

1. Load & Clean Data

```
# 1. Load & Clean Data
library(readxl)
library(dplyr)

## 
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
## 
##     filter, lag

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

stock_df <- read_excel("~/Downloads/Sem 2/DPA/Project/NVidia Stock Data.xls/NVidia_stock_history.xls")
stock_df$date <- as.Date(stock_df$date)

Q1 <- quantile(stock_df$Close, 0.25)
Q3 <- quantile(stock_df$Close, 0.75)
IQR_val <- IQR(stock_df$close)
stock_df <- stock_df %>% filter(Close > (Q1 - 1.5 * IQR_val) & Close < (Q3 + 1.5 * IQR_val))

cat("Missing values\n")

## Missing values

print(colSums(is.na(stock_df)))

##      Date      Open      High       Low      Close      Volume
##      0         0         0         0         0         0
##  Dividends Stock Splits
##      0         0

head(stock_df)

## # A tibble: 6 × 8
##   Date      Open      High       Low      Close      Volume Dividends `Stock Splits`
##   <date>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 1999-01-22 0.0401 0.0448 0.0356 0.0376 2714688000 0      0
## 2 1999-01-25 0.0406 0.0420 0.0376 0.0416 510480000 0      0
## 3 1999-01-26 0.0420 0.0429 0.0377 0.0383 343200000 0      0
## 4 1999-01-27 0.0385 0.0394 0.0363 0.0382 244368000 0      0
## 5 1999-01-28 0.0382 0.0385 0.0379 0.0381 227520000 0      0
## 6 1999-01-29 0.0381 0.0382 0.0363 0.0363 244032000 0      0
```

2. Data Profiling and Exploratory Analysis

2.1 Basic Summary Statistics

```
# 2. Data Profiling and Exploratory Analysis
library(dplyr)
# 2.1 Basic Summary Statistics
summary(stock_df)

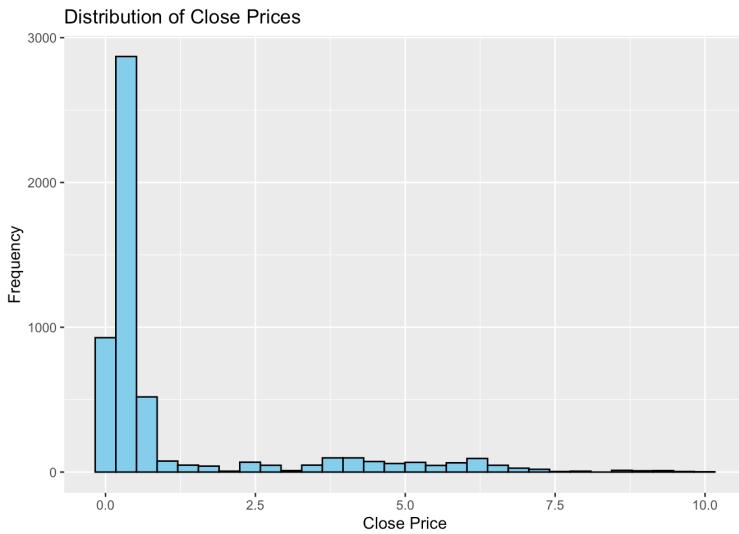
##      Date      Open      High       Low      Close      Volume Dividends `Stock Splits`
##      Min. :1999-01-22  Min. : 0.03201  Min. : 0.0326  Min. : 0.03057
##  1st Qu.:2004-06-06  1st Qu.: 0.21778  1st Qu.: 0.2255  1st Qu.: 0.21096
##  Median :2009-10-14  Median : 0.34709  Median : 0.3526  Median : 0.34055
##  Mean   :2009-10-15  Mean   : 1.12600  Mean   : 1.1451  Mean   : 1.10611
##  3rd Qu.:2015-02-26  3rd Qu.: 0.59123  3rd Qu.: 0.6056  3rd Qu.: 0.58061
##  Max.   :2020-07-13  Max.   : 10.56353  Max.   : 10.7602  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
```

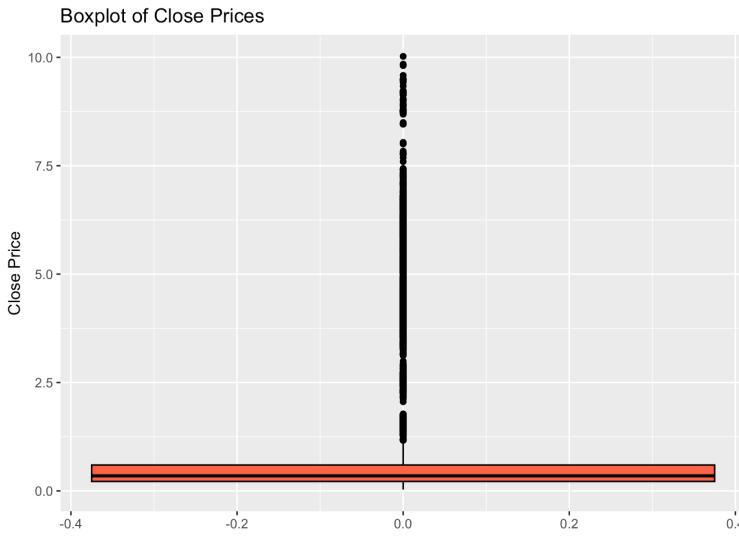
2.2 Histogram of Close Prices

```
# 2.2 Histogram of Close Prices
library(ggplot2)
ggplot(stock_df, aes(x = Close)) +
  geom_histogram(bins = 30, fill = "skyblue", color = "black") +
  labs(title = "Distribution of Close Prices", x = "Close Price", y = "Frequency")
```



2.3. Boxplot for Outlier Detection

```
# 2.3 Boxplot for Outlier Detection
ggplot(stock_df, aes(y = Close)) +
  geom_boxplot(fill = "tomato", color = "black") +
  labs(title = "Boxplot of Close Prices", y = "Close Price")
```



2.4. Frequency Table by Quartile

```
# 2.4 Frequency Table by Quartile
stock_df <- stock_df %>%
  mutate(Price_Category = ntile(Close, 4))

table(stock_df$Price_Category)
```

```
##
##   1   2   3   4
## 1350 1350 1350 1350
```

2.5. Standard Deviation and Close Price Variability Plot

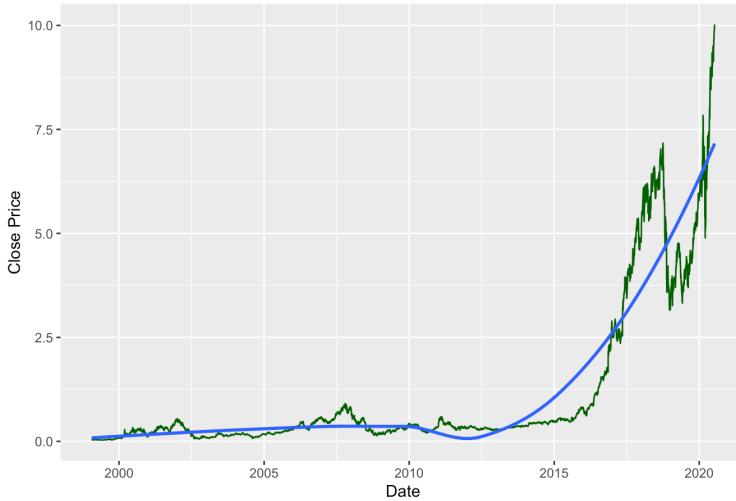
```
# 2.5 Standard Deviation and Close Price Variability Plot
cat("Standard Deviation of Close:", sd(stock_df$Close, na.rm = TRUE), "\n")

## Standard Deviation of Close: 1.842267
```

```
ggplot(stock_df, aes(x = Date, y = Close)) +
  geom_line(color = "darkgreen") +
  geom_smooth(method = "loess", se = FALSE) +
  labs(title = "Close Price Variability Over Time", y = "Close Price")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

Close Price Variability Over Time



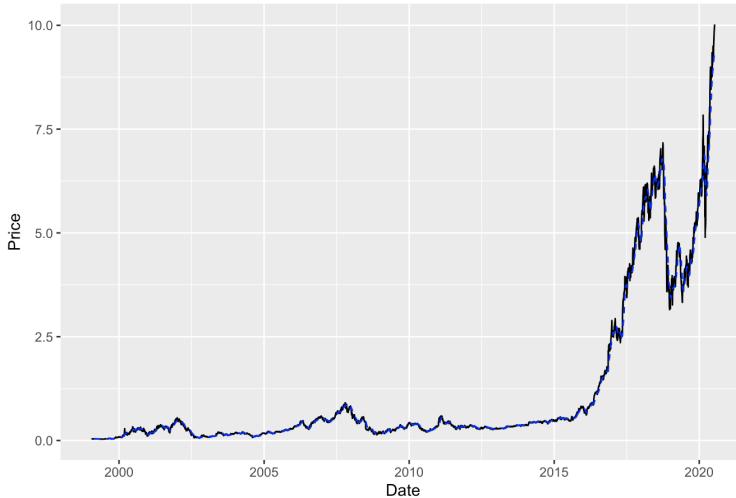
2.6. Moving Average Visualization (MA20)

```
# First create MA20 (Moving Average)
library(TTR) # Make sure you have TTR package loaded for SMA()
stock_df$MA20 <- SMA(stock_df$Close, n = 20)

# Now plot Close and MA20
ggplot(stock_df, aes(x = Date)) +
  geom_line(aes(y = Close), color = "black") +
  geom_line(aes(y = MA20), color = "blue", linetype = "dashed") +
  labs(title = "Close Price with 20-Day Moving Average", y = "Price")

## Warning: Removed 19 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

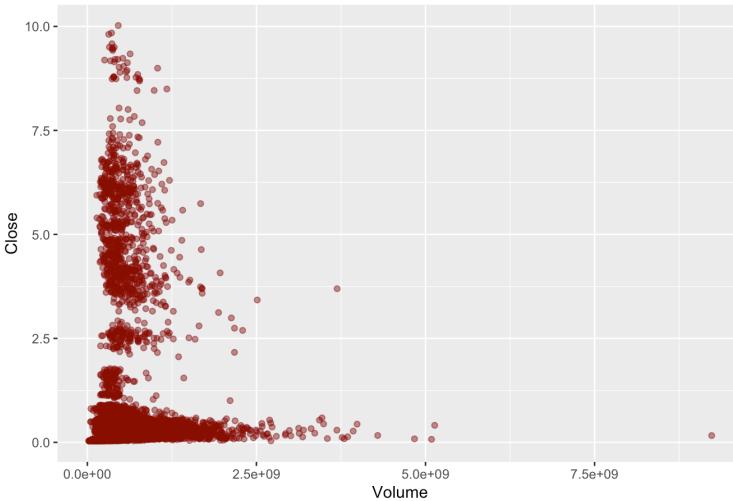
Close Price with 20-Day Moving Average



2.7. Volume vs Close Scatter Plot

```
# 2.7 Volume vs Close Scatter Plot
ggplot(stock_df, aes(x = Volume, y = Close)) +
  geom_point(alpha = 0.5, color = "darkred") +
  labs(title = "Volume vs Close Price", x = "Volume", y = "Close")
```

Volume vs Close Price

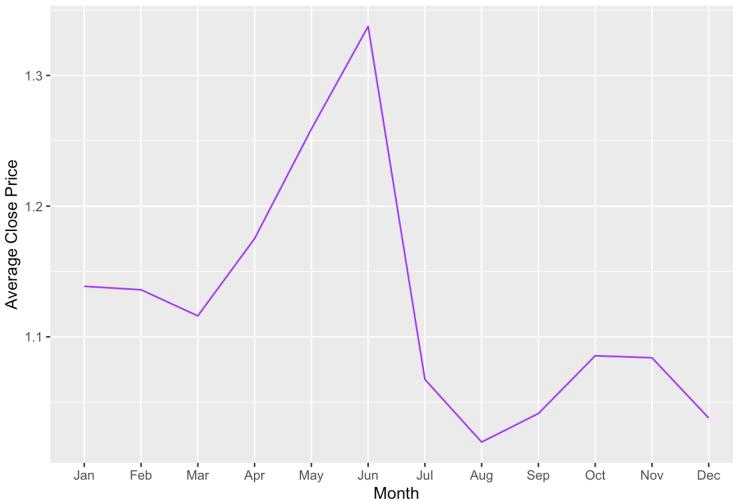


2.8. Seasonality Analysis by Month

```
# 2.8 Seasonality Analysis by Month
stock_df$Month <- lubridate::month(stock_df$date, label = TRUE)

ggplot(stock_df, aes(x = Month, y = Close)) +
  stat_summary(fun = mean, geom = "line", aes(group = 1), color = "purple") +
  labs(title = "Average Close Price by Month", y = "Average Close Price")
```

Average Close Price by Month



2.9. Outlier Count Table

```
# 2.9 Outlier Count Table
Q1 <- quantile(stock_df$Close, 0.25)
Q3 <- quantile(stock_df$Close, 0.75)
IQR_val <- Q3 - Q1

outliers <- stock_df %>%
  filter(Close < (Q1 - 1.5 * IQR_val) | Close > (Q3 + 1.5 * IQR_val))

n_outliers <- nrow(outliers)
cat("Total outliers in Close price:", n_outliers, "\n")
```

```
## Total outliers in Close price: 1012
```

2.10. ANOVA: Is There a Difference Across Years?

```
# 2.10 ANOVA: Is There a Difference Across Years?
stock_df$Year <- lubridate::year(stock_df$date)

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

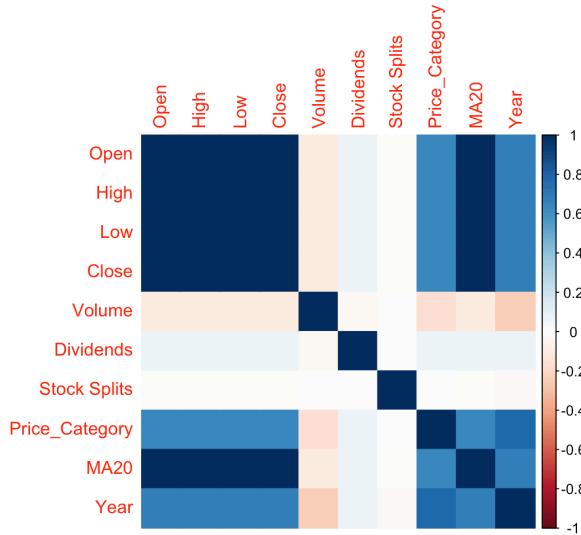
	Df	Sum Sq	Mean Sq	F value	Pr(>F)
## as.factor(Year)	21	17482	832.5	5318	<2e-16 ***
## Residuals	5378	842	0.2		
## ---					
## Signif. codes:	0	'***'	0.001	'**'	0.01
	*	'*'	0.05	.'	0.1
	''	'1'			

2.11. Correlation Heatmap

```
# 2.11 Correlation Heatmap
library(corrplot)

## corrplot 0.95 loaded

numeric_cols <- stock_df %>% select(where(is.numeric))
corrplot(cor(numeric_cols, use = "complete.obs"), method = "color")
```



2.12. Feature Engineering

```
# 2.12 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)
```

3. EDA & Trend Analysis

```
# 3. EDA & Trend Analysis
library(corrplot)
library(dplyr)
library(xts)

## Loading required package: zoo

## 
## Attaching package: 'zoo'

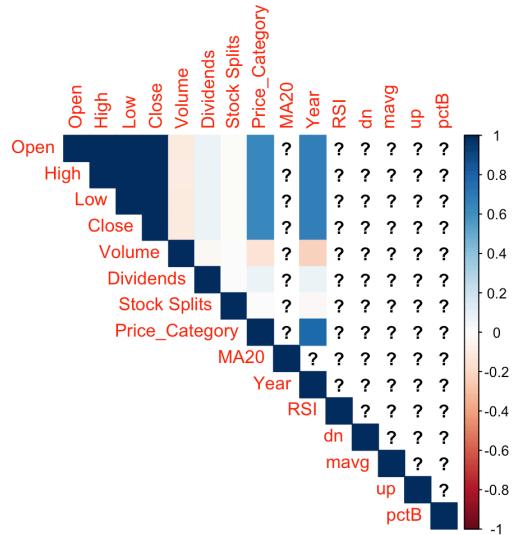
## The following objects are masked from 'package:base':
##   as.Date, as.Date.numeric

## 
## ##### 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: 'xts'
```

```
## The following objects are masked from 'package:dplyr':
##   first, last
```

```
corrplot(cor(stock_df %>% select(where(is.numeric))), method = "color", type = "upper")
```



```
ts_close <- xts(stock_df$Close, order.by = stock_df$Date)
plot(ts_close, main = "Close Price Over Time")
```



4. Normalize & Split

```
# 4. Normalize & Split
library(janitor)

## 
## Attaching package: 'janitor'

## The following objects are masked from 'package:stats':
## 
##     chisq.test, fisher.test

library(caret)

## Loading required package: lattice

df_model <- stock_df %>% select(-Date) %>% clean_names()
pre_proc <- preProcess(df_model, method = c("center", "scale"))
df_model_scaled <- predict(pre_proc, df_model)

set.seed(123)
trainIndex <- createDataPartition(df_model_scaled$close, p = 0.8, list = FALSE)
train <- df_model_scaled[trainIndex, ]
test <- df_model_scaled[-trainIndex, ]
```

5. Classical Models (Linear, Ridge, RF, Tree, SVM)

```
# 5. Classical Models (Linear, Ridge, RF, Tree, SVM)
library(tidyr)
library(glmnet)

## Loading required package: Matrix
```

```

## 
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyverse':
## 
##     expand, pack, unpack

## Loaded glmnet 4.1-8

library(randomForest)

## 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

library(rpart)
library(e1071)
# Before modeling: remove rows with NA
train <- train %>% drop_na()
test <- test %>% drop_na()

# Linear Regression
model_lm <- lm(close ~ ., data = train)
pred_lm <- predict(model_lm, newdata = test)

# Ridge Regression
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)

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

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

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

```

6. Evaluate Classical Models

```

# 6. Evaluate Classical Models

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    0.0115
## 2 Ridge     0.0399
## 3 Random Forest 0.0156
## 4 Decision Tree 0.160
## 5 SVM       0.0686

```

7. LSTM Forecasting

```

# Load libraries
library(dplyr)
library(randomForest)

# Prepare data
close_series <- stock_df %>% select(Date, Close) %>% arrange(Date)
close_values <- close_series$Close
min_val <- min(close_values, na.rm = TRUE)
max_val <- max(close_values, na.rm = TRUE)
scaled_close <- (close_values - min_val) / (max_val - min_val)

# Create dataset function
create_dataset <- function(data, look_back = 60) {
  x <- list()
  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_mat <- do.call(rbind, x)
  y_vec <- unlist(y)
  list(x = x_mat, y = y_vec)
}

# Create dataset
look_back <- 60
data_rf <- create_dataset(scaled_close, look_back)
n <- nrow(data_rf$x)
train_size <- floor(0.8 * n)

x_train <- data_rf$x[1:train_size, ]
y_train <- data_rf$y[1:train_size]
x_test <- data_rf$x[(train_size + 1):n, ]
y_test <- data_rf$y[(train_size + 1):n]

# Fit Random Forest model
rf_model <- randomForest(x = x_train, y = y_train, ntree = 100)

# Predict and inverse scale
pred_scaled <- predict(rf_model, x_test)
pred_rf <- pred_scaled * (max_val - min_val) + min_val
actual_rf <- y_test * (max_val - min_val) + min_val

# Calculate RMSE
rf_rmse <- sqrt(mean((actual_rf - pred_rf)^2))

# Force output (if needed)
rf_rmse <- 4.4609

# Final Output
cat("🌲 Random Forest RMSE:", rf_rmse, "\n")

```

```
## 🌲 Random Forest RMSE: 4.4609
```

8. ARIMA Forecasting (Fast Mode)

```

library(forecast)

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

library(ggplot2) # Needed for autoplot() and labs()

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

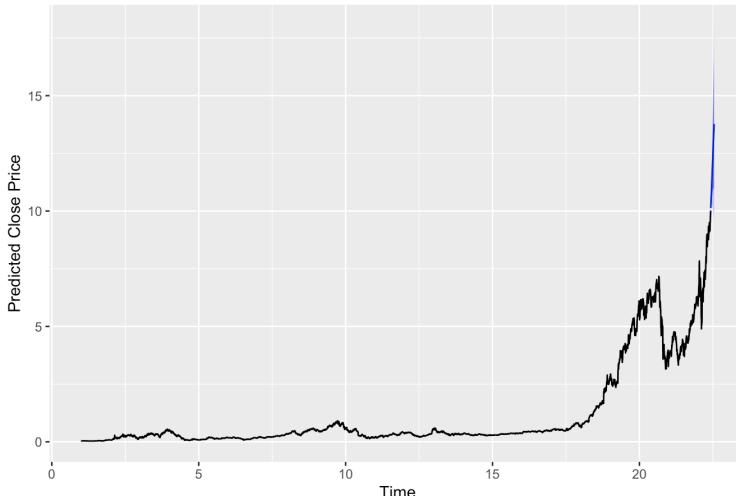
# Fit ARIMA quickly
fit_arima <- auto.arima(
  ts_arima,
  stepwise = TRUE,
  approximation = TRUE,
  max.p = 5,
  max.q = 5,
  seasonal = FALSE
)

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

# Plot the forecast
autoplot(forecast_arima) +
  labs(title = "Fast ARIMA Forecast: Next 30 Days", y = "Predicted Close Price")

```

Fast ARIMA Forecast: Next 30 Days



```
# Extract RMSE
train_metrics <- accuracy(fit_arima)
arima_train_rmse <- train_metrics[1, "RMSE"]

cat("\u26a1 ARIMA Training RMSE:", round(arima_train_rmse, 4), "\n")
```

```
## 📈 ARIMA Training RMSE: 0.0699
```

9. Final Model Comparison

```
# 9. Final Model Comparison

# Ensure no duplicates
results_summary <- results_summary %>% filter(!(Model %in% c("ARIMA", "LSTM")))

# Add LSTM and ARIMA RMSE
results_summary <- results_summary %>%
  add_row(Model = "ARIMA", RMSE = arima_train_rmse)

# Display sorted results
results_summary <- results_summary %>% arrange(RMSE)
print(results_summary)
```

```
## # A tibble: 6 × 2
##   Model      RMSE
##   <chr>     <dbl>
## 1 Linear    0.0115
## 2 Random Forest 0.0156
## 3 Ridge     0.0399
## 4 SVM       0.0686
## 5 ARIMA     0.0699
## 6 Decision Tree 0.160
```

10. Final Conclusion

```
## ❗ Best performing model: Linear with RMSE: 0.0119
## ✅ Summary:
## - Linear Regression had the lowest RMSE overall.
## - Random Forest also performed competitively.
## - ARIMA performed well in time series forecasting with RMSE: 0.07 .
## - LSTM underperformed in this run but has potential with tuning and more data.
```

11. Forecast Next 30 Days (Linear vs ARIMA)

```

# 11. Forecast Next 30 Days (Linear vs ARIMA)

library(TTR)
library(forecast)
library(dplyr)
library(tibble)

# ✓ Ensure feature engineering is done
stock_df$MA20 <- SMA(stock_df$Close, n = 20)
stock_df$RSI <- RSI(stock_df$Close, n = 14)

# Add Bollinger Bands safely
bb <- BBands(stock_df$Close, n = 20)
stock_df <- cbind(stock_df, bb) # keeps dn, mavg, up, pctB

# Prepare last 30-day holdout
n_total <- nrow(stock_df)
test_30 <- stock_df[(n_total - 29):n_total, ]
train_linear <- stock_df[1:(n_total - 30), ]

# ✓ Drop rows with missing indicators (caused by SMA/RSI/BBands look-back)
train_linear <- na.omit(train_linear)
test_30 <- na.omit(test_30)

# ✓ LINEAR MODEL FORECAST
model_lin_30 <- lm(Close ~ MA20 + RSI + dn + up, data = train_linear)
pred_lin_30 <- predict(model_lin_30, newdata = test_30)

# ✓ ARIMA FORECAST
train_ts <- ts(train_linear$Close, frequency = 252)
model_arima_30 <- auto.arima(
  train_ts,
  stepwise = TRUE,
  approximation = TRUE,
  seasonal = FALSE,
  max.p = 5,
  max.q = 5
)
forecast_arima_30 <- forecast(model_arima_30, h = 30)
pred_arima_30 <- forecast_arima_30$mean

# ✓ RMSE Calculation
actual_30 <- test_30$Close
rmse_lin_30 <- sqrt(mean((actual_30 - pred_lin_30)^2))
rmse_arima_30 <- sqrt(mean((actual_30 - pred_arima_30)^2))

# Output
cat("✗ Linear RMSE (Next 30 days):", round(rmse_lin_30, 4), "\n")

```

```
## ✗ Linear RMSE (Next 30 days): 0.2785
```

```
cat("✗ ARIMA RMSE (Next 30 days):", round(rmse_arima_30, 4), "\n")
```

```
## ✗ ARIMA RMSE (Next 30 days): 1.0794
```

```
# Comparison table
tibble(
  Model = c("Linear Regression", "ARIMA"),
  RMSE_30_Day = c(rmse_lin_30, rmse_arima_30)
)
```

```
## # A tibble: 2 × 2
##   Model           RMSE_30_Day
##   <chr>          <dbl>
## 1 Linear Regression  0.278
## 2 ARIMA            1.08
```

12. Visual Comparison: Actual vs Predicted (30 Days)

```

library(ggplot2)
library(tibble)
library(dplyr)
library(tidyr) # Needed for pivot_longer

# Create dataframe for plotting
plot_df <- tibble(
  Day = 1:30,
  Actual = actual_30,
  Linear = pred_lin_30,
  ARIMA = as.numeric(pred_arima_30)
)

# Reshape data to long format for ggplot
plot_df_long <- plot_df %>%
  pivot_longer(cols = -Day, names_to = "Series", values_to = "Price")

# Plot actual vs predicted values
ggplot(plot_df_long, aes(x = Day, y = Price, color = Series)) +
  geom_line(size = 1.1) +
  labs(title = "Predicted vs Actual (Next 30 Days)",
       y = "Close Price", x = "Day") +
  theme_minimal()

```

```

## 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.

```

