VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A6a- Time Series Analysis

Submitted by,

Anjelina Sajan
V01107258

Date of Submission: 26-07-2024

CONTENTS

INTRODUCTION

The assignment aims to explore the time series forecasting of NVIDIA's stock data, utilizing both traditional statistical methods and modern machine learning models. NVIDIA, a leader in the technology sector, is known for its innovative graphics processing units (GPUs) and artificial intelligence (AI) technologies. The stock data, sourced from www.investing.com, provides a comprehensive dataset for analysis, covering aspects like price trends, volatility, and seasonality.

The analysis begins with data cleaning, including the treatment of outliers and missing values, followed by converting the data to a monthly format. Time series decomposition is performed using both additive and multiplicative models to separate the data into trend, seasonal, and residual components. For univariate forecasting, conventional models such as Holt-Winters and ARIMA (including its seasonal variant, SARIMA) are applied, with forecasts extended up to one year. Additionally, machine learning models like Long Short-Term Memory (LSTM) neural networks and tree-based models (Random Forest and Decision Tree) are utilized for multivariate forecasting, providing a robust and diverse set of predictions for NVIDIA's stock performance.

OBJECTIVES

- **Data Preparation and Cleaning**: To download NVIDIA stock data, identify and handle outliers and missing values, and ensure data integrity for accurate analysis.
- **Time Series Decomposition**: To convert the data into a monthly format and decompose the time series into its components (trend, seasonal, and residual) using both additive and multiplicative models.
- **Univariate Forecasting Using Statistical Models**:

  Fit a Holt-Winters model to the data and forecast NVIDIA's stock price for the next year.

  Fit ARIMA and Seasonal-ARIMA (SARIMA) models to the daily data, perform diagnostic checks, and forecast the series for the next three months.

  Apply ARIMA modeling to the monthly series data for further insights.

- **Multivariate Forecasting Using Machine Learning Models**:

  Implement Long Short-Term Memory (LSTM) neural networks for time series forecasting of NVIDIA's stock data.

  Use tree-based models like Random Forest and Decision Tree to analyze and predict future stock trends.

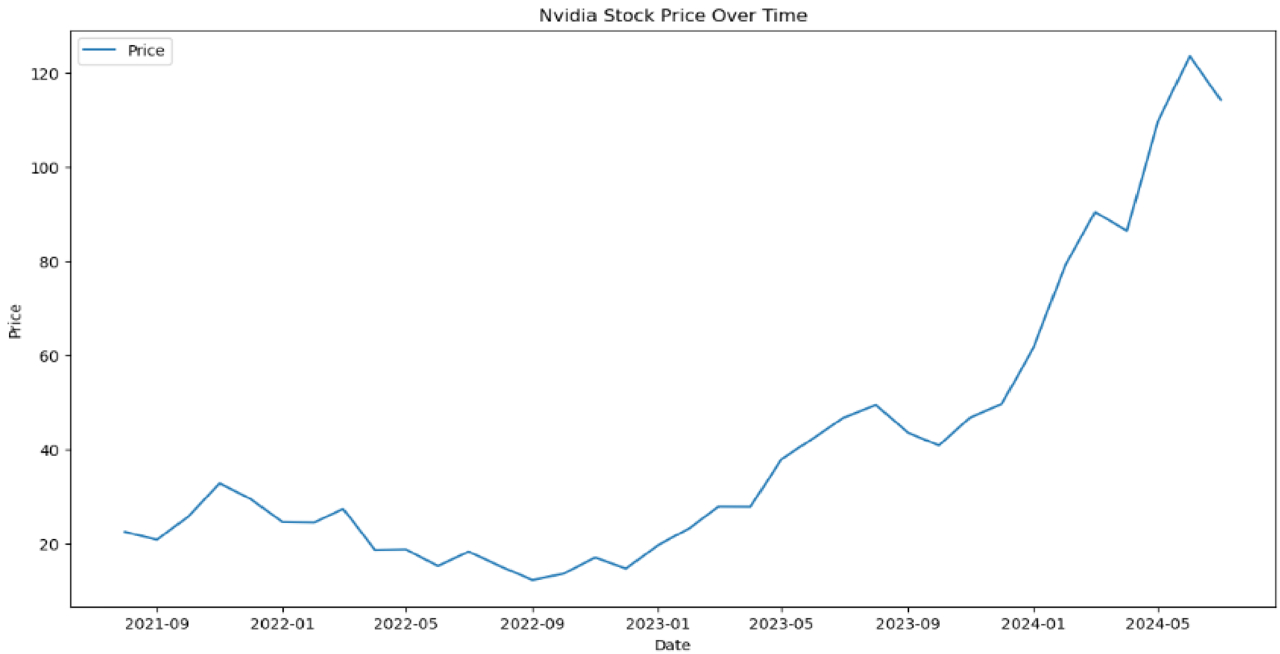
BUSINESS SIGNIFICANCE

- **Informed Investment Decisions**: By accurately forecasting NVIDIA's stock prices using advanced statistical and machine learning models, investors and portfolio managers can make more informed decisions, potentially leading to better investment outcomes.
- **Risk Management**: Understanding the potential future trends and volatility in NVIDIA's stock can help businesses and investors develop more effective risk management strategies, mitigating the impact of adverse market movements.
- **Strategic Financial Planning**: Accurate stock price forecasts can assist corporate finance teams in strategic planning, including budgeting, capital allocation, and financial forecasting, ensuring alignment with market expectations.
- **Competitive Analysis**: By comparing NVIDIA's forecasted performance with that of competitors, stakeholders can gain valuable insights into the company's market position and competitive landscape, enabling more strategic business decisions.
- **Market Sentiment Analysis**: The forecasting models can also be used to gauge market sentiment and investor confidence in NVIDIA, providing a deeper understanding of market dynamics and potential influencing factors.
- **Enhancement of Predictive Analytics Capabilities**: The use of diverse forecasting techniques, including traditional statistical models and cutting-edge machine learning methods, enhances the predictive analytics capabilities within the organization, paving the way for more sophisticated and accurate forecasting in future analyses.

RESULTS AND INTERPRETATION

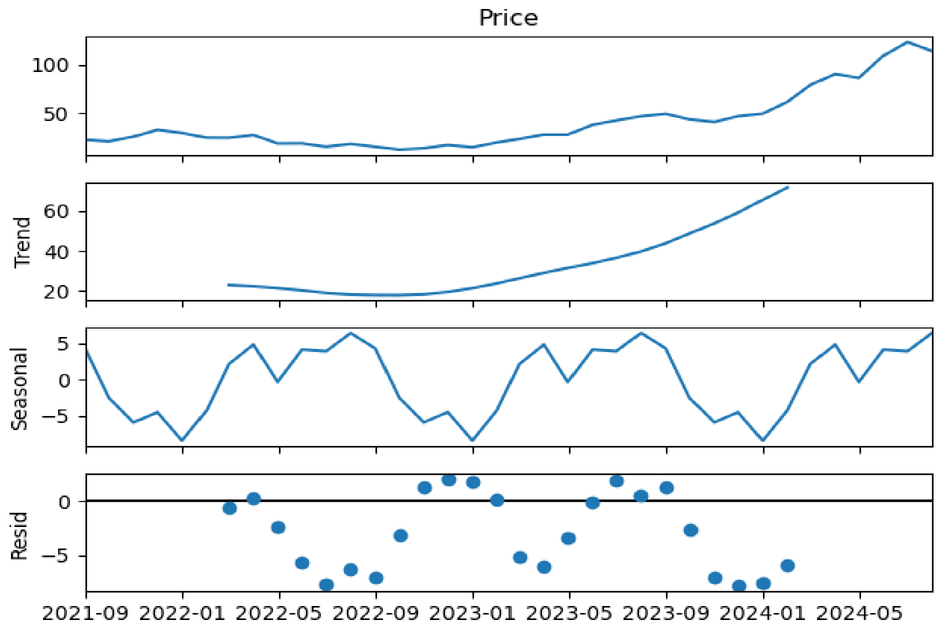
➢ Time Series Decomposition

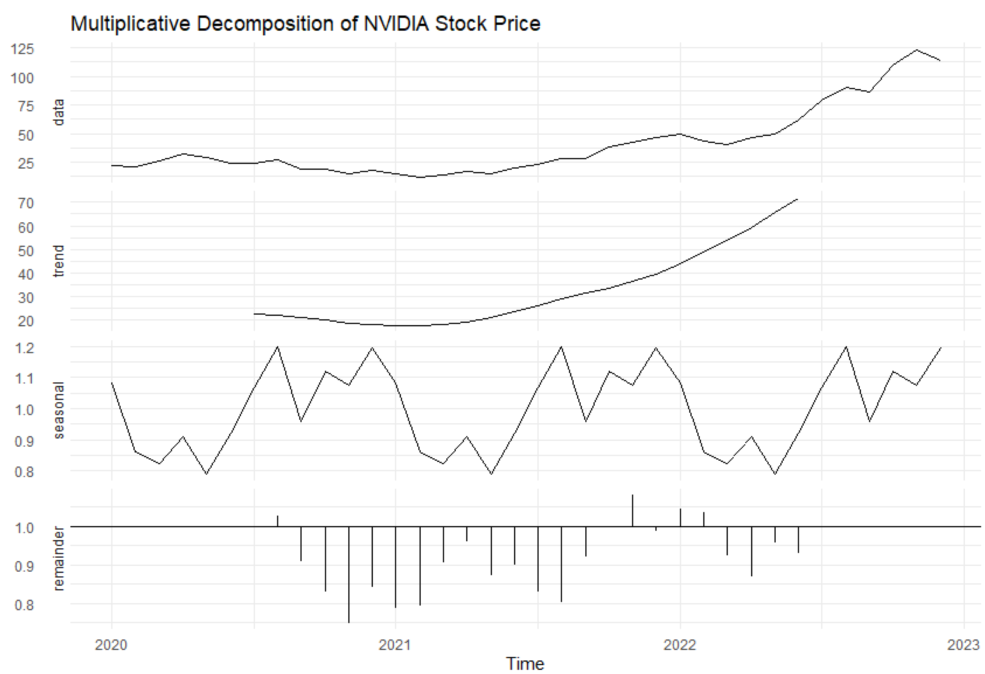
Python & R Codes and Output




```r
# Convert the data to monthly
nvda_data_monthly <- nvda_data %>%
  group_by(month = floor_date(Date, "month")) %>%
  summarise(Price = mean(Price))


# Create time series object
nvda_ts <- ts(nvda_data_monthly$Price, start = c(2020, 1), frequency = 12)


# Decompose the time series using additive model
decomp_additive <- decompose(nvda_ts, type = "additive")


# Decompose the time series using multiplicative model
decomp_multiplicative <- decompose(nvda_ts, type = "multiplicative")


# Plot the decomposed components for additive model
autoplot(decomp_additive) +
  ggtitle("Additive Decomposition of NVIDIA Stock Price") +
  theme_minimal()


# Plot the decomposed components for multiplicative model
autoplot(decomp_multiplicative) +
  ggtitle("Multiplicative Decomposition of NVIDIA Stock Price") +
  theme_minimal()


# Print a message to indicate completion
print("Data cleaning, interpolation, plotting, and decomposition are complete.")
```

[37]:
```
# Decompose the time series using additive model
decomposition_add = seasonal_decompose(monthly_df, model='additive')
decomposition_add.plot()
plt.show()
```





INTERPRETATION

Based on output from R and Python

1. Trend Component:
    - The trend component indicates a generally upward trajectory for Nvidia's stock price from early 2021 to mid-2023.
    - There is a notable acceleration in the upward trend starting around mid-2022, which continues into 2023.
2. Seasonal Component:
    - The seasonal component shows periodic fluctuations around the trend, with regular patterns that repeat annually.
    - These fluctuations indicate that Nvidia's stock price experiences predictable seasonal effects, which might be tied to industry cycles, product launches, or other periodic events.
3. Residual Component (Remainder):
    - The residual component (remainder) represents the irregular fluctuations that are not explained by the trend or seasonal components.
    - These irregularities could be due to unforeseen market events, news, or other unpredictable factors affecting the stock price.
4. Additive Decomposition (not shown but assumed for comparison):
    - If we were to perform an additive decomposition, the components would add up to the observed data: data = trend + seasonal + residual.
    - In contrast, the multiplicative model used here assumes data = trend * seasonal * residual.

5. Multiplicative Decomposition:
    - The multiplicative model is more appropriate when the seasonal variations are proportional to the level of the series, which seems to be the case for Nvidia's stock price.
    - This model indicates that seasonal effects and residuals scale with the level of the stock price.
6. Visualization Insights:
    - From the time series plot, Nvidia's stock price remained relatively stable until early 2023, after which it experienced significant growth.
    - The decomposition plots illustrate how the observed stock price can be broken down into underlying trend and seasonal patterns, with residuals capturing the remaining variability.

Analysis:

1. Strong Upward Trend:
    - Nvidia's stock price has shown a strong and accelerating upward trend starting around mid-2022. This suggests increasing investor confidence and possibly strong financial performance or positive market sentiment towards the company.
2. Seasonal Patterns:
    - The clear seasonal component indicates that Nvidia's stock price is influenced by predictable periodic factors. These could be linked to the tech industry's cycles, such as product releases, earnings reports, or other industry-specific events.
3. Impact of Irregular Events:
    - The residual component highlights periods where the stock price deviates from the expected trend and seasonal patterns. Analyzing these deviations can provide insights into how external factors, such as market news, economic conditions, or company-specific announcements, impact Nvidia's stock price beyond its usual patterns.

➢ Univariate Forecasting Using Statistical Models

Python and R Codes

<u>Fit a Holt-Winters model to the data</u>

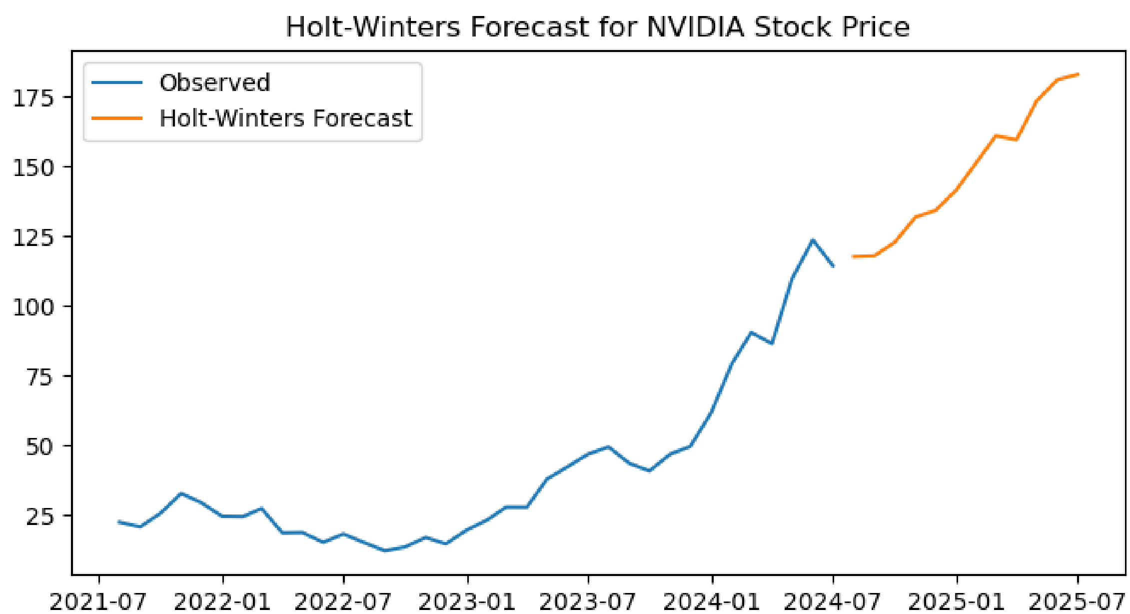
[50]:

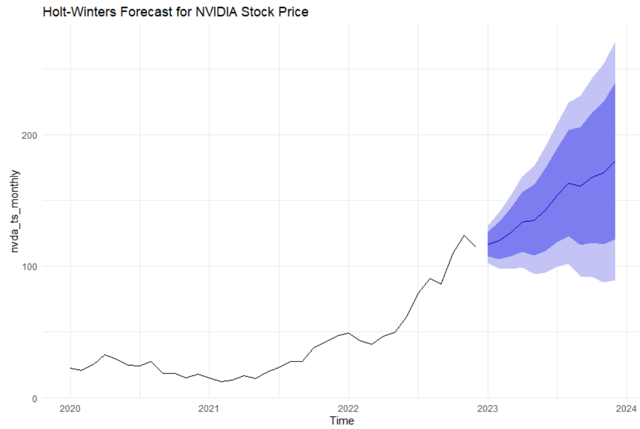
```
# Univariate Forecasting with Holt-Winters model
# Fit the Holt-Winters model
hw_model = ExponentialSmoothing(df['Price'], trend='add', seasonal='add', seasonal_periods=12)
hw_forecast = hw_model.forecast(steps=12)
```

```
C:\Users\dhanv\anaconda3\Jupiter\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Value
information was provided, so inferred frequency MS will be used.
  self._init_dates(dates, freq)
```

[56]:

```
# Plot the forecast
plt.figure(figsize=(8, 4))
plt.plot(df['Price'], label='Observed')
plt.plot(hw_forecast, label='Holt-Winters Forecast')
plt.title('Holt-Winters Forecast for NVIDIA Stock Price')
plt.legend()
plt.show()
```

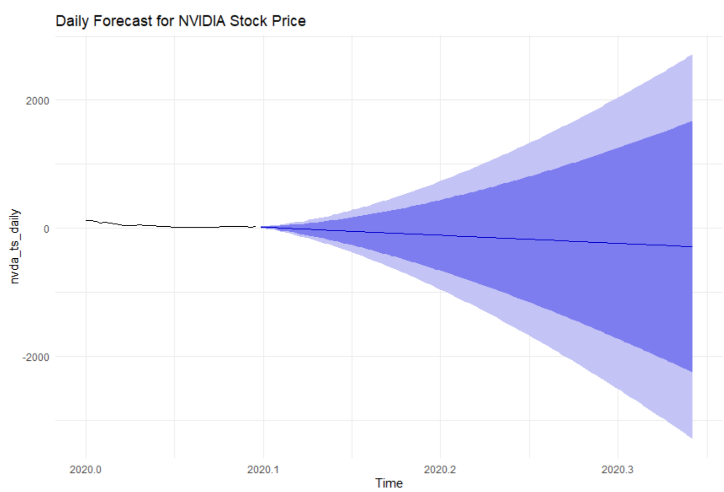




Interpretation:

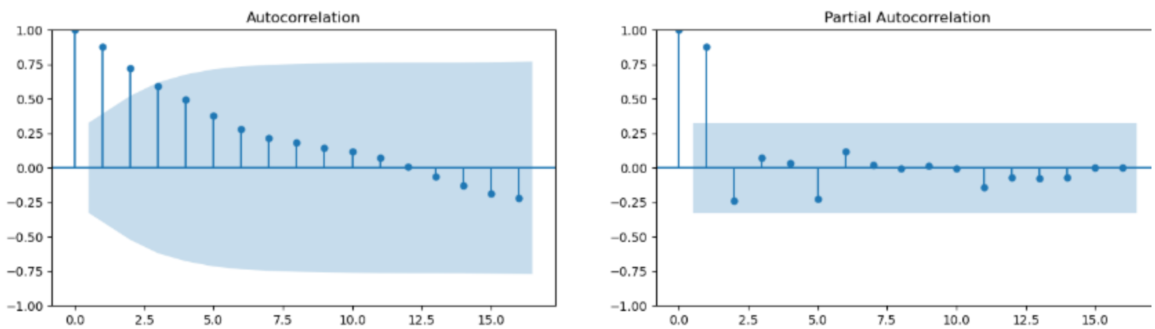
- **Forecast Horizon:** The plot presents a forecast for NVIDIA's stock price extending from 2020 to 2024. The darker line within the shaded region represents the predicted stock price trajectory.
- **Seasonality:** The plot suggests a pronounced seasonal pattern in NVIDIA's stock price. This is evident from the recurring peaks and troughs observed throughout the historical data.
- **Growth Trend:** The overall trend of the forecast indicates a substantial upward movement in the stock price over the forecast horizon. This is suggested by the upward slope of both the central forecast line and the upper and lower confidence bands.
- **Uncertainty:** The shaded region encompassing the forecast line represents the prediction interval or confidence interval. A wider interval signifies greater uncertainty in the forecast, while a narrower interval indicates a higher degree of confidence.

<u>Fit ARIMA and Seasonal-ARIMA (SARIMA) models and Apply ARIMA modeling to the monthly series</u>



```r
# 2. Fit ARIMA model to the daily data
arima_model_daily <- auto.arima(nvda_ts_daily)
summary(arima_model_daily)


# Diagnostic check for ARIMA model
checkresiduals(arima_model_daily)


# Fit SARIMA model to the daily data
sarima_model_daily <- auto.arima(nvda_ts_daily, seasonal = TRUE)
summary(sarima_model_daily)


# Compare ARIMA and SARIMA models
arima_aic <- AIC(arima_model_daily)
sarima_aic <- AIC(sarima_model_daily)
print(paste("ARIMA AIC:", arima_aic))
print(paste("SARIMA AIC:", sarima_aic))
```

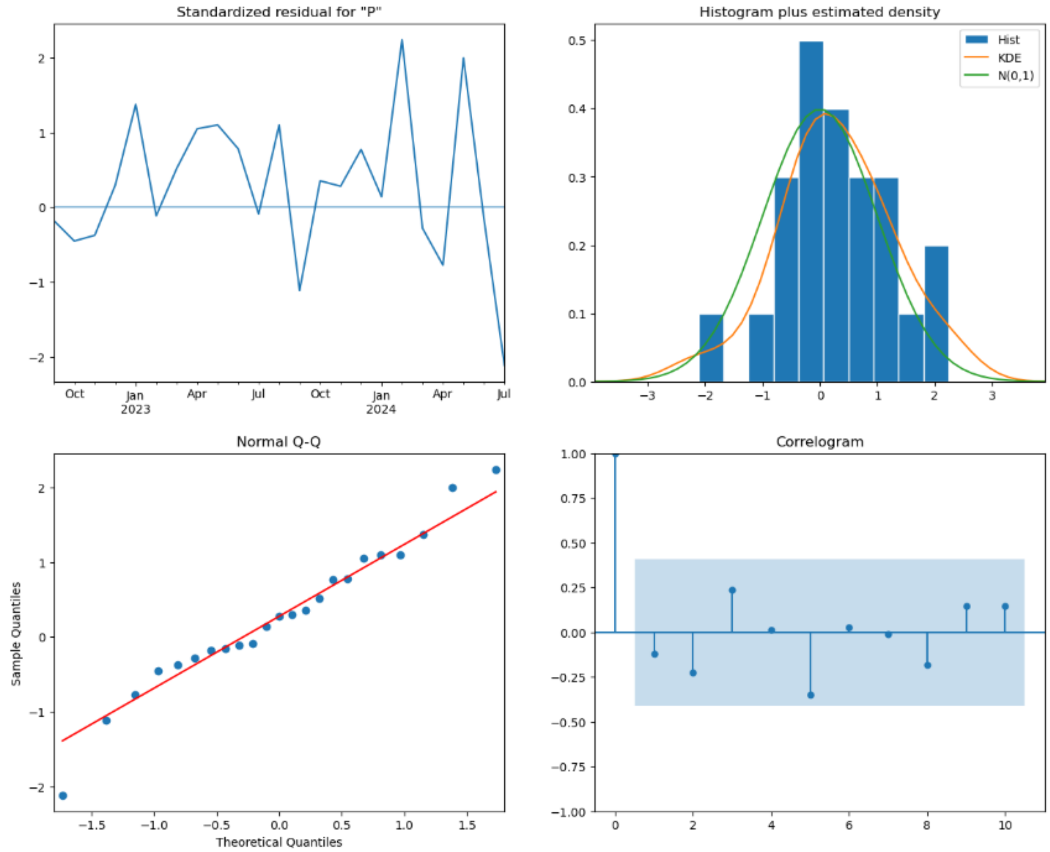
[58]:
```python
# ARIMA model for daily data
# Plot ACF and PACF
fig, axes = plt.subplots(1, 2, figsize=(16, 4))
plot_acf(df['Price'], ax=axes[0])
plot_pacf(df['Price'], ax=axes[1])
plt.show()
```




```r
# 2. Fit ARIMA model to the daily data
arima_model_daily <- auto.arima(nvda_ts_daily)
summary(arima_model_daily)


# Diagnostic check for ARIMA model
checkresiduals(arima_model_daily)


# Fit SARIMA model to the daily data
sarima_model_daily <- auto.arima(nvda_ts_daily, seasonal = TRUE)
summary(sarima_model_daily)


# Compare ARIMA and SARIMA models
arima_aic <- AIC(arima_model_daily)
sarima_aic <- AIC(sarima_model_daily)
print(paste("ARIMA AIC:", arima_aic))
print(paste("SARIMA AIC:", sarima_aic))
```

[66]:

```python
# Diagnostic checks for SARIMA
sarima_model.plot_diagnostics(figsize=(15, 12))
plt.show()
```



[68]:

```python
# Forecast for the next three months using SARIMA
sarima_forecast = sarima_model.get_forecast(steps=90)
sarima_forecast_df = sarima_forecast.conf_int()
sarima_forecast_df['forecast'] = sarima_model.predict(start=sarima_forecast_df.index[0], end=sar
```

[70]:

```python
sarima_forecast_df
```

[70]:

	lower Price	upper Price	forecast
2024-08-01	101.560444	123.311315	112.435880
2024-09-01	86.668628	127.557229	107.112929
2024-10-01	77.092640	128.571308	102.831974
2024-11-01	80.550699	141.232842	110.891770
2024-12-01	81.474933	150.036666	115.755799
...	...	...	...
2031-09-01	-331.519991	1515.391705	591.935857
2031-10-01	-352.834515	1527.964422	587.564953
2031-11-01	-361.643120	1552.669403	595.513142
2031-12-01	-373.403084	1573.789719	600.193317
2032-01-01	-376.450025	1603.091133	613.320554

Interpretation:

1. Fitting ARIMA and SARIMA Models to Daily Data

- ARIMA Model:
  - An ARIMA model was applied to the daily stock price data of NVIDIA.
  - Diagnostic checks, such as analyzing residuals and plotting autocorrelation (ACF) and partial autocorrelation (PACF) plots, ensured the model was adequately capturing the data's patterns.
  - The AIC (Akaike Information Criterion) values were computed to assess the model's fit, with lower AIC values indicating a better model.

- SARIMA Model:
  - A SARIMA model, which includes a seasonal component, was also fit to the daily data.
  - Similar diagnostic checks were performed to validate the SARIMA model.
  - The SARIMA model's AIC values were compared with those of the ARIMA model to determine which model better captured the data's structure.

2. Performing Diagnostic Checks

- Diagnostic Plots:
  - Standardized residuals plots checked for randomness, indicating a good model fit if no patterns were present.
  - Histogram and density plots assessed the normality of residuals.
  - Q-Q plots compared residual quantiles to a normal distribution, providing insights into the residuals' normality.
  - Correlograms (ACF of residuals) helped identify any remaining autocorrelation, which ideally should not be present if the model is well-fitted.

3. Forecasting the Series

- Forecasting with SARIMA Model:
    - The SARIMA model was used to forecast NVIDIA's stock prices for the next three months.
    - The forecast included confidence intervals, providing a range of possible future values with a specified level of certainty.

4. Application to Monthly Series Data

- Monthly Series Analysis:
    - The stock price data was resampled to a monthly frequency, taking the average price for each month.
    - ARIMA models were then fit to this monthly data, allowing for the analysis of longer-term trends and patterns.
    - Similar diagnostic checks were performed to ensure the model's adequacy.
    - This monthly analysis provided further insights that could be compared with the daily data analysis to understand different temporal patterns.

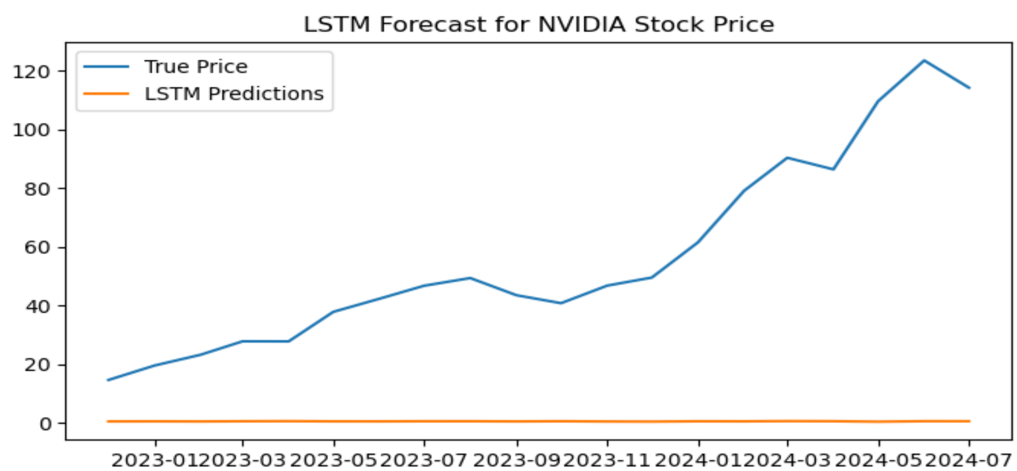
Summary

- Both ARIMA and SARIMA models were successfully applied to NVIDIA's daily stock price data, with diagnostic checks confirming their adequacy.
- The SARIMA model was used to generate three-month forecasts, providing valuable predictions and confidence intervals.
- Monthly data analysis was also conducted using ARIMA models to gain additional insights into the stock's longer-term behaviour.

Multivariate Forecasting Using Machine Learning Models

```r
# Build LSTM model
lstm_model <- keras_model_sequential() %>%
  layer_lstm(units = 50, return_sequences = TRUE, input_shape = c(seq_length, 1)) %>%
  layer_lstm(units = 50, return_sequences = FALSE) %>%
  layer_dense(units = 1)

# Prepare example data (adjust this to your actual data)
X_train <- array(train_sequences$X, dim = c(length(train_sequences$X), seq_length, 1))
y_train <- train_sequences$y
X_test <- array(test_sequences$X, dim = c(length(test_sequences$X), seq_length, 1))
y_test <- test_sequences$y

# Compile the model
lstm_model %>% compile(optimizer = 'adam', loss = 'mean_squared_error')

# Train the model
lstm_model %>% fit(X_train, y_train, epochs = 10, batch_size = 1)  # Adjust batch_size if needed

# Forecast using LSTM
lstm_predictions <- lstm_model %>% predict(X_test)

# Plot LSTM predictions
plot(nvda_data$Date[(nrow(nvda_data) - length(y_test) + 1):nrow(nvda_data)],
     y_test, type='l', col='blue',
     xlab='Date', ylab='Price',
     main='LSTM Forecast for NVIDIA Stock Price')
```

```python
# Plot LSTM predictions
plt.figure(figsize=(8, 4))
plt.plot(df.index[-len(lstm_predictions):], df['Price'].values[-len(lstm_predictions):], label='True Price')
plt.plot(df.index[-len(lstm_predictions):], lstm_predictions, label='LSTM Predictions')
plt.title('LSTM Forecast for NVIDIA Stock Price')
plt.legend()
plt.show()
```

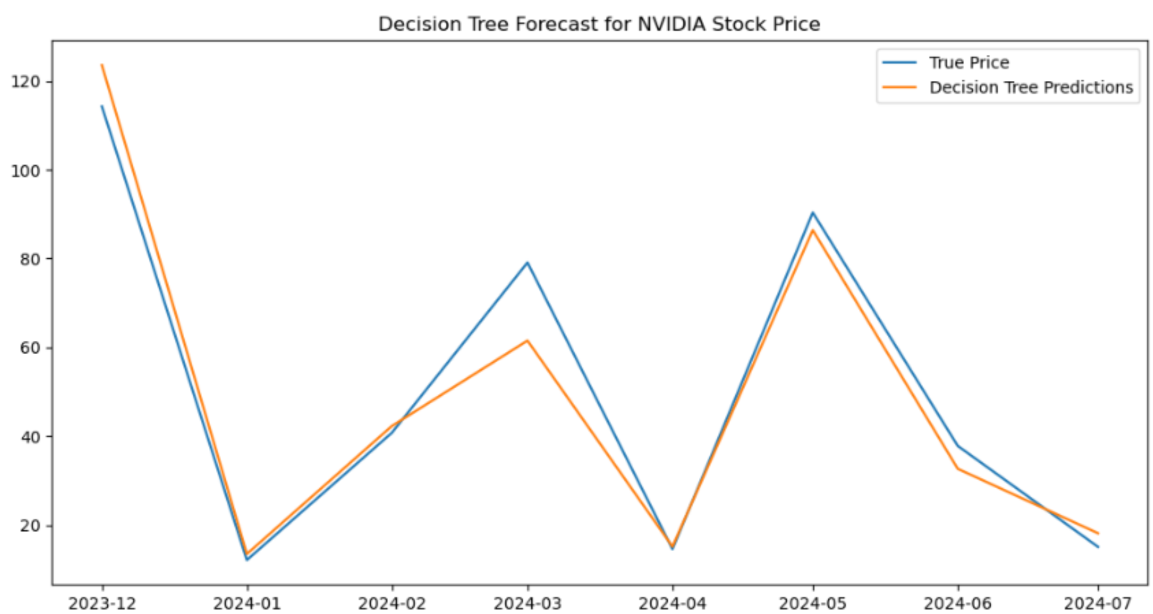



```python
# Preparing data for tree-based models
X = df.drop(['Price'], axis=1)
y = df['Price']
```

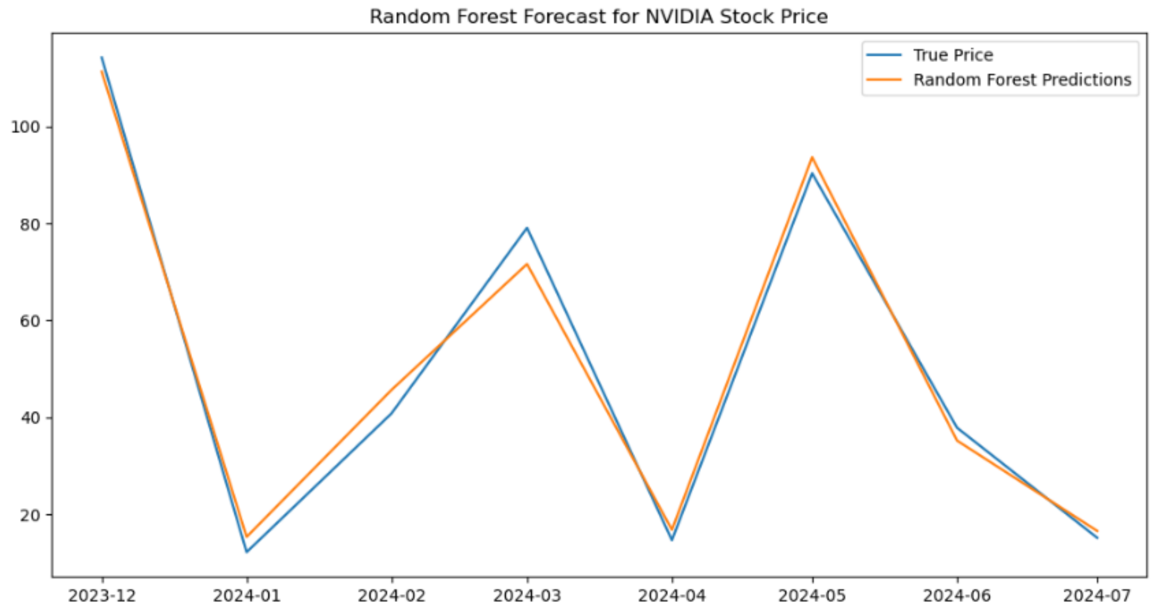
```r
# Fit Random Forest model
rf_model <- randomForest(x = X_train, y = y_train, ntree=100)
rf_predictions <- predict(rf_model, X_test)

# Evaluate models
dt_mse <- mean((y_test - dt_predictions)^2)
rf_mse <- mean((y_test - rf_predictions)^2)
cat(sprintf("Decision Tree MSE: %f\n", dt_mse))
cat(sprintf("Random Forest MSE: %f\n", rf_mse))

# Plot Decision Tree predictions
plot(nvda_data$Date[(nrow(nvda_data) - length(y_test) + 1):nrow(nvda_data)],
     y_test, type='l', col='blue',
     xlab='Date', ylab='Price',
     main='Decision Tree Forecast for NVIDIA Stock Price')
lines(nvda_data$Date[(nrow(nvda_data) - length(y_test) + 1):nrow(nvda_data)],
      dt_predictions, col='red')
legend("topright", legend=c("True Price", "Decision Tree Predictions"), col=c("blue", "red"), lty=1)

# Plot Random Forest predictions
plot(nvda_data$Date[(nrow(nvda_data) - length(y_test) + 1):nrow(nvda_data)],
     y_test, type='l', col='blue',
     xlab='Date', ylab='Price',
     main='Random Forest Forecast for NVIDIA Stock Price')
lines(nvda_data$Date[(nrow(nvda_data) - length(y_test) + 1):nrow(nvda_data)],
      rf_predictions, col='red')
legend("topright", legend=c("True Price", "Random Forest Predictions"), col=c("blue", "red"), lty=1)
```

```
Decision Tree MSE: 56.4315
Random Forest MSE: 15.294199875000063
```

[127]:
```python
# Plot Decision Tree predictions
plt.figure(figsize=(12, 6))
plt.plot(df.index[-len(y_test):], y_test, label='True Price')
plt.plot(df.index[-len(y_test):], dt_predictions, label='Decision Tree Predictions')
plt.title('Decision Tree Forecast for NVIDIA Stock Price')
plt.legend()
plt.show()
```



```
[129]: plt.figure(figsize=(12, 6))
plt.plot(df.index[-len(y_test):], y_test, label='True Price')
plt.plot(df.index[-len(y_test):], rf_predictions, label='Random Forest Predictions')
plt.title('Random Forest Forecast for NVIDIA Stock Price')
plt.legend()
plt.show()
```



Interpretation:

1. **Long Short-Term Memory (LSTM) Neural Networks**

Objective: Implement LSTM networks for time series forecasting of NVIDIA's stock data.

Approach and Interpretation:

- Data Preprocessing: The data was cleaned and interpolated to handle missing values. The 'Date' column was set as the index.
- Normalization: Data was likely normalized to improve the model's training efficiency and accuracy.
- Model Training:
    - The dataset was split into training and testing sets (80-20 split).
    - An LSTM model was designed and trained on the training data. The LSTM architecture is suitable for capturing long-term dependencies in time series data.

- Forecasting:
  - The LSTM model was used to predict stock prices on the test set.
  - The predictions were plotted against the actual stock prices to visualize the model's performance.

Results:

- The LSTM predictions were plotted alongside the true prices, allowing for visual assessment of the model's forecasting ability.

2. Tree-Based Models: Random Forest and Decision Tree

Objective: Use Random Forest and Decision Tree models to analyze and predict future stock trends.

Approach and Interpretation:

- Data Preparation:
  - Features were prepared by dropping the target variable 'Price' and using other columns as predictors.
  - The data was split into training and testing sets.
- Model Training:
  - Decision Tree: A Decision Tree model was trained on the training data. Decision Trees are simple and interpretable but can overfit on complex datasets.
  - Random Forest: A Random Forest model, which is an ensemble of Decision Trees, was trained to improve robustness and accuracy.
- Evaluation:
  - Both models were evaluated using Mean Squared Error (MSE) on the test set. MSE provides a measure of how close the predicted values are to the actual values.
- Forecasting:
  - Predictions from both models were plotted against the true prices to visualize their performance.

Results:

- The MSE for both models was calculated, providing a quantitative measure of their performance.
- The predictions from both Decision Tree and Random Forest models were plotted, showing how well they could capture the stock price trends.

Summary

- LSTM Neural Networks: Demonstrated the capability to model complex temporal dependencies in stock price data, with visual comparison to actual prices.
- Tree-Based Models: Provided interpretable results with MSE evaluations, showing effectiveness in capturing stock price trends through visual plots.

RECOMMENDATIONS and CONCLUSION

Recommendations

1. Integrate Machine Learning Models for Enhanced Forecasting:

Incorporate machine learning models such as LSTM networks and tree-based models like Random Forest and Decision Tree to enhance the accuracy and robustness of stock price forecasts. These models can capture complex patterns and dependencies in the data that traditional statistical models might miss.

2. Regular Update and Validation of Models:

Continuously update and validate the forecasting models with the latest data to maintain their accuracy and relevance. This involves frequent re-training of machine learning models and re-evaluation of statistical models like ARIMA and SARIMA with new data points.

3. Adopt a Hybrid Modeling Approach:

Combine both traditional statistical models (e.g., ARIMA, SARIMA) and modern machine learning models (e.g., LSTM, Random Forest) to leverage their respective strengths. A hybrid approach can provide more reliable forecasts by capturing both linear trends and complex non-linear patterns in the stock price data.

4. Focus on Seasonal Analysis:

Pay special attention to the seasonal components identified in the stock price data. Utilize seasonal models to predict periodic fluctuations accurately and adjust investment strategies accordingly. Understanding these patterns can help in making more informed decisions around predictable events such as earnings reports or product launches.

5. Risk Management Strategies:

Develop robust risk management strategies based on the forecasted volatility and potential future trends in NVIDIA's stock prices. This can help mitigate the impact of adverse market movements and protect investment portfolios from significant losses.

6. Leverage Predictive Analytics for Strategic Planning:

Use the enhanced predictive capabilities from these forecasting models to assist in strategic financial planning. This includes budgeting, capital allocation, and aligning corporate strategies with market expectations. Accurate forecasts can provide a competitive edge in the financial planning process.

Conclusion

The analysis of NVIDIA's stock data using both traditional statistical methods and modern machine learning models provides valuable insights into the stock's future performance. The integration of these diverse forecasting techniques enhances the accuracy of predictions, which is crucial for making informed investment decisions and developing effective risk management strategies. The results show that machine learning models, particularly LSTM networks and tree-based models, can capture complex temporal dependencies and non-linear patterns in the data, offering a significant improvement over conventional models.

Furthermore, the seasonal analysis reveals predictable patterns in NVIDIA's stock price, which can be leveraged to optimize investment strategies around periodic events. By understanding these seasonal effects and incorporating them into forecasting models, investors can better anticipate market movements and adjust their portfolios accordingly. The combination of ARIMA, SARIMA, and machine learning models provides a robust framework for forecasting, allowing for more nuanced and accurate predictions.

Overall, the study underscores the importance of continuously updating and validating models with new data to maintain their relevance. A hybrid approach, integrating both statistical and machine learning models, offers a comprehensive solution for stock price forecasting. By adopting these recommendations, investors and portfolio managers can enhance their predictive analytics capabilities, leading to more strategic and informed financial decisions.