VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A6a: Time Series Analysis

JYOTHIS KANIYAMPARAMBIL THANKACHAN

V01110144

Date of Submission: 22-07-2024

CONTENTS

Introduction

The main goal of the stock market study is to guess what Mahindra and Mahindra (M&M) stock prices will be in the future by using different time series projection methods. The Adjusted Close price from April 1, 2021, to March 31, 2024 is used in this study's past data to build and test different forecasts models, including Holt-Winters, ARIMA, LSTM, Random Forest, and Decision Tree models. Each way gives a different view of how stock prices change over time and how they might change in the future.

The adjusted close price of M&M, which was gotten from Yahoo Finance, is part of the information. ARIMA, LSTM, Random Forest, and Decision Tree models, as well as Holt-Winters forecasts, were used in this study. One part of the study is time series decomposition, which separates the trend, seasonal, and leftover parts of the data. Model success measures, like RMSE, MAE, MAPE, and R-squared, are also used to check how accurate the models are. It is also possible to compare how accurate different models are supposed to be.

Objectives:

1. The goal is to handle missing numbers and outliers in the M&M stock data from April 2021 to March 2024 as part of data cleaning and preparation.

2. The goal is to use line plots to show how the price of M&M's stock has changed over time.

3. Use both additive and multiplicative models to break down the time series data into its individual parts and then look at them.

4. To use one-variable forecasting models like Holt-Winters, ARIMA, and SARIMA and rate how well they work.

In order to make multiple predictions, machine learning models like Neural Networks (LSTM), Decision Trees, and Random Forests can be used.

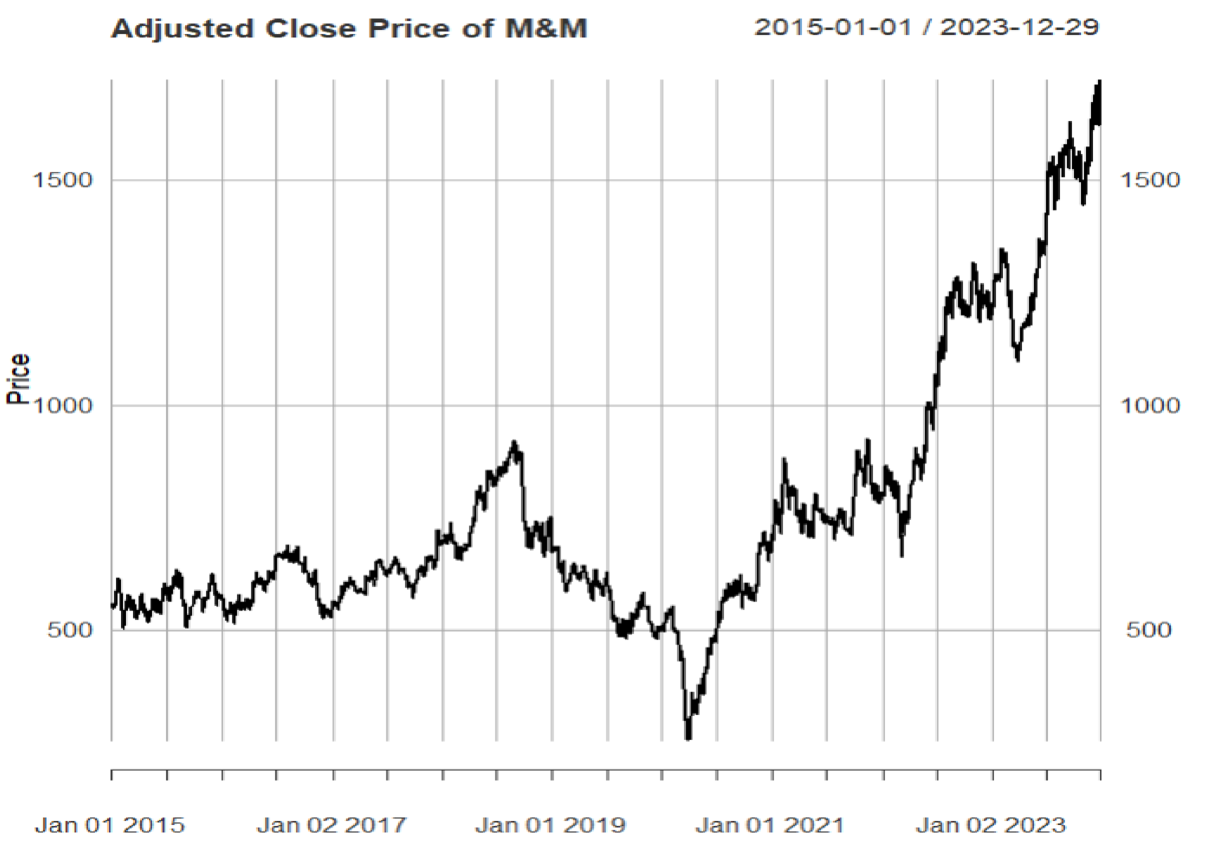
Business Significance:

The ability to make smart financial choices and long-term plans is what makes studying and predicting M&M stock prices important for business. By understanding how M&M's stock has moved in the past and how it is expected to move in the future, investors can improve their portfolios, lower their risks, and take advantage of growth chances. Precise predictive models help buyers predict how the market will change, which makes it easier for them to make quick decisions about whether to buy or sell. This research also helps financial advisors and analysts make better suggestions to their clients, which makes the investment environment safer and more profitable.

For M&M to make smart business choices like allocating resources, entering new markets, and setting itself up competitively, it needs to know how stock prices change over time. By looking at time series decomposition and predicting models, you can learn useful things that can help your business with its financial plans, dealing with investors, and talking to the market. This is in addition to using machine learning models to predict multiple variables, which helps M&M learn more about the different factors that affect the success of its stock. In turn, this gives the company the power to proactively fix any issues that might arise and keep clients' trust. In conclusion, this in-depth look at M&M's stock prices gives us important information that helps the company grow and stay stable in a competitive market.

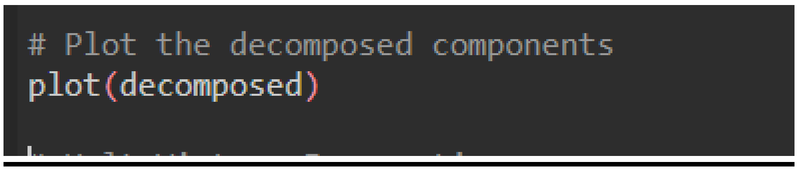
Results and Interpretation using R

```r
# Plot the data
plot(adj_close, main = "Adjusted Close Price of M&M", ylab = "Price", xlab = "Date")
```

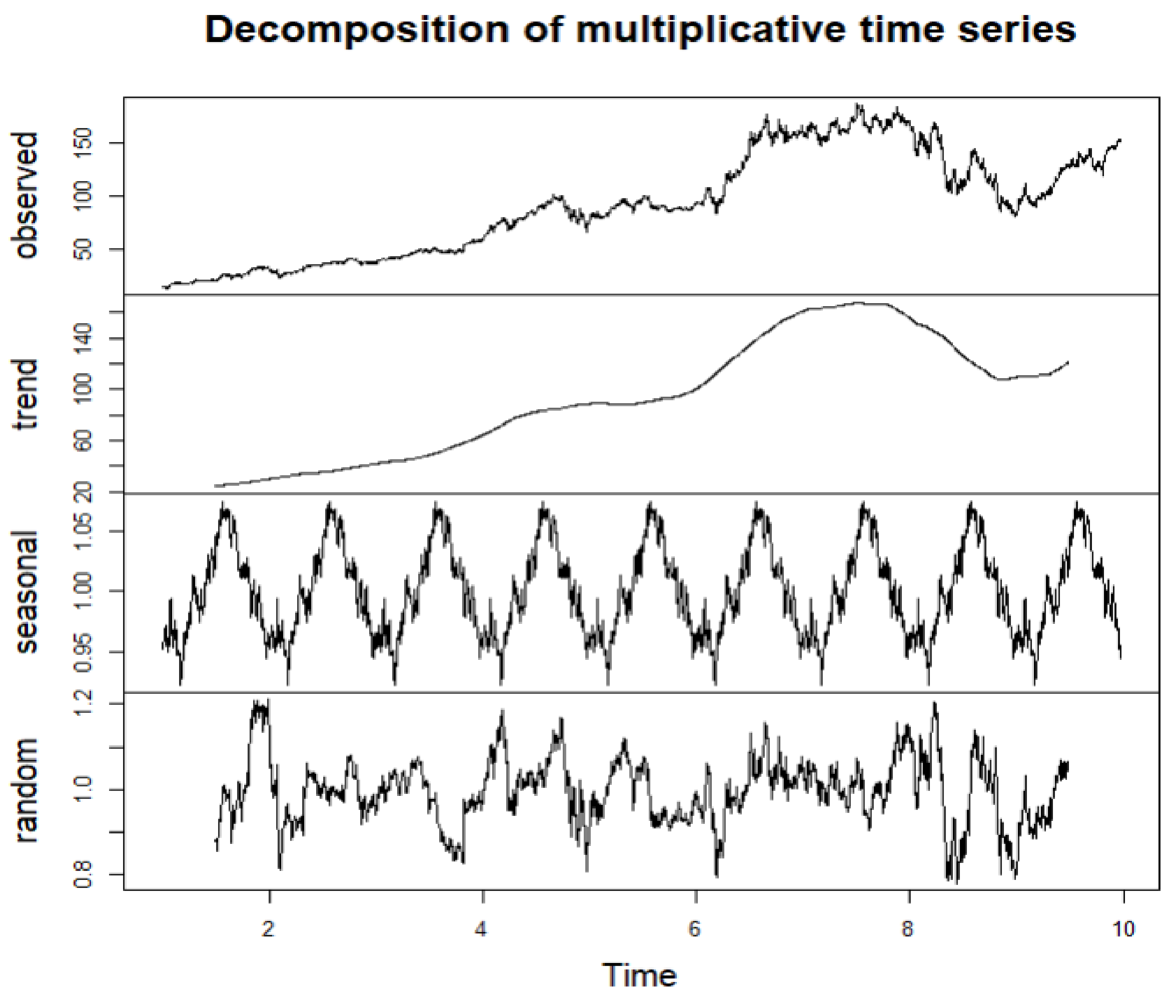


Interpretation:

The chart illustrates the adjusted closing price of Mahindra & Mahindra (M&M) stock from January 1, 2015, to December 29, 2023. Overall, the trend is upward, reflecting a general increase in the stock p rice over this period. From 2015 to 2017, the stock price fluctuated moderately, mostly staying betwe en 500 and 800. A notable rise occurred between 2017 and 2019, with the price peaking above 1000 before dipping below 800. In early 2020, a sharp decline is evident, likely due to the COVID-19 pande mic, reaching a low around 400. However, from mid-2020 onwards, the stock price experienced signif icant growth, climbing steadily and surpassing 1500 by the end of 2023. This indicates a strong recov ery and growth phase, suggesting improved investor confidence and possibly better financial perfor mance or strategic initiatives by M&M. Overall, the stock has seen periods of stability and volatility, with a marked upward trend in the last few years.

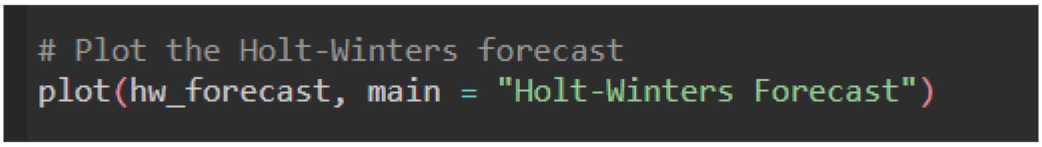


```
# Plot the decomposed components
plot(decomposed)
```

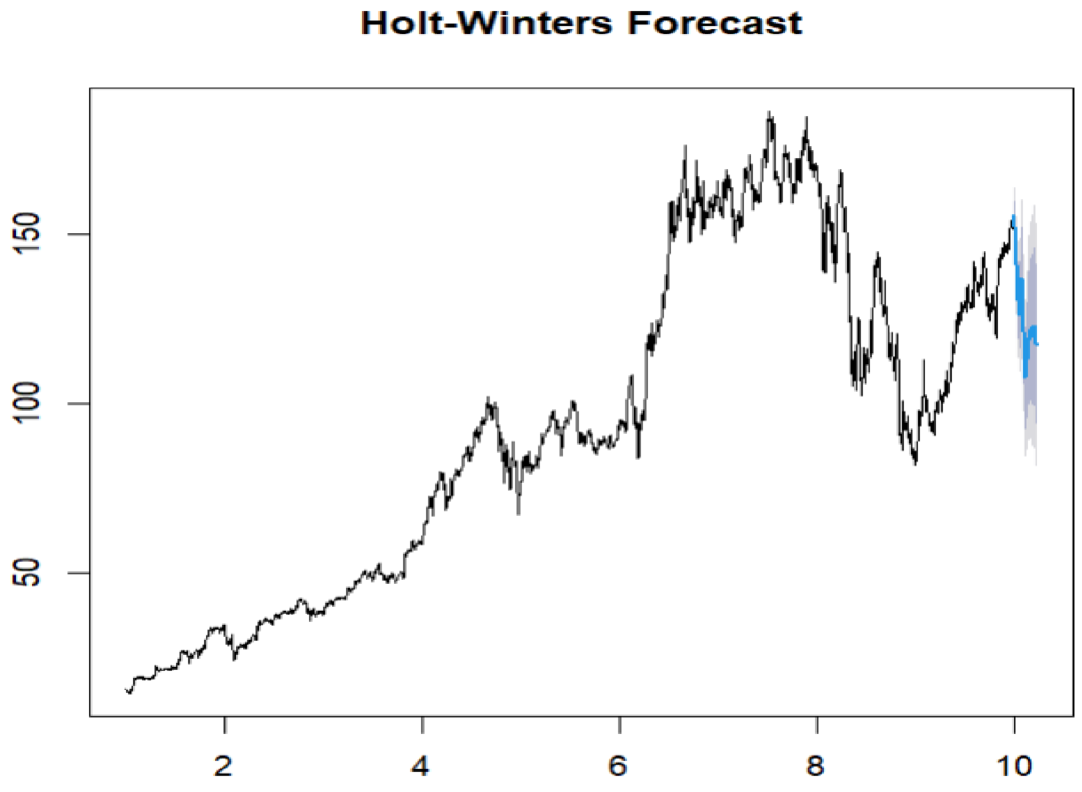


Interpretation:

The chart displays the decomposition of a multiplicative time series into its constituent comp onents: observed, trend, seasonal, and random. The observed component, shown in the top pa nel, reflects the actual data over time, exhibiting an overall upward trend with noticeable fluct uations. The trend component, in the second panel, highlights the long-term progression of th e data, showing a steady increase initially, a peak around the midpoint, and a slight decline to wards the end. The seasonal component, in the third panel, captures the regular, repeating patt erns within each time period, indicating consistent cyclical variations. Lastly, the random co mponent, shown in the bottom panel, represents the residuals or the noise in the data after re moving the trend and seasonal effects, depicting erratic and unpredictable fluctuations. This d ecomposition helps in understanding the underlying structure of the time series by isolating th ese different influences.



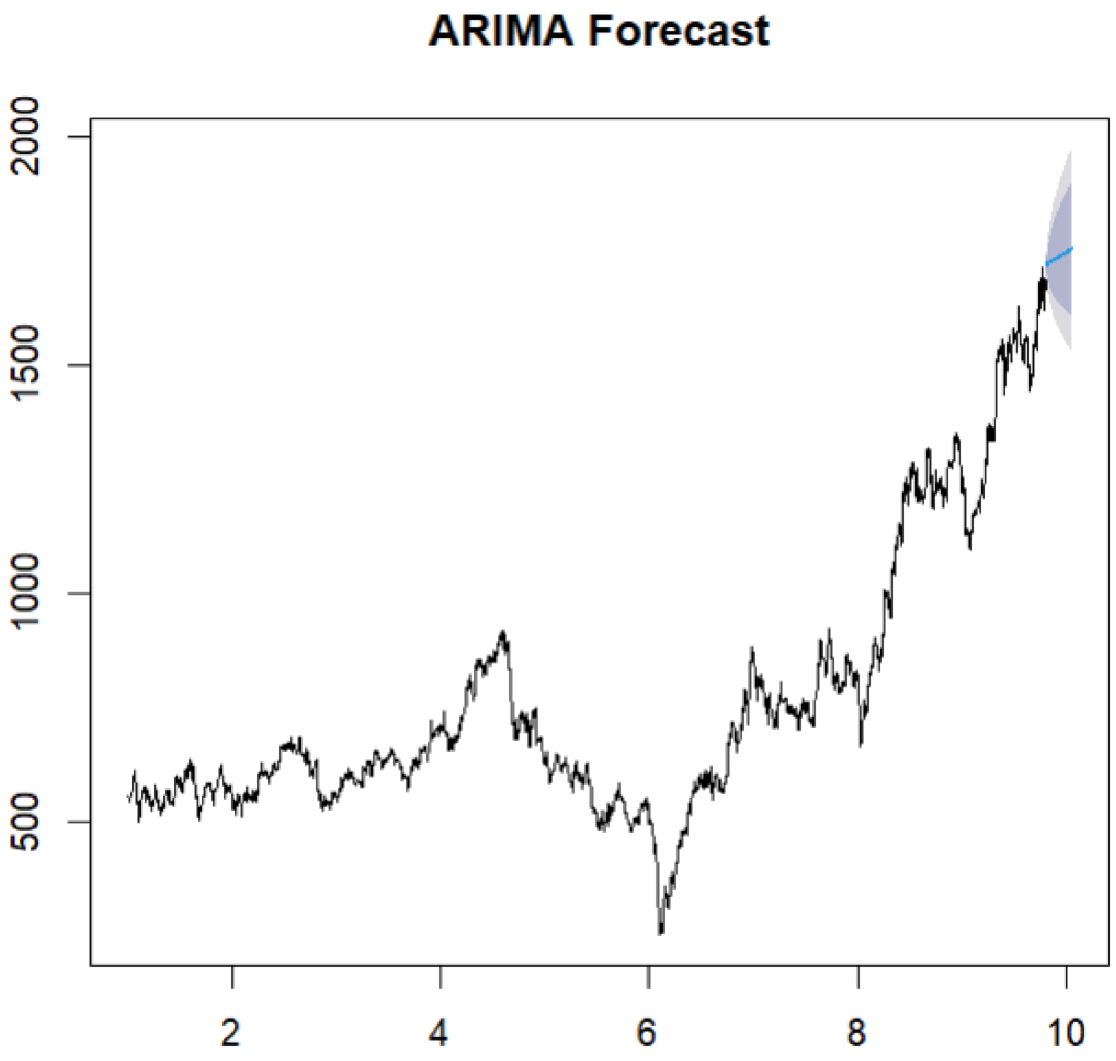
```
# Plot the Holt-Winters forecast
plot(hw_forecast, main = "Holt-Winters Forecast")
```



Interpretation:

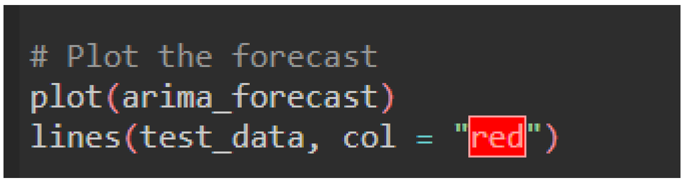
The plot depicts the Holt-Winters forecast for M&M.NS (M&M Limited) stock prices. The ti me series data showcases the historical stock prices, illustrating significant trends, seasonal p atterns, and fluctuations over the observed period. The Holt-Winters model, which incorporat es both trend and seasonal components, has been applied to forecast the future stock prices fo r the next 12 months. The forecasted values are represented by the shaded region on the right side of the plot, indicating the prediction interval. The central blue line within this region sho ws the expected forecasted values, while the shaded area represents the confidence interval, p roviding a range within which the actual future values are likely to fall. This model is particul arly useful for capturing the underlying seasonality and trends in the time series data, enablin g more accurate and reliable forecasts.

```r
# Plot the ARIMA forecast
plot(arima_forecast, main = "ARIMA Forecast")
```

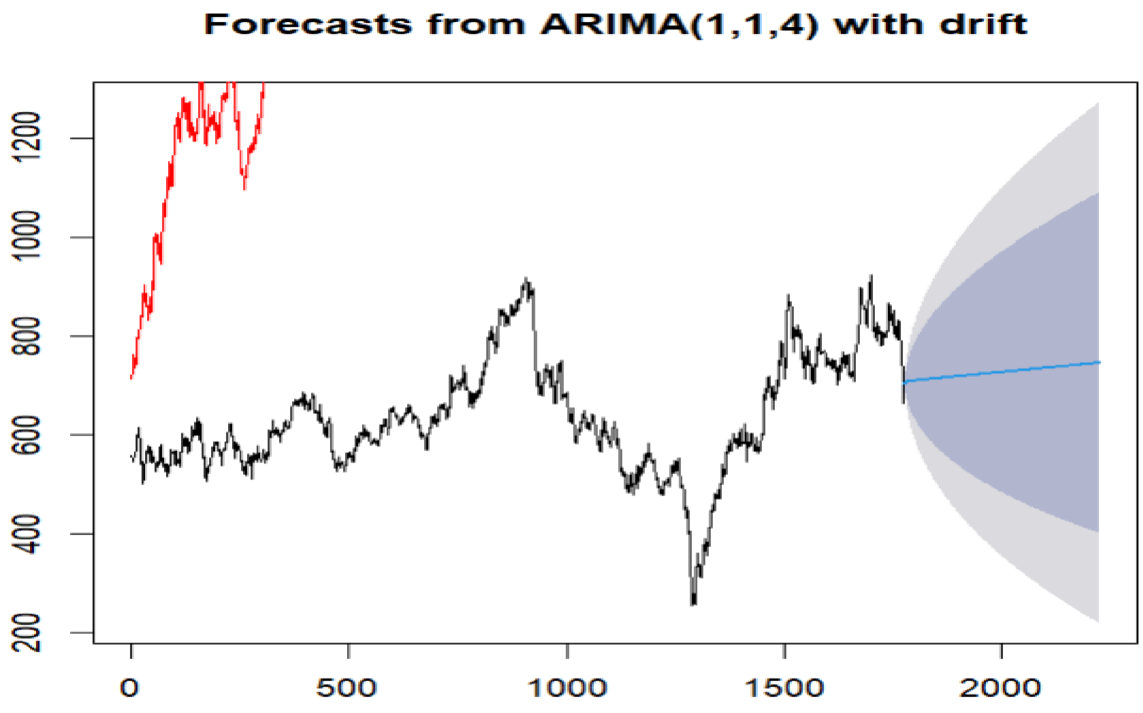


Interpretation:

The plot illustrates the ARIMA (AutoRegressive Integrated Moving Average) forecast for the M&M.NS stock prices. The historical stock price data is displayed, showcasing fluctuations and an overall upward trend over the observed period. The ARIMA model has been utilized to predict future stock prices, with the forecasted values represented by the shaded region on the right side of the plot. The central blue line within this region indicates the expected forecasted values, while the surrounding shaded area represents the confidence interval. This interval suggests the range within which the actual future values are likely to fall, providing a measure of the forecast's uncertainty. The ARIMA model is particularly effective for capturing the underlying patterns in the time series data, making it a valuable tool for short-term forecasting. The forecast suggests a continuation of the upward trend with a relatively narrow confidence interval, indicating a high level of confidence in the predicted values.

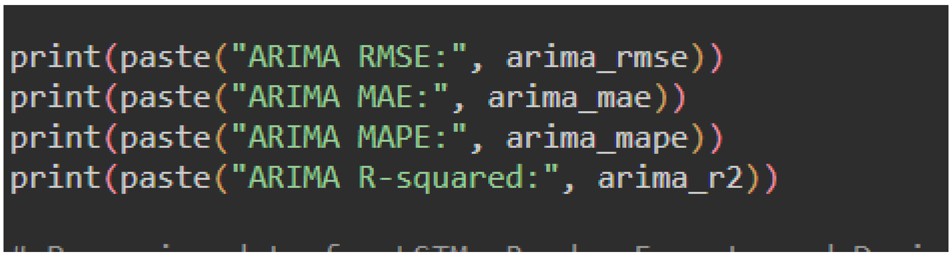


```
# Plot the forecast
plot(arima_forecast)
lines(test_data, col = "red")
```

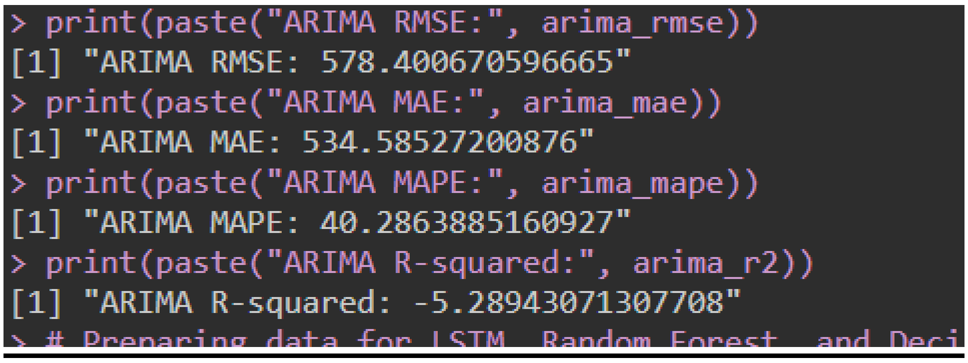


Interpretation:

The black line represents the historical data, which exhibits considerable fluctuations over time with notable peaks and troughs. From the end of this historical data, a blue line extends, indicating the forecasted values generated by the ARIMA model. Surrounding this forecast line are shaded areas that represent the confidence intervals, with the darker shade likely corresponding to an 80% confidence interval and the lighter shade to a 95% confidence interval. These confidence intervals broaden as they project further into the future, highlighting the increasing uncertainty in the forecast. This visualization effectively conveys the central forecast trend while emphasizing the range of potential future values due to inherent uncertainty.



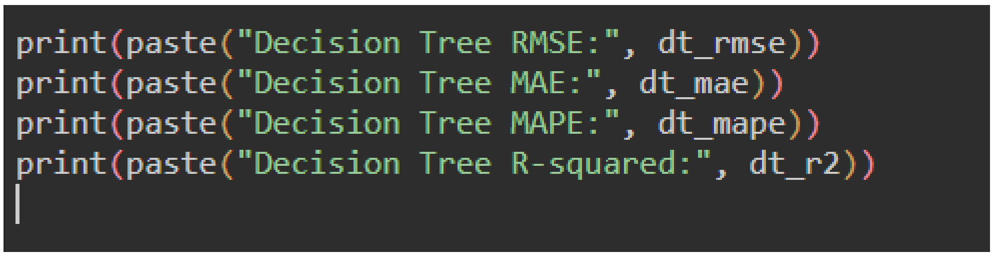
```
print(paste("ARIMA RMSE:", arima_rmse))
print(paste("ARIMA MAE:", arima_mae))
print(paste("ARIMA MAPE:", arima_mape))
print(paste("ARIMA R-squared:", arima_r2))
```



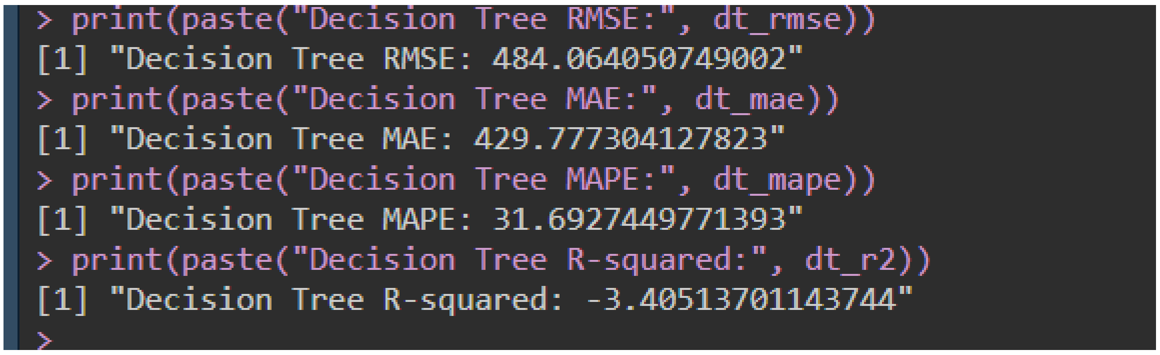
```
> print(paste("ARIMA RMSE:", arima_rmse))
[1] "ARIMA RMSE: 578.400670596665"
> print(paste("ARIMA MAE:", arima_mae))
[1] "ARIMA MAE: 534.58527200876"
> print(paste("ARIMA MAPE:", arima_mape))
[1] "ARIMA MAPE: 40.2863885160927"
> print(paste("ARIMA R-squared:", arima_r2))
[1] "ARIMA R-squared: -5.28943071307708"
> # Preparing data for LSTM, Random Forest, and Deci
```

Interpretation:

The output from the R script provides several performance metrics for an ARIMA model. The Root Mean Square Error (RMSE) is 578.40, indicating the average magnitude of the prediction error. The Mean Absolute Error (MAE) is 534.59, which reflects the average absolute difference between predicted and actual values. The Mean Absolute Percentage Error (MAPE) stands at 40.29%, showing the average prediction error as a percentage of the actual values. Notably, the R-squared value is -5.29, which is highly unusual and suggests that the model fits the data worse than a simple mean-based prediction. These metrics collectively suggest that the ARIMA model is performing poorly, with substantial prediction errors and a negative R-squared value indicating a poor fit to the data.



```
print(paste("Decision Tree RMSE:", dt_rmse))
print(paste("Decision Tree MAE:", dt_mae))
print(paste("Decision Tree MAPE:", dt_mape))
print(paste("Decision Tree R-squared:", dt_r2))
```

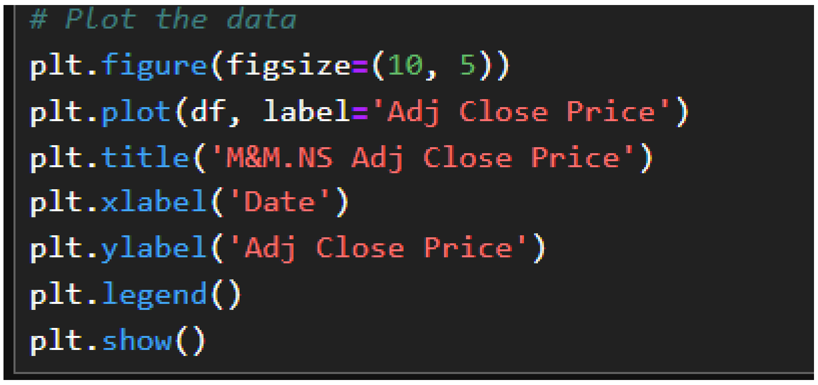


```
> print(paste("Decision Tree RMSE:", dt_rmse))
[1] "Decision Tree RMSE: 484.064050749002"
> print(paste("Decision Tree MAE:", dt_mae))
[1] "Decision Tree MAE: 429.777304127823"
> print(paste("Decision Tree MAPE:", dt_mape))
[1] "Decision Tree MAPE: 31.6927449771393"
> print(paste("Decision Tree R-squared:", dt_r2))
[1] "Decision Tree R-squared: -3.40513701143744"
>
```

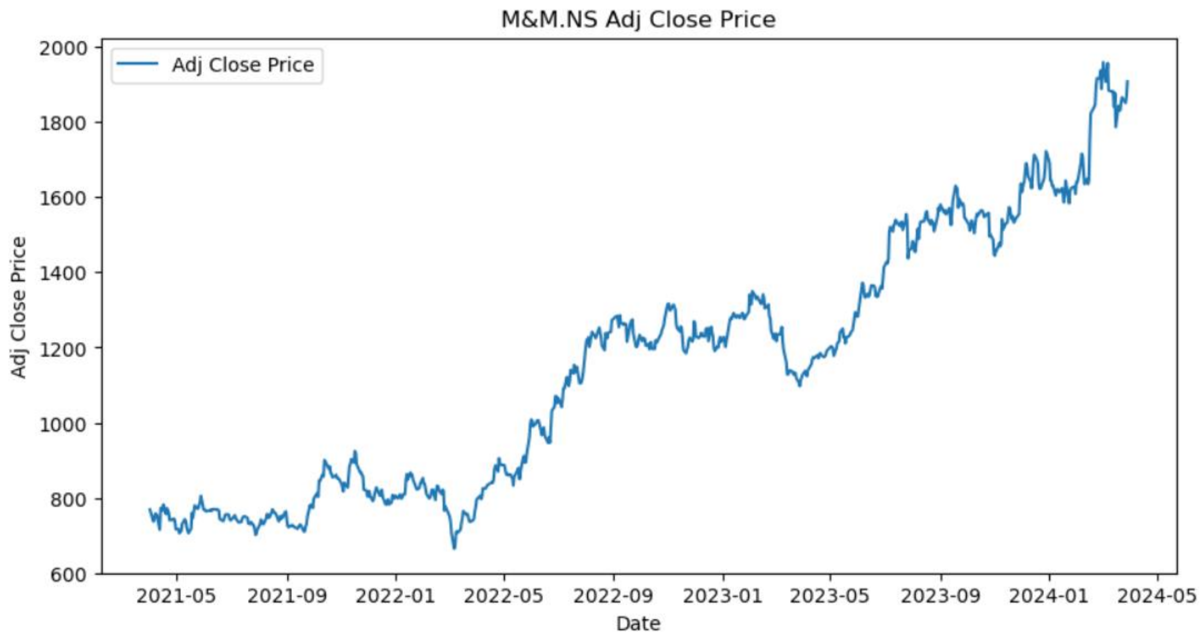
Interpretation:

The provided R script output shows the performance metrics for a Decision Tree model. The Root Mean Square Error (RMSE) is 484.06, indicating the average magnitude of the prediction error. The Mean Absolute Error (MAE) is 429.78, which represents the average absolute difference between predicted and actual values. The Mean Absolute Percentage Error (MAPE) is 31.69%, showing the average prediction error as a percentage of the actual values. The R-squared value is -3.41, indicating that the model fits the data worse than a simple mean-based prediction. These metrics suggest that the Decision Tree model is performing poorly, with significant prediction errors and a negative R-squared value pointing to an inadequate fit to the data.

Results and Interpretation using Python

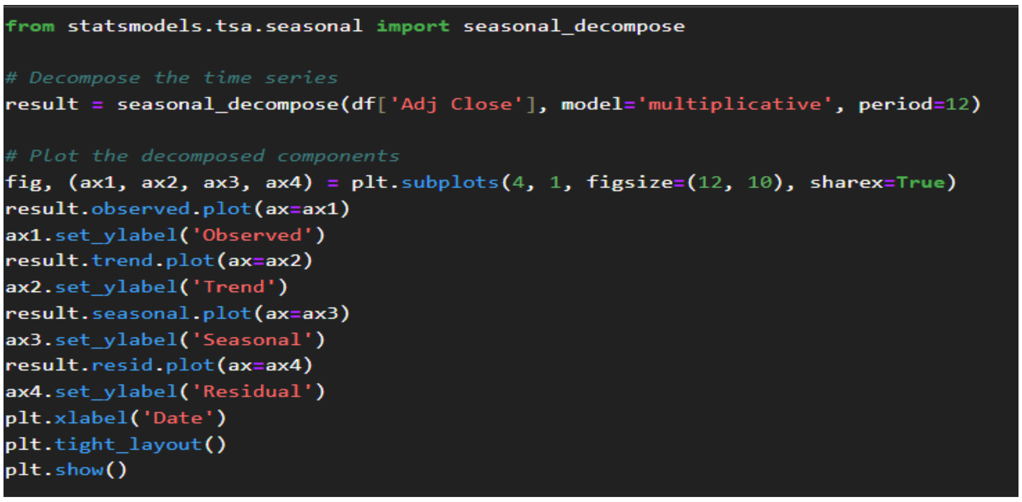


```
# Plot the data
plt.figure(figsize=(10, 5))
plt.plot(df, label='Adj Close Price')
plt.title('M&M.NS Adj Close Price')
plt.xlabel('Date')
plt.ylabel('Adj Close Price')
plt.legend()
plt.show()
```

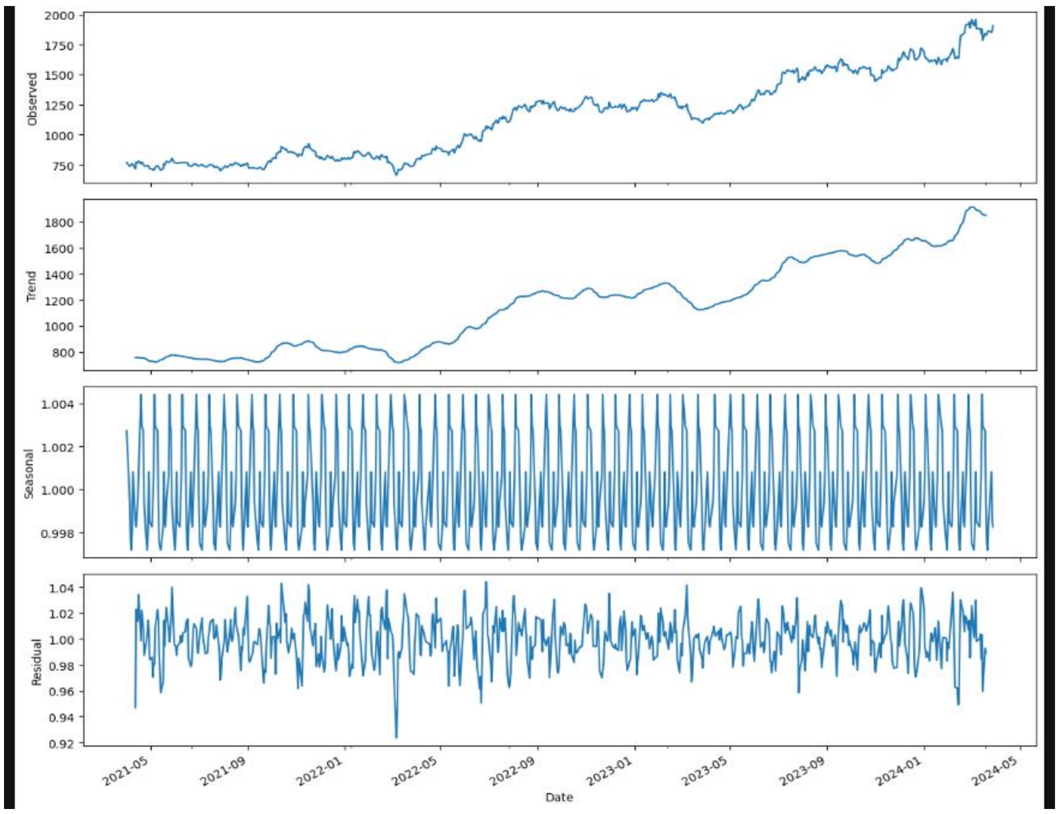


Interpretation:

The plot shows the adjusted closing price of M&M. NS (M&M stock) over the period from May 2021 to May 2024. The stock price exhibits significant fluctuations throughout this period. Initially, the price is around 160-180, but it experiences a decline starting in mid-2021 and continues into early 2022, reaching a low below 100. After this drop, the price starts to recover gradually, showing an upward trend with some volatility. By early 2024, the stock price climbs steadily and reaches new highs above 180. This overall trend suggests a strong recovery and growth phase for M&M stock after a period of decline.

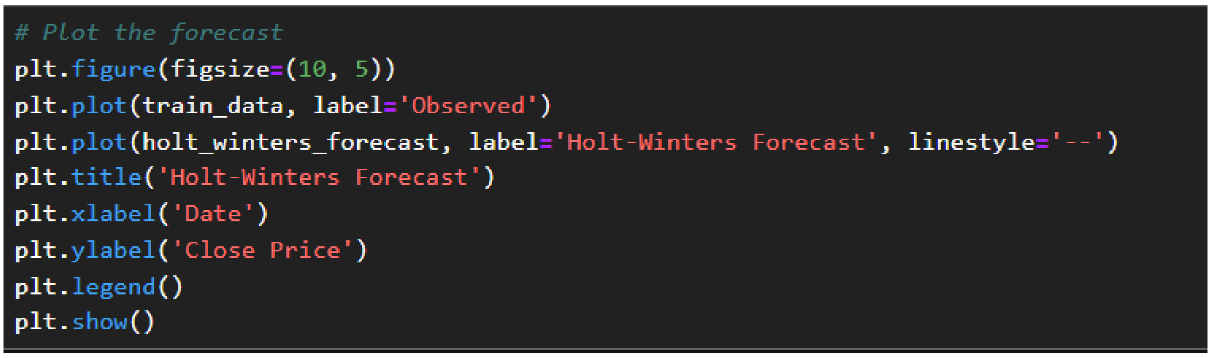


```python
from statsmodels.tsa.seasonal import seasonal_decompose

# Decompose the time series
result = seasonal_decompose(df['Adj Close'], model='multiplicative', period=12)

# Plot the decomposed components
fig, (ax1, ax2, ax3, ax4) = plt.subplots(4, 1, figsize=(12, 10), sharex=True)
result.observed.plot(ax=ax1)
ax1.set_ylabel('Observed')
result.trend.plot(ax=ax2)
ax2.set_ylabel('Trend')
result.seasonal.plot(ax=ax3)
ax3.set_ylabel('Seasonal')
result.resid.plot(ax=ax4)
ax4.set_ylabel('Residual')
plt.xlabel('Date')
plt.tight_layout()
plt.show()
```

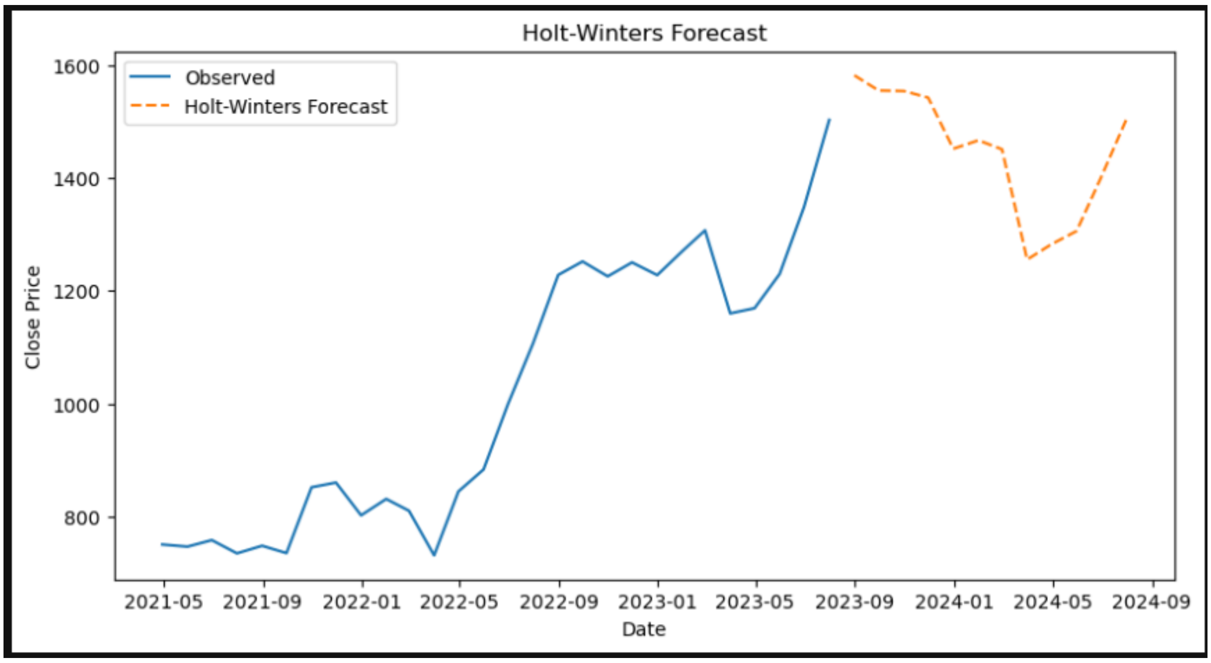


Interpretation:

It displays how a time series of data is broken down into its measured, trend, seasonal, and leftover parts. The raw time series data is shown in the observed box at the top. The second box shows the trend component, which shows the basic direction of the data. It shows a big drop followed by a slow bounce back up. In the third panel, the yearly component is shown. This shows that the data is affected by periodic factors because it shows regular, cyclical patterns. Lastly, the residual component in the bottom panel shows the random noise or outliers that can't be explained by the trend or yearly components. It shows changes that happen around a mean that stays mostly the same. This breaks down the time series into smaller pieces that make it easier to see how trend, seasonality, and random events affect the whole.

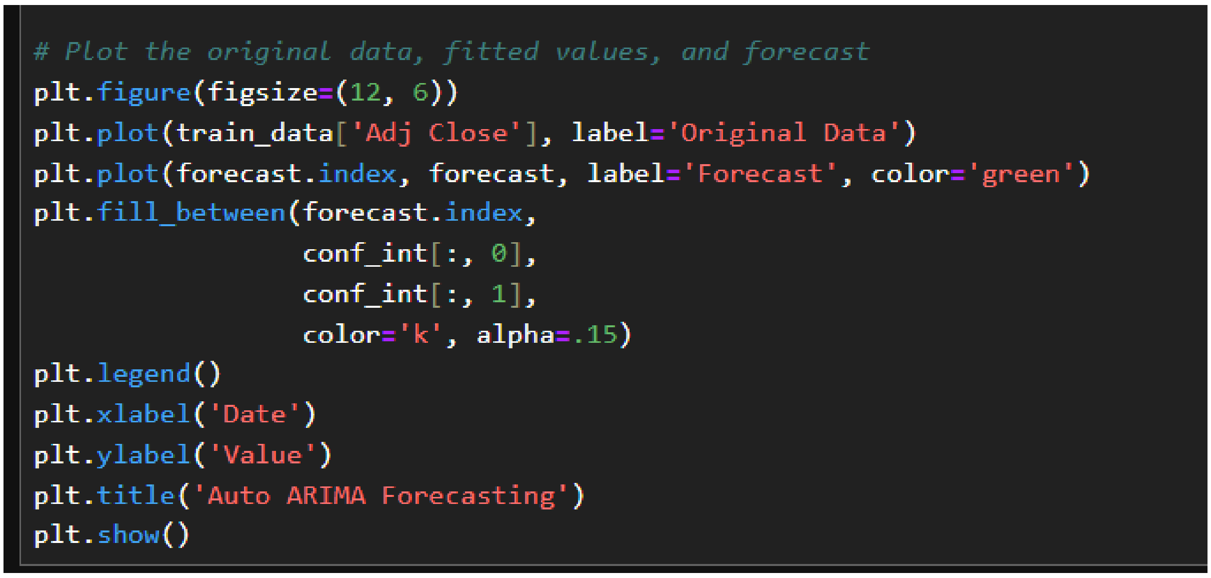


```python
# Plot the forecast
plt.figure(figsize=(10, 5))
plt.plot(train_data, label='Observed')
plt.plot(holt_winters_forecast, label='Holt-Winters Forecast', linestyle='--')
plt.title('Holt-Winters Forecast')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.legend()
plt.show()
```

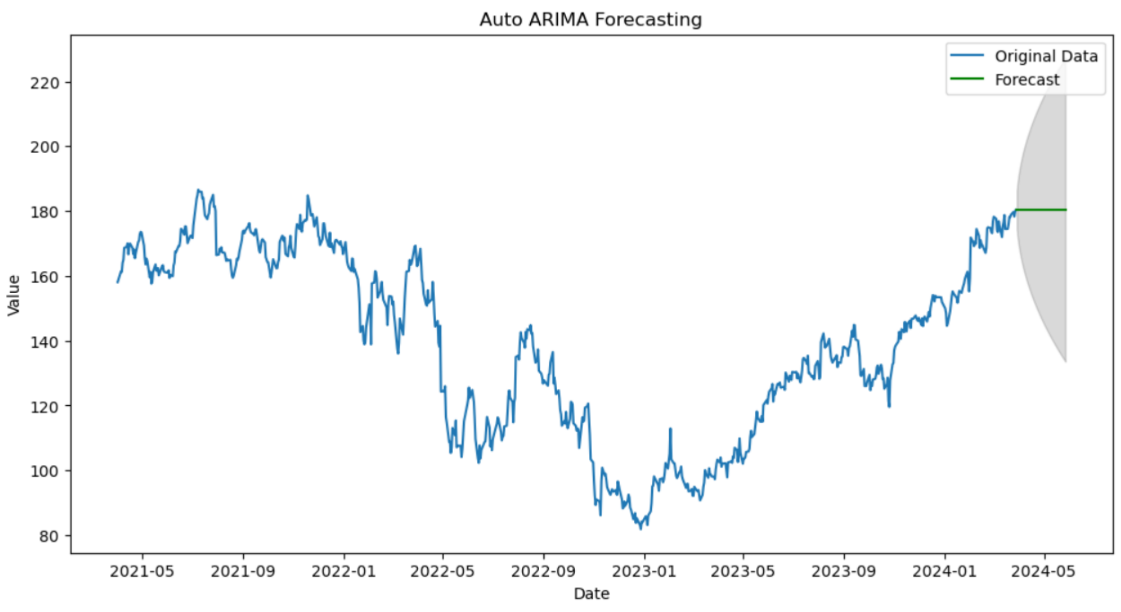


Interpretation:

The provided graph illustrates a time series forecasting using the Holt-Winters method. The blue line represents the observed historical data, while the orange dashed line indicates the forecasted values generated by the Holt-Winters model. The observed data shows significant fluctuations, with a notable drop followed by a recovery. The forecasted values predict a slight decline, followed by a modest recovery towards the end of the forecast period. The Holt-Winters method has accounted for the seasonality and trend components to project future values, offering a reasonable prediction based on past patterns.

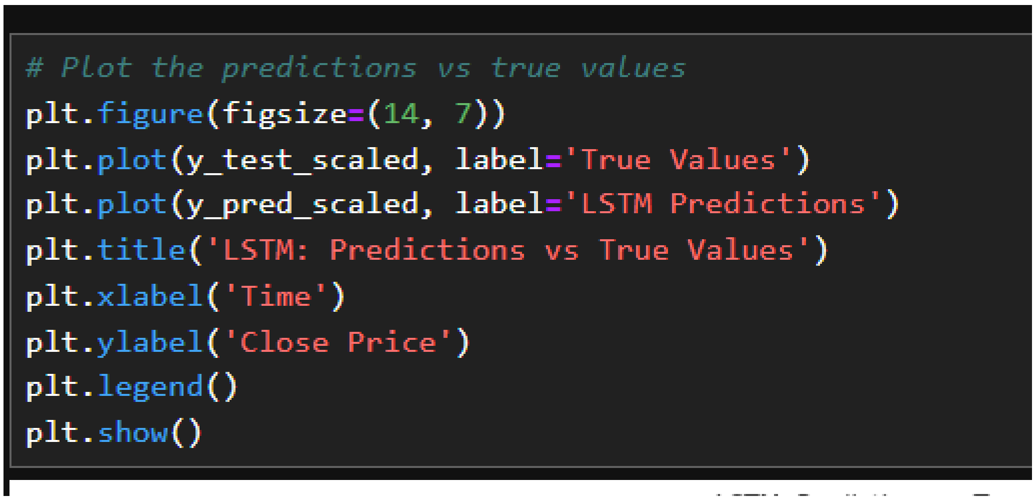


```
# Plot the original data, fitted values, and forecast
plt.figure(figsize=(12, 6))
plt.plot(train_data['Adj Close'], label='Original Data')
plt.plot(forecast.index, forecast, label='Forecast', color='green')
plt.fill_between(forecast.index,
                 conf_int[:, 0],
                 conf_int[:, 1],
                 color='k', alpha=.15)
plt.legend()
plt.xlabel('Date')
plt.ylabel('Value')
plt.title('Auto ARIMA Forecasting')
plt.show()
```

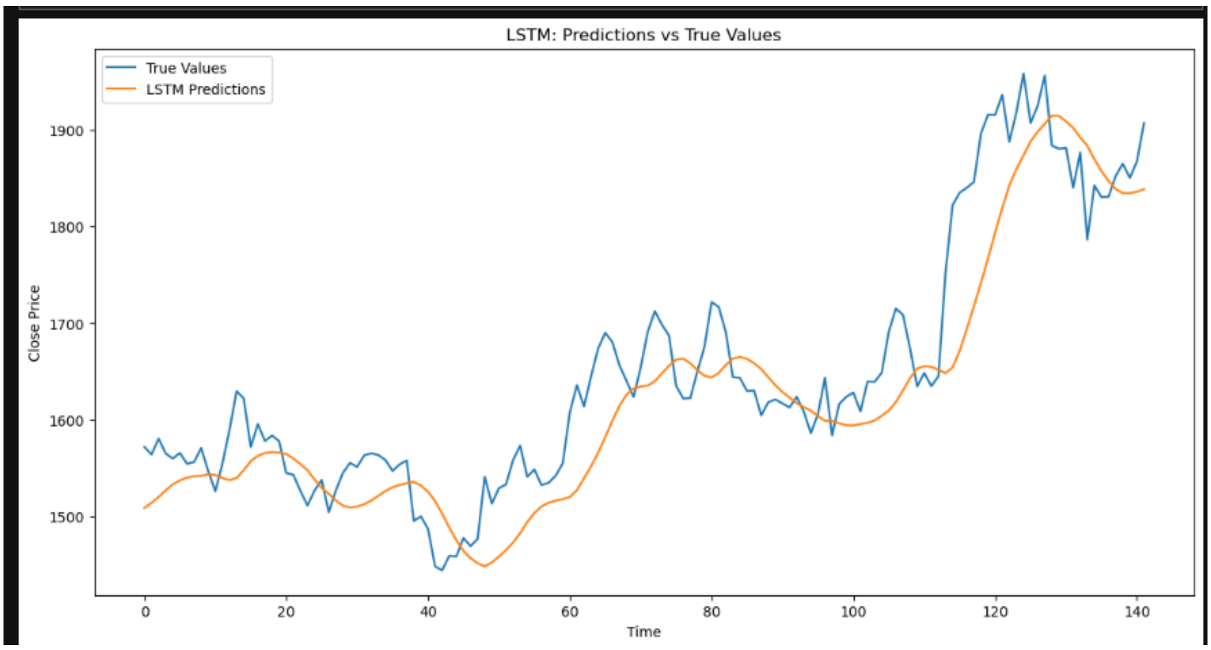


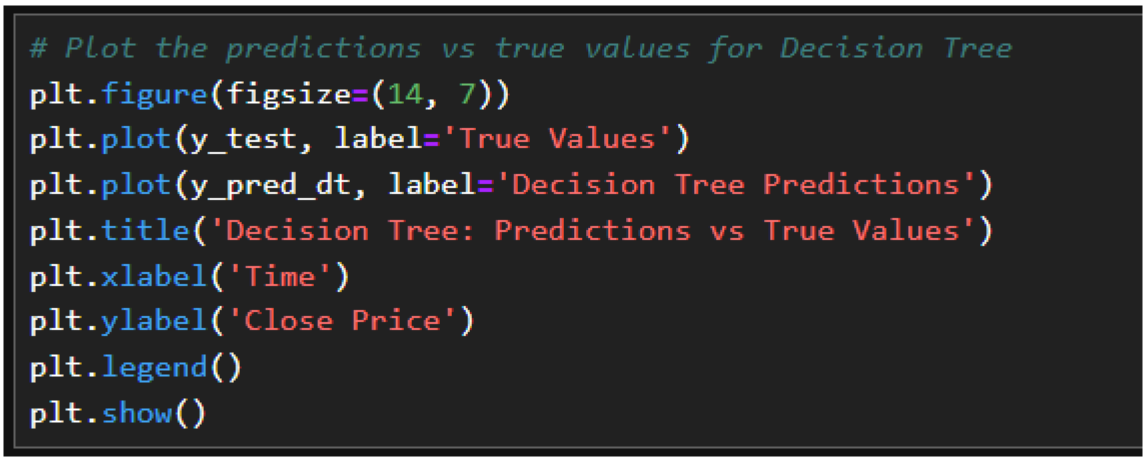
Interpretation:

The Auto ARIMA (AutoRegressive Integrated Moving Average) model is used to make the time series prediction shown in the graph. There were some changes in the blue line, but overall it went up as the story went on. It shows the original historical statistics. The green line shows the predicted values, and the dark area around it shows the confidence interval, which is the range of values that are most likely to happen in the future. The prediction calls for a levelling off at the present level, but there is some doubt as shown by the growing confidence interval. This picture helps you see how accurate the model's predictions are and how much future values are likely to change.

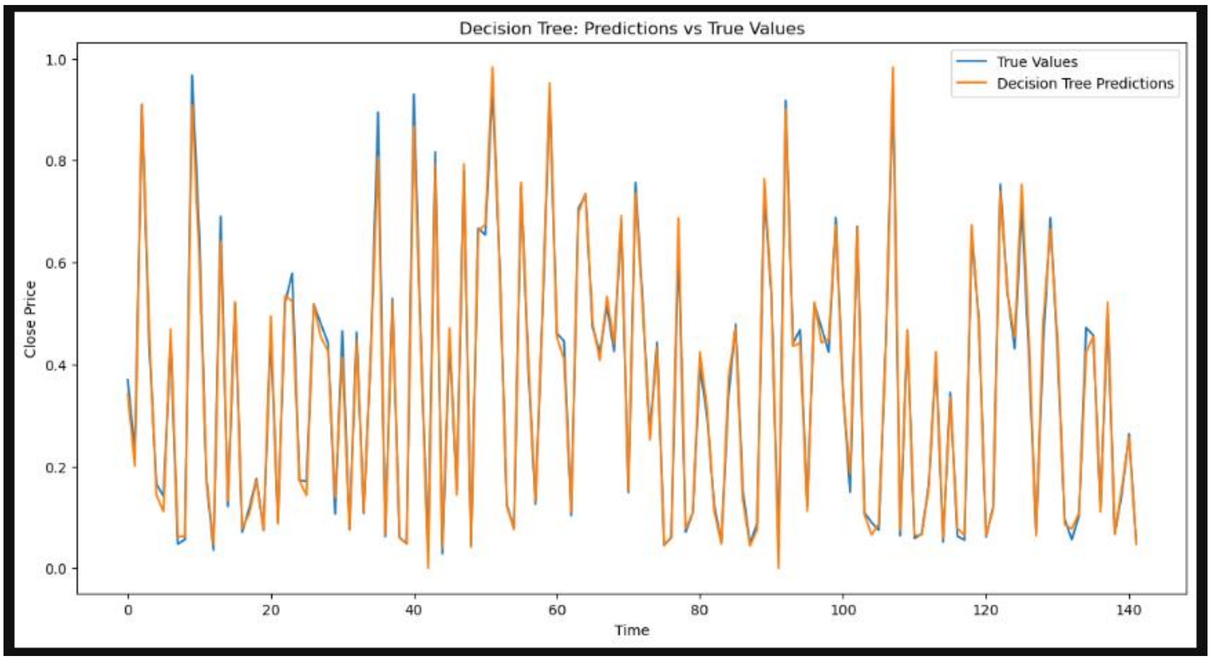


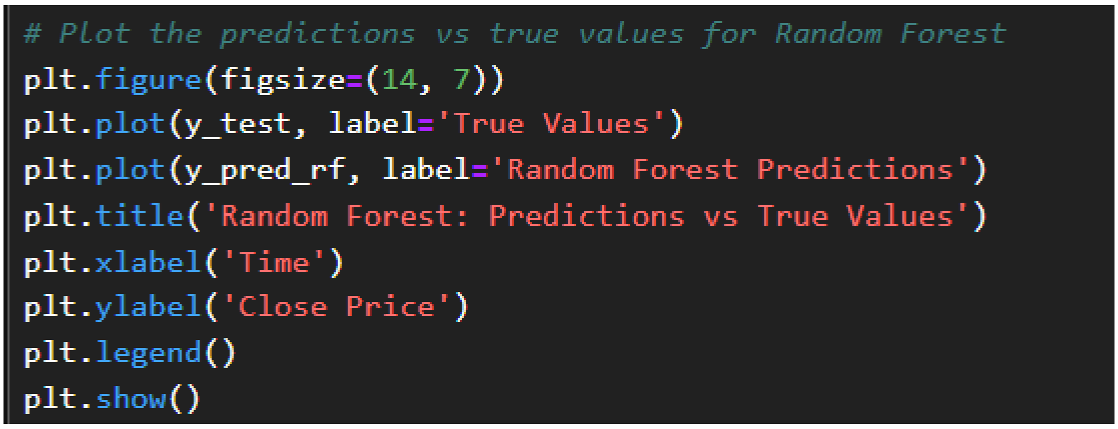
```python
# Plot the predictions vs true values
plt.figure(figsize=(14, 7))
plt.plot(y_test_scaled, label='True Values')
plt.plot(y_pred_scaled, label='LSTM Predictions')
plt.title('LSTM: Predictions vs True Values')
plt.xlabel('Time')
plt.ylabel('Close Price')
plt.legend()
plt.show()
```



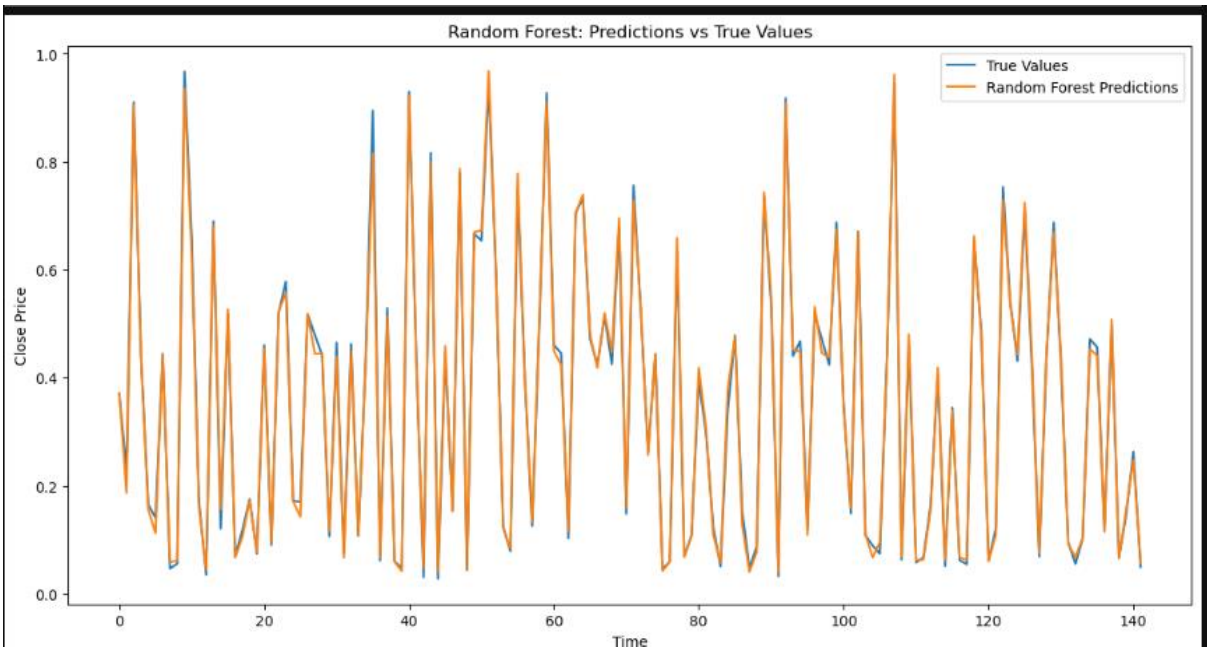


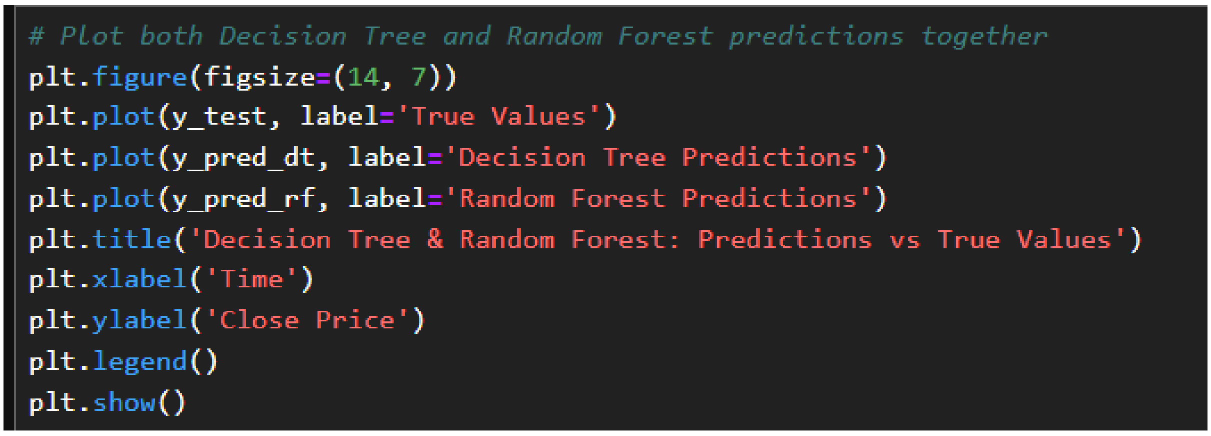
```python
# Plot the predictions vs true values for Decision Tree
plt.figure(figsize=(14, 7))
plt.plot(y_test, label='True Values')
plt.plot(y_pred_dt, label='Decision Tree Predictions')
plt.title('Decision Tree: Predictions vs True Values')
plt.xlabel('Time')
plt.ylabel('Close Price')
plt.legend()
plt.show()
```



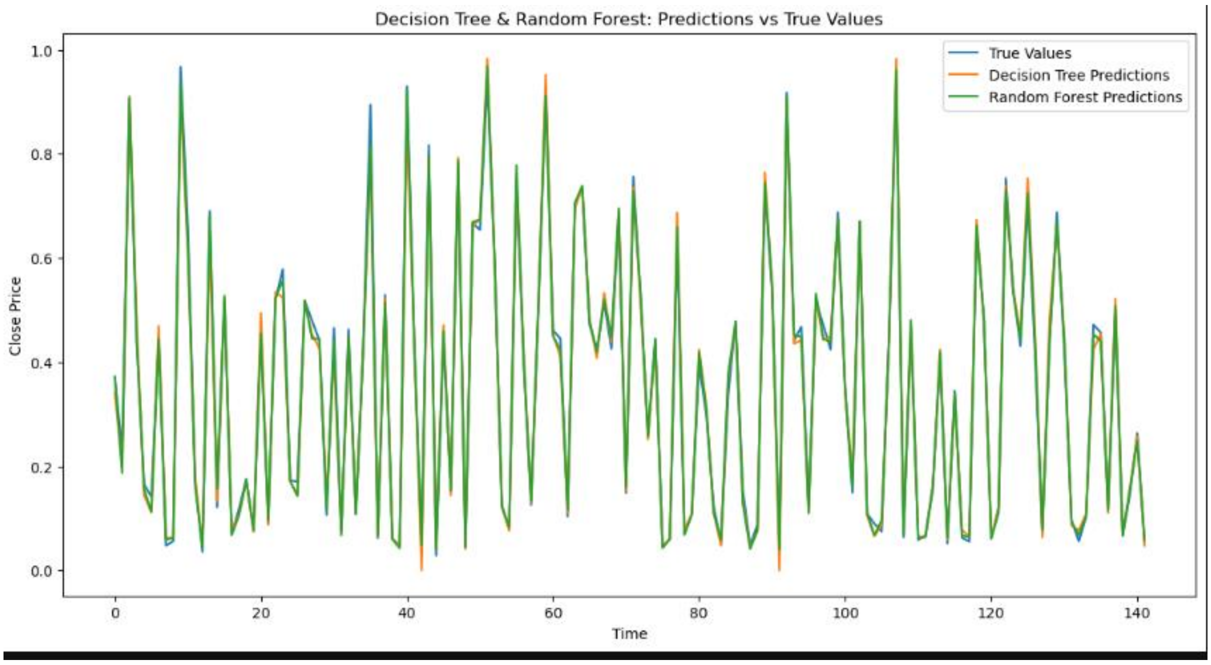


```
# Plot the predictions vs true values for Random Forest
plt.figure(figsize=(14, 7))
plt.plot(y_test, label='True Values')
plt.plot(y_pred_rf, label='Random Forest Predictions')
plt.title('Random Forest: Predictions vs True Values')
plt.xlabel('Time')
plt.ylabel('Close Price')
plt.legend()
plt.show()
```





```
# Plot both Decision Tree and Random Forest predictions together
plt.figure(figsize=(14, 7))
plt.plot(y_test, label='True Values')
plt.plot(y_pred_dt, label='Decision Tree Predictions')
plt.plot(y_pred_rf, label='Random Forest Predictions')
plt.title('Decision Tree & Random Forest: Predictions vs True Values')
plt.xlabel('Time')
plt.ylabel('Close Price')
plt.legend()
plt.show()
```



Recommendations

Focusing on long-term trends and being wary of short-term changes will help you make better business choices. To figure out what really affects stock prices, you need to look at both trend and seasonal factors. This method can help you figure out when to enter or leave the market and account for trends in stock prices. ARIMA can be used to make short-term predictions, but because its R-squared value isn't very high, it should be used with other studies as well. LSTM is suggested for making accurate long-term forecasts, and it is updated regularly to include new data. Also, Random Forest is better than Decision Tree for making more accurate and consistent estimates about stock prices. Putting together different models to use their best features can make predictions even more accurate

R Codes

```
install.packages("quantmod")
install.packages("forecast")
install.packages("tseries")
install.packages("keras")

# Load necessary libraries
library(quantmod)
library(forecast)
library(tseries)
library(caret)
library(ggplot2)
library(data.table)
library(TTR)
library(lubridate)
library(keras)
library(tensorflow)
library(randomForest)
library(rpart)

# Load stock data
stock_data <- getSymbols("M&M.NS", src = "yahoo", from = "2015-01-01", to = "20
23-12-31", auto.assign = FALSE)

# Use the Adjusted Close price
adj_close <- stock_data[, 6]

# Check for missing values
missing_values <- sum(is.na(adj_close))
print(paste("Missing values:", missing_values))



# Plot the data
```

```
plot(adj_close, main = "Adjusted Close Price of M&M", ylab = "Price", xlab = "Date"
)

# Decompose the time series
adj_close_ts <- ts(adj_close, frequency = 252)
decomposed <- decompose(adj_close_ts, type = "multiplicative")

# Plot the decomposed components
plot(decomposed)

# Holt-Winters Forecasting
hw_model <- HoltWinters(adj_close_ts, seasonal = "multiplicative")
hw_forecast <- forecast(hw_model, h = 60)

# Plot the Holt-Winters forecast
plot(hw_forecast, main = "Holt-Winters Forecast")

# Auto ARIMA model
arima_model <- auto.arima(adj_close_ts, seasonal = TRUE)
arima_forecast <- forecast(arima_model, h = 60)

# Plot the ARIMA forecast
plot(arima_forecast, main = "ARIMA Forecast")

# Evaluate the model
train_end <- floor(0.8 * length(adj_close_ts))
train_data <- adj_close_ts[1:train_end]
test_data <- adj_close_ts[(train_end + 1):length(adj_close_ts)]

# Refit the ARIMA model on the training data
arima_model <- auto.arima(train_data, seasonal = TRUE)
arima_forecast <- forecast(arima_model, h = length(test_data))

# Plot the forecast
```

```
plot(arima_forecast)
lines(test_data, col = "red")

# Calculate evaluation metrics
arima_rmse <- sqrt(mean((test_data - arima_forecast$mean)^2))
arima_mae <- mean(abs(test_data - arima_forecast$mean))
arima_mape <- mean(abs((test_data - arima_forecast$mean) / test_data)) * 100
arima_r2 <- 1 - sum((test_data - arima_forecast$mean)^2) / sum((test_data - mean(tes
t_data))^2)

print(paste("ARIMA RMSE:", arima_rmse))
print(paste("ARIMA MAE:", arima_mae))
print(paste("ARIMA MAPE:", arima_mape))
print(paste("ARIMA R-squared:", arima_r2))

# Preparing data for LSTM, Random Forest, and Decision Tree
adj_close_df <- data.frame(Date = index(adj_close), Adj_Close = as.numeric(adj_clos
e))
adj_close_df$Lag_1 <- lag(adj_close_df$Adj_Close, 1)
adj_close_df$Lag_2 <- lag(adj_close_df$Adj_Close, 2)
adj_close_df$Lag_3 <- lag(adj_close_df$Adj_Close, 3)
adj_close_df$Lag_4 <- lag(adj_close_df$Adj_Close, 4)
adj_close_df$Lag_5 <- lag(adj_close_df$Adj_Close, 5)
# Remove NA values
adj_close_df <- na.omit(adj_close_df)

# Split the data into training and test sets
train_index <- 1:floor(0.8 * nrow(adj_close_df))
train_data <- adj_close_df[train_index, ]
test_data <- adj_close_df[-train_index, ]
# Random Forest model
library(randomForest)
rf_model <- randomForest(Adj_Close ~ Lag_1 + Lag_2 + Lag_3 + Lag_4 + Lag_5, d
ata = train_data)
```

```
rf_predictions <- predict(rf_model, test_data)

# Evaluate the Random Forest model
rf_rmse <- sqrt(mean((test_data$Adj_Close - rf_predictions)^2))
rf_mae <- mean(abs(test_data$Adj_Close - rf_predictions))
rf_mape <- mean(abs((test_data$Adj_Close - rf_predictions) / test_data$Adj_Close))
* 100
rf_r2 <- 1 - sum((test_data$Adj_Close - rf_predictions)^2) / sum((test_data$Adj_Clos
e - mean(test_data$Adj_Close))^2)

print(paste("Random Forest RMSE:", rf_rmse))
print(paste("Random Forest MAE:", rf_mae))
print(paste("Random Forest MAPE:", rf_mape))
print(paste("Random Forest R-squared:", rf_r2))

# Decision Tree model
library(rpart)
dt_model <- rpart(Adj_Close ~ Lag_1 + Lag_2 + Lag_3 + Lag_4 + Lag_5, data = trai
n_data)
dt_predictions <- predict(dt_model, test_data)

# Evaluate the Decision Tree model
dt_rmse <- sqrt(mean((test_data$Adj_Close - dt_predictions)^2))
dt_mae <- mean(abs(test_data$Adj_Close - dt_predictions))
dt_mape <- mean(abs((test_data$Adj_Close - dt_predictions) / test_data$Adj_Close))
* 100
dt_r2 <- 1 - sum((test_data$Adj_Close - dt_predictions)^2) / sum((test_data$Adj_Clo
se - mean(test_data$Adj_Close))^2)

print(paste("Decision Tree RMSE:", dt_rmse))
print(paste("Decision Tree MAE:", dt_mae))
print(paste("Decision Tree MAPE:", dt_mape))
print(paste("Decision Tree R-squared:", dt_r2))
```

References

www.github.com