import numpy  
import matplotlib.pyplot as plt  
from pandas import read\_excel  
import math  
from keras.models import Sequential  
from keras.layers import Dense  
from keras.layers import LSTM  
from sklearn.preprocessing import MinMaxScaler  
from sklearn.metrics import mean\_squared\_error  
from keras.utils.vis\_utils import plot\_model  
  
  
# 创建数据集  
def create\_dataset(dataset, look\_back=1):  
 dataX, dataY = [], []  
 for i in range(len(dataset) - look\_back - 1):  
 a = dataset[i:(i + look\_back), 0]  
 dataX.append(a)  
 dataY.append(dataset[i + look\_back, 0])  
 return numpy.array(dataX), numpy.array(dataY)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # 加载数据  
 dataframe = read\_excel(r"C:\Users\阿韩想养二哈\Desktop\a.xlsx", skipfooter=3)  
 dataset = dataframe.values  
 # 将整型变为float  
 dataset = dataset.astype('float32')  
  
 # 数据处理，归一化至0~1之间  
 scaler = MinMaxScaler(feature\_range=(0, 1))  
 dataset = scaler.fit\_transform(dataset)  
  
 # 划分训练集和测试集  
 train\_size = int(len(dataset) \* 0.67)  
 test\_size = len(dataset) - train\_size  
 train, test = dataset[0:train\_size, :], dataset[train\_size:len(dataset), :]  
  
 # 创建测试集和训练集  
 look\_back = 1  
 trainX, trainY = create\_dataset(train, look\_back) # 单步预测  
 testX, testY = create\_dataset(test, look\_back)  
  
 # 调整输入数据的格式  
 trainX = numpy.reshape(trainX, (trainX.shape[0], look\_back, trainX.shape[1])) # （样本个数，1，输入的维度）  
 testX = numpy.reshape(testX, (testX.shape[0], look\_back, testX.shape[1]))  
  
 # 创建LSTM神经网络模型  
 model = Sequential()  
 model.add(LSTM(120, input\_shape=(trainX.shape[1], trainX.shape[2]))) # 输入维度为1，时间窗的长度为1，隐含层神经元节点个数为120  
 model.add(Dense(1))  
 model.compile(loss='mean\_squared\_error', optimizer='adam')  
 model.fit(trainX, trainY, epochs=100, batch\_size=1, verbose=2)  
  
 # 绘制网络结构  
 plot\_model(model, to\_file='E:/model.png', show\_shapes=True);  
  
 # 预测  
 trainPredict = model.predict(trainX)  
 testPredict = model.predict(testX)  
  
 # 反归一化  
 trainPredict = scaler.inverse\_transform(trainPredict)  
 trainY = scaler.inverse\_transform([trainY])  
 testPredict = scaler.inverse\_transform(testPredict)  
 testY = scaler.inverse\_transform([testY])  
  
 # 计算得分  
 trainScore = math.sqrt(mean\_squared\_error(trainY[0], trainPredict[:, 0]))  
 print('Train Score: %.2f RMSE' % (trainScore))  
 testScore = math.sqrt(mean\_squared\_error(testY[0], testPredict[:, 0]))  
 print('Test Score: %.2f RMSE' % (testScore))  
  
 # 绘图  
 trainPredictPlot = numpy.empty\_like(dataset)  
 trainPredictPlot[:, :] = numpy.nan  
 trainPredictPlot[look\_back:len(trainPredict) + look\_back, :] = trainPredict  
 testPredictPlot = numpy.empty\_like(dataset)  
 testPredictPlot[:, :] = numpy.nan  
 testPredictPlot[len(trainPredict) + (look\_back \* 2) + 1:len(dataset) - 1, :] = testPredict  
 plt.plot(scaler.inverse\_transform(dataset))  
 plt.plot(trainPredictPlot)  
 plt.plot(testPredictPlot)  
 plt.show()