**Stock Price Analysis and Automated Insight Generation Using Technical Indicators**

**By – Vasav Srivastava and Utkarsh Bhagat**

**Introduction**

Stock market prediction has attracted extensive attention from investors, researchers, and financial analysts. While machine learning and deep learning have gained popularity recently, technical indicators like Simple Moving Average (SMA), Relative Strength Index (RSI), and Moving Average Convergence Divergence (MACD) have been traditionally used to support trading decisions.

This project focuses on developing an automated system to fetch real-world stock data, calculate major technical indicators, visualize patterns, and generate financial insights using a Large Language Model (LLM).

Research Question: Can the integration of traditional technical indicators with automated language models accelerate and enhance stock analysis for investors?

Significance: Automating technical analysis democratizes financial tools for non-expert investors and saves significant time for professional analysts, supporting better decision-making.

**Related Work**

Technical indicators have been foundational tools for decades. Key literature highlights include:

* SMA (Simple Moving Average): Used to smooth price data, identify trends, and generate trading signals (Murphy, Technical Analysis of the Financial Markets).
* RSI (Relative Strength Index): Introduced by J. Welles Wilder in 1978, RSI measures the speed and change of price movements to identify overbought or oversold conditions.
* MACD (Moving Average Convergence Divergence): Developed by Gerald Appel, MACD detects momentum changes by comparing short-term and long-term EMAs.

Studies such as Patel et al. (2015) explored machine learning approaches but emphasized that hybrid models combining technical analysis and AI yield better results.

Gap Identified: Traditional technical analysis is manual, slow, and subjective. There is a need for automated interpretation to improve speed, consistency, and accessibility.

**Proposed Methodology**

The method involves two main parts:

1. Numerical Calculation of Indicators using Python:
   * SMA: SMA\_n(t) = (P(t) + P(t-1) + ... + P(t-n+1)) / n
   * RSI: RSI = 100 - (100 / (1 + RS)) where RS = (Average Gain) / (Average Loss)
   * MACD: MACD = EMA\_short(t) - EMA\_long(t) Signal Line = EMA\_MACD(9 periods)
2. Automated Insight Generation:
   * Use EleutherAI/gpt-neo-2.7B from Hugging Face to interpret the indicators and produce short financial analysis.

Tools & Libraries:

* yfinance for fetching data
* pandas, numpy for calculations
* matplotlib for visualization
* transformers for text generation

Install Necessary Libraries

!pip install yfinance pandas numpy matplotlib transformers

Import Libraries

import yfinance as yf  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
from transformers import pipeline

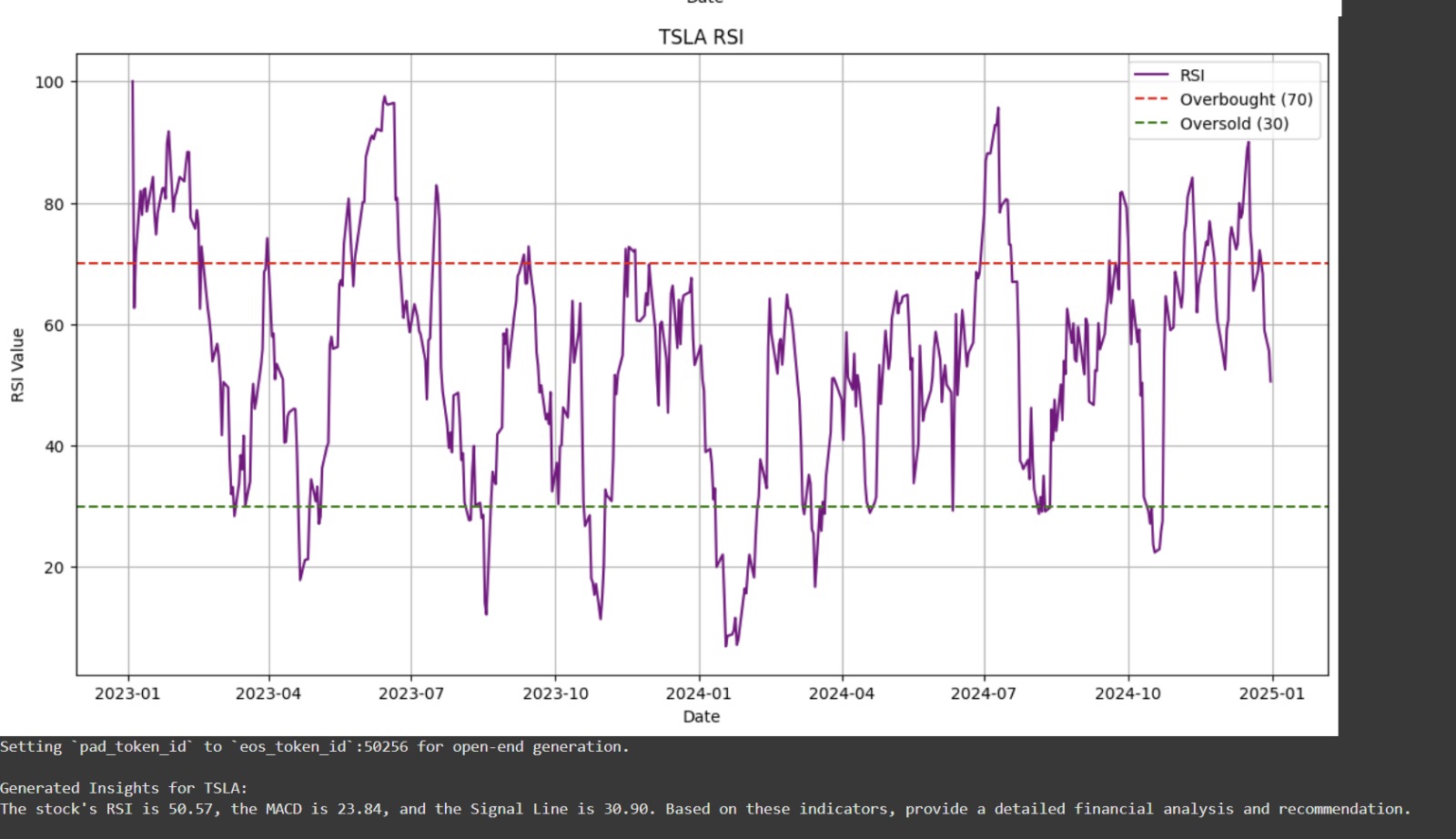
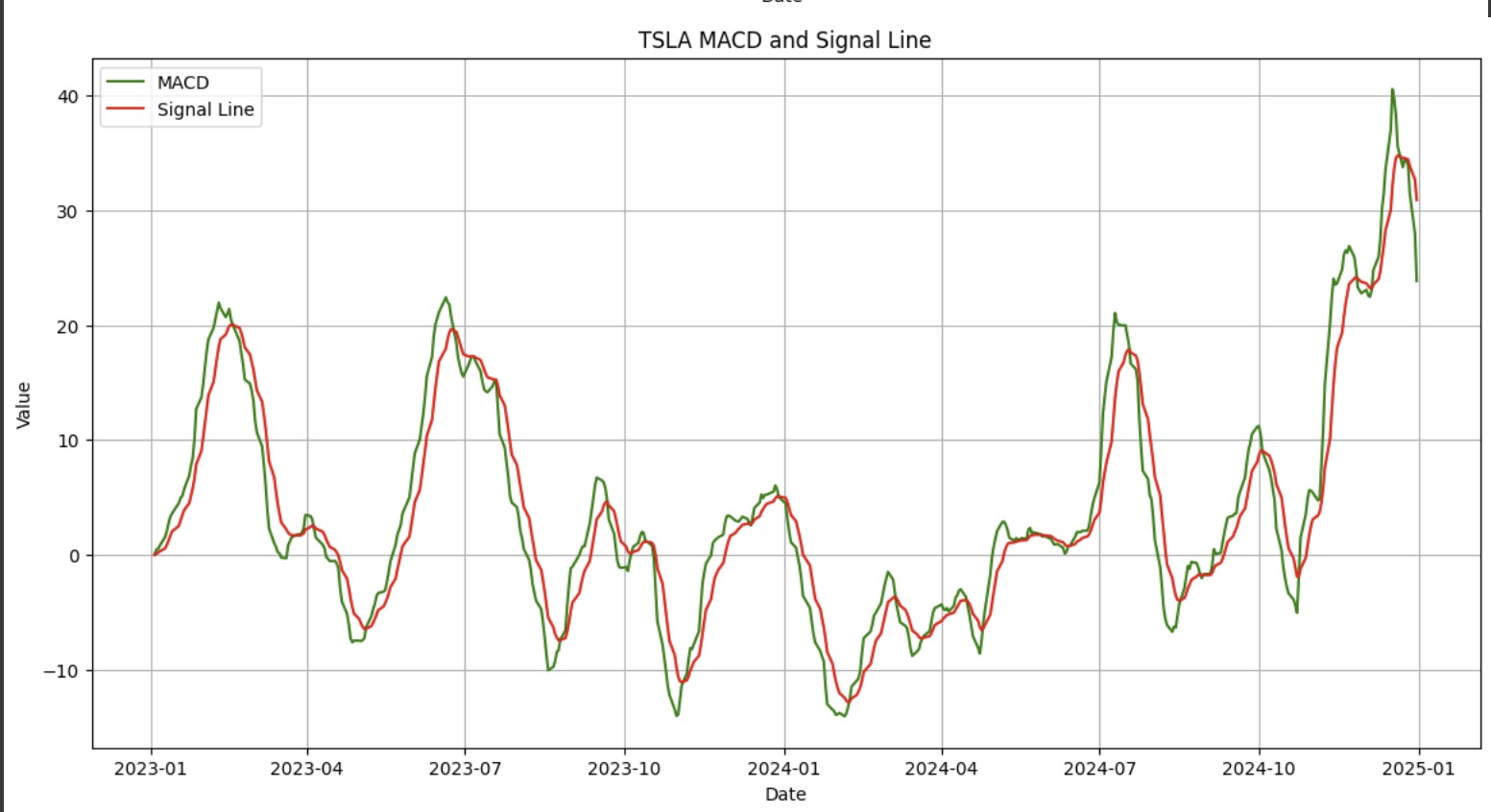
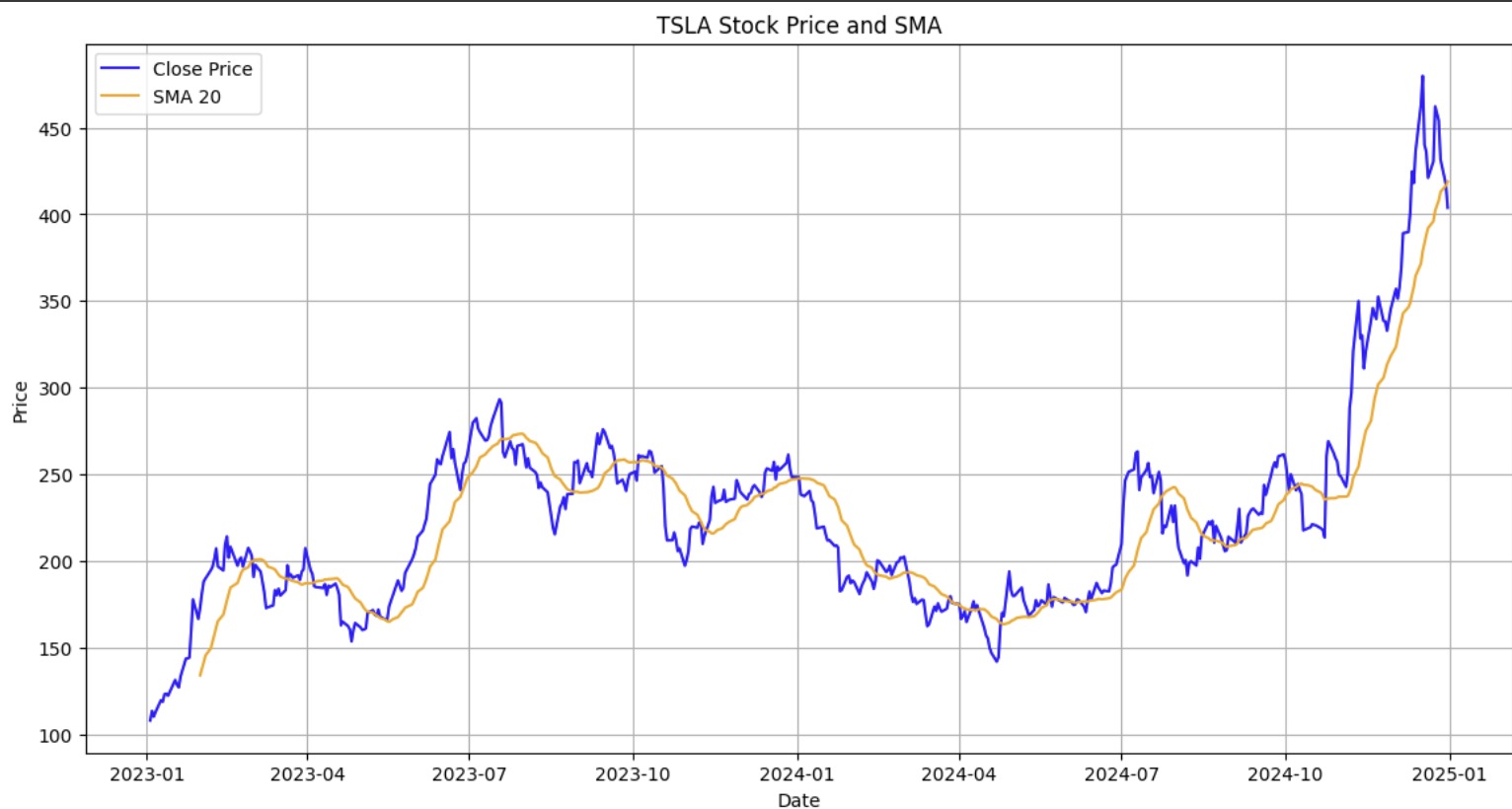
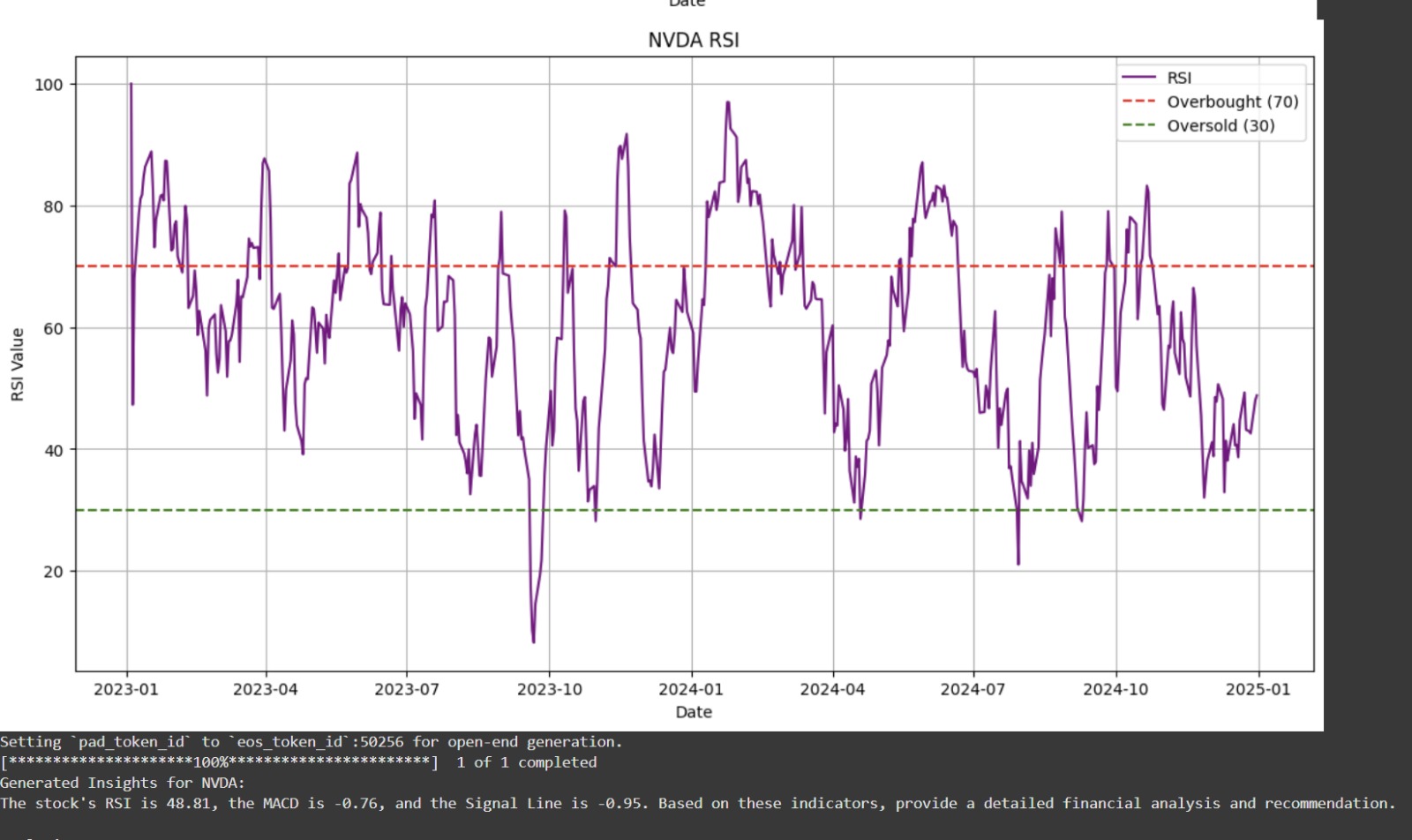
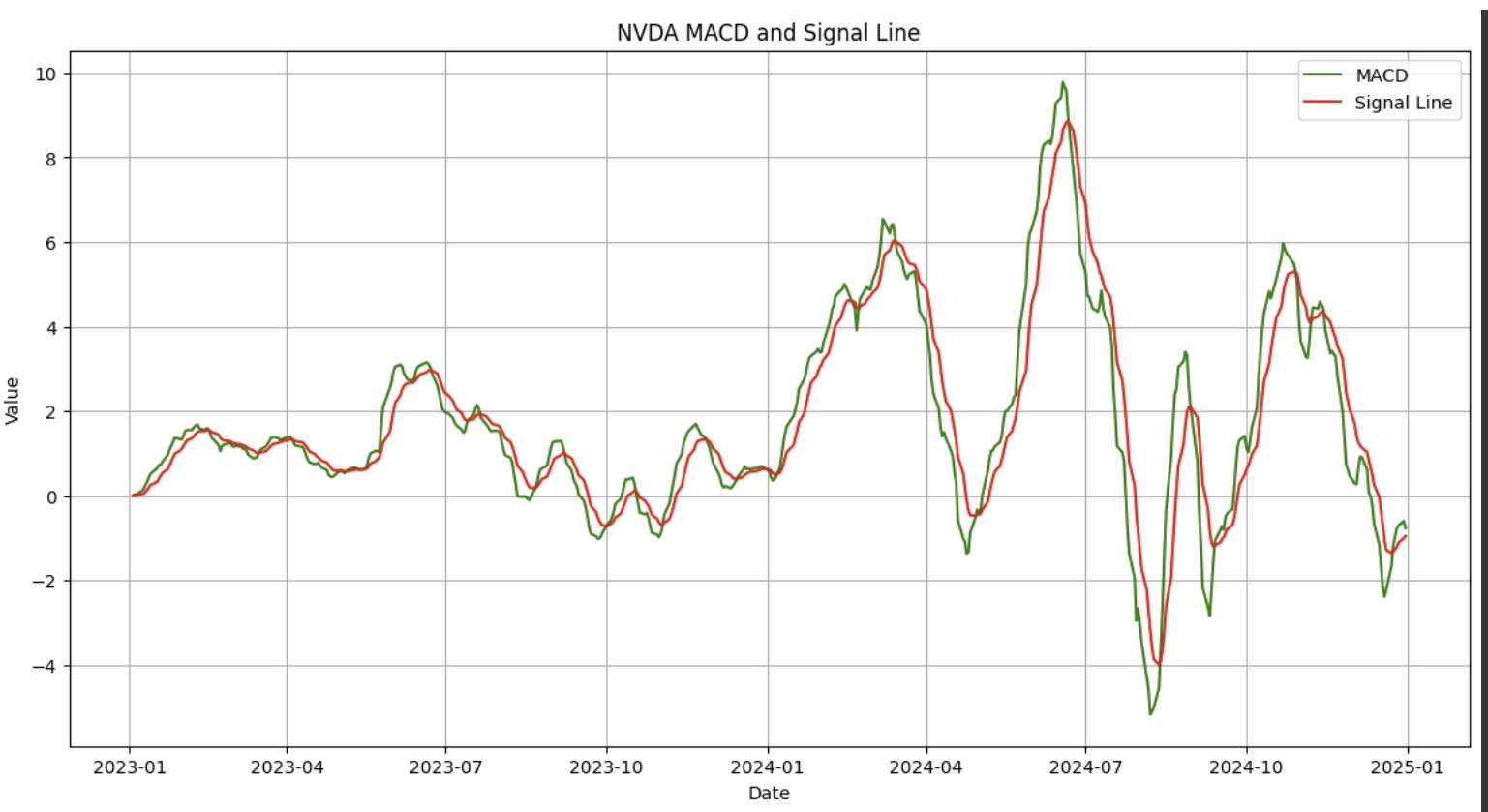
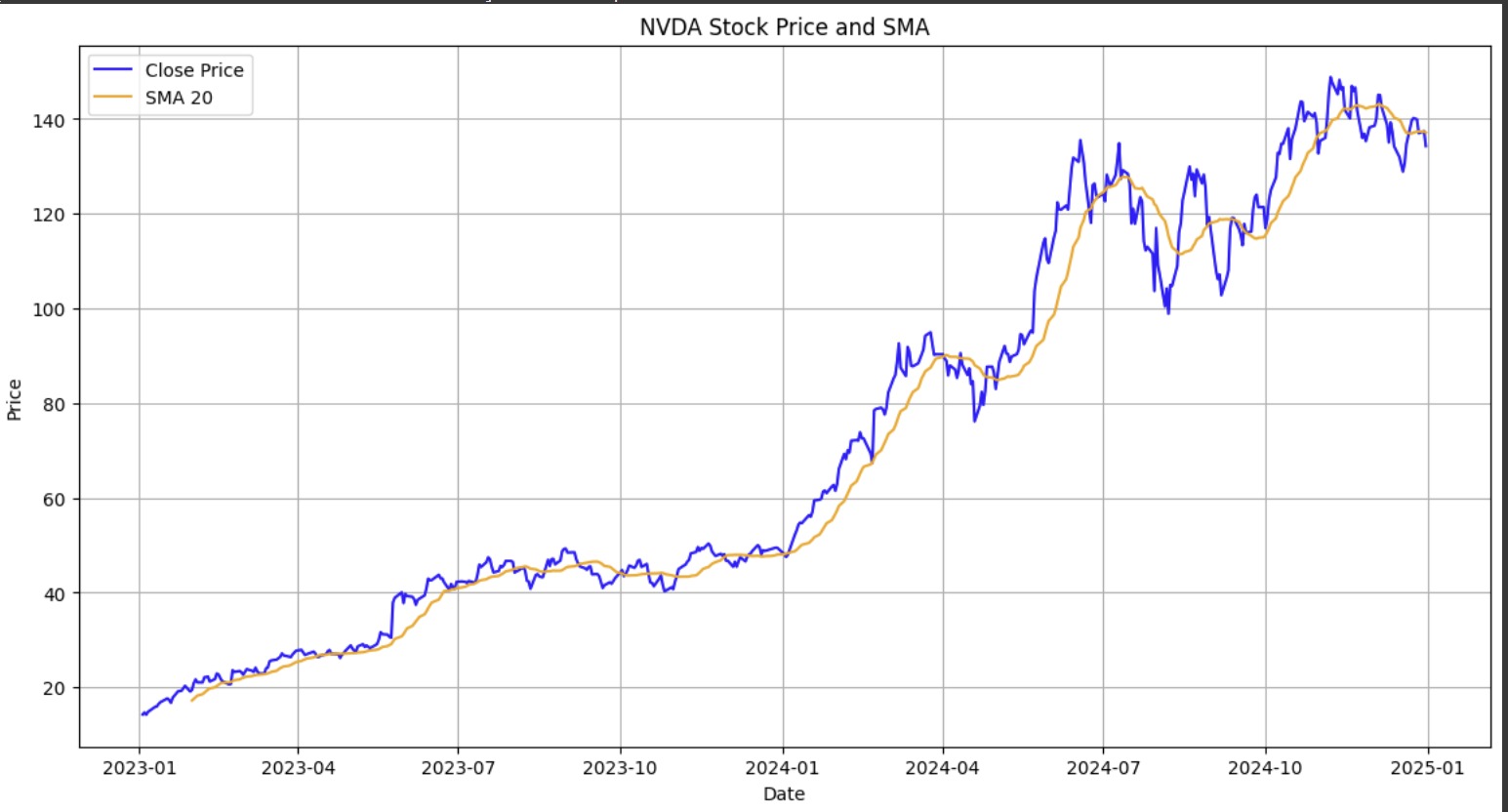
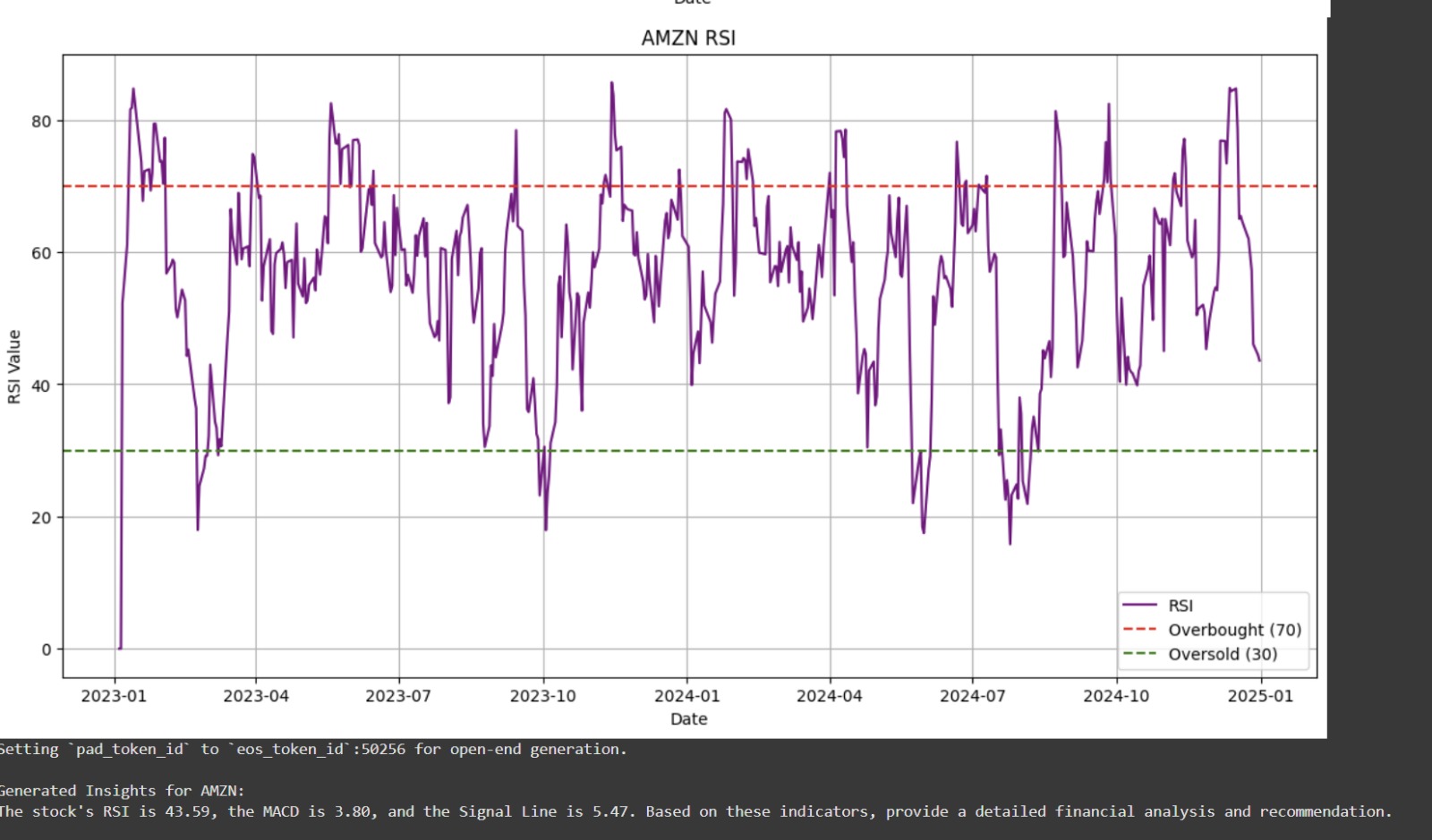
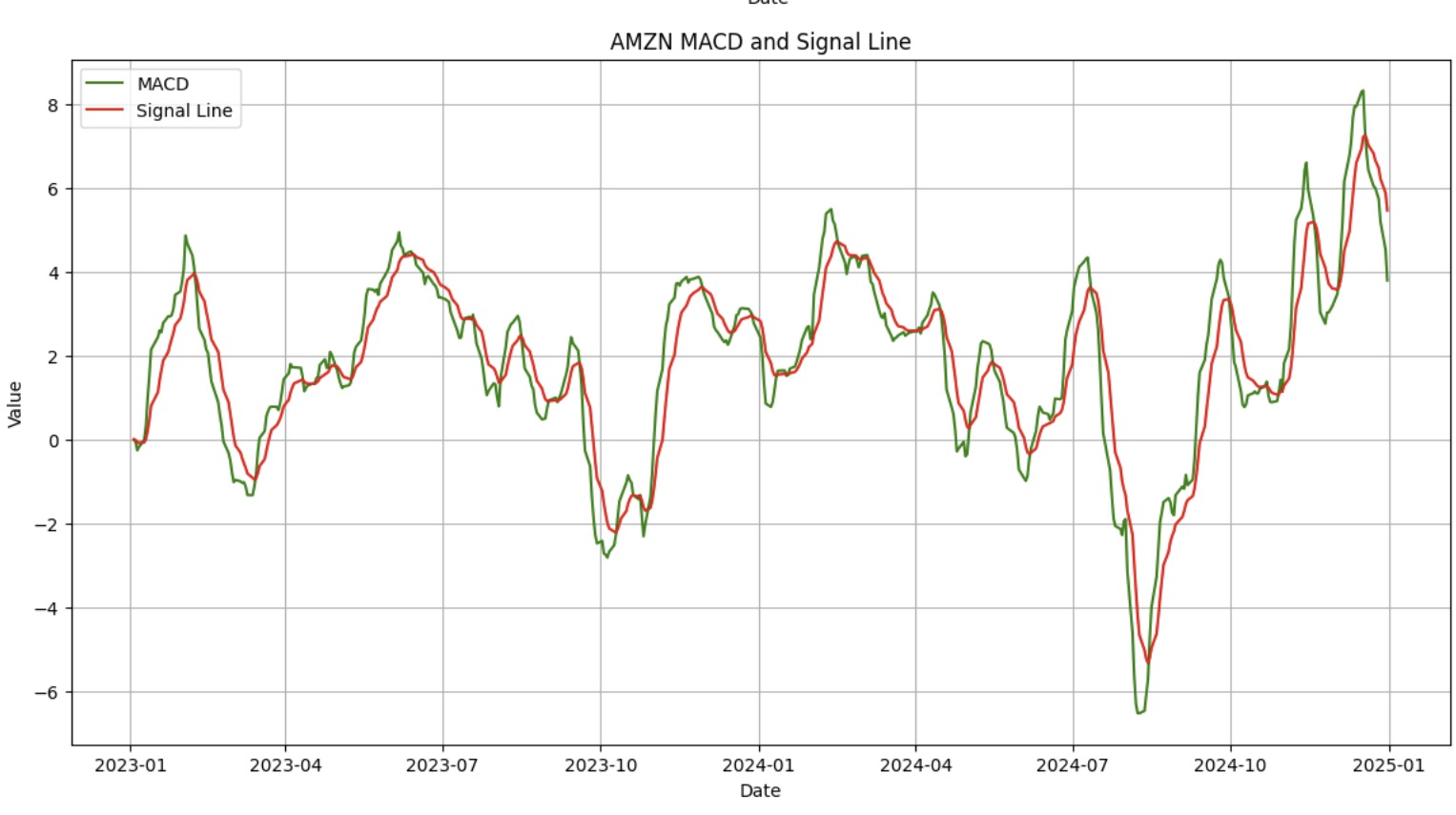
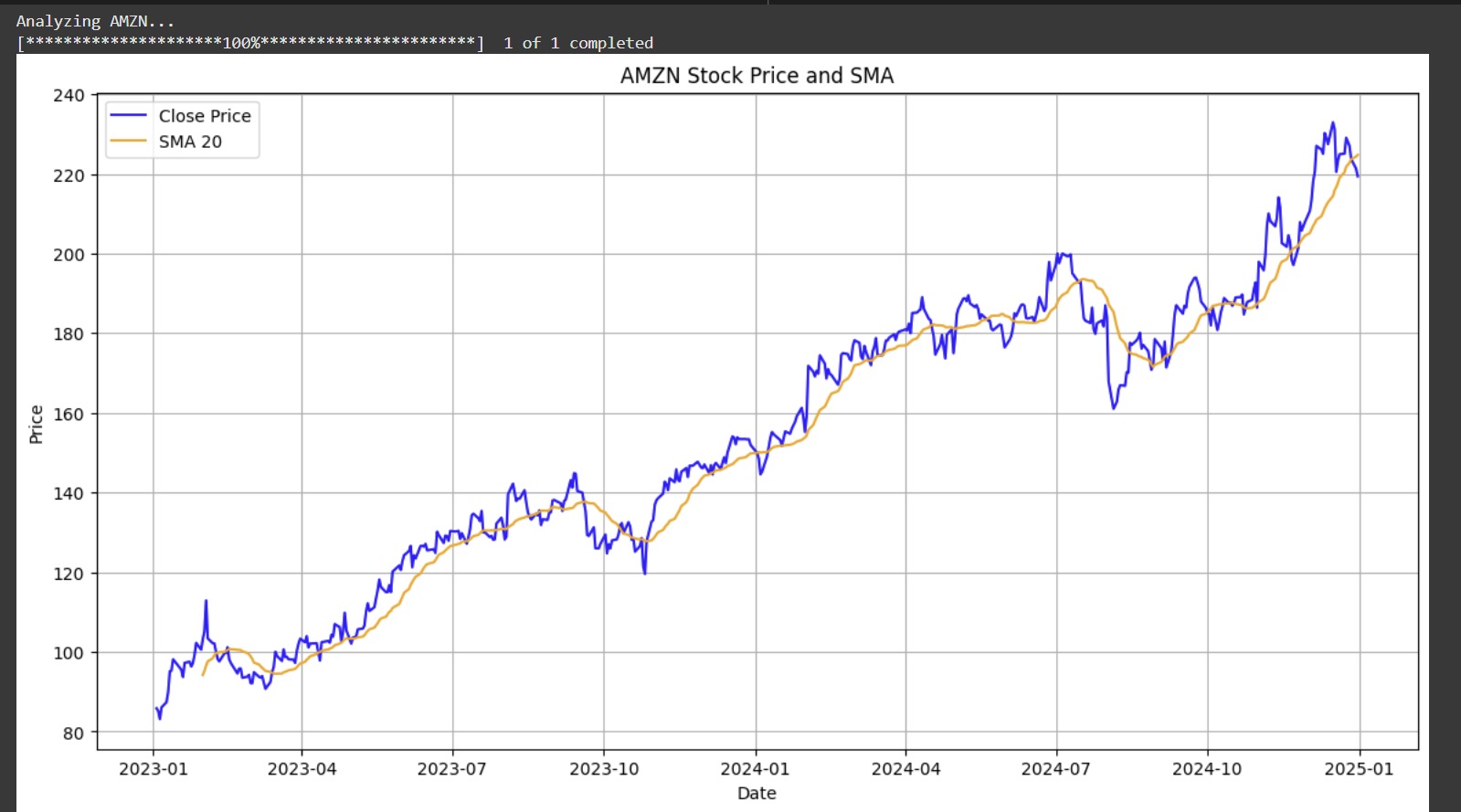
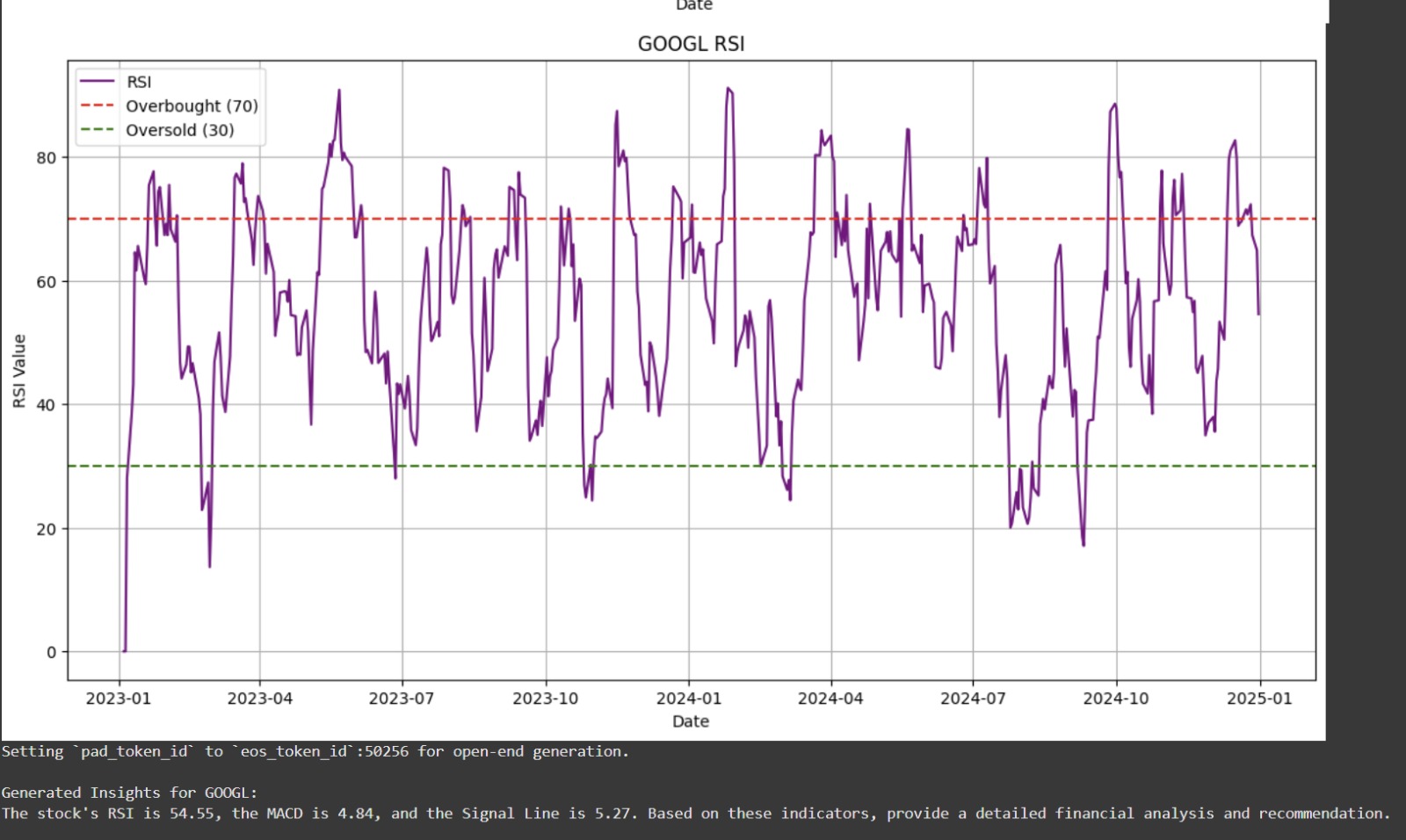
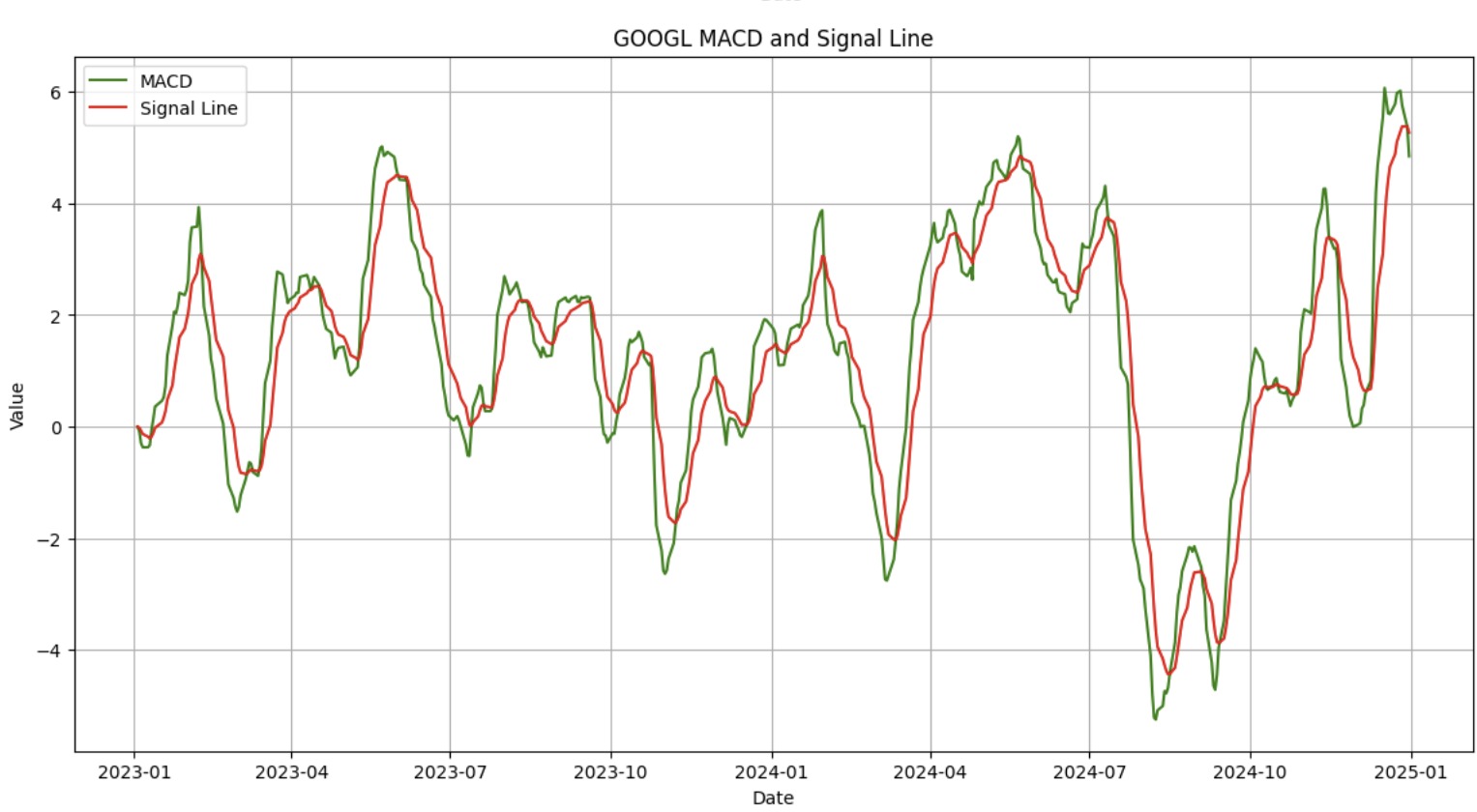
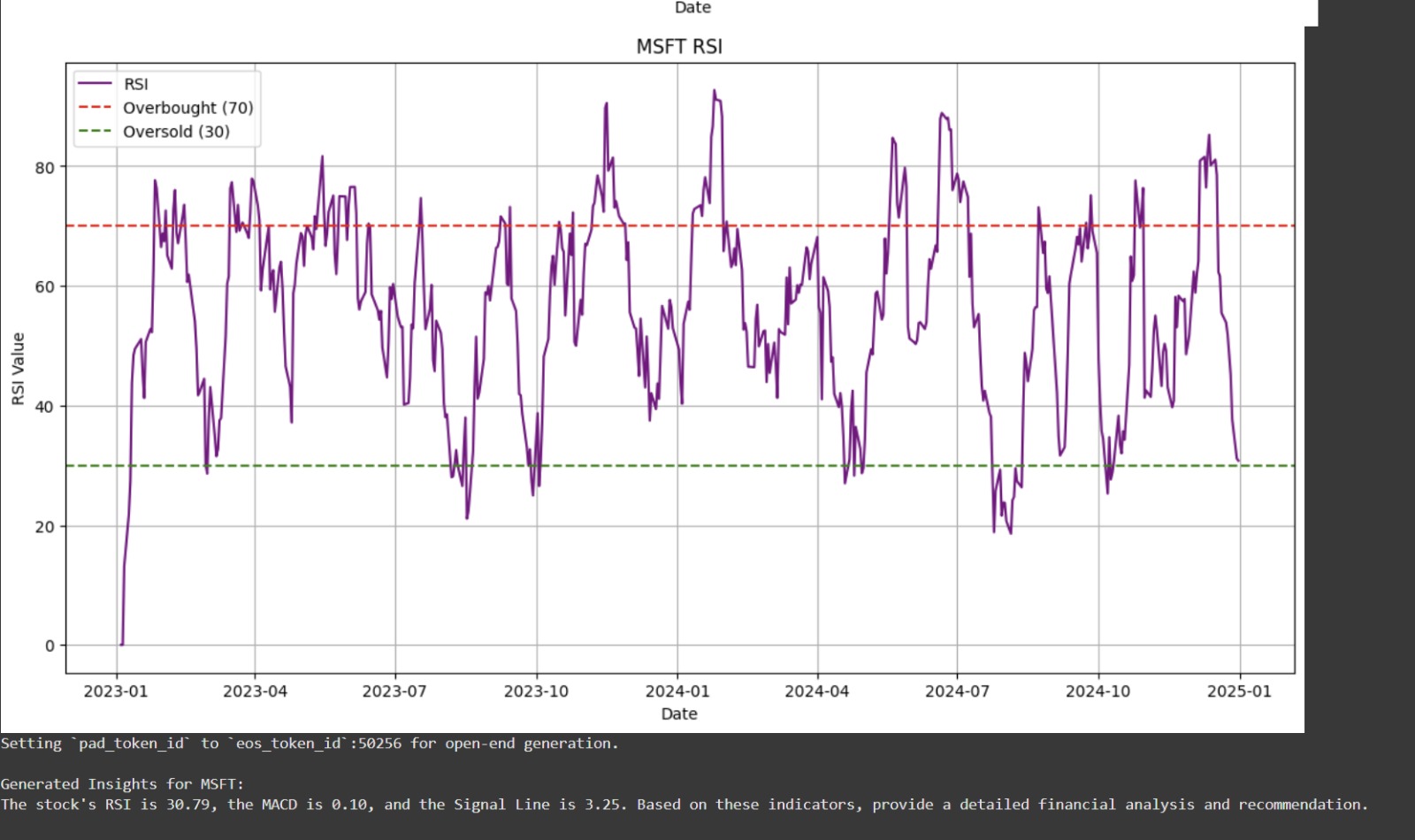
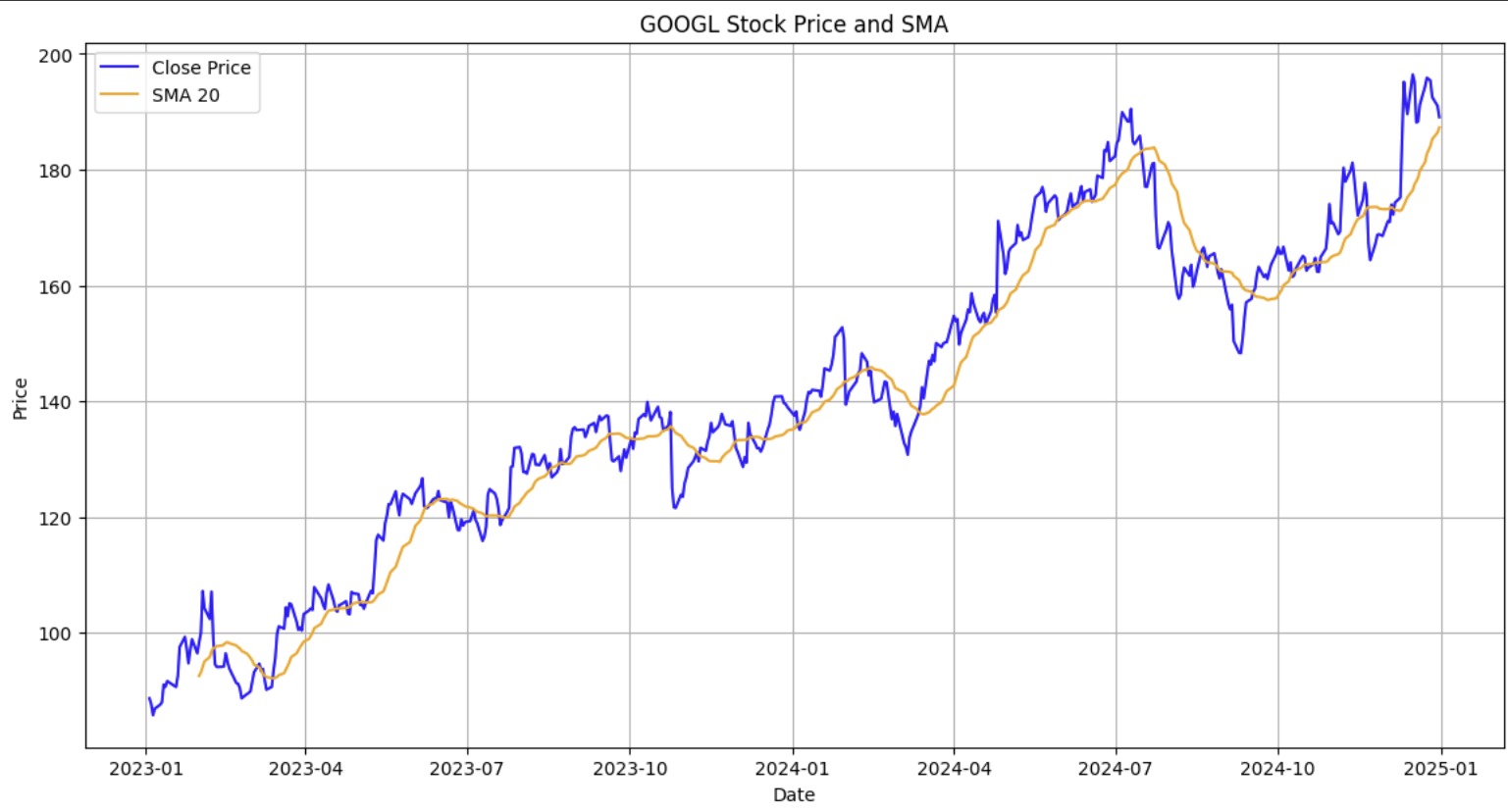
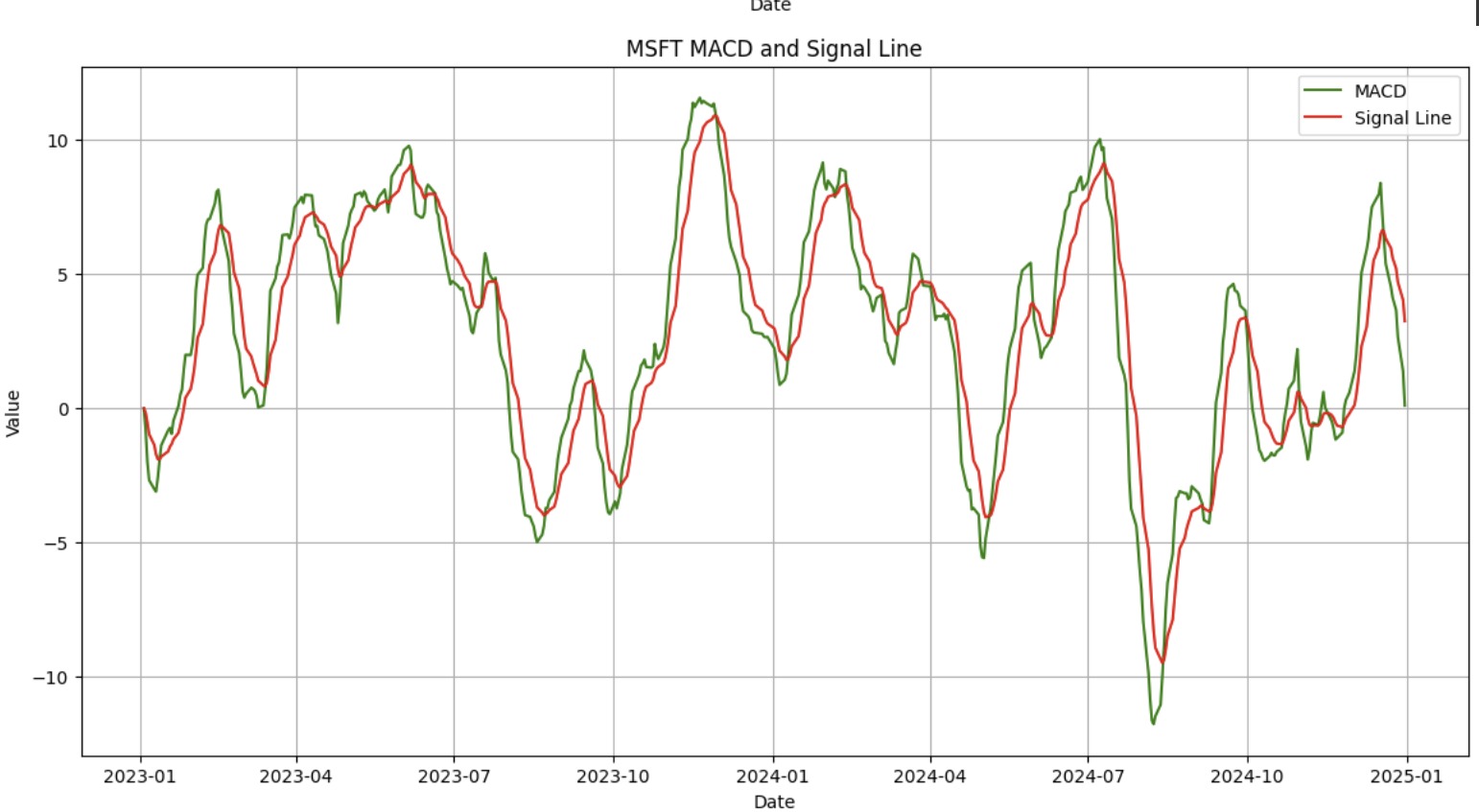
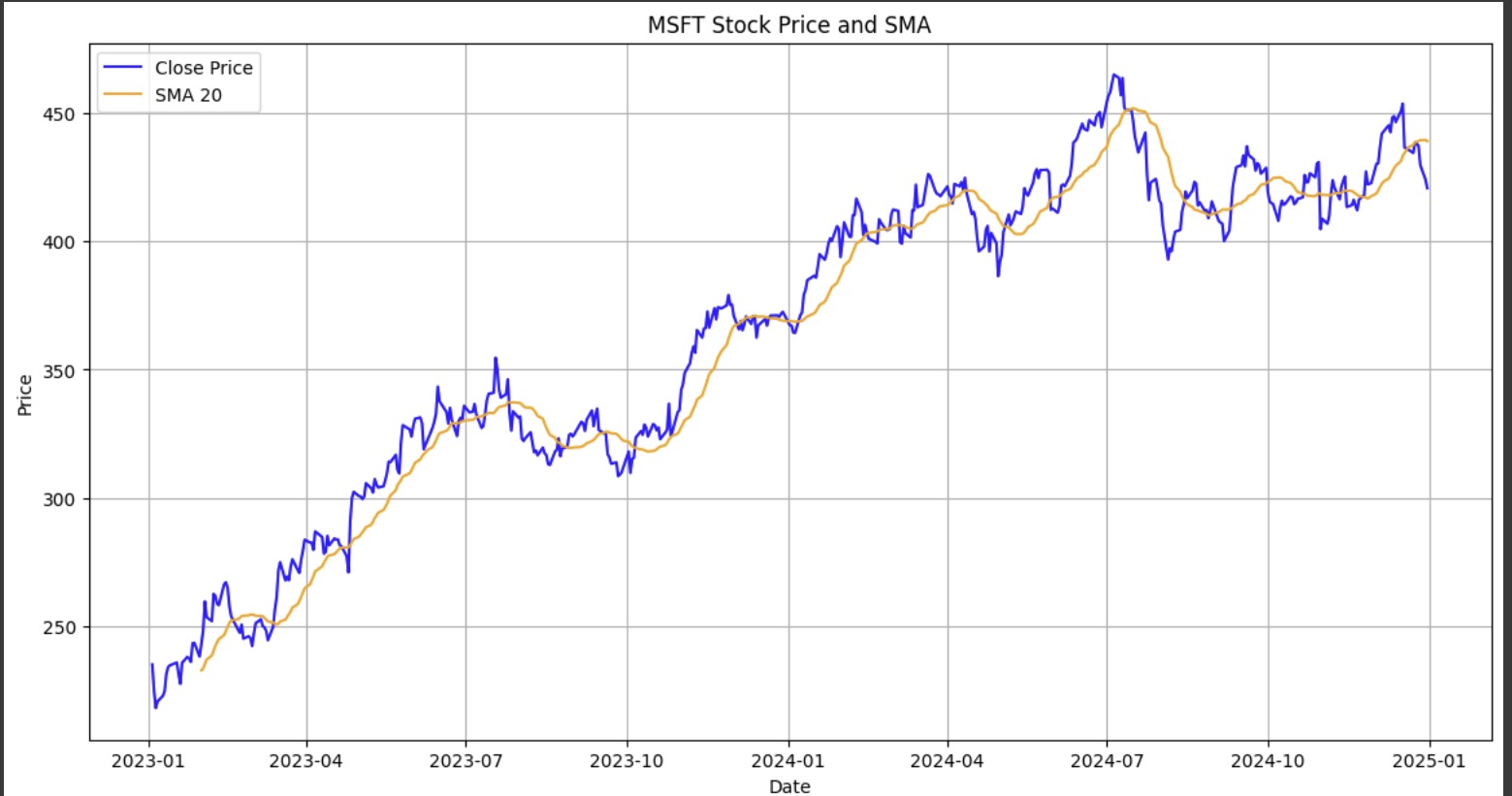
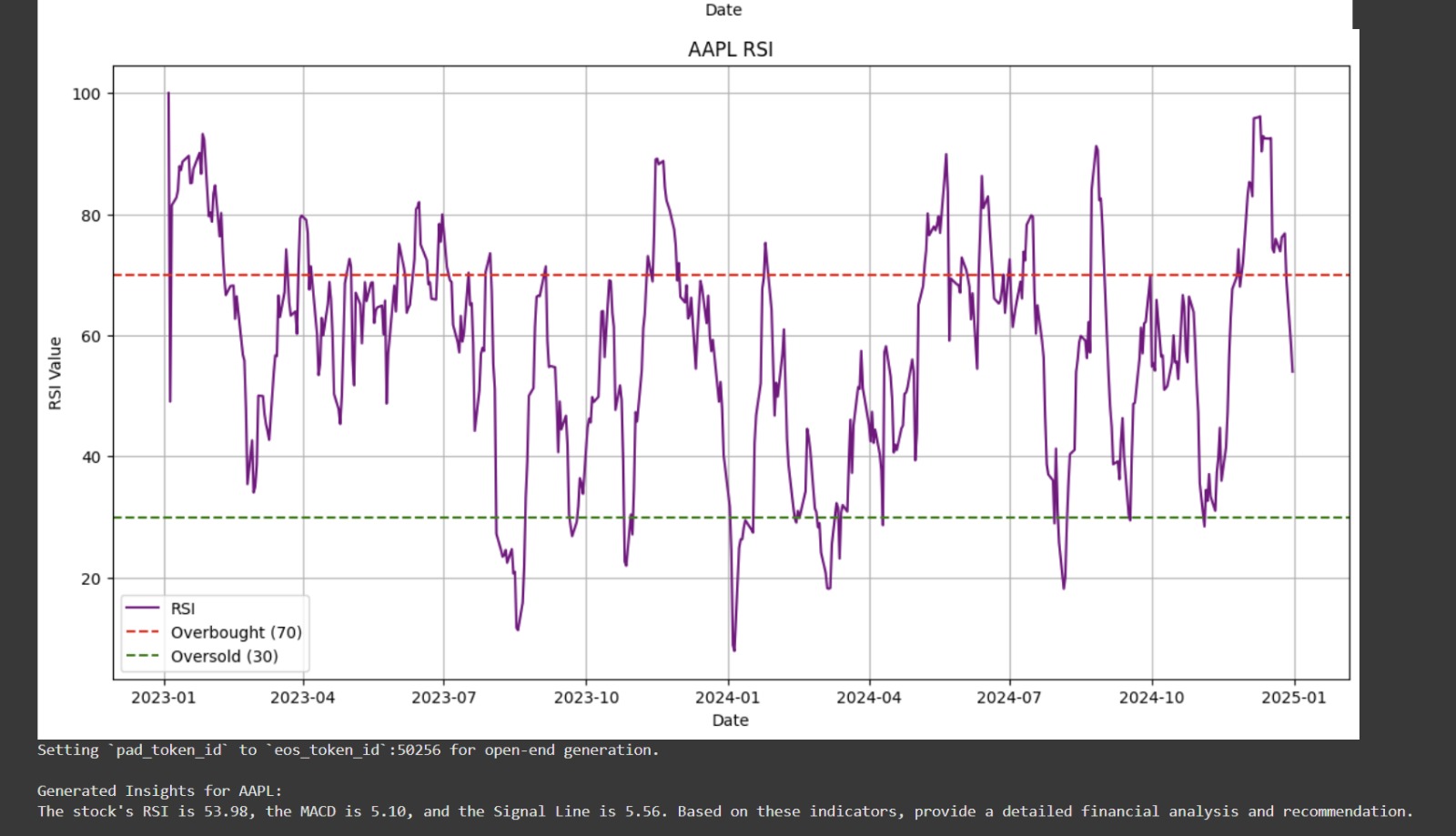
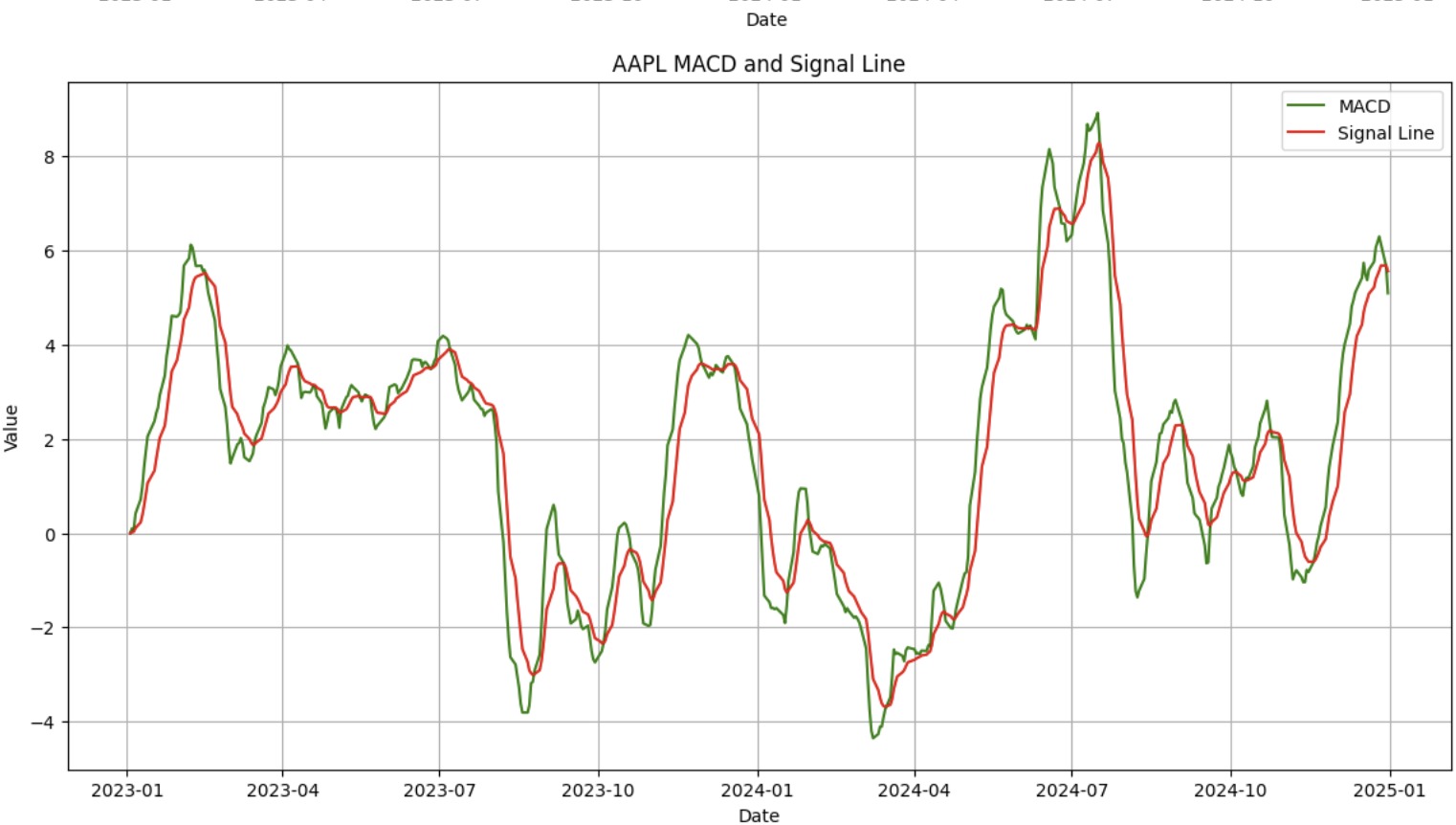
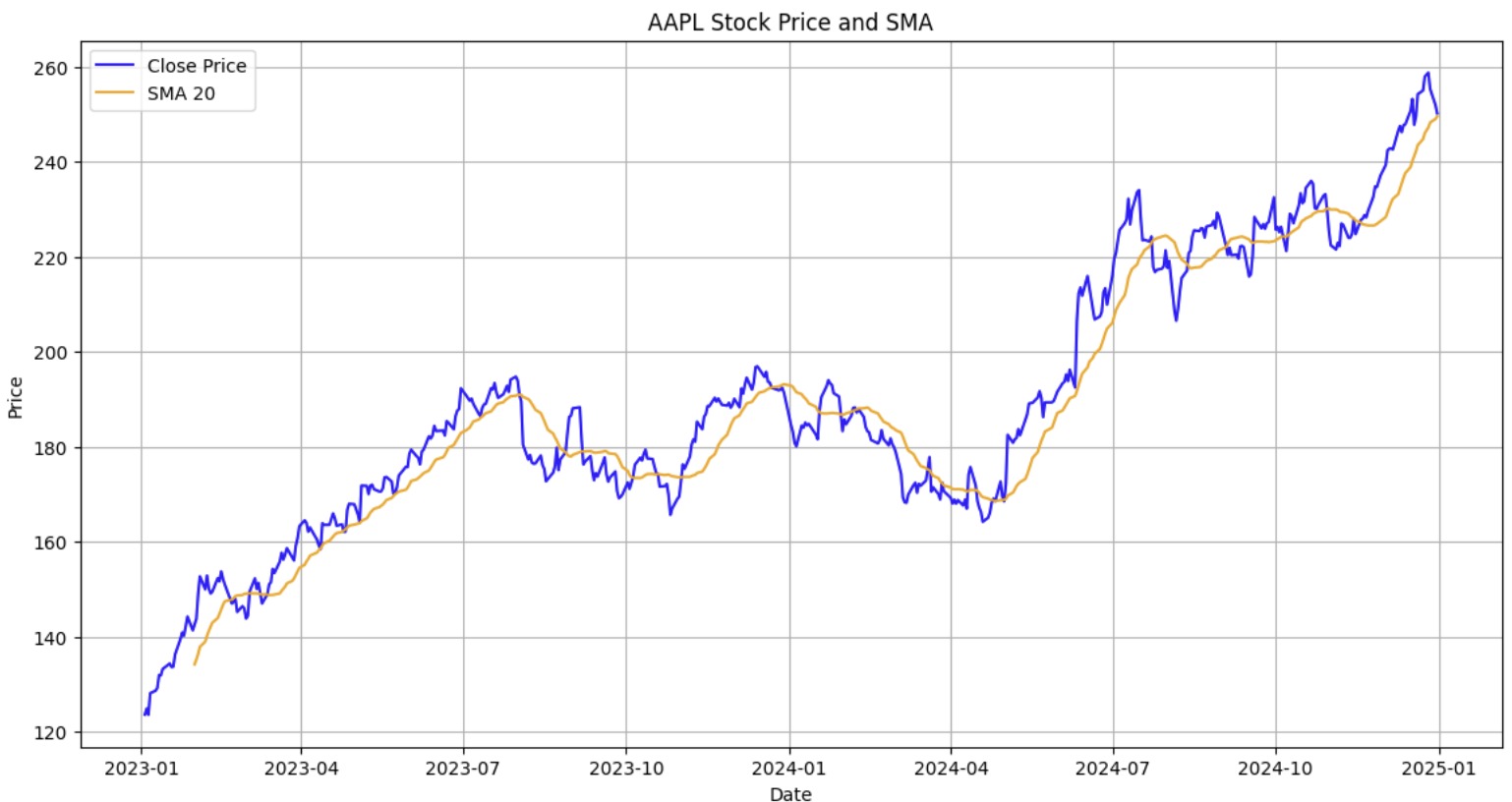
Function Definition

# Function to calculate RSI  
def calculate\_rsi(data, window=14):  
    delta = data.diff(1)  
    gain = delta.where(delta > 0, 0)  
    loss = -delta.where(delta < 0, 0)  
  
    avg\_gain = gain.rolling(window=window, min\_periods=1).mean()  
    avg\_loss = loss.rolling(window=window, min\_periods=1).mean()  
  
    rs = avg\_gain / avg\_loss  
    rsi = 100 - (100 / (1 + rs))  
    return rsi  
  
# Function to calculate MACD  
def calculate\_macd(data, short\_window=12, long\_window=26, signal\_window=9):  
    short\_ema = data.ewm(span=short\_window, adjust=False).mean()  
    long\_ema = data.ewm(span=long\_window, adjust=False).mean()  
    macd = short\_ema - long\_ema  
    signal = macd.ewm(span=signal\_window, adjust=False).mean()  
    return macd, signal  
  
# Fetch stock data  
def fetch\_stock\_data(ticker, start\_date, end\_date):  
    data = yf.download(ticker, start=start\_date, end=end\_date)  
    return data  
  
# Analyze a single stock  
def analyze\_stock(ticker, start\_date, end\_date, llm):  
    data = fetch\_stock\_data(ticker, start\_date, end\_date)  
  
    data['SMA\_20'] = data['Close'].rolling(window=20).mean()  
    data['RSI'] = calculate\_rsi(data['Close'])  
    data['MACD'], data['Signal\_Line'] = calculate\_macd(data['Close'])  
  
    plt.figure(figsize=(14, 7))  
    plt.plot(data['Close'], label='Close Price', color='blue')  
    plt.plot(data['SMA\_20'], label='SMA 20', color='orange')  
    plt.title(f'{ticker} Stock Price and SMA')  
    plt.xlabel('Date')  
    plt.ylabel('Price')  
    plt.legend()  
    plt.grid()  
    plt.show()  
  
    plt.figure(figsize=(14, 7))  
    plt.plot(data['MACD'], label='MACD', color='green')  
    plt.plot(data['Signal\_Line'], label='Signal Line', color='red')  
    plt.title(f'{ticker} MACD and Signal Line')  
    plt.xlabel('Date')  
    plt.ylabel('Value')  
    plt.legend()  
    plt.grid()  
    plt.show()  
  
    plt.figure(figsize=(14, 7))  
    plt.plot(data['RSI'], label='RSI', color='purple')  
    plt.axhline(70, color='red', linestyle='--', label='Overbought (70)')  
    plt.axhline(30, color='green', linestyle='--', label='Oversold (30)')  
    plt.title(f'{ticker} RSI')  
    plt.xlabel('Date')  
    plt.ylabel('RSI Value')  
    plt.legend()  
    plt.grid()  
    plt.show()  
  
    rsi\_value = data['RSI'].iloc[-1]  
    macd\_value = data['MACD'].iloc[-1]  
    signal\_value = data['Signal\_Line'].iloc[-1]  
  
    prompt = (  
        f"The stock's RSI is {rsi\_value:.2f}, the MACD is {macd\_value:.2f}, and the Signal Line is {signal\_value:.2f}. "  
        "Based on these indicators, provide a detailed financial analysis and recommendation for this stock."  
    )  
    insights = llm(prompt, max\_length=100, truncation=True)  
    generated\_text = insights[0]['generated\_text']  
  
    cleaned\_text = generated\_text.split("recommendation")[0] + "recommendation."  
    print(f"\nGenerated Insights for {ticker}:")  
    print(cleaned\_text)

Main Function

def analyze\_top\_nasdaq\_stocks():  
    nasdaq\_stocks = ["AAPL", "MSFT", "GOOGL", "AMZN", "NVDA", "TSLA", "META", "PEP", "AVGO", "COST"]  
    start\_date = "2023-01-01"  
    end\_date = "2025-01-01"  
  
    llm = pipeline("text-generation", model="EleutherAI/gpt-neo-2.7B")  
  
    for ticker in nasdaq\_stocks:  
        print(f"\nAnalyzing {ticker}...")  
        analyze\_stock(ticker, start\_date, end\_date, llm)  
  
analyze\_top\_nasdaq\_stocks()

**Experiment Setup and Result Discussion**



Datasets:

* Stocks: AAPL, MSFT, GOOGL, AMZN, NVDA, TSLA, META, PEP, AVGO, COST
* Source: Yahoo Finance
* Date Range: January 1, 2023 – January 1, 2025

Observations:

* Visualizations of each stock.
* Insights generated through LLM.
* Identified overbought, oversold, and crossover signals.

**Comparison**

Manual vs Automated Analysis:

* Manual analysis is slow and subjective.
* Automated insights are quicker, more consistent.

Comparison with past studies:

* Matches traditional methods in simplicity.
* Offers instant, low-bias interpretations.

**Conclusion**

Combining traditional technical indicators with LLMs creates a powerful, fast, and accessible stock analysis tool.

Future Work:

* Include fundamental data analysis.
* Fine-tune LLMs on financial datasets.
* Expand to commodities and forex markets.

**References**

* Murphy, J. J. (1999). Technical Analysis of the Financial Markets.
* Wilder, J. W. (1978). New Concepts in Technical Trading Systems.
* Patel et al. (2015). Predicting stock and stock price index movement.
* Investopedia on SMA, RSI, MACD.
* HuggingFace Transformers documentation.
* Yahoo Finance via yfinance API.