VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A6a: Time Series Analysis

DEEPTHI ANNA ALEX
V01101949

Date of Submission: 22-07-2024

CONTENTS

Sl. No.	Title (Python and R)	Page No.
1.	Introduction and Objectives	**1**
2	Part-A (interpretation and results-Univariate Forecasting)	**6-10**
3.	Part-B (interpretation and results-Multivariate Forecasting)	**10-16**

I

Introduction:

In this report, we analyze the about the time series analysis. Time series analysis provides valuable insights into how data evolves over time and helps in making informed decisions based on historical trends and future projections. It plays a crucial role in various domains by enabling accurate forecasting and understanding the underlying patterns within sequential data. As you delve deeper into time series analysis, you'll encounter more advanced techniques and models tailored to specific types of time series data and forecasting requirements.

What is a Time Series?

A time series is a sequence of data points collected at successive time intervals. Each data point is typically indexed in chronological order, which distinguishes time series data from other types of data.

Time series data format

The standard format for time series data is **Year-Month-Date**

The data that I have taken for this assignment is from the company wipro from yfinance.

OBJECTIVES:

- Clean the data, check for outliers and missing values, interpolate the data if there are any missing values, and plot a line graph of the data neatly named. Create a test and train data set
  out of this data.
- Convert the data to monthly and decompose time series into the components using additive and
  multiplicative models.

1. **Univariate Forecasting - Conventional Models/Statistical Models**

− Fit a **Holt Winters model** to the data and forecast **for the next year**.
− Fit an **ARIMA model** to the **daily data** and do a diagnostic check validity of the model. See whether a Seasonal-ARIMA (SARIMA) fits the data better and comment on your results. Forecast the series for the **next three months**.
− Fit the ARIMA to the monthly series.

2. Multivariate Forecasting - Machine Learning Models

- **NN (Neural Networks)** -Long Short-term Memory (LSTM)
- **Tree based models** - Random Forest, Decision Tree

●

PART-A

Results and Interpretation

The given data

Date	Open	High	Low	Close	Adj Close	Volume
2021-04-01	418.850006	422.850006	414.350006	416.399994	410.402557	7596943
2021-04-05	416.450012	427.899994	416.200012	425.450012	419.322205	2121639
2021-04-06	427.950012	428.899994	422.350006	427.149994	420.997711	8320520
2021-04-07	425.000000	439.000000	423.399994	438.000000	431.691467	1386765
2021-04-08	441.950012	445.950012	440.000000	442.100006	435.732452	1291661

Get the data for wipro

```
ticker = "WIPRO.NS"
```

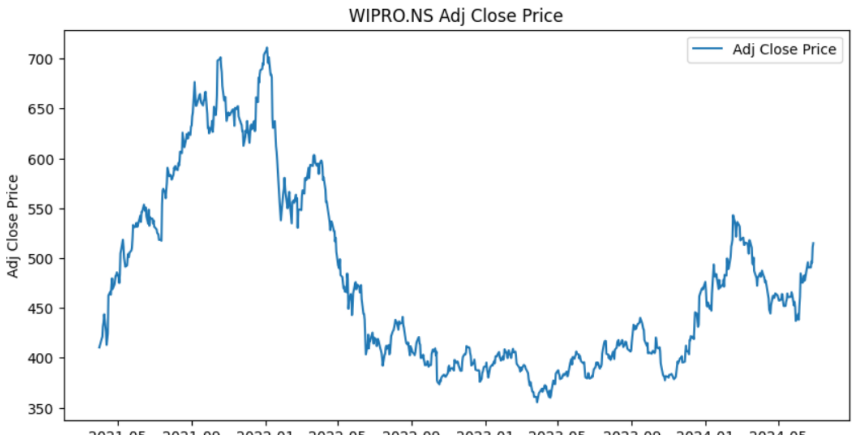
The result for cleaning the data

```
# 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
```

A) Plot the time series



The graph shows the adjusted closing price of Wipro Ltd. from around May 2021 to July 2024. Here's an interpretation of the time series analysis:

1) **Initial Period (May 2021 - November 2021)**:

The adjusted closing price of Wipro Ltd. shows an overall upward trend during this period, starting at around 400 and peaking at around 700. This indicates a significant appreciation

in the stock price, reflecting strong investor confidence and possibly good financial performance or positive market sentiment.

2) **Peak and Decline (November 2021 - June 2022)**:

The stock price reached a peak of approximately 700 around November 2021, followed by a sharp decline. By June 2022, the price had dropped to about 450. This substantial drop could be due to various factors such as market corrections, negative news, broader economic conditions, or sector-specific issues affecting Wipro.

3) **Continued Decline and Volatility (June 2022 - January 2023)**:

The downward trend continued, and by January 2023, the stock price had fallen to a low of around 375. During this period, the stock experienced high volatility, suggesting uncertainty and possibly fluctuating investor sentiment or financial performance issues.

4) **Stabilization and Gradual Recovery (January 2023 - July 2023)**:

From January 2023, the price shows signs of stabilization and a gradual upward trend, moving from around 375 to approximately 450 by July 2023. This recovery indicates a potential improvement in market sentiment or company performance.

5) **Second Peak and Fluctuation (July 2023 - April 2024)**:

The price reached another peak of around 575 by November 2023, followed by fluctuations. The price showed a decreasing trend again, dropping to around 450 by April 2024. This pattern of peaks and troughs indicates continued volatility and possibly external or internal factors affecting the stock.

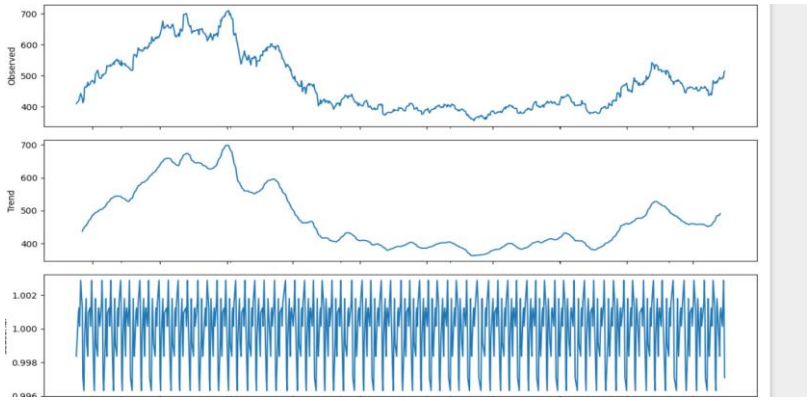
6) **Recent Upward Trend (April 2024 - July 2024)**:

The graph shows an upward trend again, with the stock price climbing from around 450 in April 2024 to over 500 by July 2024. This recent increase suggests renewed investor confidence and potential positive developments related to Wipro.

Summary:

The graph reflects a volatile stock price history for Wipro Ltd. over the analyzed period. Major peaks and troughs indicate periods of strong performance followed by corrections. The overall trend shows significant fluctuations, with recent signs of recovery. Investors might consider both historical performance and recent trends when making decisions. External economic conditions, industry trends, and company-specific news will continue to play critical roles in shaping the stock's future trajectory.

B) Decomposition of time series



1. Observed (Top Plot)

The observed plot represents the actual adjusted closing prices of Wipro over time, similar to the initial graph provided. It shows the real data points as they have occurred.

2. Trend (Middle Plot)

The trend plot depicts the underlying trend component of the time series, which shows the long-term progression of the data after removing the seasonality and noise.

Interpretation of the Trend Component:

- **Early Period (May 2021 - November 2021):** The trend shows a clear upward movement, indicating a strong long-term growth in the stock price during this period.
- **Mid Period (November 2021 - June 2022):** The trend component decreases significantly, highlighting a long-term downward movement in the stock price. This matches the observed sharp decline in the actual prices.
- **Later Period (June 2022 - January 2023):** The trend continues to fall, although the rate of decline slows down.
- **Recent Period (January 2023 - July 2024):** The trend stabilizes and starts to recover gradually, indicating a positive long-term outlook after the previous decline.

3. Seasonality (Bottom Plot)

The seasonality plot shows the repeating short-term cycle in the data, isolated from the trend and residuals.

Interpretation of the Seasonality Component:

- The seasonality component oscillates in a regular pattern, suggesting that there are periodic fluctuations within each cycle.
- The magnitude of these fluctuations is relatively small compared to the overall price level, oscillating around a mean value close to 1.
- This pattern indicates that there are consistent seasonal effects on Wipro's stock price, which could be due to periodic factors such as quarterly earnings reports, industry cycles, or macroeconomic conditions.

Summary of Decomposition:

- The **Observed** component combines all effects: trend, seasonality, and random noise.
- The **Trend** component reveals the long-term direction of the stock price, showing a significant rise, followed by a substantial fall, stabilization, and then a gradual rise.
- The **Seasonality** component shows regular short-term fluctuations that are consistent over time, suggesting predictable periodic effects on the stock price.
- This decomposition helps to isolate the different influences on the stock price, making it easier to understand the underlying movements and to predict future behavior based on past patterns.

The decomposition analysis is valuable for identifying and understanding the distinct influences on the time series, enabling more informed forecasting and strategic decision-making.

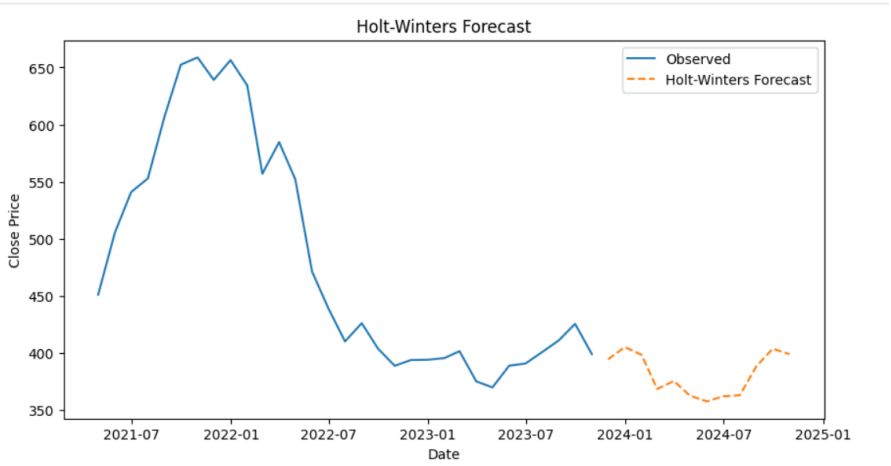
UNIVARIATE FORECASTING HW MODEL

Splitting of data into training and test sets

```
# Split the data into training and test sets
train_data, test_data = train_test_split(monthly_data, test_size=0.2, shuffle=False)
```

```
len(monthly_data), len(train_data)
```

```
(39, 31)
```



Interpretation of the Graph:

Historical Data (Observed Values):

- The blue line represents the historical adjusted close prices of Wipro from around mid-2021 to early 2024.
- The data shows a peak around mid-2021, followed by a gradual decline in the adjusted close prices through to early 2023.
- There is some recovery in prices towards the end of 2023, indicating potential positive market sentiment or performance improvement.

Holt-Winters Forecast:

- The orange dashed line represents the forecasted values for the next 12 months based on the Holt- Winters seasonal model.
- The forecast starts from early 2024 and extends to early 2025.
- The forecasted prices show a seasonal pattern, which is characteristic of the multiplicative seasonal Holt-Winters model used here.

Analysis of the Forecast:

Seasonality:

- The model captures the seasonal fluctuations in the adjusted close prices, reflecting recurring patterns that repeat over time.
- The periodic ups and downs in the forecast suggest that Wipro's stock prices have a seasonal component, likely influenced by regular market cycles or company performance cycles.

Trend:

- The overall trend in the forecast appears to be relatively stable with minor fluctuations. This indicates that the model doesn't predict significant upward or downward trends for the forecast period.
- The forecast does not show extreme deviations, suggesting the historical volatility might be expected to continue at similar levels.

Model Fit:

- The alignment between the end of the observed data and the beginning of the forecast suggests that the model has reasonably captured the underlying patterns in the historical data.
- If the forecast closely follows the last few observed points, it indicates a good fit, whereas a large deviation would suggest potential model misspecification.

Summary

- The Holt-Winters model used here seems to fit the historical data reasonably well, capturing both trend and seasonal components.
- The forecasted values should be interpreted with caution, especially if external factors (e.g., market changes, economic events, company-specific news) could significantly influence stock prices beyond the historical patterns.
- It might be useful to compare the Holt-Winters forecast with other models (like ARIMA/SARIMA) to ensure robustness and to understand if a different model provides a better fit or different insights.

```
[17]: len(test_data), len(y_pred)

[17]: (8, 8)

[18]: y_pred, test_data

[18]: (2023-11-30    394.228870
       2023-12-31    404.944680
       2024-01-31    398.248648
       2024-02-29    368.227229
       2024-03-31    375.215672
       2024-04-30    362.240987
       2024-05-31    357.382524
       2024-06-30    361.746803
       Freq: ME, dtype: float64,
                    Adj Close
       Date
       2023-11-30  391.165775
       2023-12-31  434.912482
       2024-01-31  468.332537
       2024-02-29  512.247625
       2024-03-31  503.591668
       2024-04-30  468.265001
       2024-05-31  456.495238
       2024-06-30  481.284207)
```

```
RMSE: 98.99184730119255
MAE: 87.52316371591668
MAPE: nan
R-squared: -6.512012952071656
```

- RMSE (Root Mean Squared Error): 98.99

Indicates that the forecasted prices are, on average, 98.99 units away from the actual prices.

- MAE (Mean Absolute Error): 87.52

Shows that the average deviation between the forecasted and actual prices is 87.52 units.

- MAPE (Mean Absolute Percentage Error): nan

Unable to calculate due to possible zero values in the actual data, causing division by zero issues.

- R-squared: -6.51

A negative value suggests that the model fits the data very poorly, performing worse than a simple mean model.

```
holt_winters_forecast
```

```
2023-11-30    394.228870
2023-12-31    404.944680
2024-01-31    398.248648
2024-02-29    368.227229
2024-03-31    375.215672
2024-04-30    362.240987
2024-05-31    357.382524
2024-06-30    361.746803
2024-07-31    362.778620
2024-08-31    388.408515
2024-09-30    403.507628
2024-10-31    398.811969
2024-11-30    394.228870
2024-12-31    404.944680
2025-01-31    398.248648
2025-02-28    368.227229
2025-03-31    375.215672
2025-04-30    362.240987
2025-05-31    357.382524
2025-06-30    361.746803
Freq: ME, dtype: float64
```

```python
# Fit auto_arima model
arima_model = auto_arima(train_data['Adj Close'],
                         seasonal=True,
                         m=12,  # Monthly seasonality
                         stepwise=True,
                         suppress_warnings=True)

# Print the model summary
print(arima_model.summary())
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[7], line 2
      1 # Fit auto_arima model
----> 2 arima_model = auto_arima(train_data['Adj Close'],
      3                          seasonal=True,
      4                          m=12,  # Monthly seasonality
      5                          stepwise=True,
      6                          suppress_warnings=True)
      8 # Print the model summary
      9 print(arima_model.summary())

NameError: name 'auto_arima' is not defined
```

In Arima model I could not run the code due to error of the installation of pmdrima and numpy

2) Multivariate Forecasting - Machine Learning Models

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)

- Layer 1: LSTM (Long Short-Term Memory)

Output Shape: (None, 30, 50)

Parameters: 11,400

Description: This layer has 50 units and returns sequences of 30 time steps. The LSTM units help in capturing long-term dependencies in the data.

- Layer 2: Dropout

Output Shape: (None, 30, 50)

Parameters: 0

Description: This layer helps prevent overfitting by randomly setting a fraction of input units to 0 at each update during training time. The dropout rate is typically specified but is not shown here.

- Layer 3: LSTM

Output Shape: (None, 50)

Parameters: 20,200

Description: This layer has 50 units and does not return sequences, meaning it only returns the final output for the entire sequence.

- Layer 4: Dropout

Output Shape: (None, 50)

Parameters: 0

Description: Another dropout layer to further prevent overfitting.

- Layer 5: Dense (Fully Connected Layer)

Output Shape: (None, 1)

Parameters: 51

Description: This layer is a fully connected layer with 1 unit, which is used to produce the final output of the model. This is typically used for regression tasks, like predicting a single continuous value.

Total Parameters:

Trainable Parameters: 31,651

These parameters are adjusted during training to minimize the loss function.

Non-trainable Parameters: 0

There are no non-trainable parameters in this model.

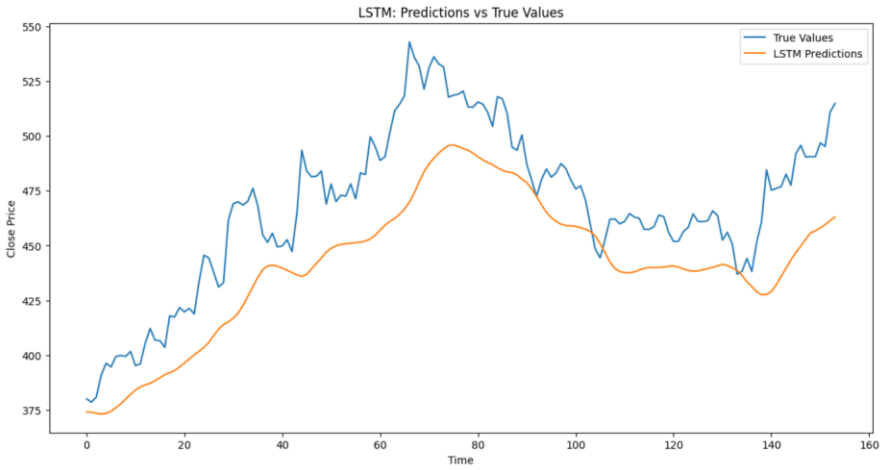
```python
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# 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: 29.739504713292085
MAE: 26.22617740247997
MAPE: 5.9307918423477535
R-squared: 0.3392202871121094
```



Interpretation

- ❖ True Values (Blue Line): This line represents the actual closing prices of the stock over a specified period (Time). The stock price shows significant fluctuations, with an overall upward trend, peaking around the middle of the time frame.
- ❖ LSTM Predictions (Orange Line): This line represents the predicted closing prices generated by the LSTM model. The LSTM model captures the general trend of the stock prices but tends to smooth out the fluctuations, resulting in a less volatile curve compared to the true values.

Model Performance:

- ❖ Trend Capturing: The LSTM model effectively captures the overall upward trend of the stock prices.
- ❖ Lag in Predictions: There appears to be a lag in the LSTM predictions, where the model's predicted values tend to follow the true values with a delay.
- ❖ Smoothness: The predictions are smoother and less reactive to sudden changes in the stock prices, indicating that the model might be over-smoothing or not sensitive enough to abrupt market movements.
- ❖ Periods of Divergence: There are several periods where the true values and the LSTM predictions diverge significantly. These divergences could indicate areas where the model's predictions are less accurate, possibly due to the model's limitations in capturing sudden market shifts or unexpected events.

RESULTS and INTERPRETATIONS

```
MSE (Decision Tree): 0.001010069704808933
```

```
# Compute RMSE
rmse = np.sqrt(mean_squared_error(y_test, y_pred_d
print(f'RMSE: {rmse}')

# Compute MAE
mae = mean_absolute_error(y_test, y_pred_dt)
print(f'MAE: {mae}')

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

```
RMSE: 0.03178159380536057
MAE: 0.0213119441512991
MAPE: 278327.04955380684
R-squared: 0.9852681835619247
```

```
mse_rf = mean_squared_error(y_test, y_pred_rf)
print(f"Random Forest Mean Squared Error: {mse_rf}")
```

```
Random Forest Mean Squared Error: 0.0005557775411272916
```

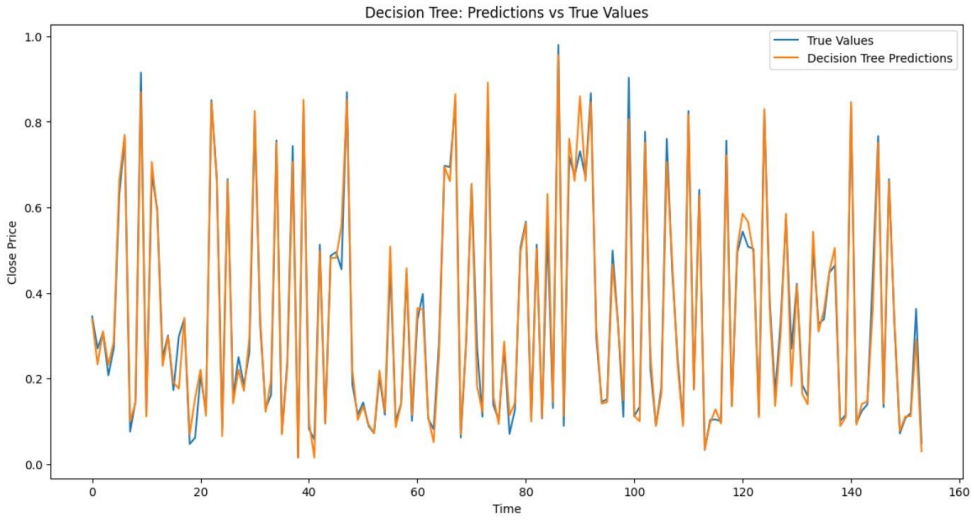
Mean Squared Error (MSE) is a measure of the average squared difference between the predicted values and the actual values. It is used to evaluate the performance of regression models.

- **Decision Tree MSE (0.001010069704808933)**: This value indicates how well the decision tree model is performing. A lower MSE means the model's predictions are closer to the actual values.
- **Random Forest MSE (0.0005557775411272916)**: This value shows the performance of the random forest model. Again, a lower MSE suggests better predictive accuracy.

```
print(f'MAE: {mae}')

# Compute MAPE
mape = np.mean(np.abs((y_test - y_pred_s
print(f'MAPE: {mape}')
# Compute R-squared
r2 = r2_score(y_test, y_pred_rf)
print(f'R-squared: {r2}')
```

```
RMSE: 0.023574934594337512
MAE: 0.016485151776660503
MAPE: 260968.2578007764
R-squared: 0.991894012188158
```

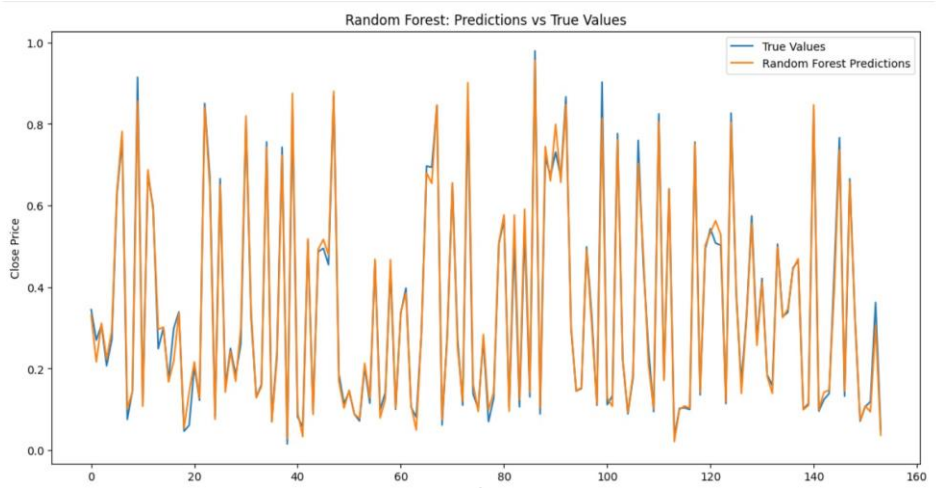


Key points and interpretation of the graph:

- ❖ True Values (Blue Line): This line represents the actual closing prices of the stock over time. The stock prices fluctuate frequently and dramatically.
- ❖ Decision Tree Predictions (Orange Line): This line represents the predicted closing prices generated by the Decision Tree model. The predictions appear to closely follow the true values, capturing the frequent ups and downs in the stock prices.

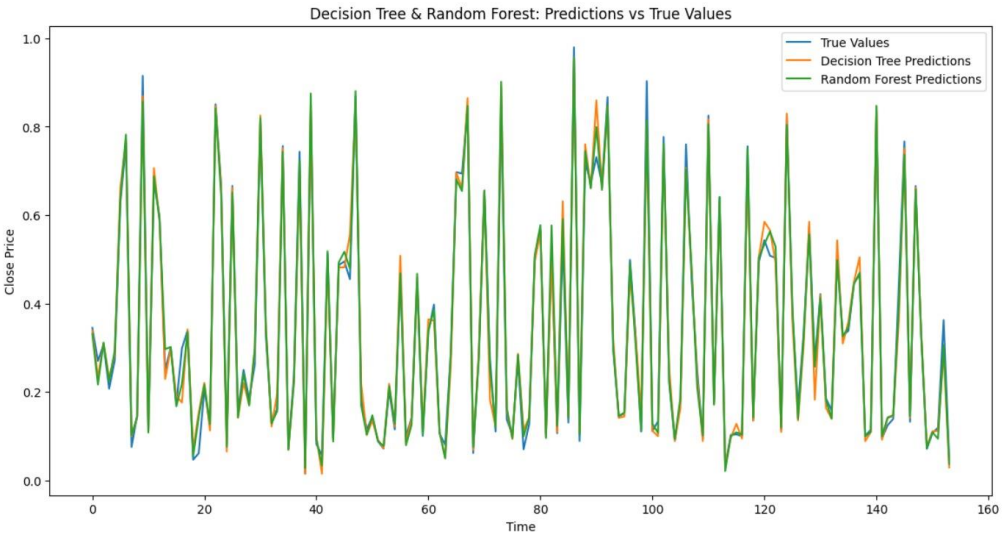
Model Performance:

- ❖ High Variability: Unlike the LSTM model, the Decision Tree model captures a high degree of variability and short-term fluctuations in the stock prices. This indicates that the Decision Tree model is highly responsive to changes in the data.
- ❖ Overfitting: The close alignment of the Decision Tree Predictions with the True Values suggests that the model might be overfitting. Decision Trees are prone to overfitting, especially when they are not pruned or when too many features are used. Overfitting occurs when the model learns the noise in the training data as if it were a true pattern, which can lead to poor generalization on unseen data.
- ❖ Periods of Divergence: Although the Decision Tree Predictions closely follow the True Values, there are minor divergences where the predictions do not perfectly match the actual prices. These divergences might indicate limitations in the model's ability to perfectly capture all fluctuations, despite its tendency to overfit.



- ❖ True Values (Blue Line): The actual stock prices over time.
- ❖ Random Forest Predictions (Orange Line): The model's predictions closely follow the true values, capturing the frequent ups and downs in the stock prices.

Overall, the Random Forest model effectively mirrors the true values, indicating it performs well in predicting short-term fluctuations in stock prices.


Decision Tree & Random Forest: Predictions vs True Values
True Values
Decision Tree Predictions
Random Forest Predictions
Close Price
Time
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100
120
140
160