import numpy as np  
import pandas as pd  
import math  
import matplotlib.pyplot as plt  
from sklearn.preprocessing import MinMaxScaler from sklearn.metrics import mean\_squared\_error from keras.models import Sequential

from keras.layers import Dense, Activation from keras.layers import LSTM

np.random.seed(7)

dataset = pd.read\_csv('F:\\SEM8\\Security\\ISS dataset-feature extraction.csv', usecols=[1,2,3,4,5,6,7,8,9,10])

dataset = dataset.reindex(index = dataset.index[::-1]) obs = np.arange(1, len(dataset) + 1, 1)  
OHLC\_avg = dataset.mean(axis = 1)

HLC\_avg = dataset[['Frame\_size','Epoch\_time','Time\_delta\_from\_previous\_displayed\_pkt','Coloring\_ru le\_name','len','File\_data','Total\_length','Time\_since\_previous\_frame\_in\_this\_tcp\_stream','Tc p\_payload','Label']].mean(axis = 1)

close\_val = dataset[['Label']]  
plt.plot(obs, OHLC\_avg, 'r', label = 'OHLC avg')  
plt.plot(obs, HLC\_avg, 'b', label = 'HLC avg')  
plt.plot(obs, close\_val, 'g', label = 'Closing price')  
plt.legend(loc = 'upper right')  
plt.show()  
OHLC\_avg = np.reshape(OHLC\_avg.values, (len(OHLC\_avg),1)) # 1664 scaler = MinMaxScaler(feature\_range=(0, 1))  
OHLC\_avg = scaler.fit\_transform(OHLC\_avg)

train\_OHLC = int(len(OHLC\_avg) \* 0.75) test\_OHLC = len(OHLC\_avg) - train\_OHLC

train\_OHLC, test\_OHLC = OHLC\_avg[0:train\_OHLC,:], OHLC\_avg[train\_OHLC:len(OHLC\_avg),:]

def new\_dataset(dataset, step\_size): data\_X, data\_Y = [], []

for i in range(len(dataset)-step\_size-1):  
a = dataset[i:(i+step\_size), 0] data\_X.append(a) data\_Y.append(dataset[i + step\_size, 0])

return np.array(data\_X), np.array(data\_Y)

trainX, trainY = new\_dataset(train\_OHLC, 1) testX, testY = new\_dataset(test\_OHLC, 1)

trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1])) testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1])) step\_size = 1

model = Sequential()

model.add(LSTM(32, input\_shape=(1, step\_size), return\_sequences = True))#hidden nodes=32

model.add(LSTM(16)) model.add(Dense(1)) model.add(Activation('relu')) print(model.summary())

model.compile(loss='mean\_squared\_error', optimizer='adagrad',metrics=['mse','accuracy']) # Try SGD, adam, adagrad and compare!!!

model.fit(trainX, trainY, epochs=50, batch\_size=16, verbose=2)

trainPredict = model.predict(trainX) testPredict = model.predict(testX)

trainPredict = scaler.inverse\_transform(trainPredict) trainY = scaler.inverse\_transform([trainY]) testPredict = scaler.inverse\_transform(testPredict) testY = scaler.inverse\_transform([testY])

trainScore = math.sqrt(mean\_squared\_error(trainY[0], trainPredict[:,0])) print('Train RMSE: %.2f' % (trainScore))

testScore = math.sqrt(mean\_squared\_error(testY[0], testPredict[:,0])) print('Test RMSE: %.2f' % (testScore))