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

# Step 1: Load stock data

ticker = 'AAPL' # Example: Apple Inc.

df = yf.download(ticker, start='2015-01-01', end='2024-01-01')

data = df['Close'].values.reshape(-1, 1)

# Step 2: Normalize data

scaler = MinMaxScaler(feature\_range=(0, 1))

scaled\_data = scaler.fit\_transform(data)

# Step 3: Create time series dataset

def create\_dataset(data, time\_step=60):

X, y = [], []

for i in range(time\_step, len(data)):

X.append(data[i-time\_step:i])

y.append(data[i])

return np.array(X), np.array(y)

time\_step = 60

X, y = create\_dataset(scaled\_data, time\_step)

X = X.reshape(X.shape[0], X.shape[1], 1) # [samples, time\_steps, features]

# Step 4: Split into training and test sets

split = int(0.8 \* len(X))

X\_train, y\_train = X[:split], y[:split]

X\_test, y\_test = X[split:], y[split:]

# Step 5: Build LSTM model

model = Sequential([

LSTM(50, return\_sequences=True, input\_shape=(time\_step, 1)),

LSTM(50),

Dense(1)

])

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

model.fit(X\_train, y\_train, epochs=10, batch\_size=32)

# Step 6: Predict

predictions = model.predict(X\_test)

predictions = scaler.inverse\_transform(predictions)

y\_test\_actual = scaler.inverse\_transform(y\_test)

# Step 7: Plot results

plt.figure(figsize=(10,6))

plt.plot(y\_test\_actual, label='Actual Price')

plt.plot(predictions, label='Predicted Price')

plt.title('Stock Price Prediction')

plt.xlabel('Time')

plt.ylabel('Stock Price')

plt.legend()

plt.show()