import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import yfinance as yf

from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import mean\_squared\_error

from keras.models import Sequential

from keras.layers import LSTM, Dense

# Download historical stock data from Yahoo Finance

ticker\_symbol = "AAPL"

start\_date = "2010-01-01"

end\_date = "2021-01-01"

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

# Select the 'Close' prices

data = data[['Close']]

# Normalize the data

scaler = MinMaxScaler()

data\_scaled = scaler.fit\_transform(data)

# Split data into training and testing sets

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

train\_data, test\_data = data\_scaled[:train\_size], data\_scaled[train\_size:]

# Create sequences for LSTM training

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

    sequences = []

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

        sequence = data[i:i+sequence\_length]

        target = data[i+sequence\_length]

        sequences.append((sequence, target))

    return sequences

sequence\_length = 10

train\_sequences = create\_sequences(train\_data, sequence\_length)

test\_sequences = create\_sequences(test\_data, sequence\_length)

X\_train = np.array([seq for seq, \_ in train\_sequences])

y\_train = np.array([target for \_, target in train\_sequences])

X\_test = np.array([seq for seq, \_ in test\_sequences])

y\_test = np.array([target for \_, target in test\_sequences])

# Build the LSTM model

model = Sequential()

model.add(LSTM(50, input\_shape=(sequence\_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

train\_predictions = model.predict(X\_train)

test\_predictions = model.predict(X\_test)

# Inverse transform the predictions to get actual stock prices

train\_predictions = scaler.inverse\_transform(train\_predictions)

test\_predictions = scaler.inverse\_transform(test\_predictions)

# Calculate RMSE

train\_rmse = np.sqrt(mean\_squared\_error(data.values[sequence\_length:train\_size], train\_predictions))

test\_rmse = np.sqrt(mean\_squared\_error(data.values[train\_size+sequence\_length:], test\_predictions))

print(f"Train RMSE: {train\_rmse}")

print(f"Test RMSE: {test\_rmse}")

# Plot the predictions

\

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

plt.plot(data.index[sequence\_length:train\_size], data.values[sequence\_length:train\_size], label='Actual Train')

plt.plot(data.index[train\_size+sequence\_length:], data.values[train\_size+sequence\_length:], label='Actual Test')

plt.plot(data.index[sequence\_length:train\_size], train\_predictions, label='Predicted Train')

plt.plot(data.index[train\_size+sequence\_length:], test\_predictions, label='Predicted Test')

plt.legend()

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

plt.xlabel("Date")

plt.ylabel("Stock Price")

plt.show()