SmartStock: Sentiment-Enhanced LSTM Predictions

Project Objective

This project explores the use of LSTM (Long Short-Term Memory) neural networks to predict AAPL, AMD, AMZN, INTC, META, MSFT, PYPL, and TSLA stock prices, integrating historical stock data with Google Trends data for enhanced predictive accuracy. LSTMs are particularly suited for time series forecasting due to their ability to capture long-term dependencies. By leveraging Google Trends, which provides insights into public interest and search volume for NVIDIA-related terms, the model can incorporate sentiment-driven features. The project demonstrates data preprocessing, feature engineering, and model training. This hybrid approach highlights how combining financial data with external indicators like Google Trends can improve stock price prediction accuracy.

Business Benefits of Stock Price Prediction with LSTM and Google Trends

1. **Enhanced Decision-Making**:
    - By accurately forecasting stock prices, businesses and investors can make informed decisions on buy/sell strategies, optimizing portfolio management and increasing returns.
2. **Incorporation of Public Sentiment**:
    - Using Google Trends data allows businesses to gauge market sentiment and its impact on stock movements, providing a competitive edge in understanding market dynamics.
3. **Risk Mitigation**:
    - Predictive insights help in identifying potential downturns or market volatility, enabling timely actions to minimize financial losses and manage risks effectively.

4. **Strategic Planning**:
    - Companies can align their strategies with predicted market trends, improving resource allocation and enhancing operational efficiency in trading and investment activities.

Import Libraries

In [251…
```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, r2_score
from pytrends.request import TrendReq
import yfinance as yf
from datetime import datetime
from math import sqrt
import matplotlib.pyplot as plt
import pandas_ta as ta
```

In [251…
```
import warnings
# Suppress all warnings
warnings.filterwarnings("ignore")
```

Combined Dataset

Combining the **sentiment**, **quantitative** ensures **google** across both datasets.

Sentiment Dataset

In [251…
```
df_sentiment = pd.read_csv('content/stock_tweets.csv')
```

```
# Define a function to display both head and tail of a DataFrame
def display_head_tail(data, n=5):
    # Concatenate the head and tail of the DataFrame
    combined = pd.concat([df_sentiment.head(n), df_sentiment.tail(n)])
    # Display the combined result
    return combined
# Example usage with df_combined
result = display_head_tail(df_sentiment, n=5)
# Display the result
result
```

Out[251…

	Date	Tweet	Stock Name	Company Name
0	2022-09-29 23:41:16+00:00	Mainstream media has done an amazing job at br...	TSLA	Tesla, Inc.
1	2022-09-29 23:24:43+00:00	Tesla delivery estimates are at around 364k fr...	TSLA	Tesla, Inc.
2	2022-09-29 23:18:08+00:00	3/ Even if I include 63.0M unvested RSUs as of...	TSLA	Tesla, Inc.
3	2022-09-29 22:40:07+00:00	@RealDanODowd @WholeMarsBlog @Tesla Hahaha why...	TSLA	Tesla, Inc.
4	2022-09-29 22:27:05+00:00	@RealDanODowd @Tesla Stop trying to kill kids,...	TSLA	Tesla, Inc.
80788	2021-10-07 17:11:57+00:00	Some of the fastest growing tech stocks on the...	XPEV	XPeng Inc.
80789	2021-10-04 17:05:59+00:00	With earnings on the horizon, here is a quick ...	XPEV	XPeng Inc.
80790	2021-10-01 04:43:41+00:00	Our record delivery results are a testimony of...	XPEV	XPeng Inc.
80791	2021-10-01 00:03:32+00:00	We delivered 10,412 Smart EVs in Sep 2021, rea...	XPEV	XPeng Inc.
80792	2021-09-30 10:22:52+00:00	Why can XPeng P5 deliver outstanding performan...	XPEV	XPeng Inc.

Sentiment Analysis DATA is for (2022/09/29-2021/09/30)

Analyzing Tweet Sentiment with VADER Sentiment Analysis

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a robust tool for sentiment analysis, particularly well-suited for social media text like tweets. It is a lexicon-based sentiment analysis tool designed to capture sentiment intensity, including context-aware aspects like negations, exclamations, and slang.

For this project, VADER is used to analyze tweet sentiment related to specific topics or stock performance. It categorizes sentiments as positive, negative, or neutral, assigning a compound score to indicate overall sentiment strength. This analysis helps uncover public opinion trends, which can be valuable for market sentiment evaluation or understanding user perceptions on social media.

In [251…

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
import pandas as pd
# Initialize sentiment analyzer
analyzer = SentimentIntensityAnalyzer()
# Perform sentiment analysis
df_sentiment['scores'] = df_sentiment['Tweet'].apply(lambda text: analyzer.polarity_scores(text))
df_sentiment['compound'] = df_sentiment['scores'].apply(lambda score_dict: score_dict['compound'])
df_sentiment['Date'] = pd.to_datetime(df_sentiment['Date']).dt.date
# List of stocks to include
selected_stocks = ['AAPL', 'MSFT', 'AMZN', 'META', 'TSLA', 'AMD', 'INTC', 'PYPL']
# Filter the dataset for the selected stocks
filtered_df = df_sentiment[df_sentiment['Stock Name'].isin(selected_stocks)]
# Ensure the Date column is properly formatted (strip time if present)
filtered_df['Date'] = pd.to_datetime(filtered_df['Date']).dt.date
# Group by Date and Stock Name, calculate the mean compound score
average_sentiment_by_date = (
    filtered_df.groupby(['Date', 'Stock Name'], as_index=False)['compound']
    .mean()
    .rename(columns={'compound': 'average_sentiment'})
```

```
)
# Verify no duplicates for the same stock and date
duplicates_check = average_sentiment_by_date.duplicated(subset=['Date', 'Stock Name']).sum()
print(f"Number of duplicate rows: {duplicates_check}")
# Display the result
print("Average sentiment scores by date for the selected stocks:")
print(average_sentiment_by_date)
# Example: Filter results for TSLA
stock_name = 'TSLA'
tsla_sentiment = average_sentiment_by_date[average_sentiment_by_date['Stock Name'] == stock_name]
print(f"\nAverage sentiment scores for {stock_name} by date:")
print(tsla_sentiment)
```

```
Number of duplicate rows: 0
Average sentiment scores by date for the selected stocks:
           Date Stock Name  average_sentiment
0     2021-09-30       AAPL           0.098900
1     2021-09-30        AMD           0.384217
2     2021-09-30       AMZN           0.256980
3     2021-09-30       META           0.472467
4     2021-09-30       MSFT           0.256980
...          ...        ...                ...
2552  2022-09-29       INTC          -0.827100
2553  2022-09-29       META          -0.051600
2554  2022-09-29       MSFT           0.086080
2555  2022-09-29       PYPL           0.630000
2556  2022-09-29       TSLA           0.057709

[2557 rows x 3 columns]

Average sentiment scores for TSLA by date:
           Date Stock Name  average_sentiment
5     2021-09-30       TSLA           0.251634
11    2021-10-01       TSLA           0.222412
17    2021-10-02       TSLA           0.307714
23    2021-10-03       TSLA           0.267680
30    2021-10-04       TSLA           0.151775
...          ...        ...                ...
2529  2022-09-25       TSLA           0.150072
2535  2022-09-26       TSLA           0.114828
2541  2022-09-27       TSLA           0.225460
2548  2022-09-28       TSLA           0.129845
2556  2022-09-29       TSLA           0.057709

[365 rows x 3 columns]
```

Display Sentiment data

In [251…

```
def display_head_tail(data, n=5):
```

```
    # Concatenate the head and tail of the DataFrame
    combined = pd.concat([df_sentiment.head(n), df_sentiment.tail(n)])
    # Display the combined result
    return combined
# Example usage with df_combined
result = display_head_tail(df_sentiment, n=5)
# Display the result
result
```

Out[251…

	Date	Tweet	Stock Name	Company Name	scores	compound
0	2022-09-29	Mainstream media has done an amazing job at br...	TSLA	Tesla, Inc.	{'neg': 0.125, 'neu': 0.763, 'pos': 0.113, 'co...	0.0772
1	2022-09-29	Tesla delivery estimates are at around 364k fr...	TSLA	Tesla, Inc.	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound...	0.0000
2	2022-09-29	3/ Even if I include 63.0M unvested RSUs as of...	TSLA	Tesla, Inc.	{'neg': 0.0, 'neu': 0.954, 'pos': 0.046, 'comp...	0.2960
3	2022-09-29	@RealDanODowd @WholeMarsBlog @Tesla Hahaha why...	TSLA	Tesla, Inc.	{'neg': 0.273, 'neu': 0.59, 'pos': 0.137, 'com...	-0.7568
4	2022-09-29	@RealDanODowd @Tesla Stop trying to kill kids,...	TSLA	Tesla, Inc.	{'neg': 0.526, 'neu': 0.474, 'pos': 0.0, 'comp...	-0.8750
80788	2021-10-07	Some of the fastest growing tech stocks on the...	XPEV	XPeng Inc.	{'neg': 0.0, 'neu': 0.925, 'pos': 0.075, 'comp...	0.1779
80789	2021-10-04	With earnings on the horizon, here is a quick ...	XPEV	XPeng Inc.	{'neg': 0.0, 'neu': 0.938, 'pos': 0.062, 'comp...	0.3818
80790	2021-10-01	Our record delivery results are a testimony of...	XPEV	XPeng Inc.	{'neg': 0.0, 'neu': 0.903, 'pos': 0.097, 'comp...	0.4215
80791	2021-10-01	We delivered 10,412 Smart EVs in Sep 2021, rea...	XPEV	XPeng Inc.	{'neg': 0.0, 'neu': 0.889, 'pos': 0.111, 'comp...	0.5423
80792	2021-09-30	Why can XPeng P5 deliver outstanding performan...	XPEV	XPeng Inc.	{'neg': 0.0, 'neu': 0.805, 'pos': 0.195, 'comp...	0.7783

In [252…

```
# Sort the DataFrame by Stock Name and then by Date
average_sentiment_by_date_sorted = average_sentiment_by_date.sort_values(by=['Stock Name', 'Date'])
# Save to CSV
output_file = "average_sentiment_by_date_sorted.csv"  # Update with your desired file path
average_sentiment_by_date_sorted.to_csv(output_file, index=False)
print(f"Output saved to: {output_file}")
```

```
Output saved to: average_sentiment_by_date_sorted.csv
```

In [252…
```python
def display_head_tail(data, n=5):
    # Concatenate the head and tail of the DataFrame
    combined = pd.concat([average_sentiment_by_date_sorted.head(n), average_sentiment_by_date_sorted.tai
    # Display the combined result
    return combined
# Example usage with df_combined
result = display_head_tail(average_sentiment_by_date_sorted, n=5)
# Display the result
result
```

Out[252…

	Date	Stock Name	average_sentiment
0	2021-09-30	AAPL	0.098900
6	2021-10-01	AAPL	0.248255
12	2021-10-02	AAPL	0.007525
18	2021-10-03	AAPL	0.822500
24	2021-10-04	AAPL	0.122830
2529	2022-09-25	TSLA	0.150072
2535	2022-09-26	TSLA	0.114828
2541	2022-09-27	TSLA	0.225460
2548	2022-09-28	TSLA	0.129845
2556	2022-09-29	TSLA	0.057709

In [252…
```python
df_sentiment_final = average_sentiment_by_date_sorted
df_sentiment_final.shape
```

Out[252… (2557, 3)

Stock Quantilative DATA is for (2022/09/29-2021/09/30)

In [252…

```
import yfinance as yf
import pandas as pd
def download_stock_data(ticker, start_date="2021-09-30", end_date="2022-09-29"):
    """Download stock data for a given ticker from Yahoo Finance."""
    # Download stock price data, including actions (e.g., dividends)
    data = yf.download(ticker, start=start_date, end=end_date, actions=True)
    data.columns = data.columns.get_level_values(0)  # Flatten any multi-level column structure
    data['Ticker'] = ticker  # Add a column to identify the ticker
    return data
# List of selected stocks
selected_stocks = ['AAPL', 'MSFT', 'AMZN', 'META', 'TSLA', 'AMD', 'INTC', 'PYPL']
# Initialize an empty DataFrame to store all data
df_stock_data = pd.DataFrame()
# Loop through each stock and append the data to the combined DataFrame
for stock in selected_stocks:
    stock_data = download_stock_data(stock)
    df_stock_data = pd.concat([df_stock_data, stock_data], axis=0)
# Reset the index of the combined DataFrame for consistency
df_stock_data.reset_index(inplace=True)
# Keep only the date part in the Date column
df_stock_data['Date'] = pd.to_datetime(df_stock_data['Date']).dt.date
# Display the first few rows of the combined DataFrame
df_stock_data.head()
```

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

Out [252…

Price	Date	Adj Close	Close	Dividends	High	Low	Open	Stock Splits	Volume	Ticker
0	2021-09-30	139.016602	141.500000	0.0	144.380005	141.279999	143.660004	0.0	89056700	AAPL
1	2021-10-01	140.146408	142.649994	0.0	142.919998	139.110001	141.899994	0.0	94639600	AAPL
2	2021-10-04	136.697983	139.139999	0.0	142.210007	138.270004	141.759995	0.0	98322000	AAPL
3	2021-10-05	138.633453	141.110001	0.0	142.240005	139.360001	139.490005	0.0	80861100	AAPL
4	2021-10-06	139.507812	142.000000	0.0	142.149994	138.369995	139.470001	0.0	83221100	AAPL

In [252…

```
def display_head_tail(data, n=5):
    # Concatenate the head and tail of the DataFrame
    combined = pd.concat([df_stock_data.head(n), df_stock_data.tail(n)])
    # Display the combined result
    return combined
# Example usage with df_combined
result = display_head_tail(df_stock_data, n=5)
# Display the result
result
```

Out[252…

Price	Date	Adj Close	Close	Dividends	High	Low	Open	Stock Splits	Volume	Ticker
0	2021-09-30	139.016602	141.500000	0.0	144.380005	141.279999	143.660004	0.0	89056700	AAPL
1	2021-10-01	140.146408	142.649994	0.0	142.919998	139.110001	141.899994	0.0	94639600	AAPL
2	2021-10-04	136.697983	139.139999	0.0	142.210007	138.270004	141.759995	0.0	98322000	AAPL
3	2021-10-05	138.633453	141.110001	0.0	142.240005	139.360001	139.490005	0.0	80861100	AAPL
4	2021-10-06	139.507812	142.000000	0.0	142.149994	138.369995	139.470001	0.0	83221100	AAPL
2003	2022-09-22	87.660004	87.660004	0.0	91.500000	87.099998	90.855003	0.0	12996900	PYPL
2004	2022-09-23	86.970001	86.970001	0.0	87.629997	85.680000	86.919998	0.0	12175900	PYPL
2005	2022-09-26	84.260002	84.260002	0.0	89.080002	84.129997	87.065002	0.0	16936400	PYPL
2006	2022-09-27	85.750000	85.750000	0.0	87.695000	84.580002	85.349998	0.0	11627600	PYPL
2007	2022-09-28	91.120003	91.120003	0.0	91.650002	85.559998	85.830002	0.0	14925900	PYPL

In [252…

```
df_stock_data_final = df_stock_data
df_stock_data_final.shape
```

Out[252… (2008, 10)

Google Trend DATA is for (2022/09/29-2021/09/30)

In [252…

```
# import time
# import pandas as pd
# from pytrends.request import TrendReq

# def fetch_google_trends(keyword, timeframe="2021-09-30 2022-09-29"):
#     """Fetch Google Trends data for the specified keyword."""
#     pytrends = TrendReq(hl='en-US', tz=360)
#     pytrends.build_payload([keyword], cat=0, timeframe=timeframe, geo='US', gprop='')
#     trends_data = pytrends.interest_over_time()
#     # Drop the 'isPartial' column if it exists
#     if 'isPartial' in trends_data.columns:
#         trends_data = trends_data.drop(columns=['isPartial'])
#     trends_data = trends_data.rename(columns={keyword: 'Google_Trends'})
#     trends_data['Ticker'] = keyword  # Add the Ticker column
#     return trends_data
# # List of selected keywords (stock tickers)
# keywords = ['AAPL', 'MSFT', 'AMZN', 'META', 'TSLA', 'AMD', 'INTC', 'PYPL']
# # Initialize an empty DataFrame for combined data
# combined_trends_data = pd.DataFrame()
# # Loop through each keyword and append the data
# for keyword in keywords:
#     try:
#         keyword_data = fetch_google_trends(keyword)
#         combined_trends_data = pd.concat([combined_trends_data, keyword_data], axis=0)
#     except Exception as e:
#         print(f"Failed to fetch data for {keyword}: {e}")
# # Reset the index and ensure 'Date' is a column
# combined_trends_data.reset_index(inplace=True)
# combined_trends_data.rename(columns={'date': 'Date'}, inplace=True)
# # Save to CSV
# combined_trends_data.to_csv('google_trends_data.csv', index=False)
# # Display the first few rows of the combined DataFrame
# print(combined_trends_data.head())
```

In [252…
```
df_google_trend = pd.read_csv('content/google_trends_data.csv')
# Define a function to display both head and tail of a DataFrame
def display_head_tail(data, n=5):
    # Concatenate the head and tail of the DataFrame
    combined = pd.concat([df_google_trend.head(n), df_google_trend.tail(n)])
    # Display the combined result
    return combined
# Example usage with df_combined
result = display_head_tail(df_google_trend, n=5)
# Display the result
result
```

Out[252…

	Date	Google_Trends	Ticker
0	2021-09-26	51	AAPL
1	2021-10-03	54	AAPL
2	2021-10-10	48	AAPL
3	2021-10-17	55	AAPL
4	2021-10-24	77	AAPL
419	2022-08-28	11	PYPL
420	2022-09-04	9	PYPL
421	2022-09-11	12	PYPL
422	2022-09-18	11	PYPL
423	2022-09-25	10	PYPL

In [252…
```
import pandas as pd
# Load the Google Trends data
df_google_trend = pd.read_csv('content/google_trends_data.csv')
# Convert 'Date' column to datetime format
```

```
df_google_trend['Date'] = pd.to_datetime(df_google_trend['Date'])
# Define the complete date range
start_date = "2021-09-30"
end_date = "2022-09-29"
all_dates = pd.date_range(start=start_date, end=end_date)
# Create a DataFrame with all dates for each Ticker
tickers = df_google_trend['Ticker'].unique()
complete_data = pd.DataFrame()
for ticker in tickers:
    # Filter data for the current Ticker
    ticker_data = df_google_trend[df_google_trend['Ticker'] == ticker]
    # Reindex with all dates
    ticker_data = ticker_data.set_index('Date')
    ticker_data = ticker_data.reindex(all_dates, fill_value=0)
    # Reset index and add Ticker column
    ticker_data = ticker_data.reset_index()
    ticker_data.rename(columns={'index': 'Date'}, inplace=True)
    ticker_data['Ticker'] = ticker
    # Append to the complete DataFrame
    complete_data = pd.concat([complete_data, ticker_data])
# Save the complete DataFrame
complete_data.to_csv('content/complete_google_trends_data.csv', index=False)
# Display the first few rows of the complete data
complete_data.head()
```

Out[252…

	Date	Google_Trends	Ticker
0	2021-09-30	0	AAPL
1	2021-10-01	0	AAPL
2	2021-10-02	0	AAPL
3	2021-10-03	54	AAPL
4	2021-10-04	0	AAPL

In [252…
```
df_google_trend_complete = pd.read_csv('content/complete_google_trends_data.csv')
# Define a function to display both head and tail of a DataFrame
def display_head_tail(data, n=5):
    # Concatenate the head and tail of the DataFrame
    combined = pd.concat([df_google_trend_complete.head(n), df_google_trend_complete.tail(n)])
    # Display the combined result
    return combined
# Example usage with df_combined
result = display_head_tail(df_google_trend_complete, n=5)
# Display the result
result
```

Out[252…

	Date	Google_Trends	Ticker
0	2021-09-30	0	AAPL
1	2021-10-01	0	AAPL
2	2021-10-02	0	AAPL
3	2021-10-03	54	AAPL
4	2021-10-04	0	AAPL
2915	2022-09-25	10	PYPL
2916	2022-09-26	0	PYPL
2917	2022-09-27	0	PYPL
2918	2022-09-28	0	PYPL
2919	2022-09-29	0	PYPL

In [253…
```
df_google_trend_final = df_google_trend_complete
df_google_trend_final.shape
```

Out[253… (2920, 3)

Now we have 3 different datasets - lets merge all 3

In [253…
```
print(df_google_trend_final.shape)
print(df_sentiment_final.shape)
print(df_stock_data_final.shape)
```

```
(2920, 3)
(2557, 3)
(2008, 10)
```

In [253…
```
# Rename the column "Stock Name" to "Ticker"
df_sentiment_final.rename(columns={"Stock Name": "Ticker"}, inplace=True)

# Display the updated DataFrame
print(df_sentiment_final.head())
```

```
         Date Ticker  average_sentiment
0  2021-09-30   AAPL           0.098900
6  2021-10-01   AAPL           0.248255
12 2021-10-02   AAPL           0.007525
18 2021-10-03   AAPL           0.822500
24 2021-10-04   AAPL           0.122830
```

In [253…
```
# Convert 'Date' column to datetime format
df_stock_data_final['Date'] = pd.to_datetime(df_stock_data_final['Date'])

# Define the complete date range
start_date = "2021-09-30"
end_date = "2022-09-29"
all_dates = pd.date_range(start=start_date, end=end_date)
# Create a DataFrame with all dates for each Ticker
tickers = df_sentiment_final['Ticker'].unique()
complete_data = pd.DataFrame()
for ticker in tickers:
    # Filter data for the current Ticker
    ticker_data = df_stock_data_final[df_stock_data_final['Ticker'] == ticker]
```

```
    # Reindex with all dates
    ticker_data = ticker_data.set_index('Date')
    ticker_data = ticker_data.reindex(all_dates, fill_value=0)

    # Reset index and add Ticker column
    ticker_data = ticker_data.reset_index()
    ticker_data.rename(columns={'index': 'Date'}, inplace=True)
    ticker_data['Ticker'] = ticker

    # Append to the complete DataFrame
    complete_data = pd.concat([complete_data, ticker_data])
df_stock_data_final_adjusted = complete_data

# Display the first few rows of the complete data
df_stock_data_final_adjusted.head()
```

Out [253…

Price	Date	Adj Close	Close	Dividends	High	Low	Open	Stock Splits	Volume	Ticker
0	2021-09-30	139.016602	141.500000	0.0	144.380005	141.279999	143.660004	0.0	89056700	AAPL
1	2021-10-01	140.146408	142.649994	0.0	142.919998	139.110001	141.899994	0.0	94639600	AAPL
2	2021-10-02	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0	AAPL
3	2021-10-03	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0	AAPL
4	2021-10-04	136.697983	139.139999	0.0	142.210007	138.270004	141.759995	0.0	98322000	AAPL

In [253…

```
# Save the DataFrame to a CSV file
df_stock_data_final_adjusted.to_csv('df_stock_data_final_adjusted.csv', index=False)
```

```
print("DataFrame has been saved to 'df_stock_data_final_adjusted.csv'.")
```

```
DataFrame has been saved to 'df_stock_data_final_adjusted.csv'.
```

In [253…
```
df_stock_data_final_adjusted.shape
```

Out[253… (2920, 10)

In [253…
```
# Convert 'Date' column to datetime format
df_sentiment_final['Date'] = pd.to_datetime(df_sentiment_final['Date'])

# Define the complete date range
start_date = "2021-09-30"
end_date = "2022-09-29"
all_dates = pd.date_range(start=start_date, end=end_date)
# Create a DataFrame with all dates for each Ticker
tickers = df_sentiment_final['Ticker'].unique()
complete_data = pd.DataFrame()
for ticker in tickers:
    # Filter data for the current Ticker
    ticker_data = df_sentiment_final[df_sentiment_final['Ticker'] == ticker]

    # Reindex with all dates
    ticker_data = ticker_data.set_index('Date')
    ticker_data = ticker_data.reindex(all_dates, fill_value=0)

    # Reset index and add Ticker column
    ticker_data = ticker_data.reset_index()
    ticker_data.rename(columns={'index': 'Date'}, inplace=True)
    ticker_data['Ticker'] = ticker

    # Append to the complete DataFrame
    complete_data = pd.concat([complete_data, ticker_data])
df_sentiment_final_adjusted = complete_data

# Display the first few rows of the complete data
df_sentiment_final_adjusted.head()
```

Out[253…

	Date	Ticker	average_sentiment
0	2021-09-30	AAPL	0.098900
1	2021-10-01	AAPL	0.248255
2	2021-10-02	AAPL	0.007525
3	2021-10-03	AAPL	0.822500
4	2021-10-04	AAPL	0.122830

In [253…
```
print(df_google_trend_final.shape)
print(df_sentiment_final_adjusted.shape)
print(df_stock_data_final_adjusted.shape)
```

```
(2920, 3)
(2920, 3)
(2920, 10)
```

In [253…
```
df_google_trend_final.head()
```

Out[253…

	Date	Google_Trends	Ticker
0	2021-09-30	0	AAPL
1	2021-10-01	0	AAPL
2	2021-10-02	0	AAPL
3	2021-10-03	54	AAPL
4	2021-10-04	0	AAPL

In [253…
```
df_sentiment_final_adjusted.head()
```

Out[253…

	Date	Ticker	average_sentiment
0	2021-09-30	AAPL	0.098900
1	2021-10-01	AAPL	0.248255
2	2021-10-02	AAPL	0.007525
3	2021-10-03	AAPL	0.822500
4	2021-10-04	AAPL	0.122830

In [254…

```
df_stock_data_final_adjusted.head()
```

Out[254…

Price	Date	Adj Close	Close	Dividends	High	Low	Open	Stock Splits	Volume	Ticker
0	2021-09-30	139.016602	141.500000	0.0	144.380005	141.279999	143.660004	0.0	89056700	AAPL
1	2021-10-01	140.146408	142.649994	0.0	142.919998	139.110001	141.899994	0.0	94639600	AAPL
2	2021-10-02	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0	AAPL
3	2021-10-03	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0	AAPL
4	2021-10-04	136.697983	139.139999	0.0	142.210007	138.270004	141.759995	0.0	98322000	AAPL

In [254…

```
import pandas as pd
# Ensure the DataFrames are loaded into your environment
# Example: df_stock_data_final_adjusted = pd.read_csv('df_stock_data_final_adjusted.csv')
# Example: df_google_trend_final = pd.read_csv('df_google_trend_final.csv')
# Convert 'Date' columns to datetime format for consistency
```

```
df_stock_data_final_adjusted['Date'] = pd.to_datetime(df_stock_data_final_adjusted['Date'])
df_google_trend_final['Date'] = pd.to_datetime(df_google_trend_final['Date'])
# Merge the 'Google_Trends' column from df_google_trend_final into df_stock_data_final_adjusted
df_stock_data_final_adjusted = pd.merge(
    df_stock_data_final_adjusted,
    df_google_trend_final[['Date', 'Ticker', 'Google_Trends']],
    on=['Date', 'Ticker'],
    how='left'
)
```

In [254…

```
# Display the first few rows of the updated DataFrame
df_stock_data_final_adjusted.tail()
```

Out[254…

	Date	Adj Close	Close	Dividends	High	Low	Open	Stock Splits	Volume	Ticker	Go
2915	2022-09-25	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0	TSLA	
2916	2022-09-26	276.010010	276.010010	0.0	284.089996	270.309998	271.829987	0.0	58076900	TSLA	
2917	2022-09-27	282.940002	282.940002	0.0	288.670013	277.510010	283.839996	0.0	61925200	TSLA	
2918	2022-09-28	287.809998	287.809998	0.0	289.000000	277.570007	283.079987	0.0	54664800	TSLA	
2919	2022-09-29	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0	TSLA	

In [254…

```
import pandas as pd
# Ensure the DataFrames are loaded into your environment
```

```
# Example: df_stock_data_final_adjusted = pd.read_csv('df_stock_data_final_adjusted.csv')
# Example: df_google_trend_final = pd.read_csv('df_google_trend_final.csv')
# Convert 'Date' columns to datetime format for consistency
df_stock_data_final_adjusted['Date'] = pd.to_datetime(df_stock_data_final_adjusted['Date'])
df_sentiment_final_adjusted['Date'] = pd.to_datetime(df_sentiment_final_adjusted['Date'])
# Merge the 'Google_Trends' column from df_google_trend_final into df_stock_data_final_adjusted
df_stock_data_final_adjusted = pd.merge(
    df_stock_data_final_adjusted,
    df_sentiment_final_adjusted[['Date', 'Ticker', 'average_sentiment']],
    on=['Date', 'Ticker'],
    how='left'
)
```

In [254…

```
# Display the first few rows of the updated DataFrame
df_stock_data_final_adjusted.head()
```

Out[254…

	Date	Adj Close	Close	Dividends	High	Low	Open	Stock Splits	Volume	Ticker	Google_
0	2021-09-30	139.016602	141.500000	0.0	144.380005	141.279999	143.660004	0.0	89056700	AAPL	
1	2021-10-01	140.146408	142.649994	0.0	142.919998	139.110001	141.899994	0.0	94639600	AAPL	
2	2021-10-02	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0	AAPL	
3	2021-10-03	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0	AAPL	
4	2021-10-04	136.697983	139.139999	0.0	142.210007	138.270004	141.759995	0.0	98322000	AAPL	

In [254…
```
# Drop rows where both 'Close' and 'Open' columns are 0.000000
df_stock_data_final_adjusted = df_stock_data_final_adjusted[
    ~((df_stock_data_final_adjusted['Close'] == 0.000000) &
      (df_stock_data_final_adjusted['Open'] == 0.000000))
]
```

In [254…
```
def calculate_technical_indicators(data):
#Overlap Studies
    # Daily return
 data['daily_return'] = data['Close'].pct_change()
 data['ma'] = ta.ma(data['Close'],timeperiod=10)
 data['ema'] = ta.ema(data['Close'],timeperiod=10)
 data['dema'] = ta.dema(data['Close'],timeperiod=10)
 data['kamaw'] = ta.kama(data['Close'],timeperiod=10)
 data['ma'] = ta.wma(data['Close'],timeperiod=10)
 data['midprice'] = ta.midprice(data['High'], data['Low'],timeperiod=10)

#Momentum Indicator
 adx = ta.adx(data['High'], data['Low'], data['Close'])
 data = pd.concat([data, adx], axis=1)
 data['bop'] = ta.bop(data['Open'], data['High'], data['Low'], data['Close'])
 data['cmo'] = ta.cmo(data['Close'],timeperiod=10)
 data['mfi'] = ta.mfi(data['High'], data['Low'], data['Close'], data['Volume'])
 data['RrocOC'] = ta.roc(data['Close'],timeperiod=10)
 data['willr'] = ta.willr(data['High'], data['Low'], data['Close'], timeperiod=14)

#Volume
 data['ad'] = ta.ad(data['High'], data['Low'], data['Close'], data['Volume'])
 data['obv'] = ta.obv(data['Close'], data['Volume'])

#Volatility
 data['natr'] = ta.natr(data['High'], data['Low'], data['Close'], timeperiod=14)
 data['atr'] = ta.atr(data['High'], data['Low'], data['Close'], timeperiod=14)
 data['true_range'] = ta.true_range(data['High'], data['Low'], data['Close'])

 data['rsi'] = ta.rsi(data['Close'])
```

```
# Moving Average Volatility (Standard Deviation of Returns)
# Interaction feature: Volume * Price Change
 data['volume_price_interaction'] = data['Volume'] * data['daily_return']
#MISC
 tsi = ta.tsi(data['Close'], timeperiod=14)
 data = pd.concat([data, tsi], axis=1)
# Replace inf values with NaN
 data.replace([float('inf'), float('-inf')], float('nan'), inplace=True)
 return data
```

In [254…
```
# Display the first few rows of the updated DataFrame
df_stock_data_final_adjusted = calculate_technical_indicators(df_stock_data_final_adjusted)
```

In [254…
```
df_stock_data_final_adjusted.columns
```

Out[254…
```
Index(['Date', 'Adj Close', 'Close', 'Dividends', 'High', 'Low', 'Open',
       'Stock Splits', 'Volume', 'Ticker', 'Google_Trends',
       'average_sentiment', 'daily_return', 'ma', 'ema', 'dema', 'kamaw',
       'midprice', 'ADX_14', 'DMP_14', 'DMN_14', 'bop', 'cmo', 'mfi', 'RrocOC',
       'willr', 'ad', 'obv', 'natr', 'atr', 'true_range', 'rsi',
       'volume_price_interaction', 'TSI_13_25_13', 'TSIs_13_25_13'],
      dtype='object')
```

In [254…
```
# Move 'daily_return' to the second column
cols = list(df_stock_data_final_adjusted.columns)  # Get all column names
cols.insert(1, cols.pop(cols.index("daily_return")))  # Move 'daily_return' to second position
df_stock_data_final_adjusted = df_stock_data_final_adjusted[cols]  # Reorder the columns
```

In [255…
```
# Define a function to display both head and tail of a DataFrame
def display_head_tail(data, n=10):
    # Concatenate the head and tail of the DataFrame
    combined = pd.concat([df_stock_data_final_adjusted.head(n), df_stock_data_final_adjusted.tail(n)])
    # Display the combined result
    return combined
# Example usage with df_combined
result = display_head_tail(df_stock_data_final_adjusted, n=5)
```

```
# Display the result
result
```

Out [255...

	Date	daily_return	Adj Close	Close	Dividends	High	Low	Open	Stock Splits	Volum
0	2021-09-30	NaN	139.016602	141.500000	0.0	144.380005	141.279999	143.660004	0.0	8905670
1	2021-10-01	0.008127	140.146408	142.649994	0.0	142.919998	139.110001	141.899994	0.0	9463960
4	2021-10-04	-0.024606	136.697983	139.139999	0.0	142.210007	138.270004	141.759995	0.0	9832200
5	2021-10-05	0.014158	138.633453	141.110001	0.0	142.240005	139.360001	139.490005	0.0	8086110
6	2021-10-06	0.006307	139.507812	142.000000	0.0	142.149994	138.369995	139.470001	0.0	8322110
2912	2022-09-22	-0.040592	288.589996	288.589996	0.0	301.290009	285.820007	299.859985	0.0	7054540
2913	2022-09-23	-0.045948	275.329987	275.329987	0.0	284.500000	272.820007	283.089996	0.0	6374840
2916	2022-09-26	0.002470	276.010010	276.010010	0.0	284.089996	270.309998	271.829987	0.0	5807690
2917	2022-09-27	0.025108	282.940002	282.940002	0.0	288.670013	277.510010	283.839996	0.0	6192520
2918	2022-09-28	0.017212	287.809998	287.809998	0.0	289.000000	277.570007	283.079987	0.0	5466480

10 rows × 35 columns

In [255…
```
from sklearn.preprocessing import LabelEncoder
# Initialize the LabelEncoder
label_encoder = LabelEncoder()
# Apply Label Encoding to the 'Ticker' column
df_stock_data_final_adjusted['Ticker_Label'] = label_encoder.fit_transform(df_stock_data_final_adjusted[
# Display the first few rows to verify the changes
print(df_stock_data_final_adjusted[['Ticker', 'Ticker_Label']].head())
# If you need to map labels back to the original tickers:
ticker_mapping = dict(zip(label_encoder.classes_, label_encoder.transform(label_encoder.classes_)))
print("Ticker to Label Mapping:", ticker_mapping)
```

```
  Ticker  Ticker_Label
0   AAPL             0
1   AAPL             0
4   AAPL             0
5   AAPL             0
6   AAPL             0
Ticker to Label Mapping: {'AAPL': 0, 'AMD': 1, 'AMZN': 2, 'INTC': 3, 'META': 4, 'MSFT': 5, 'PYPL': 6, 'TS
LA': 7}
```

In [255…
```
df_stock_data_final_adjusted.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2008 entries, 0 to 2918
Data columns (total 36 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Date                     2008 non-null   datetime64[ns]
 1   daily_return             2007 non-null   float64
 2   Adj Close                2008 non-null   float64
 3   Close                    2008 non-null   float64
 4   Dividends                2008 non-null   float64
 5   High                     2008 non-null   float64
 6   Low                      2008 non-null   float64
 7   Open                     2008 non-null   float64
 8   Stock Splits             2008 non-null   float64
 9   Volume                   2008 non-null   int64
```

```
 10  Ticker                    2008 non-null   object
 11  Google_Trends             2008 non-null   int64
 12  average_sentiment         2008 non-null   float64
 13  ma                        1999 non-null   float64
 14  ema                       1999 non-null   float64
 15  dema                      1999 non-null   float64
 16  kamaw                     1999 non-null   float64
 17  midprice                  2007 non-null   float64
 18  ADX_14                    1981 non-null   float64
 19  DMP_14                    1994 non-null   float64
 20  DMN_14                    1994 non-null   float64
 21  bop                       2008 non-null   float64
 22  cmo                       1994 non-null   float64
 23  mfi                       1995 non-null   float64
 24  RrocOC                    1998 non-null   float64
 25  willr                     1995 non-null   float64
 26  ad                        2008 non-null   float64
 27  obv                       2008 non-null   float64
 28  natr                      1995 non-null   float64
 29  atr                       1994 non-null   float64
 30  true_range                2007 non-null   float64
 31  rsi                       1994 non-null   float64
 32  volume_price_interaction  2007 non-null   float64
 33  TSI_13_25_13              1984 non-null   float64
 34  TSIs_13_25_13             1984 non-null   float64
 35  Ticker_Label              2008 non-null   int64
dtypes: datetime64[ns](1), float64(31), int64(3), object(1)
memory usage: 645.0+ KB
```

In [255…

```
# Remove duplicate columns from the DataFrame
df_stock_data_final_adjusted = df_stock_data_final_adjusted.loc[:, ~df_stock_data_final_adjusted.columns.
# Display the updated DataFrame
df_stock_data_final_adjusted.columns
```

Out[255…
```
Index(['Date', 'daily_return', 'Adj Close', 'Close', 'Dividends', 'High',
       'Low', 'Open', 'Stock Splits', 'Volume', 'Ticker', 'Google_Trends',
       'average_sentiment', 'ma', 'ema', 'dema', 'kamaw', 'midprice', 'ADX_14',
       'DMP_14', 'DMN_14', 'bop', 'cmo', 'mfi', 'RrocOC', 'willr', 'ad', 'obv',
       'natr', 'atr', 'true_range', 'rsi', 'volume_price_interaction',
       'TSI_13_25_13', 'TSIs_13_25_13', 'Ticker_Label'],
      dtype='object')
```

In [255…
```
df_stock_data_final_adjusted.isna().sum()
```

Out[255…
```
Date                        0
daily_return                1
Adj Close                   0
Close                       0
Dividends                   0
High                        0
Low                         0
Open                        0
Stock Splits                0
Volume                      0
Ticker                      0
Google_Trends               0
average_sentiment           0
ma                          9
ema                         9
dema                        9
kamaw                       9
midprice                    1
ADX_14                     27
DMP_14                     14
DMN_14                     14
bop                         0
cmo                        14
mfi                        13
RrocOC                     10
willr                      13
```

```
ad                          0
obv                         0
natr                       13
atr                        14
true_range                  1
rsi                        14
volume_price_interaction    1
TSI_13_25_13               24
TSIs_13_25_13              24
Ticker_Label                0
dtype: int64
```

In [255…

```python
import pandas as pd
from sklearn.impute import KNNImputer
# Drop non-numeric columns
df_stock_data_final_imp = df_stock_data_final_adjusted
# Define columns to apply imputation
columns_to_impute = ['daily_return', 'ma', 'ema', 'dema',
                     'kamaw', 'midprice', 'ADX_14', 'DMP_14', 'DMN_14',
                     'cmo', 'mfi', 'RrocOC', 'willr', 'natr', 'atr',
                     'true_range', 'rsi', 'volume_price_interaction',
                     'TSI_13_25_13', 'TSIs_13_25_13']
# Drop 'Volatility' column if it is constant or irrelevant
columns_to_impute = [col for col in columns_to_impute if df_stock_data_final_imp[col].nunique() > 1]
# Select columns for imputation
df_to_impute = df_stock_data_final_imp[columns_to_impute]
# Apply KNNImputer
imputer = KNNImputer(n_neighbors=5)
df_imputed_array = imputer.fit_transform(df_to_impute)
# Convert back to DataFrame
df_imputed = pd.DataFrame(df_imputed_array, columns=columns_to_impute, index=df_to_impute.index)
# Combine imputed data with the original DataFrame
df_stock_data_final_imp.update(df_imputed)
# Display the updated DataFrame
df_stock_data_final_imp.head()
```

Out[255…

	Date	daily_return	Adj Close	Close	Dividends	High	Low	Open	Stock Splits	Volume	…
0	2021-09-30	0.006278	139.016602	141.500000	0.0	144.380005	141.279999	143.660004	0.0	89056700	…
1	2021-10-01	0.008127	140.146408	142.649994	0.0	142.919998	139.110001	141.899994	0.0	94639600	…
4	2021-10-04	-0.024606	136.697983	139.139999	0.0	142.210007	138.270004	141.759995	0.0	98322000	…
5	2021-10-05	0.014158	138.633453	141.110001	0.0	142.240005	139.360001	139.490005	0.0	80861100	…
6	2021-10-06	0.006307	139.507812	142.000000	0.0	142.149994	138.369995	139.470001	0.0	83221100	…

5 rows × 36 columns

In [255…
```
df_stock_data_final_imp.isna().sum()
```

Out[255…
```
Date                    0
daily_return            0
Adj Close               0
Close                   0
Dividends               0
High                    0
Low                     0
Open                    0
Stock Splits            0
Volume                  0
Ticker                  0
```

```
Google_Trends               0
average_sentiment           0
ma                          0
ema                         0
dema                        0
kamaw                       0
midprice                    0
ADX_14                      0
DMP_14                      0
DMN_14                      0
bop                         0
cmo                         0
mfi                         0
RrocOC                      0
willr                       0
ad                          0
obv                         0
natr                        0
atr                         0
true_range                  0
rsi                         0
volume_price_interaction    0
TSI_13_25_13                0
TSIs_13_25_13               0
Ticker_Label                0
dtype: int64
```

In [255…

```
# Convert all column names to lowercase
df_stock_data_final_imp.columns = df_stock_data_final_imp.columns.str.lower()
# Display the updated column names
print(df_stock_data_final_imp.columns)
```

```
Index(['date', 'daily_return', 'adj close', 'close', 'dividends', 'high',
       'low', 'open', 'stock splits', 'volume', 'ticker', 'google_trends',
       'average_sentiment', 'ma', 'ema', 'dema', 'kamaw', 'midprice', 'adx_14',
       'dmp_14', 'dmn_14', 'bop', 'cmo', 'mfi', 'rrococ', 'willr', 'ad', 'obv',
       'natr', 'atr', 'true_range', 'rsi', 'volume_price_interaction',
       'tsi_13_25_13', 'tsis_13_25_13', 'ticker_label'],
      dtype='object')
```

In [255…
```
df_stock_data_final_imp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2008 entries, 0 to 2918
Data columns (total 36 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   date                     2008 non-null   datetime64[ns]
 1   daily_return             2008 non-null   float64
 2   adj close                2008 non-null   float64
 3   close                    2008 non-null   float64
 4   dividends                2008 non-null   float64
 5   high                     2008 non-null   float64
 6   low                      2008 non-null   float64
 7   open                     2008 non-null   float64
 8   stock splits             2008 non-null   float64
 9   volume                   2008 non-null   int64
 10  ticker                   2008 non-null   object
 11  google_trends            2008 non-null   int64
 12  average_sentiment        2008 non-null   float64
 13  ma                       2008 non-null   float64
 14  ema                      2008 non-null   float64
 15  dema                     2008 non-null   float64
 16  kamaw                    2008 non-null   float64
 17  midprice                 2008 non-null   float64
 18  adx_14                   2008 non-null   float64
 19  dmp_14                   2008 non-null   float64
 20  dmn_14                   2008 non-null   float64
 21  bop                      2008 non-null   float64
```

```
 22  cmo                         2008 non-null   float64
 23  mfi                         2008 non-null   float64
 24  rrococ                      2008 non-null   float64
 25  willr                       2008 non-null   float64
 26  ad                          2008 non-null   float64
 27  obv                         2008 non-null   float64
 28  natr                        2008 non-null   float64
 29  atr                         2008 non-null   float64
 30  true_range                  2008 non-null   float64
 31  rsi                         2008 non-null   float64
 32  volume_price_interaction    2008 non-null   float64
 33  tsi_13_25_13                2008 non-null   float64
 34  tsis_13_25_13               2008 non-null   float64
 35  ticker_label                2008 non-null   int64
dtypes: datetime64[ns](1), float64(31), int64(3), object(1)
memory usage: 645.0+ KB
```

Basic Data Exploration

Check Distribution Pattern for Independent Variables

Analyzing the distribution pattern of independent variables is crucial for understanding the underlying data structure. It helps identify trends, outliers, and any potential skewness in the variables. By visualizing the distributions, we can determine whether any transformations or adjustments are needed to make the data more suitable for model training.

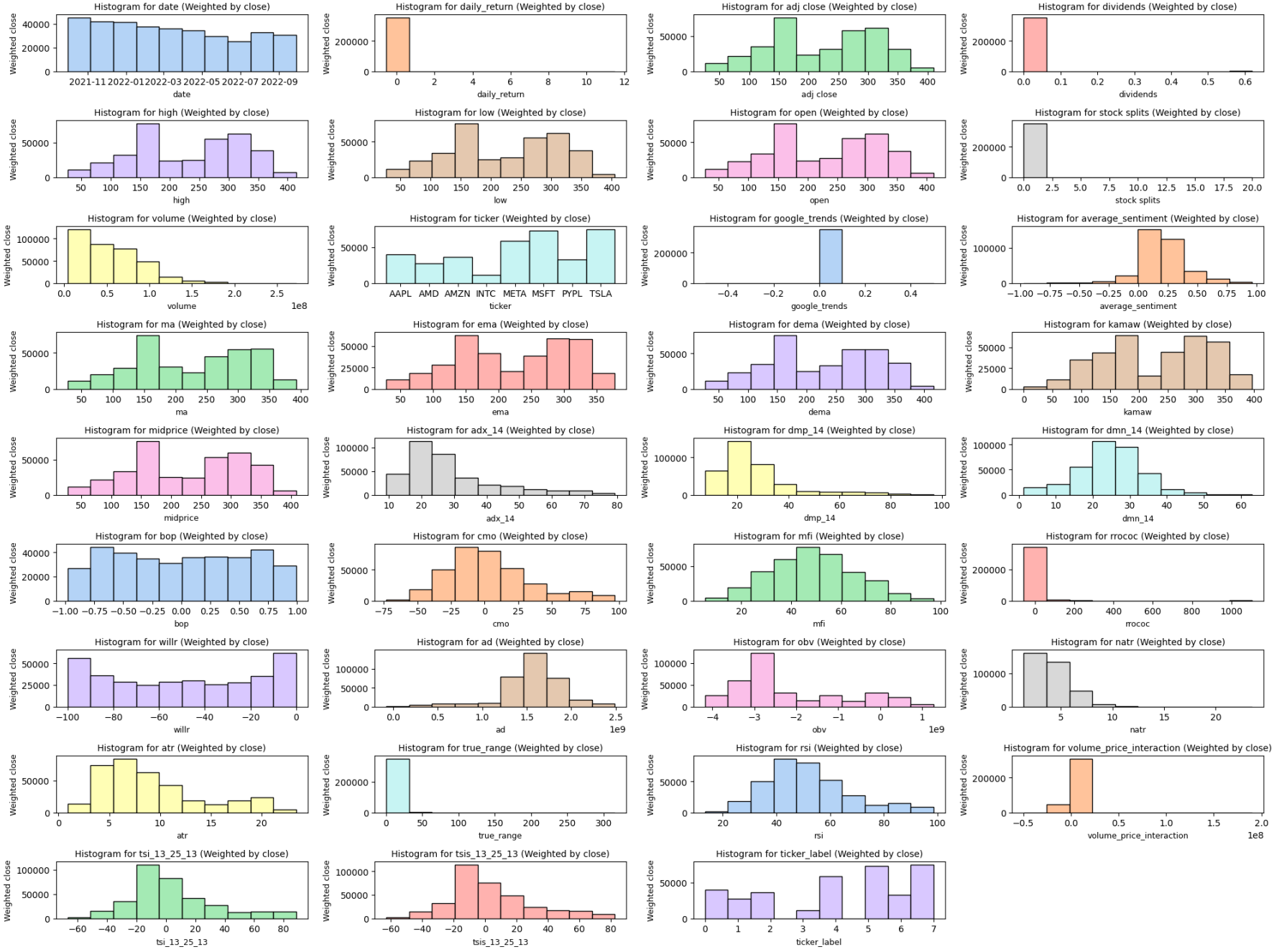
Examining the distribution of each variable provides insights into whether the data is normally distributed, if there are any extreme values (outliers), or if the data is skewed in any direction. This information is essential for making decisions on data transformations and improving model performance.

By understanding the distribution patterns, we can ensure that the dataset is well-prepared for model training, leading to better accuracy and robustness in predictions.

In [258…

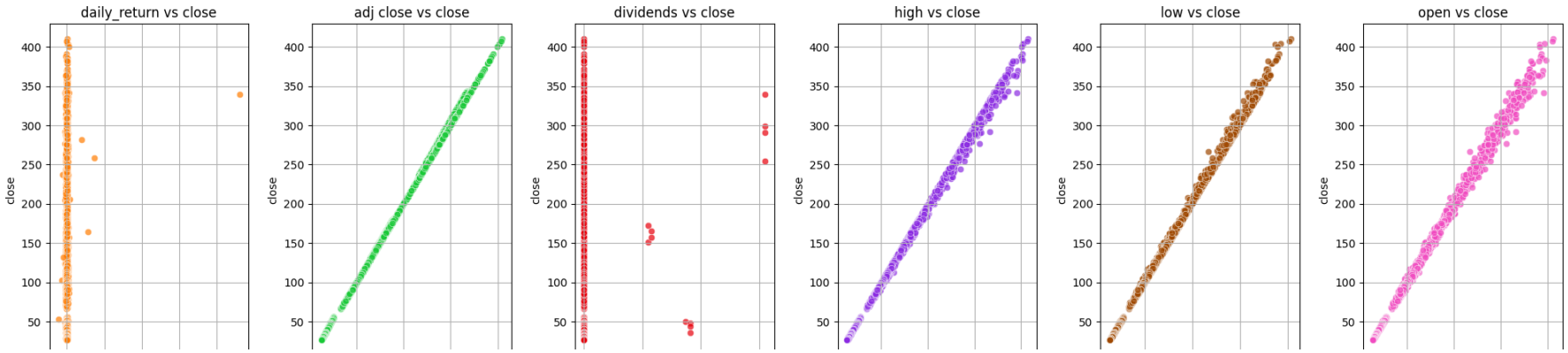
```
import matplotlib.pyplot as plt
import seaborn as sns

# List of features
features = df_stock_data_final_imp.columns.tolist()
target = 'close'

# Remove the target from the feature list
features.remove(target)

# Use Seaborn's pastel palette
colors = sns.color_palette("pastel", 10)

# Set up the number of rows and columns for subplots
n_cols = 4
n_rows = (len(features) + n_cols - 1) // n_cols  # Calculate rows required for given columns

# Create subplots
fig, axes = plt.subplots(n_rows, n_cols, figsize=(20, 15))
axes = axes.flatten()  # Flatten the axes array for easier access
# Plot each feature in a subplot
for i, feature in enumerate(features):
    sns.histplot(
        x=df_stock_data_final_imp[feature],
        weights=df_stock_data_final_imp[target],
        ax=axes[i],
        bins=10,
        color=colors[i % len(colors)],  # Cycle through the pastel colors
        alpha=0.8
    )
    axes[i].set_title(f"Histogram for {feature} (Weighted by {target})", fontsize=10)
    axes[i].set_xlabel(feature, fontsize=9)
    axes[i].set_ylabel(f"Weighted {target}", fontsize=9)
# Remove any empty subplots
for i in range(len(features), len(axes)):
    fig.delaxes(axes[i])
```

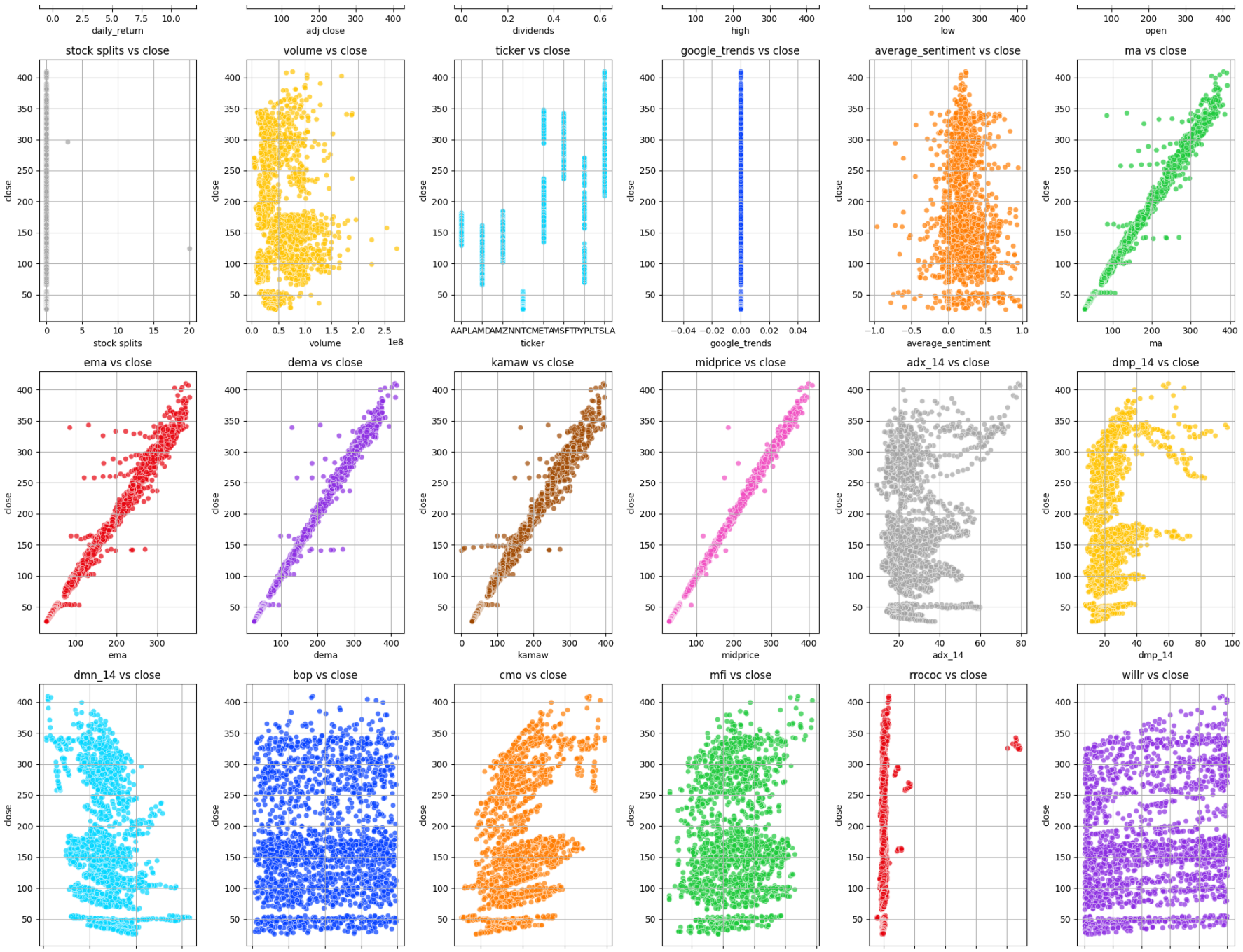
```
# Adjust the layout
plt.tight_layout()
plt.show()
```

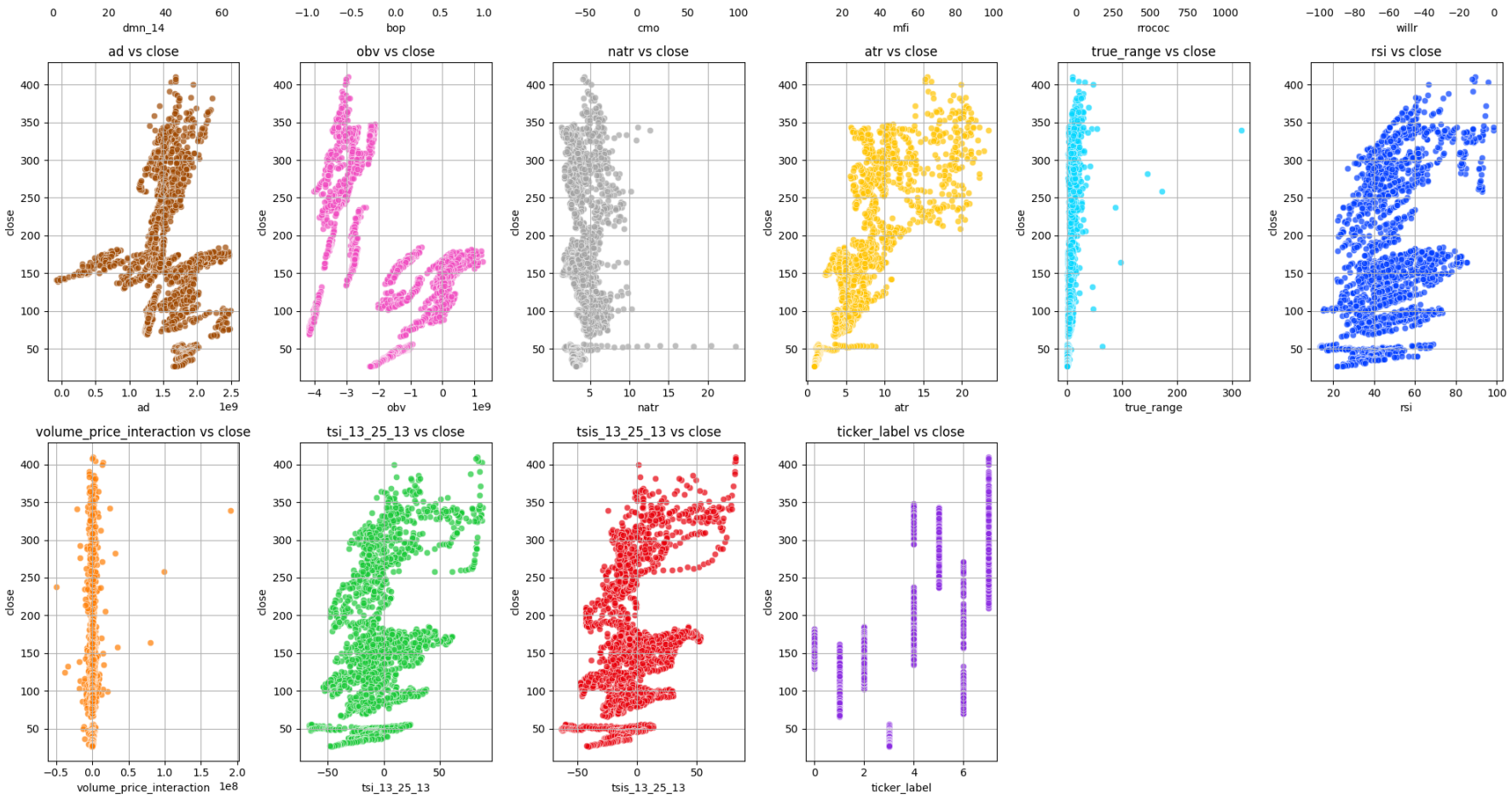

Histogram for date (Weighted by close)
Histogram for daily_return (Weighted by close)
Histogram for adj close (Weighted by close)
Histogram for dividends (Weighted by close)
Histogram for high (Weighted by close)
Histogram for low (Weighted by close)
Histogram for open (Weighted by close)
Histogram for stock splits (Weighted by close)
Histogram for volume (Weighted by close)
Histogram for ticker (Weighted by close)
Histogram for google_trends (Weighted by close)
Histogram for average_sentiment (Weighted by close)
Histogram for ma (Weighted by close)
Histogram for ema (Weighted by close)
Histogram for dema (Weighted by close)
Histogram for kamaw (Weighted by close)
Histogram for midprice (Weighted by close)
Histogram for adx_14 (Weighted by close)
Histogram for dmp_14 (Weighted by close)
Histogram for dmn_14 (Weighted by close)
Histogram for bop (Weighted by close)
Histogram for cmo (Weighted by close)
Histogram for mfi (Weighted by close)
Histogram for rrococ (Weighted by close)
Histogram for willr (Weighted by close)
Histogram for ad (Weighted by close)
Histogram for obv (Weighted by close)
Histogram for natr (Weighted by close)
Histogram for atr (Weighted by close)
Histogram for true_range (Weighted by close)
Histogram for rsi (Weighted by close)
Histogram for volume_price_interaction (Weighted by close)
Histogram for tsi_13_25_13 (Weighted by close)
Histogram for tsis_13_25_13 (Weighted by close)
Histogram for ticker_label (Weighted by close)
Weighted close
date
daily_return
adj close
dividends
high
low
open
stock splits
volume
1e8
ticker
AAPL AMD AMZN INTC META MSFT PYPL TSLA
google_trends
average_sentiment
ma
ema
dema
kamaw
midprice
adx_14
dmp_14
dmn_14
bop
cmo
mfi
rrococ
willr
ad
1e9
obv
1e9
natr
atr
true_range
rsi
volume_price_interaction
1e8
tsi_13_25_13
tsis_13_25_13
ticker_label

In [258…

```
import seaborn as sns
import matplotlib.pyplot as plt

# Define independent features and the target
Independent_features = [col for col in df_stock_data_final_imp.columns.tolist() if col not in ['close',
target = 'close'
# Create subplots for each independent feature vs Daily_Return
plt.figure(figsize=(20, 30))  # Adjust figure size to fit all plots
palette = sns.color_palette("bright")  # Use bright color palette
for i, feature in enumerate(Independent_features, start=1):
    plt.subplot(6, 6, i)  # Create a 6x2 grid for subplots
    sns.scatterplot(
        x=feature,
        y=target,
        data=df_stock_data_final_imp,
        alpha=0.7,
        color=palette[i % len(palette)]  # Cycle through bright colors
    )
    plt.title(f"{feature} vs {target}")
    plt.xlabel(feature)
    plt.ylabel(target)
    plt.grid(True)

plt.tight_layout()
plt.show()
```




daily_return
adj close
dividends
high
low
open
stock splits vs close
volume vs close
ticker vs close
google_trends vs close
average_sentiment vs close
ma vs close
ema vs close
dema vs close
kamaw vs close
midprice vs close
adx_14 vs close
dmp_14 vs close
dmn_14 vs close
bop vs close
cmo vs close
mfi vs close
rrococ vs close
willr vs close
close
stock splits
volume
1e8
ticker
AAPL AMD AMZN INTC META MSFT PYPL TSLA
google_trends
average_sentiment
ma
ema
dema
kamaw
midprice
adx_14
dmp_14



Correlation

The **correlation matrix** is a powerful tool used to measure the relationships between independent variables in a dataset. It helps identify how strongly pairs of variables are linearly related to each other. Correlation values range from **-1 to 1**:

- A value close to **1** indicates a strong positive correlation, meaning as one variable increases, the other tends to increase as well.
- A value close to **-1** indicates a strong negative correlation, where one variable increases as the other decreases.

- A value around **0** indicates little to no linear relationship between the variables.

Importance of the Correlation Matrix:

1. **Identify Multicollinearity**: If two or more independent variables are highly correlated, it can lead to multicollinearity, which can negatively impact the model's performance. High multicollinearity makes it difficult to determine the individual effect of each variable on the dependent variable.
2. **Feature Selection**: By analyzing the correlation matrix, we can detect variables that are redundant or too closely related to others. This helps in deciding which features to retain and which to drop for building a simpler and more efficient model.
3. **Insights into Relationships**: Understanding correlations helps in interpreting the underlying relationships in the data. For example, we can see how features like **temperature** and **atemp** (feels-like temperature) are closely related, which allows us to make more informed decisions on feature engineering and selection.

By using the correlation matrix effectively, we can reduce redundancy in the dataset, improve model performance, and gain insights into the relationships between variables, making it a critical step in the data preprocessing phase.

In [256…
```
df_stock_data_final_imp.head()
```

Out[256…

	date	daily_return	adj close	close	dividends	high	low	open	stock splits	volume	...
0	2021-09-30	0.006278	139.016602	141.500000	0.0	144.380005	141.279999	143.660004	0.0	89056700	...
1	2021-10-01	0.008127	140.146408	142.649994	0.0	142.919998	139.110001	141.899994	0.0	94639600	...
4	2021-10-04	-0.024606	136.697983	139.139999	0.0	142.210007	138.270004	141.759995	0.0	98322000	...
5	2021-10-05	0.014158	138.633453	141.110001	0.0	142.240005	139.360001	139.490005	0.0	80861100	...
6	2021-10-06	0.006307	139.507812	142.000000	0.0	142.149994	138.369995	139.470001	0.0	83221100	...

5 rows × 36 columns

In [256…
```
df_stock_data_final_imp.columns
```

Out[256…
```
Index(['date', 'daily_return', 'adj close', 'close', 'dividends', 'high',
       'low', 'open', 'stock splits', 'volume', 'ticker', 'google_trends',
       'average_sentiment', 'ma', 'ema', 'dema', 'kamaw', 'midprice', 'adx_14',
       'dmp_14', 'dmn_14', 'bop', 'cmo', 'mfi', 'rrococ', 'willr', 'ad', 'obv',
       'natr', 'atr', 'true_range', 'rsi', 'volume_price_interaction',
       'tsi_13_25_13', 'tsis_13_25_13', 'ticker_label'],
      dtype='object')
```

In [256…
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Compute the correlation matrix
# Drop 'Ticker' and 'Date' columns from the DataFrame
df_stock_data_final_imp_cleaned = df_stock_data_final_imp.drop(columns=['ticker'])
correlation_matrix = df_stock_data_final_imp_cleaned.corr()
# Display the correlation matrix as a table (rounded to 2 decimal places)
print("Correlation Matrix:")
print(correlation_matrix.round(2))
# Plot the heatmap for the correlation matrix
plt.figure(figsize=(20, 12))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm', vmin=-1, vmax=1, cbar=True)
# Set the title for the heatmap
plt.title('Correlation Matrix Heatmap', fontsize=16)
# Display the heatmap
plt.tight_layout()
plt.show()
```

```
Correlation Matrix:
                        date  daily_return  adj close  close  dividends  \
date                    1.00         -0.05      -0.28  -0.28      -0.00
daily_return           -0.05          1.00       0.05   0.05      -0.01
adj close              -0.28          0.05       1.00   1.00       0.01
close                  -0.28          0.05       1.00   1.00       0.01
dividends              -0.00         -0.01       0.01   0.01       1.00
high                   -0.28          0.05       1.00   1.00       0.01
low                    -0.28          0.05       1.00   1.00       0.02
open                   -0.28          0.05       1.00   1.00       0.01
stock splits            0.02          0.00      -0.01  -0.01      -0.00
volume                  0.02         -0.03      -0.03  -0.04      -0.03
google_trends            NaN           NaN        NaN    NaN        NaN
average_sentiment      -0.08          0.04       0.09   0.08      -0.03
ma                     -0.27         -0.03       0.99   0.99       0.02
ema                    -0.26         -0.03       0.98   0.98       0.02
dema                   -0.28         -0.01       0.99   0.99       0.01
kamaw                  -0.26         -0.01       0.99   0.99       0.01
midprice               -0.28          0.00       1.00   1.00       0.02
adx_14                 -0.50          0.03       0.21   0.21       0.02
dmp_14                 -0.31          0.18       0.33   0.33      -0.00
```

dmn_14	0.08	-0.10	-0.32	-0.32	-0.01
bop	0.01	0.07	0.03	0.03	-0.00
cmo	-0.22	0.13	0.39	0.39	0.01
mfi	-0.03	0.02	0.15	0.15	0.01
rrococ	-0.16	0.31	0.16	0.16	-0.01
willr	-0.15	0.10	0.22	0.22	-0.00
ad	0.12	0.01	-0.05	-0.05	-0.01
obv	-0.10	-0.01	-0.38	-0.38	0.00
natr	-0.04	0.11	-0.03	-0.04	-0.05
atr	-0.20	0.08	0.78	0.77	-0.03
true_range	-0.13	0.71	0.37	0.37	-0.02
rsi	-0.22	0.13	0.39	0.39	0.01
volume_price_interaction	-0.06	0.84	0.06	0.06	-0.01
tsi_13_25_13	-0.21	0.08	0.45	0.45	0.01
tsis_13_25_13	-0.19	-0.02	0.48	0.48	0.01
ticker_label	-0.00	0.01	0.51	0.51	-0.00

	high	low	open	stock splits	volume	...	ad	\
date	-0.28	-0.28	-0.28	0.02	0.02	...	0.12	
daily_return	0.05	0.05	0.05	0.00	-0.03	...	0.01	
adj close	1.00	1.00	1.00	-0.01	-0.03	...	-0.05	
close	1.00	1.00	1.00	-0.01	-0.04	...	-0.05	
dividends	0.01	0.02	0.01	-0.00	-0.03	...	-0.01	
high	1.00	1.00	1.00	-0.01	-0.03	...	-0.05	
low	1.00	1.00	1.00	-0.01	-0.05	...	-0.06	
open	1.00	1.00	1.00	-0.01	-0.04	...	-0.06	
stock splits	-0.01	-0.01	-0.01	1.00	0.05	...	0.01	
volume	-0.03	-0.05	-0.04	0.05	1.00	...	-0.03	
google_trends	NaN	NaN	NaN	NaN	NaN	...	NaN	
average_sentiment	0.08	0.08	0.08	0.00	0.02	...	0.09	
ma	0.99	0.99	0.99	-0.01	-0.03	...	-0.07	
ema	0.99	0.98	0.99	-0.01	-0.03	...	-0.07	
dema	0.99	0.99	0.99	-0.01	-0.04	...	-0.06	
kamaw	0.99	0.99	0.99	-0.01	-0.03	...	-0.05	
midprice	1.00	1.00	1.00	-0.01	-0.04	...	-0.06	
adx_14	0.21	0.21	0.21	0.01	-0.09	...	-0.04	
dmp_14	0.33	0.34	0.33	0.01	0.02	...	0.04	

```
dmn_14                    -0.31 -0.32 -0.31         -0.01   -0.00  ...  0.01
bop                        0.01  0.01 -0.01         -0.01    0.02  ...  0.00
cmo                        0.38  0.39  0.38          0.01    0.03  ... -0.01
mfi                        0.15  0.15  0.14          0.02    0.06  ...  0.02
rrococ                     0.15  0.16  0.15          0.00   -0.06  ...  0.01
willr                      0.21  0.21  0.20          0.03    0.01  ... -0.00
ad                        -0.05 -0.06 -0.06          0.01   -0.03  ...  1.00
obv                       -0.39 -0.38 -0.38          0.01    0.50  ... -0.05
natr                      -0.02 -0.04 -0.03          0.00    0.23  ...  0.13
atr                        0.78  0.76  0.77         -0.00    0.14  ...  0.07
true_range                 0.38  0.36  0.37         -0.00    0.17  ...  0.03
rsi                        0.38  0.39  0.38          0.01    0.03  ... -0.01
volume_price_interaction   0.05  0.05  0.04          0.01   -0.04  ...  0.02
tsi_13_25_13               0.44  0.45  0.45         -0.00    0.04  ...  0.01
tsis_13_25_13              0.48  0.48  0.48         -0.02    0.07  ...  0.00
ticker_label               0.51  0.50  0.51         -0.01   -0.41  ...  0.13

                           obv  natr   atr  true_range   rsi  \
date                      -0.10 -0.04 -0.20       -0.13 -0.22
daily_return              -0.01  0.11  0.08        0.71  0.13
adj close                 -0.38 -0.03  0.78        0.37  0.39
close                     -0.38 -0.04  0.77        0.37  0.39
dividends                  0.00 -0.05 -0.03       -0.02  0.01
high                      -0.39 -0.02  0.78        0.38  0.38
low                       -0.38 -0.04  0.76        0.36  0.39
open                      -0.38 -0.03  0.77        0.37  0.38
stock splits               0.01  0.00 -0.00       -0.00  0.01
volume                     0.50  0.23  0.14        0.17  0.03
google_trends               NaN   NaN   NaN         NaN   NaN
average_sentiment          0.08 -0.02  0.02        0.03  0.08
ma                        -0.38 -0.04  0.76        0.32  0.32
ema                       -0.38 -0.03  0.76        0.32  0.29
dema                      -0.37 -0.05  0.76        0.32  0.37
kamaw                     -0.40 -0.00  0.79        0.34  0.30
midprice                  -0.38 -0.04  0.77        0.34  0.37
adx_14                    -0.10 -0.01  0.12        0.04  0.27
dmp_14                     0.06 -0.10  0.19        0.18  0.90
```

dmn_14	-0.07	0.32	-0.08	-0.06	-0.91
bop	0.05	-0.07	-0.01	-0.06	0.23
cmo	0.08	-0.25	0.15	0.11	1.00
mfi	0.13	-0.20	0.01	-0.02	0.68
rrococ	-0.02	0.15	0.20	0.23	0.36
willr	0.09	-0.21	0.06	0.02	0.78
ad	-0.05	0.13	0.07	0.03	-0.01
obv	1.00	-0.22	-0.46	-0.21	0.08
natr	-0.22	1.00	0.50	0.37	-0.25
atr	-0.46	0.50	1.00	0.50	0.15
true_range	-0.21	0.37	0.50	1.00	0.11
rsi	0.08	-0.25	0.15	0.11	1.00
volume_price_interaction	-0.00	0.07	0.07	0.61	0.22
tsi_13_25_13	0.07	-0.25	0.19	0.12	0.90
tsis_13_25_13	0.08	-0.27	0.18	0.07	0.72
ticker_label	-0.94	0.23	0.60	0.27	0.02

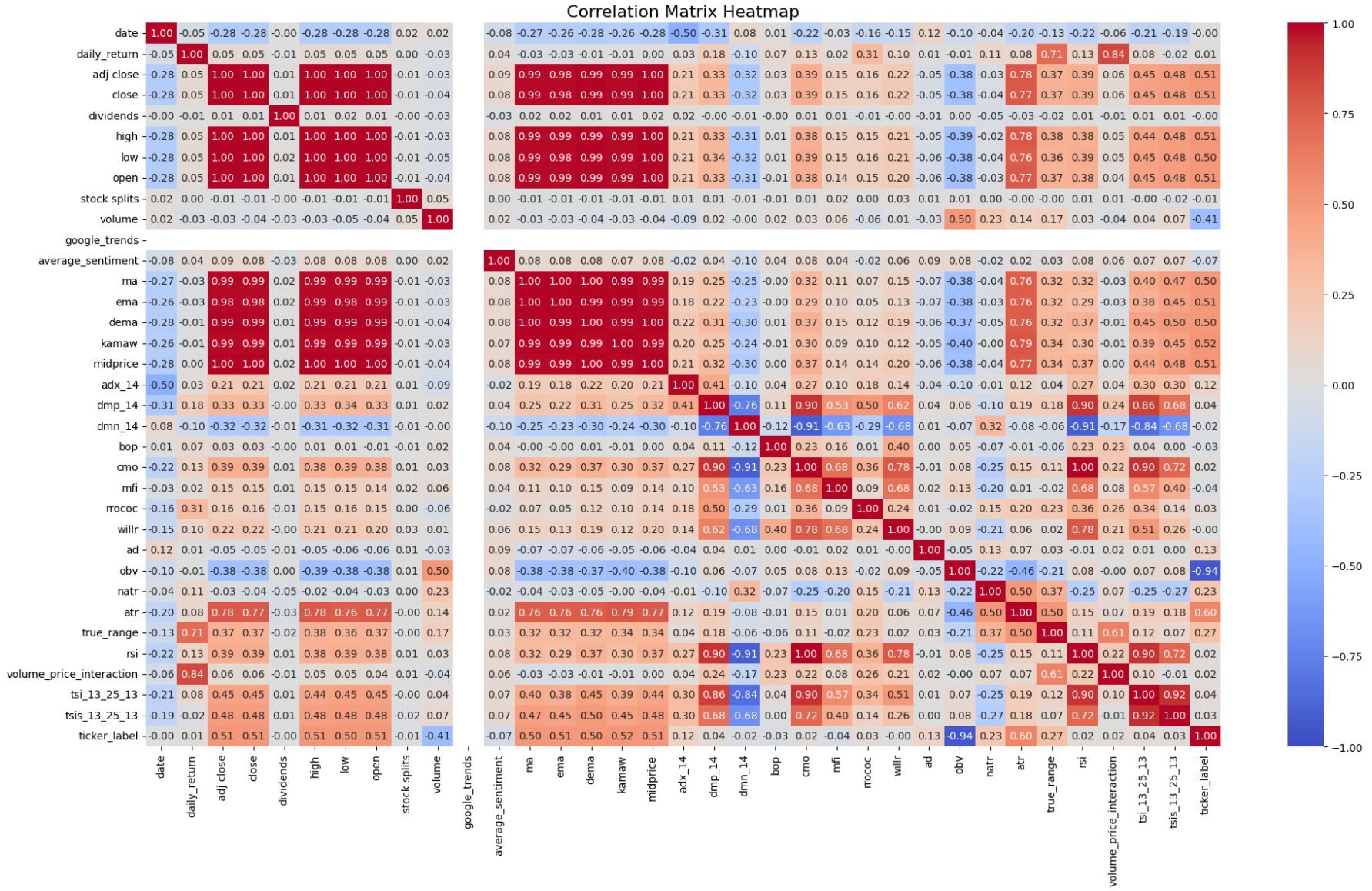
	volume_price_interaction	tsi_13_25_13	\
date	-0.06	-0.21	
daily_return	0.84	0.08	
adj close	0.06	0.45	
close	0.06	0.45	
dividends	-0.01	0.01	
high	0.05	0.44	
low	0.05	0.45	
open	0.04	0.45	
stock splits	0.01	-0.00	
volume	-0.04	0.04	
google_trends	NaN	NaN	
average_sentiment	0.06	0.07	
ma	-0.03	0.40	
ema	-0.03	0.38	
dema	-0.01	0.45	
kamaw	-0.01	0.39	
midprice	0.00	0.44	
adx_14	0.04	0.30	
dmp_14	0.24	0.86	

dmn_14	-0.17	-0.84
bop	0.23	0.04
cmo	0.22	0.90
mfi	0.08	0.57
rrococ	0.26	0.34
willr	0.21	0.51
ad	0.02	0.01
obv	-0.00	0.07
natr	0.07	-0.25
atr	0.07	0.19
true_range	0.61	0.12
rsi	0.22	0.90
volume_price_interaction	1.00	0.10
tsi_13_25_13	0.10	1.00
tsis_13_25_13	-0.01	0.92
ticker_label	0.02	0.04

	tsis_13_25_13	ticker_label
date	-0.19	-0.00
daily_return	-0.02	0.01
adj close	0.48	0.51
close	0.48	0.51
dividends	0.01	-0.00
high	0.48	0.51
low	0.48	0.50
open	0.48	0.51
stock splits	-0.02	-0.01
volume	0.07	-0.41
google_trends	NaN	NaN
average_sentiment	0.07	-0.07
ma	0.47	0.50
ema	0.45	0.51
dema	0.50	0.50
kamaw	0.45	0.52
midprice	0.48	0.51
adx_14	0.30	0.12
dmp_14	0.68	0.04

dmn_14	-0.68	-0.02
bop	0.00	-0.03
cmo	0.72	0.02
mfi	0.40	-0.04
rrococ	0.14	0.03
willr	0.26	-0.00
ad	0.00	0.13
obv	0.08	-0.94
natr	-0.27	0.23
atr	0.18	0.60
true_range	0.07	0.27
rsi	0.72	0.02
volume_price_interaction	-0.01	0.02
tsi_13_25_13	0.92	0.04
tsis_13_25_13	1.00	0.03
ticker_label	0.03	1.00

[35 rows x 35 columns]



Key Observations and Recommendations

Key Observations

1. Strong Positive Correlations

Feature Group	Correlation with Price Features	Insights
`adj close`, `close`, `high`, `low`, `open`	≈ 1.00	Highly interdependent; redundant information, select one representative feature like `close`.

| `daily_return` | 0.73 | Strongly influenced by price-related features.
| `ma`, `ema` | ≈ 1 | Highly correlated , we can drop one | `sma_20` (20-day moving average) | 0.54 | Valuable for capturing medium-term trends. | | `volume` and `rsi_volume` | ≈ 0.46 | Volume-based metrics contribute to price movements. |

2. Negative Correlations

Feature	Correlation with Price Features	Insights
`google_trends`	-0.40	Suggests search trends increase as prices decline, indicating sentiment-related insights.
`day_of_week`	-0.53	Weekly patterns impact market activity.
`is_weekend`	-0.68	Weekend effect on stock price trends.
`volume_price_tatio`	-0.50	Potential for detecting overbought or oversold conditions.
`weighted_sentiment`	-0.24	Weak negative correlation, but may add sentiment context.

3. Weak or Insignificant Correlations

Feature	Correlation	Insights

`dividends`	≈ 0.04	Insignificant for short-term predictions.
`stock_splits`	≈ 0.00	Little predictive value.
`7d_max_volume`	≈ 0.00	Low relevance for stock price predictions.

Recommendations

1. Feature Selection

- **Drop** redundant features: `high`, `low`, and `open` (retain `close` or `adj close` as a representative).
- **Drop** redundant features: `ma`, `ema`, retain one
- **Exclude** features with weak correlations: `dividends`, `stock_splits`, and `7d_max_volume`.
- **Retain** features with meaningful dynamics: `daily_return`, `sma_20`, `rsi_volume`, and `volume_price_tatio`.

2. Feature Transformation

- Create **lagged features** for `google_trends`, `volume`, and `daily_return` to capture temporal dependencies.
- Use **moving averages** for `volume` and `weighted_sentiment` to smooth short-term volatility and highlight trends.

3. Sentiment Features

- Investigate the relationship between `google_trends` and stock prices. Combine with `average_sentiment` or `weighted_sentiment` to uncover hidden patterns.

4. Time-Based Features

- Retain **`day_of_week`** and **`is_weekend`** to capture weekly seasonality effects in stock price movements.

5. Dimensionality Reduction

- Apply **PCA** to combine highly correlated features (e.g., price-related features) into fewer components while retaining variance.

6. Multicollinearity Handling

- Use **Ridge Regression** or **Lasso Regression** to mitigate multicollinearity caused by strongly correlated features.

Conclusion

Focus on retaining features with strong or moderate correlations while engineering temporal and aggregated features (e.g., moving averages, lagged features). Remove features with weak correlations to simplify the model and improve interpretability. Sentiment-based features (`google_trends` and `weighted_sentiment`) could be valuable for identifying market sentiment and its impact on price trends.

Importance of StandardScaler and PCA

StandardScaler

StandardScaler is a preprocessing technique used to standardize features by removing the mean and scaling them to unit variance. It ensures that each feature contributes equally to the model and prevents bias toward features with larger values. Here's why it's important:

1. **Normalization of Features**:
    - Different features in the dataset may have varying scales. For example, temperature may be measured in degrees, while wind speed is measured in meters per second. Without scaling, features with larger values might dominate

the learning process, leading to biased predictions.

2. **Model Performance**:
    - Many machine learning algorithms, especially those that use distance metrics (like **k-nearest neighbors** or **support vector machines**), perform better when features are on a similar scale. **StandardScaler** ensures that all features contribute equally to the model.
3. **Stability of Coefficients**:
    - Scaling improves the stability of the model coefficients, especially in models like **linear regression** and **logistic regression**, where coefficients represent the influence of each feature on the target variable. Unscaled features may lead to large coefficients, which are harder to interpret and can cause overfitting.

Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a dimensionality reduction technique that transforms the original set of features into a new set of uncorrelated features, called **principal components**. These components capture the most important information from the original features. Here's why PCA is important:

1. **Reducing Dimensionality**:
    - High-dimensional datasets can be difficult to work with and can lead to overfitting. PCA helps by reducing the number of features while retaining as much variance (information) as possible. This simplifies the dataset and improves model efficiency.
2. **Eliminating Redundancy**:
    - PCA helps in eliminating multicollinearity by creating new, uncorrelated features (principal components). This is particularly useful when the original features are highly correlated, as PCA focuses on capturing the most meaningful information from the data.
3. **Improving Computational Efficiency**:

- By reducing the dimensionality of the dataset, PCA reduces the computational burden on machine learning algorithms, especially for large datasets. This leads to faster training and prediction times without sacrificing much accuracy.

4. **Visualizing High-Dimensional Data**:

   - PCA allows us to visualize high-dimensional data in 2D or 3D, making it easier to understand patterns and relationships in the data that would otherwise be difficult to interpret.

By using **StandardScaler** and **PCA** together, we can ensure that the dataset is properly scaled and that only the most important features are retained for model training, leading to a more efficient and accurate predictive model.

In [256…
```
df_stock_data_final_imp_cleaned.columns
```

Out[256…
```
Index(['date', 'daily_return', 'adj close', 'close', 'dividends', 'high',
       'low', 'open', 'stock splits', 'volume', 'google_trends',
       'average_sentiment', 'ma', 'ema', 'dema', 'kamaw', 'midprice', 'adx_14',
       'dmp_14', 'dmn_14', 'bop', 'cmo', 'mfi', 'rrococ', 'willr', 'ad', 'obv',
       'natr', 'atr', 'true_range', 'rsi', 'volume_price_interaction',
       'tsi_13_25_13', 'tsis_13_25_13', 'ticker_label'],
      dtype='object')
```

In [257…
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import pandas as pd
df_stock_data_final_imp_cleaned = df_stock_data_final_imp.drop(columns=['ma', 'ema', 'dema', 'kamaw'])
# Define the features to combine (price-related features)
price_features = ['high', 'low', 'open']
non_price_features = ['daily_return', 'close', 'dividends', 'stock splits', 'volume', 'google_trends',
       'average_sentiment', 'midprice', 'adx_14',
       'dmp_14', 'dmn_14', 'bop', 'cmo', 'mfi', 'rrococ', 'willr', 'ad', 'obv',
       'natr', 'atr', 'true_range', 'rsi', 'tsi_13_25_13', 'tsis_13_25_13',
       'ticker_label']

# Separate the price-related features
```

```
price_data = df_stock_data_final_imp_cleaned[price_features]
# Standardize the price-related features
scaler = StandardScaler()
price_data_scaled = scaler.fit_transform(price_data)
# Apply PCA to combine price-related features
pca = PCA(n_components=2)  # Retain 2 principal components
price_pca = pca.fit_transform(price_data_scaled)
# Create a DataFrame for the PCA components
price_pca_df = pd.DataFrame(price_pca, columns=['price_pc1', 'price_pc2'])
# Concatenate the PCA components with the remaining features
non_price_data = df_stock_data_final_imp_cleaned[non_price_features]
df_stock_data_pca = pd.concat([non_price_data.reset_index(drop=True), price_pca_df.reset_index(drop=True)
# Display the explained variance ratio for PCA components
explained_variance_ratio = pca.explained_variance_ratio_
print("Explained Variance Ratio for Price Components:")
for i, var in enumerate(explained_variance_ratio, start=1):
 print(f"Price_PC{i}: {var:.4f}")
# Show the first few rows of the transformed DataFrame
df_stock_data_pca.head()
```

```
Explained Variance Ratio for Price Components:
Price_PC1: 0.9995
Price_PC2: 0.0003
```

Out[257…

	daily_return	close	dividends	stock splits	volume	google_trends	average_sentiment	midprice	adx_14
0	0.006278	141.500000	0.0	0.0	89056700	0	0.098900	176.917852	27.535055
1	0.008127	142.649994	0.0	0.0	94639600	0	0.248255	141.745003	32.263659
2	-0.024606	139.139999	0.0	0.0	98322000	0	0.122830	140.595001	28.133316
3	0.014158	141.110001	0.0	0.0	80861100	0	0.331000	140.255005	24.482109
4	0.006307	142.000000	0.0	0.0	83221100	0	0.243520	140.305000	29.144991

5 rows × 27 columns

In [257…
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
# Compute the correlation matrix
# Drop 'Ticker' and 'Date' columns from the DataFrame
#data_tesla_cleaned = data_tesla_pca.drop(columns=['ticker', 'date'])
correlation_matrix = df_stock_data_pca.corr()
# Display the correlation matrix as a table (rounded to 2 decimal places)
print("Correlation Matrix:")
print(correlation_matrix.round(2))
# Plot the heatmap for the correlation matrix
plt.figure(figsize=(20, 12))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm', vmin=-1, vmax=1, cbar=True)
# Set the title for the heatmap
plt.title('Correlation Matrix Heatmap', fontsize=16)
# Display the heatmap
plt.tight_layout()
plt.show()
```

```
Correlation Matrix:
                   daily_return  close  dividends  stock splits  volume  \
daily_return               1.00   0.05      -0.01          0.00   -0.03
close                      0.05   1.00       0.01         -0.01   -0.04
dividends                 -0.01   0.01       1.00         -0.00   -0.03
stock splits               0.00  -0.01      -0.00          1.00    0.05
volume                    -0.03  -0.04      -0.03          0.05    1.00
google_trends               NaN    NaN        NaN           NaN     NaN
average_sentiment          0.04   0.08      -0.03          0.00    0.02
midprice                   0.00   1.00       0.02         -0.01   -0.04
adx_14                     0.03   0.21       0.02          0.01   -0.09
dmp_14                     0.18   0.33      -0.00          0.01    0.02
dmn_14                    -0.10  -0.32      -0.01         -0.01   -0.00
bop                        0.07   0.03      -0.00         -0.01    0.02
cmo                        0.13   0.39       0.01          0.01    0.03
mfi                        0.02   0.15       0.01          0.02    0.06
rrococ                     0.31   0.16      -0.01          0.00   -0.06
willr                      0.10   0.22      -0.00          0.03    0.01
```

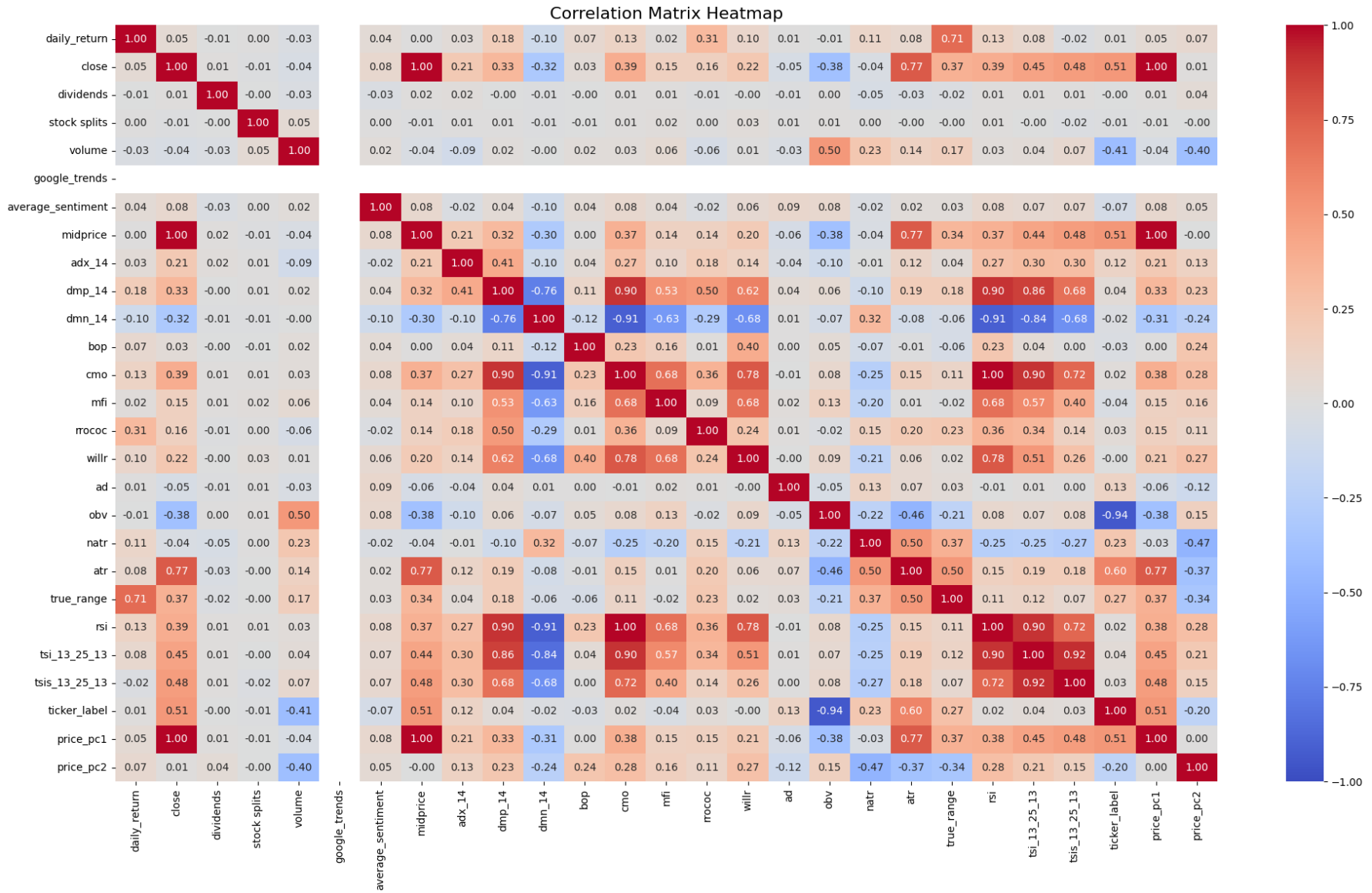
ad	0.01	-0.05	-0.01	0.01	-0.03
obv	-0.01	-0.38	0.00	0.01	0.50
natr	0.11	-0.04	-0.05	0.00	0.23
atr	0.08	0.77	-0.03	-0.00	0.14
true_range	0.71	0.37	-0.02	-0.00	0.17
rsi	0.13	0.39	0.01	0.01	0.03
tsi_13_25_13	0.08	0.45	0.01	-0.00	0.04
tsis_13_25_13	-0.02	0.48	0.01	-0.02	0.07
ticker_label	0.01	0.51	-0.00	-0.01	-0.41
price_pc1	0.05	1.00	0.01	-0.01	-0.04
price_pc2	0.07	0.01	0.04	-0.00	-0.40

	google_trends	average_sentiment	midprice	adx_14	dmp_14	\
daily_return	NaN	0.04	0.00	0.03	0.18	
close	NaN	0.08	1.00	0.21	0.33	
dividends	NaN	-0.03	0.02	0.02	-0.00	
stock splits	NaN	0.00	-0.01	0.01	0.01	
volume	NaN	0.02	-0.04	-0.09	0.02	
google_trends	NaN	NaN	NaN	NaN	NaN	
average_sentiment	NaN	1.00	0.08	-0.02	0.04	
midprice	NaN	0.08	1.00	0.21	0.32	
adx_14	NaN	-0.02	0.21	1.00	0.41	
dmp_14	NaN	0.04	0.32	0.41	1.00	
dmn_14	NaN	-0.10	-0.30	-0.10	-0.76	
bop	NaN	0.04	0.00	0.04	0.11	
cmo	NaN	0.08	0.37	0.27	0.90	
mfi	NaN	0.04	0.14	0.10	0.53	
rrococ	NaN	-0.02	0.14	0.18	0.50	
willr	NaN	0.06	0.20	0.14	0.62	
ad	NaN	0.09	-0.06	-0.04	0.04	
obv	NaN	0.08	-0.38	-0.10	0.06	
natr	NaN	-0.02	-0.04	-0.01	-0.10	
atr	NaN	0.02	0.77	0.12	0.19	
true_range	NaN	0.03	0.34	0.04	0.18	
rsi	NaN	0.08	0.37	0.27	0.90	
tsi_13_25_13	NaN	0.07	0.44	0.30	0.86	
tsis_13_25_13	NaN	0.07	0.48	0.30	0.68	

```
ticker_label                NaN              -0.07      0.51    0.12    0.04
price_pc1                   NaN               0.08      1.00    0.21    0.33
price_pc2                   NaN               0.05     -0.00    0.13    0.23

                    ...   obv  natr   atr  true_range   rsi  tsi_13_25_13  \
daily_return        ... -0.01  0.11  0.08        0.71  0.13          0.08
close               ... -0.38 -0.04  0.77        0.37  0.39          0.45
dividends           ...  0.00 -0.05 -0.03       -0.02  0.01          0.01
stock splits        ...  0.01  0.00 -0.00       -0.00  0.01         -0.00
volume              ...  0.50  0.23  0.14        0.17  0.03          0.04
google_trends       ...   NaN   NaN   NaN         NaN   NaN           NaN
average_sentiment   ...  0.08 -0.02  0.02        0.03  0.08          0.07
midprice            ... -0.38 -0.04  0.77        0.34  0.37          0.44
adx_14              ... -0.10 -0.01  0.12        0.04  0.27          0.30
dmp_14              ...  0.06 -0.10  0.19        0.18  0.90          0.86
dmn_14              ... -0.07  0.32 -0.08       -0.06 -0.91         -0.84
bop                 ...  0.05 -0.07 -0.01       -0.06  0.23          0.04
cmo                 ...  0.08 -0.25  0.15        0.11  1.00          0.90
mfi                 ...  0.13 -0.20  0.01       -0.02  0.68          0.57
rrococ              ... -0.02  0.15  0.20        0.23  0.36          0.34
willr               ...  0.09 -0.21  0.06        0.02  0.78          0.51
ad                  ... -0.05  0.13  0.07        0.03 -0.01          0.01
obv                 ...  1.00 -0.22 -0.46       -0.21  0.08          0.07
natr                ... -0.22  1.00  0.50        0.37 -0.25         -0.25
atr                 ... -0.46  0.50  1.00        0.50  0.15          0.19
true_range          ... -0.21  0.37  0.50        1.00  0.11          0.12
rsi                 ...  0.08 -0.25  0.15        0.11  1.00          0.90
tsi_13_25_13        ...  0.07 -0.25  0.19        0.12  0.90          1.00
tsis_13_25_13       ...  0.08 -0.27  0.18        0.07  0.72          0.92
ticker_label        ... -0.94  0.23  0.60        0.27  0.02          0.04
price_pc1           ... -0.38 -0.03  0.77        0.37  0.38          0.45
price_pc2           ...  0.15 -0.47 -0.37       -0.34  0.28          0.21

                    tsis_13_25_13  ticker_label  price_pc1  price_pc2
daily_return                -0.02          0.01       0.05       0.07
close                        0.48          0.51       1.00       0.01
dividends                    0.01         -0.00       0.01       0.04
```

stock splits	−0.02	−0.01	−0.01	−0.00
volume	0.07	−0.41	−0.04	−0.40
google_trends	NaN	NaN	NaN	NaN
average_sentiment	0.07	−0.07	0.08	0.05
midprice	0.48	0.51	1.00	−0.00
adx_14	0.30	0.12	0.21	0.13
dmp_14	0.68	0.04	0.33	0.23
dmn_14	−0.68	−0.02	−0.31	−0.24
bop	0.00	−0.03	0.00	0.24
cmo	0.72	0.02	0.38	0.28
mfi	0.40	−0.04	0.15	0.16
rrococ	0.14	0.03	0.15	0.11
willr	0.26	−0.00	0.21	0.27
ad	0.00	0.13	−0.06	−0.12
obv	0.08	−0.94	−0.38	0.15
natr	−0.27	0.23	−0.03	−0.47
atr	0.18	0.60	0.77	−0.37
true_range	0.07	0.27	0.37	−0.34
rsi	0.72	0.02	0.38	0.28
tsi_13_25_13	0.92	0.04	0.45	0.21
tsis_13_25_13	1.00	0.03	0.48	0.15
ticker_label	0.03	1.00	0.51	−0.20
price_pc1	0.48	0.51	1.00	0.00
price_pc2	0.15	−0.20	0.00	1.00

[27 rows x 27 columns]



In [257…
```
df_stock_data_pca.head()
```

Out[257…

	daily_return	close	dividends	stock splits	volume	google_trends	average_sentiment	midprice	adx_14
0	0.006278	141.500000	0.0	0.0	89056700	0	0.098900	176.917852	27.535055
1	0.008127	142.649994	0.0	0.0	94639600	0	0.248255	141.745003	32.263659
2	-0.024606	139.139999	0.0	0.0	98322000	0	0.122830	140.595001	28.133316
3	0.014158	141.110001	0.0	0.0	80861100	0	0.331000	140.255005	24.482109
4	0.006307	142.000000	0.0	0.0	83221100	0	0.243520	140.305000	29.144991

5 rows × 27 columns

In [258…

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from keras.models import Sequential
from keras.layers import LSTM, Dense, Conv1D, Flatten
import matplotlib.pyplot as plt

# Placeholder for results
results = []

# Load the dataset (ensure `data_tesla` is defined)
merged_df = df_stock_data_pca
# Clean and standardize column names
merged_df.columns = merged_df.columns.str.strip().str.lower()
# Group by each stock
grouped_stocks = merged_df.groupby('ticker_label')

# Iterate through each stock group
for stock_label, stock_data in grouped_stocks:
    stock_name = tickers[stock_label]  # Map stock_label to the corresponding ticker
```

```
    print(f"Processing stock: {stock_name}")
    # Filter and select relevant columns
    Independent_features = [col for col in df_stock_data_pca.columns.tolist() if col not in ['close']]
    target = 'close'  # Target variable
    stock_data = stock_data[Independent_features + [target]].dropna()
    if len(stock_data) < 20:  # Skip if there's not enough data
        print(f"Not enough data for stock {stock_name}. Skipping...")
        continue

    # Standardize the data using StandardScaler
    scaler = StandardScaler()
    scaled_data = scaler.fit_transform(stock_data)
    scaled_df = pd.DataFrame(scaled_data, columns=Independent_features + [target])

    # Create sequences for LSTM and CNN
    timesteps = 10
    X, y = [], []
    for i in range(timesteps, len(scaled_df)):
        X.append(scaled_df.iloc[i-timesteps:i, :-1].values)
        y.append(scaled_df.iloc[i, -1])
    X, y = np.array(X), np.array(y)

    # Split into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Define the LSTM model
    lstm_model = Sequential([
        LSTM(50, activation='relu', input_shape=(X_train.shape[1], X_train.shape[2]), return_sequences=Tr
        LSTM(50, activation='relu'),
        Dense(1)
    ])
    lstm_model.compile(optimizer='adam', loss='mse')

    # Define the CNN model
    cnn_model = Sequential([
        Conv1D(filters=64, kernel_size=2, activation='relu', input_shape=(X_train.shape[1], X_train.shape
        Flatten(),
```

```
        Dense(50, activation='relu'),
        Dense(1)
    ])
    cnn_model.compile(optimizer='adam', loss='mse')

    # Train both models
    print(f"Training LSTM model for stock: {stock_name}")
    lstm_model.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_test, y_test), verbose=

    print(f"Training CNN model for stock: {stock_name}")
    cnn_model.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_test, y_test), verbose=1

    # Make predictions for both models
    lstm_y_pred = lstm_model.predict(X_test).flatten()
    cnn_y_pred = cnn_model.predict(X_test).flatten()

    # Calculate R² and RMSE for both models
    lstm_r2 = r2_score(y_test, lstm_y_pred)
    lstm_rmse = np.sqrt(mean_squared_error(y_test, lstm_y_pred))

    cnn_r2 = r2_score(y_test, cnn_y_pred)
    cnn_rmse = np.sqrt(mean_squared_error(y_test, cnn_y_pred))

    results.append({
        'Stock': stock_name,
        'LSTM_R²': lstm_r2,
        'LSTM_RMSE': lstm_rmse,
        'CNN_R²': cnn_r2,
        'CNN_RMSE': cnn_rmse
    })

    # Plot the results side-by-side
    plt.figure(figsize=(12, 6))
    plt.plot(y_test, label='Actual Price', linestyle='--')
    plt.plot(lstm_y_pred, label='LSTM Predicted Price', linestyle='--')
    plt.plot(cnn_y_pred, label='CNN Predicted Price', linestyle='--')
    plt.title(f'Stock Price Prediction for {stock_name}\nLSTM (R²: {lstm_r2:.2f}, RMSE: {lstm_rmse:.2f})
```

```
    plt.legend()
    plt.xlabel('Time')
    plt.ylabel('Price')
    plt.tight_layout()
    plt.show()

# Create a DataFrame for the results
results_df = pd.DataFrame(results)

# Display the results
print(results_df)
```

```
Processing stock: AAPL
Training LSTM model for stock: AAPL
Epoch 1/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 2s 48ms/step - loss: 0.9720 - val_loss: 0.6774
Epoch 2/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.6052 - val_loss: 0.5309
Epoch 3/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.3383 - val_loss: 0.3479
Epoch 4/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.2496 - val_loss: 0.1775
Epoch 5/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.1547 - val_loss: 0.1330
Epoch 6/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.1302 - val_loss: 0.1016
Epoch 7/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0982 - val_loss: 0.0766
Epoch 8/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0860 - val_loss: 0.0625
Epoch 9/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0714 - val_loss: 0.0584
Epoch 10/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0758 - val_loss: 0.0612
Epoch 11/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0537 - val_loss: 0.0554
Epoch 12/50
```

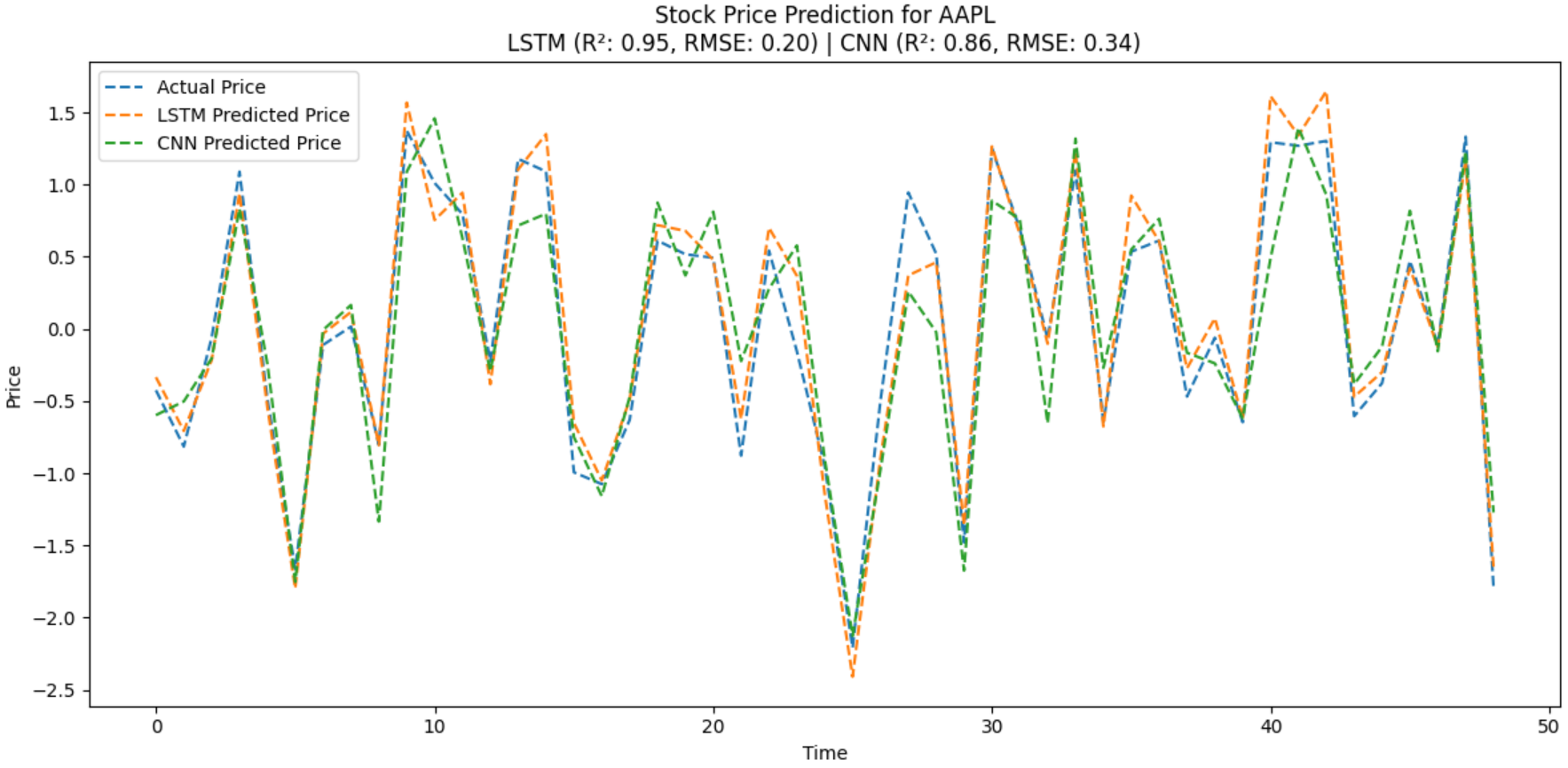
```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0656 - val_loss: 0.0528
Epoch 13/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0542 - val_loss: 0.0505
Epoch 14/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0494 - val_loss: 0.0500
Epoch 15/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0455 - val_loss: 0.0475
Epoch 16/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0476 - val_loss: 0.0463
Epoch 17/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0512 - val_loss: 0.0435
Epoch 18/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0468 - val_loss: 0.0418
Epoch 19/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0415 - val_loss: 0.0418
Epoch 20/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0444 - val_loss: 0.0403
Epoch 21/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0404 - val_loss: 0.0439
Epoch 22/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0457 - val_loss: 0.0385
Epoch 23/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0369 - val_loss: 0.0401
Epoch 24/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0443 - val_loss: 0.0365
Epoch 25/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 0.0407 - val_loss: 0.0359
Epoch 26/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0351 - val_loss: 0.0360
Epoch 27/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0384 - val_loss: 0.0367
Epoch 28/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0380 - val_loss: 0.0318
Epoch 29/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0358 - val_loss: 0.0337
Epoch 30/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0330 - val_loss: 0.0324
```

```
Epoch 31/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0329 - val_loss: 0.0360
Epoch 32/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0307 - val_loss: 0.0309
Epoch 33/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0333 - val_loss: 0.0332
Epoch 34/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0290 - val_loss: 0.0317
Epoch 35/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0355 - val_loss: 0.0321
Epoch 36/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0314 - val_loss: 0.0340
Epoch 37/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0379 - val_loss: 0.0326
Epoch 38/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0294 - val_loss: 0.0413
Epoch 39/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0271 - val_loss: 0.0306
Epoch 40/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0299 - val_loss: 0.0338
Epoch 41/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0260 - val_loss: 0.0348
Epoch 42/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0273 - val_loss: 0.0342
Epoch 43/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0269 - val_loss: 0.0394
Epoch 44/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0273 - val_loss: 0.0372
Epoch 45/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - loss: 0.0277 - val_loss: 0.0354
Epoch 46/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0291 - val_loss: 0.0373
Epoch 47/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0281 - val_loss: 0.0391
Epoch 48/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0278 - val_loss: 0.0347
Epoch 49/50
```

```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0256 - val_loss: 0.0383
Epoch 50/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0247 - val_loss: 0.0403
Training CNN model for stock: AAPL
Epoch 1/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - loss: 1.0993 - val_loss: 0.6544
Epoch 2/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.4112 - val_loss: 0.2899
Epoch 3/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.1602 - val_loss: 0.1985
Epoch 4/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0922 - val_loss: 0.1750
Epoch 5/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0614 - val_loss: 0.1629
Epoch 6/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0466 - val_loss: 0.1450
Epoch 7/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0347 - val_loss: 0.1396
Epoch 8/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0325 - val_loss: 0.1435
Epoch 9/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0293 - val_loss: 0.1290
Epoch 10/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0210 - val_loss: 0.1271
Epoch 11/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0157 - val_loss: 0.1250
Epoch 12/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0135 - val_loss: 0.1231
Epoch 13/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0108 - val_loss: 0.1228
Epoch 14/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0078 - val_loss: 0.1214
Epoch 15/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0075 - val_loss: 0.1195
Epoch 16/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0057 - val_loss: 0.1195
Epoch 17/50
```

```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0046 - val_loss: 0.1185
Epoch 18/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0040 - val_loss: 0.1186
Epoch 19/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0036 - val_loss: 0.1167
Epoch 20/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0027 - val_loss: 0.1181
Epoch 21/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0023 - val_loss: 0.1186
Epoch 22/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0019 - val_loss: 0.1169
Epoch 23/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0019 - val_loss: 0.1168
Epoch 24/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0013 - val_loss: 0.1176
Epoch 25/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0017 - val_loss: 0.1169
Epoch 26/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 9.8806e-04 - val_loss: 0.1169
Epoch 27/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 6.2293e-04 - val_loss: 0.1166
Epoch 28/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 8.3970e-04 - val_loss: 0.1171
Epoch 29/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 3.9791e-04 - val_loss: 0.1165
Epoch 30/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 5.5817e-04 - val_loss: 0.1168
Epoch 31/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.1390e-04 - val_loss: 0.1163
Epoch 32/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 7.2929e-04 - val_loss: 0.1161
Epoch 33/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.7273e-04 - val_loss: 0.1163
Epoch 34/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3.4062e-04 - val_loss: 0.1172
Epoch 35/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 3.1731e-04 - val_loss: 0.1167
```

```
Epoch 36/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2.7199e-04 - val_loss: 0.1164
Epoch 37/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.1617e-04 - val_loss: 0.1160
Epoch 38/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3.0446e-04 - val_loss: 0.1167
Epoch 39/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.7182e-04 - val_loss: 0.1164
Epoch 40/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.5139e-04 - val_loss: 0.1165
Epoch 41/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.8123e-04 - val_loss: 0.1162
Epoch 42/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.1806e-04 - val_loss: 0.1167
Epoch 43/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.8219e-04 - val_loss: 0.1159
Epoch 44/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 2.7737e-04 - val_loss: 0.1180
Epoch 45/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.6473e-04 - val_loss: 0.1160
Epoch 46/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.6611e-04 - val_loss: 0.1169
Epoch 47/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.4573e-04 - val_loss: 0.1162
Epoch 48/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 1.1937e-04 - val_loss: 0.1165
Epoch 49/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.7902e-04 - val_loss: 0.1170
Epoch 50/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.1555e-04 - val_loss: 0.1162
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 175ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
```




```
Processing stock: AMD
Training LSTM model for stock: AMD
Epoch 1/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 2s 58ms/step - loss: 0.7835 - val_loss: 0.4215
Epoch 2/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.3849 - val_loss: 0.2350
Epoch 3/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.3072 - val_loss: 0.1481
Epoch 4/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.1742 - val_loss: 0.1167
Epoch 5/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.1309 - val_loss: 0.0888
```

```
Epoch 6/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0946 - val_loss: 0.0734
Epoch 7/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0607 - val_loss: 0.0522
Epoch 8/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0525 - val_loss: 0.0438
Epoch 9/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0378 - val_loss: 0.0402
Epoch 10/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0374 - val_loss: 0.0369
Epoch 11/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0370 - val_loss: 0.0336
Epoch 12/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0299 - val_loss: 0.0323
Epoch 13/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0274 - val_loss: 0.0305
Epoch 14/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0248 - val_loss: 0.0291
Epoch 15/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0242 - val_loss: 0.0268
Epoch 16/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0253 - val_loss: 0.0262
Epoch 17/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0239 - val_loss: 0.0257
Epoch 18/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0239 - val_loss: 0.0265
Epoch 19/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0216 - val_loss: 0.0260
Epoch 20/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0205 - val_loss: 0.0287
Epoch 21/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0214 - val_loss: 0.0261
Epoch 22/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - loss: 0.0205 - val_loss: 0.0294
Epoch 23/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0203 - val_loss: 0.0242
Epoch 24/50
```

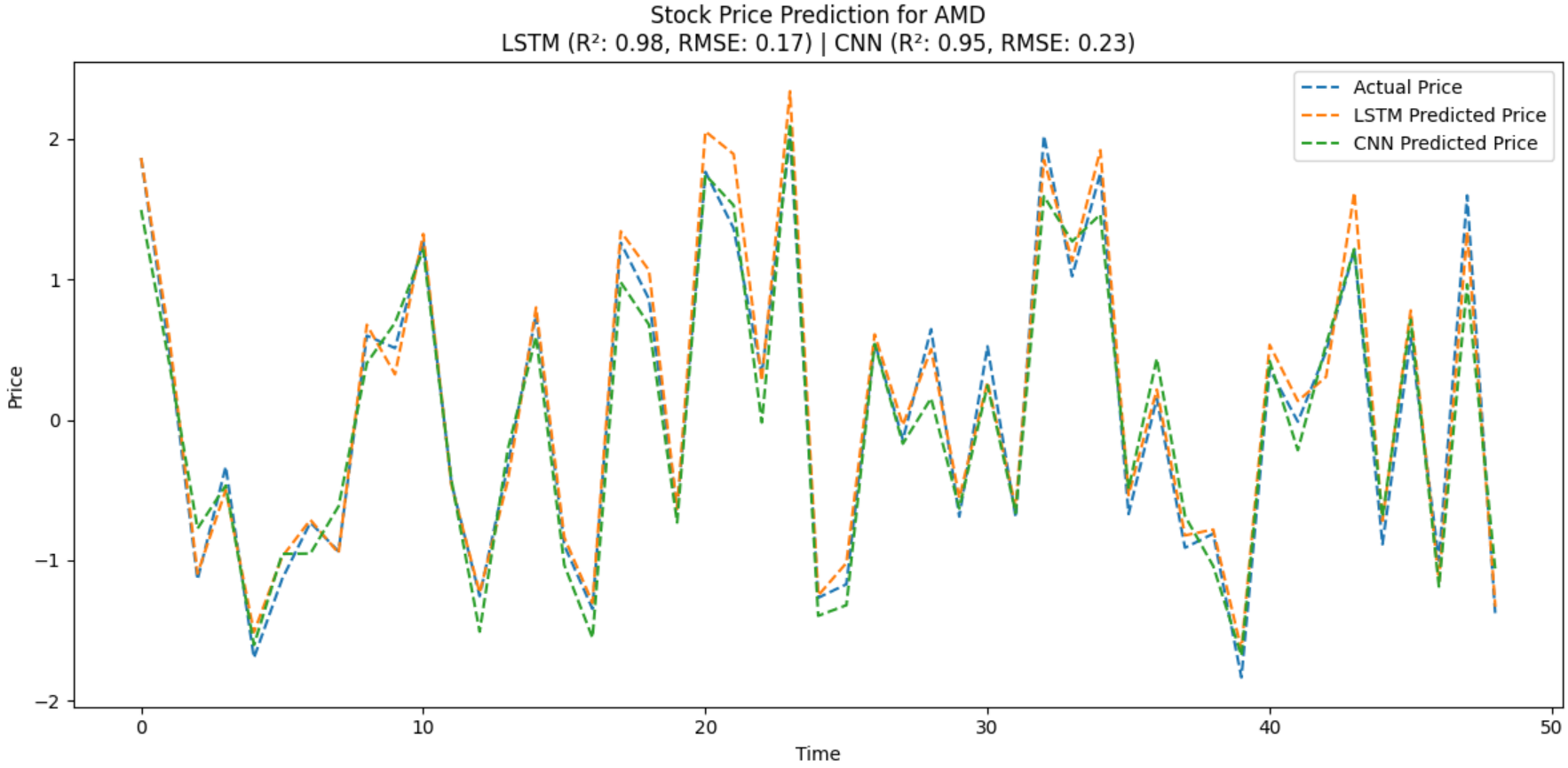
```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0213 - val_loss: 0.0314
Epoch 25/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0227 - val_loss: 0.0249
Epoch 26/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0257 - val_loss: 0.0259
Epoch 27/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0220 - val_loss: 0.0252
Epoch 28/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0199 - val_loss: 0.0255
Epoch 29/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0214 - val_loss: 0.0245
Epoch 30/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0186 - val_loss: 0.0227
Epoch 31/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0165 - val_loss: 0.0274
Epoch 32/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0184 - val_loss: 0.0245
Epoch 33/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0192 - val_loss: 0.0223
Epoch 34/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0185 - val_loss: 0.0274
Epoch 35/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0177 - val_loss: 0.0256
Epoch 36/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0171 - val_loss: 0.0268
Epoch 37/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0165 - val_loss: 0.0244
Epoch 38/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0184 - val_loss: 0.0279
Epoch 39/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0172 - val_loss: 0.0254
Epoch 40/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0164 - val_loss: 0.0294
Epoch 41/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0150 - val_loss: 0.0240
Epoch 42/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0148 - val_loss: 0.0257
```

```
Epoch 43/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0144 - val_loss: 0.0270
Epoch 44/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0155 - val_loss: 0.0243
Epoch 45/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0163 - val_loss: 0.0274
Epoch 46/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0153 - val_loss: 0.0276
Epoch 47/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0156 - val_loss: 0.0261
Epoch 48/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0163 - val_loss: 0.0301
Epoch 49/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0139 - val_loss: 0.0244
Epoch 50/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0141 - val_loss: 0.0280
Training CNN model for stock: AMD
Epoch 1/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 1s 19ms/step - loss: 0.9024 - val_loss: 0.1487
Epoch 2/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.2125 - val_loss: 0.1110
Epoch 3/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.1518 - val_loss: 0.0858
Epoch 4/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0654 - val_loss: 0.0948
Epoch 5/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0495 - val_loss: 0.0504
Epoch 6/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0341 - val_loss: 0.0579
Epoch 7/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0244 - val_loss: 0.0469
Epoch 8/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0205 - val_loss: 0.0567
Epoch 9/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0145 - val_loss: 0.0572
Epoch 10/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0128 - val_loss: 0.0440
```

```
Epoch 11/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0084 - val_loss: 0.0532
Epoch 12/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0082 - val_loss: 0.0453
Epoch 13/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0059 - val_loss: 0.0504
Epoch 14/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0042 - val_loss: 0.0488
Epoch 15/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0039 - val_loss: 0.0493
Epoch 16/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0029 - val_loss: 0.0515
Epoch 17/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0020 - val_loss: 0.0477
Epoch 18/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0021 - val_loss: 0.0536
Epoch 19/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0015 - val_loss: 0.0481
Epoch 20/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0015 - val_loss: 0.0531
Epoch 21/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0011 - val_loss: 0.0482
Epoch 22/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 9.9667e-04 - val_loss: 0.0527
Epoch 23/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 7.6986e-04 - val_loss: 0.0495
Epoch 24/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 5.5521e-04 - val_loss: 0.0524
Epoch 25/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 5.9317e-04 - val_loss: 0.0520
Epoch 26/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3.3827e-04 - val_loss: 0.0503
Epoch 27/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3.4044e-04 - val_loss: 0.0524
Epoch 28/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 2.7401e-04 - val_loss: 0.0501
Epoch 29/50
```

```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.3329e-04 - val_loss: 0.0522
Epoch 30/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.6251e-04 - val_loss: 0.0517
Epoch 31/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.6603e-04 - val_loss: 0.0518
Epoch 32/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 1.0542e-04 - val_loss: 0.0523
Epoch 33/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 7.6927e-05 - val_loss: 0.0515
Epoch 34/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 5.4931e-05 - val_loss: 0.0523
Epoch 35/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 5.9003e-05 - val_loss: 0.0518
Epoch 36/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 4.7808e-05 - val_loss: 0.0522
Epoch 37/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 4.1995e-05 - val_loss: 0.0526
Epoch 38/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.7853e-05 - val_loss: 0.0520
Epoch 39/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3.2209e-05 - val_loss: 0.0522
Epoch 40/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.2497e-05 - val_loss: 0.0526
Epoch 41/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.6781e-05 - val_loss: 0.0521
Epoch 42/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 1.2318e-05 - val_loss: 0.0525
Epoch 43/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 1.3587e-05 - val_loss: 0.0525
Epoch 44/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 8.0564e-06 - val_loss: 0.0523
Epoch 45/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 7.1072e-06 - val_loss: 0.0528
Epoch 46/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 7.1354e-06 - val_loss: 0.0524
Epoch 47/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 4.9129e-06 - val_loss: 0.0524
```

```
Epoch 48/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 4.2358e-06 - val_loss: 0.0526
Epoch 49/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 4.9702e-06 - val_loss: 0.0525
Epoch 50/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3.3344e-06 - val_loss: 0.0525
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 142ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
```




```
Processing stock: AMZN
Training LSTM model for stock: AMZN
Epoch 1/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 2s 49ms/step - loss: 0.8346 - val_loss: 0.5544
```

```
Epoch 2/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 0.5878 - val_loss: 0.3315
Epoch 3/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.3054 - val_loss: 0.1901
Epoch 4/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.1900 - val_loss: 0.1565
Epoch 5/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.1594 - val_loss: 0.1259
Epoch 6/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.1004 - val_loss: 0.0827
Epoch 7/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0824 - val_loss: 0.0689
Epoch 8/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0691 - val_loss: 0.0530
Epoch 9/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0508 - val_loss: 0.0438
Epoch 10/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0459 - val_loss: 0.0389
Epoch 11/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0377 - val_loss: 0.0303
Epoch 12/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0346 - val_loss: 0.0279
Epoch 13/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0339 - val_loss: 0.0242
Epoch 14/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0287 - val_loss: 0.0219
Epoch 15/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0287 - val_loss: 0.0194
Epoch 16/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0232 - val_loss: 0.0199
Epoch 17/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0213 - val_loss: 0.0167
Epoch 18/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0225 - val_loss: 0.0196
Epoch 19/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0241 - val_loss: 0.0162
Epoch 20/50
```

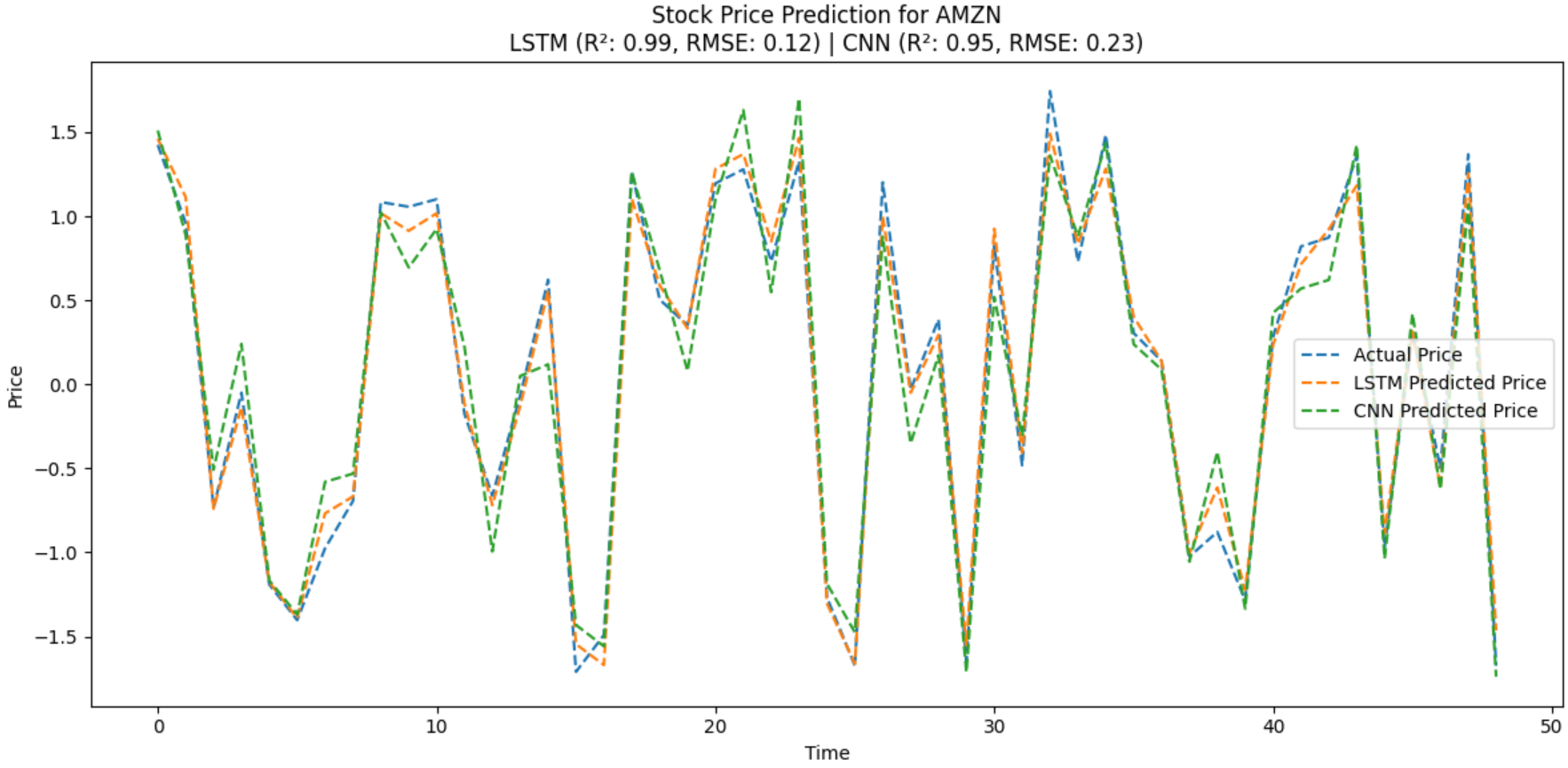
```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0197 - val_loss: 0.0155
Epoch 21/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0221 - val_loss: 0.0184
Epoch 22/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.0211 - val_loss: 0.0160
Epoch 23/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0201 - val_loss: 0.0188
Epoch 24/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0192 - val_loss: 0.0151
Epoch 25/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0172 - val_loss: 0.0150
Epoch 26/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0172 - val_loss: 0.0180
Epoch 27/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0182 - val_loss: 0.0142
Epoch 28/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0195 - val_loss: 0.0169
Epoch 29/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0158 - val_loss: 0.0137
Epoch 30/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0198 - val_loss: 0.0167
Epoch 31/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0146 - val_loss: 0.0148
Epoch 32/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0182 - val_loss: 0.0183
Epoch 33/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0150 - val_loss: 0.0143
Epoch 34/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0162 - val_loss: 0.0150
Epoch 35/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0172 - val_loss: 0.0126
Epoch 36/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0150 - val_loss: 0.0155
Epoch 37/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0168 - val_loss: 0.0157
Epoch 38/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0184 - val_loss: 0.0141
```

```
Epoch 39/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0197 - val_loss: 0.0141
Epoch 40/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0122 - val_loss: 0.0152
Epoch 41/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0139 - val_loss: 0.0138
Epoch 42/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0162 - val_loss: 0.0166
Epoch 43/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0149 - val_loss: 0.0137
Epoch 44/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0148 - val_loss: 0.0132
Epoch 45/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0165 - val_loss: 0.0148
Epoch 46/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0158 - val_loss: 0.0165
Epoch 47/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0130 - val_loss: 0.0159
Epoch 48/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0162 - val_loss: 0.0149
Epoch 49/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0156 - val_loss: 0.0141
Epoch 50/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0131 - val_loss: 0.0136
Training CNN model for stock: AMZN
Epoch 1/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.7911 - val_loss: 0.2260
Epoch 2/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.2422 - val_loss: 0.1156
Epoch 3/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.1157 - val_loss: 0.1180
Epoch 4/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0894 - val_loss: 0.0925
Epoch 5/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0532 - val_loss: 0.0759
Epoch 6/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0429 - val_loss: 0.0636
```

```
Epoch 7/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0293 - val_loss: 0.0745
Epoch 8/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0235 - val_loss: 0.0610
Epoch 9/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0179 - val_loss: 0.0663
Epoch 10/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0142 - val_loss: 0.0542
Epoch 11/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0126 - val_loss: 0.0560
Epoch 12/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0112 - val_loss: 0.0527
Epoch 13/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0079 - val_loss: 0.0543
Epoch 14/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0057 - val_loss: 0.0546
Epoch 15/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0047 - val_loss: 0.0560
Epoch 16/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0039 - val_loss: 0.0548
Epoch 17/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0035 - val_loss: 0.0551
Epoch 18/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0027 - val_loss: 0.0552
Epoch 19/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0020 - val_loss: 0.0543
Epoch 20/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0020 - val_loss: 0.0537
Epoch 21/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0017 - val_loss: 0.0544
Epoch 22/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0012 - val_loss: 0.0547
Epoch 23/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0012 - val_loss: 0.0549
Epoch 24/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 9.5518e-04 - val_loss: 0.0539
Epoch 25/50
```

```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 9.0847e-04 - val_loss: 0.0542
Epoch 26/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 6.3980e-04 - val_loss: 0.0545
Epoch 27/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 4.9163e-04 - val_loss: 0.0543
Epoch 28/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 5.3045e-04 - val_loss: 0.0538
Epoch 29/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 4.1911e-04 - val_loss: 0.0545
Epoch 30/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 3.4027e-04 - val_loss: 0.0540
Epoch 31/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2.9039e-04 - val_loss: 0.0541
Epoch 32/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3.0742e-04 - val_loss: 0.0541
Epoch 33/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.0863e-04 - val_loss: 0.0542
Epoch 34/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3.0747e-04 - val_loss: 0.0538
Epoch 35/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2.9696e-04 - val_loss: 0.0540
Epoch 36/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2.1855e-04 - val_loss: 0.0543
Epoch 37/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.4238e-04 - val_loss: 0.0540
Epoch 38/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.2462e-04 - val_loss: 0.0540
Epoch 39/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.0467e-04 - val_loss: 0.0540
Epoch 40/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 7.7181e-05 - val_loss: 0.0537
Epoch 41/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 6.3039e-05 - val_loss: 0.0539
Epoch 42/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 8.0045e-05 - val_loss: 0.0538
Epoch 43/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 5.9346e-05 - val_loss: 0.0539
```

```
Epoch 44/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 5.7428e-05 - val_loss: 0.0542
Epoch 45/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.9073e-05 - val_loss: 0.0539
Epoch 46/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2.9277e-05 - val_loss: 0.0540
Epoch 47/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.9336e-05 - val_loss: 0.0540
Epoch 48/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2.4976e-05 - val_loss: 0.0539
Epoch 49/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2.8094e-05 - val_loss: 0.0539
Epoch 50/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.4958e-05 - val_loss: 0.0539
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 164ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
```




```
Processing stock: INTC
Training LSTM model for stock: INTC
Epoch 1/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 2s 62ms/step - loss: 0.7464 - val_loss: 0.4391
Epoch 2/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.3153 - val_loss: 0.2223
Epoch 3/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.1422 - val_loss: 0.1264
Epoch 4/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0752 - val_loss: 0.0817
Epoch 5/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0718 - val_loss: 0.0502
```

```
Epoch 6/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0418 - val_loss: 0.0496
Epoch 7/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0337 - val_loss: 0.0467
Epoch 8/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0273 - val_loss: 0.0400
Epoch 9/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0210 - val_loss: 0.0351
Epoch 10/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0217 - val_loss: 0.0330
Epoch 11/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0203 - val_loss: 0.0329
Epoch 12/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0195 - val_loss: 0.0310
Epoch 13/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0167 - val_loss: 0.0286
Epoch 14/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0155 - val_loss: 0.0304
Epoch 15/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0160 - val_loss: 0.0286
Epoch 16/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0160 - val_loss: 0.0270
Epoch 17/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0161 - val_loss: 0.0274
Epoch 18/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0147 - val_loss: 0.0252
Epoch 19/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0156 - val_loss: 0.0253
Epoch 20/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0127 - val_loss: 0.0247
Epoch 21/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0128 - val_loss: 0.0256
Epoch 22/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0131 - val_loss: 0.0238
Epoch 23/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0124 - val_loss: 0.0229
Epoch 24/50
```

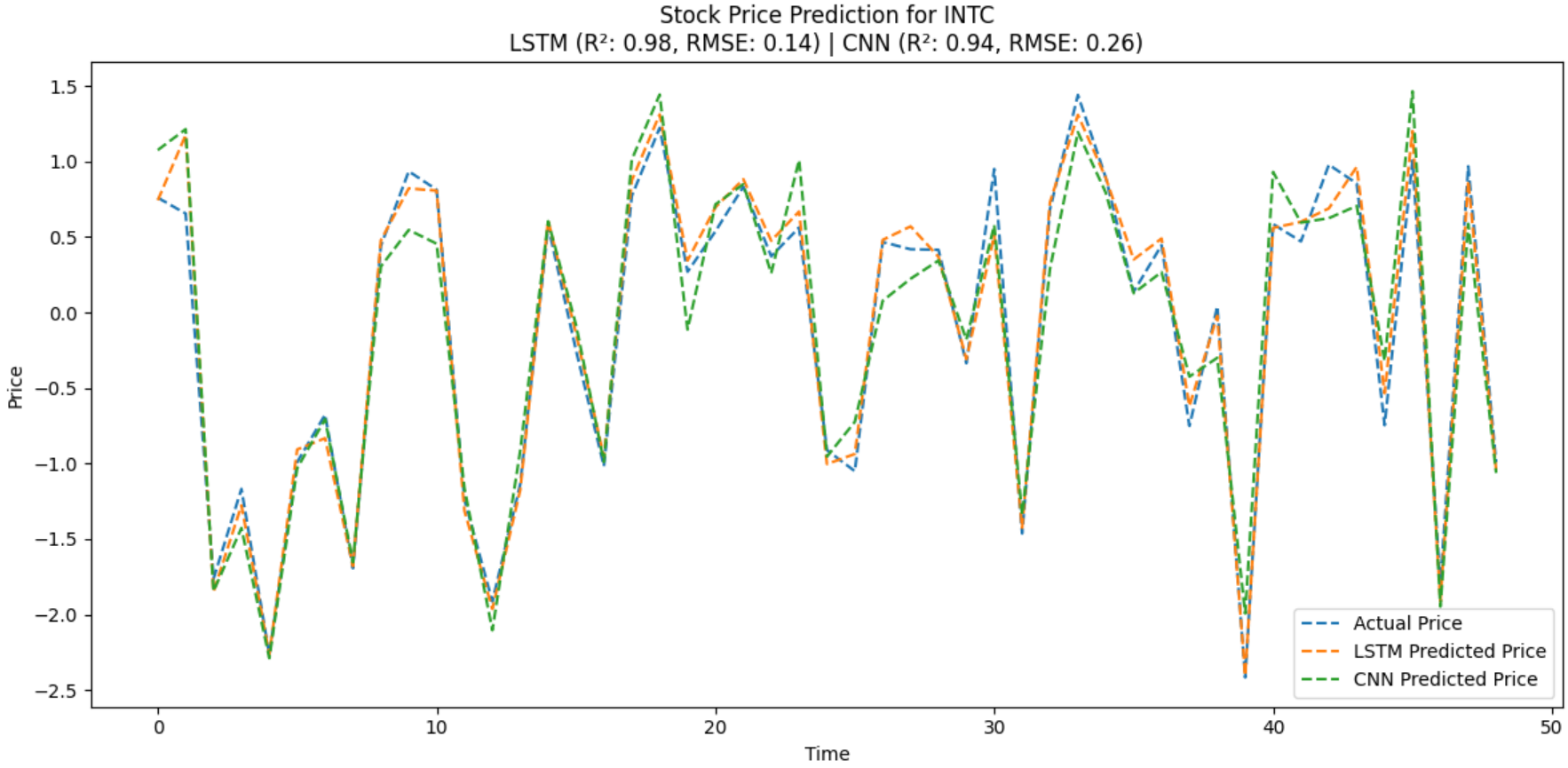
```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0132 - val_loss: 0.0233
Epoch 25/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0109 - val_loss: 0.0225
Epoch 26/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0112 - val_loss: 0.0220
Epoch 27/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0119 - val_loss: 0.0213
Epoch 28/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0102 - val_loss: 0.0229
Epoch 29/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0108 - val_loss: 0.0213
Epoch 30/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0113 - val_loss: 0.0244
Epoch 31/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0113 - val_loss: 0.0213
Epoch 32/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0105 - val_loss: 0.0223
Epoch 33/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0097 - val_loss: 0.0205
Epoch 34/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0100 - val_loss: 0.0220
Epoch 35/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0105 - val_loss: 0.0210
Epoch 36/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0105 - val_loss: 0.0208
Epoch 37/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0084 - val_loss: 0.0213
Epoch 38/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0098 - val_loss: 0.0199
Epoch 39/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0102 - val_loss: 0.0218
Epoch 40/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0098 - val_loss: 0.0211
Epoch 41/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0089 - val_loss: 0.0211
Epoch 42/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0083 - val_loss: 0.0206
```

```
Epoch 43/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0083 - val_loss: 0.0207
Epoch 44/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0084 - val_loss: 0.0217
Epoch 45/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0081 - val_loss: 0.0194
Epoch 46/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0087 - val_loss: 0.0217
Epoch 47/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0081 - val_loss: 0.0205
Epoch 48/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0091 - val_loss: 0.0198
Epoch 49/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0085 - val_loss: 0.0203
Epoch 50/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0083 - val_loss: 0.0201
Training CNN model for stock: INTC
Epoch 1/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 0.8136 - val_loss: 0.3684
Epoch 2/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.2463 - val_loss: 0.2041
Epoch 3/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0945 - val_loss: 0.1739
Epoch 4/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0639 - val_loss: 0.1155
Epoch 5/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0440 - val_loss: 0.0890
Epoch 6/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0276 - val_loss: 0.0764
Epoch 7/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0252 - val_loss: 0.0724
Epoch 8/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0190 - val_loss: 0.0712
Epoch 9/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0127 - val_loss: 0.0709
Epoch 10/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0122 - val_loss: 0.0707
```

```
Epoch 11/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0101 - val_loss: 0.0722
Epoch 12/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0075 - val_loss: 0.0732
Epoch 13/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0067 - val_loss: 0.0725
Epoch 14/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0057 - val_loss: 0.0720
Epoch 15/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0047 - val_loss: 0.0719
Epoch 16/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0043 - val_loss: 0.0710
Epoch 17/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0033 - val_loss: 0.0715
Epoch 18/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0030 - val_loss: 0.0710
Epoch 19/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0029 - val_loss: 0.0713
Epoch 20/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0025 - val_loss: 0.0713
Epoch 21/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0018 - val_loss: 0.0711
Epoch 22/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0016 - val_loss: 0.0708
Epoch 23/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0017 - val_loss: 0.0709
Epoch 24/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0014 - val_loss: 0.0705
Epoch 25/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0012 - val_loss: 0.0705
Epoch 26/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 9.2068e-04 - val_loss: 0.0702
Epoch 27/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 8.4683e-04 - val_loss: 0.0700
Epoch 28/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 6.9396e-04 - val_loss: 0.0700
Epoch 29/50
```

```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 6.9094e-04 - val_loss: 0.0702
Epoch 30/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 5.5206e-04 - val_loss: 0.0698
Epoch 31/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 4.7185e-04 - val_loss: 0.0699
Epoch 32/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3.7155e-04 - val_loss: 0.0698
Epoch 33/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3.1360e-04 - val_loss: 0.0696
Epoch 34/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3.0207e-04 - val_loss: 0.0698
Epoch 35/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.1401e-04 - val_loss: 0.0697
Epoch 36/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.2875e-04 - val_loss: 0.0695
Epoch 37/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.8701e-04 - val_loss: 0.0695
Epoch 38/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.5083e-04 - val_loss: 0.0697
Epoch 39/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.5963e-04 - val_loss: 0.0694
Epoch 40/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.2070e-04 - val_loss: 0.0697
Epoch 41/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 9.3711e-05 - val_loss: 0.0692
Epoch 42/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.0329e-04 - val_loss: 0.0695
Epoch 43/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 8.2315e-05 - val_loss: 0.0694
Epoch 44/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 6.2199e-05 - val_loss: 0.0692
Epoch 45/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 5.4085e-05 - val_loss: 0.0692
Epoch 46/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 5.1810e-05 - val_loss: 0.0694
Epoch 47/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 4.6157e-05 - val_loss: 0.0693
```

```
Epoch 48/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3.4175e-05 - val_loss: 0.0693
Epoch 49/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 3.5505e-05 - val_loss: 0.0693
Epoch 50/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.7224e-05 - val_loss: 0.0692
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 145ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
```




```
Processing stock: META
Training LSTM model for stock: META
Epoch 1/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 2s 48ms/step - loss: 0.8490 - val_loss: 0.5039
```

```
Epoch 2/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 0.4454 - val_loss: 0.2270
Epoch 3/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.1755 - val_loss: 0.1114
Epoch 4/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0943 - val_loss: 0.0515
Epoch 5/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0574 - val_loss: 0.0384
Epoch 6/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0416 - val_loss: 0.0354
Epoch 7/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0384 - val_loss: 0.0332
Epoch 8/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0301 - val_loss: 0.0356
Epoch 9/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0209 - val_loss: 0.0291
Epoch 10/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0228 - val_loss: 0.0233
Epoch 11/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0187 - val_loss: 0.0210
Epoch 12/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0169 - val_loss: 0.0206
Epoch 13/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0194 - val_loss: 0.0209
Epoch 14/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0154 - val_loss: 0.0176
Epoch 15/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0152 - val_loss: 0.0189
Epoch 16/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0143 - val_loss: 0.0160
Epoch 17/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0135 - val_loss: 0.0182
Epoch 18/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0144 - val_loss: 0.0154
Epoch 19/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0151 - val_loss: 0.0156
Epoch 20/50
```

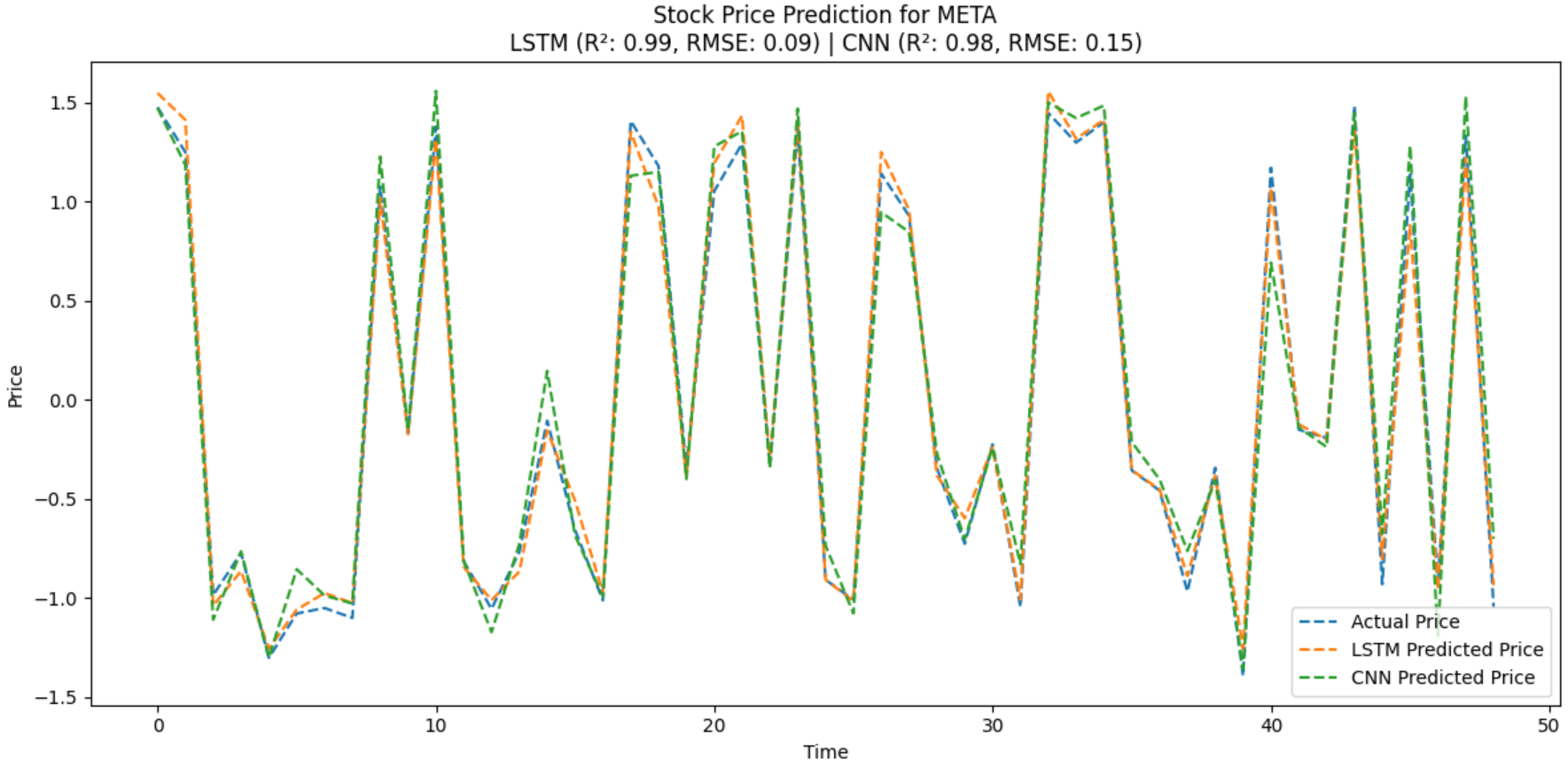
```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0114 - val_loss: 0.0143
Epoch 21/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0101 - val_loss: 0.0158
Epoch 22/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0137 - val_loss: 0.0123
Epoch 23/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0094 - val_loss: 0.0114
Epoch 24/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0110 - val_loss: 0.0125
Epoch 25/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0101 - val_loss: 0.0096
Epoch 26/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0093 - val_loss: 0.0143
Epoch 27/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0130 - val_loss: 0.0109
Epoch 28/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0105 - val_loss: 0.0114
Epoch 29/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0098 - val_loss: 0.0100
Epoch 30/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0134 - val_loss: 0.0129
Epoch 31/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0084 - val_loss: 0.0103
Epoch 32/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0102 - val_loss: 0.0102
Epoch 33/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0066 - val_loss: 0.0078
Epoch 34/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0097 - val_loss: 0.0089
Epoch 35/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0075 - val_loss: 0.0088
Epoch 36/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0082 - val_loss: 0.0110
Epoch 37/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0083 - val_loss: 0.0079
Epoch 38/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0077 - val_loss: 0.0125
```

```
Epoch 39/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0079 - val_loss: 0.0075
Epoch 40/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0063 - val_loss: 0.0078
Epoch 41/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0066 - val_loss: 0.0092
Epoch 42/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0072 - val_loss: 0.0093
Epoch 43/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0089 - val_loss: 0.0068
Epoch 44/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0085 - val_loss: 0.0088
Epoch 45/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0079 - val_loss: 0.0079
Epoch 46/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0054 - val_loss: 0.0070
Epoch 47/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0071 - val_loss: 0.0102
Epoch 48/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0065 - val_loss: 0.0067
Epoch 49/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0070 - val_loss: 0.0102
Epoch 50/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0063 - val_loss: 0.0080
Training CNN model for stock: META
Epoch 1/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 1.0366 - val_loss: 0.4977
Epoch 2/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.2832 - val_loss: 0.0611
Epoch 3/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0978 - val_loss: 0.0867
Epoch 4/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.1025 - val_loss: 0.0560
Epoch 5/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0467 - val_loss: 0.0518
Epoch 6/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0382 - val_loss: 0.0325
```

```
Epoch 7/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0221 - val_loss: 0.0277
Epoch 8/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0185 - val_loss: 0.0243
Epoch 9/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0132 - val_loss: 0.0261
Epoch 10/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0126 - val_loss: 0.0266
Epoch 11/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0094 - val_loss: 0.0221
Epoch 12/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0082 - val_loss: 0.0254
Epoch 13/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0077 - val_loss: 0.0231
Epoch 14/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0052 - val_loss: 0.0245
Epoch 15/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0048 - val_loss: 0.0222
Epoch 16/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0059 - val_loss: 0.0236
Epoch 17/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0035 - val_loss: 0.0236
Epoch 18/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0038 - val_loss: 0.0233
Epoch 19/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0031 - val_loss: 0.0232
Epoch 20/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0028 - val_loss: 0.0232
Epoch 21/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0021 - val_loss: 0.0234
Epoch 22/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0020 - val_loss: 0.0229
Epoch 23/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0020 - val_loss: 0.0233
Epoch 24/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0019 - val_loss: 0.0227
Epoch 25/50
```

```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0014 - val_loss: 0.0235
Epoch 26/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0014 - val_loss: 0.0230
Epoch 27/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0013 - val_loss: 0.0237
Epoch 28/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0013 - val_loss: 0.0230
Epoch 29/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0012 - val_loss: 0.0237
Epoch 30/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 9.9607e-04 - val_loss: 0.0229
Epoch 31/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 9.4703e-04 - val_loss: 0.0232
Epoch 32/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 7.0176e-04 - val_loss: 0.0230
Epoch 33/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 7.2254e-04 - val_loss: 0.0236
Epoch 34/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 7.3337e-04 - val_loss: 0.0230
Epoch 35/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.0450e-04 - val_loss: 0.0230
Epoch 36/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.5037e-04 - val_loss: 0.0233
Epoch 37/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 5.8419e-04 - val_loss: 0.0229
Epoch 38/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 6.3032e-04 - val_loss: 0.0234
Epoch 39/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 4.2548e-04 - val_loss: 0.0231
Epoch 40/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 3.9619e-04 - val_loss: 0.0230
Epoch 41/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.0132e-04 - val_loss: 0.0233
Epoch 42/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.9446e-04 - val_loss: 0.0232
Epoch 43/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3.3737e-04 - val_loss: 0.0230
```

```
Epoch 44/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3.2072e-04 - val_loss: 0.0232
Epoch 45/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.5308e-04 - val_loss: 0.0232
Epoch 46/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3.3969e-04 - val_loss: 0.0229
Epoch 47/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.8680e-04 - val_loss: 0.0232
Epoch 48/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.2816e-04 - val_loss: 0.0228
Epoch 49/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2.1733e-04 - val_loss: 0.0229
Epoch 50/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.7593e-04 - val_loss: 0.0230
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 158ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
```




```
Processing stock: MSFT
Training LSTM model for stock: MSFT
Epoch 1/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 2s 50ms/step - loss: 0.9829 - val_loss: 0.4530
Epoch 2/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.5578 - val_loss: 0.2507
Epoch 3/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.2895 - val_loss: 0.2175
Epoch 4/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 0.2503 - val_loss: 0.1182
Epoch 5/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.1389 - val_loss: 0.0723
```

```
Epoch 6/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0895 - val_loss: 0.0754
Epoch 7/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0869 - val_loss: 0.0621
Epoch 8/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0767 - val_loss: 0.0482
Epoch 9/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0756 - val_loss: 0.0460
Epoch 10/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0610 - val_loss: 0.0508
Epoch 11/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0520 - val_loss: 0.0329
Epoch 12/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0453 - val_loss: 0.0288
Epoch 13/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0454 - val_loss: 0.0319
Epoch 14/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0443 - val_loss: 0.0260
Epoch 15/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0364 - val_loss: 0.0260
Epoch 16/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0375 - val_loss: 0.0252
Epoch 17/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0372 - val_loss: 0.0231
Epoch 18/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0370 - val_loss: 0.0242
Epoch 19/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0311 - val_loss: 0.0217
Epoch 20/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0285 - val_loss: 0.0218
Epoch 21/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0288 - val_loss: 0.0205
Epoch 22/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0268 - val_loss: 0.0199
Epoch 23/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0230 - val_loss: 0.0194
Epoch 24/50
```

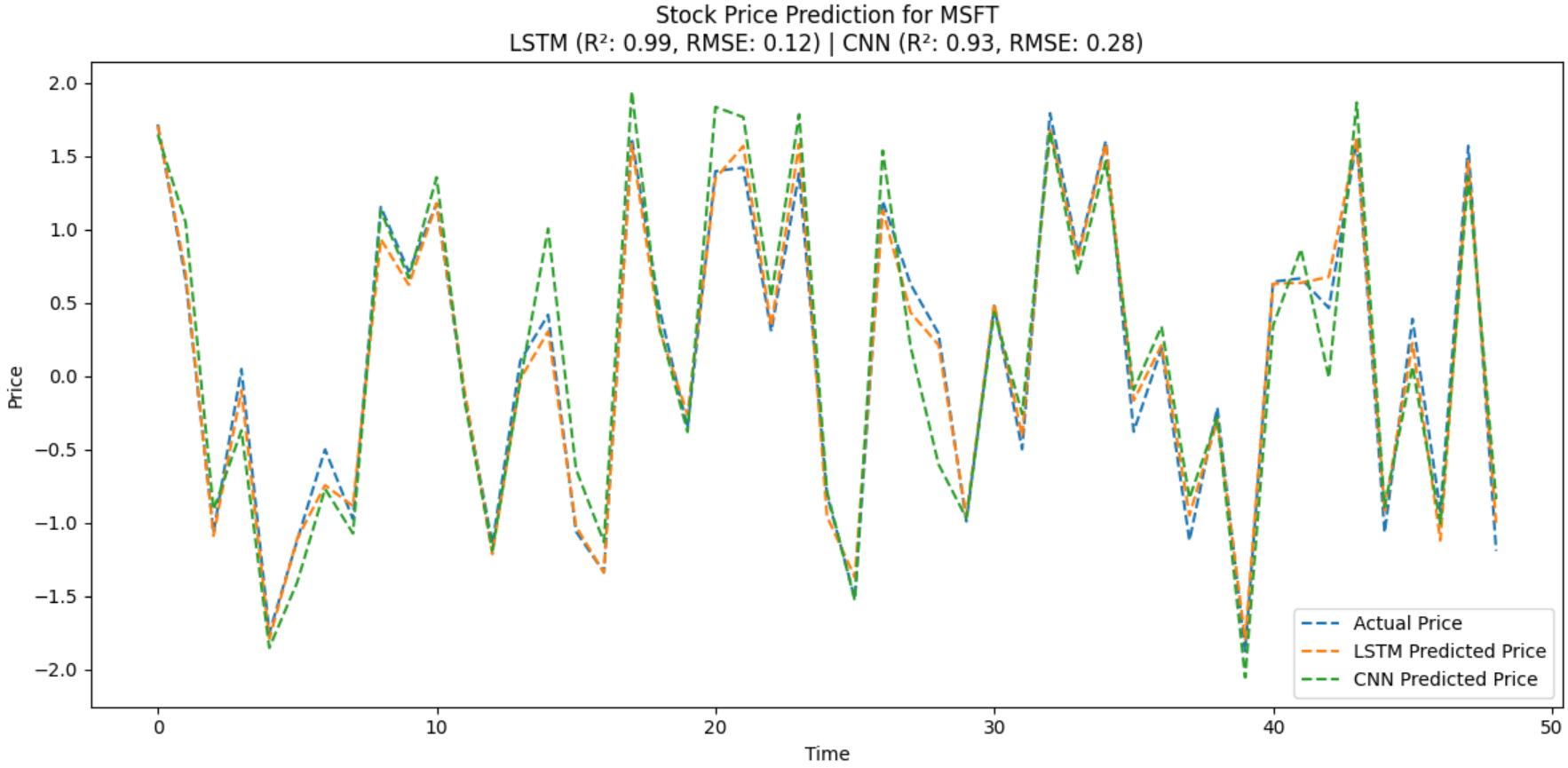
```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0229 - val_loss: 0.0204
Epoch 25/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0277 - val_loss: 0.0190
Epoch 26/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0277 - val_loss: 0.0186
Epoch 27/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0254 - val_loss: 0.0183
Epoch 28/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0244 - val_loss: 0.0188
Epoch 29/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0238 - val_loss: 0.0185
Epoch 30/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0208 - val_loss: 0.0176
Epoch 31/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0222 - val_loss: 0.0188
Epoch 32/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0195 - val_loss: 0.0155
Epoch 33/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0222 - val_loss: 0.0167
Epoch 34/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0223 - val_loss: 0.0150
Epoch 35/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0199 - val_loss: 0.0152
Epoch 36/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0195 - val_loss: 0.0173
Epoch 37/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0223 - val_loss: 0.0164
Epoch 38/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0203 - val_loss: 0.0161
Epoch 39/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0160 - val_loss: 0.0162
Epoch 40/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0207 - val_loss: 0.0150
Epoch 41/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0172 - val_loss: 0.0152
Epoch 42/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0176 - val_loss: 0.0152
```

```
Epoch 43/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0189 - val_loss: 0.0144
Epoch 44/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0186 - val_loss: 0.0146
Epoch 45/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0168 - val_loss: 0.0159
Epoch 46/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0200 - val_loss: 0.0151
Epoch 47/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0202 - val_loss: 0.0160
Epoch 48/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0193 - val_loss: 0.0159
Epoch 49/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0172 - val_loss: 0.0171
Epoch 50/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - loss: 0.0167 - val_loss: 0.0134
Training CNN model for stock: MSFT
Epoch 1/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.7671 - val_loss: 0.3822
Epoch 2/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.2082 - val_loss: 0.1474
Epoch 3/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.1022 - val_loss: 0.1734
Epoch 4/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.1013 - val_loss: 0.1193
Epoch 5/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0487 - val_loss: 0.1013
Epoch 6/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0428 - val_loss: 0.1354
Epoch 7/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0322 - val_loss: 0.1012
Epoch 8/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0280 - val_loss: 0.0806
Epoch 9/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0199 - val_loss: 0.1051
Epoch 10/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0159 - val_loss: 0.0790
```

```
Epoch 11/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0151 - val_loss: 0.0930
Epoch 12/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0090 - val_loss: 0.0839
Epoch 13/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0081 - val_loss: 0.0839
Epoch 14/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0057 - val_loss: 0.0879
Epoch 15/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0050 - val_loss: 0.0796
Epoch 16/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0038 - val_loss: 0.0852
Epoch 17/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0031 - val_loss: 0.0817
Epoch 18/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0019 - val_loss: 0.0826
Epoch 19/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0017 - val_loss: 0.0820
Epoch 20/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0015 - val_loss: 0.0838
Epoch 21/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0014 - val_loss: 0.0814
Epoch 22/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0011 - val_loss: 0.0818
Epoch 23/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 9.4335e-04 - val_loss: 0.0823
Epoch 24/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 5.8565e-04 - val_loss: 0.0814
Epoch 25/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 7.5730e-04 - val_loss: 0.0840
Epoch 26/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 5.9023e-04 - val_loss: 0.0825
Epoch 27/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 6.2520e-04 - val_loss: 0.0821
Epoch 28/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 4.8003e-04 - val_loss: 0.0824
Epoch 29/50
```

```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 4.1061e-04 - val_loss: 0.0804
Epoch 30/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.6698e-04 - val_loss: 0.0815
Epoch 31/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.0990e-04 - val_loss: 0.0809
Epoch 32/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.3539e-04 - val_loss: 0.0816
Epoch 33/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.3094e-04 - val_loss: 0.0813
Epoch 34/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.2453e-04 - val_loss: 0.0819
Epoch 35/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.3070e-04 - val_loss: 0.0809
Epoch 36/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 8.1276e-05 - val_loss: 0.0817
Epoch 37/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 6.5259e-05 - val_loss: 0.0807
Epoch 38/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3.6985e-05 - val_loss: 0.0808
Epoch 39/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.9596e-05 - val_loss: 0.0809
Epoch 40/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.5216e-05 - val_loss: 0.0809
Epoch 41/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.5137e-05 - val_loss: 0.0812
Epoch 42/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.5666e-05 - val_loss: 0.0806
Epoch 43/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.2688e-05 - val_loss: 0.0809
Epoch 44/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.6278e-05 - val_loss: 0.0806
Epoch 45/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.4528e-05 - val_loss: 0.0808
Epoch 46/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.0956e-05 - val_loss: 0.0807
Epoch 47/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 9.7618e-06 - val_loss: 0.0805
```

```
Epoch 48/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 9.1137e-06 - val_loss: 0.0809
Epoch 49/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 8.2446e-06 - val_loss: 0.0805
Epoch 50/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 6.3965e-06 - val_loss: 0.0808
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 148ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
```




```
Processing stock: PYPL
Training LSTM model for stock: PYPL
Epoch 1/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 2s 50ms/step - loss: 0.9496 - val_loss: 0.5875
```

```
Epoch 2/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.5145 - val_loss: 0.3181
Epoch 3/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.2788 - val_loss: 0.0987
Epoch 4/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.1184 - val_loss: 0.0494
Epoch 5/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - loss: 0.0574 - val_loss: 0.0599
Epoch 6/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0427 - val_loss: 0.0485
Epoch 7/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0453 - val_loss: 0.0255
Epoch 8/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0177 - val_loss: 0.0225
Epoch 9/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0188 - val_loss: 0.0184
Epoch 10/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0086 - val_loss: 0.0133
Epoch 11/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0122 - val_loss: 0.0121
Epoch 12/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0087 - val_loss: 0.0150
Epoch 13/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0069 - val_loss: 0.0117
Epoch 14/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0059 - val_loss: 0.0103
Epoch 15/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0076 - val_loss: 0.0107
Epoch 16/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0053 - val_loss: 0.0103
Epoch 17/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0053 - val_loss: 0.0106
Epoch 18/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0069 - val_loss: 0.0096
Epoch 19/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0047 - val_loss: 0.0095
Epoch 20/50
```

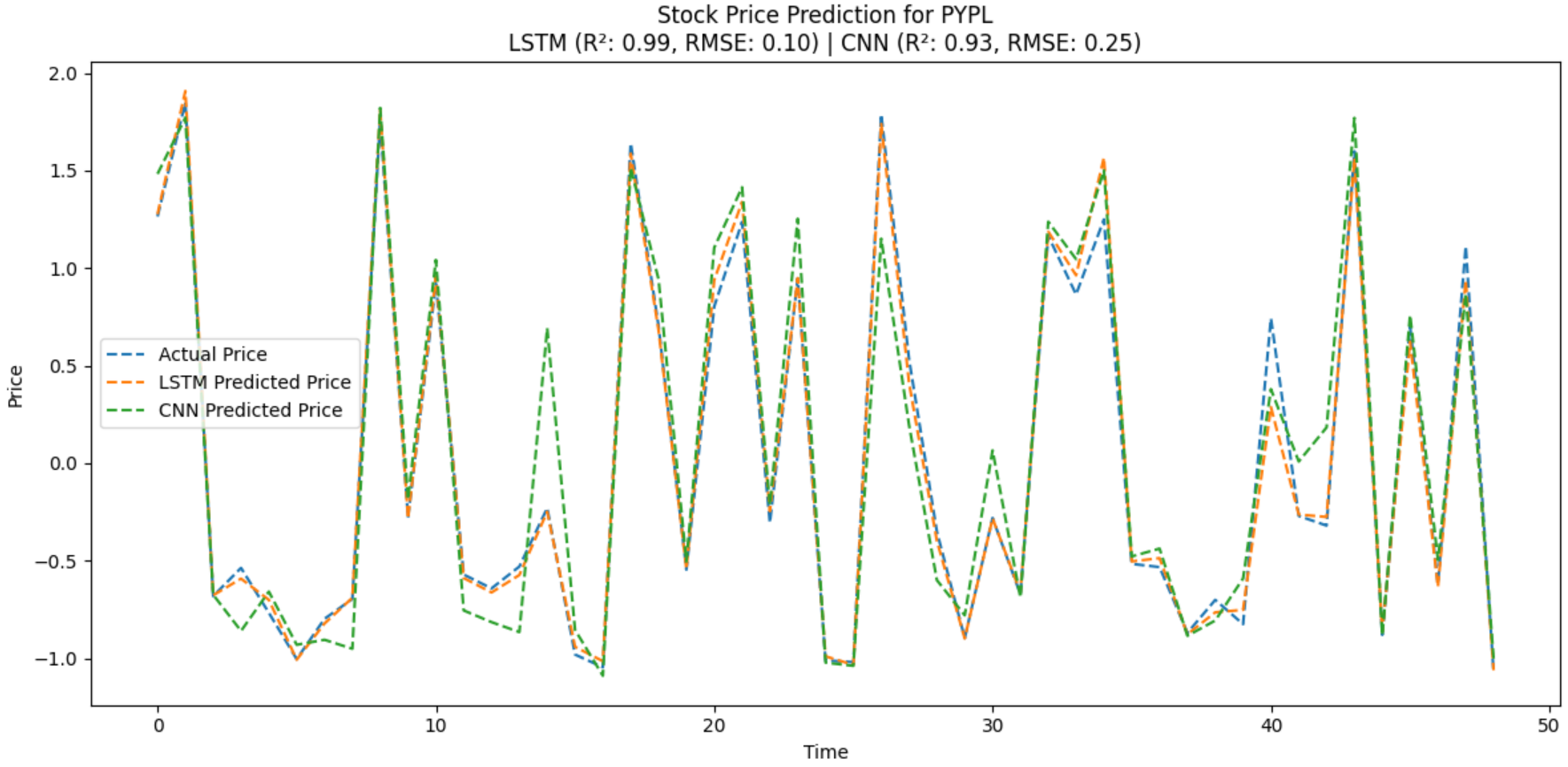
```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0047 - val_loss: 0.0109
Epoch 21/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0053 - val_loss: 0.0092
Epoch 22/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0052 - val_loss: 0.0093
Epoch 23/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0045 - val_loss: 0.0103
Epoch 24/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0044 - val_loss: 0.0091
Epoch 25/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0052 - val_loss: 0.0088
Epoch 26/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0049 - val_loss: 0.0102
Epoch 27/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0039 - val_loss: 0.0093
Epoch 28/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0041 - val_loss: 0.0087
Epoch 29/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0038 - val_loss: 0.0097
Epoch 30/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0035 - val_loss: 0.0093
Epoch 31/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0040 - val_loss: 0.0092
Epoch 32/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0040 - val_loss: 0.0092
Epoch 33/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0033 - val_loss: 0.0091
Epoch 34/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0040 - val_loss: 0.0094
Epoch 35/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0030 - val_loss: 0.0092
Epoch 36/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0043 - val_loss: 0.0091
Epoch 37/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0035 - val_loss: 0.0089
Epoch 38/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0042 - val_loss: 0.0091
```

```
Epoch 39/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0025 - val_loss: 0.0096
Epoch 40/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0032 - val_loss: 0.0088
Epoch 41/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0039 - val_loss: 0.0098
Epoch 42/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0028 - val_loss: 0.0087
Epoch 43/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0035 - val_loss: 0.0095
Epoch 44/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0036 - val_loss: 0.0089
Epoch 45/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0034 - val_loss: 0.0091
Epoch 46/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0022 - val_loss: 0.0092
Epoch 47/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0026 - val_loss: 0.0091
Epoch 48/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0031 - val_loss: 0.0093
Epoch 49/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0030 - val_loss: 0.0097
Epoch 50/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0033 - val_loss: 0.0091
Training CNN model for stock: PYPL
Epoch 1/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 0.3192 - val_loss: 0.1976
Epoch 2/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.1250 - val_loss: 0.1354
Epoch 3/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0726 - val_loss: 0.1018
Epoch 4/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0311 - val_loss: 0.0700
Epoch 5/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0244 - val_loss: 0.0698
Epoch 6/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0134 - val_loss: 0.0745
```

```
Epoch 7/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0137 - val_loss: 0.0640
Epoch 8/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0068 - val_loss: 0.0617
Epoch 9/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0061 - val_loss: 0.0636
Epoch 10/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0047 - val_loss: 0.0650
Epoch 11/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0027 - val_loss: 0.0606
Epoch 12/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0026 - val_loss: 0.0640
Epoch 13/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0019 - val_loss: 0.0631
Epoch 14/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0017 - val_loss: 0.0607
Epoch 15/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0012 - val_loss: 0.0638
Epoch 16/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 8.4629e-04 - val_loss: 0.0625
Epoch 17/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 6.3914e-04 - val_loss: 0.0632
Epoch 18/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 6.7001e-04 - val_loss: 0.0622
Epoch 19/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 4.3736e-04 - val_loss: 0.0636
Epoch 20/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.4944e-04 - val_loss: 0.0626
Epoch 21/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.8688e-04 - val_loss: 0.0633
Epoch 22/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 2.3878e-04 - val_loss: 0.0627
Epoch 23/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.9264e-04 - val_loss: 0.0631
Epoch 24/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.3702e-04 - val_loss: 0.0627
Epoch 25/50
```

```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.4027e-04 - val_loss: 0.0633
Epoch 26/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.0898e-04 - val_loss: 0.0628
Epoch 27/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 1.0112e-04 - val_loss: 0.0627
Epoch 28/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 9.1381e-05 - val_loss: 0.0634
Epoch 29/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 7.7689e-05 - val_loss: 0.0626
Epoch 30/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 6.4405e-05 - val_loss: 0.0632
Epoch 31/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 5.7238e-05 - val_loss: 0.0628
Epoch 32/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 3.5409e-05 - val_loss: 0.0632
Epoch 33/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.5907e-05 - val_loss: 0.0631
Epoch 34/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.9925e-05 - val_loss: 0.0631
Epoch 35/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.5797e-05 - val_loss: 0.0632
Epoch 36/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 1.5768e-05 - val_loss: 0.0629
Epoch 37/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.6556e-05 - val_loss: 0.0632
Epoch 38/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.0040e-05 - val_loss: 0.0630
Epoch 39/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 7.8322e-06 - val_loss: 0.0633
Epoch 40/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 1.0116e-05 - val_loss: 0.0631
Epoch 41/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 5.7617e-06 - val_loss: 0.0630
Epoch 42/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 6.8035e-06 - val_loss: 0.0634
Epoch 43/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 5.9993e-06 - val_loss: 0.0630
```

```
Epoch 44/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 3.9815e-06 - val_loss: 0.0632
Epoch 45/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2.7551e-06 - val_loss: 0.0631
Epoch 46/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.7539e-06 - val_loss: 0.0631
Epoch 47/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.2788e-06 - val_loss: 0.0631
Epoch 48/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.2178e-06 - val_loss: 0.0631
Epoch 49/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 8.4079e-07 - val_loss: 0.0631
Epoch 50/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 4.3094e-07 - val_loss: 0.0631
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 176ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
```




```
Processing stock: TSLA
Training LSTM model for stock: TSLA
Epoch 1/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 15s 82ms/step - loss: 0.8852 - val_loss: 0.5060
Epoch 2/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.5672 - val_loss: 0.3112
Epoch 3/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.2686 - val_loss: 0.2347
Epoch 4/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.2181 - val_loss: 0.1818
Epoch 5/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.1514 - val_loss: 0.1609
```

```
Epoch 6/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.1507 - val_loss: 0.1348
Epoch 7/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.1184 - val_loss: 0.1089
Epoch 8/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0922 - val_loss: 0.1049
Epoch 9/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0800 - val_loss: 0.0965
Epoch 10/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0760 - val_loss: 0.0855
Epoch 11/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0702 - val_loss: 0.0842
Epoch 12/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0717 - val_loss: 0.0781
Epoch 13/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0587 - val_loss: 0.0696
Epoch 14/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0667 - val_loss: 0.0847
Epoch 15/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0624 - val_loss: 0.0652
Epoch 16/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0623 - val_loss: 0.0644
Epoch 17/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0543 - val_loss: 0.0715
Epoch 18/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0459 - val_loss: 0.0609
Epoch 19/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0525 - val_loss: 0.0618
Epoch 20/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0558 - val_loss: 0.0634
Epoch 21/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0468 - val_loss: 0.0586
Epoch 22/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0513 - val_loss: 0.0818
Epoch 23/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0455 - val_loss: 0.0615
Epoch 24/50
```

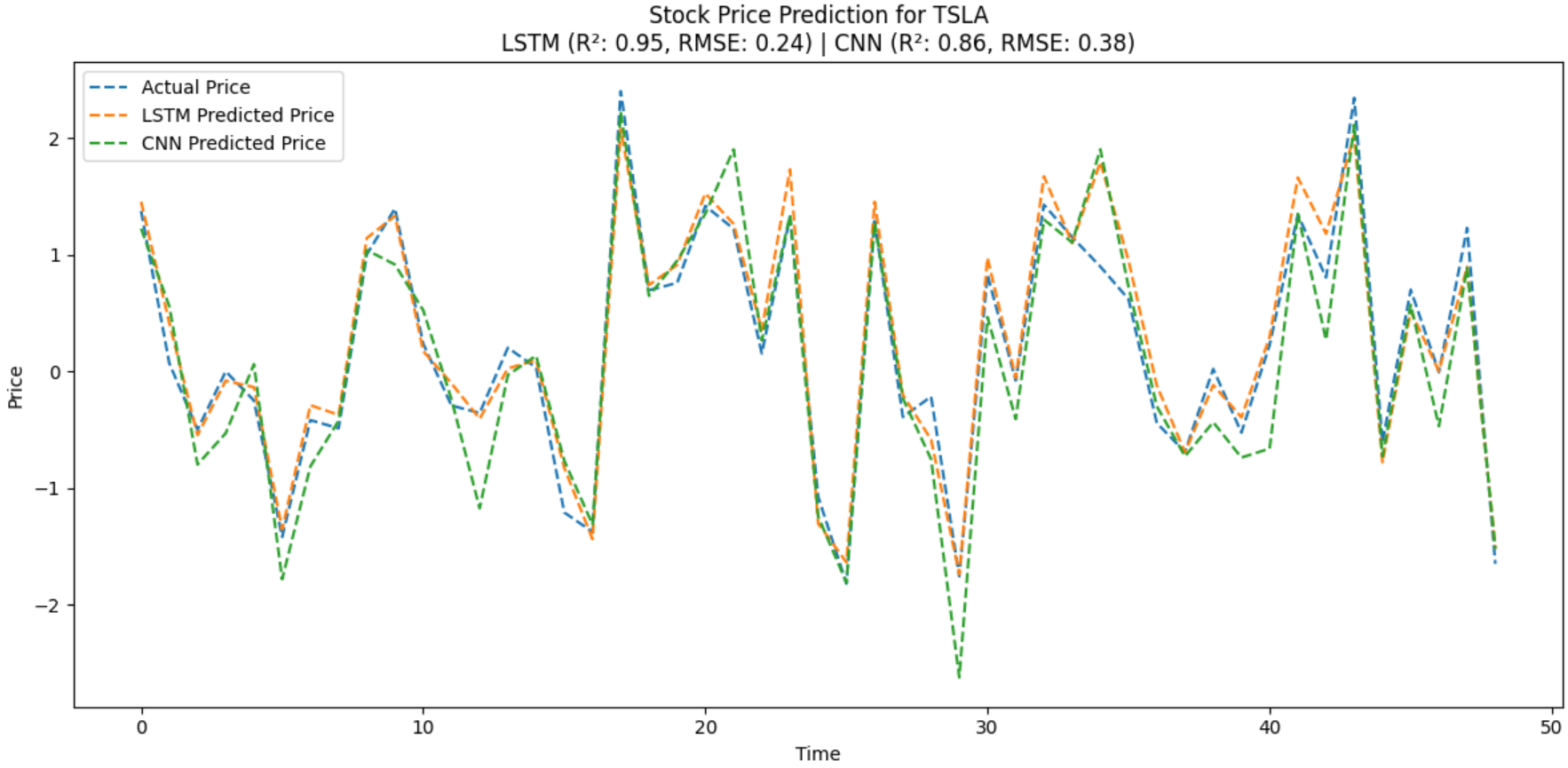
```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0468 - val_loss: 0.0562
Epoch 25/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0496 - val_loss: 0.0704
Epoch 26/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0446 - val_loss: 0.0548
Epoch 27/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0460 - val_loss: 0.0556
Epoch 28/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0430 - val_loss: 0.0609
Epoch 29/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0411 - val_loss: 0.0551
Epoch 30/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0446 - val_loss: 0.0581
Epoch 31/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0534 - val_loss: 0.0677
Epoch 32/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0476 - val_loss: 0.0565
Epoch 33/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0472 - val_loss: 0.0539
Epoch 34/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0406 - val_loss: 0.0649
Epoch 35/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0478 - val_loss: 0.0511
Epoch 36/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0345 - val_loss: 0.0535
Epoch 37/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0359 - val_loss: 0.0667
Epoch 38/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 0.0433 - val_loss: 0.0640
Epoch 39/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0466 - val_loss: 0.0611
Epoch 40/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0418 - val_loss: 0.0545
Epoch 41/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0422 - val_loss: 0.0574
Epoch 42/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0370 - val_loss: 0.0515
```

```
Epoch 43/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0399 - val_loss: 0.0511
Epoch 44/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0340 - val_loss: 0.0506
Epoch 45/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0328 - val_loss: 0.0633
Epoch 46/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0375 - val_loss: 0.0477
Epoch 47/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0307 - val_loss: 0.0618
Epoch 48/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0336 - val_loss: 0.0503
Epoch 49/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0311 - val_loss: 0.0542
Epoch 50/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0251 - val_loss: 0.0560
Training CNN model for stock: TSLA
Epoch 1/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.6424 - val_loss: 0.6026
Epoch 2/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.2719 - val_loss: 0.2450
Epoch 3/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.1707 - val_loss: 0.2534
Epoch 4/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.1006 - val_loss: 0.2310
Epoch 5/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0721 - val_loss: 0.1870
Epoch 6/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0747 - val_loss: 0.1804
Epoch 7/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0500 - val_loss: 0.1875
Epoch 8/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0388 - val_loss: 0.1437
Epoch 9/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0283 - val_loss: 0.1524
Epoch 10/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0238 - val_loss: 0.1617
```

```
Epoch 11/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.0240 - val_loss: 0.1403
Epoch 12/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0153 - val_loss: 0.1356
Epoch 13/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0162 - val_loss: 0.1418
Epoch 14/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0123 - val_loss: 0.1349
Epoch 15/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0086 - val_loss: 0.1414
Epoch 16/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0072 - val_loss: 0.1378
Epoch 17/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0066 - val_loss: 0.1372
Epoch 18/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0053 - val_loss: 0.1376
Epoch 19/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0044 - val_loss: 0.1447
Epoch 20/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0034 - val_loss: 0.1312
Epoch 21/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0035 - val_loss: 0.1486
Epoch 22/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0032 - val_loss: 0.1318
Epoch 23/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0023 - val_loss: 0.1455
Epoch 24/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0021 - val_loss: 0.1380
Epoch 25/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0017 - val_loss: 0.1394
Epoch 26/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0015 - val_loss: 0.1419
Epoch 27/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 8.6051e-04 - val_loss: 0.1382
Epoch 28/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 7.2108e-04 - val_loss: 0.1414
Epoch 29/50
```

```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 6.6557e-04 - val_loss: 0.1406
Epoch 30/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 5.4650e-04 - val_loss: 0.1413
Epoch 31/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 3.6830e-04 - val_loss: 0.1410
Epoch 32/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 2.9043e-04 - val_loss: 0.1404
Epoch 33/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2.1968e-04 - val_loss: 0.1421
Epoch 34/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.7240e-04 - val_loss: 0.1422
Epoch 35/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 1.3079e-04 - val_loss: 0.1434
Epoch 36/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 1.3989e-04 - val_loss: 0.1405
Epoch 37/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.0758e-04 - val_loss: 0.1423
Epoch 38/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 8.7771e-05 - val_loss: 0.1426
Epoch 39/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 5.9267e-05 - val_loss: 0.1418
Epoch 40/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.8562e-05 - val_loss: 0.1414
Epoch 41/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 4.7806e-05 - val_loss: 0.1433
Epoch 42/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3.5362e-05 - val_loss: 0.1426
Epoch 43/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.4468e-05 - val_loss: 0.1416
Epoch 44/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.4610e-05 - val_loss: 0.1434
Epoch 45/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.7586e-05 - val_loss: 0.1436
Epoch 46/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.4864e-05 - val_loss: 0.1427
Epoch 47/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.0466e-05 - val_loss: 0.1424
```

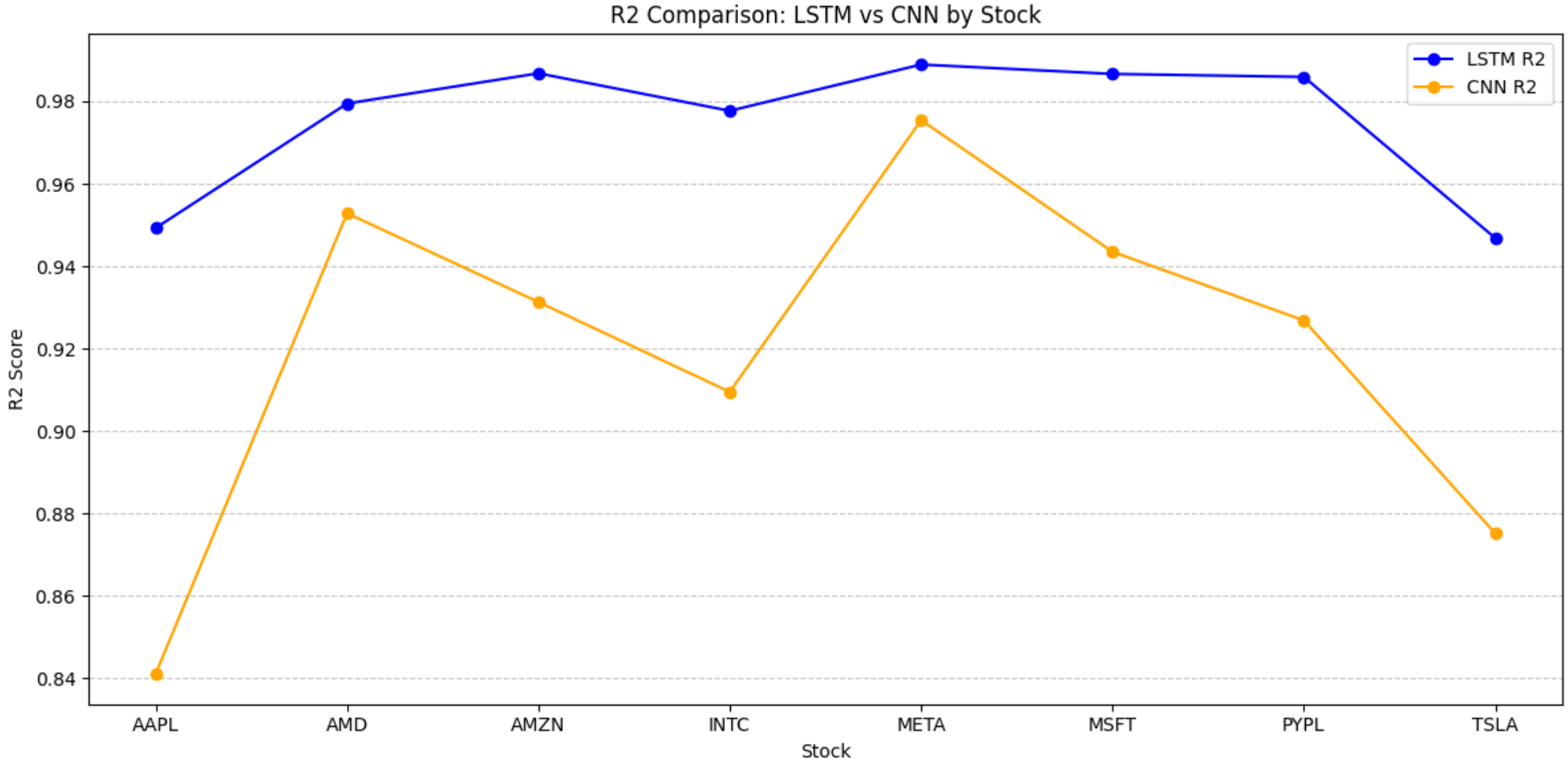
```
Epoch 48/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 6.3289e-06 - val_loss: 0.1433
Epoch 49/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 5.7975e-06 - val_loss: 0.1430
Epoch 50/50
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 7.8211e-06 - val_loss: 0.1427
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 218ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
```

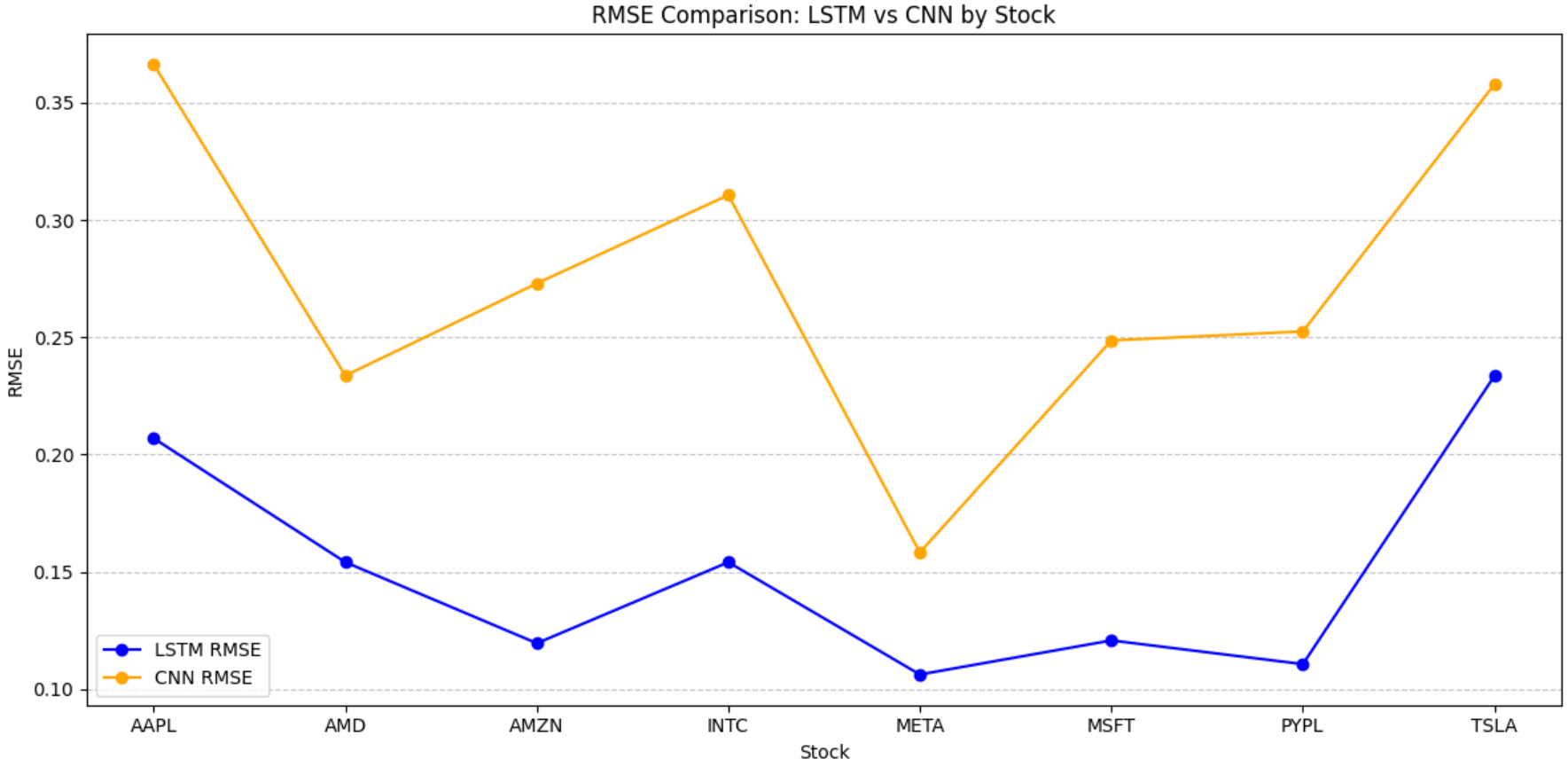


	Stock	LSTM_R²	LSTM_RMSE	CNN_R²	CNN_RMSE
0	AAPL	0.952192	0.200866	0.862276	0.340926
1	AMD	0.975816	0.167331	0.954620	0.229214
2	AMZN	0.987479	0.116530	0.950292	0.232185
3	INTC	0.981107	0.141905	0.935096	0.263018
4	META	0.992201	0.089246	0.977447	0.151767
5	MSFT	0.987727	0.115872	0.926114	0.284303
6	PYPL	0.989556	0.095407	0.927633	0.251143
7	TSLA	0.945357	0.236642	0.860770	0.377738

In [258…

```
# Line graph to compare R2 for LSTM and CNN
plt.figure(figsize=(12, 6))
plt.plot(results_df['Stock'], results_df['LSTM_R²'], marker='o', label='LSTM R2', color='blue')
plt.plot(results_df['Stock'], results_df['CNN_R²'], marker='o', label='CNN R2', color='orange')
plt.title("R2 Comparison: LSTM vs CNN by Stock")
plt.xlabel("Stock")
plt.ylabel("R2 Score")
plt.legend()
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

# Line graph to compare RMSE for LSTM and CNN
plt.figure(figsize=(12, 6))
plt.plot(results_df['Stock'], results_df['LSTM_RMSE'], marker='o', label='LSTM RMSE', color='blue')
plt.plot(results_df['Stock'], results_df['CNN_RMSE'], marker='o', label='CNN RMSE', color='orange')
plt.title("RMSE Comparison: LSTM vs CNN by Stock")
plt.xlabel("Stock")
plt.ylabel("RMSE")
plt.legend()
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```


R2 Comparison: LSTM vs CNN by Stock
LSTM R2
CNN R2
R2 Score
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.84
AAPL
AMD
AMZN
INTC
META
MSFT
PYPL
TSLA
Stock



LSTM vs CNN Stock Price Prediction

The performance of both **LSTM** and **CNN** models for predicting stock prices was evaluated using **R^2 (coefficient of determination)** and **RMSE (root mean squared error)** metrics across eight prominent stocks: AAPL, AMD, AMZN, INTC, META, MSFT, PYPL, and TSLA. Both models performed well overall, with LSTM consistently achieving lower RMSE and higher R^2 values compared to CNN

Stock-Wise Observations:

1. **AAPL**:
    - **LSTM**: (R^2 = 0.952), RMSE = 0.201
    - **CNN**: (R^2 = 0.862), RMSE = 0.341
    - **Conclusion**: LSTM performs significantly better for AAPL, capturing variance more effectively and reducing error.
2. **AMD**:
    - **LSTM**: (R^2 = 0.976), RMSE = 0.167
    - **CNN**: (R^2 = 0.955), RMSE = 0.229
    - **Conclusion**: LSTM outperforms CNN for AMD with better accuracy and lower error.
3. **AMZN**:
    - **LSTM**: (R^2 = 0.987), RMSE = 0.117
    - **CNN**: (R^2 = 0.950), RMSE = 0.232
    - **Conclusion**: LSTM provides better predictions for AMZN, with significantly improved metrics.
4. **INTC**:
    - **LSTM**: (R^2 = 0.981), RMSE = 0.142
    - **CNN**: (R^2 = 0.935), RMSE = 0.263
    - **Conclusion**: LSTM delivers more accurate predictions and lower error for INTC.
5. **META**:
    - **LSTM**: (R^2 = 0.992), RMSE = 0.089
    - **CNN**: (R^2 = 0.977), RMSE = 0.152
    - **Conclusion**: Both models perform exceptionally well for META, but LSTM achieves nearly perfect predictions.
6. **MSFT**:
    - **LSTM**: (R^2 = 0.988), RMSE = 0.116
    - **CNN**: (R^2 = 0.926), RMSE = 0.284
    - **Conclusion**: LSTM outperforms CNN for MSFT with significantly higher accuracy and lower error.

7. **PYPL**:
    - **LSTM**: (R^2 = 0.990), RMSE = 0.095
    - **CNN**: (R^2 = 0.928), RMSE = 0.251
    - **Conclusion**: LSTM provides better predictions for PYPL, reducing error and improving accuracy.
8. **TSLA**:
    - **LSTM**: (R^2 = 0.945), RMSE = 0.237
    - **CNN**: (R^2 = 0.861), RMSE = 0.378
    - **Conclusion**: TSLA remains challenging for both models, but LSTM handles its volatility better.

Model-Wise Observations:

1. **LSTM**:
    - **Performance**: LSTM consistently achieves higher (R^2) and lower RMSE across all stocks.
    - **Strengths**: Excels in capturing temporal dependencies, particularly for stocks with strong trends (e.g., META, AMZN, MSFT).
    - **Weaknesses**: Performs slightly less effectively for highly volatile stocks like TSLA, though still better than CNN.
2. **CNN**:
    - **Performance**: CNN achieves reasonable performance but lags behind LSTM in both metrics.
    - **Strengths**: Performs well for stocks with moderate volatility (e.g., AMD, META).
    - **Weaknesses**: Struggles with highly volatile stocks (e.g., TSLA) and stable stocks with clear temporal patterns (e.g., MSFT).

General Insights:

- **Stable Stocks (e.g., AAPL, MSFT)**:
    - LSTM provides superior predictions, achieving higher accuracy and lower error compared to CNN.

- **Volatile Stocks (e.g., TSLA, META)**:
  - LSTM outperforms CNN by better capturing the complexity of price movements, especially for volatile stocks.

Conclusion:

LSTM demonstrates superior performance in stock price prediction, achieving consistently higher (R^2) values and lower RMSE across all stocks. Its stability across both stable stocks (e.g., MSFT, AAPL) and volatile stocks (e.g., TSLA, META) highlights its suitability for modeling diverse stock behaviors. While CNN performs well in some cases, its overall accuracy and robustness are limited compared to LSTM.

For future enhancements, hybrid architectures combining LSTM and CNN could be explored to leverage the strengths of both models. Additionally, incorporating features like Google Trends, sentiment analysis, and macroeconomic indicators could further boost predictive accuracy.