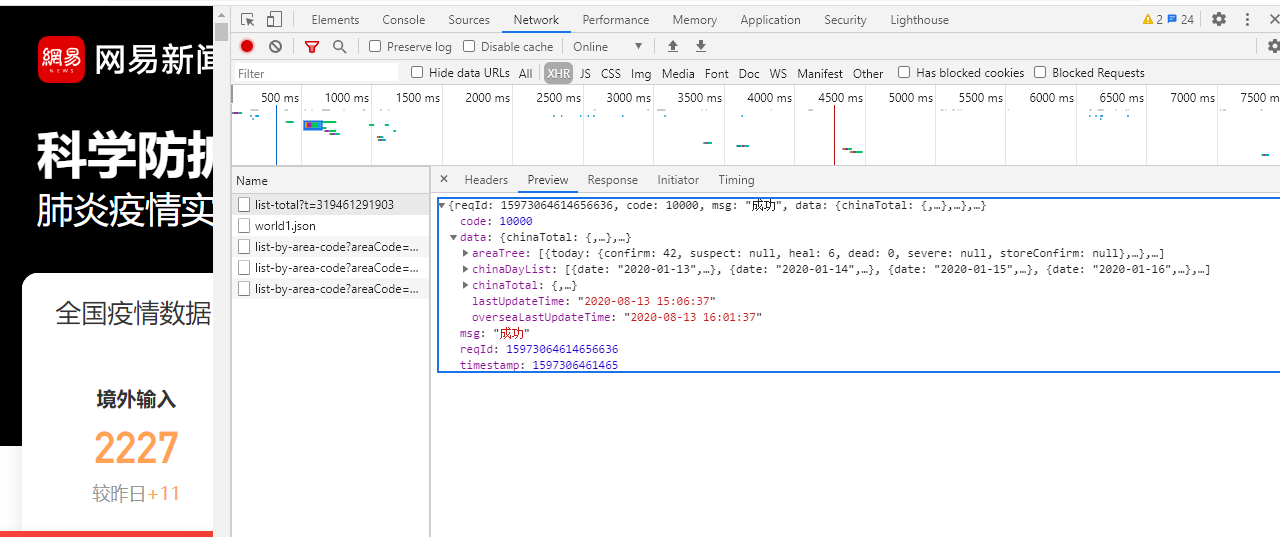
### 新冠疫情未来预测Python

对于新冠未来预测的代码实现，主要分为三个部分：国内疫情实时动态的爬取、累积新冠患者的Logistic模型预测和每日确诊患者的LSTM模型预测。

首先对于国内历史疫情数据源的获取，小组考虑将新闻媒体的播报平台作为数据源，选取网易的疫情播报平台，数据内容非常丰富，且公信度也比较高。

 找到存放着关于疫情的数据，先对这个地址进行爬虫。

import requests  
import pandas as pd  
import time  
pd.set\_option('max\_rows',500)  
headers = { 'user-agent': 'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.149 Safari/537.36' }# 设置请求头，伪装为浏览器  
url = 'https://c.m.163.com/ug/api/wuhan/app/data/list-total'  
r = requests.get(url, headers=headers) # 使用requests发起请求  
print(r.status\_code)

返回后的内容是一个几十万长度的字符串，由于字符串格式不方便进行分析，并且在网页预览中发现数据为类似字典的json格式，所以我们将其转为json格式。

import json  
data\_json = json.loads(r.text)  
data\_json.keys()

在data中存放着我们需要的数据，因此我们取出数据。

data = data\_json['data'] # 取出json中的数据  
print(data.keys())

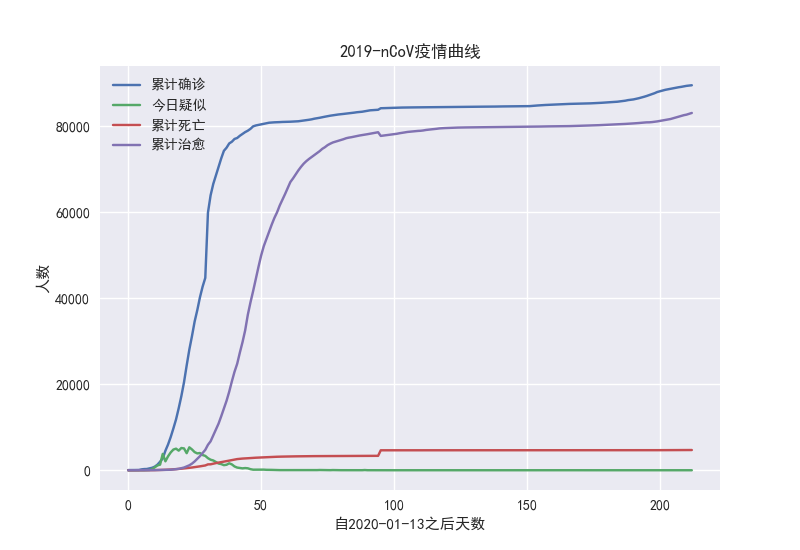
在数据data中，总共有四个键，'chinaTotal', 'chinaDayList', 'lastUpdateTime', 'areaTree'，其中chinaDayList存放着中国的历史数据，提取数据，保存数据。

chinaDayList = data['chinaDayList']  
print(chinaDayList[0])  
alltime\_China = get\_data(chinaDayList,['date','lastUpdateTime'])  
alltime\_China.head()

save\_data(alltime\_China,'alltime\_China')

第二部分是新冠患者的Logistic模型预测。导入爬虫下载的csv文件（每日运行爬虫生成最新文件，在代码中需手动修改一些数据，例如数据文件的导入和最小二乘法里的数据对应）。

China\_data = pd.read\_csv('alltime\_China\_2020\_08\_13.csv', encoding = 'gbk')# 读取文件

 观察部分数据在2019-nCoV疫情曲线的变化（时间轴较长，不显示日期作为横坐标），结合在疫情传播中，由于刚开始社会各界并不重视，疫情传播的环境阻力较小，因此增长率较大；当疫情进行到一定阶段，随着如戴口罩、感染源隔离、倡导减少活动等各项措施的实施，疫情传播的环境阻力增大，增长率减低，则传播曲线呈现出S形。

所以小组利用Logistic曲线的特性，通过调参模拟疫情的历史变化趋势，并预测未来的累积新冠患者疫情数据。

构建Logistic模型，定义预测误差函数。

def logistic\_increase\_function(p, t):# 构建Logistic模型  
 K, a, b = p  
 exp\_value = np.exp(-a \* (t - b))  
 return K / (1 + exp\_value)  
def err\_f(p, t, y):# 定义预测误差函数  
 return logistic\_increase\_function(p, t) - y

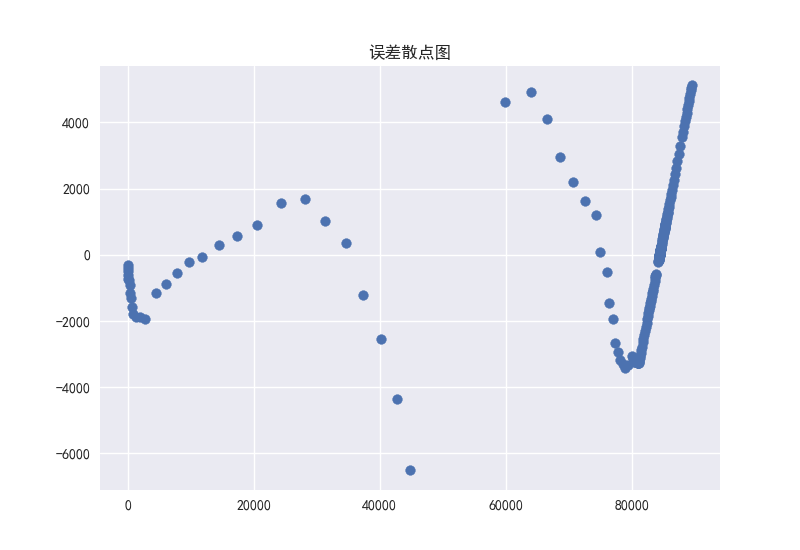
设置参数初始值，初始值只要不是太离谱，最终都会收敛，再利用最小二乘法求解参数。

logistic\_p0 = [90000, 0.8, 20] # 设置参数初始值  
t = np.array([i + 1 for i in range(213)])  
China\_y = China\_data['total\_confirm'].values  
logistic\_params = leastsq(err\_f, logistic\_p0, args=(t, China\_y))# 利用最小二乘法求解参数  
China\_p = logistic\_params[0]

绘制误差散点图。

China\_predict\_data = logistic\_increase\_function(China\_p, t)# 利用定义的Logistic预测  
China\_e = China\_y - China\_predict\_data# 预测的误差  
plt.scatter(China\_y, China\_e)# 绘制误差的散点图

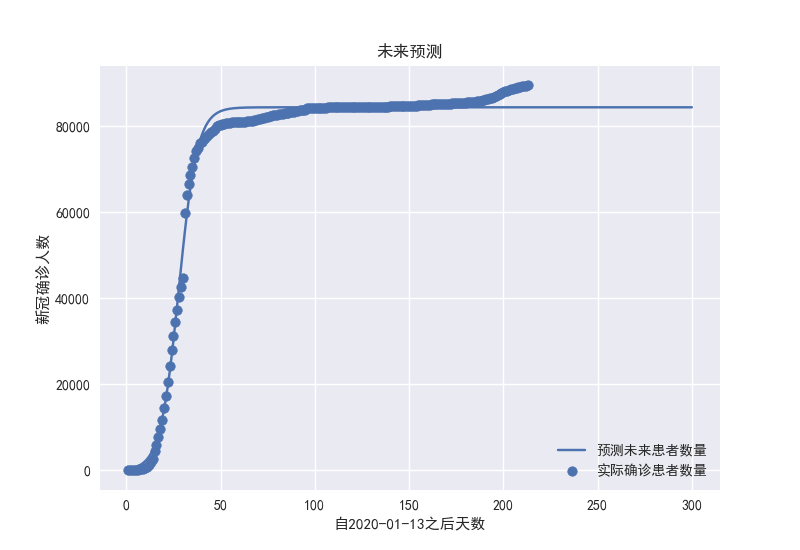
从散点图中可以看出，在80000新冠累积患者上误差较大。



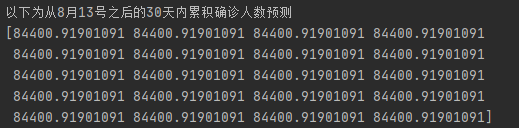
预测未来的疫情走势，8月12日是第213天。

future\_t = [i + 1 for i in range(0, 300)]# 预测中国疫情未来走势  
China\_future\_fit = logistic\_increase\_function(China\_p, future\_t)

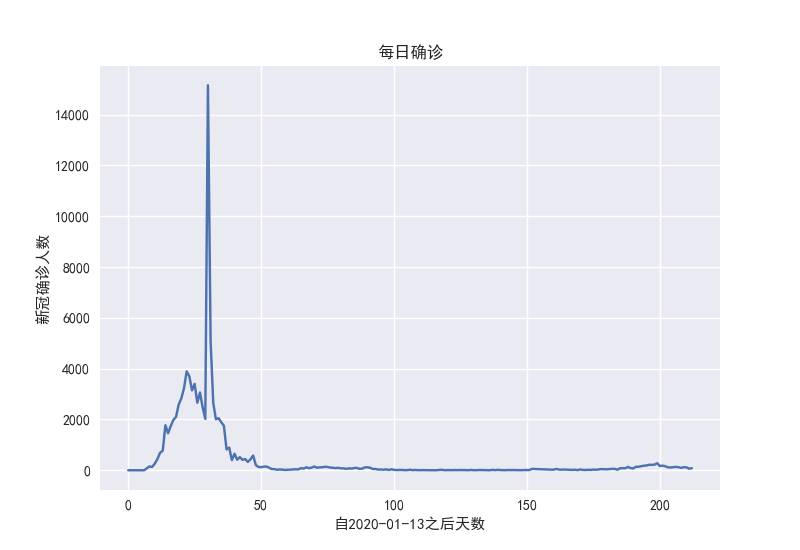
plt.scatter(t, China\_y, label="实际确诊患者数量")# 绘图  
plt.plot(future\_t, China\_future\_fit, label="预测未来患者数量")



根据1月13日-8月13日的数据，logistic模型预测结果显示我国疫情已稳定，不会发生增长，然而最近由于国外疫情影响，我国累积新冠确诊患者仍在微弱上升，模型不符合实际，效果一般。



第三部分是每日确诊患者的LSTM模型预测，首先观察自2020年1月13日之后天数的每日确诊。最高的在2月12日，自2月19日后趋于平稳。



设置随机种子，标准化数据。

np.random.seed(7)  
scaler = MinMaxScaler(feature\_range=(0, 1))  
dataset = scaler.fit\_transform(dataset.reshape(-1,1))

设置时间滑窗，创建训练集。

look\_back = 7

trainX, trainY = create\_dataset(train, look\_back)

trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))# 对训练集x做reshape

搭建lstm网络。

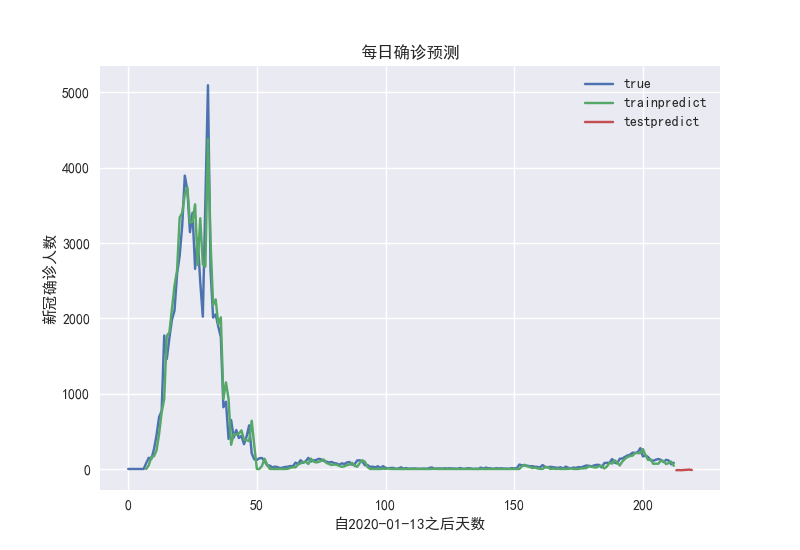
model = Sequential()  
model.add(LSTM(20, input\_shape=(1, look\_back)))#输出节点为25，输入的每个样本长度为look\_back  
model.add(Dense(1))#添加一个全连接层，输出维度为1  
model.compile(loss='mean\_squared\_error', optimizer='adam')#使用均方差做损失函数，优化器用adam  
reduce\_lr = ReduceLROnPlateau(monitor='val\_loss', patience=10, mode='max')  
model.fit(trainX, trainY, epochs=500, batch\_size=1, verbose=2, callbacks=[reduce\_lr])#训练模型，500epoch，批次为1，每一个epoch显示一次日志，学习率动态减小

预测后7天每日确诊人数。

testx = [0.]\*(7+look\_back)  
testx[0:look\_back] = train[-look\_back:]  
testx = np.array(testx)  
testx = testx.astype('float64')  
testPredict = [0]\*7  
for i in range(7):  
 testxx = testx[-look\_back:]  
 testxx = np.reshape(testxx, (1, 1, look\_back))  
 testy = model.predict(testxx)  
 testx[look\_back+i] = testy  
 testPredict[i] = testy  
testPredict = np.array(testPredict)  
testPredict = np.reshape(testPredict,(7,1))

反标准化。

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



从LSTM模型预测结果可以看出，预测后七天数据接近0。



RMSE均方根误差分数较好，但是由于现实干扰因素过多，模型效果不贴合实际，效果一般。