averages( 7 and 30-day),
# volatility and Previous/lag values) for any stock
create_stock_features <- function(data, stock_symbol) {
    data %>%
        mutate(
            # Moving Averages
            !!paste0(stock_symbol, "_MA7") := zoo::rollmean(get(paste0(stock_symbol, "_Close")),
                                                      7, fill = NA, align = "right"),
            !!paste0(stock_symbol, "_MA30") := zoo::rollmean(get(paste0(stock_symbol, "_Close")),
                                                      30, fill = NA, align = "right"),
            !!paste0(stock_symbol, "_Volatility") := zoo::rollapply(get(paste0(stock_symbol, "_Returns"
                                                              7, sd, fill = NA, align = "right"),
            !!paste0(stock_symbol, "_Lag1") := lag(get(paste0(stock_symbol, "_Close")))
        )
}
```

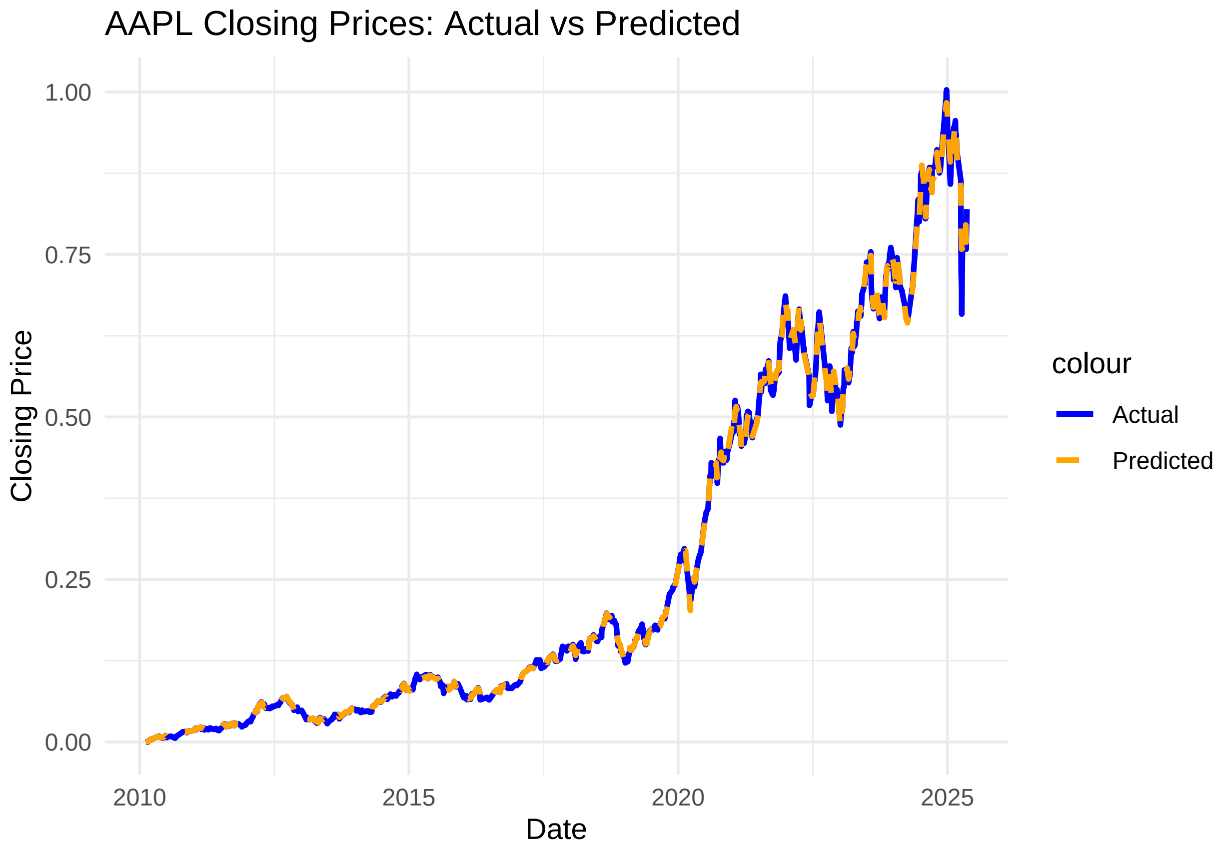
```r
# Function to predict stock price
predict_stock_price <- function(merged_data_df, stock_symbol) {
    # Handle missing values first
    merged_data_df <- na.omit(merged_data_df)

    # Create features for the specific stock
    stock_features <- c(paste0(stock_symbol, "_MA7"),
                       paste0(stock_symbol, "_MA30"),
                       paste0(stock_symbol, "_Volatility"),
                       paste0(stock_symbol, "_Lag1"))


    # Combine with market and macro features
    predict_features <- c(
        stock_features,
        "Interest_Rates", "Inflation", "GDP", "NASDAQ_Close", "SP500_Close",
        "SP500_MA7", "SP500_MA30", "SP500_Volatility",
        "NASDAQ_MA7", "NASDAQ_MA30", "NASDAQ_Volatility",
        "Arab_Spring", "COVID_19_Pandemic", "Russia_Ukraine_War",
        "European_Sovereign_Debt_Crisis"
    )

    # Verify all features exist in the dataset
    missing_features <- predict_features[!predict_features %in% names(merged_data_df)]
    if(length(missing_features) > 0) {
        stop("Missing features: ", paste(missing_features, collapse = ", "))
    }

    # Train-Test Split
    set.seed(123)
    target_col <- paste0(stock_symbol, "_Close")
    train_index <- createDataPartition(merged_data_df[[target_col]], p = 0.8, list = FALSE)
    train_data <- merged_data_df[train_index, ]
    test_data <- merged_data_df[-train_index, ]

    # Scaling
    scaler <- preProcess(train_data[, -1], method = c("range"))  # Exclude date column
    train_data_scaled <- predict(scaler, train_data)
    test_data_scaled <- predict(scaler, test_data)

    # Filter relevant columns and ensure no missing values
    train_data_scaled_filtered <- train_data_scaled %>%
        select(all_of(predict_features), target = !!target_col) %>%
        na.omit()

    test_data_scaled_filtered <- test_data_scaled %>%
        select(all_of(predict_features), target = !!target_col) %>%
        na.omit()

    # Check if we have enough data after filtering
    if(nrow(train_data_scaled_filtered) < 10 || nrow(test_data_scaled_filtered) < 10) {
        stop("Not enough data after removing missing values")
    }
```

```r
# Build Model
model <- randomForest(
              target ~ .,
              data = train_data_scaled_filtered,
              ntree = 100,
              na.action = na.omit)

# Feature Importance
importance_values <- importance(model)
cat("\nFeature Importance for", stock_symbol, ":\n")
print(importance_values[order(-importance_values[, 1]), ])

# Predictions
predictions <- predict(model, test_data_scaled_filtered %>% select(-target))
actual <- test_data_scaled_filtered$target

# Metrics
metrics <- list(
    RMSE = sqrt(mean((predictions - actual)^2)),
    MAE = mean(abs(predictions - actual)),
    MAPE = mean(abs((predictions - actual) / actual)) * 100,
    sMAPE = mean(2 * abs(predictions - actual) / (abs(predictions) + abs(actual))) * 100
)

# Visualization
plots <- list(
    scatter = ggplot(data = data.frame(Actual = actual, Predicted = predictions),
                aes(x = Actual, y = Predicted)) +
        geom_point(color = "blue", alpha = 0.6) +
        geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed") +
        labs(title = paste(stock_symbol, "Actual vs. Predicted Closing Prices"),
             x = "Actual Closing Price",
             y = "Predicted Closing Price") +
        theme_minimal(),

    timeseries = ggplot(data = data.frame(
        Date = test_data$Date,
        Actual = actual,
        Predicted = predictions
    ), aes(x = Date)) +
        geom_line(aes(y = Actual, color = "Actual"), size = 1) +
        geom_line(aes(y = Predicted, color = "Predicted"),
                  size = 1, linetype = "dashed") +
        scale_color_manual(values = c("Actual" = "blue", "Predicted" = "orange")) +
        labs(title = paste(stock_symbol, "Closing Prices: Actual vs Predicted"),
             x = "Date",
             y = "Closing Price") +
        theme_minimal()
)

return(list(
    model = model,
    metrics = metrics,
```

```r
        predictions = predictions,
        actual = actual,
        plots = plots
    ))
}

# Process each stock
stock_symbols <- c("AAPL", "AMZN", "MSFT", "NVDA")
results <- list()

for(symbol in stock_symbols) {
    cat("\nProcessing", symbol, "...\n")
    tryCatch({
        # Create features
        merged_data_df <- create_stock_features(merged_data_df, symbol)

        # Predict and store results
        results[[symbol]] <- predict_stock_price(merged_data_df, symbol)

        # Print metrics
        cat("\nMetrics for", symbol, ":\n")
        print(results[[symbol]]$metrics)

        # Display plots
        print(results[[symbol]]$plots$scatter)
        print(results[[symbol]]$plots$timeseries)
    }, error = function(e) {
        cat("Error processing", symbol, ":", e$message, "\n")
    })
}
```

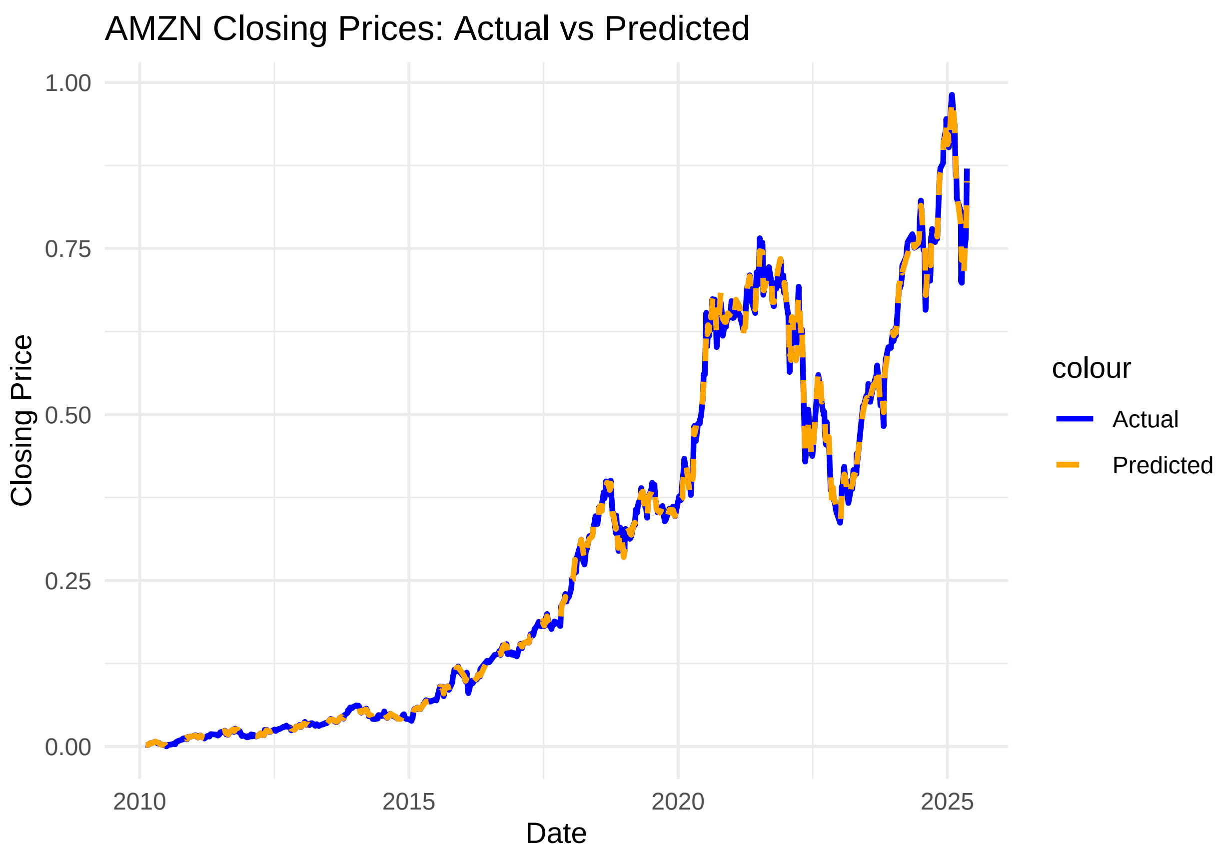
```
##
## Processing AAPL ...
##
## Feature Importance for AAPL :
##                  NASDAQ_MA30                     AAPL_MA30
##                 5.592792e+01                  4.779570e+01
##                    Inflation                     AAPL_Lag1
##                 4.362816e+01                  3.289781e+01
##                     AAPL_MA7                    NASDAQ_MA7
##                 2.431005e+01                  1.576890e+01
##                 NASDAQ_Close                    SP500_MA30
##                 7.237752e+00                  5.250821e+00
##                    SP500_MA7                   SP500_Close
##                 3.139022e+00                  2.614787e+00
##                          GDP              COVID_19_Pandemic
##                 1.083361e+00                  1.285733e-01
##               Interest_Rates              SP500_Volatility
##                 7.635769e-02                  2.400927e-02
##              AAPL_Volatility             NASDAQ_Volatility
##                 2.222120e-02                  1.898415e-02
##                  Arab_Spring European_Sovereign_Debt_Crisis
##                 1.402879e-02                  2.088092e-04
```

```
##             Russia_Ukraine_War
##                   1.407891e-04
##
## Metrics for AAPL :
## $RMSE
## [1] 0.007707057
##
## $MAE
## [1] 0.003359324
##
## $MAPE
## [1] Inf
##
## $sMAPE
## [1] 2.266057
```



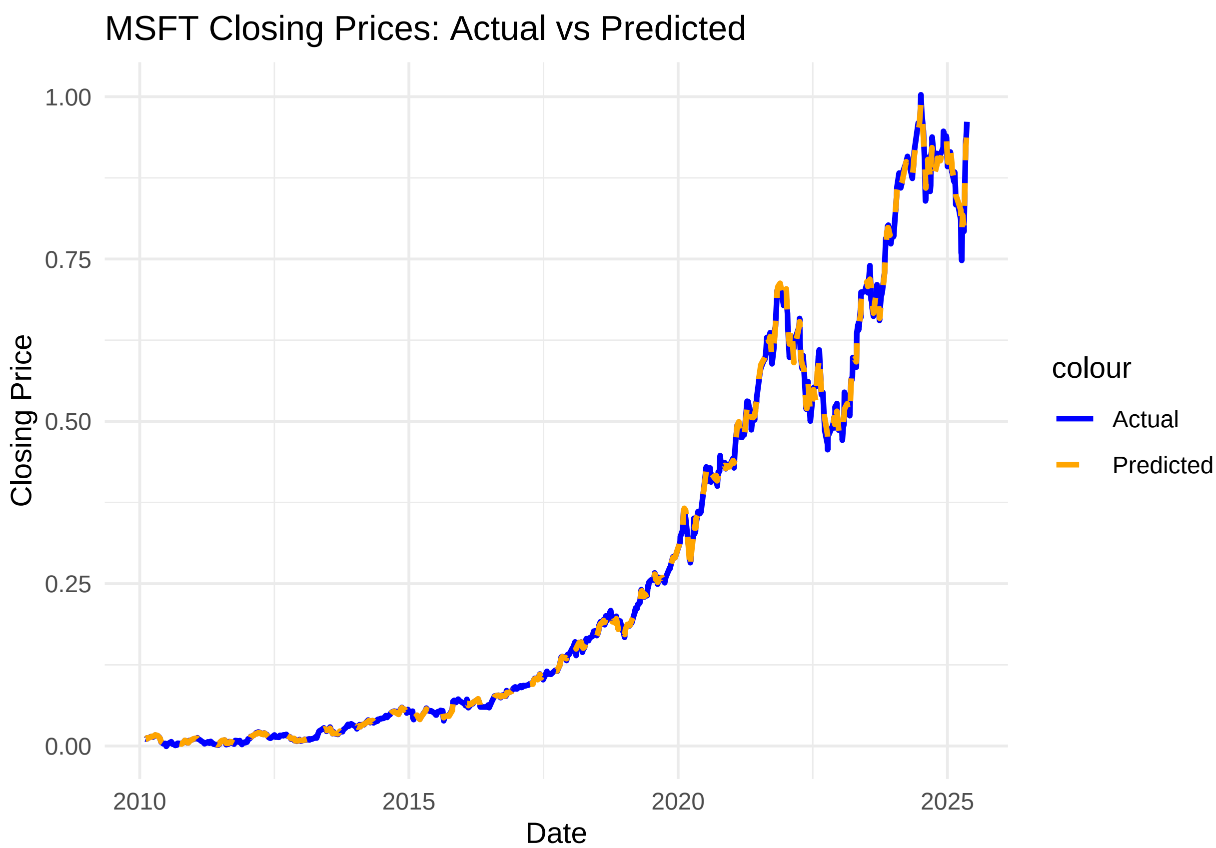

```
##
## Processing AMZN ...
##
## Feature Importance for AMZN :
##                   AMZN_Lag1                      AMZN_MA7
##                8.547265e+01                  4.275674e+01
##                   AMZN_MA30                   NASDAQ_MA30
##                2.761771e+01                  2.392632e+01
##                NASDAQ_Close                     Inflation
##                2.081131e+01                  1.473820e+01
##                   SP500_MA7                    NASDAQ_MA7
##                6.497506e+00                  6.335889e+00
##                 SP500_Close                    SP500_MA30
##                3.601858e+00                  2.275681e+00
##                         GDP                Interest_Rates
##                5.072488e-01                  2.546917e-01
##           COVID_19_Pandemic             NASDAQ_Volatility
##                1.827638e-01                  4.212127e-02
##            SP500_Volatility               AMZN_Volatility
##                3.461901e-02                  3.369960e-02
## European_Sovereign_Debt_Crisis          Russia_Ukraine_War
##                1.154631e-02                  8.491211e-03
##                 Arab_Spring
##                1.912785e-05
##
## Metrics for AMZN :
```

```
## $RMSE
## [1] 0.007827621
##
## $MAE
## [1] 0.004171797
##
## $MAPE
## [1] 3.286127
##
## $sMAPE
## [1] 2.345391
```



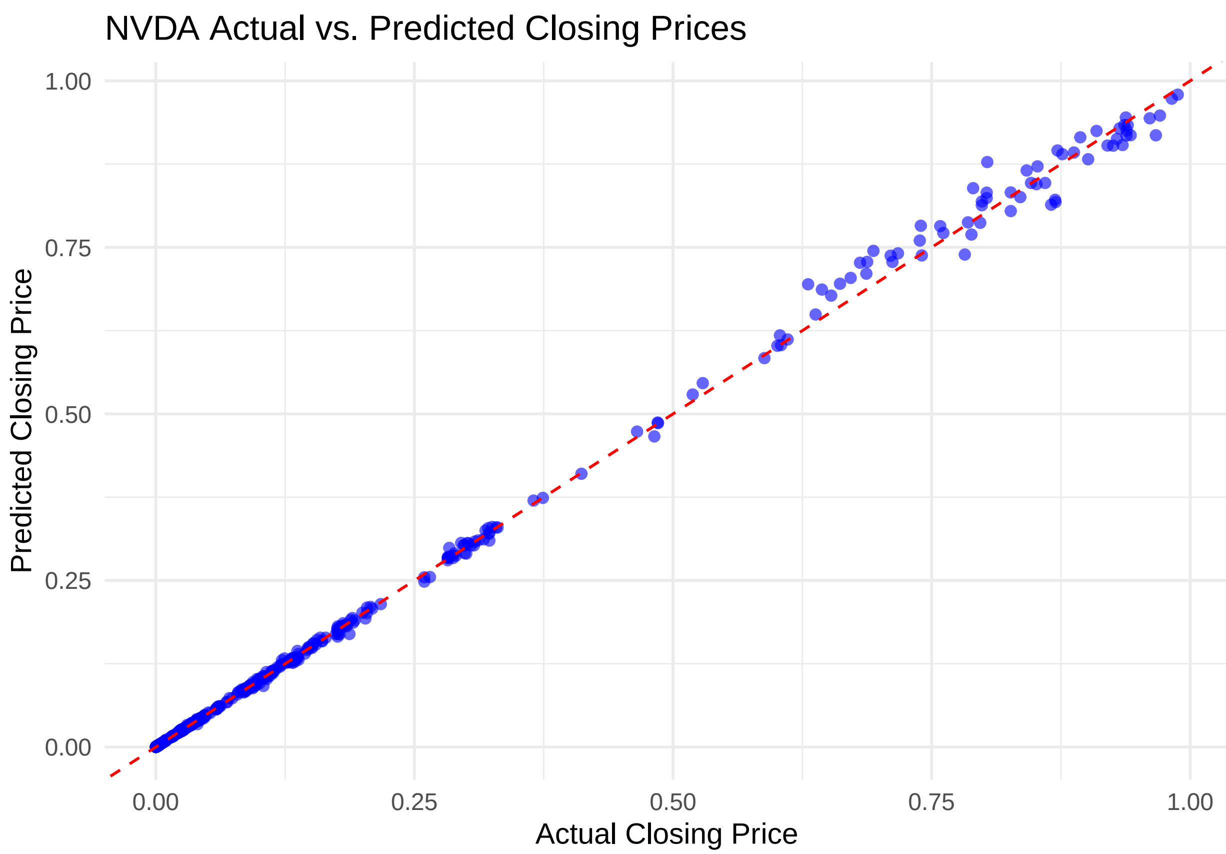

```
##
## Processing MSFT ...
##
## Feature Importance for MSFT :
##                    MSFT_MA7                    MSFT_MA30
##                58.553636198                 57.600791230
##                   MSFT_Lag1                 NASDAQ_Close
##                43.472146054                 41.860929622
##                  NASDAQ_MA7                  NASDAQ_MA30
##                32.795634039                 16.907546816
##                   Inflation                   SP500_MA30
##                10.337098076                  6.251538737
##                         GDP                    SP500_MA7
##                 4.239359329                  3.836441520
##                 SP500_Close               Interest_Rates
##                 1.844496009                  0.302776344
## European_Sovereign_Debt_Crisis             SP500_Volatility
##                 0.059461880                  0.039381671
##           NASDAQ_Volatility              MSFT_Volatility
##                 0.028114038                  0.018865873
##          Russia_Ukraine_War            COVID_19_Pandemic
##                 0.003850333                  0.001705957
##                 Arab_Spring
##                 0.001195424
##
## Metrics for MSFT :
```

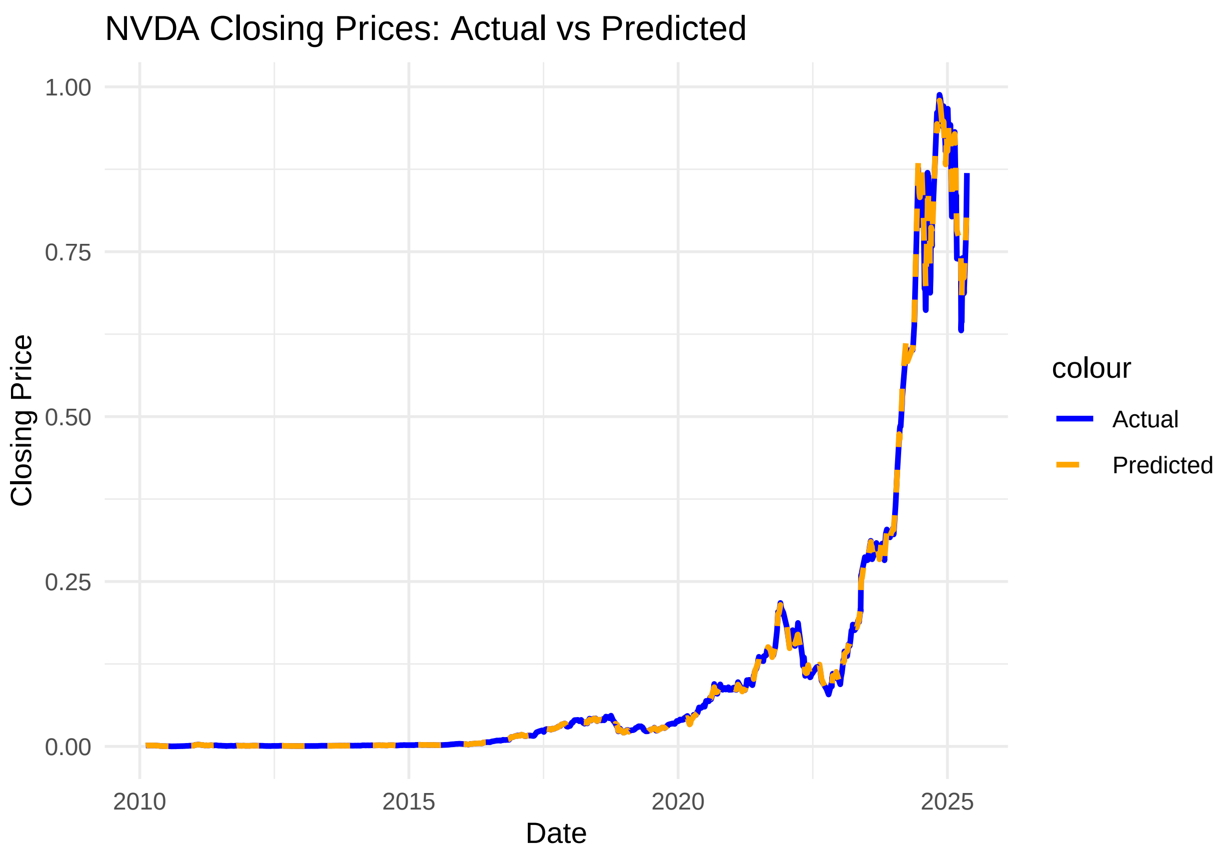
```
## $RMSE
## [1] 0.005930075
##
## $MAE
## [1] 0.003202745
##
## $MAPE
## [1] Inf
##
## $sMAPE
## [1] 3.659824
```




```
##
## Processing NVDA ...
##
## Feature Importance for NVDA :
##                    NVDA_MA7                       NVDA_MA30
##                2.803190e+01                    2.511539e+01
##                   NVDA_Lag1                     SP500_Close
##                2.359413e+01                    2.300818e+01
##                   SP500_MA7                      SP500_MA30
##                2.073735e+01                    1.673181e+01
##                   Inflation                             GDP
##                4.962530e+00                    4.421731e+00
##                 NASDAQ_MA30                      NASDAQ_MA7
##                2.063005e+00                    1.870645e+00
##                NASDAQ_Close                  Interest_Rates
##                9.231906e-01                    1.043357e-01
##             NVDA_Volatility               NASDAQ_Volatility
##                6.989336e-02                    3.807415e-02
##            SP500_Volatility European_Sovereign_Debt_Crisis
##                3.596806e-02                    1.080690e-03
##           COVID_19_Pandemic               Russia_Ukraine_War
##                1.053530e-03                    3.102243e-04
##                 Arab_Spring
##                6.276233e-09
##
## Metrics for NVDA :
```

```
## $RMSE
## [1] 0.008227324
##
## $MAE
## [1] 0.002684289
##
## $MAPE
## [1] 2.906543
##
## $sMAPE
## [1] 2.865644
```






```r
# 1. Compare Performance Metrics Across Stocks
compare_performance <- function(results, stock_symbols) {
    # Combine metrics for all stocks
    metrics_df <- data.frame(
        Stock = stock_symbols,
        RMSE = sapply(results, function(x) x$metrics$RMSE),
        MAE = sapply(results, function(x) x$metrics$MAE),
        sMAPE = sapply(results, function(x) x$metrics$sMAPE)
    )

    # Create comparison plots
    metrics_long <- tidyr::pivot_longer(metrics_df,
                                        cols = c("RMSE", "MAE", "sMAPE"),
                                        names_to = "Metric",
                                        values_to = "Value")

    comparison_plot <- ggplot(metrics_long, aes(x = Stock, y = Value, fill = Stock)) +
        geom_bar(stat = "identity") +
        facet_wrap(~Metric, scales = "free_y") +
        labs(title = "Performance Metrics Comparison Across Stocks",
             y = "Value") +
        theme_minimal() +
        theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```r
    return(list(metrics = metrics_df, plot = comparison_plot))
}

# 2. Analyze Feature Importance Across Stocks
analyze_feature_importance <- function(results, stock_symbols) {
    # Extract and combine feature importance for all stocks
    importance_list <- lapply(stock_symbols, function(symbol) {
        imp <- as.data.frame(importance(results[[symbol]]$model))
        imp$Feature <- rownames(imp)
        imp$Stock <- symbol
        return(imp)
    })

    importance_df <- do.call(rbind, importance_list)

    # Create heatmap of feature importance
    importance_plot <- ggplot(importance_df,
                              aes(x = Stock, y = Feature, fill = IncNodePurity)) +
        geom_tile() +
        scale_fill_gradient(low = "white", high = "steelblue") +
        labs(title = "Feature Importance Heatmap Across Stocks",
             fill = "Importance") +
        theme_minimal() +
        theme(axis.text.x = element_text(angle = 45, hjust = 1))

    return(list(importance = importance_df, plot = importance_plot))
}

# 3. Create Portfolio Analysis of stocks
create_portfolio_predictions <- function(results, stock_symbols, test_data) {
    # First get minimum length across all results to ensure consistency
    min_rows <- min(sapply(results, function(x) length(x$actual)))

    # Create initial dataframe with the minimum number of rows
    portfolio_df <- data.frame(
        Date = tail(test_data$Date, min_rows),
        Portfolio_Actual = 0,
        Portfolio_Predicted = 0
    )

    # Calculate equal weights
    weights <- rep(1/length(stock_symbols), length(stock_symbols))
    names(weights) <- stock_symbols

    # Add each stock's contribution to portfolio
    for(symbol in stock_symbols) {
        # Take only the last min_rows from each result
        actual_values <- tail(results[[symbol]]$actual, min_rows)
        predicted_values <- tail(results[[symbol]]$predictions, min_rows)

        # Add individual stock data
        portfolio_df[[paste0(symbol, "_Actual")]] <- actual_values
        portfolio_df[[paste0(symbol, "_Predicted")]] <- predicted_values
```

```r
        # Add weighted contribution to portfolio
        portfolio_df$Portfolio_Actual <- portfolio_df$Portfolio_Actual +
            actual_values * weights[symbol]
        portfolio_df$Portfolio_Predicted <- portfolio_df$Portfolio_Predicted +
            predicted_values * weights[symbol]
    }

    # Calculate portfolio metrics
    portfolio_metrics <- list(
        RMSE = sqrt(mean((portfolio_df$Portfolio_Predicted - portfolio_df$Portfolio_Actual)^2)),
        MAE = mean(abs(portfolio_df$Portfolio_Predicted - portfolio_df$Portfolio_Actual)),
        sMAPE = mean(2 * abs(portfolio_df$Portfolio_Predicted - portfolio_df$Portfolio_Actual) /
                    (abs(portfolio_df$Portfolio_Predicted) + abs(portfolio_df$Portfolio_Actual))) * 100
    )


    portfolio_plot <- ggplot(portfolio_df, aes(x = Date)) +
        geom_line(aes(y = Portfolio_Actual, color = "Actual"), size = 1) +
        geom_line(aes(y = Portfolio_Predicted, color = "Predicted"),
                  size = 1, linetype = "dashed") +
        labs(title = "Portfolio Performance: Actual vs Predicted",
             x = "Date", y = "Portfolio Value",
             color = "Type") +
        scale_color_manual(values = c("Actual" = "blue", "Predicted" = "red")) +
        theme_minimal() +
        theme(legend.position = "bottom")

    # Create individual stock performance plots
    stock_plots <- list()
    for(symbol in stock_symbols) {
        stock_plots[[symbol]] <- ggplot(portfolio_df, aes(x = Date)) +
            geom_line(aes_string(y = paste0(symbol, "_Actual"), color = "'Actual'")) +
            geom_line(aes_string(y = paste0(symbol, "_Predicted"), color = "'Predicted'")) +
            labs(title = paste(symbol, "Performance"),
                 x = "Date", y = "Value") +
            scale_color_manual(values = c("Actual" = "blue", "Predicted" = "red")) +
            theme_minimal() +
            theme(legend.position = "bottom")
    }

    return(list(
        portfolio_data = portfolio_df,
        metrics = portfolio_metrics,
        portfolio_plot = portfolio_plot,
        stock_plots = stock_plots,
        weights = weights
    ))
}

# Get test_data from one of the stock predictions
first_symbol <- stock_symbols[1]
test_data_length <- length(results[[first_symbol]]$actual)
# Run analyses
```

```r
performance_comparison <- compare_performance(results, stock_symbols)
feature_analysis <- analyze_feature_importance(results, stock_symbols)

# Create date sequence for portfolio analysis
date_sequence <- tail(merged_data_df$Date, test_data_length)
test_data <- data.frame(Date = date_sequence)

portfolio_results <- create_portfolio_predictions(results, stock_symbols, test_data)
```

Performance Prediction for Close Price of Stocks

```
## Warning: ‘aes_string()‘ was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with ‘aes()‘.
## i See also ‘vignette("ggplot2-in-packages")‘ for more information.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

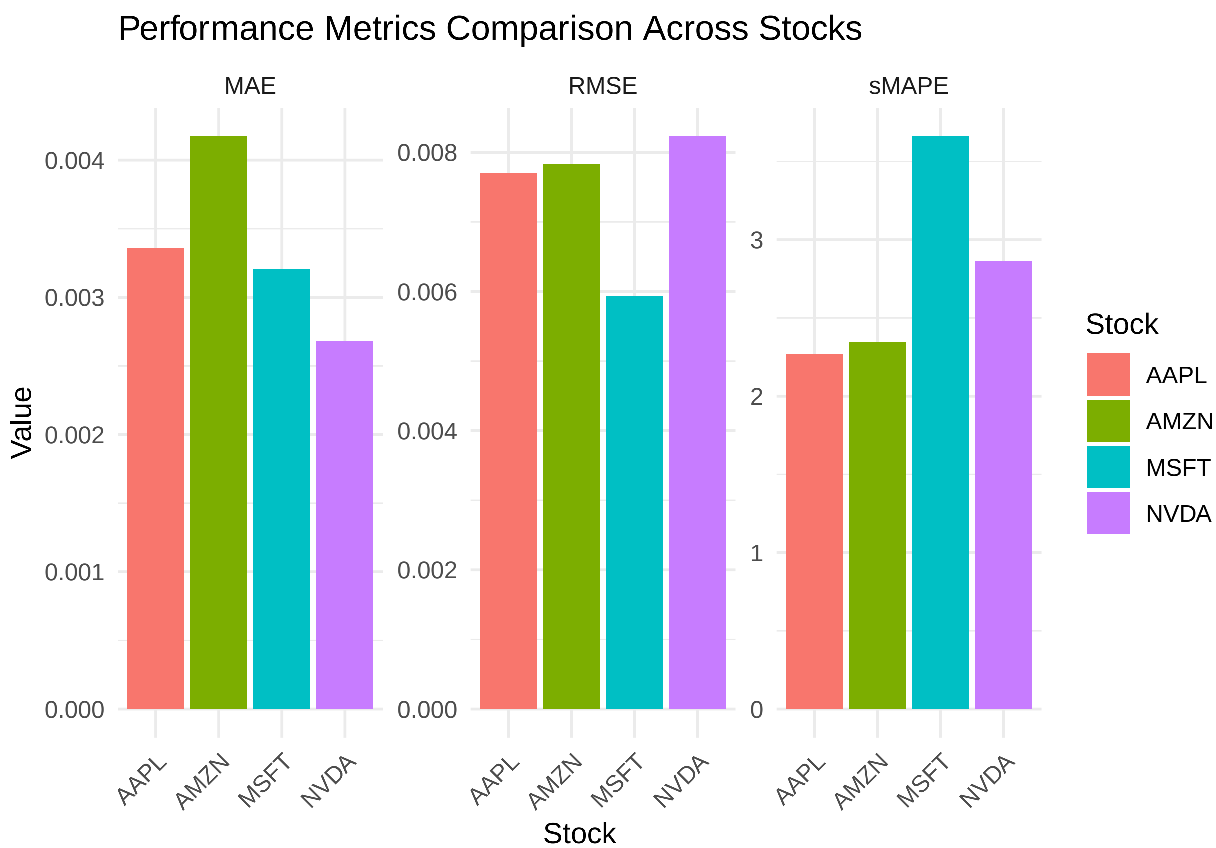
```r
# Display results
cat("\nPerformance Comparison Across Stocks:\n")
```

```
##
## Performance Comparison Across Stocks:
```

```r
print(performance_comparison$metrics)
```

```
##      Stock        RMSE         MAE    sMAPE
## AAPL  AAPL 0.007707057 0.003359324 2.266057
## AMZN  AMZN 0.007827621 0.004171797 2.345391
## MSFT  MSFT 0.005930075 0.003202745 3.659824
## NVDA  NVDA 0.008227324 0.002684289 2.865644
```

```r
print(performance_comparison$plot)
```




```r
cat("\nFeature Importance Analysis:\n")
```

```
##
## Feature Importance Analysis:
```

```r
print(feature_analysis$importance)
```

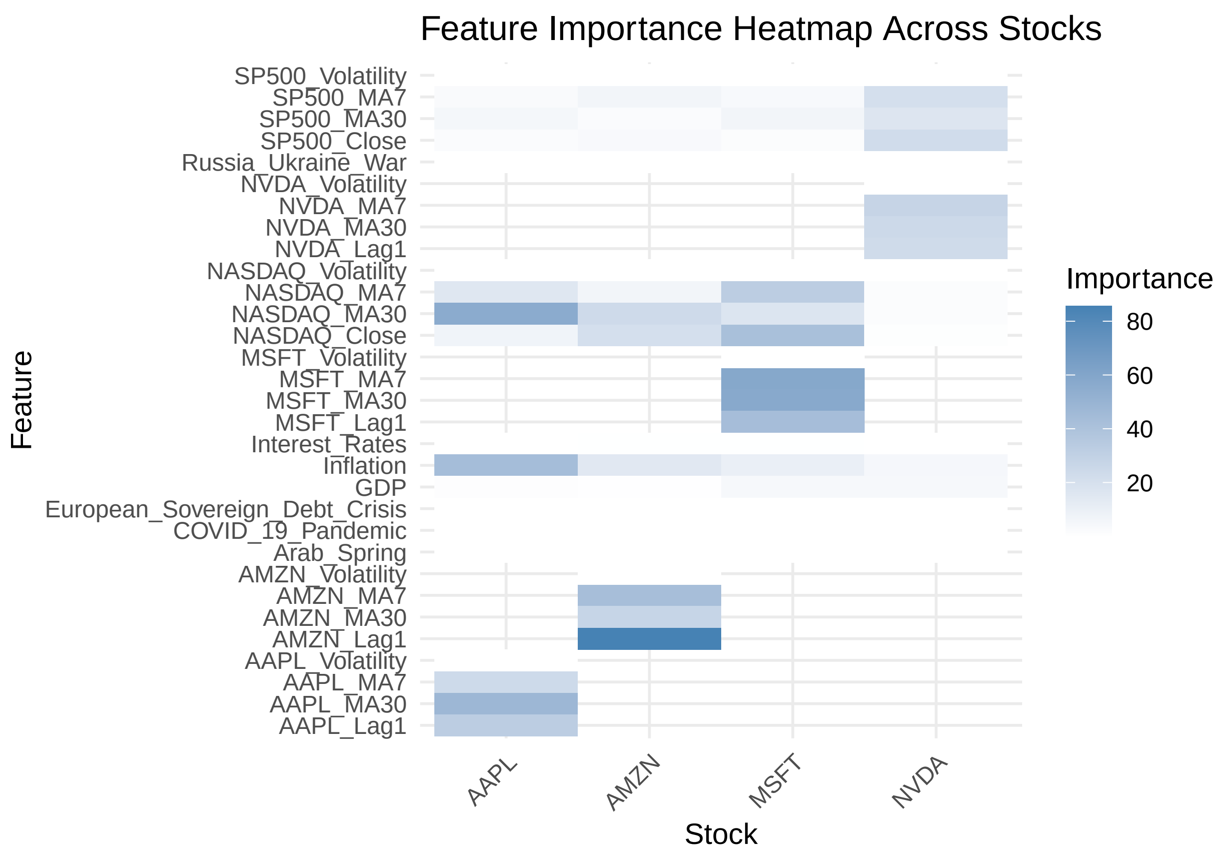
```
##                    IncNodePurity           Feature
## AAPL_MA7            2.431005e+01          AAPL_MA7
## AAPL_MA30           4.779570e+01         AAPL_MA30
## AAPL_Volatility     2.222120e-02   AAPL_Volatility
## AAPL_Lag1           3.289781e+01         AAPL_Lag1
## Interest_Rates      7.635769e-02    Interest_Rates
## Inflation           4.362816e+01         Inflation
## GDP                 1.083361e+00               GDP
## NASDAQ_Close        7.237752e+00      NASDAQ_Close
## SP500_Close         2.614787e+00       SP500_Close
## SP500_MA7           3.139022e+00         SP500_MA7
## SP500_MA30          5.250821e+00        SP500_MA30
## SP500_Volatility    2.400927e-02  SP500_Volatility
## NASDAQ_MA7          1.576890e+01        NASDAQ_MA7
## NASDAQ_MA30         5.592792e+01       NASDAQ_MA30
## NASDAQ_Volatility   1.898415e-02 NASDAQ_Volatility
## Arab_Spring         1.402879e-02       Arab_Spring
```

```
## COVID_19_Pandemic                 1.285733e-01             COVID_19_Pandemic
## Russia_Ukraine_War                1.407891e-04            Russia_Ukraine_War
## European_Sovereign_Debt_Crisis    2.088092e-04 European_Sovereign_Debt_Crisis
## AMZN_MA7                          4.275674e+01                      AMZN_MA7
## AMZN_MA30                         2.761771e+01                     AMZN_MA30
## AMZN_Volatility                   3.369960e-02               AMZN_Volatility
## AMZN_Lag1                         8.547265e+01                     AMZN_Lag1
## Interest_Rates1                   2.546917e-01                Interest_Rates
## Inflation1                        1.473820e+01                     Inflation
## GDP1                              5.072488e-01                           GDP
## NASDAQ_Close1                     2.081131e+01                  NASDAQ_Close
## SP500_Close1                      3.601858e+00                   SP500_Close
## SP500_MA71                        6.497506e+00                     SP500_MA7
## SP500_MA301                       2.275681e+00                    SP500_MA30
## SP500_Volatility1                 3.461901e-02              SP500_Volatility
## NASDAQ_MA71                       6.335889e+00                    NASDAQ_MA7
## NASDAQ_MA301                      2.392632e+01                   NASDAQ_MA30
## NASDAQ_Volatility1                4.212127e-02             NASDAQ_Volatility
## Arab_Spring1                      1.912785e-05                   Arab_Spring
## COVID_19_Pandemic1                1.827638e-01             COVID_19_Pandemic
## Russia_Ukraine_War1               8.491211e-03            Russia_Ukraine_War
## European_Sovereign_Debt_Crisis1   1.154631e-02 European_Sovereign_Debt_Crisis
## MSFT_MA7                          5.855364e+01                      MSFT_MA7
## MSFT_MA30                         5.760079e+01                     MSFT_MA30
## MSFT_Volatility                   1.886587e-02               MSFT_Volatility
## MSFT_Lag1                         4.347215e+01                     MSFT_Lag1
## Interest_Rates2                   3.027763e-01                Interest_Rates
## Inflation2                        1.033710e+01                     Inflation
## GDP2                              4.239359e+00                           GDP
## NASDAQ_Close2                     4.186093e+01                  NASDAQ_Close
## SP500_Close2                      1.844496e+00                   SP500_Close
## SP500_MA72                        3.836442e+00                     SP500_MA7
## SP500_MA302                       6.251539e+00                    SP500_MA30
## SP500_Volatility2                 3.938167e-02              SP500_Volatility
## NASDAQ_MA72                       3.279563e+01                    NASDAQ_MA7
## NASDAQ_MA302                      1.690755e+01                   NASDAQ_MA30
## NASDAQ_Volatility2                2.811404e-02             NASDAQ_Volatility
## Arab_Spring2                      1.195424e-03                   Arab_Spring
## COVID_19_Pandemic2                1.705957e-03             COVID_19_Pandemic
## Russia_Ukraine_War2               3.850333e-03            Russia_Ukraine_War
## European_Sovereign_Debt_Crisis2   5.946188e-02 European_Sovereign_Debt_Crisis
## NVDA_MA7                          2.803190e+01                      NVDA_MA7
## NVDA_MA30                         2.511539e+01                     NVDA_MA30
## NVDA_Volatility                   6.989336e-02               NVDA_Volatility
## NVDA_Lag1                         2.359413e+01                     NVDA_Lag1
## Interest_Rates3                   1.043357e-01                Interest_Rates
## Inflation3                        4.962530e+00                     Inflation
## GDP3                              4.421731e+00                           GDP
## NASDAQ_Close3                     9.231906e-01                  NASDAQ_Close
## SP500_Close3                      2.300818e+01                   SP500_Close
## SP500_MA73                        2.073735e+01                     SP500_MA7
## SP500_MA303                       1.673181e+01                    SP500_MA30
## SP500_Volatility3                 3.596806e-02              SP500_Volatility
## NASDAQ_MA73                       1.870645e+00                    NASDAQ_MA7
```

```
## NASDAQ_MA303                     2.063005e+00                    NASDAQ_MA30
## NASDAQ_Volatility3               3.807415e-02              NASDAQ_Volatility
## Arab_Spring3                     6.276233e-09                    Arab_Spring
## COVID_19_Pandemic3               1.053530e-03              COVID_19_Pandemic
## Russia_Ukraine_War3              3.102243e-04             Russia_Ukraine_War
## European_Sovereign_Debt_Crisis3  1.080690e-03 European_Sovereign_Debt_Crisis
##                                 Stock
## AAPL_MA7                         AAPL
## AAPL_MA30                        AAPL
## AAPL_Volatility                  AAPL
## AAPL_Lag1                        AAPL
## Interest_Rates                   AAPL
## Inflation                        AAPL
## GDP                              AAPL
## NASDAQ_Close                     AAPL
## SP500_Close                      AAPL
## SP500_MA7                        AAPL
## SP500_MA30                       AAPL
## SP500_Volatility                 AAPL
## NASDAQ_MA7                       AAPL
## NASDAQ_MA30                      AAPL
## NASDAQ_Volatility                AAPL
## Arab_Spring                      AAPL
## COVID_19_Pandemic                AAPL
## Russia_Ukraine_War               AAPL
## European_Sovereign_Debt_Crisis   AAPL
## AMZN_MA7                         AMZN
## AMZN_MA30                        AMZN
## AMZN_Volatility                  AMZN
## AMZN_Lag1                        AMZN
## Interest_Rates1                  AMZN
## Inflation1                       AMZN
## GDP1                             AMZN
## NASDAQ_Close1                    AMZN
## SP500_Close1                     AMZN
## SP500_MA71                       AMZN
## SP500_MA301                      AMZN
## SP500_Volatility1                AMZN
## NASDAQ_MA71                      AMZN
## NASDAQ_MA301                     AMZN
## NASDAQ_Volatility1               AMZN
## Arab_Spring1                     AMZN
## COVID_19_Pandemic1               AMZN
## Russia_Ukraine_War1              AMZN
## European_Sovereign_Debt_Crisis1  AMZN
## MSFT_MA7                         MSFT
## MSFT_MA30                        MSFT
## MSFT_Volatility                  MSFT
## MSFT_Lag1                        MSFT
## Interest_Rates2                  MSFT
## Inflation2                       MSFT
## GDP2                             MSFT
## NASDAQ_Close2                    MSFT
## SP500_Close2                     MSFT
```

```
## SP500_MA72                         MSFT
## SP500_MA302                        MSFT
## SP500_Volatility2                  MSFT
## NASDAQ_MA72                        MSFT
## NASDAQ_MA302                       MSFT
## NASDAQ_Volatility2                 MSFT
## Arab_Spring2                       MSFT
## COVID_19_Pandemic2                 MSFT
## Russia_Ukraine_War2                MSFT
## European_Sovereign_Debt_Crisis2    MSFT
## NVDA_MA7                           NVDA
## NVDA_MA30                          NVDA
## NVDA_Volatility                    NVDA
## NVDA_Lag1                          NVDA
## Interest_Rates3                    NVDA
## Inflation3                         NVDA
## GDP3                               NVDA
## NASDAQ_Close3                      NVDA
## SP500_Close3                       NVDA
## SP500_MA73                         NVDA
## SP500_MA303                        NVDA
## SP500_Volatility3                  NVDA
## NASDAQ_MA73                        NVDA
## NASDAQ_MA303                       NVDA
## NASDAQ_Volatility3                 NVDA
## Arab_Spring3                       NVDA
## COVID_19_Pandemic3                 NVDA
## Russia_Ukraine_War3                NVDA
## European_Sovereign_Debt_Crisis3    NVDA
```

```
print(feature_analysis$plot)
```



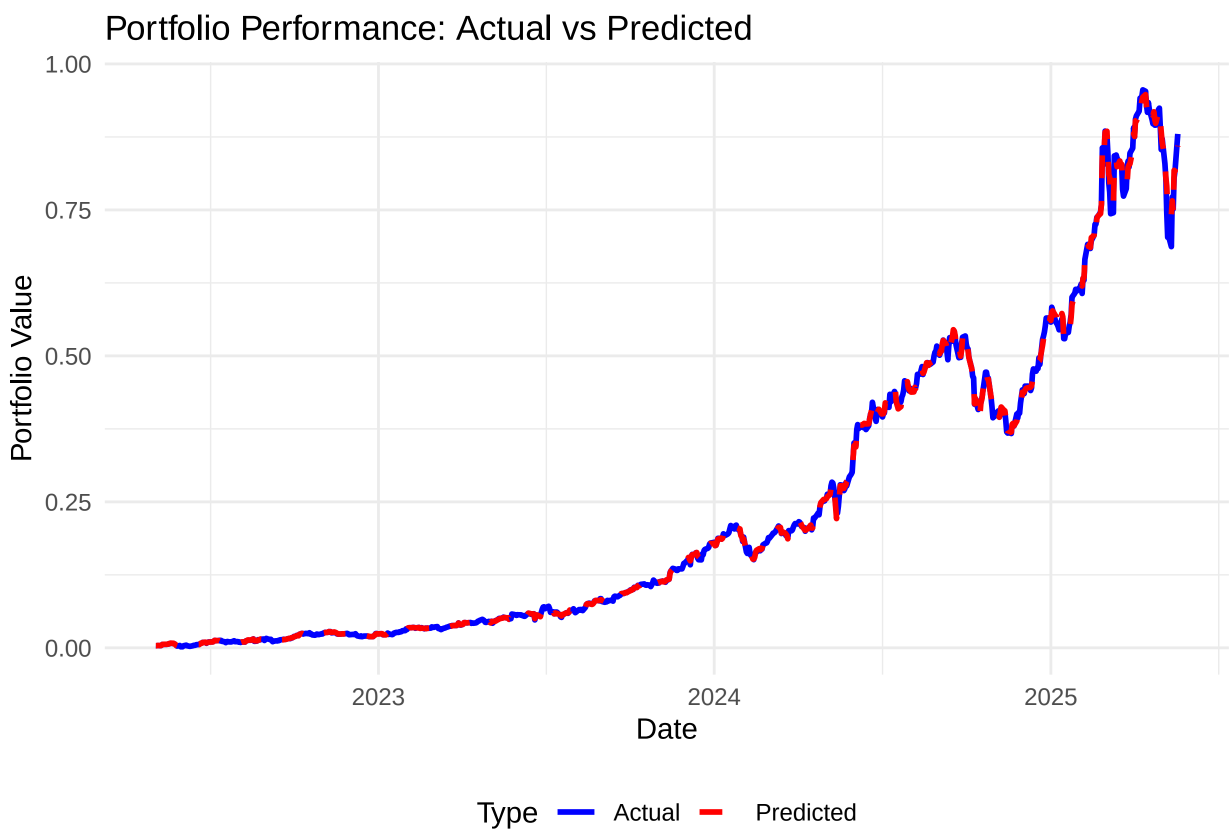

```r
cat("\nPortfolio Analysis:\n")
```

```
##
## Portfolio Analysis:
```

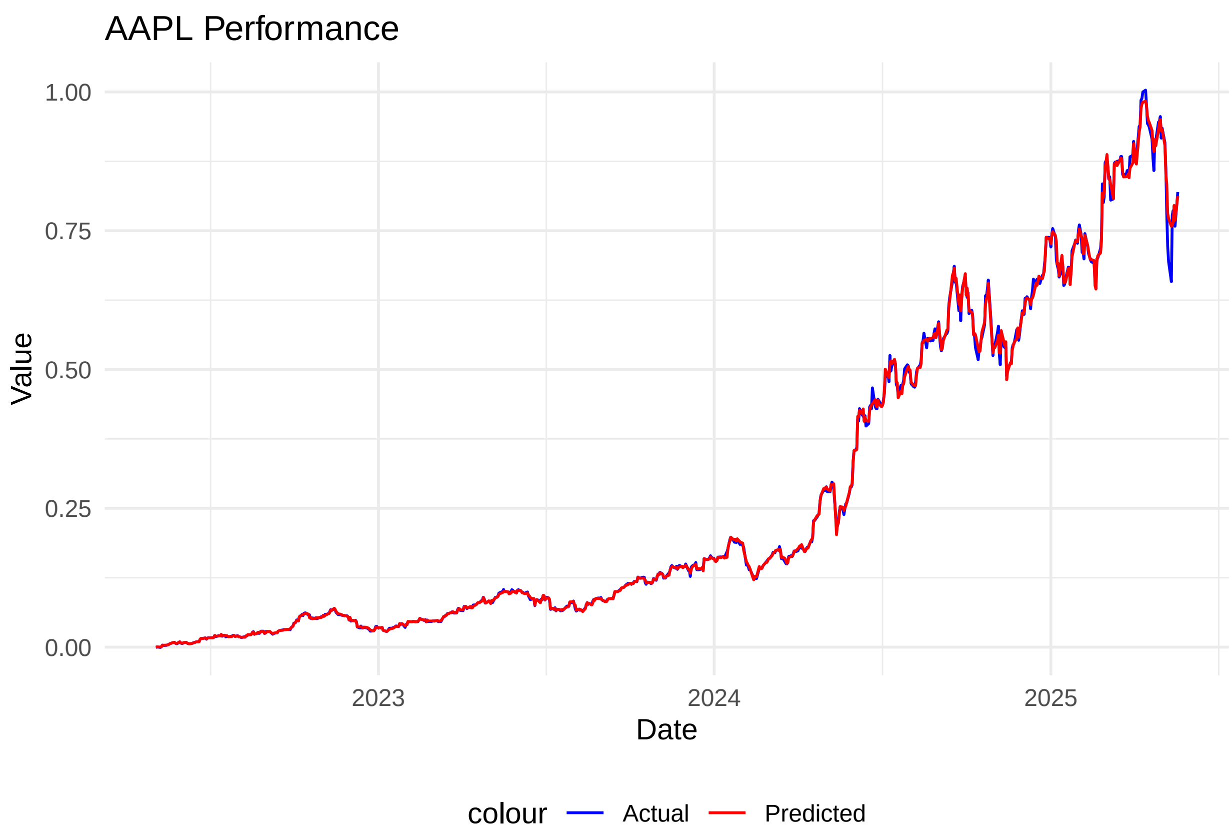
```r
print(portfolio_results$metrics)
```

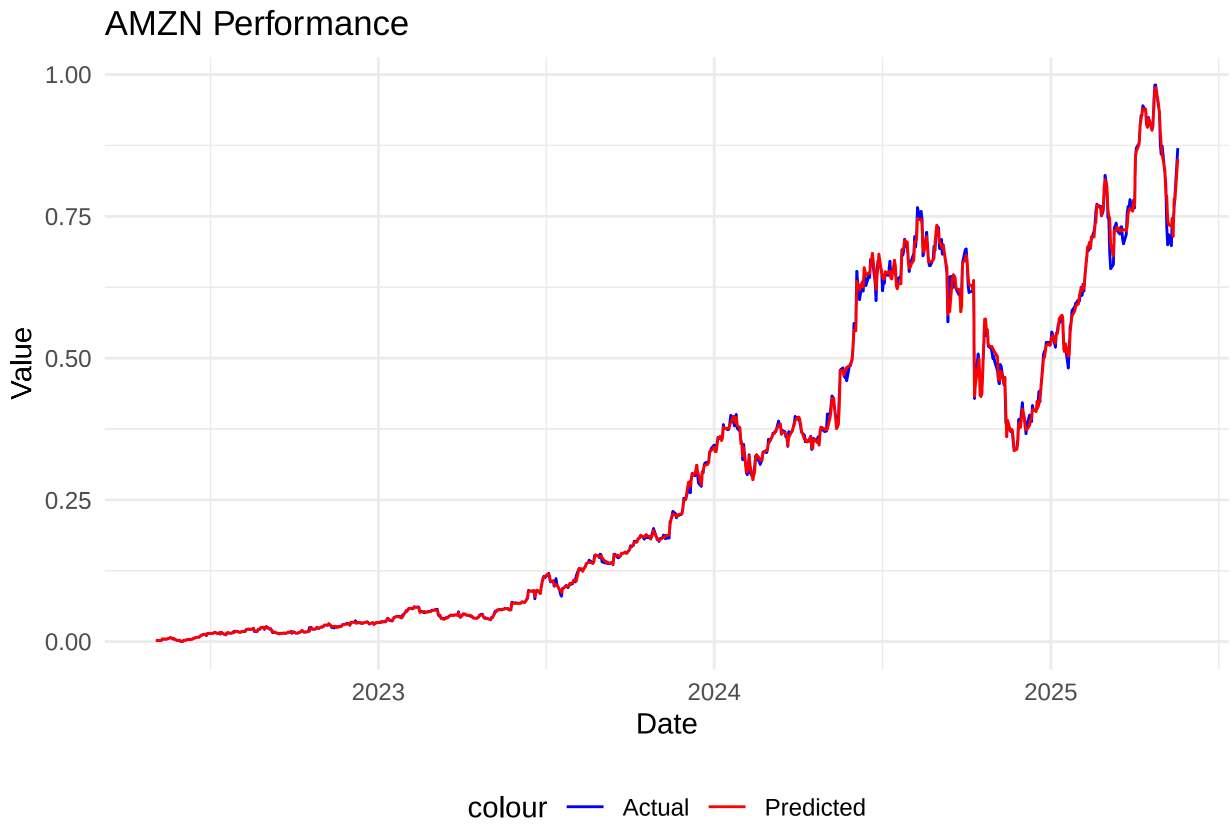
```
## $RMSE
## [1] 0.005354143
##
## $MAE
## [1] 0.002318068
##
## $sMAPE
## [1] 1.37692
```

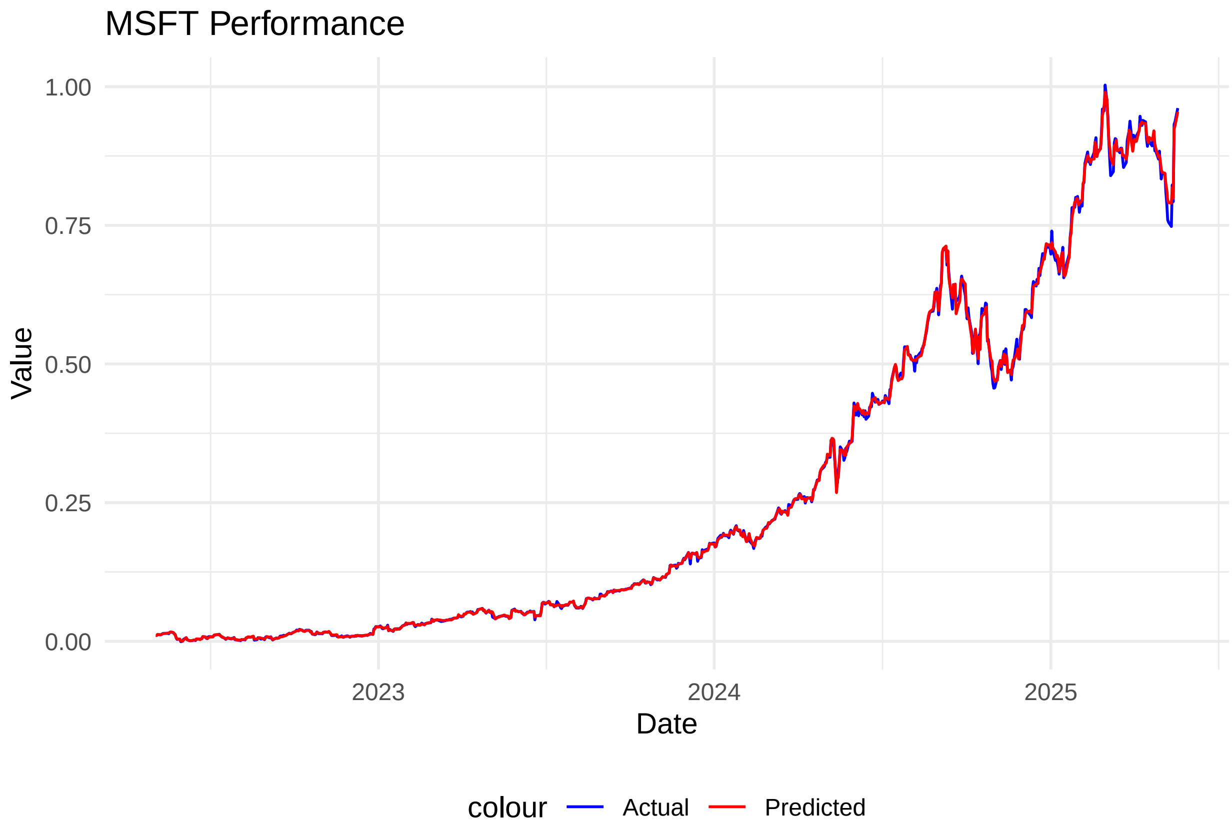
```r
print(portfolio_results$portfolio_plot)
```

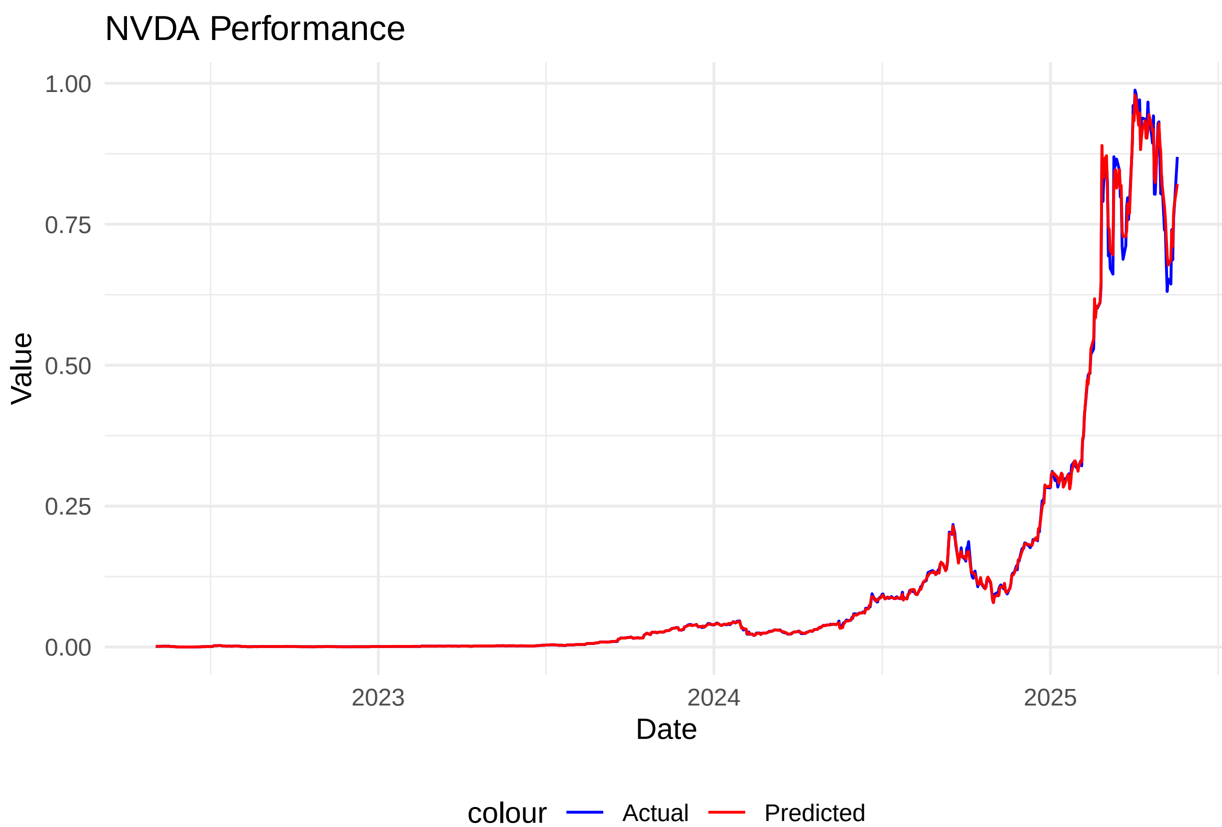



```r
# Display individual stock plots
for(symbol in stock_symbols) {
    print(portfolio_results$stock_plots[[symbol]])
}
```


AAPL Performance
1.00
0.75
0.50
0.25
0.00
Value
2023
2024
2025
Date
colour
Actual
Predicted


AMZN Performance
1.00
0.75
0.50
0.25
0.00
Value
2023
2024
2025
Date
colour
Actual
Predicted


MSFT Performance
1.00
0.75
0.50
0.25
0.00
Value
2023
2024
2025
Date
colour
Actual
Predicted




```r
# Print portfolio weights
cat("\nPortfolio Weights:\n")
```

```
##
## Portfolio Weights:
```

```r
print(portfolio_results$weights)
```

```
## AAPL AMZN MSFT NVDA
## 0.25 0.25 0.25 0.25
```

Feature and Model Building for Stocks Returns

We are calculating various indicators to predict returns. Here are some:

- Price Momentum: Help to understand the price movements like short-term, medium and long-term.
- RSI (Relative Strength Index): Measures momentum by comparing the magnitude of recent gains to recent losses, Shows if a stock is overbought or oversold
- RSMA(Relative Moving Average): Smooths out RSI fluctuations, Reduces impact of outliers, More reliable trend identification
- MACD (Moving Average Convergence Divergence):Shows relationship between two moving averages

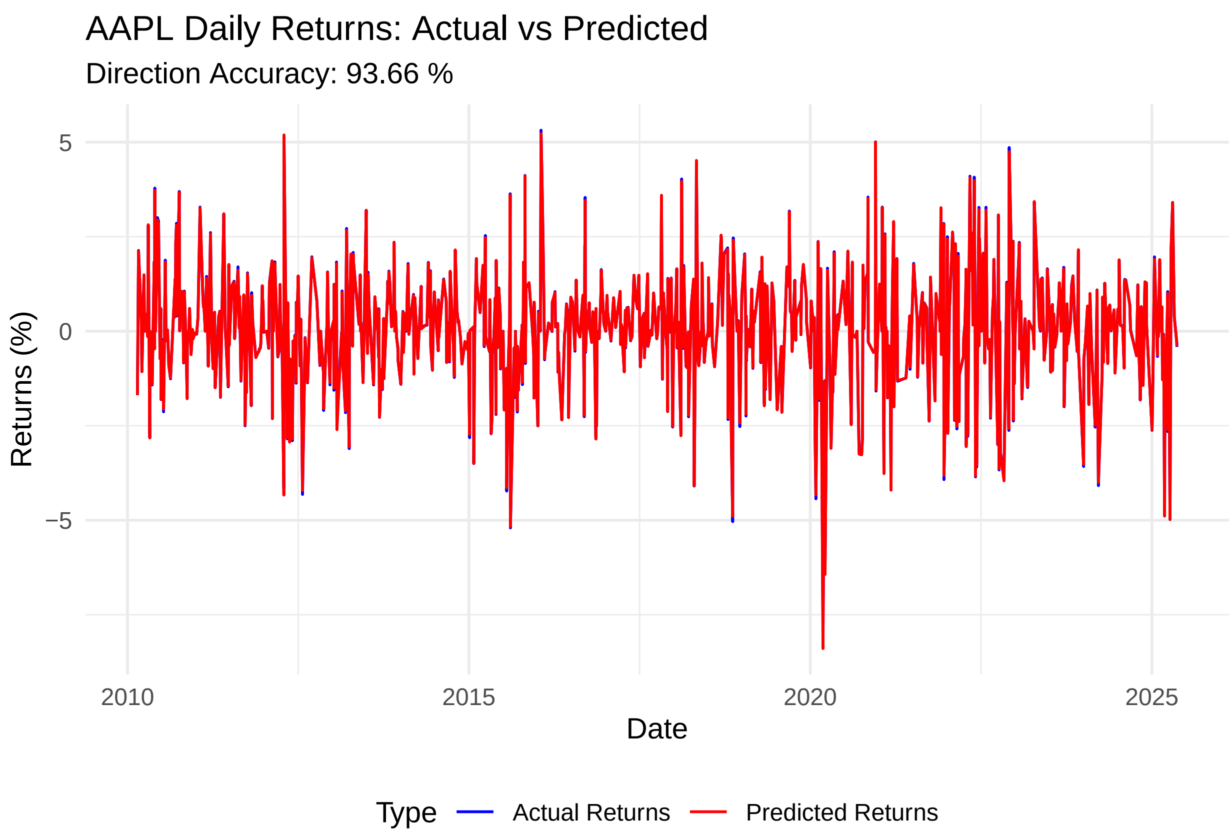
```r
# Function to create features for any stock
create_stock_features <- function(data, symbol) {
    data %>%
        mutate(
            # Price Momentum
            !!paste0(symbol, "_Return_1D") := (get(paste0(symbol, "_Close"))/lag(get(paste0(symbol, "_Cl
            !!paste0(symbol, "_Return_3D") := (get(paste0(symbol, "_Close"))/lag(get(paste0(symbol, "_Cl
            !!paste0(symbol, "_Return_5D") := (get(paste0(symbol, "_Close"))/lag(get(paste0(symbol, "_Cl
            !!paste0(symbol, "_Return_10D") := (get(paste0(symbol, "_Close"))/lag(get(paste0(symbol, "_C
            !!paste0(symbol, "_Return_20D") := (get(paste0(symbol, "_Close"))/lag(get(paste0(symbol, "_C

            # Technical Indicators
            !!paste0(symbol, "_RSI") := RSI(get(paste0(symbol, "_Close")), n = 14),
            !!paste0(symbol, "_RSI_MA") := SMA(RSI(get(paste0(symbol, "_Close")), n = 14), n = 3),
            !!paste0(symbol, "_MACD") := MACD(get(paste0(symbol, "_Close")))[, "macd"],
            !!paste0(symbol, "_Signal") := MACD(get(paste0(symbol, "_Close")))[, "signal"],
            !!paste0(symbol, "_MACD_Hist") := MACD(get(paste0(symbol, "_Close")))[, "macd"] -
                                          MACD(get(paste0(symbol, "_Close")))[, "signal"],

            # Moving Averages
            !!paste0(symbol, "_MA7") := rollmean(get(paste0(symbol, "_Close")), 7, fill = NA, align = "r
            !!paste0(symbol, "_MA20") := rollmean(get(paste0(symbol, "_Close")), 20, fill = NA, align =
            !!paste0(symbol, "_MA30") := rollmean(get(paste0(symbol, "_Close")), 30, fill = NA, align =

            # Volatility Measures
            !!paste0(symbol, "_Vol_5D") := rollapply(get(paste0(symbol, "_Returns")), 5, sd, fill = NA,
            !!paste0(symbol, "_Vol_10D") := rollapply(get(paste0(symbol, "_Returns")), 10, sd, fill = NA
            !!paste0(symbol, "_Vol_22D") := rollapply(get(paste0(symbol, "_Returns")), 22, sd, fill = NA

            # Relative Performance (Shows if stock moves with or against market,Identifies sector-speci
            !!paste0(symbol, "_vs_SP500") := get(paste0(symbol, "_Returns")) - SP500_Returns,
            !!paste0(symbol, "_vs_NASDAQ") := get(paste0(symbol, "_Returns")) - NASDAQ_Returns,
            !!paste0(symbol, "_Relative_Strength") := (get(paste0(symbol, "_Close"))/lag(get(paste0(symb
                                                       (SP500_Close/lag(SP500_Close, 20))
        )
}

# Function to create features list for a stock
get_stock_features <- function(symbol) {
    c(
        paste0(symbol, "_Returns_Lag", 1:10),
        paste0(symbol, "_Return_", c("1D", "3D", "5D", "10D", "20D")),
        paste0(symbol, "_", c("RSI", "RSI_MA", "MACD", "Signal", "MACD_Hist")),
        paste0(symbol, "_MA", c(7, 20, 30)),
        paste0(symbol, "_Vol_", c("5D", "10D", "22D")),
        paste0(symbol, "_vs_", c("SP500", "NASDAQ")),
        paste0(symbol, "_Relative_Strength")
    )
}

# Function to predict stock returns
predict_stock_returns <- function(data, symbol) {
    # Create lagged returns
```

```r
for(i in 1:10) {
    data[[paste0(symbol, "_Returns_Lag", i)]] <- lag(data[[paste0(symbol, "_Returns")]], i)
}

# Get features for this stock
stock_features <- c(
    get_stock_features(symbol),
    "SP500_Returns", "NASDAQ_Returns", "Market_Vol",
    paste0("SP500_Returns_Lag", 1:10),
    paste0("NASDAQ_Returns_Lag", 1:10),
    "Arab_Spring", "COVID_19_Pandemic", "Russia_Ukraine_War",
    "European_Sovereign_Debt_Crisis"
)

# Split data
set.seed(123)
train_index <- createDataPartition(data[[paste0(symbol, "_Returns")]], p = 0.8, list = FALSE)
train_data <- data[train_index, ]
test_data <- data[-train_index, ]

# Scale features
scaler <- preProcess(train_data[, stock_features], method = c("center", "scale"))
train_scaled <- predict(scaler, train_data)
test_scaled <- predict(scaler, test_data)

# Prepare matrices
train_matrix <- as.matrix(train_scaled[, stock_features])
test_matrix <- as.matrix(test_scaled[, stock_features])

# Train model
xgb_params <- list(
    objective = "reg:squarederror",
    max_depth = 6,
    eta = 0.03,
    subsample = 0.8,
    colsample_bytree = 0.8,
    min_child_weight = 3,
    gamma = 0.1
)

model <- xgboost(
    data = train_matrix,
    label = train_scaled[[paste0(symbol, "_Returns")]],
    params = xgb_params,
    nrounds = 1000,
    early_stopping_rounds = 50,
    eval_metric = "rmse",
    verbose = 0
)

# Make predictions
predictions <- predict(model, test_matrix)
```

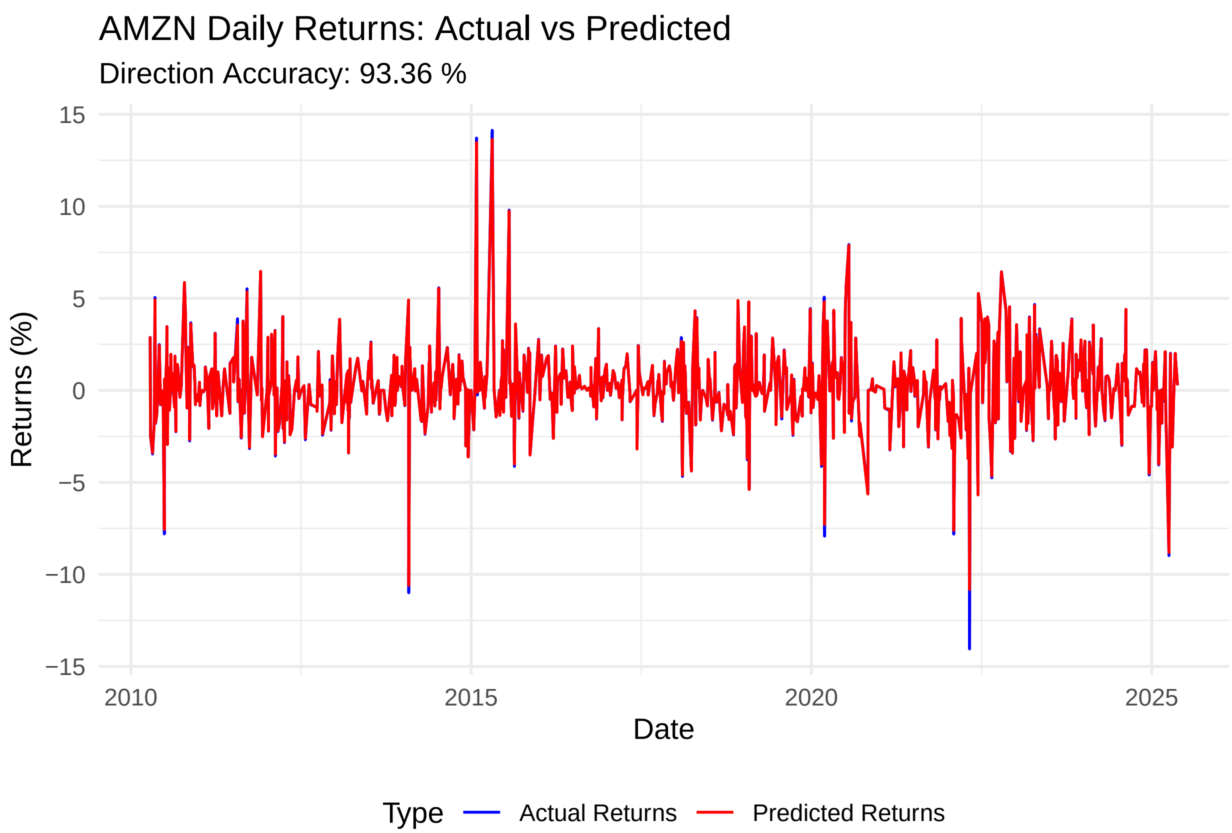
```r
    # Calculate metrics
    # R2: Measures how well predictions match actual returns
    # Direction Accuracy: Measures how often model predicts correct price movement direction
    metrics <- list(
        RMSE = sqrt(mean((predictions - test_scaled[[paste0(symbol, "_Returns")]])^2)),
        MAE = mean(abs(predictions - test_scaled[[paste0(symbol, "_Returns")]])),
        R2 = cor(predictions, test_scaled[[paste0(symbol, "_Returns")]])^2,
        Direction_Accuracy = mean(sign(predictions) == sign(test_scaled[[paste0(symbol, "_Returns")]]))
    )

    # Create plot
    plot_data <- data.frame(
        Date = test_data$Date,
        Actual = test_scaled[[paste0(symbol, "_Returns")]],
        Predicted = predictions
    )

    plot <- ggplot(plot_data, aes(x = Date)) +
        geom_line(aes(y = Actual, color = "Actual Returns")) +
        geom_line(aes(y = Predicted, color = "Predicted Returns")) +
        labs(title = paste(symbol, "Daily Returns: Actual vs Predicted"),
             subtitle = paste("Direction Accuracy:",
                              round(metrics$Direction_Accuracy * 100, 2), "%"),
             x = "Date",
             y = "Returns (%)",
             color = "Type") +
        scale_color_manual(values = c("Actual Returns" = "blue",
                                      "Predicted Returns" = "red")) +
        theme_minimal() +
        theme(legend.position = "bottom")

    return(list(
        model = model,
        metrics = metrics,
        predictions = predictions,
        actual = test_scaled[[paste0(symbol, "_Returns")]],
        plot = plot,
        importance = xgb.importance(feature_names = stock_features, model = model)
    ))
}

# Process all stocks
stock_symbols <- c("AAPL", "AMZN", "MSFT", "NVDA")
results <- list()

# Create market features first
merged_data_df <- merged_data_df %>%
    mutate(
       # SP500_Returns = (SP500_Close/lag(SP500_Close) - 1) * 100,
        # Nasdaq_Returns = (NASDAQ_Close/lag(NASDAQ_Close) - 1) * 100,
        Market_Vol = rollapply(SP500_Returns, 10, sd, fill = NA, align = "right")
    )
```

```r
# Create market lags
for(i in 1:10) {
    merged_data_df[[paste0("SP500_Returns_Lag", i)]] <- lag(merged_data_df$SP500_Returns, i)
    merged_data_df[[paste0("NASDAQ_Returns_Lag", i)]] <- lag(merged_data_df$NASDAQ_Returns, i)
}

#names(merged_data_df)

# Process each stock
for(symbol in stock_symbols) {
    cat("\nProcessing", symbol, "...\n")

    # Create features
    merged_data_df <- create_stock_features(merged_data_df, symbol)

    # Remove NAs
    merged_data_df <- na.omit(merged_data_df)

    # Predict returns
    results[[symbol]] <- predict_stock_returns(merged_data_df, symbol)

    # Print metrics
    cat("\nMetrics for", symbol, ":\n")
    print(results[[symbol]]$metrics)

    # Display plot
    print(results[[symbol]]$plot)
}
```

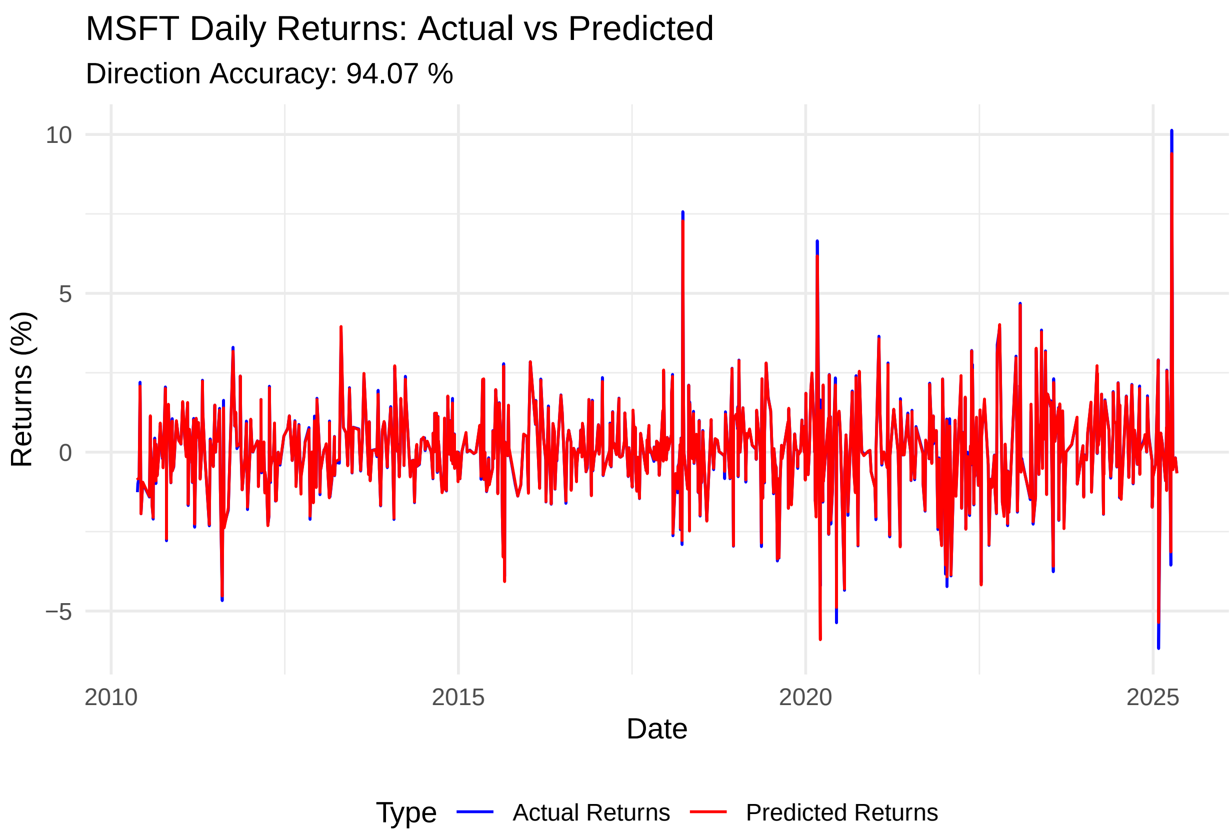
```
##
## Processing AAPL ...
##
## Metrics for AAPL :
## $RMSE
## [1] 0.05610808
##
## $MAE
## [1] 0.02295264
##
## $R2
## [1] 0.9987548
##
## $Direction_Accuracy
## [1] 0.9365672
```



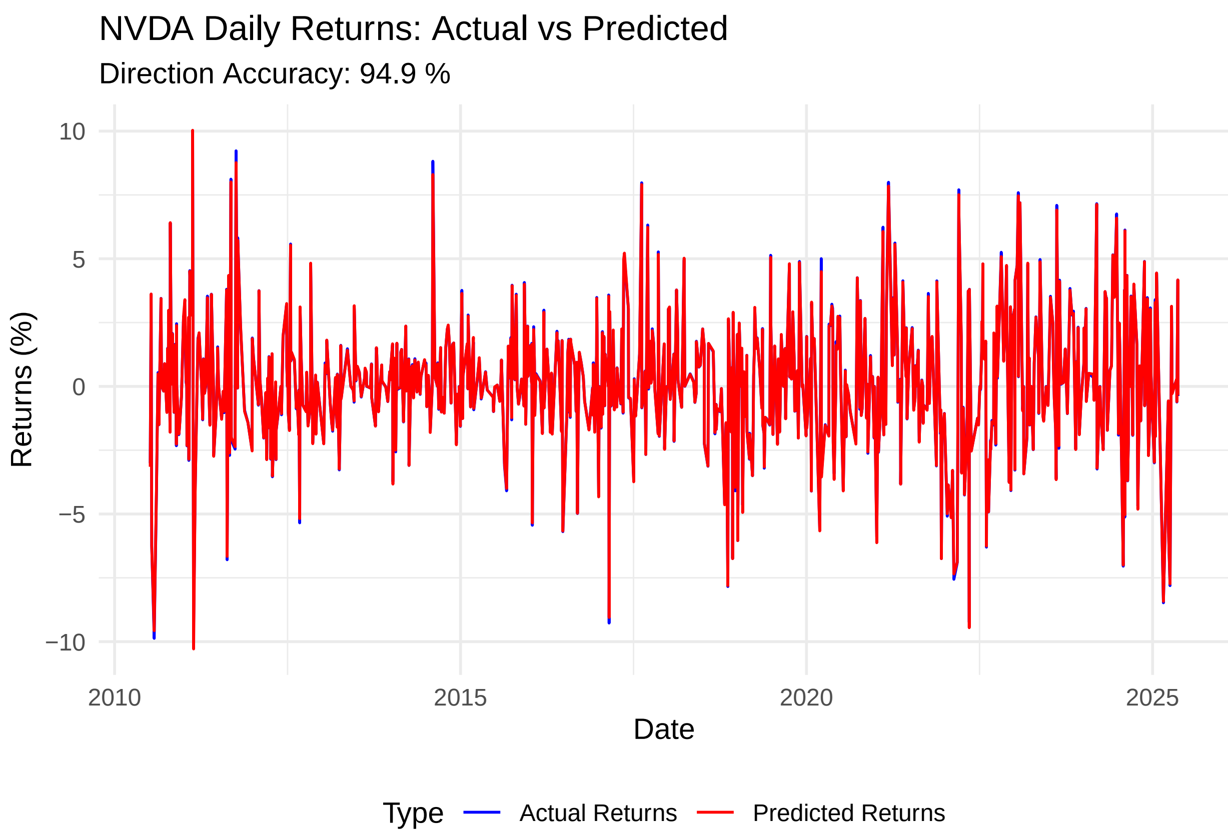

```
##
## Processing AMZN ...
##
## Metrics for AMZN :
## $RMSE
## [1] 0.1251896
##
## $MAE
## [1] 0.02862972
##
## $R2
## [1] 0.9968882
##
## $Direction_Accuracy
## [1] 0.933584
```




```
##
## Processing MSFT ...
##
## Metrics for MSFT :
## $RMSE
## [1] 0.09631905
##
## $MAE
## [1] 0.0403319
##
## $R2
## [1] 0.9956032
##
## $Direction_Accuracy
## [1] 0.9406566
```




```
##
## Processing NVDA ...
##
## Metrics for NVDA :
## $RMSE
## [1] 0.07570507
##
## $MAE
## [1] 0.03886412
##
## $R2
## [1] 0.9991426
##
## $Direction_Accuracy
## [1] 0.9489796
```




```r
# Compare performance across stocks
performance_df <- data.frame(
    Stock = stock_symbols,
    Direction_Accuracy = sapply(results, function(x) x$metrics$Direction_Accuracy),
    RMSE = sapply(results, function(x) x$metrics$RMSE),
    R2 = sapply(results, function(x) x$metrics$R2)
)

print("\nPerformance Comparison:")
```

```
## [1] "\nPerformance Comparison:"
```

```r
print(performance_df)
```

```
##      Stock Direction_Accuracy       RMSE        R2
## AAPL  AAPL          0.9365672 0.05610808 0.9987548
## AMZN  AMZN          0.9335840 0.12518964 0.9968882
## MSFT  MSFT          0.9406566 0.09631905 0.9956032
## NVDA  NVDA          0.9489796 0.07570507 0.9991426
```

```r
# Functions for analyzing returns predictions
# 1. Compare Performance Metrics Across Stocks
```

```r
compare_returns_performance <- function(results, stock_symbols) {
    # Combine metrics for all stocks
    metrics_df <- data.frame(
        Stock = stock_symbols,
        Direction_Accuracy = sapply(results, function(x) x$metrics$Direction_Accuracy * 100),
        RMSE = sapply(results, function(x) x$metrics$RMSE),
        R2 = sapply(results, function(x) x$metrics$R2)
    )

    # Create comparison plots
    metrics_long <- tidyr::pivot_longer(metrics_df,
                                        cols = c("Direction_Accuracy", "RMSE", "R2"),
                                        names_to = "Metric",
                                        values_to = "Value")

    comparison_plot <- ggplot(metrics_long, aes(x = Stock, y = Value, fill = Stock)) +
        geom_bar(stat = "identity") +
        facet_wrap(~Metric, scales = "free_y") +
        labs(title = "Returns Prediction Performance Across Stocks",
             y = "Value") +
        theme_minimal() +
        theme(axis.text.x = element_text(angle = 45, hjust = 1))

    return(list(metrics = metrics_df, plot = comparison_plot))
}

# 2. Feature Importance Analysis
analyze_returns_importance <- function(results, stock_symbols) {
    # Extract and combine feature importance as before
    importance_list <- lapply(stock_symbols, function(symbol) {
        imp <- as.data.frame(results[[symbol]]$importance)
        imp$Stock <- symbol
        return(imp)
    })

    importance_df <- do.call(rbind, importance_list)

    # Get top features
    top_features <- importance_df %>%
        group_by(Feature) %>%
        summarise(avg_importance = mean(Gain)) %>%
        top_n(15, avg_importance) %>%
        pull(Feature)

    importance_df_filtered <- importance_df %>%
        filter(Feature %in% top_features)

    # Create improved heatmap
    importance_plot <- ggplot(importance_df_filtered,
                              aes(x = Stock, y = reorder(Feature, Gain))) +
        geom_tile(aes(fill = Gain)) +
        scale_fill_gradient(low = "white", high = "steelblue") +
        labs(title = "Top Features Importance Heatmap",
```

```
            y = "Feature",
            x = "Stock",
            fill = "Importance") +
        theme_minimal() +
        theme(
            axis.text.x = element_text(angle = 45, hjust = 1),
            axis.text.y = element_text(size = 8),  # Adjust text size
            panel.grid.major = element_blank(),    # Remove grid lines
            panel.grid.minor = element_blank(),
            axis.text = element_text(color = "black"), # Make text darker
            plot.margin = unit(c(1, 1, 1, 2), "cm")   # Fixed margin syntax
        )

    return(list(importance = importance_df_filtered, plot = importance_plot))
}
# 3. Portfolio Returns Analysis
analyze_portfolio_returns <- function(results, stock_symbols) {
    # Get the shortest length among all results to ensure consistency
    min_length <- min(sapply(results, function(x) length(x$actual)))

    # Create portfolio returns dataframe
    portfolio_df <- data.frame(
        Portfolio_Actual = rep(0, min_length),
        Portfolio_Predicted = rep(0, min_length)
    )

    # Equal weights
    weights <- rep(1/length(stock_symbols), length(stock_symbols))
    names(weights) <- stock_symbols

    # Combine returns
    for(symbol in stock_symbols) {
        # Take only the minimum length of data
        portfolio_df[[paste0(symbol, "_Actual")]] <- results[[symbol]]$actual[1:min_length]
        portfolio_df[[paste0(symbol, "_Predicted")]] <- results[[symbol]]$predictions[1:min_length]

        portfolio_df$Portfolio_Actual <- portfolio_df$Portfolio_Actual +
            results[[symbol]]$actual[1:min_length] * weights[symbol]
        portfolio_df$Portfolio_Predicted <- portfolio_df$Portfolio_Predicted +
            results[[symbol]]$predictions[1:min_length] * weights[symbol]
    }

    # Add dates (take from first stock's data)
    portfolio_df$Date <- tail(merged_data_df$Date, min_length)

    # Calculate metrics
    metrics <- list(
        RMSE = sqrt(mean((portfolio_df$Portfolio_Predicted - portfolio_df$Portfolio_Actual)^2)),
        Direction_Accuracy = mean(sign(portfolio_df$Portfolio_Predicted) ==
                                  sign(portfolio_df$Portfolio_Actual)) * 100,
        R2 = cor(portfolio_df$Portfolio_Predicted, portfolio_df$Portfolio_Actual)^2
    )
```

```r
    # Create portfolio plot
    portfolio_plot <- ggplot(portfolio_df, aes(x = Date)) +
        geom_line(aes(y = Portfolio_Actual, color = "Actual Returns")) +
        geom_line(aes(y = Portfolio_Predicted, color = "Predicted Returns")) +
        labs(title = "Portfolio Returns: Actual vs Predicted",
             subtitle = paste("Direction Accuracy:",
                              round(metrics$Direction_Accuracy, 2), "%"),
             x = "Date",
             y = "Returns (%)",
             color = "Type") +
        scale_color_manual(values = c("Actual Returns" = "blue",
                                      "Predicted Returns" = "red")) +
        theme_minimal()

    return(list(
        data = portfolio_df,
        metrics = metrics,
        plot = portfolio_plot
    ))
}

# Update the analysis call
returns_performance <- compare_returns_performance(results, stock_symbols)
returns_importance <- analyze_returns_importance(results, stock_symbols)
returns_portfolio <- analyze_portfolio_returns(results, stock_symbols)  # Removed test_data parameter

# Display results
cat("\nReturns Performance Comparison Across Stocks:\n")
```

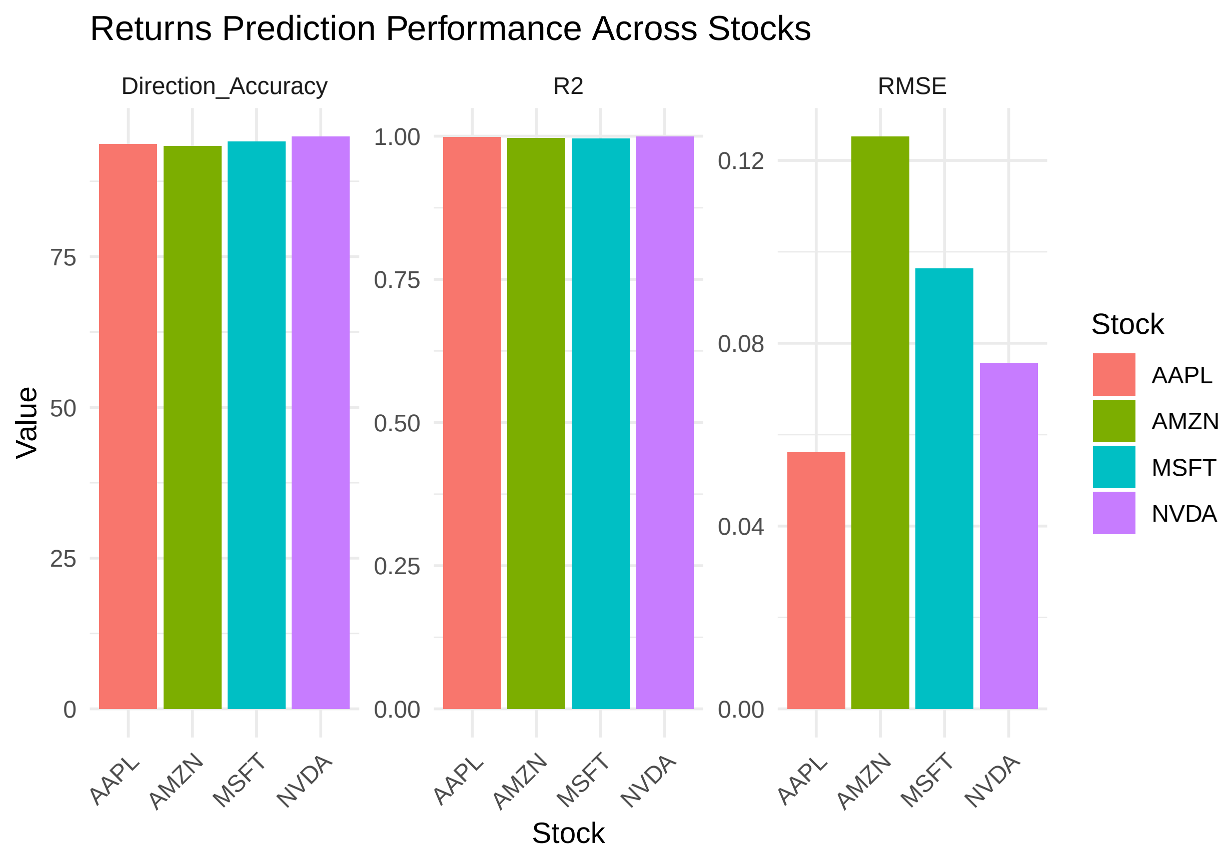
Performance Prediction for Stocks Returns

```
##
## Returns Performance Comparison Across Stocks:
```

```r
print(returns_performance$metrics)
```

```
##      Stock Direction_Accuracy       RMSE        R2
## AAPL  AAPL           93.65672 0.05610808 0.9987548
## AMZN  AMZN           93.35840 0.12518964 0.9968882
## MSFT  MSFT           94.06566 0.09631905 0.9956032
## NVDA  NVDA           94.89796 0.07570507 0.9991426
```

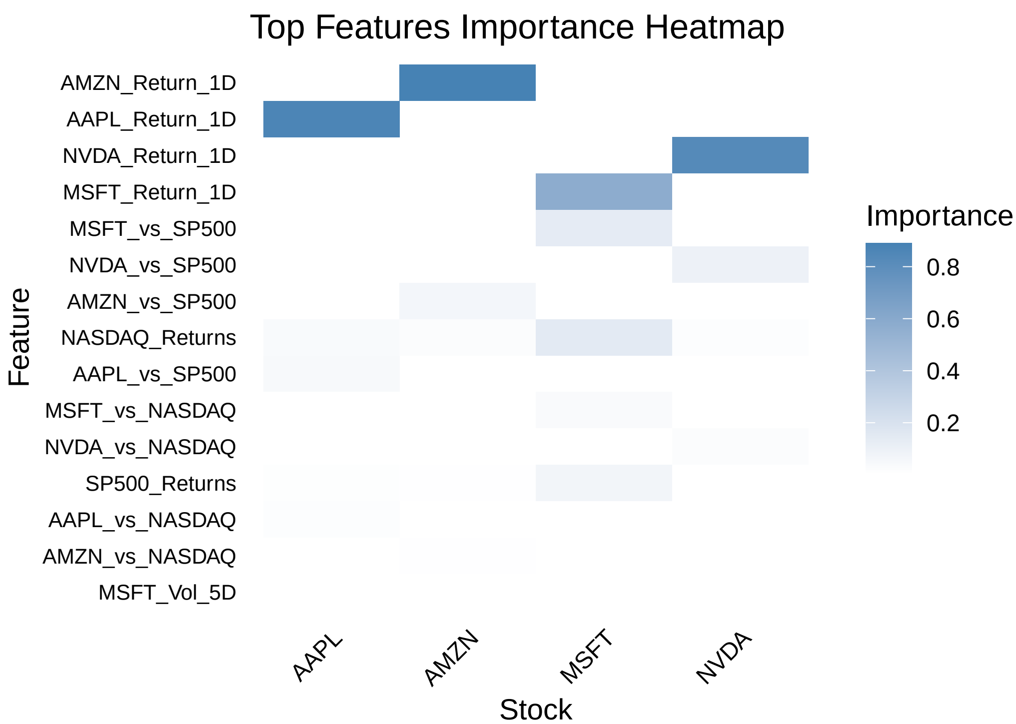
```r
print(returns_performance$plot)
```




```r
cat("\nReturns Feature Importance Analysis:\n")
```

```
##
## Returns Feature Importance Analysis:
```

```r
print(returns_importance$plot)
```




```r
cat("\nPortfolio Returns Analysis:\n")
```

```
##
## Portfolio Returns Analysis:
```

```r
print(returns_portfolio$metrics)
```

```
## $RMSE
## [1] 0.04545498
##
## $Direction_Accuracy
## [1] 99.61735
##
## $R2
## [1] 0.9980093
```

```r
print(returns_portfolio$plot)
```