import numpy as np

import pandas as pd

import yfinance as yf

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential

from keras.layers import Dense, LSTM

from datetime import datetime, timedelta

from forex\_python.converter import CurrencyRates

# Step 1: Download stock data from Yahoo Finance

def download\_stock\_data(ticker, start\_date, end\_date):

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

return data

# Step 2: Preprocess the data (Scaling and reshaping)

def preprocess\_data(data):

close\_prices = data['Close'].values.reshape(-1, 1)

scaler = MinMaxScaler(feature\_range=(0, 1)) # Scale between 0 and 1

scaled\_data = scaler.fit\_transform(close\_prices)

return scaled\_data, scaler

# Step 3: Create the dataset for training the LSTM model

def create\_datasets(scaled\_data, time\_step=60):

x\_train, y\_train = [], []

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

x\_train.append(scaled\_data[i-time\_step:i, 0])

y\_train.append(scaled\_data[i, 0])

return np.array(x\_train), np.array(y\_train)

# Step 4: Build the LSTM model

def build\_lstm\_model(input\_shape):

model = Sequential()

model.add(LSTM(units=50, return\_sequences=True, input\_shape=input\_shape))

model.add(LSTM(units=50, return\_sequences=False))

model.add(Dense(units=25))

model.add(Dense(units=1))

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

return model

# Step 5: Predict the stock price for tomorrow and day after tomorrow

def predict\_prices(model, scaler, last\_60\_days):

last\_60\_days\_scaled = scaler.transform(last\_60\_days.values.reshape(-1, 1))

# Predict tomorrow's price

X\_test = np.array([last\_60\_days\_scaled])

X\_test = np.reshape(X\_test, (X\_test.shape[0], X\_test.shape[1], 1))

predicted\_tomorrow = model.predict(X\_test)

predicted\_tomorrow\_price = scaler.inverse\_transform(predicted\_tomorrow)[0, 0]

# Append tomorrow's prediction to last 59 days to predict day after tomorrow

new\_input = np.append(last\_60\_days\_scaled[1:], predicted\_tomorrow)

new\_input = np.reshape(new\_input, (1, new\_input.shape[0], 1))

predicted\_day\_after\_tomorrow = model.predict(new\_input)

predicted\_day\_after\_tomorrow\_price = scaler.inverse\_transform(predicted\_day\_after\_tomorrow)[0, 0]

return predicted\_tomorrow\_price, predicted\_day\_after\_tomorrow\_price

# Step 6: Fetch USD to INR conversion rate

def fetch\_usd\_to\_inr():

currency\_converter = CurrencyRates()

try:

return currency\_converter.get\_rate('USD', 'INR') # Removed timeout

except Exception as e:

print(f"Error fetching exchange rate: {e}")

return 82.0 # Example default value in case of error

# Step 7: Main function

def main():

ticker = "GOOG" # Example: Google Inc. (You can change the stock symbol)

start\_date = "2015-01-01"

end\_date = datetime.today().strftime('%Y-%m-%d')

# Download and preprocess stock data

data = download\_stock\_data(ticker, start\_date, end\_date)

scaled\_data, scaler = preprocess\_data(data)

# Create training datasets

time\_step = 60 # Use last 60 days to predict the next day

x\_train, y\_train = create\_datasets(scaled\_data, time\_step)

# Reshape the data for LSTM

x\_train = np.reshape(x\_train, (x\_train.shape[0], x\_train.shape[1], 1))

# Build and train the LSTM model

model = build\_lstm\_model((x\_train.shape[1], 1))

model.fit(x\_train, y\_train, batch\_size=1, epochs=1) # Train with 1 epoch for demonstration

# Predict tomorrow and day after tomorrow's stock price

last\_60\_days = data['Close'][-60:] # Get the last 60 days of closing prices

predicted\_tomorrow, predicted\_day\_after\_tomorrow = predict\_prices(model, scaler, last\_60\_days)

# Fetch the USD to INR conversion rate

usd\_to\_inr\_rate = fetch\_usd\_to\_inr()

# Convert predictions to INR

predicted\_tomorrow\_inr = predicted\_tomorrow \* usd\_to\_inr\_rate

predicted\_day\_after\_tomorrow\_inr = predicted\_day\_after\_tomorrow \* usd\_to\_inr\_rate

# Print the predictions

current\_price = data['Close'].iloc[-1] # Get the last closing price (scalar)

tomorrow = datetime.today() + timedelta(days=1)

day\_after\_tomorrow = datetime.today() + timedelta(days=2)

print(f"Predicted price for {ticker} on {tomorrow.strftime('%Y-%m-%d')} (INR): ₹{predicted\_tomorrow\_inr:.2f}")

print(f"Predicted price for {ticker} on {day\_after\_tomorrow.strftime('%Y-%m-%d')} (INR): ₹{predicted\_day\_after\_tomorrow\_inr:.2f}")

print(f"Today's actual closing price (USD): ${current\_price.iloc[-1]:.2f}")

print(f"USD to INR Conversion Rate: {usd\_to\_inr\_rate:.2f}")

if predicted\_day\_after\_tomorrow > predicted\_tomorrow:

print("The stock price is predicted to go up the day after tomorrow.")

else:

print("The stock price is predicted to go down the day after tomorrow.")

if \_name\_ == "\_main\_":

main()import numpy as np

import pandas as pd

import yfinance as yf

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential

from keras.layers import Dense, LSTM

from datetime import datetime, timedelta

from forex\_python.converter import CurrencyRates

# Step 1: Download stock data from Yahoo Finance

def download\_stock\_data(ticker, start\_date, end\_date):

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

return data

# Step 2: Preprocess the data (Scaling and reshaping)

def preprocess\_data(data):

close\_prices = data['Close'].values.reshape(-1, 1)

scaler = MinMaxScaler(feature\_range=(0, 1)) # Scale between 0 and 1

scaled\_data = scaler.fit\_transform(close\_prices)

return scaled\_data, scaler

# Step 3: Create the dataset for training the LSTM model

def create\_datasets(scaled\_data, time\_step=60):

x\_train, y\_train = [], []

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

x\_train.append(scaled\_data[i-time\_step:i, 0])

y\_train.append(scaled\_data[i, 0])

return np.array(x\_train), np.array(y\_train)

# Step 4: Build the LSTM model

def build\_lstm\_model(input\_shape):

model = Sequential()

model.add(LSTM(units=50, return\_sequences=True, input\_shape=input\_shape))

model.add(LSTM(units=50, return\_sequences=False))

model.add(Dense(units=25))

model.add(Dense(units=1))

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

return model

# Step 5: Predict the stock price for tomorrow and day after tomorrow

def predict\_prices(model, scaler, last\_60\_days):

last\_60\_days\_scaled = scaler.transform(last\_60\_days.values.reshape(-1, 1))

# Predict tomorrow's price

X\_test = np.array([last\_60\_days\_scaled])

X\_test = np.reshape(X\_test, (X\_test.shape[0], X\_test.shape[1], 1))

predicted\_tomorrow = model.predict(X\_test)

predicted\_tomorrow\_price = scaler.inverse\_transform(predicted\_tomorrow)[0, 0]

# Append tomorrow's prediction to last 59 days to predict day after tomorrow

new\_input = np.append(last\_60\_days\_scaled[1:], predicted\_tomorrow)

new\_input = np.reshape(new\_input, (1, new\_input.shape[0], 1))

predicted\_day\_after\_tomorrow = model.predict(new\_input)

predicted\_day\_after\_tomorrow\_price = scaler.inverse\_transform(predicted\_day\_after\_tomorrow)[0, 0]

return predicted\_tomorrow\_price, predicted\_day\_after\_tomorrow\_price

# Step 6: Fetch USD to INR conversion rate

def fetch\_usd\_to\_inr():

currency\_converter = CurrencyRates()

try:

return currency\_converter.get\_rate('USD', 'INR') # Removed timeout

except Exception as e:

print(f"Error fetching exchange rate: {e}")

return 82.0 # Example default value in case of error

# Step 7: Main function

def main():

ticker = "GOOG" # Example: Google Inc. (You can change the stock symbol)

start\_date = "2015-01-01"

end\_date = datetime.today().strftime('%Y-%m-%d')

# Download and preprocess stock data

data = download\_stock\_data(ticker, start\_date, end\_date)

scaled\_data, scaler = preprocess\_data(data)

# Create training datasets

time\_step = 60 # Use last 60 days to predict the next day

x\_train, y\_train = create\_datasets(scaled\_data, time\_step)

# Reshape the data for LSTM

x\_train = np.reshape(x\_train, (x\_train.shape[0], x\_train.shape[1], 1))

# Build and train the LSTM model

model = build\_lstm\_model((x\_train.shape[1], 1))

model.fit(x\_train, y\_train, batch\_size=1, epochs=1) # Train with 1 epoch for demonstration

# Predict tomorrow and day after tomorrow's stock price

last\_60\_days = data['Close'][-60:] # Get the last 60 days of closing prices

predicted\_tomorrow, predicted\_day\_after\_tomorrow = predict\_prices(model, scaler, last\_60\_days)

# Fetch the USD to INR conversion rate

usd\_to\_inr\_rate = fetch\_usd\_to\_inr()

# Convert predictions to INR

predicted\_tomorrow\_inr = predicted\_tomorrow \* usd\_to\_inr\_rate

predicted\_day\_after\_tomorrow\_inr = predicted\_day\_after\_tomorrow \* usd\_to\_inr\_rate

# Print the predictions

current\_price = data['Close'].iloc[-1] # Get the last closing price (scalar)

tomorrow = datetime.today() + timedelta(days=1)

day\_after\_tomorrow = datetime.today() + timedelta(days=2)

print(f"Predicted price for {ticker} on {tomorrow.strftime('%Y-%m-%d')} (INR): ₹{predicted\_tomorrow\_inr:.2f}")

print(f"Predicted price for {ticker} on {day\_after\_tomorrow.strftime('%Y-%m-%d')} (INR): ₹{predicted\_day\_after\_tomorrow\_inr:.2f}")

print(f"Today's actual closing price (USD): ${current\_price.iloc[-1]:.2f}")

print(f"USD to INR Conversion Rate: {usd\_to\_inr\_rate:.2f}")

if predicted\_day\_after\_tomorrow > predicted\_tomorrow:

print("The stock price is predicted to go up the day after tomorrow.")

else:

print("The stock price is predicted to go down the day after tomorrow.")

if \_name\_ == "\_main\_":

main()