# Import necessary libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import yfinance as yf

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

from sklearn.metrics import mean\_squared\_error

# Download historical stock price data

ticker\_symbol = "AAPL"  # Example: Apple Inc. stock

start\_date = "2010-01-01"

end\_date = "2021-01-01"

data = yf.download(ticker\_symbol, start=start\_date, end=end\_date)

# Extract the closing price

df = data[['Close']]

# Normalize the data

scaler = MinMaxScaler()

df['Close'] = scaler.fit\_transform(df['Close'].values.reshape(-1, 1))

# Split the data into training and testing sets

train\_size = int(len(df) \* 0.8)

train\_data, test\_data = df[:train\_size], df[train\_size:]

# Create sequences for the LSTM model

def create\_sequences(data, seq\_length):

    sequences = []

    target = []

    for i in range(len(data) - seq\_length):

        seq = data[i:i+seq\_length]

        label = data[i+seq\_length]

        sequences.append(seq)

        target.append(label)

    return np.array(sequences), np.array(target)

seq\_length = 10  # Number of previous days' closing prices to use for prediction

X\_train, y\_train = create\_sequences(train\_data.values, seq\_length)

X\_test, y\_test = create\_sequences(test\_data.values, seq\_length)

# Build the LSTM model

model = Sequential()

model.add(LSTM(50, activation='relu', input\_shape=(seq\_length, 1)))

model.add(Dense(1))

model.compile(optimizer='adam', loss='mean\_squared\_error')

# Train the model

model.fit(X\_train, y\_train, epochs=50, batch\_size=32)

# Make predictions on the test data

predicted\_stock\_prices = model.predict(X\_test)

# Inverse transform the predictions to get actual prices

predicted\_stock\_prices = scaler.inverse\_transform(predicted\_stock\_prices)

# Calculate Mean Squared Error

mse = mean\_squared\_error(test\_data[seq\_length:], predicted\_stock\_prices)

print("Mean Squared Error:", mse)

# Visualize the results

plt.figure(figsize=(12, 6))

plt.plot(test\_data.index[seq\_length:], test\_data[seq\_length:], label='Actual Price')

plt.plot(test\_data.index[seq\_length:], predicted\_stock\_prices, label='Predicted Price')

plt.legend()

plt.title(f'{ticker\_symbol} Stock Price Prediction')

plt.show()