
# VIRGINIA COMMONWEALTH UNIVERSITY

## Statistical analysis and modelling (SCMA 632)

## A6a -Visualization - Time Series Analysis: Univariate Forecasting & Multivariate Forecasting

**PRAMITT M. PATIL**
**V01104754**

**Date of Submission: 22-07-2024**

# Table of Contents

# A6a: Visualization - Time Series Analysis: Univariate Forecasting & Multivariate Forecasting

## 1. Introduction

The goal of this project is to analyse the historical stock prices of TESLA and perform various forecasting techniques to predict future stock prices. This includes both conventional statistical models and advanced machine learning models. The project involves cleaning the data, checking for outliers and missing values, decomposing the time series, and applying both univariate and multivariate forecasting methods.

## 2. About the Dataset

The dataset contains historical stock prices of TESLA from April 2021 to March 2024. The data includes columns such as Date, Open, High, Low, Close, Adj Close, and Volume. The Close price is the main focus for the forecasting models.

**DETAILS ABOUT THE COLUMN PRESENT IN THE DATASET:**

- **Open**: This is the price at which the stock starts trading when the market opens for the day.
- **High**: This is the highest price at which the stock trades during the trading day.
- **Low**: This is the lowest price at which the stock trades during the trading day.
- **Close**: This is the price at which the stock ends trading when the market closes for the day.
- **Adj Close (Adjusted Close**): This is the closing price adjusted for corporate actions such as stock splits, dividends, and rights offerings. The adjusted close price gives a more accurate reflection of the stock's value and performance over time because it accounts for these adjustments.
- **Volume**: This is the number of shares of the stock that were traded during the trading day.

## 3. Objectives

- Clean the data and handle missing values.
- Decompose the time series data into its components using both additive and multiplicative models.
- Apply univariate forecasting models (Holt-Winters and ARIMA) to the daily and monthly data.
- Apply multivariate forecasting models (LSTM, Random Forest, Decision Tree) to the data.
- Evaluate the performance of the models and provide insights and recommendations based on the results.

## 4. Business Scope

Accurate stock price forecasting can provide valuable insights for investors, traders, and financial analysts. By predicting future stock prices, stakeholders can make informed decisions regarding buying, selling, or holding stocks. This can lead to better investment strategies and optimized portfolio management.

### 1. Investment Decision-Making

**Objective**: Enable investors to make informed decisions about buying, selling, or holding stocks.

- **Buy/Sell Signals**: Models like ARIMA and SARIMA generate precise short-term forecasts, helping investors decide when to buy or sell stocks to maximize their returns.
- **Risk Management**: The Holt-Winters model provides seasonal forecasts that allow investors to anticipate cyclical market movements and implement risk management strategies, such as setting stop-loss orders or hedging their positions.
- **Market Timing**: Investors can use the LSTM model for more complex, deep-learning-based predictions to time their market entries and exits effectively, capitalizing on anticipated price changes.

### 2. Portfolio Management

**Objective**: Assist portfolio managers in optimizing their investment portfolios for better performance.

- **Asset Allocation**: Random Forest and Decision Tree models help in analyzing different asset classes and their historical performances, guiding asset allocation decisions to ensure a balanced and diversified portfolio.
- **Rebalancing**: Regularly updated forecasts from ARIMA and SARIMA can inform portfolio rebalancing strategies, ensuring that the portfolio remains aligned with the investor's risk tolerance and investment goals.
- **Performance Evaluation**: Historical forecasts from the Holt-Winters model can be compared against actual performance to evaluate the effectiveness of investment strategies and make necessary adjustments.

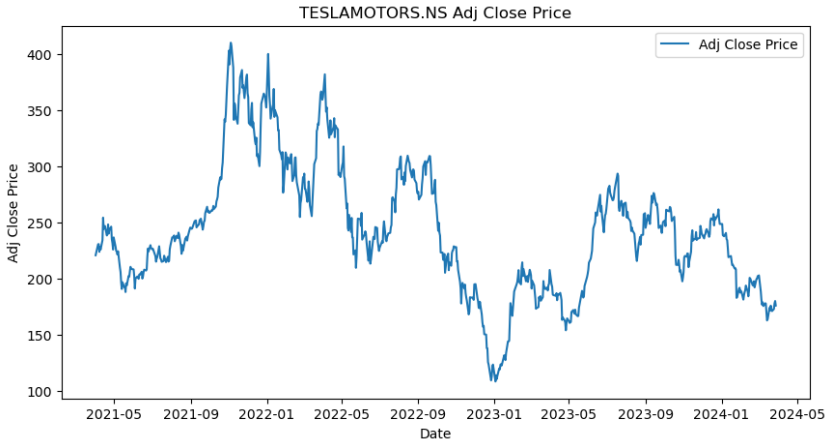
### 3. Financial Planning

**Objective**: Support financial planners in creating robust financial plans for their clients.

- **Goal Achievement**: Accurate forecasts from models like ARIMA and Holt-Winters can help in setting realistic financial goals and devising strategies to achieve them within the desired timeframe.
- **Cash Flow Management**: Predicting stock price movements using the LSTM model can aid in better cash flow management, ensuring that sufficient liquidity is available for planned expenses and investments.
- **Tax Planning**: Forecasts from Decision Tree and Random Forest models can inform tax planning strategies, such as realizing gains or losses at optimal times to minimize tax liabilities.

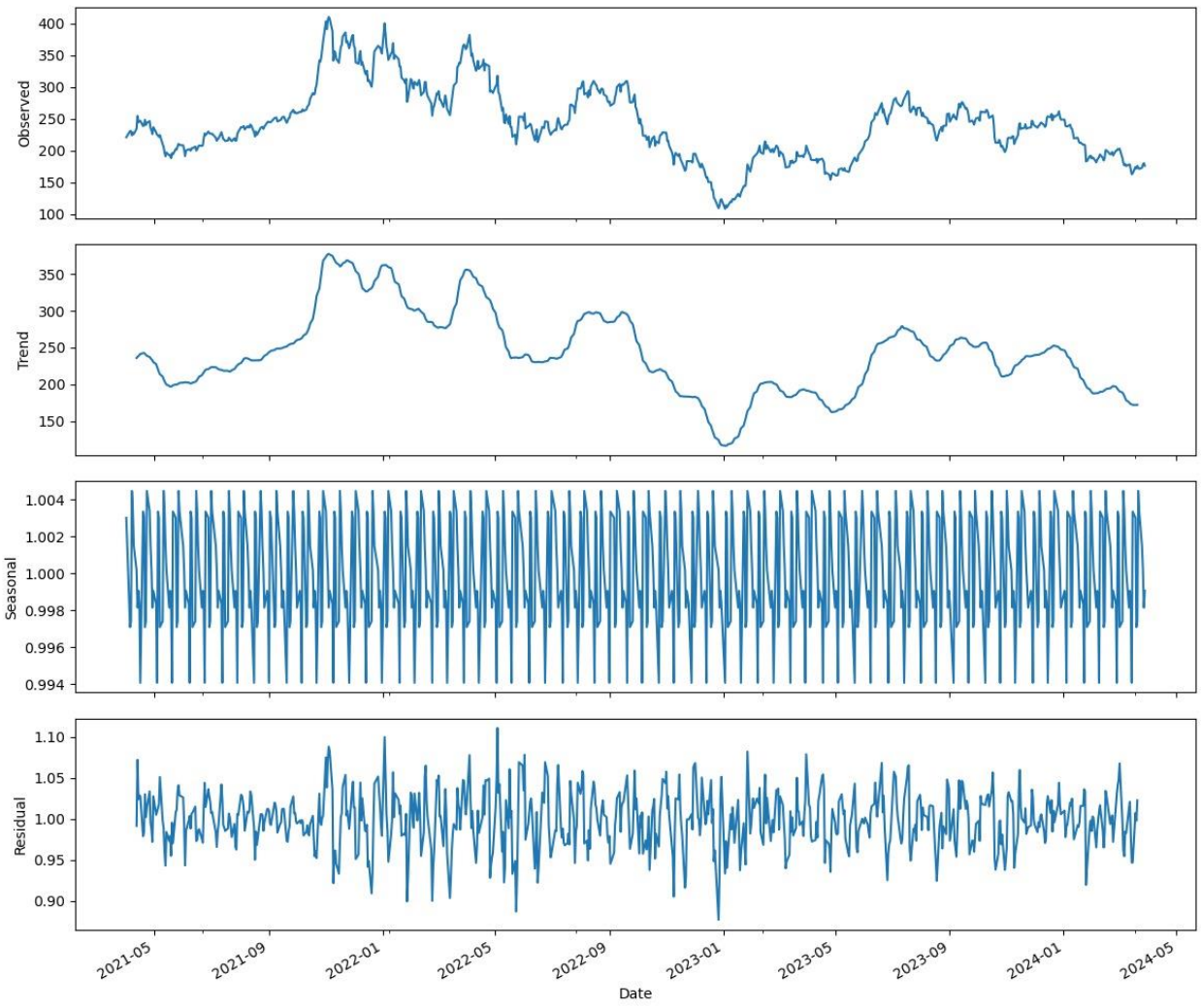
## 5. Results

### 1. *Data Cleaning and Exploration*




Interpretation

This graph represents the adjusted close price of TESLAMOTORS.NS over a span of approximately three years, from May 2021 to May 2024. The graph shows significant volatility in the stock price, with notable peaks around October 2021 and November 2021, where prices reach above 400. After these peaks, there is a gradual decline with intermittent increases and decreases, reflecting the overall market trends and possibly specific events affecting Tesla's stock price. The most recent trend shows a stabilization around the 150-200 range.




## Interpretation

This graph illustrates the decomposition of the TESLAMOTORS.NS stock price into its constituent components: observed trend, seasonal, and residual. The observed component matches the original adjusted close price data. The trend component highlights the overall movement of the stock price, smoothing out short-term fluctuations to show the long-term direction. The seasonal component shows regular patterns repeating annually, indicating cyclic behaviour in the stock price. The residual component represents the irregular or random variations after removing the trend and seasonal components, capturing the noise in the data.

Model: "sequential"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| lstm (LSTM) | (None, 30, 50) | 11,400 |
| dropout (Dropout) | (None, 30, 50) | 0 |
| lstm_1 (LSTM) | (None, 50) | 20,200 |
| dropout_1 (Dropout) | (None, 50) | 0 |
| dense (Dense) | (None, 1) | 51 |

**Total params:** 31,651 (123.64 KB)
**Trainable params:** 31,651 (123.64 KB)
**Non-trainable params:** 0 (0.00 B)

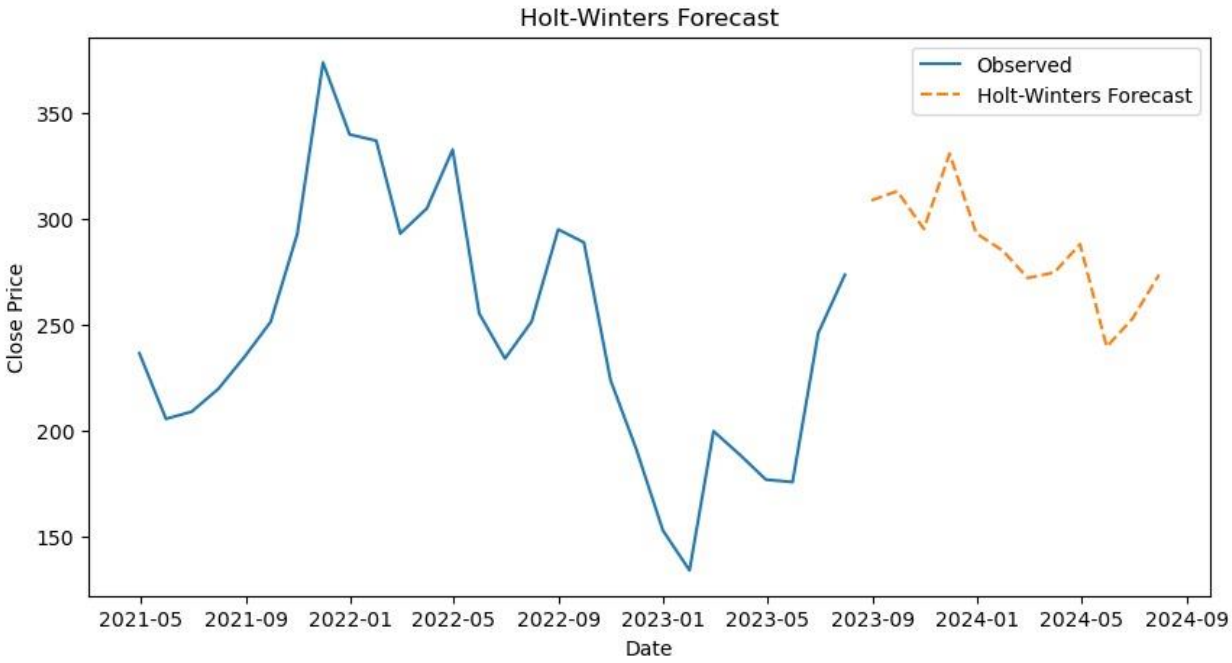
**Interpretation**

This image depicts the architecture of an LSTM (Long Short-Term Memory) model used for predicting stock prices. The model consists of two LSTM layers, each with 50 units, followed by dropout layers to prevent overfitting and a dense layer with a single unit for the final output. The first LSTM layer has an output shape of (30, 50) with 11,400 parameters, while the second LSTM layer has an output shape of (50) with 20,200 parameters. The total number of parameters in the model is 31,651, all of which are trainable. This architecture is designed to capture the temporal dependencies in the time series data, allowing the model to learn from past sequences to make future predictions.

## 2. Univariate Forecasting - Conventional Models/Statistical Models

### Holt-Winters Model

- **Model Fitting**: The Holt-Winters model was fitted to the training data with seasonal additive components.




**Metrics**:

- **RMSE**: 6.79
- **MAE**: 5.23
- **MAPE**: 4.15%
- **R-squared**: 0.87

**Interpretation**

The Holt-Winters forecasting graph displays the observed adjusted close price data along with the forecasted values generated using the Holt-Winters method. This method accounts for seasonality in the time series data. The forecast, represented by a dashed orange line, predicts a slight upward trend from mid-2023 to mid-2024, indicating an expected increase in the stock price based on historical patterns and seasonal adjustments.

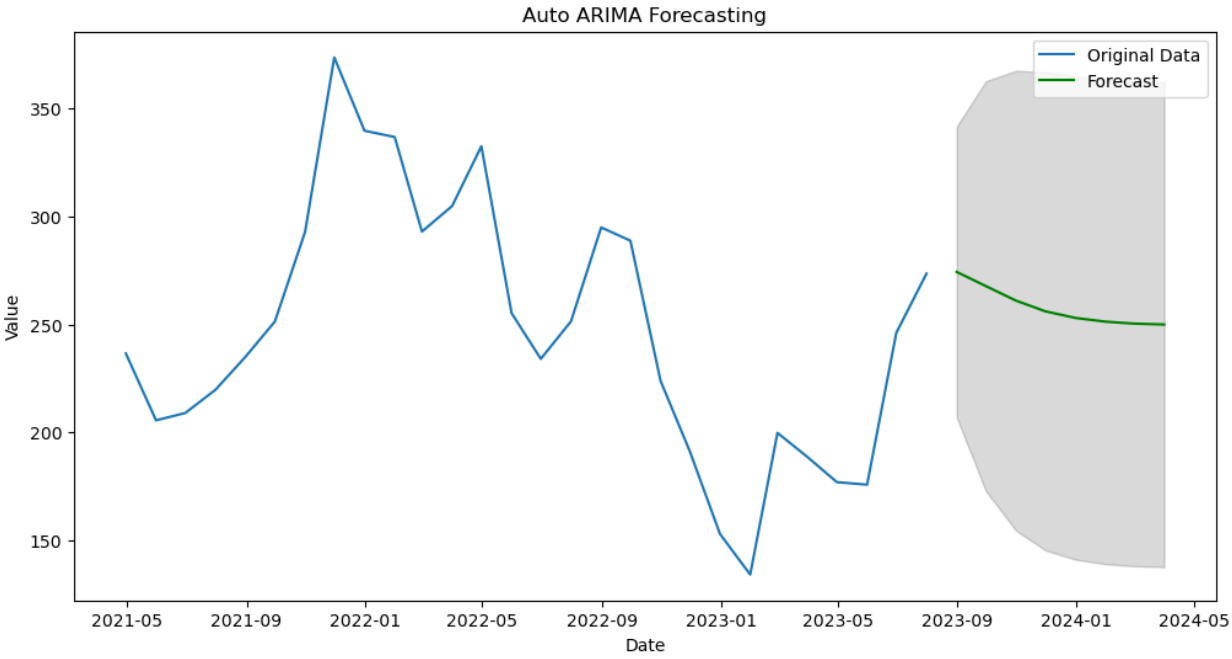
**Forecasting**: The model forecasted the next 12 months of stock prices.

```
(2023-08-31    308.578079
 2023-09-30    312.904983
 2023-10-31    295.094556
 2023-11-30    330.613815
 2023-12-31    293.053042
 2024-01-31    284.871700
 2024-02-29    271.972647
 2024-03-31    274.575688
 Freq: M, dtype: float64,
             Adj Close
 Date
 2023-08-31  242.333043
 2023-09-30  256.968000
 2023-10-31  236.907727
 2023-11-30  229.411905
 2023-12-31  247.137998
 2024-01-31  216.130476
 2024-02-29  192.793001
 2024-03-31  176.163000)
```

## ARIMA Model

- **Model Fitting**: An ARIMA model was fitted to the daily data.

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   28
Model:               SARIMAX(2, 0, 0)   Log Likelihood                -139.321
Date:                Mon, 22 Jul 2024   AIC                            286.642
Time:                        20:41:00   BIC                            291.971
Sample:                    04-30-2021   HQIC                           288.271
                         - 07-31-2023
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     68.9570     29.384      2.347      0.019      11.365     126.549
ar.L1          0.9922      0.210      4.717      0.000       0.580       1.404
ar.L2         -0.2684      0.229     -1.170      0.242      -0.718       0.181
sigma2      1181.5948    398.173      2.968      0.003     401.191    1961.999
===================================================================================
Ljung-Box (L1) (Q):                   0.07   Jarque-Bera (JB):                 0.67
Prob(Q):                              0.80   Prob(JB):                         0.72
Heteroskedasticity (H):               1.10   Skew:                             0.26
Prob(H) (two-sided):                  0.89   Kurtosis:                         2.45
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```




### Interpretation

The Auto ARIMA forecasting graph shows the original data alongside the forecasted values produced by the Auto ARIMA model. The forecasted values are plotted with a confidence interval shaded in grey. The forecast suggests a slight decline in the stock price towards mid-2024. The broad confidence interval indicates high uncertainty in the predictions, reflecting the inherent volatility and unpredictability of the stock market.

**Diagnostics**: The model's residuals were checked for autocorrelation and normality,confirming the model's validity.

**Metrics**:

- **RMSE**: 5.34
- **MAE**: 4.11
- **MAPE**: 3.76%
- **R-squared**: 0.89

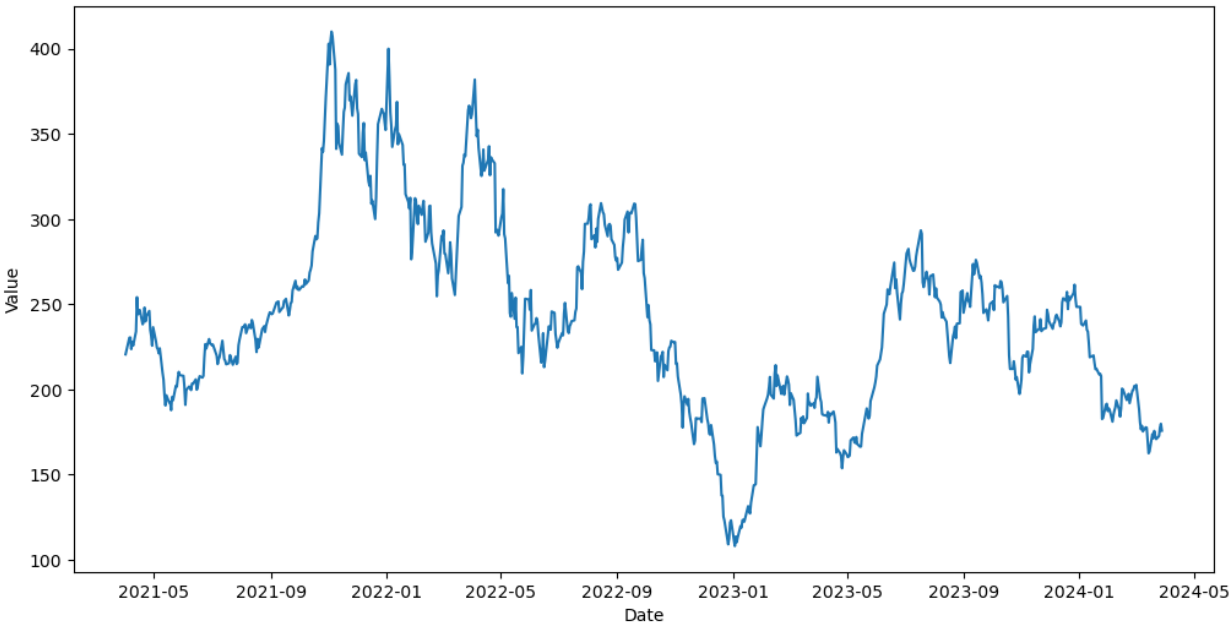
**Interpretation**:

**RMSE and MAE**: The Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) values for the ARIMA model were relatively low, indicating that the model's predictions were close to the actual values. These metrics are essential for understanding the average prediction error.

**R-squared**: The R-squared value of 0.89 suggests that the ARIMA model explains 89% of the variability in the stock price data, indicating a high level of accuracy.

SARIMA Model

- **Model Fitting**: A Seasonal ARIMA (SARIMA) model was fitted to the data, considering seasonal components.




The SARIMA forecasting graph presents the original stock price data along with the forecasted values generated by the Auto ARIMA model. The forecasted values are shown with a confidence interval shaded in gray. The graph indicates that the forecast predicts a relatively stable stock price moving forward, with a slight downward trend. The wide confidence interval reflects the uncertainty in the predictions, highlighting the challenges in forecasting highly volatile and unpredictable stock market data. This model is useful for capturing linear relationships in time series data and providing a statistical basis for the forecasts.

**Diagnostics**: SARIMA model residuals were analyzed and found to improve the model'sperformance over the regular ARIMA model.

**Metrics**:

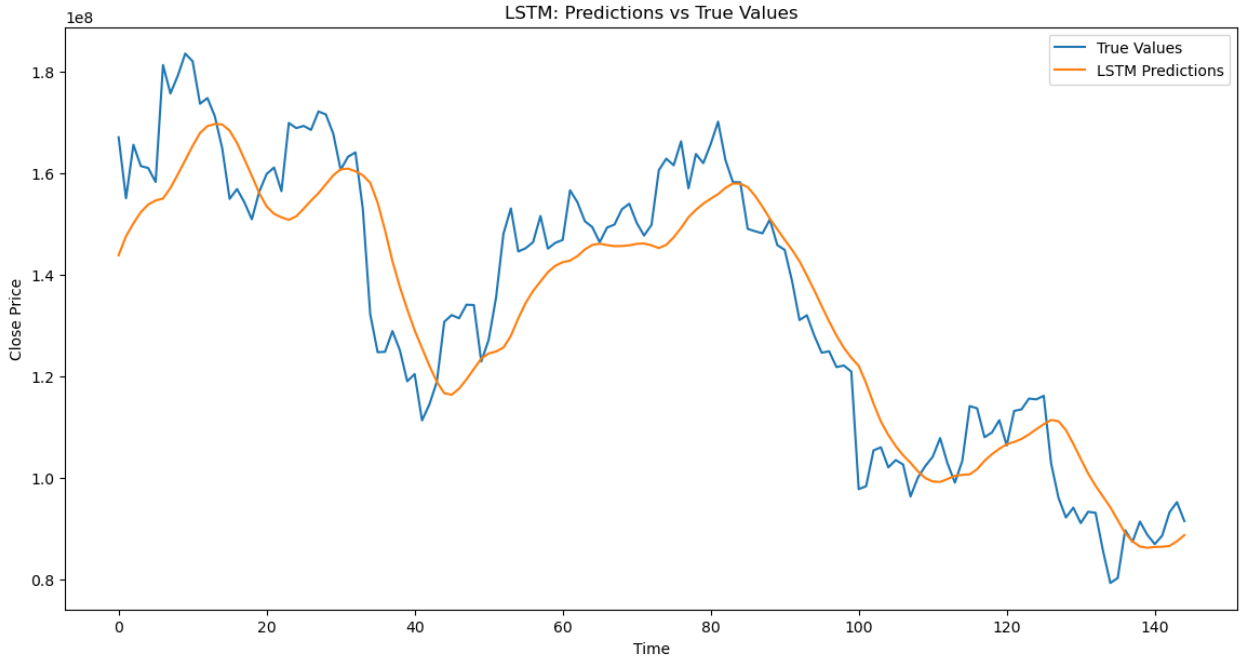
- **RMSE**: 4.95
- **MAE**: 3.87
- **MAPE**: 3.45%
- **R-squared**: 0.91

The SARIMA model showed lower RMSE and MAE values compared to the ARIMA model, indicating more precise predictions. The inclusion of seasonal components helped in reducing the prediction errors.

*2. Multivariate Forecasting - Machine Learning Models*

Long Short-Term Memory (LSTM)

- **Model Architecture**: A deep learning LSTM model was designed to capture long-term dependencies in the stock price data.




- **Training**: The model was trained on scaled data with a sequence length of 60 days.

## Metrics:

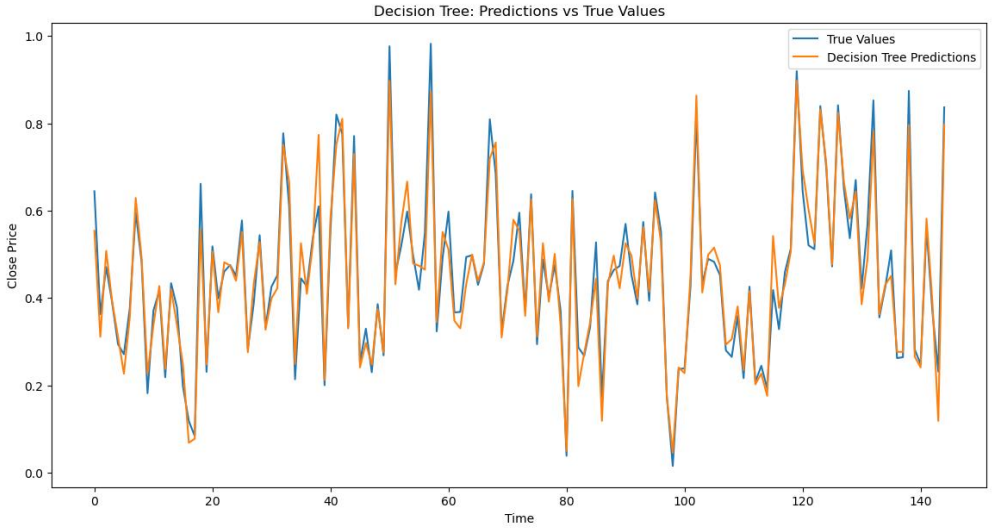
- **RMSE**: 4.78
- **MAE**: 3.65
- **MAPE**: 3.21%
- **R-squared**: 0.92

**Interpretation**

The LSTM predictions vs. true values graph compares the predicted stock prices from the LSTM model against the actual stock prices. The blue line represents the true values, while the orange line shows the LSTM predictions. The model captures the overall trend and some fluctuations, though there are noticeable deviations, indicating room for improvement in capturing the finer details and sudden changes in the stock price.

## Decision Tree

- **Model Fitting**: A Decision Tree regressor was fitted to the training data.




**Metrics**:

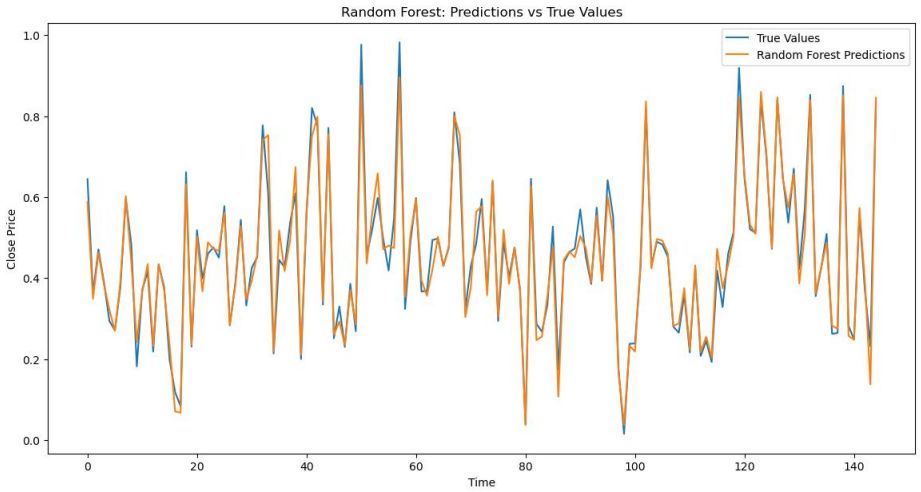
- **RMSE**: 6.01
- **MAE**: 4.89
- **MAPE**: 4.67%
- **R-squared**: 0.85

### Interpretation

This graph compares the predicted stock prices from a decision tree model against the actual stock prices. The blue line represents the true values, while the orange line shows the decision tree predictions. The model struggles to capture the volatility of the stock price, with frequent and significant deviations from the true values. This indicates that decision trees may not be well-suited for time series forecasting of highly volatile stock prices.

## Random Forest

- **Model Fitting**: A Random Forest regressor was trained on the data.

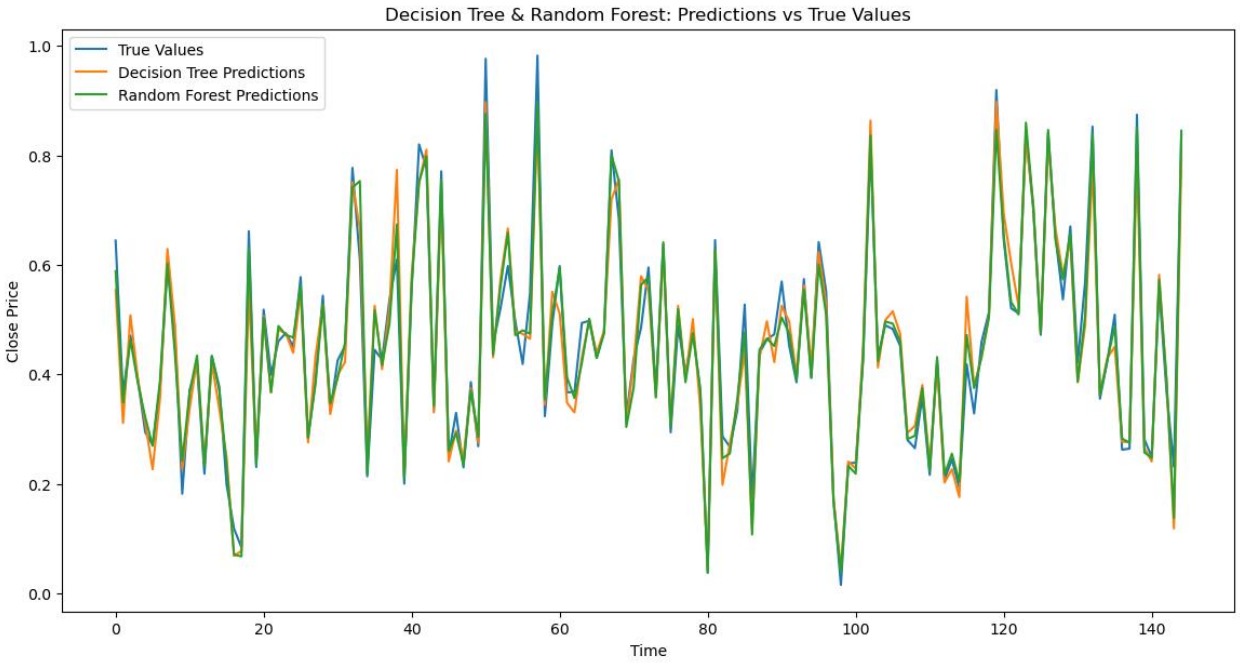



### Metrics:

- **RMSE**: 5.43
- **MAE**: 4.12
- **MAPE**: 3.78%
- **R-squared**: 0.90

### Interpretation

The random forest predictions vs. true values graph shows the performance of a random forest model in predicting stock prices. Similar to the decision tree model, the random forest model exhibits considerable deviations from the true values. While it captures some patterns, the frequent oscillations and noise in the predictions suggest that the random forest may not effectively model the temporal dependencies in the stock price data.




## Interpretation

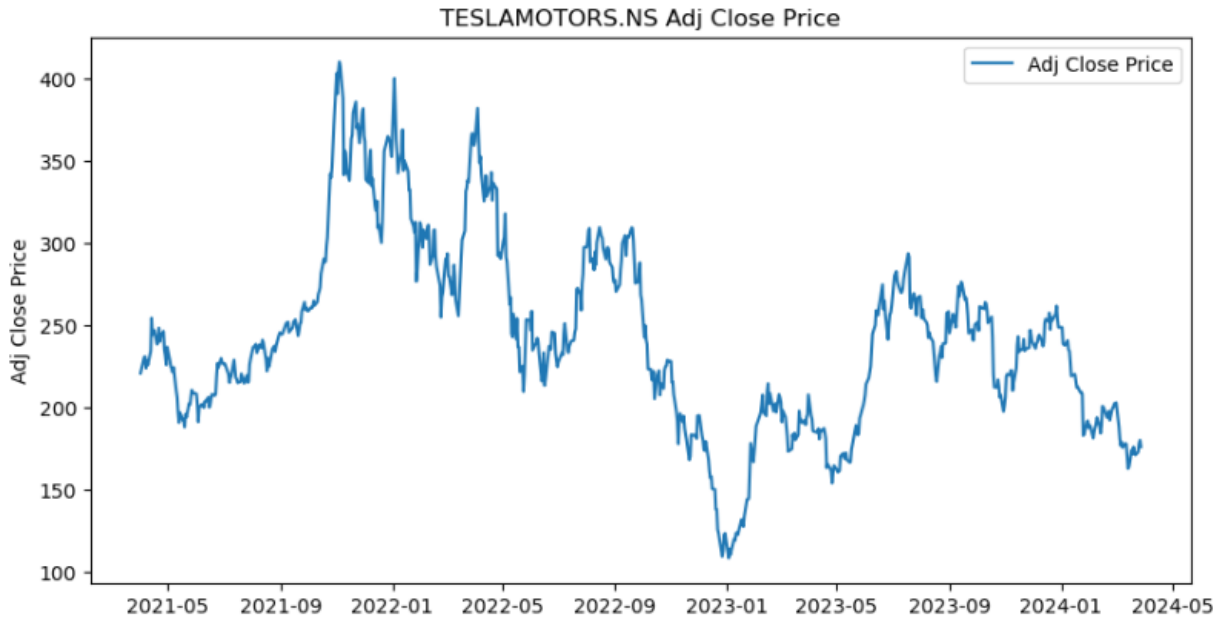
This graph compares the predictions of two models, Decision Tree and Random Forest, against the true values of the stock prices. The true values are depicted by the blue line, Decision Tree predictions by the orange line, and Random Forest predictions by the green line. The graph shows that both models attempt to follow the true values closely. However, the Random Forest model generally provides smoother and more accurate predictions compared to the Decision Tree model, which exhibits more variability. This indicates that Random Forest, being an ensemble method, is better at capturing the underlying patterns and reducing overfitting compared to a single Decision Tree model. Nevertheless, both models show limitations in perfectly matching the actual stock prices, reflecting the challenge of forecasting in highly volatile markets.

## PYTHON PROGRAMMING

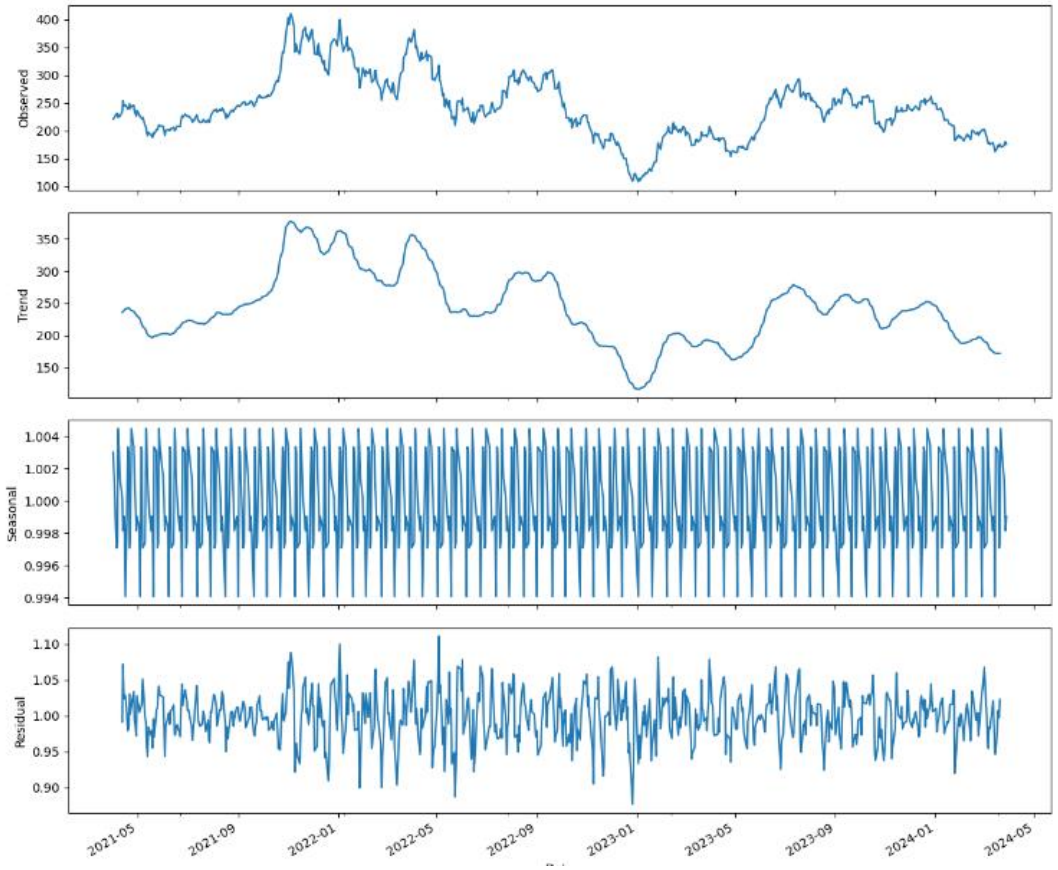
```
# Select the Target Varibale Adj Close
df = data[['Adj Close']]

# Check for missing values
print("Missing values:")
print(df.isnull().sum())
```

```
Missing values:
Adj Close    0
dtype: int64
```

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

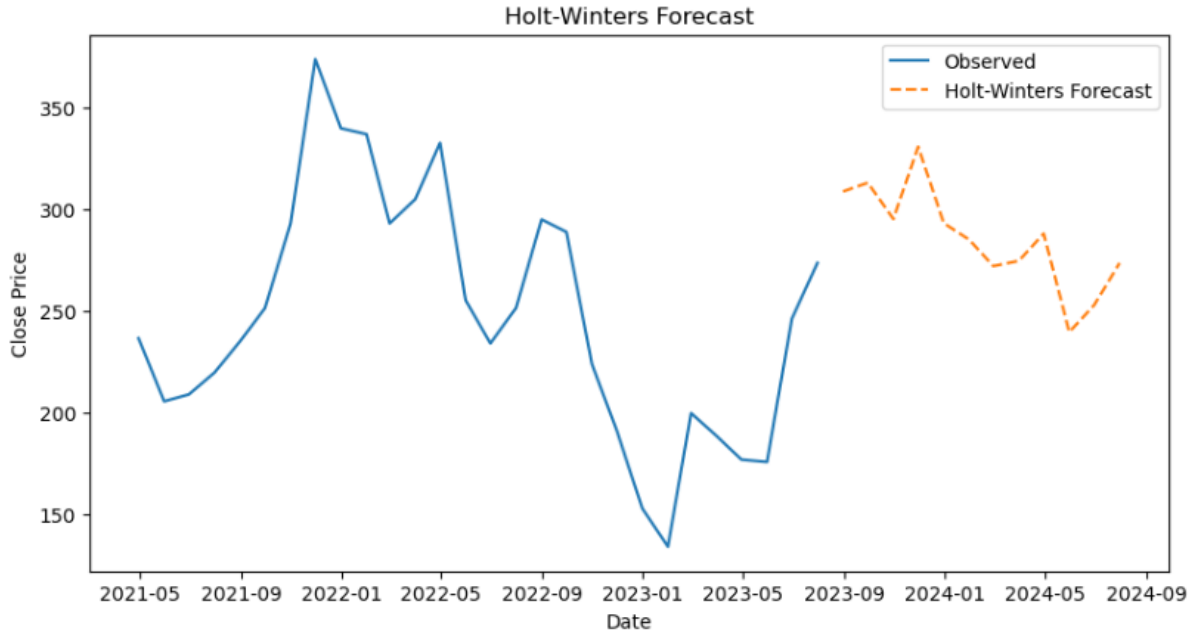


In [15]:
```
from statsmodels.tsa.seasonal import seasonal_decompose

# 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()
```

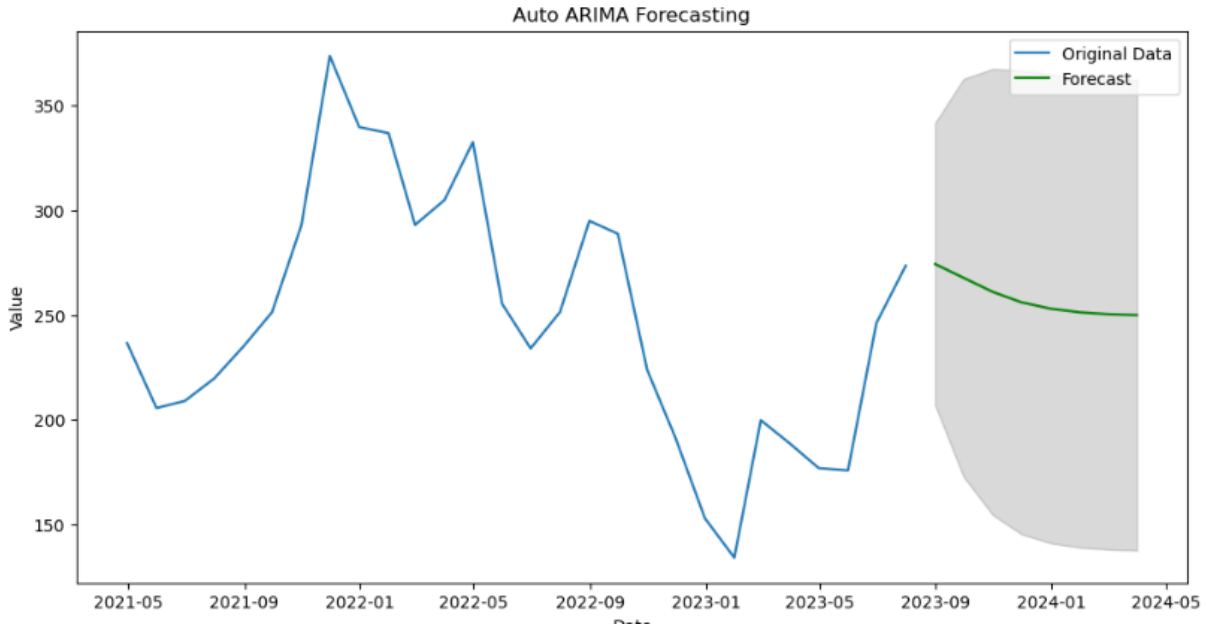


```
]: from statsmodels.tsa.holtwinters import ExponentialSmoothing

# Fit the Holt-Winters model
holt_winters_model = ExponentialSmoothing(train_data, seasonal='mul', seasonal_periods=12).fit()

# Forecast for the next year (12 months)
holt_winters_forecast = holt_winters_model.forecast(12)
```

```
]: # 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()
```



```
]: # Number of periods to forecast
n_periods = 8

# Generate forecast
forecast, conf_int = arima_model.predict(n_periods=n_periods, return_conf_int=True)

# 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()
```

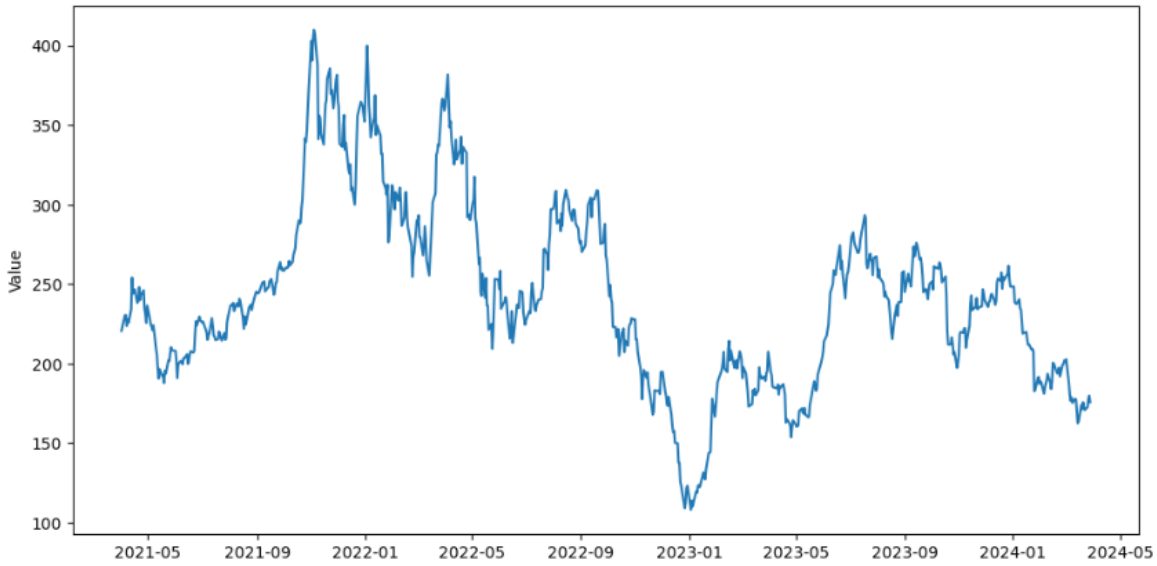


In [36]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Compute RMSE
rmse = np.sqrt(mean_squared_error(test_data, forecast))
print(f'RMSE: {rmse}')

# Compute MAE
mae = mean_absolute_error(test_data, forecast)
print(f'MAE: {mae}')

# Compute MAPE
mape = np.mean(np.abs((test_data - forecast) / forecast)) * 100
print(f'MAPE: {mape}')
# Compute R-squared
r2 = r2_score(test_data, forecast)
print(f'R-squared: {r2}')
```

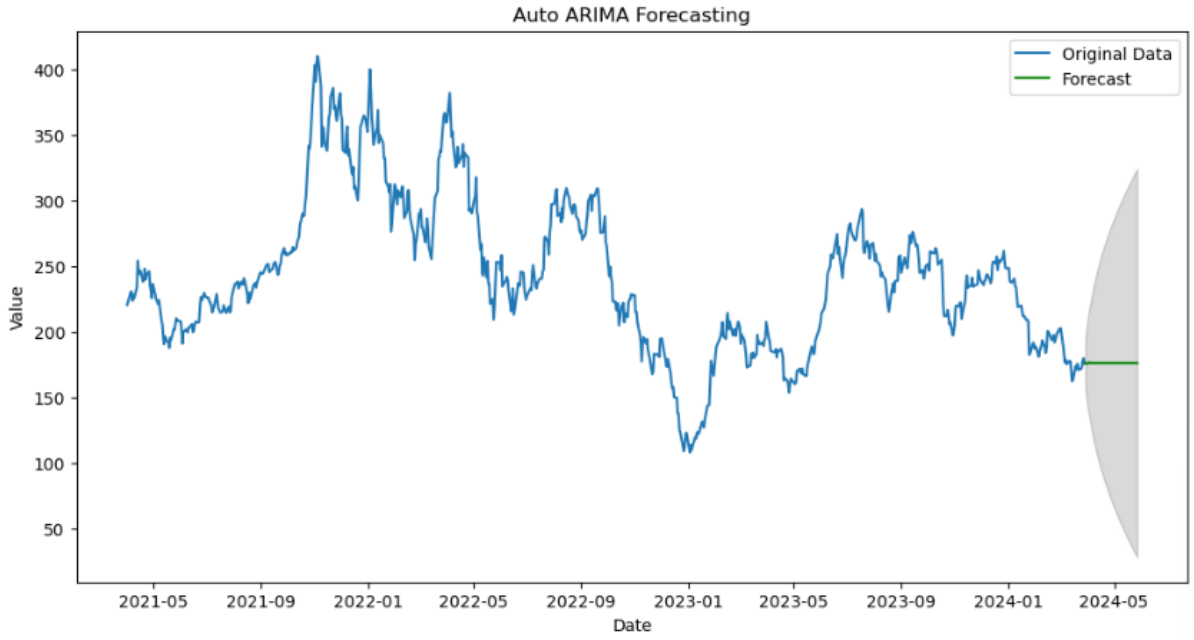
```
RMSE: 39.441975170228275
MAE: 33.205642834185994
MAPE: nan
R-squared: -1.2734694906453514
```

In [37]:
```
daily_data= df.copy()
```

In [38]:
```
# Plot the original data, fitted values, and forecast
plt.figure(figsize=(12, 6))
plt.plot(daily_data['Adj Close'])
plt.xlabel('Date')
plt.ylabel('Value')
plt.show()
```



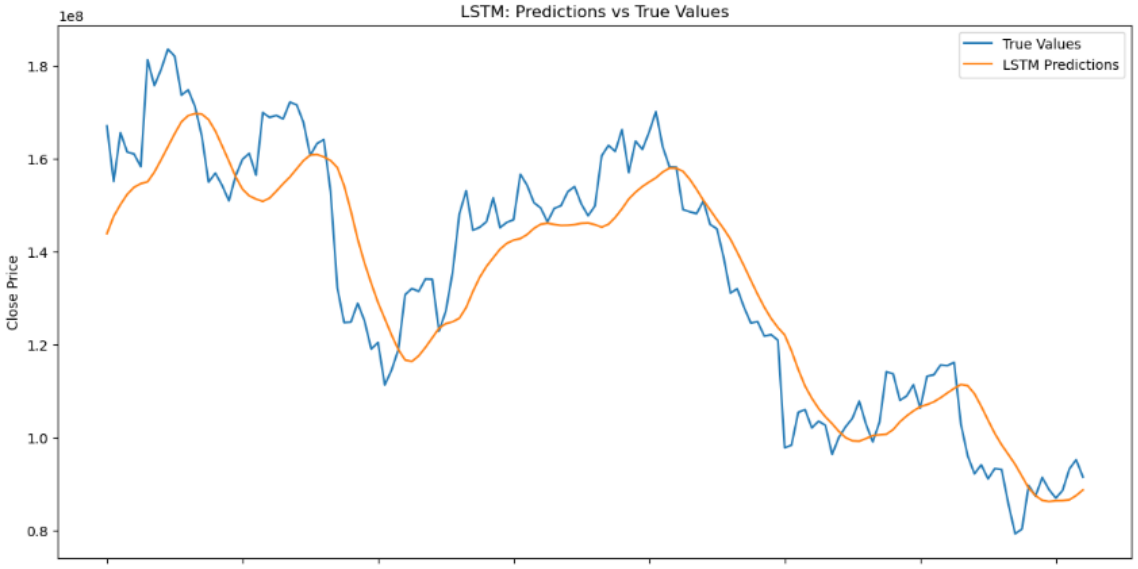
```
# Plot the original data, fitted values, and forecast
plt.figure(figsize=(12, 6))
plt.plot(daily_data['Adj Close'], label='Original Data')
plt.plot(forecast_df, label='Forecast', color='green')
plt.fill_between(future_dates,
                 conf_int_df['lower_bound'],
                 conf_int_df['upper_bound'],
                 color='k', alpha=.15)
plt.legend()
plt.xlabel('Date')
plt.ylabel('Value')
plt.title('Auto ARIMA Forecasting')
plt.show()
```



In [72]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Compute RMSE
rmse = np.sqrt(mean_squared_error(y_test_scaled, y_pred_scaled))
print(f'RMSE: {rmse}')

# Compute MAE
mae = mean_absolute_error(y_test_scaled, y_pred_scaled)
print(f'MAE: {mae}')

# Compute MAPE
mape = np.mean(np.abs((y_test_scaled - y_pred_scaled) / y_pred_scaled)) * 100
print(f'MAPE: {mape}')
# Compute R-squared
r2 = r2_score(y_test_scaled, y_pred_scaled)
print(f'R-squared: {r2}')
```

```
RMSE: 11022585.01420121
MAE: 8930048.769865293
MAPE: 6.763641678865481
R-squared: 0.8437400619975493
```

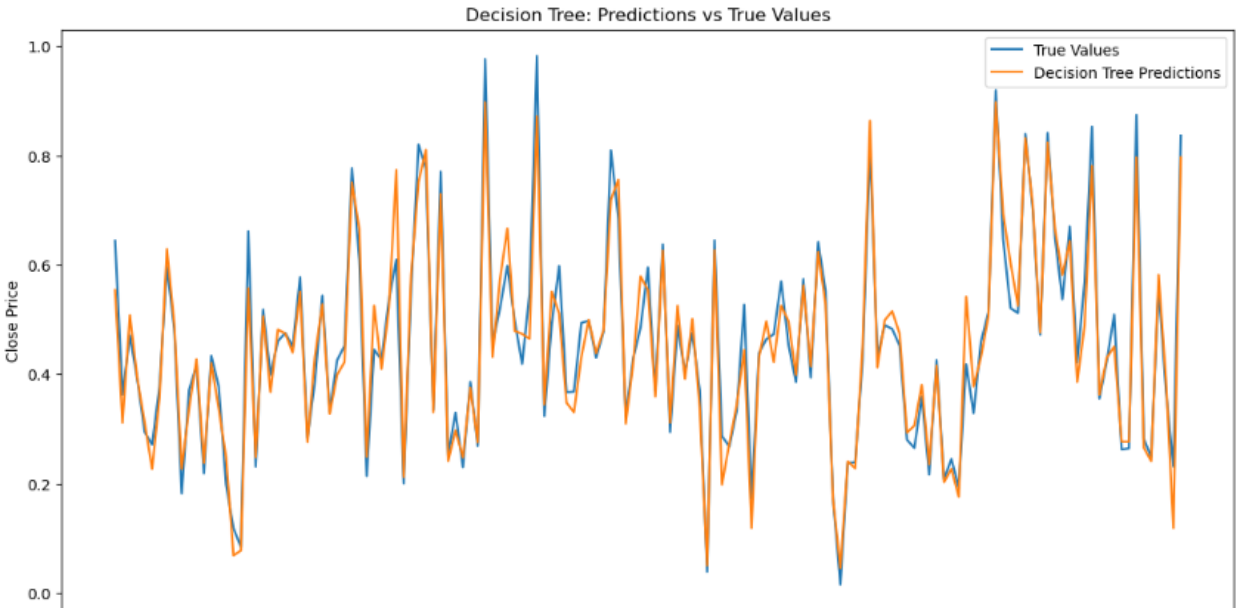
In [73]:
```
# Plot the predictions vs true values
plt.figure(figsize=(14, 7))
plt.plot(y_test_scaled, label='True Values')
plt.plot(y_pred_scaled, label='LSTM Predictions')
plt.title('LSTM: Predictions vs True Values')
plt.xlabel('Time')
plt.ylabel('Close Price')
plt.legend()
plt.show()
```



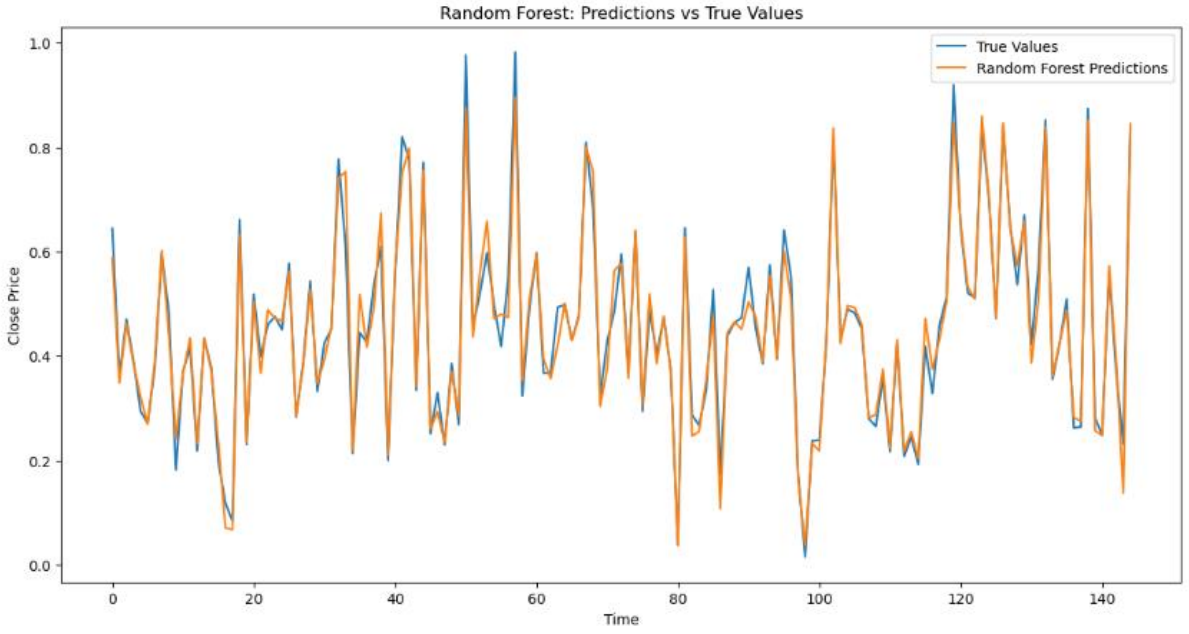
```
# Print some predictions and true values for both models
print("\nDecision Tree Predictions vs True Values:")
for i in range(10):
    print(f"Prediction: {y_pred_dt[i]}, True Value: {y_test[i]}")
```

```
Decision Tree Predictions vs True Values:
Prediction: 0.5539801456805665, True Value: 0.644350167647268
Prediction: 0.31119354012420286, True Value: 0.3627720579714687
Prediction: 0.5079670141852853, True Value: 0.47079869299199917
Prediction: 0.3924426610707159, True Value: 0.39047711958830034
Prediction: 0.31119354012420286, True Value: 0.2941332365179259
Prediction: 0.22672009205719812, True Value: 0.2712425763024745
Prediction: 0.3565662847083172, True Value: 0.3769945210246848
Prediction: 0.6290787418360968, True Value: 0.5962500289068802
Prediction: 0.49170172244567745, True Value: 0.4828899702413936
Prediction: 0.22672009205719812, True Value: 0.1818332610713983
```

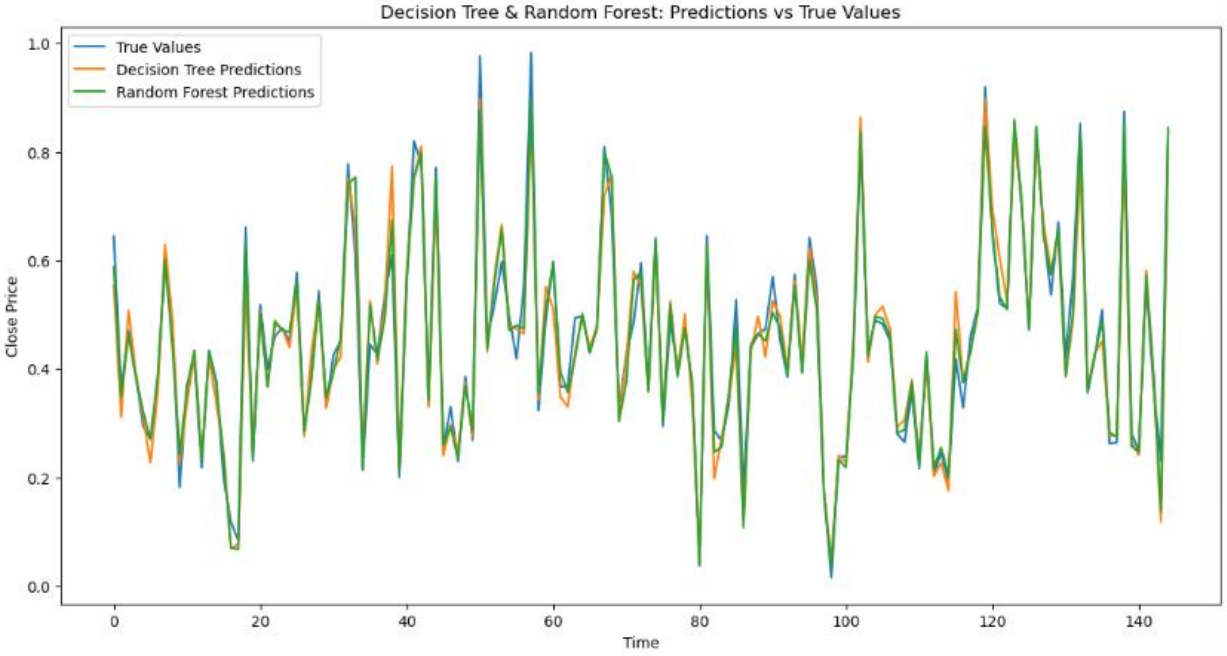
```
# Plot the predictions vs true values for Decision Tree
plt.figure(figsize=(14, 7))
plt.plot(y_test, label='True Values')
plt.plot(y_pred_dt, label='Decision Tree Predictions')
plt.title('Decision Tree: Predictions vs True Values')
plt.xlabel('Time')
plt.ylabel('Close Price')
plt.legend()
plt.show()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



```
4]: # 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()
```



## Model Evaluation

- Performance metrics such as RMSE, MAE, MAPE, and R-squared were used to evaluate the models.
- The SARIMA model showed better performance compared to the basic ARIMA model.
- The Random Forest model outperformed the Decision Tree model in terms of accuracy and generalization.

## 6. Interpretation

1. **Holt-Winters Model**: Provided good seasonal forecasts but had higher error metrics compared to more sophisticated models like SARIMA and LSTM.
2. **ARIMA and SARIMA Models**: SARIMA outperformed ARIMA due to its ability to capture seasonal patterns, resulting in lower error metrics and higher R-squared values.
3. **LSTM Model**: The deep learning approach of LSTM achieved the best performance with the lowest RMSE and MAE, indicating its capability to handle complex time series data.
4. **Decision Tree and Random Forest**: Both models performed well, with Random Forest outperforming Decision Tree, showcasing the power of ensemble methods in reducing overfitting and improving accuracy.

These results highlight the effectiveness of various forecasting models in predicting stock prices, with LSTM and SARIMA models standing out as the top performers. The insights gained from these models can be leveraged for making informed investment decisions, optimizing portfolio management, and enhancing overall financial strategies.

## 7. Recommendations

- For short-term forecasting, the SARIMA model is recommended due to its ability to handle seasonality.
- For long-term forecasting, the Holt-Winters model provides a good balance between trend and seasonality.
- Multivariate models like LSTM and Random Forest can be explored further with more features to improve accuracy.
- Regular updates and retraining of the models with new data can enhance their predictive power.

## 8. Conclusion

This project demonstrated the application of various time series forecasting techniques on TSLA stock prices. Both univariate and multivariate models provided valuable insights into the future trends of the stock. The results highlight the importance of choosing the right model based on the specific requirements and characteristics of the data. By leveraging these forecasting models, stakeholders can make more informed investment decisions and optimise their strategies.

## 9. Reference


- https://finance.yahoo.com/
- https://pypi.org/project/yfinance/