빅데이터사례연구 프로젝트

- Forecasting NVDA Stock Prices -

2024년 06월 21일

빅데이터사례연구

청주대학교 데이터사이언스학과

2021012800 베네딕터스 에스라 헤르노오

Table of Contents

Table of Contents **1**
1. Introduction **2**
1.1. Overview of the Project 2
1.2. Importance of the Project 2
1.3. Project Scope and Limitations 2
2. Background **3**
2.1. Introduction to NVDA (NVIDIA Corporation) 3
3. Data Collection **4**
3.1. Sources of Data 4
3.2. Data Overview 4
4. Data Preprocessing **5**
4.1. Data Formatting 5
4.2. Missing Values 5
5. Exploratory Data Analysis **6**
5.1. NVDA Stock Price Mean per Daily, Monthly, and Yearly 6
5.2. Visualizing NVDA Stock Price Trends Over Time **7**
5.3. Identifying Seasonal Trends or Patterns 7
6. Modeling **9**
6.1. Linear Regression Model 9
6.2. Auto Regressive Integrated Moving Average Model 10
7. Evaluation **11**
8. Results **13**
8.1. Forecasted NVDA Stock Prices for Future Periods 13
8.2. Interpretation of Results and Insights Gained 14
8.3. Limitations and Assumptions of the Forecasting Model 14
9. Conclusion **15**
References **16**
Appendix **17**
A. Data Source 17
A.1. Data 17
A.2. Data table 17
B. Python Code 17
B.1. Data Import and Exploration 17
B.2. Data Cleaning and Processing 18
B.3. Exploratory Data Analysis 18
B.4. Regression Model 21
B.5. ARIMA Model 22

1. Introduction

1.1. Overview of the Project

As a high-growth company focused on computing hardware and AI development, I am interested in exploring and analyzing NVIDIA Corporation (NVDA), especially their stock price since AI is growing rapidly, NVDA stock price have increased significantly since early 2024. This project provides a unique understanding into NVDA's historical stock performance and forecasts its future performance using data analytics techniques.

The approach involves comprehensive data collection from a financial database, Yahoo Finance, followed by preprocessing to ensure the data quality for the model fitting. Through Exploratory Data Analysis (EDA), I found unique patterns and trends in NVDA's stock prices, followed by Machine Learning driven models like ARIMA to construct a forecasting model.

1.2. Importance of the Project

The importance of this project is to provide useful insights and forecasts the stock price of NVIDIA Corporation (NVDA). By analyzing historical data and identifying seasonal patterns, this project aims to offer valuable information for personal decision-making processes such as investment strategies.

As NVDA continues to innovate and invest in its AI hardware and model developments, insights from this analysis will also provide useful information into the trends within the technology sectors. Understanding how NVDA's stock price correlates with technological advancements and market sentiment can offer valuable industry perspectives.

1.3. Project Scope and Limitations

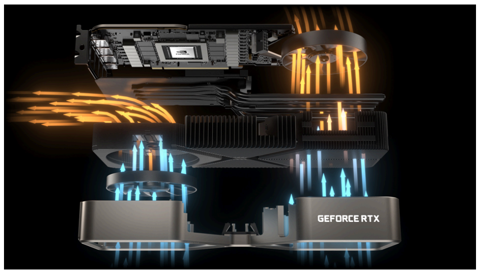
This project is conducted as a personal practice and as a final exam assignment for the 빅데이터사례연구 (Big Data Case Study) course. All analyses will be performed using Python. The results of this project are intended solely for educational purposes and to provide insight into the process of stock price forecasting. These results should not be used for actual investment strategy decisions.

2. Background

2.1. Introduction to NVDA (NVIDIA Corporation)

NVIDIA Corporation, often referred to simply as NVIDIA or NVDA (*NASDAQ: NVDA*), stands as a global leader in the field of graphics processing units (GPUs) and AI computing technologies. Founded in 1993 by Jensen Huang, Chris Malachowsky, and Curtis Priem, NVIDIA has evolved from its origins in gaming graphics to become a powerhouse driving advancements across multiple industries.

Headquartered in Santa Clara, California, NVIDIA has established itself as a cornerstone of innovation in the tech sector. The company's GPUs are integral to high-performance computing, artificial intelligence (AI), and deep learning applications, powering breakthroughs in fields ranging from healthcare and automotive to finance and scientific research.



NVIDIA's GPUs are at the heart of its success. Originally designed to render images in computer graphics and video games, these processors have proven to be exceptionally versatile. They are now critical components in data centers, supercomputers, and even personal devices, accelerating tasks that require immense computational power. For instance, GPUs are essential for training AI models, performing complex simulations, and analyzing large datasets, making them indispensable tools in modern technological advancement.

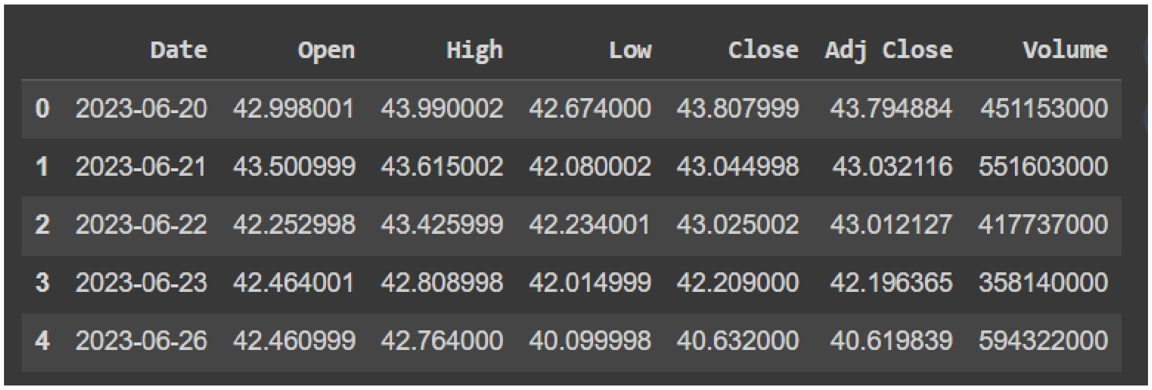
3. Data Collection

3.1. Sources of Data

The source of NVDA stock data history is from finance.yahoo.com, spanning the period from June 20, 2023 to June 19, 2024. This time frame was selected due to NVDA's substantial stock price increase of approximately 1100% since October 2022. Focusing on this specific date range ensures the data used for model fitting.

By focusing on this period, the result is expected to gain useful information about NVDA market trends. This approach aims to optimize model performance, particularly in predicting NVDA's stock prices up to seven days into the future.

3.2. Data Overview



	Date	Open	High	Low	Close	Adj Close	Volume
0	2023-06-20	42.998001	43.990002	42.674000	43.807999	43.794884	451153000
1	2023-06-21	43.500999	43.615002	42.080002	43.044998	43.032116	551603000
2	2023-06-22	42.252998	43.425999	42.234001	43.025002	43.012127	417737000
3	2023-06-23	42.464001	42.808998	42.014999	42.209000	42.196365	358140000
4	2023-06-26	42.460999	42.764000	40.099998	40.632000	40.619839	594322000

The dataset collected from finance.yahoo.com spans from June 20, 2023, to June 19, 2024, and comprises seven columns: Date, Open, High, Low, Close, Adjusted Close, and Volume. This structured dataset includes 252 entries, corresponding to daily trading sessions over the specified period.

- Date: This column denotes the trading date for each entry, ranging from June 20, 2023, to June 19, 2024.
- Open: Represents the opening price of NVDA stock at the beginning of each trading day.
- High: Indicates the highest price reached by NVDA stock during the trading day.
- Low: Denotes the lowest price reached by NVDA stock during the trading day.

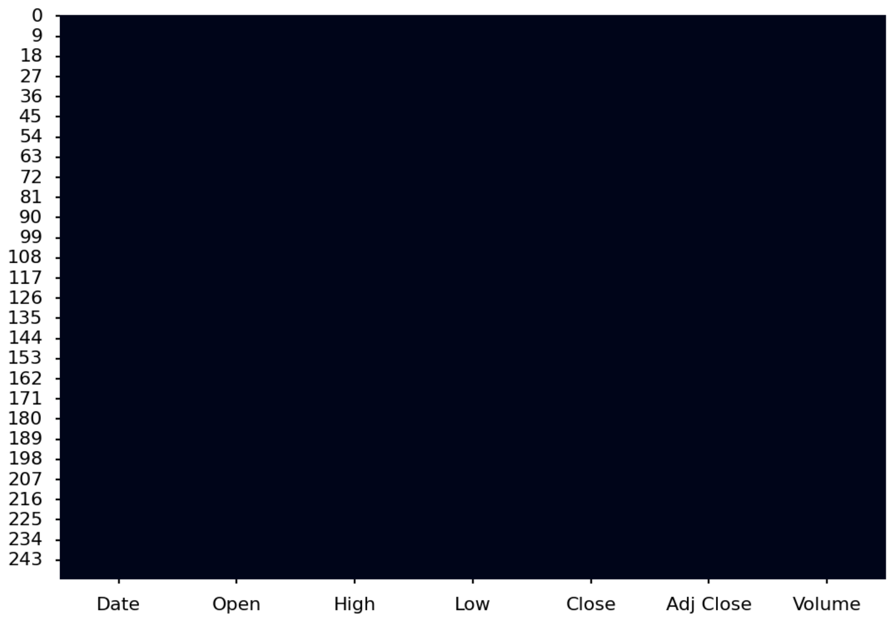
- Close: Represents the closing price of NVDA stock at the end of each trading day.
- Adjusted Close: Reflects the closing price adjusted for any corporate actions such as dividends, stock splits, or other adjustments.
- Volume: Represents the total number of NVDA shares traded during the trading day.

4. Data Preprocessing

4.1. Data Formatting

The first step involved converting the 'Date' column into the index of the dataset. By setting the date as the index using python’s ‘pandas’ package, the dataset's organization is optimized for time-series analysis. This transformation allows for straightforward access to historical data points based on specific dates, facilitating efficient modeling and analysis of NVDA's stock price trends over time.

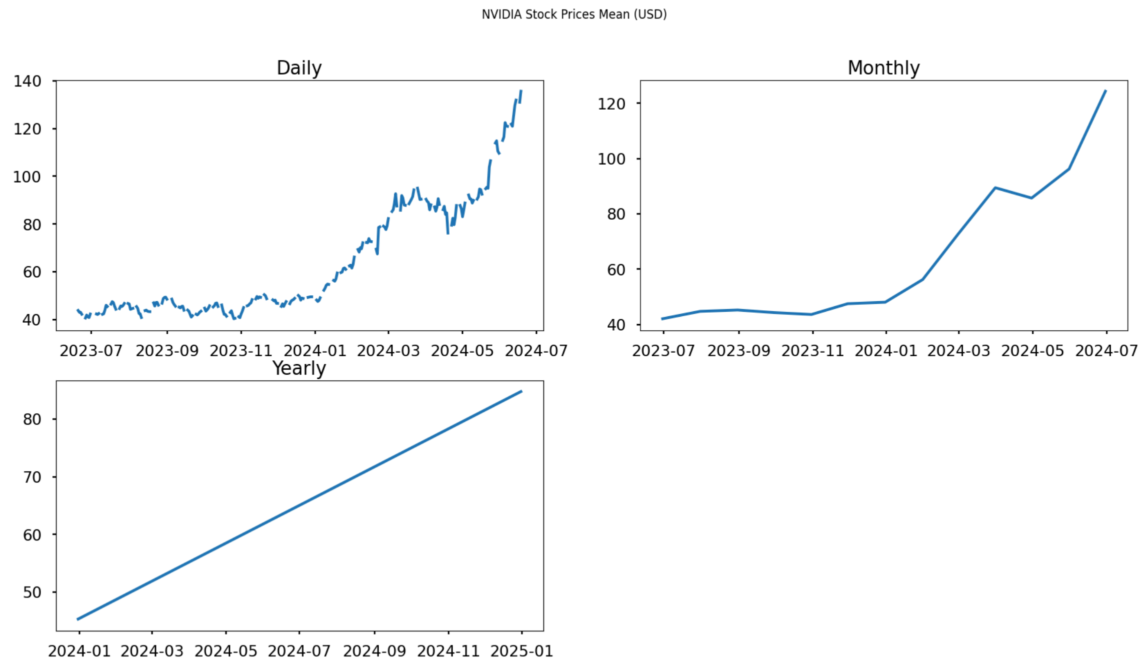
4.2. Missing Values



Upon inspection, it was found that there were no missing values present in the dataset. This aspect ensures data completeness, eliminating the need for imputation or interpolation techniques that are typically required when dealing with missing data points. The absence of missing values ensures the reliability and integrity of the dataset, enabling accurate modeling and forecasting of NVDA's stock prices without compromising data quality.

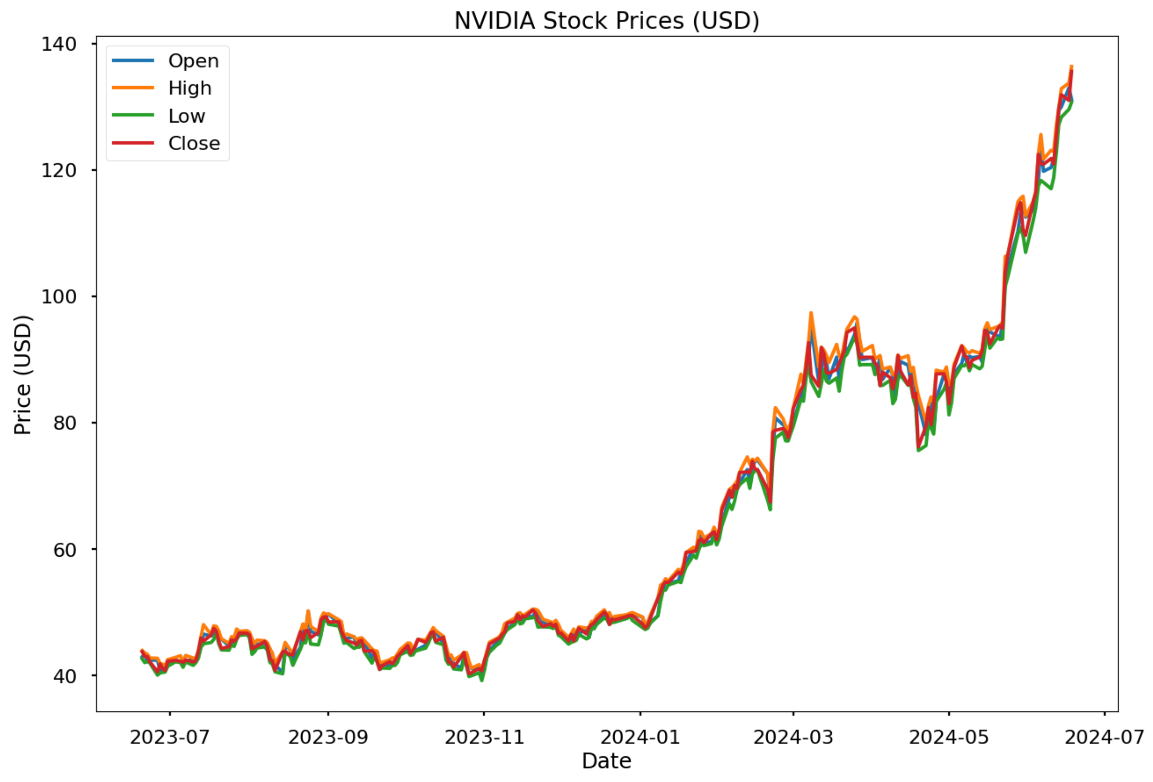
5. Exploratory Data Analysis

5.1. NVDA Stock Price Mean per Daily, Monthly, and Yearly



In this section of the report, I analyze NVIDIA Corporation's (NVDA) stock prices from June 20, 2023, to June 19, 2024, by computing the mean closing prices across daily, monthly, and yearly intervals. The dataset was resampled to aggregate daily data into monthly and yearly averages, providing insights into different time horizons of NVDA's stock performance. The plots illustrate an increase in stock price every time horizon. A trend line shows how significantly NVDA's stock price increased during the period of time.

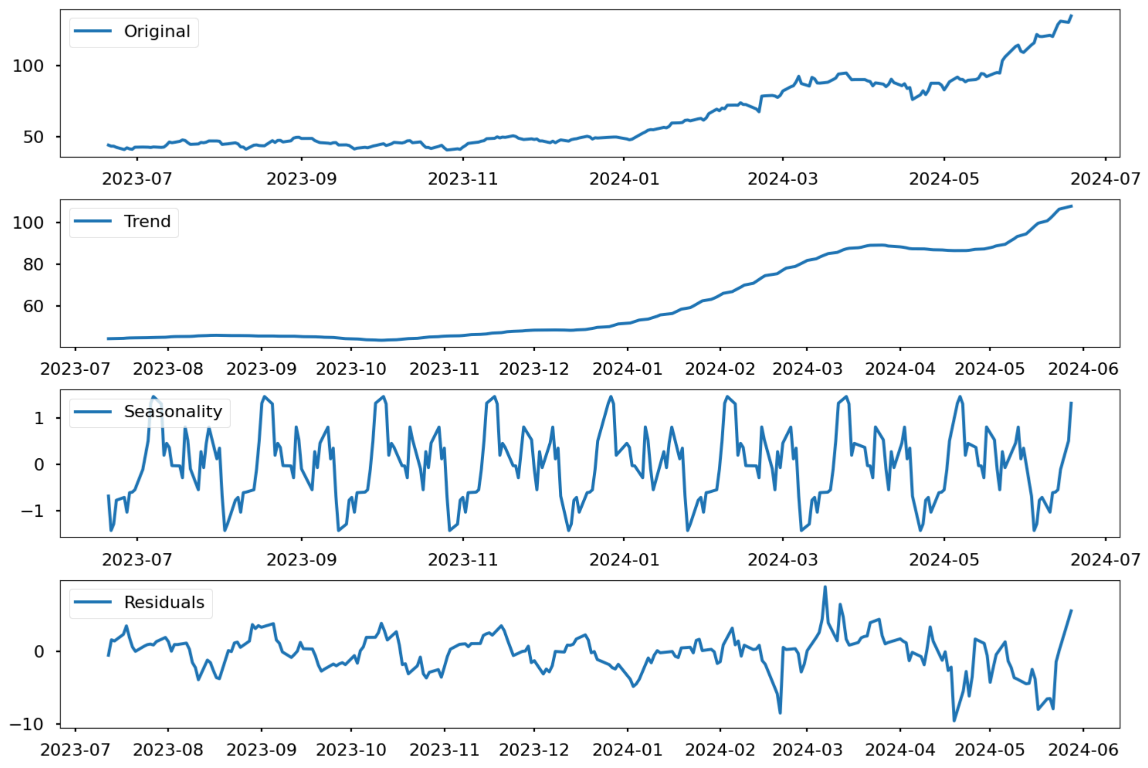
5.2. Visualizing NVDA Stock Price Trends Over Time



This section focuses on visualizing the historical trends of NVIDIA Corporation's (NVDA) stock prices from June 20, 2023, to June 19, 2024. The visualization utilizes a line plot to display NVDA's daily open, high, low, and close prices. Each line represents one of these price points over a selected period of time, offering insights into daily fluctuations and overall trends in NVDA's stock price. This depiction allows for a detailed examination of how NVDA's stock has traded throughout the year, revealing volatility, support/resistance levels, and potential patterns that may influence future price movements.

5.3. Identifying Seasonal Trends or Patterns

In this section, seasonal trends and patterns in NVIDIA Corporation's (NVDA) stock prices from June 20, 2023, to June 19, 2024, are analyzed using decomposition techniques. The dataset is decomposed into four components: original data, trend, seasonality, and residual components, each providing unique insights into NVDA's stock price dynamics.



Decomposition Analysis

The decomposition process involves breaking down the time series of NVDA's daily closing prices into:

- Original Data: The actual observed daily closing prices of NVDA stock.
- Trend Component: The long-term progression or trend in NVDA's stock prices, highlighting overall directionality.
- Seasonal Component: Patterns that repeat at fixed intervals within the data, such as daily, weekly, or monthly cycles.
- Residual Component: Variations or irregularities in the data that remain after removing the trend and seasonal components, capturing random fluctuations and unexpected events.

By analyzing this, we can clearly see that the trend from NVDA’s stock prices is rising over selected period of time and one more important things to see is that we can clearly see a seasonal pattern using Decomposition techniques that maybe we cannot see if we are using by only candlestick analysis or any other technical tools.

6. Modeling

6.1. Linear Regression Model

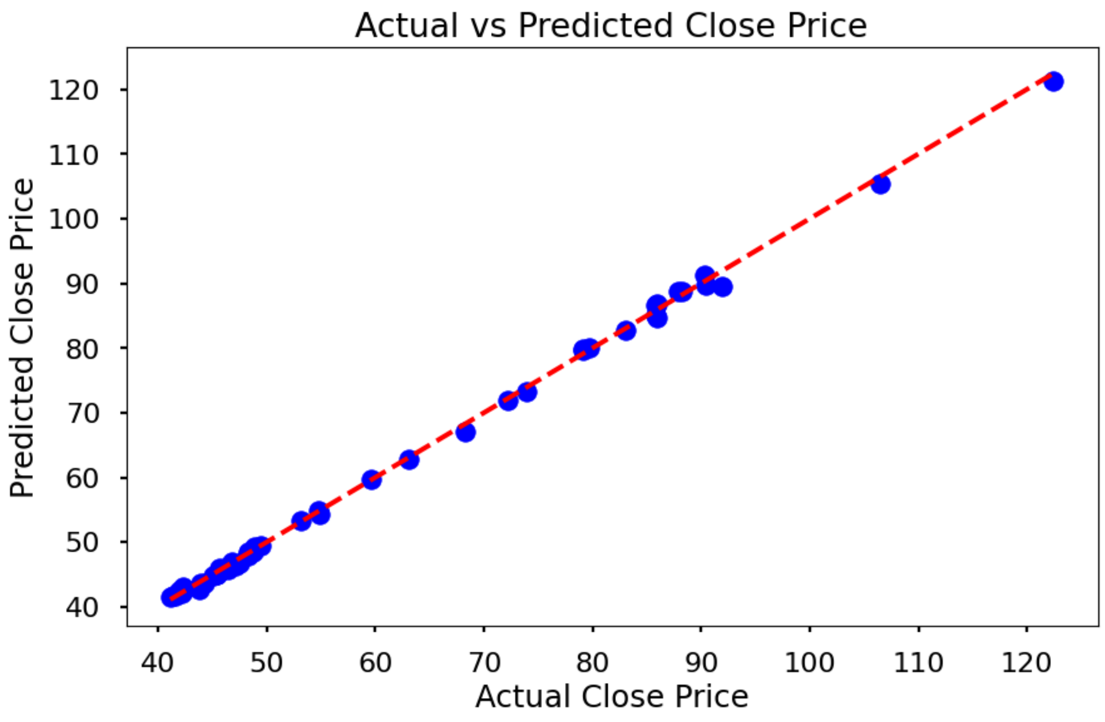
Before diving into ARIMA, a regression model was developed as a model comparison to predict NVIDIA Corporation's (NVDA) stock. The dataset was split into training (80%) and testing (20%) sets, with the training data used to fit a Linear Regression model. The regression model for predicting NVIDIA Corporation's (NVDA) daily closing stock prices using Open, High, Low, and Volume as predictors, the model can be represented as:

$$Close = \beta_0 + \beta_1 \times High + \beta_2 \times Low + \beta_3 \times Open + \beta_4 \times Volume$$

where:

• β_0 (intercept)	= - 0.394
• β_1 (Coeff for Open)	= - 6.41
• β_2 (Coeff for High)	= 0.66
• β_3 (Coeff for Low)	= 0.99
• β_3 (Coeff for Volume)	= 1.7037×10^{-10}

These coefficients provide insights into how changes in high, low, open prices, and trading volume influence the forecasted closing price of NVDA stock based on the regression model.

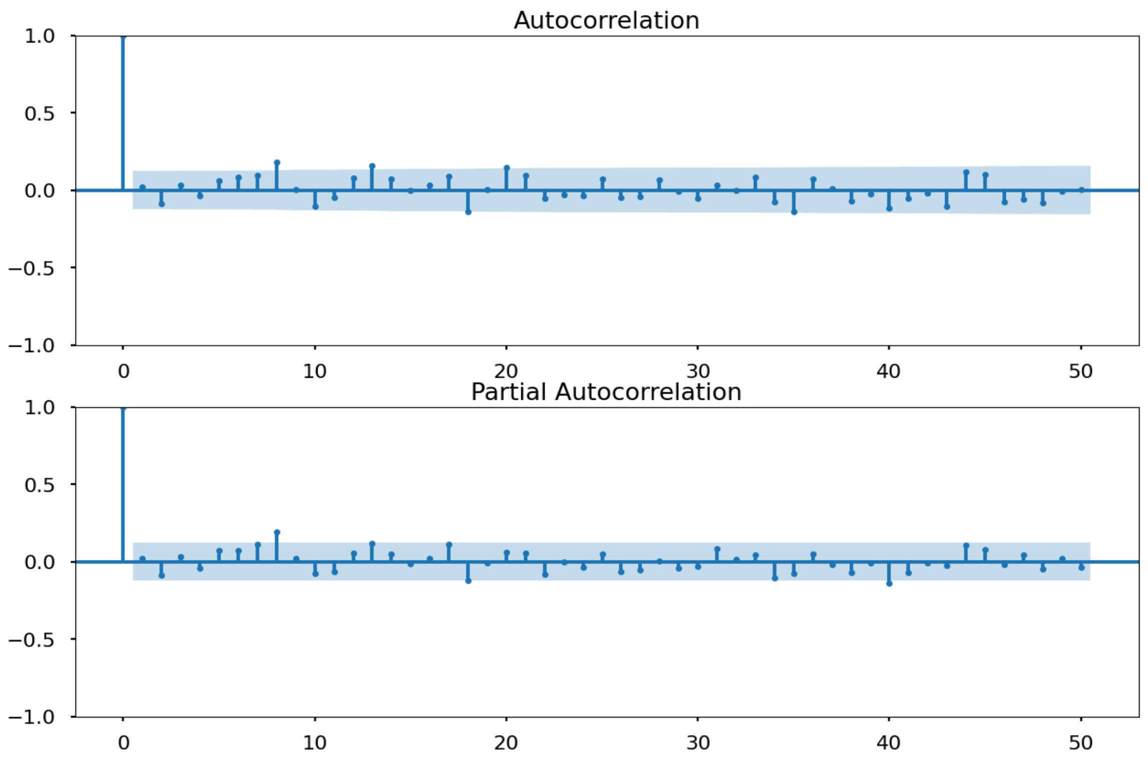


6.2. Auto Regressive Integrated Moving Average Model

Before fitting an ARIMA model, the initial step involves assessing the stationarity of the Close price data. Upon inspection, the data was found to be non-stationary, with ADF Statistics of 2.91 and p-value: 1.0, above 0.05, indicating that its mean and variance vary over time.

To address this, differencing was performed on the data. Differencing involves subtracting the previous observation from the current observation. I use first degree differencing where (*d)* is 1. This is done to remove trends and seasonality from the data. This techniques resulting ADF Statistic of -3.38 and p-value of 0.011, making it stationary

Next, I used the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots to determine the appropriate parameters for the ARIMA model. The ACF plot helps identify the number of Moving Average (MA) terms by showing the correlation of the time series with its lagged values, while the PACF plot aids in determining the number of Autoregressive (AR) terms by showing the correlation of the time series with its lagged values after removing the effects of earlier lags.



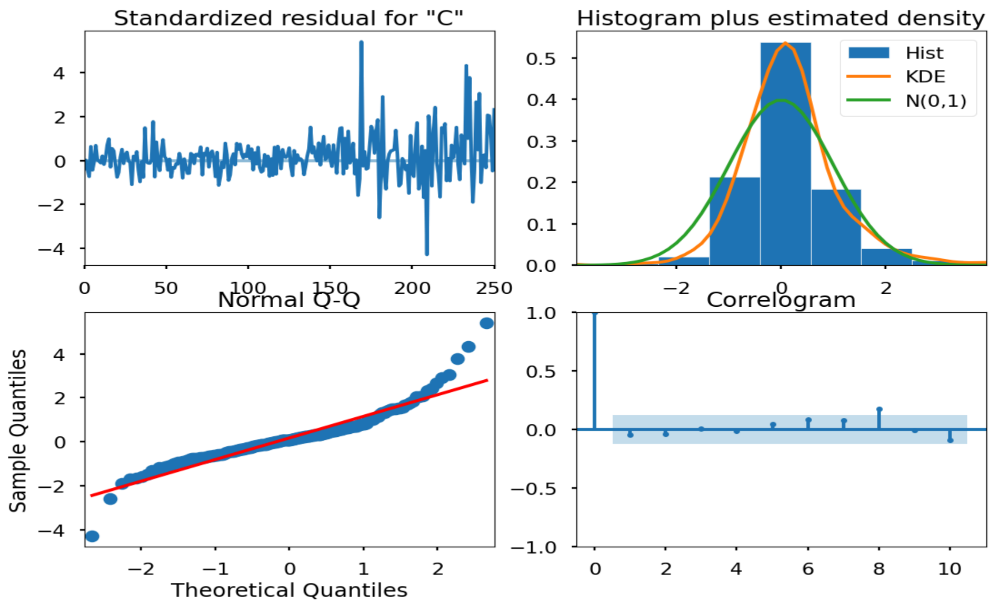
Based on the analysis of the ACF and PACF plots, the appropriate parameters for the ARIMA model were determined as follows:

- AR (p) - from PACF Plot: The PACF plot shows a significant spike at lag 1, followed by a rapid drop-off, suggesting a value of 1 for p.
- MA (q) - from ACF Plot: The ACF plot exhibits a significant spike at lag 1, followed by a rapid drop-off, indicating a value of 1 for q.
- Differencing (d): From our stationarity analysis, It is determined that the data required one differencing step to become stationary, giving d a value of 1.

Using these values, we can fit the ARIMA(1, 1, 1) model to the data. The result of the model can be summarized using python's 'statsmodels' package and provide a summarized plot insightful to determine if the model is fitted correctly.

7. Evaluation

The residual evaluation of the ARIMA(1, 1, 1) model to ensure that the model is appropriate and the residuals exhibit desirable properties.



The following four plots were used for this evaluation:

- Standardized Residuals: This plot shows that the residuals do not exhibit any discernible patterns, indicating that the model captures the underlying data structure well.
- Histogram plus Estimated Density: The histogram of the residuals follows a normal distribution, further supporting the adequacy of the model.

- Normal Q-Q Plot: This plot shows that the residuals closely follow the red line, indicating that they are normally distributed without significant anomalies.

These diagnostic plots collectively indicate that the ARIMA(1, 1, 1) model is a good fit for the data, as the residuals exhibit the expected properties of a well-fitted model.

Next step to evaluate this model is by calculating error metrics including Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Mean Squared Error (MSE).

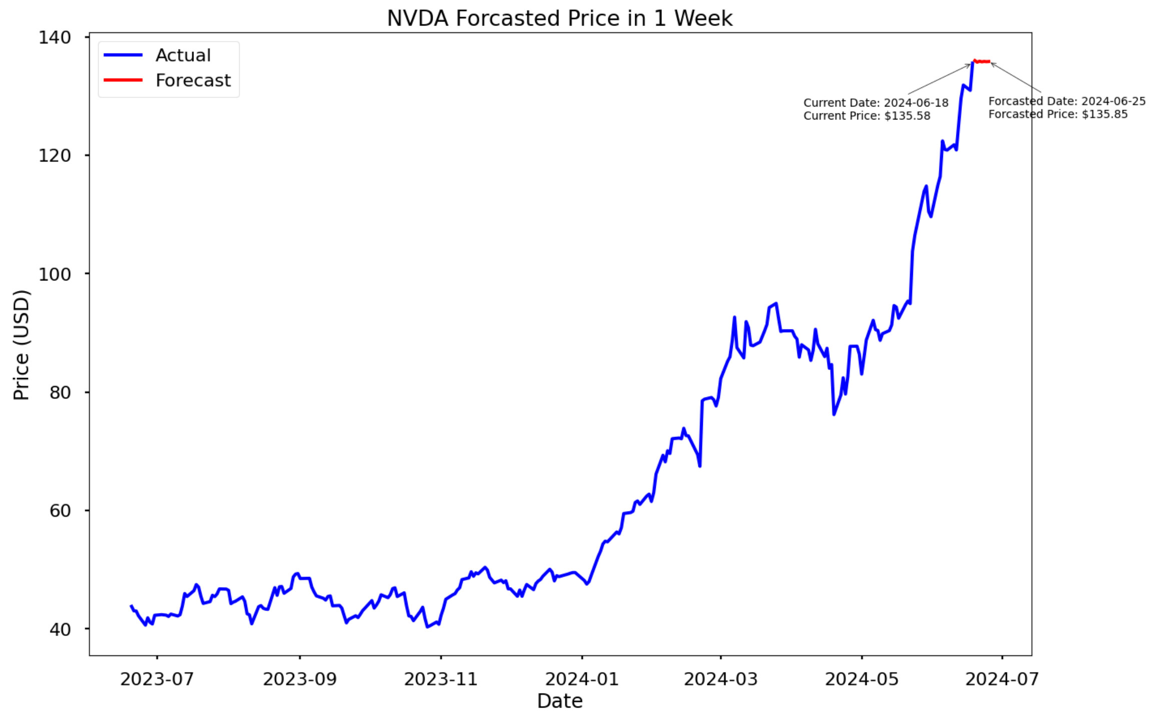
Root Mean Square Error (RMSE)
The RMSE is a measure of the differences between predicted and actual values. It is particularly useful for understanding how large the error typically is. The RMSE for our model is 76.06, suggesting that, on average, the predicted stock prices deviate from the actual prices by approximately 76.06 units.

Mean Absolute Error (MAE)
The MAE represents the average of the absolute differences between predicted and actual values. It provides a straightforward interpretation of the model's prediction error. The MAE for our model is 72.21, which means that, on average, the forecasted prices are off by 72.21 units.

Mean Squared Error (MSE)
The MSE is another measure of the differences between predicted and actual values, but it squares these differences to penalize larger errors more significantly. The MSE for our model is 5785.74.

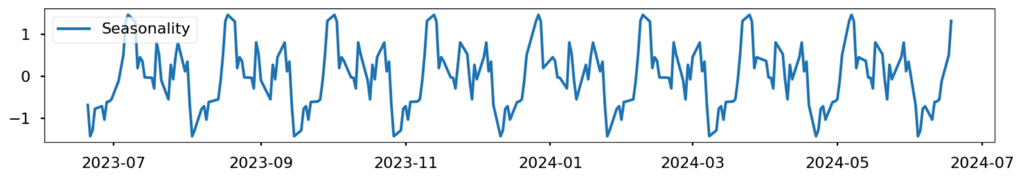
8. Results

8.1. Forecasted NVDA Stock Prices for Future Periods



For this analysis result, a plot was developed to show where the price will be going using the model. The current stock price as of June 18, 2024, is $135.58. Using the ARIMA forecasting model, it predicts that the stock price will be $135.85 in the next 7 days.This forecast suggests a slight upward trend in the immediate future.

However, our seasonality test, conducted through time series decomposition, indicates that the NVDA stock price is approaching the peak of its seasonal cycle. Historically, after reaching this peak, the stock price tends to experience a significant drop. This seasonal pattern is visually represented in the following plot:



As seen in the plot, the NVDA stock price has made a pattern similar to previous period time, showing the influence of seasonality on its movements. According to forecasted data, as per seasonality plot, it is about to hit the highest point where a significant decline will likely occur. So it will not be the best idea to take the stock around this time.

8.2. Interpretation of Results and Insights Gained

This forecast model result indicates a minor increase in the stock price, meaning the market seems stable for the next few days. However, the seasonality analysis warns us that a price drop may be coming soon. Understanding these patterns allows us to make better decisions by anticipating future price movements.

8.3. Limitations and Assumptions of the Forecasting Model

When looking at these results, it's important to remember:

1. Model Assumptions: This model is only focused on past stock prices. The actual reasons for the ups and downs of the stock price are influenced by market sentiments and other external factors. This means the model may not capture all the reasons behind price changes.
2. Data Quality: The accuracy of this model depends on the historical data that is used. Any errors or missing information in the data can affect the results.
3. External Factors: While seasonality shows regular patterns, outside influences like economic changes or new technologies can disrupt these patterns.

9. Conclusion

NVIDIA Corporation (NVDA) stock price analysis using data analysis techniques has produced significant insights. The ARIMA model used in this study has shown reasonable accuracy in predicting short-term stock price movements. Error metrics such as RMSE, MAE, and MSE show that the model's prediction fits well with the true value, thus demonstrating the model's effectiveness in this context.

In addition, seasonal analysis has revealed repeated patterns in NVDA stock prices, highlighting the importance of seasons in its movements. Stock prices usually reach their peak before they experience a significant decline, suggesting that understanding this seasonal trend is essential for making appropriate investment decisions. Therefore, investors should be careful when stock prices are nearing their peak to avoid potential losses due to subsequent declines.

However, it is important to be aware of the limitations of the forecasting models used in this analysis. This model relies heavily on historical data and does not take into account unexpected market changes or external factors. Although this model provides useful insights, it should not be the only basis for investment decision making. Including additional variables such as market sentiment, economic indicators, and industrial trends can improve model accuracy and resilience in future analyses.

In conclusion, this study provides a comprehensive understanding of NVDA's stock price trends and offers valuable insights for prospective investors. However, it is important to consider the limitations of the model and supplement it with analysis tools and other information when making investment decisions. This approach will ensure a more balanced and informed investment strategy.

References

APTЁM. (2017, November 14). *Bitcoin Price. Prediction by ARIMA*. Kaggle. https://www.kaggle.com/code/myonin/bitcoin-price-prediction-by-arima?kernelSessionId=1749542

Mishra, S. (2021, January 20). *Resample function of Pandas. Use of resample function of pandas in… | by Saloni Mishra*. Towards Data Science. Retrieved June 19, 2024, from https://towardsdatascience.com/resample-function-of-pandas-79b17ec82a78

NVIDIA Corporation (NVDA) Company Profile & Facts. (n.d.). Yahoo Finance. Retrieved June 19, 2024, from https://finance.yahoo.com/quote/NVDA/profile/

Yi, Y. (27, February 2020). *6.마케팅과 예측(forecasting) - 1.ARIMA*. youngjunyi.github.io. https://youngjunyi.github.io/analytics/2020/02/27/forecasting-in-marketing-arima.html

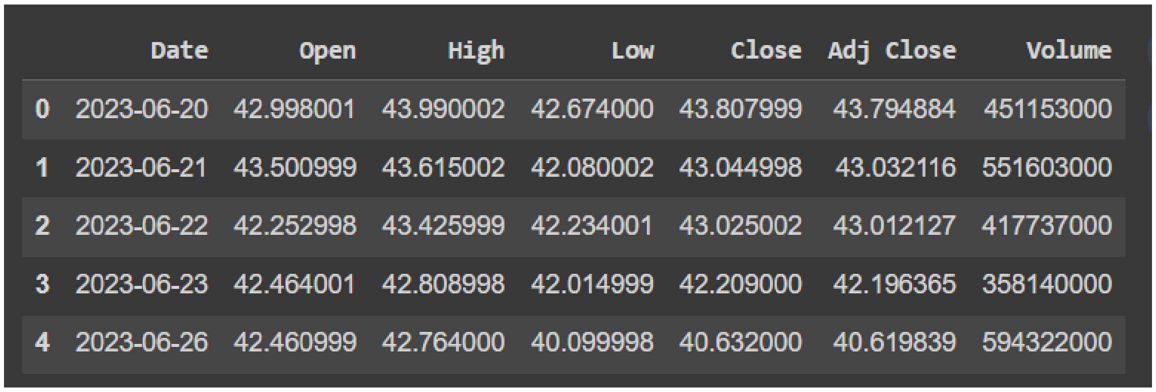
Appendix

A. Data Source

A.1. Data

https://finance.yahoo.com/quote/NVDA/history/ or NVDA.csv

A.2. Data table



	Date	Open	High	Low	Close	Adj Close	Volume
0	2023-06-20	42.998001	43.990002	42.674000	43.807999	43.794884	451153000
1	2023-06-21	43.500999	43.615002	42.080002	43.044998	43.032116	551603000
2	2023-06-22	42.252998	43.425999	42.234001	43.025002	43.012127	417737000
3	2023-06-23	42.464001	42.808998	42.014999	42.209000	42.196365	358140000
4	2023-06-26	42.460999	42.764000	40.099998	40.632000	40.619839	594322000

B. Python Code

```
# -*- coding: utf-8 -*-
"""[코드] BD_기말과제_에스라.ipynb

Automatically generated by Colab.

Original file is located at
https://colab.research.google.com/drive/1MDrDM9thuKJuGhPrVkXhVDHaVRE
fLOy6?usp=sharing
"""
```

```
B.1. Data Import and Exploration
# Packages import
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
plt.style.use('seaborn-poster')
```

```
# Data Import
df = pd.read_csv('../data/NVDA.csv', parse_dates=['Date'])
df.head(5)
df.describe()
df.info()
```

B.2. Data Cleaning and Processing

```
# Missing Values Finding
df.isna().sum()
sns.heatmap(df.isna(), cbar=False)

# Convert Date Column to Index
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)
df.head()
```

B.3. Exploratory Data Analysis

```
# Resampling Index into Day, Month, Quarter, and Year
df_daily = df.resample('D').mean()
df_monthly = df.resample('M').mean()
df_yearly = df.resample('Y').mean()

# Plotting to See Average Stock Prices each category
plt.figure(figsize=(20,10))
plt.suptitle('NVIDIA Stock Prices Mean (USD)')

plt.subplot(221)
plt.plot(df_daily.index, df_daily['Close'])
plt.title('Daily')

plt.subplot(222)
plt.plot(df_monthly.index, df_monthly['Close'])
plt.title('Monthly')

plt.subplot(223)
plt.plot(df_yearly.index, df_yearly['Close'])
plt.title('Yearly')

plt.show()

# Visualize the Time Series
plt.figure(figsize=(15,10))
for column in ['Open', 'High', 'Low', 'Close']:
  plt.plot(df.index, df[column], label=column)
plt.title('NVIDIA Stock Prices (USD)')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.legend()
plt.show()

# Decomposition to understand the patterns
from statsmodels.tsa.seasonal import seasonal_decompose

decomposition = seasonal_decompose(df['Close'], model='additive',
period=30)
trend = decomposition.trend
seasonal = decomposition.seasonal
residual = decomposition.resid

# Plotting the Decomposition
plt.figure(figsize=(15, 10))
plt.subplot(411)
plt.plot(df['Close'], label='Original')
plt.legend(loc='upper left')
plt.subplot(412)
plt.plot(trend, label='Trend')
plt.legend(loc='upper left')
plt.subplot(413)
plt.plot(seasonal, label='Seasonality')
plt.legend(loc='upper left')
plt.subplot(414)
plt.plot(residual, label='Residuals')
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```

```
# Stationary Check the data
from statsmodels.tsa.stattools import adfuller

result = adfuller(df['Close'])
print('ADF Statistic: ', result[0])
print('p-value:', result[1])
for key, value in result[4].items():
  print(key, ':', value)

if result[1] <= 0.05:
  print('Data is Stationary')
else:
  print('Data is not Stationary')

# Differencing to make the series stationary
df['Close_diff'] = df['Close'].diff()
df['Close_diff'] = df['Close_diff'].fillna(df['Close_diff'].mean())

result_diff = adfuller(df['Close_diff'])
print('ADF Statistic: ', result_diff[0])
print('p-value:', result_diff[1])
for key, value in result_diff[4].items():
  print(key, ':', value)

if result_diff[1] <= 0.05:
  print('Data is Stationary')
else:
  print('Data is not Stationary')

# Autocorrelation and Partial Autocorrelation to Determine
AutoRegresive and Inegreated for ARIMA Model
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

plt.figure(figsize=(15,10))

plt.subplot(211)
plot_acf(df['Close_diff'], lags=50, ax=plt.gca())

plt.subplot(212)
plot_pacf(df['Close_diff'], lags=50, ax=plt.gca())
```

```
plt.show()
```

```
B.4. Regression Model
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Define Predictor and Observed Value
X = df[[
    'Open', 'High', 'Low', 'Volume'
]]
y = df['Close']



# Splitting data for train and test the model
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Linear Regression Model
lm = LinearRegression().fit(X_train, y_train)
y_pred = lm.predict(X_test)

# Linear Regression Model Summary
print("Intercept:", lm.intercept_)
print("Coefficients:", lm.coef_)

# Linear Regression Model Report
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse:.2f}")
print(f"Root Mean Squared Error: {rmse:.2f}")
print(f"R-squared (Accuracy): {r2:.2%}")
```

```python
# Plotting the Model
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
ls='--', color='red')
plt.xlabel('Actual Close Price')
plt.ylabel('Predicted Close Price')
plt.title('Actual vs Predicted Close Price')
plt.show()
```

B.5. ARIMA Model

```python
# Data preparation
from statsmodels.tsa.arima.model import ARIMA

close_price = df['Close']

# Model fitting on the train data
model = ARIMA(close_price, order=(1, 1, 1))
fitted_model = model.fit()

# ARIMA summary plot
fitted_model.plot_diagnostics()
plt.show()

# Calculate error metrics
from sklearn.metrics import mean_squared_error, mean_absolute_error

pred_steps = len(close_price)
pred = fitted_model.forecast(steps=pred_steps)

rmse = np.sqrt(mean_squared_error(close_price, pred))
mae = mean_absolute_error(close_price, pred)
mse = mean_squared_error(close_price, pred)

print(f'RMSE: {rmse:.2f}')
print(f'MAE: {mae:.2f}')
print(f'MSE: {mse:.2f}')
```

```
# Forecast using model for next week price

forecast_steps = 7
forecast = fitted_model.get_forecast(steps=forecast_steps)

# Extract forecasted values and assign index
last_date = close_price.index[-1]
forecast_index = pd.date_range(start=last_date,
periods=forecast_steps + 1)
forecast_mean = forecast.predicted_mean
forecast_mean.index = forecast_index[1:]

# Plot the actual vs predicted values
plt.figure(figsize=(15, 10))
plt.plot(close_price.index,close_price, label='Actual',
color='blue')
plt.plot(forecast_mean.index, forecast_mean, label='Forecast',
color='red')
plt.title('NVDA Forcasted Price in 1 Week')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.legend()

# Annotate last Actual value on the plot
last_ac_date = close_price.index[-1]
last_ac_value = close_price.iloc[-1]
plt.annotate(f'Current Date:
{last_ac_date.strftime("%Y-%m-%d")}\nCurrent Price:
${last_ac_value:.2f}', xy=(last_ac_date, last_ac_value),
xytext=(-150, -50),
             textcoords='offset points',
arrowprops=dict(arrowstyle='->', color='black'))

# Annotate last forecasted value on the plot
last_date = forecast_mean.index[-1]
last_value = forecast_mean.iloc[-1]
plt.annotate(f'Forcasted Date:
{last_date.strftime("%Y-%m-%d")}\nForcasted Price:
${last_value:.2f}', xy=(last_date, last_value), xytext=(0, -50),
                textcoords='offset points',
arrowprops=dict(arrowstyle='->', color='black'))

plt.show()
```