VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A6a: Time Series Analysis

JESIN KANDATHY JOY

V01110163

Date of Submission: 22-07-2024

CONTENTS

Sl. No.	Title	Page No.
1.	Introduction	**1**
2.	Objective	**1**
3.	Business Significance	**1**
4.	R code results	**2**
5.	Python code results	**7**
6.	Interpretations	**21**
7.	Recommendations	**22**

Introduction:

The stock market analysis focuses on predicting future stock prices of Apple Inc. (AAPL) using various time series forecasting techniques. This analysis utilizes historical data of the Adjusted Close price from January 1, 2015, to December 31, 2023, to develop and evaluate several forecasting models, including Holt-Winters, ARIMA, LSTM, Random Forest, and Decision Tree models. Each method offers unique insights into stock price movements and their future trends.

The data includes the Adjusted Close price of AAPL from Yahoo Finance and the techniques used here are Holt-Winters forecasting, ARIMA, LSTM, Random Forest, and Decision Tree models. Some of the analysis components are Time series decomposition (trend, seasonal, and residual), model performance metrics (RMSE, MAE, MAPE, R-squared), and comparison of predictive accuracy.

Objectives:

- Data Preparation
    - Clean the Data: Handle outliers and missing values; interpolate if necessary.
    - Plot the Data: Create a labeled line graph.
    - Split the Data: Create training and testing datasets.
- Time Series Analysis
    - Convert to Monthly: Aggregate data monthly.
    - Decompose Time Series: Use additive and multiplicative models to extract trend, seasonal, and residual components.
- Univariate Forecasting
    - Holt-Winters Model: Fit and forecast for one year.
    - ARIMA Model: Fit and validate; compare with SARIMA. Forecast for three months and fit to monthly data.
- Multivariate Forecasting
    - LSTM Model: Train and evaluate.
    - Tree-Based Models: Fit and assess Random Forest and Decision Tree.

Business Significance:

- Investment Decision Making: Accurate forecasting of stock prices aids in making informed investment decisions, helping investors optimize their portfolios and maximize returns.

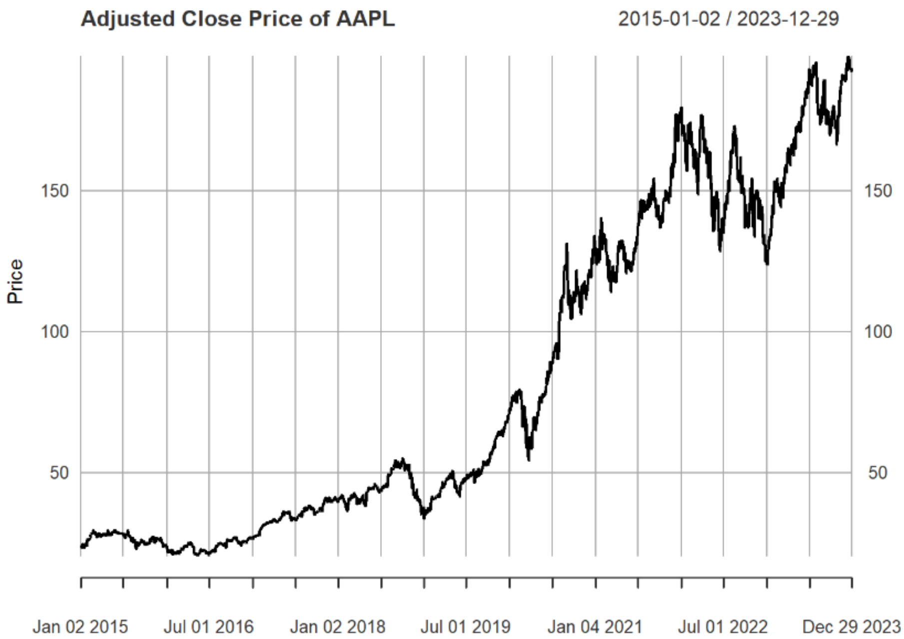
- Model Selection: Understanding the strengths and weaknesses of different forecasting models allows investors to choose the best-suited approach for predicting stock price movements.
- Risk Management: By evaluating and comparing forecasting methods, investors can better assess the risk associated with stock investments and adjust their strategies accordingly.
- Strategic Planning: Reliable forecasts help businesses and financial analysts in strategic planning and market analysis, contributing to better financial management and strategic decision-making.

R code results:

```
library(rpart)

# Load stock data
stock_data <- getSymbols("AAPL", src = "yahoo", from = "2015-01-01", to = "2023-12-31", 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))
```

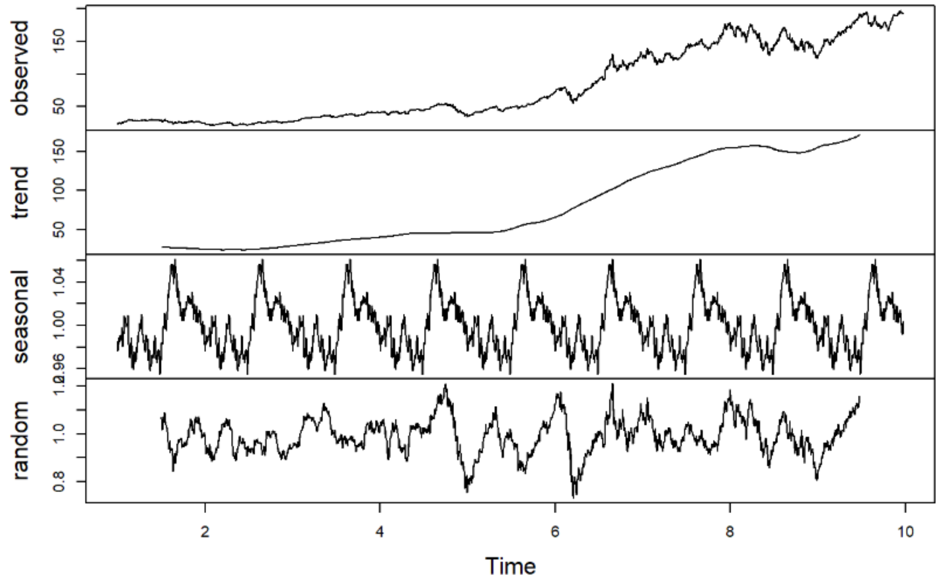
```
## [1] "Missing values: 0"
```

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



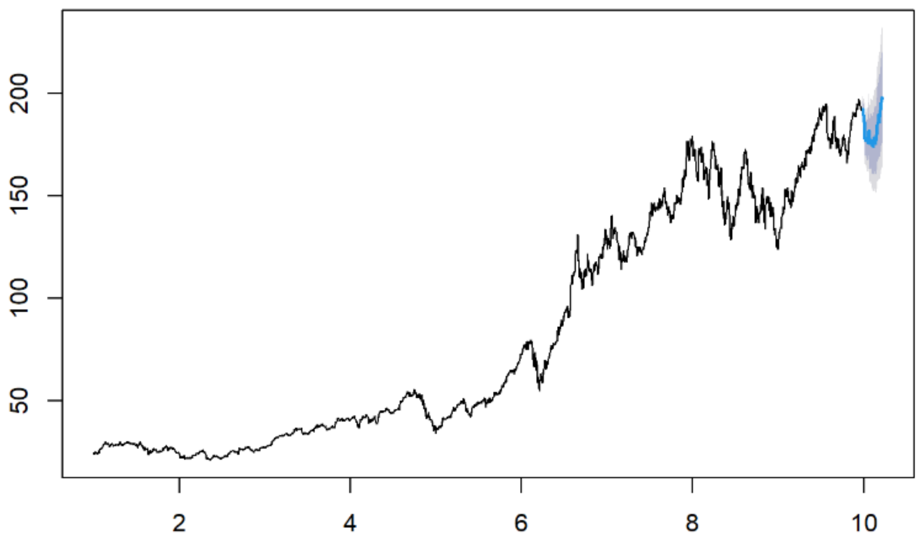
```r
# Decompose the time series
adj_close_ts <- ts(adj_close, frequency = 252)
decomposed <- decompose(adj_close_ts, type = "multiplicative")

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

Decomposition of multiplicative time series

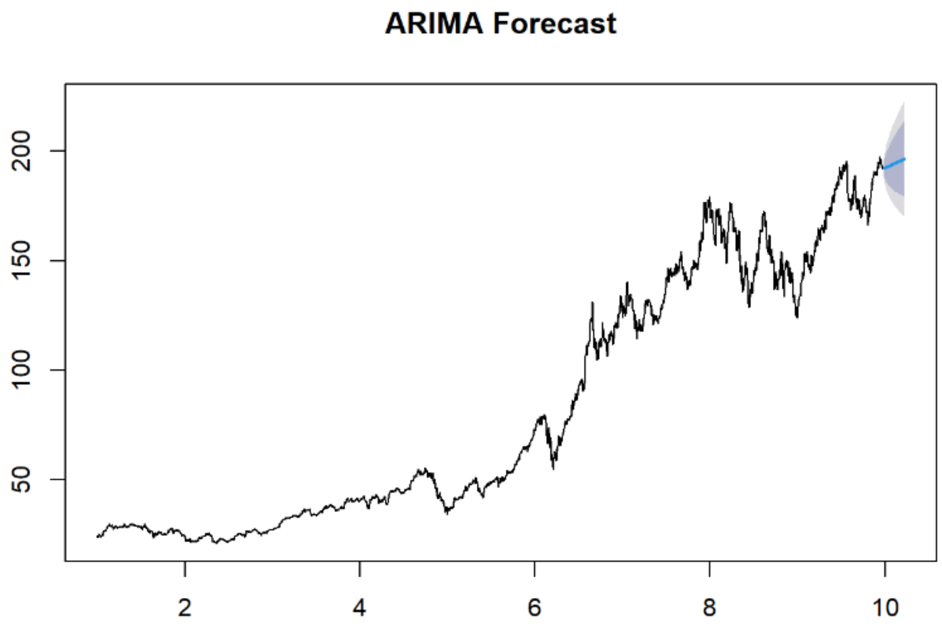



```r
# Holt-Winters Forecasting
hw_model <- HoltWinters(adj_close_ts, seasonal = "multiplicative")
hw_forecast <- forecast(hw_model, h = 60)

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

Holt-Winters Forecast



```r
# 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")
```




```r
# 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")
```

```r
# 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(test_data))^2)

print(paste("ARIMA RMSE:", arima_rmse))
```

```
## [1] "ARIMA RMSE: 15.8795725095204"
```

```r
print(paste("ARIMA MAE:", arima_mae))
```

```
## [1] "ARIMA MAE: 12.9243248418815"
```

```r
print(paste("ARIMA MAPE:", arima_mape))
```

```
## [1] "ARIMA MAPE: 8.56743946182602"
```

```r
print(paste("ARIMA R-squared:", arima_r2))
```

```
## [1] "ARIMA R-squared: 0.272438391156523"
```

```r
# Preparing data for LSTM, Random Forest, and Decision Tree
adj_close_df <- data.frame(Date = index(adj_close), Adj_Close = as.numeric(adj_close))
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, data = 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_Close - mean(test_data$Adj_Close))^2)

print(paste("Random Forest RMSE:", rf_rmse))
```

```
## [1] "Random Forest RMSE: 5.06053412596968"
```

```
## [1] "Random Forest RMSE: 5.06053412596968"
```

```r
print(paste("Random Forest MAE:", rf_mae))
```

```
## [1] "Random Forest MAE: 2.15014858456768"
```

```r
print(paste("Random Forest MAPE:", rf_mape))
```

```
## [1] "Random Forest MAPE: 1.13914372338237"
```

```r
print(paste("Random Forest R-squared:", rf_r2))
```

```
## [1] "Random Forest R-squared: 0.92611013433238"
```

```r
# Decision Tree model
library(rpart)
dt_model <- rpart(Adj_Close ~ Lag_1 + Lag_2 + Lag_3 + Lag_4 + Lag_5, data = train_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_Close - mean(test_data$Adj_Close))^2)

print(paste("Decision Tree RMSE:", dt_rmse))
```

```
## [1] "Decision Tree RMSE: 19.3767793005384"
```


```
## [1] "Decision Tree RMSE: 19.3767793005384"
```


```r
print(paste("Decision Tree MAE:", dt_mae))
```

```
## [1] "Decision Tree MAE: 15.9499643811337"
```

```r
print(paste("Decision Tree MAPE:", dt_mape))
```

```
## [1] "Decision Tree MAPE: 9.42932867703966"
```

```r
print(paste("Decision Tree R-squared:", dt_r2))
```

```
## [1] "Decision Tree R-squared: -0.0833164718969335"
```

Python code results:

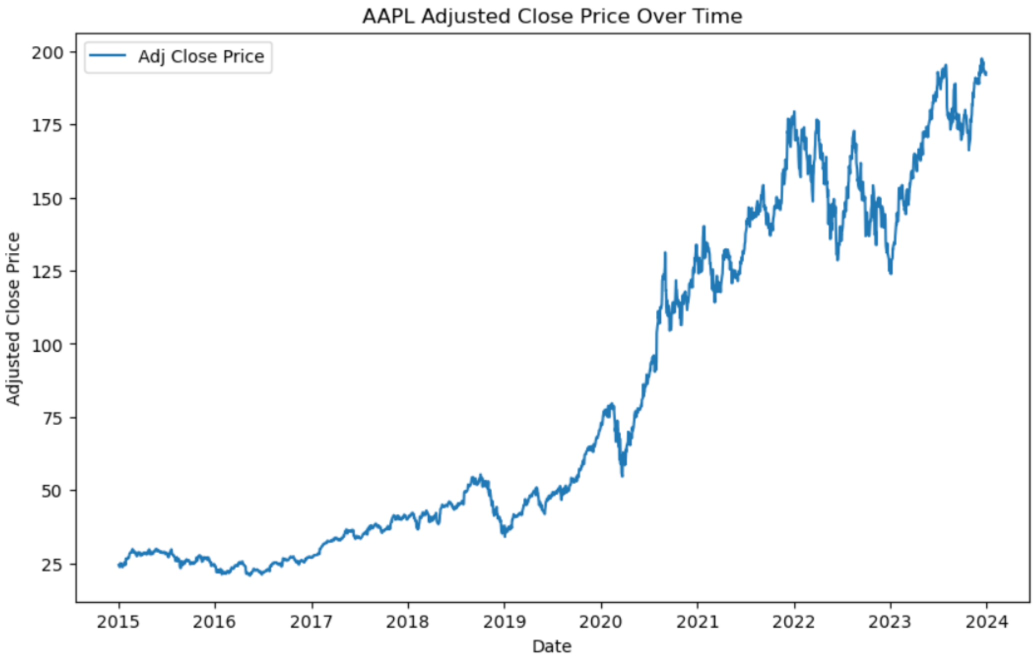
```
# Downloading data for Apple Inc.
data = yf.download('AAPL', start='2015-01-01', end='2023-12-31')

# Display the first few rows of the data
print(data.head())

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

```
[*********************100%%**********************]  1 of 1 completed
                 Open       High        Low      Close  Adj Close     Volume
Date
2015-01-02  27.847500  27.860001  26.837500  27.332500  24.402172  212818400
2015-01-05  27.072500  27.162500  26.352501  26.562500  23.714722  257142000
2015-01-06  26.635000  26.857500  26.157499  26.565001  23.716953  263188400
2015-01-07  26.799999  27.049999  26.674999  26.937500  24.049522  160423600
2015-01-08  27.307501  28.037500  27.174999  27.972500  24.973555  237458000
```

```
# Checking for missing values
missing_values = stock_data.isnull().sum()
print("Missing values:\n", missing_values)

# Checking for outliers
sns.boxplot(data=stock_data['Adj Close'])
plt.title('Boxplot of Adjusted Closing Prices')
plt.show()
```

```
C:\Users\user\anaconda3\Lib\site-packages\seaborn\categor
reated as labels (consistent with DataFrame behavior). To
  if np.isscalar(data[0]):
Missing values:
 Adj Close    0
dtype: int64
```

```python
# Plotting the line graph
plt.figure(figsize=(10, 6))
plt.plot(stock_data['Adj Close'], label='Adj Close Price')
plt.title('AAPL Adjusted Close Price Over Time')
plt.xlabel('Date')
plt.ylabel('Adjusted Close Price')
plt.legend()
plt.show()
```

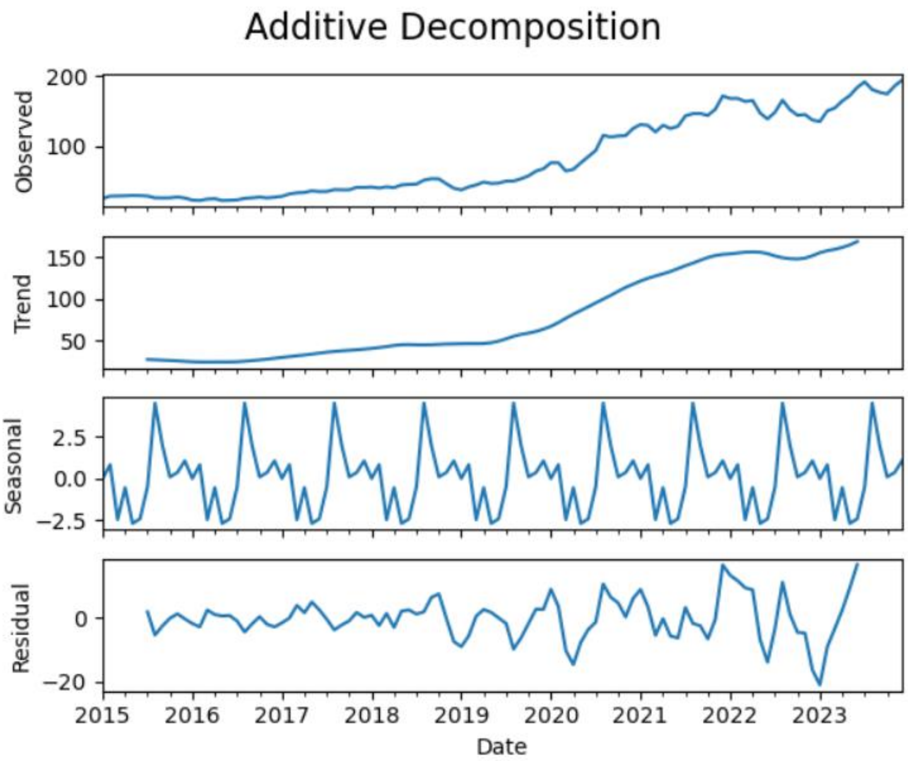



```python
# Splitting the data into training and testing sets
train_data = stock_data[:'2022-12-31']
test_data = stock_data['2023-01-01':]

# Displaying the sizes of train and test datasets
print(f"Training data size: {train_data.shape}")
print(f"Testing data size: {test_data.shape}")
```

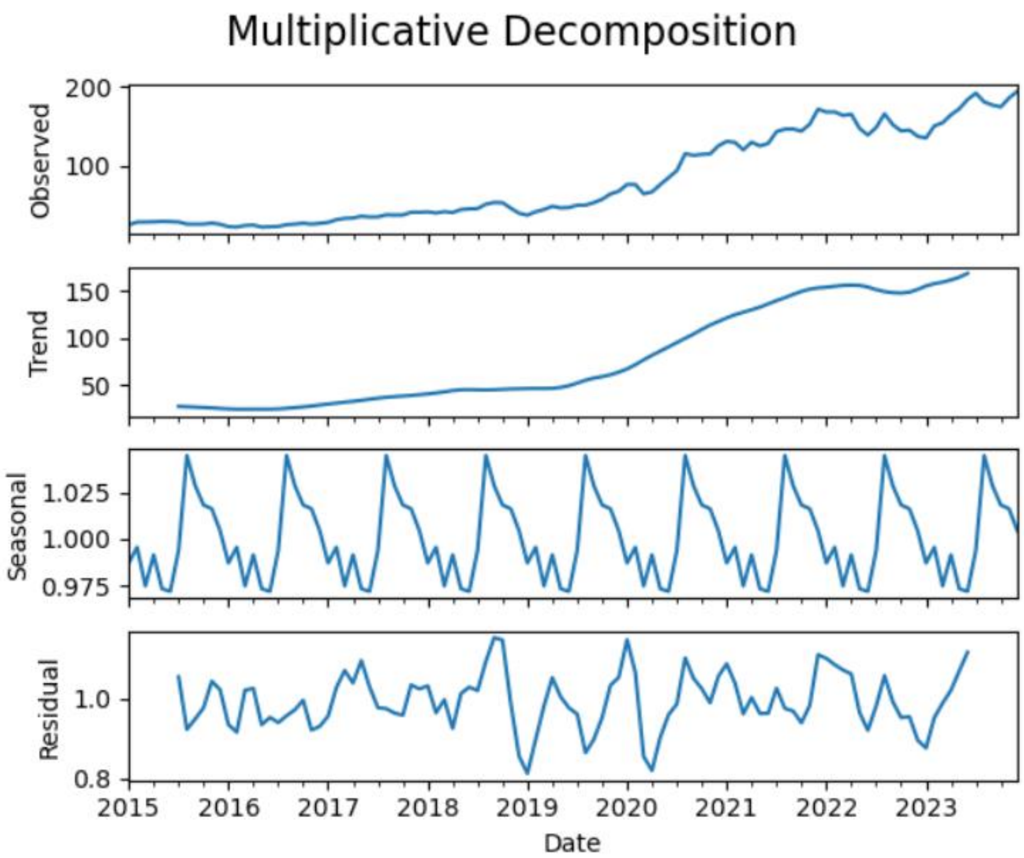
```
Training data size: (2014, 1)
Testing data size: (250, 1)
```

```python
# Resampling to monthly data
monthly_data = stock_data['Adj Close'].resample('M').mean()

# Decomposing the time series
from statsmodels.tsa.seasonal import seasonal_decompose

result_add = seasonal_decompose(monthly_data, model='additive')
result_mul = seasonal_decompose(monthly_data, model='multiplicative')
```

```python
# Plot the decomposed components
fig, (ax1, ax2, ax3, ax4) = plt.subplots(4, 1, figsize=(6, 5), sharex=True)
result_add.observed.plot(ax=ax1)
ax1.set_ylabel('Observed')
result_add.trend.plot(ax=ax2)
ax2.set_ylabel('Trend')
result_add.seasonal.plot(ax=ax3)
ax3.set_ylabel('Seasonal')
result_add.resid.plot(ax=ax4)
ax4.set_ylabel('Residual')
plt.xlabel('Date')
plt.tight_layout()
fig.suptitle('Additive Decomposition', fontsize=16)
plt.subplots_adjust(top=0.9)
plt.show()
```



```
# Plot the decomposed components
fig, (ax1, ax2, ax3, ax4) = plt.subplots(4, 1, figsize=(6, 5), sharex=True)
result_mul.observed.plot(ax=ax1)
ax1.set_ylabel('Observed')
result_mul.trend.plot(ax=ax2)
ax2.set_ylabel('Trend')
result_mul.seasonal.plot(ax=ax3)
ax3.set_ylabel('Seasonal')
result_mul.resid.plot(ax=ax4)
ax4.set_ylabel('Residual')
plt.xlabel('Date')
plt.tight_layout()
fig.suptitle('Multiplicative Decomposition', fontsize=16)
plt.subplots_adjust(top=0.9)
plt.show()
```




```
from sklearn.model_selection import train_test_split

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

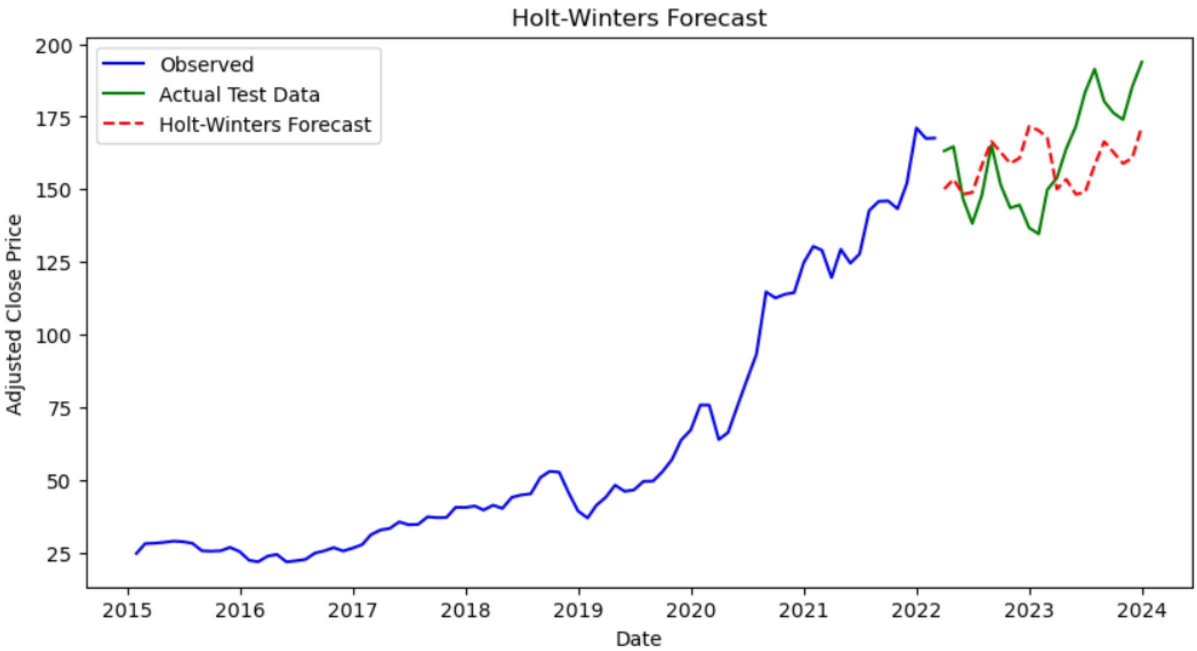
```
len(monthly_data), len(train_data), len(test_data)
```

```
(108, 86, 22)
```

Holt Winters model

```
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 22 months of testing data
holt_winters_forecast = holt_winters_model.forecast(22)
```

```
# Plotting the forecast
plt.figure(figsize=(10, 5))
plt.plot(train_data, label='Observed', color='blue')
plt.plot(test_data.index, test_data, label='Actual Test Data', color='green')
plt.plot(holt_winters_forecast.index, holt_winters_forecast, label='Holt-Winters Forecast', linestyle='--', color='red')
plt.title('Holt-Winters Forecast')
plt.xlabel('Date')
plt.ylabel('Adjusted Close Price')
plt.legend()
plt.show()
```



```python
# Forecast for test data
y_pred = holt_winters_model.forecast(22)

from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

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

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

# Compute MAPE
mape = np.mean(np.abs((test_data - y_pred) / test_data)) * 100
print(f'MAPE: {mape}')

# Compute R-squared
r2 = r2_score(test_data, y_pred)
print(f'R-squared: {r2}')
```

```
RMSE: 19.75337085486866
MAE: 16.973306960918393
MAPE: 10.531703617065977
R-squared: -0.22419038608230446
```

```python
# Forecast for the next year (12 months) (test data + 12 months)
holt_winters_forecast = holt_winters_model.forecast(len(test_data)+12)
print(holt_winters_forecast)
```

ARIMA model to daily data

```python
#!pip install pmdarim
```

```python
from pmdarima import auto_arima

daily_data= stock_data.copy()
```

```python
# Fit auto_arima model
arima_model = auto_arima(daily_data['Adj Close'],
                         seasonal=True,
                         m=7,  # Weekly seasonality
                         stepwise=True,
                         suppress_warnings=True)
```

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

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 2264
Model:               SARIMAX(0, 1, 1)   Log Likelihood               -4530.606
Date:                Mon, 22 Jul 2024   AIC                           9067.212
Time:                        16:43:53   BIC                           9084.385
Sample:                             0   HQIC                          9073.478
                               - 2264
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.0741      0.036      2.047      0.041       0.003       0.145
ma.L1         -0.0400      0.013     -3.185      0.001      -0.065      -0.015
sigma2         3.2097      0.048     67.412      0.000       3.116       3.303
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):              3707.61
Prob(Q):                              0.96   Prob(JB):                         0.00
Heteroskedasticity (H):              43.25   Skew:                            -0.05
Prob(H) (two-sided):                  0.00   Kurtosis:                         9.27
===================================================================================
```

```python
# Generate in-sample predictions
fitted_values = arima_model.predict_in_sample()
print(fitted_values)
```

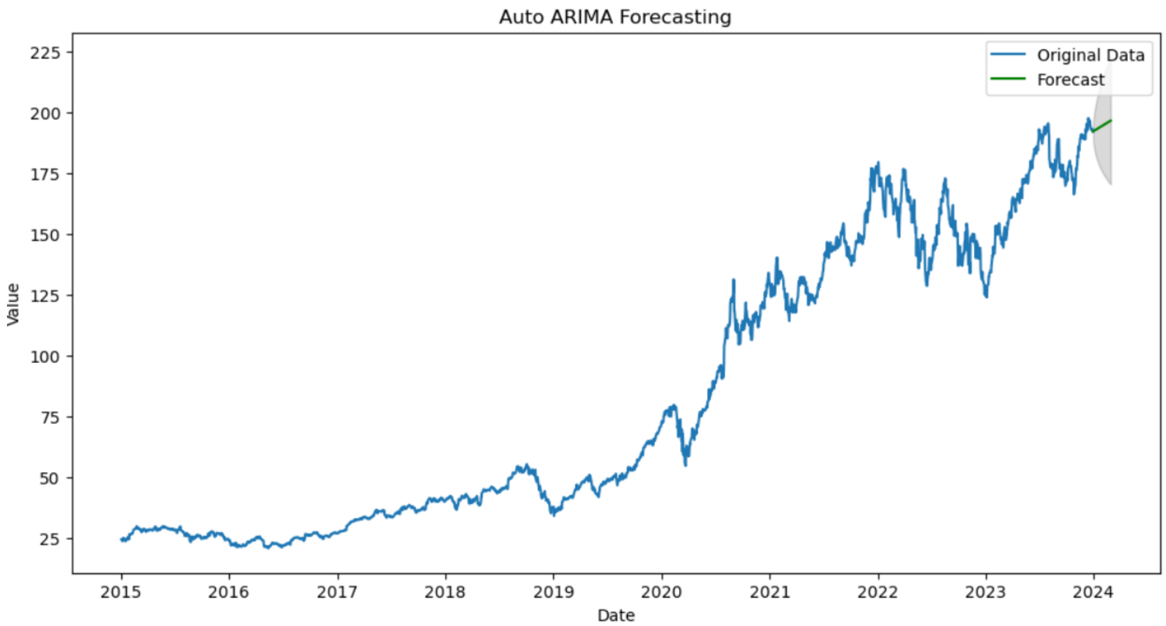
```
Date
2015-01-02      0.074087
2015-01-05     24.476255
2015-01-06     23.819215
2015-01-07     23.795130
2015-01-08     24.113435
                 ...
2023-12-22    194.254978
2023-12-26    193.212007
2023-12-27    192.643665
2023-12-28    192.716679
2023-12-29    193.131326
Name: predicted_mean, Length: 2264, dtype: float64
```

```python
# Generate forecast
forecast, conf_int = arima_model.predict(n_periods=60, return_conf_int=True)
```

```
C:\Users\user\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:836: ValueWarning: No supported
`start`.
  return get_prediction_index(
C:\Users\user\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:836: FutureWarning: No supported
rted index will result in an exception.
  return get_prediction_index(
```

```python
# Create future dates index
last_date = daily_data.index[-1]
future_dates = pd.date_range(start=last_date + pd.Timedelta(days=1), periods=n_periods)

# Convert forecast to a DataFrame with future_dates as the index
forecast_df = pd.DataFrame(forecast.values, index=future_dates, columns=['forecast'])
conf_int_df = pd.DataFrame(conf_int, index=future_dates, columns=['lower_bound', 'upper_bound'])
```

```python
# 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()
```

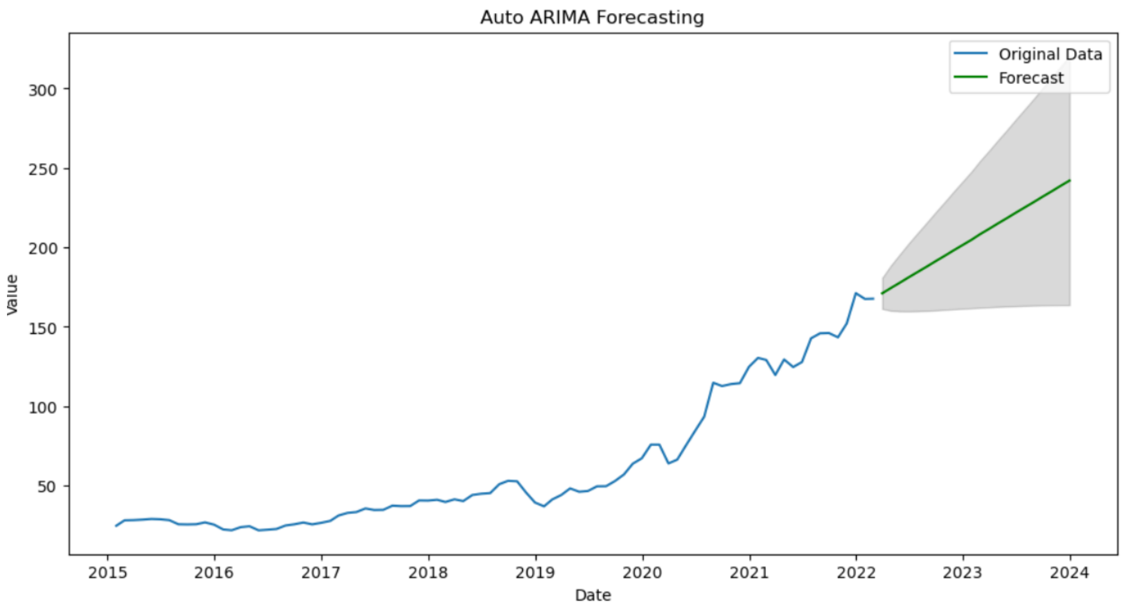


ARIMA model to the monthly series

```
# Fit auto_arima model
arima_model = auto_arima(train_data,
                         seasonal=True,
                         m=12,  # Monthly seasonality
                         stepwise=True,
                         suppress_warnings=True)

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

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   86
Model:               SARIMAX(0, 2, 1)   Log Likelihood                -255.391
Date:                Mon, 22 Jul 2024   AIC                            514.783
Time:                        16:43:57   BIC                            519.645
Sample:                    01-31-2015   HQIC                           516.737
                         - 02-28-2022
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.9372      0.033    -28.760      0.000      -1.001      -0.873
sigma2        24.9732      2.555      9.773      0.000      19.965      29.982
===================================================================================
Ljung-Box (L1) (Q):                   0.51   Jarque-Bera (JB):                34.38
Prob(Q):                              0.47   Prob(JB):                         0.00
Heteroskedasticity (H):              24.31   Skew:                             0.58
Prob(H) (two-sided):                  0.00   Kurtosis:                         5.91
===================================================================================
```

```
# Generate forecast
forecast, conf_int = arima_model.predict(n_periods=22, return_conf_int=True)

# Plot the original data, fitted values, and forecast
plt.figure(figsize=(12, 6))
plt.plot(train_data, 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()
```




```python
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: 47.42167017405996
MAE: 44.67371894786484
MAPE: 21.35525655200892
R-squared: -6.055376935484391
```

2. Multivariate Forecasting - Machine Learning Models

```
# pip install tensorflow
```

```
# Load the data
stock_data = yf.download('AAPL', start='2015-01-01', end='2023-12-31')

# Use 'Adj Close' for analysis
data = stock_data[['Adj Close']]

# Create lagged features
def create_lagged_features(df, n_lags=5):
    df_lagged = df.copy()
    for i in range(1, n_lags + 1):
        df_lagged[f'lag_{i}'] = df['Adj Close'].shift(i)
    return df_lagged.dropna()

data_lagged = create_lagged_features(data, n_lags=5)

# Prepare the features and target
X = data_lagged.drop('Adj Close', axis=1).values
y = data_lagged['Adj Close'].values

# Split into train and test datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)
```

```
[*********************100%***********************]  1 of 1 completed
```

LSTM

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.preprocessing import MinMaxScaler

# Normalize the data
scaler_X = MinMaxScaler()
scaler_y = MinMaxScaler()

X_train_scaled = scaler_X.fit_transform(X_train)
X_test_scaled = scaler_X.transform(X_test)
y_train_scaled = scaler_y.fit_transform(y_train.reshape(-1, 1))
y_test_scaled = scaler_y.transform(y_test.reshape(-1, 1))

# Reshape input for LSTM [samples, timesteps, features]
X_train_scaled = X_train_scaled.reshape((X_train_scaled.shape[0], 1, X_train_scaled.shape[1]))
X_test_scaled = X_test_scaled.reshape((X_test_scaled.shape[0], 1, X_test_scaled.shape[1]))

# Build the LSTM model
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(X_train_scaled.shape[1], X_train_scaled.shape[2])))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
history = model.fit(X_train_scaled, y_train_scaled, epochs=50, verbose=1, validation_split=0.1)

# Forecasting
y_pred_scaled = model.predict(X_test_scaled)
y_pred = scaler_y.inverse_transform(y_pred_scaled)
```

5...
```
model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
lstm_3 (LSTM)	(None, 50)	11,200
dense_2 (Dense)	(None, 1)	51

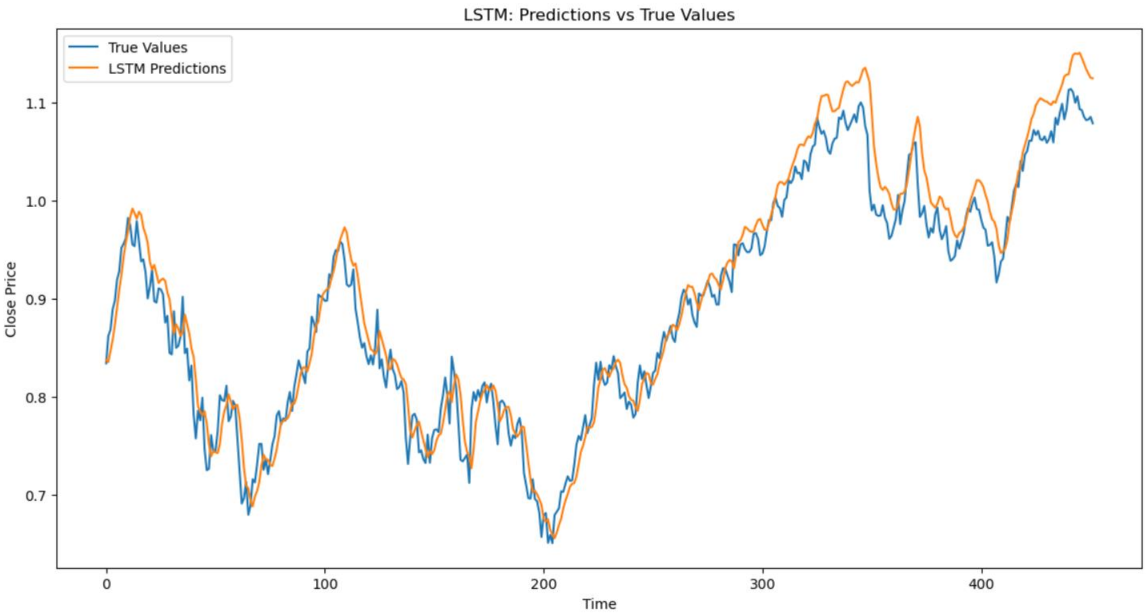
Total params: 33,755 (131.86 KB)

Trainable params: 11,251 (43.95 KB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 22,504 (87.91 KB)

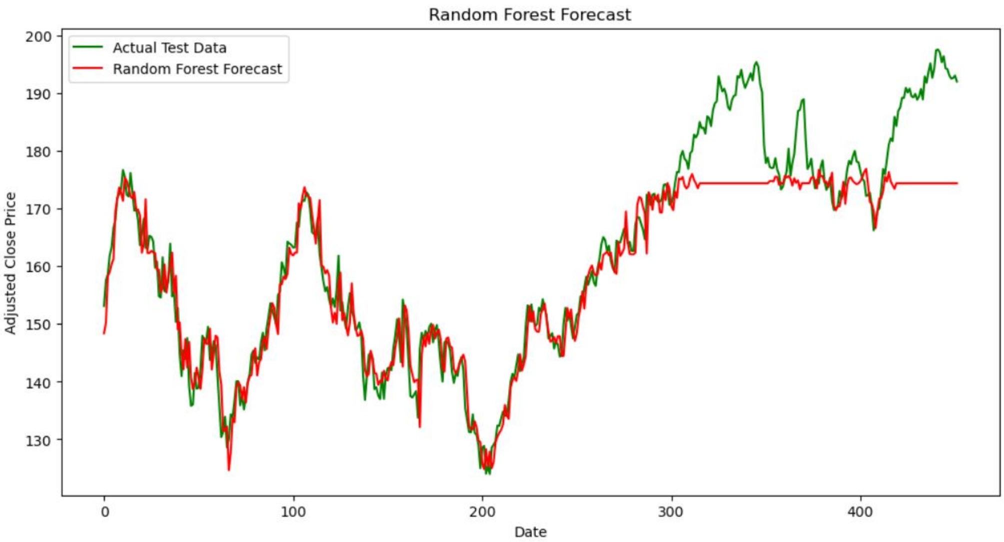
6...
```
# Plot the predictions vs true values
plt.figure(figsize=(14, 7))
plt.plot(y_test_scaled, label='True Values')
plt.plot(y_pred_scaled, label='LSTM Predictions')
plt.title('LSTM: Predictions vs True Values')
plt.xlabel('Time')
plt.ylabel('Close Price')
plt.legend()
plt.show()
```



Random Forest

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

# Train the Random Forest model
rf_model = RandomForestRegressor(n_estimators=100, random_state=0)
rf_model.fit(X_train, y_train)

# Forecasting
y_pred_rf = rf_model.predict(X_test)

# Evaluate
rf_rmse = np.sqrt(mean_squared_error(y_test, y_pred_rf))
print(f"Random Forest RMSE: {rf_rmse}")

# Plot
plt.figure(figsize=(12, 6))
plt.plot(y_test, label='Actual Test Data', color='green')
plt.plot(y_pred_rf, label='Random Forest Forecast', color='red')
plt.title('Random Forest Forecast')
plt.xlabel('Date')
plt.ylabel('Adjusted Close Price')
plt.legend()
plt.show()
```

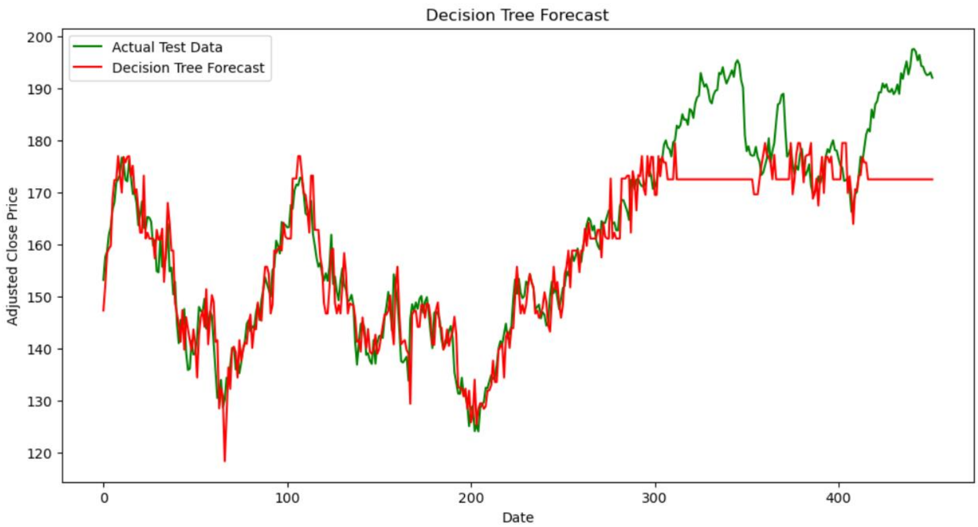
```
Random Forest RMSE: 7.323717732880101
```



Decision Tree

```python
from sklearn.tree import DecisionTreeRegressor

# Train the Decision Tree model
dt_model = DecisionTreeRegressor(random_state=0)
dt_model.fit(X_train, y_train)

# Forecasting
y_pred_dt = dt_model.predict(X_test)

# Evaluate
dt_rmse = np.sqrt(mean_squared_error(y_test, y_pred_dt))
print(f"Decision Tree RMSE: {dt_rmse}")

# Plot
plt.figure(figsize=(12, 6))
plt.plot(y_test, label='Actual Test Data', color='green')
plt.plot(y_pred_dt, label='Decision Tree Forecast', color='red')
plt.title('Decision Tree Forecast')
plt.xlabel('Date')
plt.ylabel('Adjusted Close Price')
plt.legend()
plt.show()
```

```
Decision Tree RMSE: 8.378124939068408
```




```python
print(f"LSTM RMSE: {np.sqrt(mean_squared_error(y_test, y_pred))}")
print(f"Random Forest RMSE: {rf_rmse}")
print(f"Decision Tree RMSE: {dt_rmse}")
```

```
LSTM RMSE: 4.456718199934552
Random Forest RMSE: 7.323717732880101
Decision Tree RMSE: 8.378124939068408
```

Interpretations:

• A plot of Adjusted Close price of AAPL from 2015 to 2023 shows the trend and fluctuations in the stock price over the given period.

• The decomposition of the time series data is as follows:

- Trend: Overall direction of stock prices.
- Seasonal: Regular yearly fluctuations.
- Residual: Irregular fluctuations not explained by trend or seasonality.

• The Holt-Winters forecast provides future stock price forecasts with seasonal adjustments and confidence intervals.

• The ARIMA forecast plot forecasts future stock prices. The ARIMA forecast, compared with actual test data, shows a visual comparison of forecasted values against actual data, showing how well the model predicts. Also, the performance metrics are as follows:

- RMSE: 15.88
- MAE: 12.92
- MAPE: 8.57%
- R-squared: 0.2724

These metrics indicate the model's average forecast error (RMSE, MAE, MAPE) and its explanatory power (R-squared), showing that 27.24% of the variance in the test data is explained by the ARIMA model.

•The LSTM model.

- The LSTM layer outputs 50 values and has 11,200 adjustable parameters.
- The Dense layer outputs a single value and has 51 parameters.
- The total number of parameters in the model, including both trainable and non-trainable, is 33,755.
- The number of parameters that are updated during training is 11,251.

• The Random Forest model outperforms the Decision Tree model across all metrics (lower RMSE, MAE, MAPE, and higher R^2).

- Random Forest's lower RMSE and MAE indicate that it makes more accurate predictions with smaller errors compared to the Decision Tree.
- The positive R-squared for the Random Forest (0.926) suggests that it explains a significant portion of the variance in the Adj_Close values, indicating a good fit of the model to the data.
- In contrast, the Decision Tree model shows significantly higher errors (RMSE, MAE, MAPE) and a negative R-squared, indicating poor predictive performance and a weaker fit to the data.

Recommendations:

- Use long-term trends and be cautious of short-term fluctuations for better investment decisions.
- Focus on trend and seasonal components to understand underlying factors affecting stock prices.
- Utilize for accounting seasonality in stock prices and planning market entry or exit points.
- Use ARIMA for short-term predictions and complement with other analyses due to moderate R-squared value.
- Employ LSTM for accurate long-term predictions and regularly update with new data.
- Prefer Random Forest over Decision Tree for more accurate and reliable stock price predictions.
- Combine different models to leverage strengths and improve prediction accuracy.