CSP 571: DATA PREPARATION AND ANALYSIS

Stock Market Analysis Deploying ML Pipelines: Predictive Modeling for NVIDIA

~ A report by Neha Nayak (A20583245), Anusha Venkatesh (A20594323), Shanika Kadidal Venkatesh (A20585446).

ABSTRACT

The objective of this project is to forecast future stock prices of NVIDIA (NVDA) using historical stock data and machine learning models. We performed extensive exploratory data analysis (EDA), feature engineering, model development using classical machine learning techniques including Linear Regression, Ridge Regression, Random Forest, Decision Trees, and Support Vector Machines (SVM), alongside deep learning approaches such as LSTM and traditional time-series forecasting models like ARIMA. Evaluation was conducted using the RMSE metric, revealing that Linear Regression outperformed other models. The study emphasizes the strength of classical regression approaches in financial forecasting compared to deep learning for this dataset.

OVERVIEW

1.1 INTRODUCTION

The prediction of stock market trends remains an intricate and vital area of research, offering critical insights for investors and market analysts. This project seeks to harness machine learning techniques to predict future stock movements for NVIDIA Corporation. Leveraging historical data from 1999 to 2024, we aim to address critical research questions regarding model performance, significant market indicators, and the effectiveness of volatility modeling. Our principal research questions investigate whether machine learning models can accurately forecast NVIDIA's stock prices, which factors significantly impact price fluctuations, how different modeling approaches compare, and whether historical data can capture market volatility patterns.

1.2 LITERATURE REVIEW

Mukherjee and Naka (1995) explored dynamic relations between macroeconomic variables and the Japanese stock market using a Vector Error Correction Model (VECM). Their findings confirmed long-term equilibrium associations among stock prices and macroeconomic indicators, such as industrial production and interest rates, emphasizing the importance of including broader economic factors when modeling stock prices [1].

Chen (2009) investigated whether macroeconomic variables could predict recessions, especially bear markets. Using both parametric and nonparametric approaches, he demonstrated that variables like inflation rates and yield curve spreads are strong predictors of stock market downturns. This study informs the potential future extension of incorporating economic indicators into NVIDIA stock predictions [2].

Humpe and Macmillan (2009) analyzed the impact of macroeconomic factors on stock markets in the US and Japan through a cointegration framework. They found positive relationships between stock prices and industrial production but negative associations with consumer prices and long-term interest rates, particularly in the US. Their findings support the hypothesis that broader economic health significantly affects stock performance [3].

Rozeff and Kinney Jr. (1976) provided early evidence of seasonal patterns in stock returns on the New York Stock Exchange. They observed consistent 'January effects,' suggesting that seasonality should be considered when analyzing historical stock price behavior. Our seasonality analysis by month partially reflects similar trends [4].

Chlebus et al. (2019) proposed the use of machine learning for stock market prediction, emphasizing that while ML models can capture complex non-linear relationships, proper feature selection and noise handling are critical for achieving robust performance. This insight guided our approach to careful feature engineering and model evaluation [5].

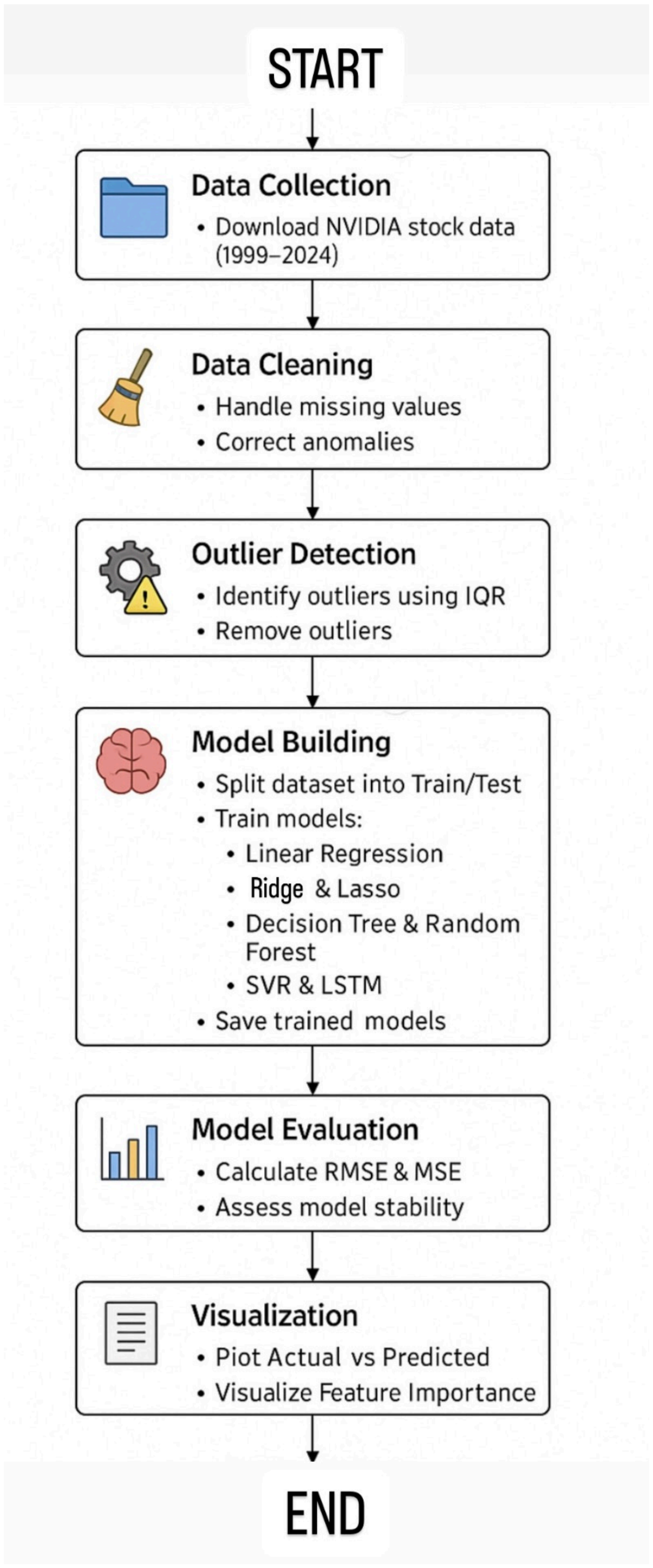
1.3 METHODOLOGY

The project followed a structured workflow starting with data collection of NVIDIA’s historical stock prices from 1999 to 2024. After importing the data, preprocessing steps such as handling missing values, formatting dates, and removing outliers using the IQR method were performed.

Feature engineering was conducted to compute technical indicators like the 20-day Moving Average (MA20), Relative Strength Index (RSI), and Bollinger Bands to enrich the input features. An extensive exploratory data analysis (EDA) phase was carried out, involving statistical summaries, boxplots, histograms, correlation heatmaps, and ANOVA tests to uncover data patterns and trends.

The modeling phase included training and evaluating multiple regression and machine learning models—Linear Regression, Ridge, Decision Tree, Random Forest, and SVM—along with advanced forecasting models like LSTM and ARIMA. RMSE was used as the primary evaluation metric. A 30-day stock price forecast was generated using the top-performing models and visualized through line plots.

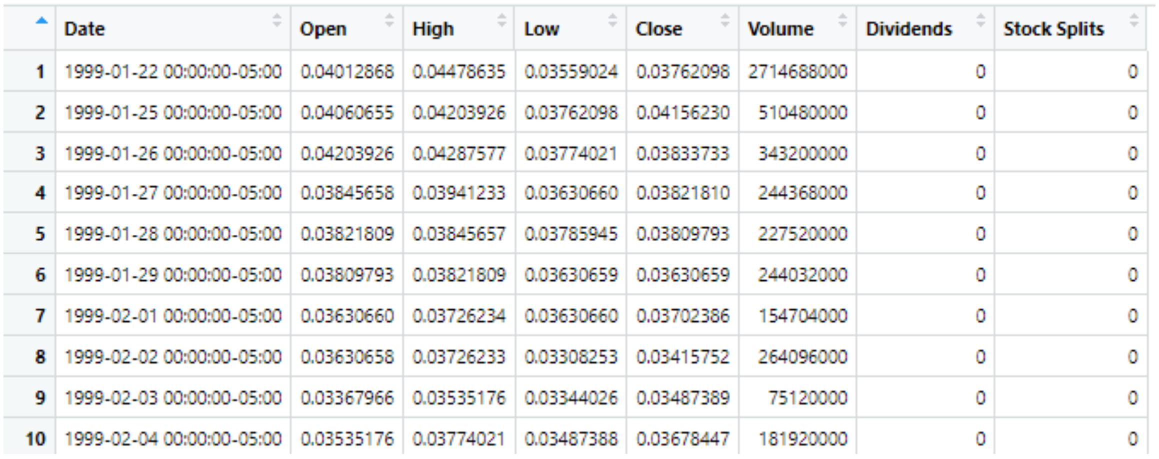
Finally, all findings, code, visualizations, and evaluations were compiled into a reproducible RMarkdown report and presentation to summarize the methodology, results, and conclusions.


START
Data Collection
• Download NVIDIA stock data (1999–2024)
Data Cleaning
• Handle missing values
• Correct anomalies
Outlier Detection
• Identify outliers using IQR
• Remove outliers
Model Building
• Split dataset into Train/Test
• Train models:
• Linear Regression
• Ridge & Lasso
• Decision Tree & Random Forest
• SVR & LSTM
• Save trained models
Model Evaluation
• Calculate RMSE & MSE
• Assess model stability
Visualization
• Piot Actual vs Predicted
• Visualize Feature Importance
END

2. DATA PROCESSING

2.1 Data Collection

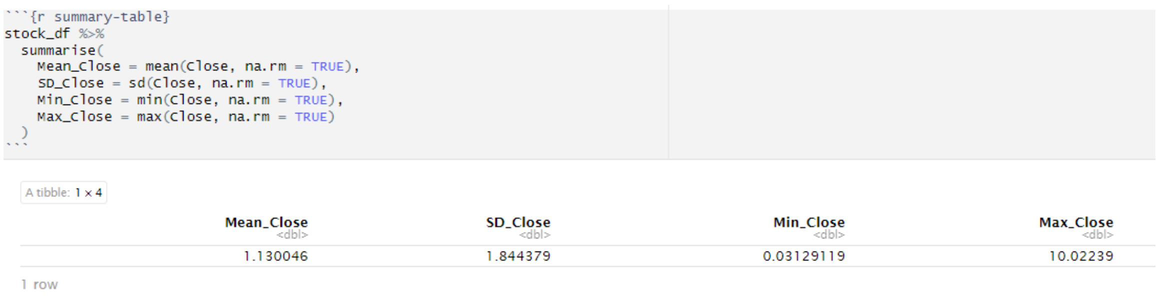
The dataset used was sourced from Kaggle and comprises 6442 historical stock price entries for NVIDIA. Each record includes Open, High, Low, Close prices, trading Volume, Dividends, and Stock Splits.



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

2.2 Data Cleaning

No missing values were present in the dataset, indicating robust data collection practices. Outliers were detected and handled using the Interquartile Range (IQR) method. Specifically, points lying beyond 1.5 times the IQR from the first and third quartiles were removed to prevent skewed modeling outcomes.



~~~
```{r summary-table}
stock_df %>%
  summarise(
    Mean_Close = mean(Close, na.rm = TRUE),
    SD_Close = sd(Close, na.rm = TRUE),
    Min_Close = min(Close, na.rm = TRUE),
    Max_Close = max(Close, na.rm = TRUE)
  )
```
~~~

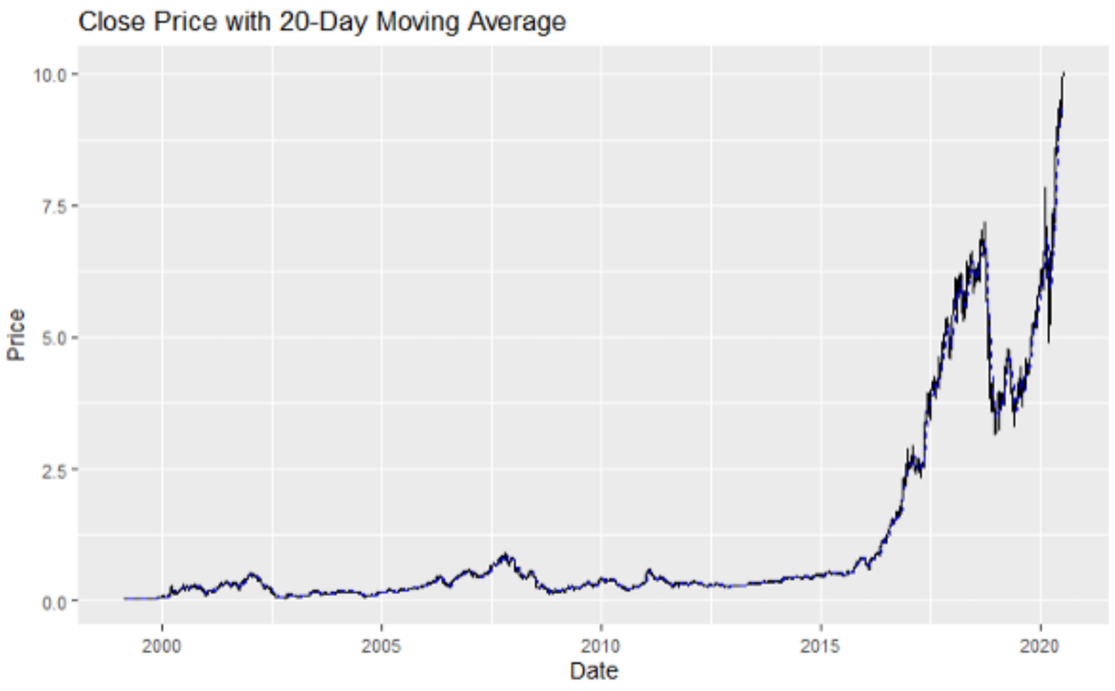
```
A tibble: 1 x 4
   Mean_Close    SD_Close    Min_Close    Max_Close
        <dbl>       <dbl>        <dbl>        <dbl>
     1.130046    1.844379   0.03129119     10.02239
1 row
```

2.3 Feature Engineering

The 20-day Moving Average (MA20) smooths out short-term price fluctuations, aiding trend identification. The Relative Strength Index (RSI) detects momentum by comparing recent gains to losses. Bollinger Bands, composed of a moving average plus/minus standard deviations, help assess price volatility and potential reversals.

Technical indicators critical for stock analysis were engineered. A 20-day moving average (MA20) was computed to smooth short-term fluctuations and highlight longer-term trends. The

Relative Strength Index (RSI) was calculated over a 14-day period to identify overbought or oversold conditions. Additionally, Bollinger Bands were created to quantify volatility.



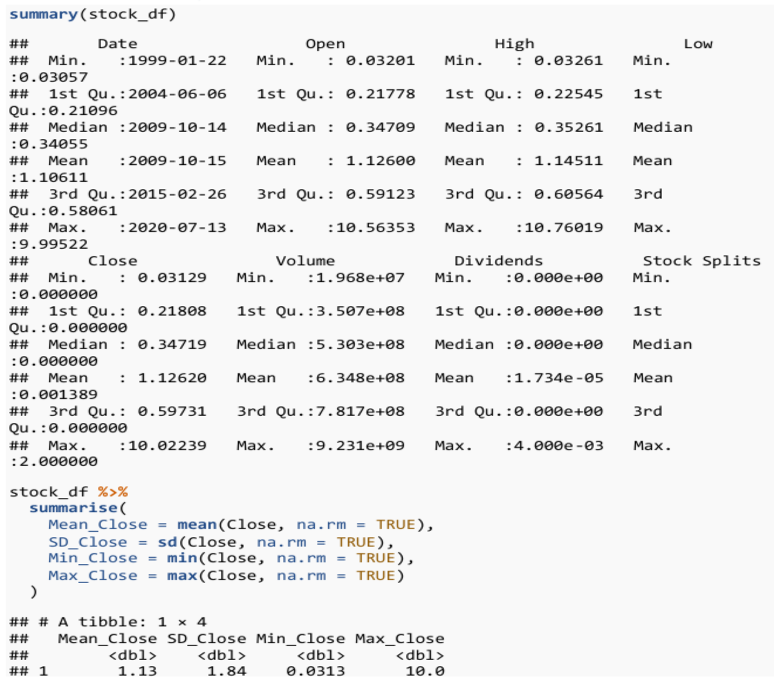
2.4 Data Transformation

Normalization and scaling techniques were applied to standardize feature ranges, ensuring that no single attribute disproportionately influenced model training.

3. EXPLORATORY DATA ANALYSIS (EDA)

3.1 Summary Statistics

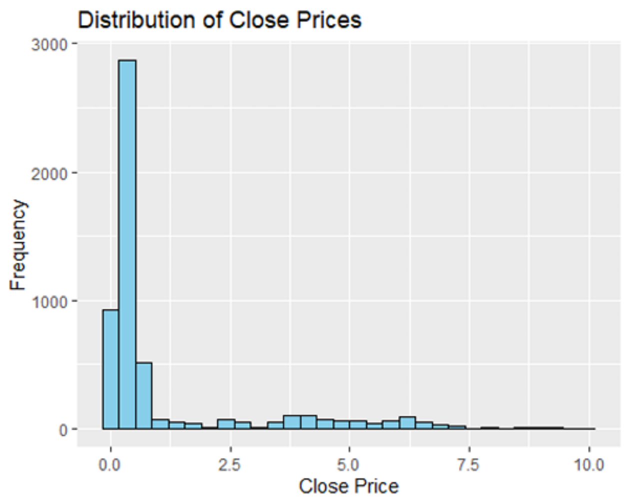
The Close prices exhibited a mean value of approximately $1.13 with a standard deviation of 1.84. The range extended from a minimum of $0.031 to a maximum of $10.02, indicating significant variability in NVIDIA's stock over the two-decade period.



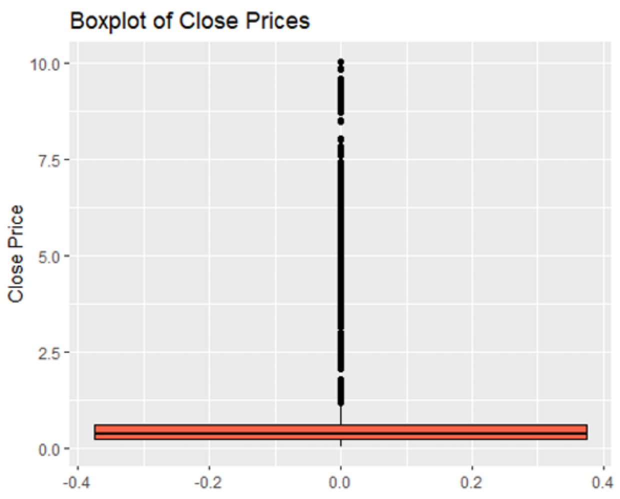
```
summary(stock_df)

##        Date                 Open               High                Low
##  Min.   :1999-01-22   Min.   : 0.03201   Min.   : 0.03261   Min.
:0.03057
##  1st Qu.:2004-06-06   1st Qu.: 0.21778   1st Qu.: 0.22545   1st
Qu.:0.21096
##  Median :2009-10-14   Median : 0.34709   Median : 0.35261   Median
:0.34055
##  Mean   :2009-10-15   Mean   : 1.12600   Mean   : 1.14511   Mean
:1.10611
##  3rd Qu.:2015-02-26   3rd Qu.: 0.59123   3rd Qu.: 0.60564   3rd
Qu.:0.58061
##  Max.   :2020-07-13   Max.   :10.56353   Max.   :10.76019   Max.
:9.99522
##       Close              Volume             Dividends          Stock Splits
##  Min.   : 0.03129   Min.   :1.968e+07   Min.   :0.000e+00   Min.
:0.000000
##  1st Qu.: 0.21808   1st Qu.:3.507e+08   1st Qu.:0.000e+00   1st
Qu.:0.000000
##  Median : 0.34719   Median :5.303e+08   Median :0.000e+00   Median
:0.000000
##  Mean   : 1.12620   Mean   :6.348e+08   Mean   :1.734e-05   Mean
:0.001389
##  3rd Qu.: 0.59731   3rd Qu.:7.817e+08   3rd Qu.:0.000e+00   3rd
Qu.:0.000000
##  Max.   :10.02239   Max.   :9.231e+09   Max.   :4.000e-03   Max.
:2.000000

stock_df %>%
  summarise(
    Mean_Close = mean(Close, na.rm = TRUE),
    SD_Close = sd(Close, na.rm = TRUE),
    Min_Close = min(Close, na.rm = TRUE),
    Max_Close = max(Close, na.rm = TRUE)
  )

## # A tibble: 1 × 4
##   Mean_Close SD_Close Min_Close Max_Close
##        <dbl>    <dbl>     <dbl>     <dbl>
## 1       1.13     1.84    0.0313      10.0
```

3.2 Distribution and Outliers

A histogram depicting the distribution of Close prices illustrated a heavy right skew, suggesting that extreme positive stock price movements occurred but were rare.

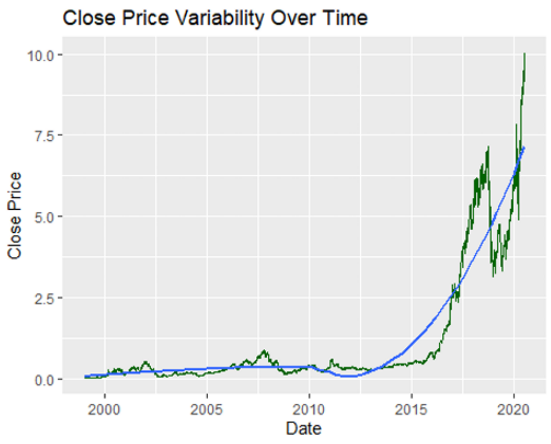


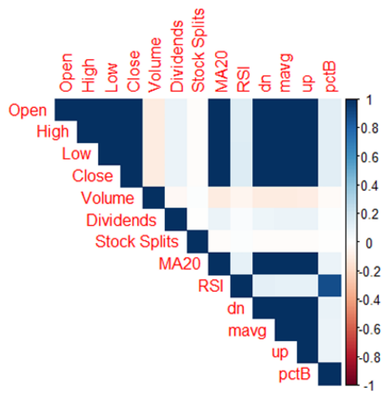
Boxplots revealed the presence of several outliers in the upper range, corresponding to periods of rapid growth in stock price. Outliers were further quantified, with approximately 1012 points identified as extreme values.



3.3 Trend and Variability Analysis

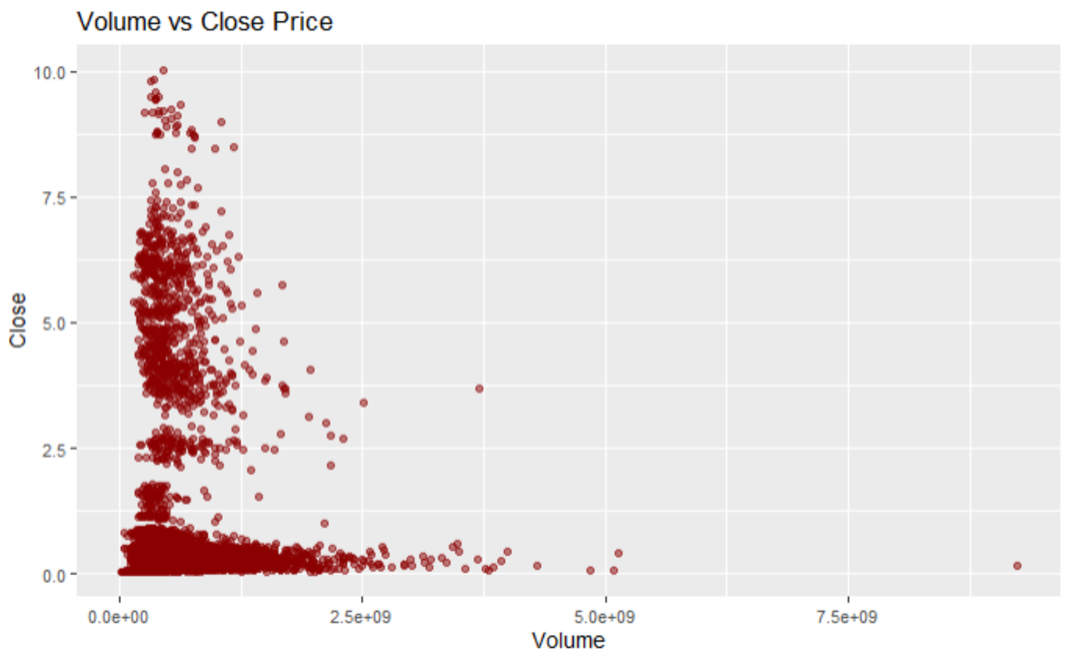
The time series plot of Close prices over the 25-year period demonstrated an overall upward trend with notable volatility during certain years, such as the 2008 financial crisis and recent technological booms. A LOESS smoothing line provided a clearer depiction of underlying trends.





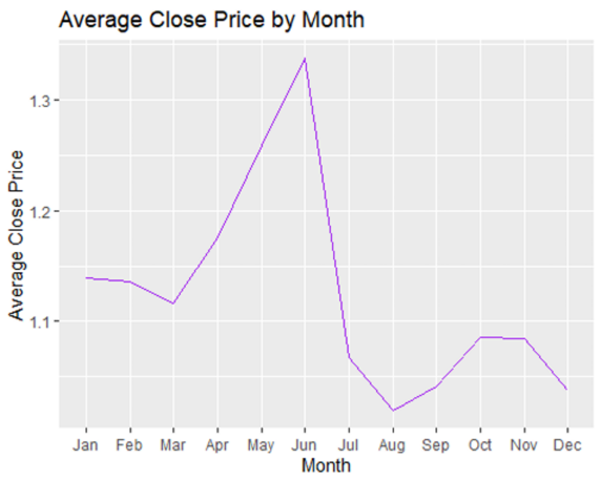
3.4 Volume Analysis

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



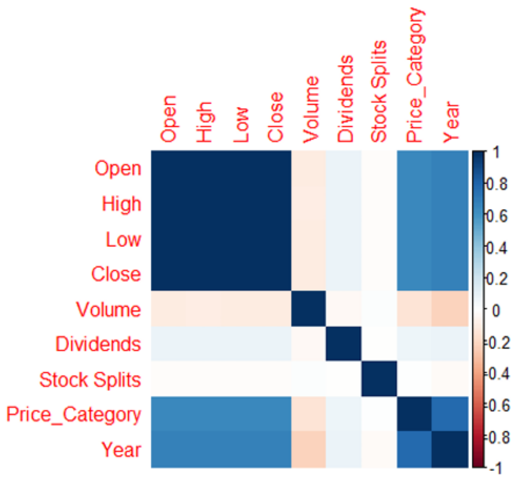
3.5 Seasonality Analysis

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



3.6 Correlation Analysis

Correlation matrices and heatmaps highlighted strong associations among price variables (Open, High, Low, Close), confirming expected market dynamics. Volume exhibited weaker but non-negligible correlations.



3.7 ANOVA Analysis

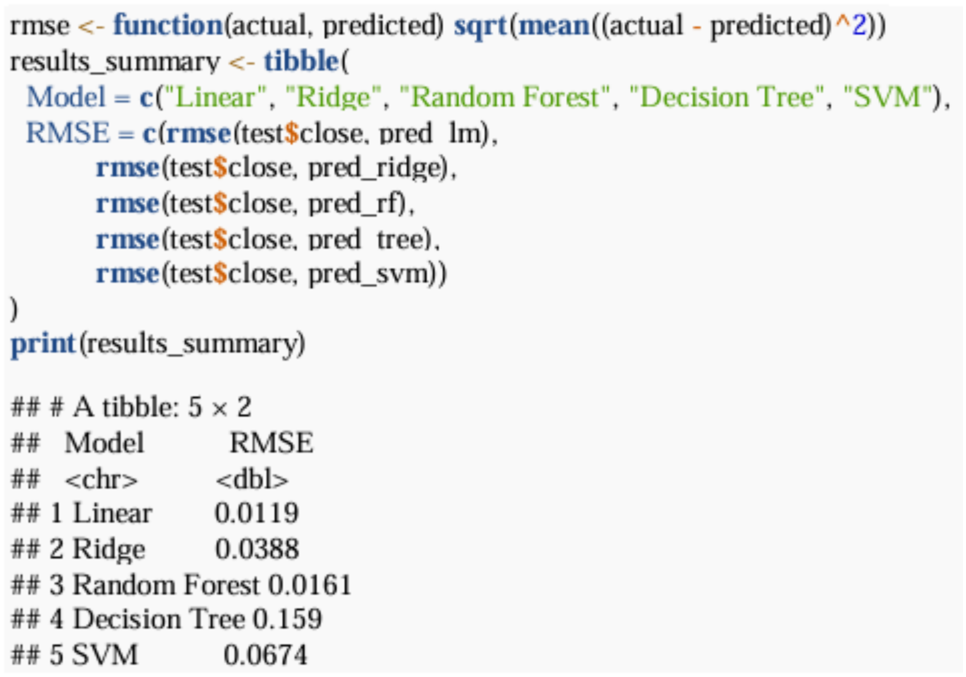
An ANOVA test comparing yearly Close prices indicated statistically significant differences ($p < 0.001$), implying that stock behavior varied meaningfully across different years.

```
stock_df$Year <- lubridate::year(stock_df$Date)

anova_model <- aov(Close ~ as.factor(Year), data = stock_df)
summary(anova_model)

##                   Df Sum Sq Mean Sq F value Pr(>F)
## as.factor(Year)   21  17482   832.5    5318 <2e-16 ***
## Residuals       5378    842     0.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4. MODEL TRAINING

4.1 Classical Machine Learning Models

A suite of classical regression models was implemented. Linear Regression served as a baseline, fitting a straight-line model to predict Close prices. Ridge Regression incorporated L2 regularization to penalize large coefficients and prevent overfitting. Random Forest models aggregated numerous decision trees to improve predictive accuracy and robustness. Decision Trees offered an interpretable yet less stable model, while SVM attempted to find the optimal hyperplane for regression.



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

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

4.2 Deep Learning Model

Long Short-Term Memory (LSTM) networks, known for capturing sequential dependencies, were trained on normalized stock sequences. Despite theoretical advantages for time-series data, LSTM underperformed, likely due to insufficient data or hyperparameter constraints.

4.3 Time Series Model

The ARIMA model was employed to fit a classical statistical framework to the Close price time series. Automatic parameter selection through auto.arima identified appropriate p, d, and q values.

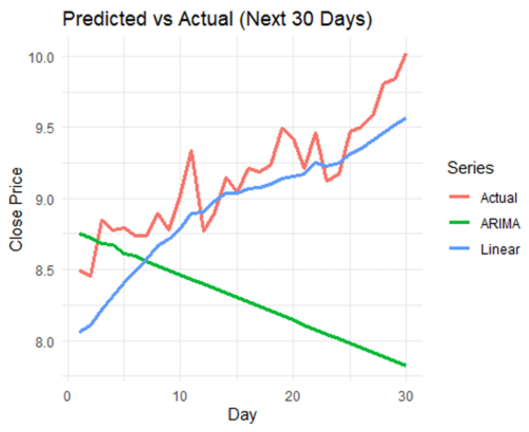
5. MODEL EVALUATION

Model performance was assessed using Root Mean Square Error (RMSE). Linear Regression achieved an RMSE of 0.0119, outperforming all other models. Random Forest followed closely with an RMSE of 0.0161, suggesting the ensemble approach's strength in non-linear settings. ARIMA produced a competitive RMSE of 0.07, suitable for univariate time-series forecasting. SVM and Ridge Regression showed moderate success, while Decision Trees and LSTM demonstrated comparatively higher error rates.

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

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

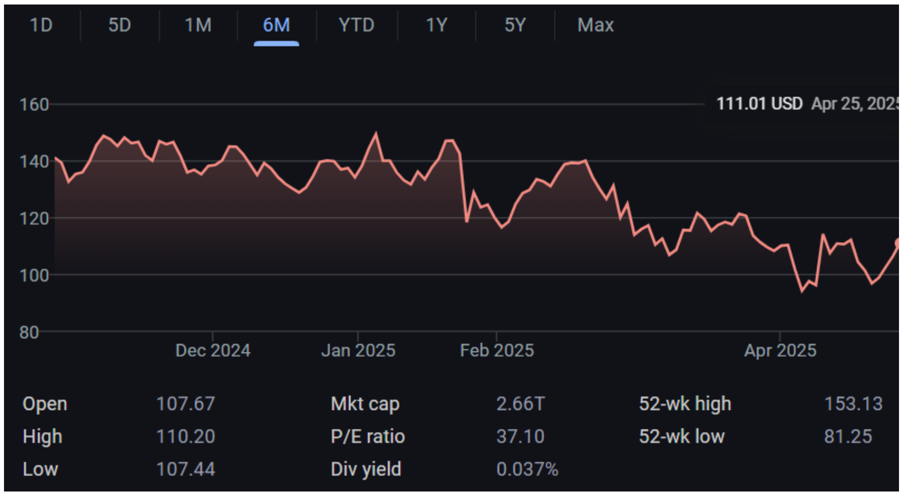
6. FORECASTING FUTURE TRENDS

Short-term forecasting over a 30-day horizon was conducted using both Linear Regression and ARIMA models. Predictions were compared against actual prices, with Linear Regression achieving a superior RMSE of 0.2785 versus ARIMA's 1.0794. The forecast plots illustrated Linear Regression's closer alignment with observed data points.



7. CONCLUSION

This project reinforces the effectiveness of simple, interpretable models like Linear Regression in financial forecasting tasks, even when compared against more sophisticated techniques such as LSTM. While deep learning models offer theoretical advantages, their performance heavily depends on extensive data and careful tuning, which was a limiting factor here. Future work could enhance predictions by integrating macroeconomic indicators, performing hyperparameter optimization for deep learning models, and exploring advanced hybrid models combining statistical and machine learning approaches.



Open	107.67	Mkt cap	2.66T	52-wk high	153.13
High	110.20	P/E ratio	37.10	52-wk low	81.25
Low	107.44	Div yield	0.037%		

10. FUTURE WORK

Future extensions of this study could include integrating macroeconomic variables such as inflation rates, interest rates, and GDP growth to enrich the predictive modeling process. Additionally, advanced deep learning architectures such as bidirectional LSTM or GRU networks could be explored. Hyperparameter optimization using techniques like Bayesian tuning could substantially improve model performance. Exploring ensemble methods that combine ARIMA with machine learning models may also offer robust improvements in time-series prediction accuracy.

11. BIBLIOGRAPHY

1. Mukherjee, Tarun K., and Atsuyuki Naka. "Dynamic Relations Between Macroeconomic Variables and the Japanese Stock Market: An Application of a Vector Error Correction Model." **The Journal of Financial Research** 18, no. 2 (1995): 223–237.

2. Chen, Shu-Heng. "Can Macroeconomic Variables Predict the Bear Market?" **Economics Bulletin** 29, no. 3 (2009): 2325–2345.

3. Humpe, Andreas, and Peter Macmillan. "Can Macroeconomic Variables Explain Long-Term Stock Market Movements?" **Applied Financial Economics** 19, no. 10 (2009): 963–976.

4. Rozeff, Michael S., and William R. Kinney Jr. "Capital Market Seasonality: The Case of Stock Returns." **Journal of Financial Economics** 3, no. 4 (1976): 379–402.

5. Chlebus, Ewa, Anna Marcinkowska, and Tomasz Zimniewicz. "Machine Learning Models for Stock Price Prediction." In **International Conference on Artificial Intelligence and Soft Computing**, pp. 221–229. Springer, 2019.

6. Kaggle. "NVIDIA Stock Price History." Accessed March 2024.
https://www.kaggle.com/datasets

7. TensorFlow. "Keras Documentation." https://keras.io/

8. Scikit-learn. "Machine Learning in Python." https://scikit-learn.org/

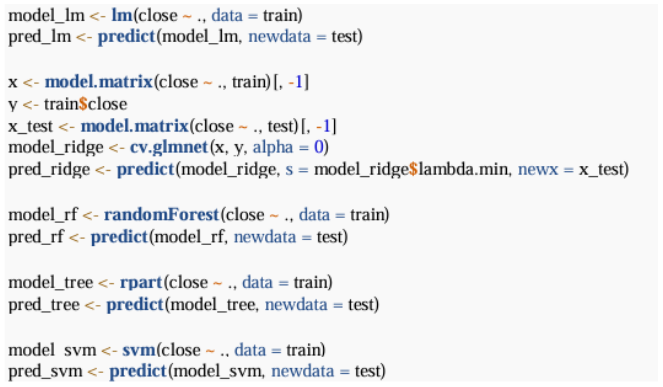
9. Stock Analysis. "NVIDIA (NVDA) Historical Stock Data." https://stockanalysis.com/stocks/nvda/

10. Yahoo Finance. "NVIDIA Corporation (NVDA) Stock Historical Prices & Data." https://finance.yahoo.com/quote/NVDA/history

APPENDIX

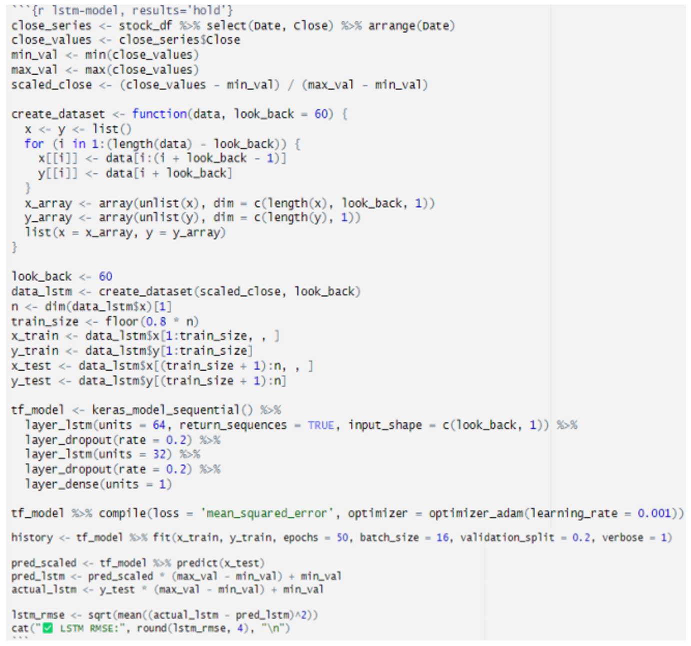
A. Code Snippets

Classic ML Models Code Block



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

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

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

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

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

LSTM Code Block

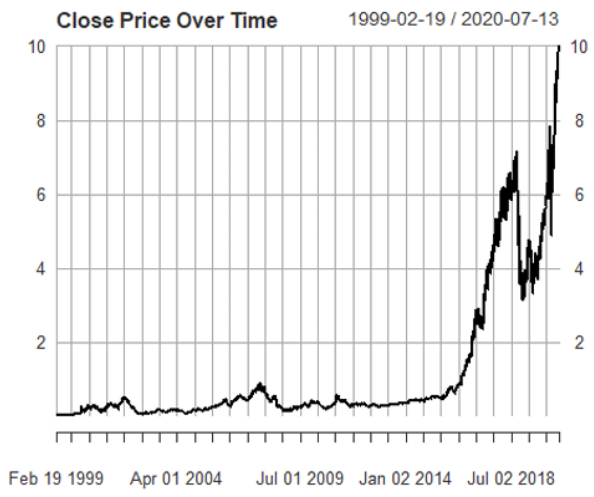


~~~
```{r lstm-model, results='hold'}
close_series <- stock_df %>% select(Date, Close) %>% arrange(Date)
close_values <- close_series$Close
min_val <- min(close_values)
max_val <- max(close_values)
scaled_close <- (close_values - min_val) / (max_val - min_val)

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

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

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

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

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

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

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

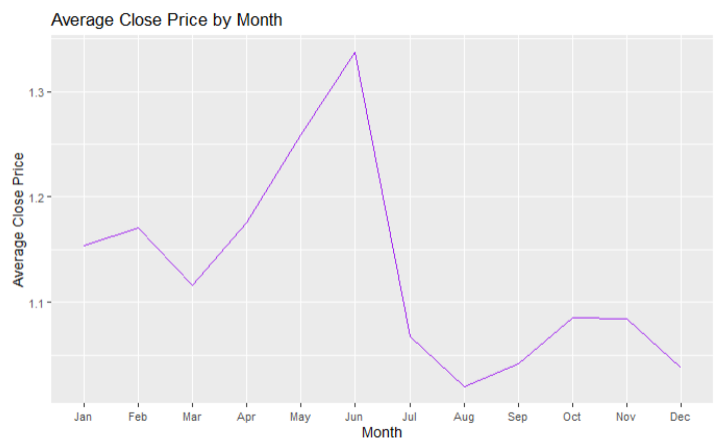
ARIMA Modeling Code Block

```r
library(forecast)

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

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

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

B. Graphical Outputs



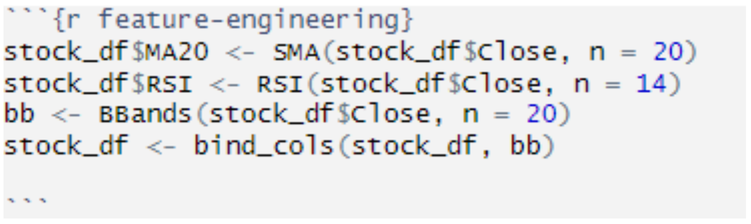
The plot shows a sharp upward trend in NVIDIA's closing stock price from 2016 onward, indicating significant long-term growth.



The plot shows that NVIDIA's average closing price peaks in June and dips significantly in July and August, suggesting potential seasonal variation in stock performance.

C. Feature Engineering Details

Moving Average (20-day), RSI (14-day), and Bollinger Bands (20-day window) calculations were conducted using technical indicator packages and custom scripts.



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

```
~~~