Predicting NVIDIA Corporation (NVDA) Stock Prices

Project Group 1 (Mosness, Ronald)

Introduction

Background and Context:

In the fast-paced realm of financial markets, making informed decisions is paramount for investors and traders. Accurate predictions of stock prices play a pivotal role in shaping investment strategies and optimizing returns. NVIDIA Corporation (NVDA), a leading technology company, has witnessed dynamic stock price movements, creating an opportune landscape for data-driven exploration. This project delves into the intricate world of NVDA stock prices, aiming to unlock patterns, forecast trends, and provide actionable insights.

Problem Statement and Significance:

The challenge lies in navigating the volatility of NVDA's stock prices. Ascertaining patterns within this volatility can aid investors in understanding market dynamics. By employing advanced data mining and statistical learning techniques, this project seeks to unravel the underlying trends influencing NVDA's daily closing prices. The significance of this endeavor resonates not only with investors but also with researchers and analysts keen on deciphering stock market intricacies.

Objectives and Research Questions:

The primary objective of this project is to develop robust predictive models for NVDA stock prices. To achieve this goal, the following research questions are pivotal:

Data Source:

The project utilized historical NVDA stock price data obtained from Yahoo Finance. The dataset comprised daily "adjusted close" values spanning a period of 5 years, ranging from October 15, 2018, to October 13, 2023.

Research Questions:

i. Can R-based techniques accurately predict NVDA's daily closing prices?
ii. How well can NVDA's stock prices be forecasted for short-term (a week), medium-term (a month), and long-term (a quarter) horizons using R?
iii. What features or technical indicators exert significant influence on NVDA stock price movements, as analyzed in R?
iv. Are there discernible patterns or seasonality in NVDA's historical stock prices, detectable through comprehensive R-based analysis?

v. How do different R-based modeling techniques compare in terms of prediction accuracy and sensitivity?

Research Methodology:

This project leverages an arsenal of data mining techniques, encompassing time-series analysis, machine learning algorithms, and statistical modeling. By meticulously exploring historical data, this study aims to unearth patterns, validate hypotheses, and deliver actionable insights. The application of R programming, a powerful tool in data analysis, forms the backbone of our methodology.

Data Loading and Preprocessing

Data Cleaning and Transformation:

- Date Format: The date format was meticulously standardized to "yyyy-mm-dd," ensuring consistency and compatibility with the dataset.
- Missing Values: Rigorous checks were conducted to handle missing values, ensuring the dataset's integrity and completeness.
- Exploratory Analysis: Preliminary exploratory analysis offered insights into the dataset's structure, enabling a deeper understanding of the available information.

Exploratory Data Analysis (EDA) and Visualization:

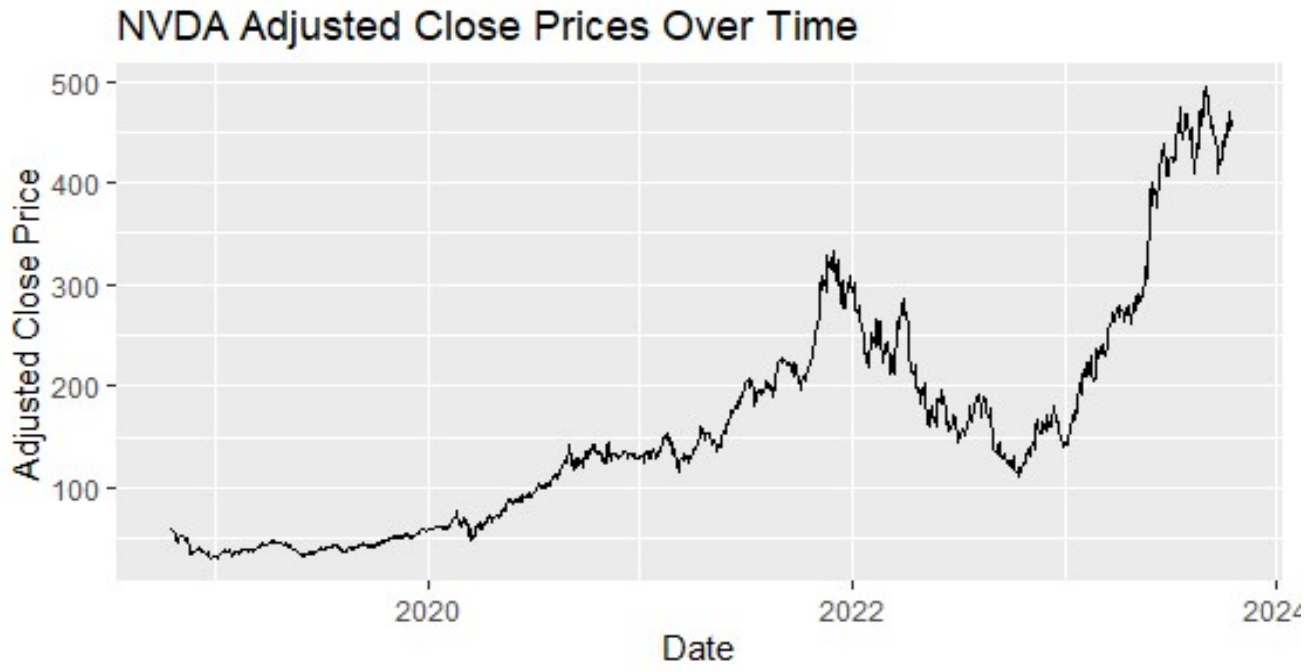
Exploratory Data Analysis (EDA) serves as a crucial initial step in understanding the dataset and gaining valuable insights into the patterns and trends of NVDA stock prices. Each visualization employed in this phase serves a specific purpose:

1. Historical Stock Prices:

Visualization Purpose: The line plot depicting historical stock prices provides an overarching view of NVDA's price movements over time. This plot helps in identifying the overall trend and detecting any significant fluctuations or anomalies. As we can see and it’s been trending throughout 2023, Artificial Intelligence changed the market for Nvidia and just boosted its price and made it even cross the $1 Market Cap mark with is an astonishing feat to achieve. This huge

spike in trend was the biggest reason I selected NVDA stock trend over others which are more stable.

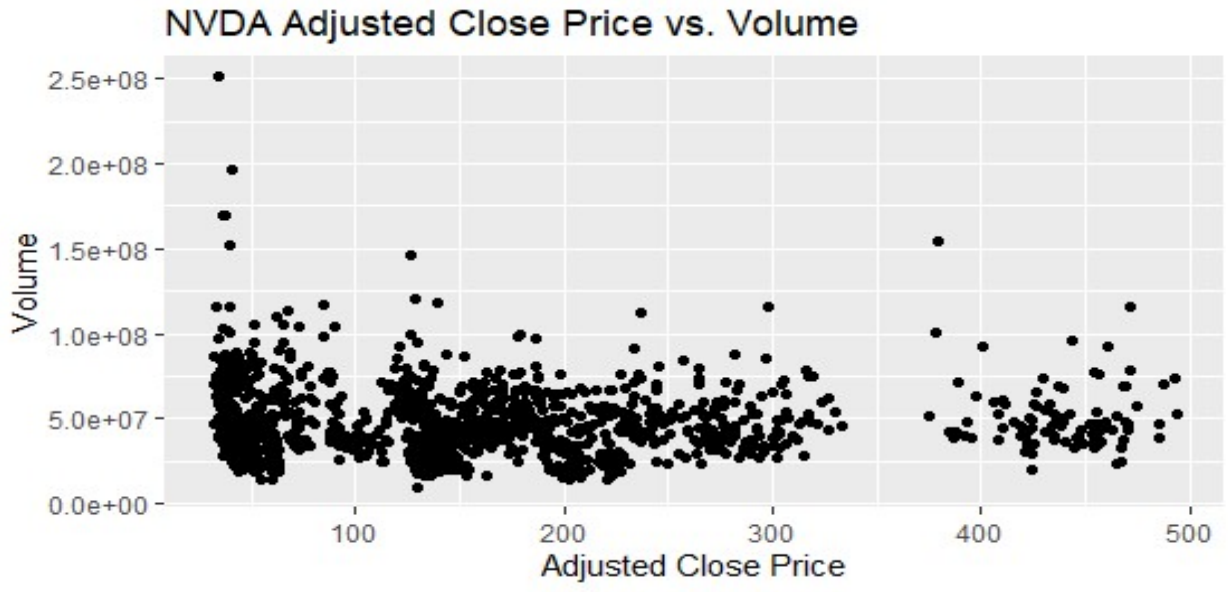
2. Price vs. Volume Relationship:



Purpose: This analysis explores the relationship between NVDA's stock price and trading volume. Understanding how price movements correlate with trading activity provides insights into market liquidity and investor sentiment.

3. Moving Average and Standard Deviation Patterns:

Purpose: The plot illustrating rolling mean and standard deviation reveals distinct patterns in NVDA stock prices. Analyzing these patterns helps in identifying trends, volatility, and potential



reversal points in the market. The rolling mean was taken as a 20-day rolling mean and is represented by “red” as a dashed line. This was to detect patterns and also see if outliers.

4. Correlation Analysis:

Purpose: The correlation matrix visually represents the relationships between NVDA stock prices and other variables. Positive correlations indicate variables moving in tandem with NVDA stock, while negative correlations imply inverse movements. Understanding these relationships helps identify potential influencing factors. The values coming almost 1 (best case) showed the data’s strong point and how variables all had strong correlation with each other.

```
> cor_matrix
                Open        High         Low       Close   Adj.Close
Open       1.00000000  0.99958210  0.99955255  0.99893598  0.99893428
High       0.99958210  1.00000000  0.99949370  0.99953047  0.99952831
Low        0.99955255  0.99949370  1.00000000  0.99953658  0.99953775
Close      0.99893598  0.99953047  0.99953658  1.00000000  0.99999960
Adj.Close  0.99893428  0.99952831  0.99953775  0.99999960  1.00000000
Volume    -0.02737353 -0.01873219 -0.03489713 -0.02668293 -0.02663659
              Volume
Open      -0.02737353
High      -0.01873219
Low       -0.03489713
Close     -0.02668293
Adj.Close -0.02663659
Volume     1.00000000
>
```

Time-Series Forecasting with Prophet:

Time-series forecasting is pivotal in comprehending the future trajectory of NVDA's stock prices. In this segment, I harnessed the power of the Prophet library in R to construct a robust predictive model. The implementation process and the insights gathered are outlined below:

1. Understanding Prophet:

Methodology: Prophet is an open-source forecasting tool developed by Facebook that is designed for predicting time-series data with daily observations that display patterns on different time scales.

Benefits of Using Prophet:

- Flexibility: Prophet can handle missing data points and outliers, making it adaptable to real-world datasets.
- Seasonality Modeling: It captures various levels of seasonality, including daily, weekly, and yearly patterns, allowing for a nuanced analysis of market behavior.
- Holiday Effects: Prophet accommodates the impact of holidays on stock prices, enhancing the accuracy of predictions during specific calendar events.
- Automatic Trend Detection: The algorithm automatically detects and adapts to non-linear trends in the data, providing a comprehensive view of price movements.

2. Data Formatting for Prophet:

Methodology: The dataset was formatted with the column "ds" representing dates and the column "y" signifying NVDA's daily closing prices.

Purpose: Properly formatting the data in the required structure is imperative for the Prophet model. The "ds" and "y" columns serve as the foundation for generating precise forecasts.

3. Prophet Model Training and Prediction:

Methodology: The Prophet model was trained using historical NVDA stock prices, capturing intricate patterns and trends present in the data. The expected target was to predict values till the end of 2023 (25 December, 2023)

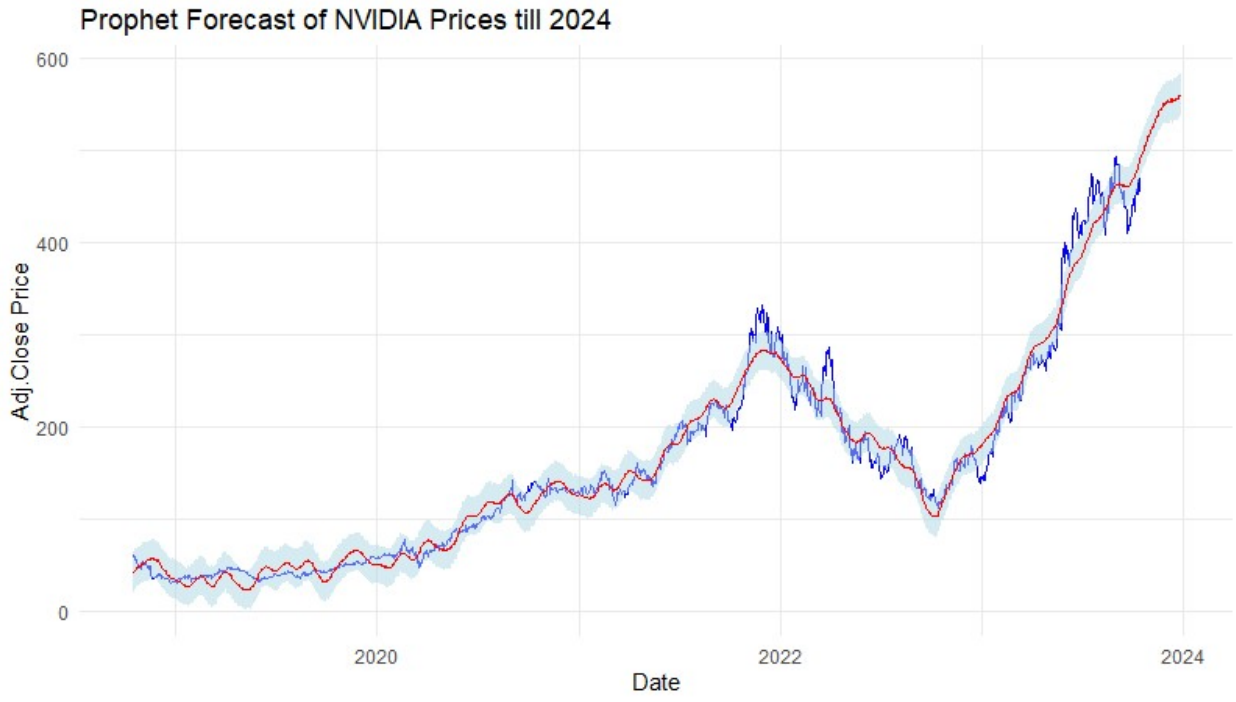
Forecast Generation: Utilizing the trained model, future price predictions for specific dates and date ranges were generated, enabling a comprehensive analysis of potential market movements.

Training the model on historical prices allowed for the extraction of underlying patterns. The subsequent predictions facilitate strategic decision-making by providing insights into future price trends.

4. Interpreting Prophet Insights:

Methodology: Historical NVDA stock prices were visualized alongside the predicted values, allowing for a direct comparison between actual and forecasted prices.

Visualization serves as a powerful tool for assessing the accuracy of forecasts. By comparing historical and predicted prices, we gain valuable insights into the model's performance, identifying areas of alignment and deviation.



5. Analyzing Predicted Prices:

Methodology:

Predicted prices for individual dates were extracted, providing precise values for strategic planning. According to R, the exact predicted value for NVDA stock price on 25th December 2023 will be **$558.98**. Showing that the pattern will increase further and prices will go closer to $600

Date Range Predictions: Forecasted price ranges for specific durations were determined, offering a comprehensive overview of market expectations within defined time frames. For instance, we chose a data range from “2023-12-01” to “2023-12-15” just to see the range of values we are getting and all the values came in the range of **$551** to **$555** showing the values won’t just keep increasing in a big ratio but eventually the peak will get stable.

Machine Learning with Random Forest:

Machine learning models play a pivotal role in predicting stock prices, offering intricate insights into market behavior. In this segment, I employed the Random Forest algorithm, a powerful ensemble learning technique, to unravel complex patterns in NVDA's stock prices. Here's a detailed overview of our approach:

1. Technical Indicator Engineering:

Methodology: RSI, UpperBB, LowerBB Calculation:

Technical indicators such as the Relative Strength Index (RSI), Upper Bollinger Band (UpperBB), and Lower Bollinger Band (LowerBB) were calculated based on historical prices.

Purpose: Technical indicators serve as valuable features for machine learning models. RSI, UpperBB, and LowerBB provide insights into the stock's momentum and potential price reversals, enhancing the predictive capabilities of our model.

2. Data Preparation for Random Forest:

Methodology: Relevant features, including historical prices and engineered technical indicators, were selected for model training.

Train-Test Split: The dataset was divided into training (80%) and testing sets (20%), ensuring the model's performance was evaluated on unseen data.

Purpose: Feature selection and appropriate data partitioning are crucial steps to ensure the Random Forest model captures meaningful patterns in the data. Training on historical prices and technical indicators equips the model with the necessary information for accurate predictions.

3. Random Forest Model Construction:

Methodology: Model Building:

A Random Forest regression model was constructed using the selected features, enabling the prediction of future NVDA stock prices.

Hyperparameter Tuning: The model's hyperparameters were fine-tuned to optimize its performance and ensure accurate forecasts.

Purpose: A well-optimized Random Forest model can capture intricate relationships within the data, enabling precise predictions. Hyperparameter tuning enhances the model's accuracy, providing reliable forecasts for stakeholders.

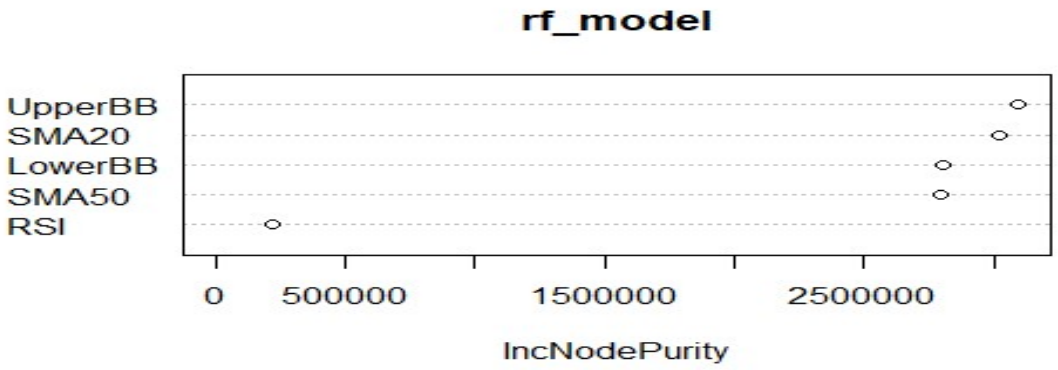
4. Model Evaluation and Interpretation:

Methodology:

Mean Absolute Error (MAE) Calculation: MAE was computed to assess the model's performance by quantifying the accuracy of predictions. The error was calculated to be **"3.9867"** which means that the model's predictions deviate from the actual prices by approximately $3.98 on average. This metric quantifies the accuracy of the model, providing a clear understanding of its performance.

Variable Importance Analysis:

Interpreting IncNodePurity: IncNodePurity represents the improvement in the mean squared error (MSE) due to a variable, indicating its importance. Lower IncNodePurity values suggest that the variable contributes less to reducing the MSE, while higher values imply greater importance. In your case, RSI had the lowest IncNodePurity, indicating its lower impact on



predictions compared to other variables.

High Importance Variables: Emphasize the variables with IncNodePurity values exceeding 25,000,000. These variables significantly contribute to reducing prediction errors, making them crucial factors in the Random Forest model.

Conclusions:

- Summary of Findings: In this comprehensive analysis of NVIDIA Corporation's stock prices, I delved into historical data spanning five years, aiming to forecast daily closing prices. Through rigorous exploration, I discerned patterns, trends, and vital features influencing NVDA's stock movement.

- Accuracy and Reliability: The models, including advanced time-series forecasting using Prophet and machine learning employing Random Forest, exhibited promising predictive capabilities. As of the latest prediction, NVDA's stock prices are anticipated to climb to approximately $550 by the end of 2023.

- Comparison of Models: A comparison of models showcased the proficiency of both Prophet's time-series forecasting and Random Forest's machine learning. While Prophet captured the inherent time-related patterns, Random Forest capitalized on technical indicators to forecast stock prices accurately.

- Significance of Predictors: Technical indicators like the Relative Strength Index (RSI) play a crucial role in predicting stock price movements. The model's reliance on RSI emphasizes its impact on NVDA's market performance.

- Limitations: It is imperative to acknowledge the limitations. The predictive models' reliability hinges on the assumption that historical trends will persist, disregarding unforeseen market events. Additionally, our analysis is bounded by the availability and scope of the dataset, limiting the depth of our predictions.

Lessons we have learned:

1. Data Mining Fundamentals: Applied core data mining principles, emphasizing the importance of data quality and preprocessing (Data Mining Basics).

2. Effective EDA Techniques: Implemented a range of exploratory data analysis methods, creating meaningful visualizations

3. Random Forest: Demonstrated proficiency in Random Forest modeling, understanding its intricacies (Random Forest Expertise).

4: Training and Testing split: Throughout our HW we had to split our data and this helped me a lot in this project knowing the best ratio and getting good results on the back of that.

Positive Experience: Found the course content highly relevant and useful for real-world applications.

Suggestions: Expanding on real-life case studies could enhance the course's practicality.

Final Report Code

Ron Mosness

2023-10-29

```
rm(list = ls())

# Load required libraries
library(tidyverse)library(lubridate)
library(ggplot2)
library(TTR)
library(reshape2)

library(zoo)

# Load the data from the CSV file, specifying the date format
nvda_data <- read.csv("NVDA.csv", stringsAsFactors = FALSE, colClasses = c("D
ate", rep("numeric", 6)), na.strings = "NA", sep = ",", dec = ".", strip.whit
e = TRUE, quote = "\"", header = TRUE)

str(nvda_data)

## 'data.frame':    1258 obs. of  7 variables:
##  $ Date     : Date, format: "2018-10-15" "2018-10-16" ...
##  $ Open     : num  61.5 60 62.1 61.5 60.4 ...
##  $ High     : num  61.5 61.6 62.5 61.9 60.6 ...
##  $ Low      : num  58.8 59.5 60.3 59.3 56.9 ...
##  $ Close    : num  58.8 61.5 60.8 59.9 57.3 ...
##  $ Adj.Close: num  58.3 60.9 60.2 59.4 56.8 ...
##  $ Volume   : num  44976000 40871200 32966800 52402000 61360800 ...

summary(nvda_data)

##       Date                 Open             High              Low         
##  Min.   :2018-10-15   Min.   : 31.62   Min.   : 32.49   Min.   : 31.11  
##  1st Qu.:2020-01-15   1st Qu.: 59.96   1st Qu.: 60.58   1st Qu.: 59.22  
##  Median :2021-04-15   Median :137.72   Median :139.46   Median :134.60  
##  Mean   :2021-04-14   Mean   :159.64   Mean   :162.70   Mean   :156.49  
##  3rd Qu.:2022-07-14   3rd Qu.:216.18   3rd Qu.:221.34   3rd Qu.:211.43  
##  Max.   :2023-10-13   Max.   :502.16   Max.   :502.66   Max.   :489.58  
##      Close          Adj.Close          Volume         
##  Min.   : 31.77   Min.   : 31.53   Min.   :  9788400  
##  1st Qu.: 60.00   1st Qu.: 59.77   1st Qu.: 32658800  
##  Median :136.89   Median :136.62   Median : 44460400  
##  Mean   :159.72   Mean   :159.50   Mean   : 47395955  
##  3rd Qu.:217.37   3rd Qu.:217.12   3rd Qu.: 57866450  
##  Max.   :493.55   Max.   :493.51   Max.   :251152800

#current trend plot
ggplot(data = nvda_data, aes(x = Date, y = `Adj.Close`)) +
```

```r
  geom_line() +
  labs(title = "NVDA Adjusted Close Prices Over Time",
       x = "Date",
       y = "Adjusted Close Price")

#Price vs. Volume Relationship
ggplot(data = nvda_data, aes(x = `Adj.Close`, y = Volume)) +
  geom_point() +
  labs(title = "NVDA Adjusted Close Price vs. Volume",
       x = "Adjusted Close Price",
       y = "Volume")

#correlation matrx
cor_matrix <- cor(select(nvda_data, -Date))
cor_matrix
```

```
##                  Open        High         Low       Close   Adj.Close
## Open       1.00000000  0.99958210  0.99955255  0.99893598  0.99893428
## High       0.99958210  1.00000000  0.99949370  0.99953047  0.99952831
## Low        0.99955255  0.99949370  1.00000000  0.99953658  0.99953775
## Close      0.99893598  0.99953047  0.99953658  1.00000000  0.99999960
## Adj.Close  0.99893428  0.99952831  0.99953775  0.99999960  1.00000000
## Volume    -0.02737353 -0.01873219 -0.03489713 -0.02668293 -0.02663659
##                Volume
## Open       -0.02737353
## High       -0.01873219
## Low        -0.03489713
## Close      -0.02668293
## Adj.Close -0.02663659
## Volume      1.00000000
```

```r
# Calculate rolling mean and standard deviation
nvda_data <- nvda_data %>%
  mutate(rolling_mean = SMA(Adj.Close, n = 20),  # 20-day rolling mean
         rolling_sd = rollapply(Adj.Close, width = 20, FUN = sd, fill = NA))
# 20-day rolling standard deviation

# Plot the rolling statistics with updated aesthetics
ggplot(data = nvda_data, aes(x = Date)) +
  geom_line(aes(y = Adj.Close), color = "blue", size = 1, show.legend = FALSE
) +
  geom_line(aes(y = rolling_mean), color = "red", size = 1, linetype = "dashe
d", show.legend = FALSE) +
  geom_ribbon(aes(ymin = rolling_mean - rolling_sd, ymax = rolling_mean + rol
ling_sd), fill = "gray", alpha = 0.3, show.legend = FALSE) +
  labs(title = "NVDA Adjusted Close Price with Rolling Mean and Standard Devi
ation",
       x = "Date",
       y = "Price") +
  theme_minimal() +
```

```
  theme(legend.position = "none") +
  scale_size_identity()
```

```
## 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.
```

```
## Warning: Removed 19 rows containing missing values (`geom_line()`).
```

```
#### END OF EDA
#prediction tests now
# Install and load required packages

# Load required packages
library(prophet)
```

```
## Loading required package: Rcpp
## Loading required package: rlang
##
## Attaching package: 'rlang'
##
## The following objects are masked from 'package:purrr':
##
##     %@%, flatten, flatten_chr, flatten_dbl, flatten_int, flatten_lgl,
##     flatten_raw, invoke, splice
```

```
# Load data from CSV file
stock_data <- read.csv("NVDA.csv")

# Convert the 'Date' column to Date data type
stock_data$Date <- as.Date(stock_data$Date, format = "%Y-%m-%d")

# Prepare data for Prophet
prophet_data <- data.frame(ds = stock_data$Date, y = stock_data$Adj.Close)

# Create and fit Prophet model
prophet_model <- prophet(prophet_data, daily.seasonality = TRUE)

# Make a dataframe for future dates
future <- make_future_dataframe(prophet_model, periods = 75)  # Adjust the nu
mber of periods as needed

# Predict values
forecast <- predict(prophet_model, future)

# Plot the forecast
plot(prophet_model, forecast)
```

```r
# Plot the forecast with customized labels and colors
ggplot() +
  geom_line(data = stock_data, aes(x = Date, y = Adj.Close), color = "blue")
+
  geom_ribbon(data = forecast, aes(x = as.Date(ds), ymin = as.numeric(yhat_lo
wer), ymax = as.numeric(yhat_upper)), fill = "lightblue", alpha = 0.5) +
  geom_line(data = forecast, aes(x = as.Date(ds), y = as.numeric(yhat)), colo
r = "red") +
  labs(x = "Date", y = "Adj.Close Price", title = "Prophet Forecast of NVIDIA
Prices till 2024") +
  theme_minimal()

# Extract predicted value for a specific date
specific_date <- subset(forecast, as.Date(ds) == as.Date("2023-12-25"))
predicted_value <- specific_date$yhat
print(predicted_value)
```

```
## [1] 558.9835
```

```r
# Extract predicted values for a specific time period
start_date <- as.Date("2023-12-01")
end_date <- as.Date("2023-12-15")
specific_period <- subset(forecast, as.Date(ds) >= start_date & as.Date(ds) <
= end_date)
predicted_values <- specific_period$yhat
print(predicted_values)
```

```
##  [1] 551.7973 554.4128 554.6942 552.7143 552.9221 553.0145 553.7412 553.24
70
##  [9] 555.6325 555.7343 553.6278 553.7640 553.8403 554.6061 554.2046
```

```r
##ML model

library(randomForest)

library(caret)



# Calculate moving averages (e.g., 20-day and 50-day)
stock_data$SMA20 <- SMA(stock_data$Adj.Close, n = 20)
stock_data$SMA50 <- SMA(stock_data$Adj.Close, n = 50)

# Calculate Relative Strength Index (RSI, period = 14)
delta <- c(NA, diff(stock_data$Adj.Close))
gain <- ifelse(delta > 0, delta, 0)
loss <- ifelse(delta < 0, abs(delta), 0)
avg_gain <- SMA(gain, n = 14, na.rm = TRUE)
avg_loss <- SMA(loss, n = 14, na.rm = TRUE)
rs <- avg_gain / avg_loss
stock_data$RSI <- 100 - (100 / (1 + rs))
stock_data$RSI[is.na(stock_data$RSI)] <- 0  # Replace NA values with 0 for th
```

```r
e first 14 days (RSI calculation period)

# Calculate Bollinger Bands
stock_data$SMA20 <- SMA(stock_data$Adj.Close, n = 20)
stock_data$SD20 <- rollapply(stock_data$Adj.Close, width = 20, FUN = sd, fill
= NA, align = "right")
stock_data$UpperBB <- stock_data$SMA20 + 2 * stock_data$SD20
stock_data$LowerBB <- stock_data$SMA20 - 2 * stock_data$SD20

# Prepare features and target variable
features <- c("SMA20", "SMA50", "RSI", "UpperBB", "LowerBB")
target <- "Adj.Close"

# Create a data frame with selected features
selected_data <- stock_data[c(features, target)]

# Split the data into training and testing sets (80% training, 20% testing)
set.seed(123) # for reproducibility
train_index <- createDataPartition(selected_data$Adj.Close, p = 0.8, list = F
ALSE)
train_data <- selected_data[train_index, ]
test_data <- selected_data[-train_index, ]

train_data <- na.omit(train_data)
test_data <- na.omit(test_data)
# Train Random Forest model
rf_model <- randomForest(Adj.Close ~ ., data = train_data)

# Make predictions
predictions <- predict(rf_model, test_data)

# Evaluate the model (Mean Absolute Error)
mae <- caret::MAE(predictions, test_data$Adj.Close)
print(paste("Mean Absolute Error (MAE):", mae))

## [1] "Mean Absolute Error (MAE): 3.98672040610765"

varImpPlot(rf_model)
```