疫情大数据分析与预测系统

> 中国石油大学(华东) 大数据结课项目

一、项目简介

本项目是一个实时的大数据分析与预测系统。

二、项目模块

2.1 数据采集

1. 数据采集部分的代码位于 `DataSource` 文件夹下。
2. 部署环境： 阿里云EMS服务器 ； 操作系统 Ubuntu 20.04
3. 项目运行

```
pip3 install pipenv
nohup pipenv run python cron.py &
```

4. 模块介绍

```
# cron.py
def update_data():
    now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    update = '/usr/local/hadoop/bin/hdfs dfs -put -f
/usr/local/DataSource/Data/{file}.csv /Data'
    update2 = '/usr/local/hadoop/bin/hdfs dfs -put -f
/usr/local/DataSource/Wuhan-2019-nCoV.csv /Data'
    update = update.format(file = datetime.now().strftime("%Y-%m-%d"))
    cmds = [
        ["pipenv", "run", "python", "dataset.py"],
        ["pipenv", "run", "python", "data-join.py"],
        ["pipenv", "run", "python", "data-to-json.py"],
    ]
    for cmd in cmds:
        print(" ".join(cmd))
        print(subprocess.check_output(cmd).decode())
    os.system(update)
    os.system(update2)

    sckey = 'SCU102943T928a7573f9b4da0fff7483cbff235de25ef75bf0512c5'
    url = "https://sc.ftqq.com/%s.send?text=来自云服务器ECS ali_ubuntu&desp=
{now},疫情数据已自动更新"%sckey
    url = url.format(now=datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
    requests.get(url)

scheduler = BlockingScheduler(timezone="Asia/Shanghai")
scheduler.add_job(update_data, 'cron', minute="9")
scheduler.start()
```

从上面的代码可以知，运行 `cron.py` 之后, 每隔一个小时，就会运行一次。每次运行都会爬取丁香园中新冠的数据，并将数据存放在 `2020-xx-xx.csv` 中(其实也会更新总表 `Wuhan-2019-nCoV.csv`)。然后运行 `hdfs` 命令，将这两个文件强制推送到 `HDFS` 集群上(虽然只有一个数据结点，可以通过http://121.41.225.123:50070/访问到)。

每次运行之后的结果如下:

-rw-r--r--	hadoop	supergroup	44.96 KB	2020/6/30 下午11:50:09	1	128 MB	2020-06-30.csv
-rw-r--r--	hadoop	supergroup	44.97 KB	2020/7/1 下午11:50:08	1	128 MB	2020-07-01.csv
-rw-r--r--	hadoop	supergroup	44.97 KB	2020/7/2 下午11:50:10	1	128 MB	2020-07-02.csv
-rw-r--r--	hadoop	supergroup	44.98 KB	2020/7/3 下午11:50:09	1	128 MB	2020-07-03.csv
-rw-r--r--	hadoop	supergroup	44.98 KB	2020/7/4 下午11:09:08	1	128 MB	2020-07-04.csv
-rw-r--r--	hadoop	supergroup	44.99 KB	2020/7/5 下午5:09:08	1	128 MB	2020-07-05.csv
-rw-r--r--	hadoop	supergroup	6.51 MB	2020/7/5 下午5:10:02	1	128 MB	Wuhan-2019-nCoV.csv
-rw-r--r--	hadoop	supergroup	60 B	2020/6/30 下午2:10:35	1	128 MB	test

2.2 数据处理

1. 数据处理部分的代码位于 `Spark` 文件夹下
2. 部署环境：腾讯云 CVM ；操作系统：ubuntu 16.04
3. 项目运行

```
nohup python3 cron.py
```

4. 模块介绍

```
# cron.py

def update_data():
    os.system("python3 ./spark_update.py")

    sckey = 'SCU102943T928a7573f9b4da0fff7483cbff235de25ef75bf0512c5'
    url = "https://sc.ftqq.com/%s.send?text=来自云服务器ECS tencent_ubuntu:
spark运行正常&desp={now},疫情数据已自动更新"%sckey
    url = url.format(now=datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
    requests.get(url)

scheduler = BlockingScheduler(timezone="Asia/Shanghai")
scheduler.add_job(update_data, 'cron', minute="20")
scheduler.start()
```

运行之后，每隔一个小时都会运行一次 `spark_update.py` 。在 `spark_update.py` 中存放的是对之前爬虫上传到 `HDFS` 服务器上数据的处理程序。

spark_update的作用很简单，主要是将HDFS上的数据下载到本地，预处理将数据转成txt格式。通过Spark sql对数据进行处理，并将结果转为DataFrame将存到另一台服务器的Mysql数据库中。

```
# spark_update.py
import os
import pymysql
import pandas as pd
from pyspark.sql import Row
```

```
from pyspark.sql.types import *
from datetime import datetime
import pyspark.sql.functions as func
from pyspark.sql import SparkSession
from sqlalchemy import create_engine
from pyspark import SparkConf,SparkContext

# 判断当前目录下是否存在Data文件夹，如果没有就新建
dir_exists = os.system('ls | grep Data')
if dir_exists:
    os.system('mkdir Data')

# 下载文件在本地
filename = datetime.now().strftime("%Y-%m-%d") + '.csv'
file_exist = os.system('ls ./Data | grep {filename}'.format(filename =
filename)) #0 表示存在
if not file_exist:
    os.system('rm -f ./Data/*')
downloadfile = 'hdfs dfs -get hdfs://121.41.225.123:9000/Data/'+filename+'
./Data'
downloadfile = downloadfile.format(date = datetime.now().strftime("%Y-%m-
%d"))
status = os.system(downloadfile)

#格式转换
data = pd.read_csv('./Data/{filename}'.format(filename = filename))
textname = datetime.now().strftime("%Y-%m-%d") + '.txt'
os.system('rm -f ./Data/{textname}'.format(textname=textname))
with open('./Data/'+textname,'a+',encoding='utf-8') as f:
    for line in data.values:

f.write((str(line[0])+'\t'+str(line[1])+'\t'+str(line[2])+'\t'+str(line[3])+
'\t'+str(line[4])+'\t'

+str(line[5])+'\t'+str(line[6])+'\t'+str(line[7])+'\t'+str(line[8])+'\t'+str
(line[9])+'\t'
                +str(line[10])+'\n'))

# 创建spark
spark = SparkSession.builder.config(conf = SparkConf()).getOrCreate()

# 创建模式
fields = [StructField("date"  ,       StringType(),   False),
          StructField("country",      StringType(),   False),
          StructField("countryCode",  StringType(),   False),
          StructField("province" ,    StringType(),   False),
          StructField("provinceCode", StringType(),   False),
          StructField("city" ,        StringType(),   False),
          StructField("cityCode" ,    StringType(),   False),
          StructField("confirmed" ,  IntegerType(),   False),
          StructField("suspected" ,  IntegerType(),   False),
          StructField("cured" ,      IntegerType(),   False),
          StructField("dead",        IntegerType(),   False),]

rdd0 =
spark.sparkContext.textFile('file:///home/hadoop/Data/{textname}'.format(tex
tname=textname))
```

```
rdd1 = rdd0.map(lambda x:x.split("\t")).map(lambda p:
Row(p[0],p[1],p[2],p[3],p[4],p[5],p[6],int(p[7]),int(p[8]),int(p[9]),int(p[1
0])))
schema = StructType(fields)

# 注册 ncov_2019临时表
shemaUsInfo = spark.createDataFrame(rdd1,schema)
shemaUsInfo.createOrReplaceTempView("ncov_2019")

# 查询各省市的情况并将结果写入mysql数据库中
df1 = spark.sql("SELECT date ,province,confirmed,suspected,cured,dead FROM
ncov_2019 WHERE countryCode='CN' AND provinceCode != 'nan' AND city =
'nan'")
df1 = df1.withColumnRenamed("date","id")
connect = create_engine('mysql+pymysql://root:root@121.41.225.123:3306/mydb?
charset=utf8')
prop = {'user':'root','password':'root','driver':'com.mysql.jdbc.Driver'}
df1.write.jdbc("jdbc:mysql://121.41.225.123/mydb",'province_total','overwrit
e', prop)

# 删除 china_total_with_date中关于今天的数据
db = pymysql.connect("121.41.225.123", "root", "root", "mydb",
charset='utf8' )
cursor = db.cursor()
sql = "DELETE FROM china_total_with_date where date = '" +
datetime.now().strftime("%Y-%m-%d") + "'"
try:
   cursor.execute(sql)
   db.commit()
except:
   print("execute sql faild")
db.close()

# 将今天的情况追加到china_toal_with_date表中
df2 = spark.sql("SELECT DATE,confirmed,suspected,cured,dead FROM ncov_2019
WHERE countryCode = 'CN' and  provinceCode = 'nan'")
df2.write.jdbc("jdbc:mysql://121.41.225.123/mydb",'china_total_with_date','a
ppend', prop)
```

2.3 数据展示

1. 数据展示部分包含 前端(Client)、后端(Server)两部分。
2. 前端采用的是`vue.js`,后端采用的是`node.js`开发
3. 项目的演示地址`http://101.200.173.171/`
4. 项目运行

   （1）后端运行:

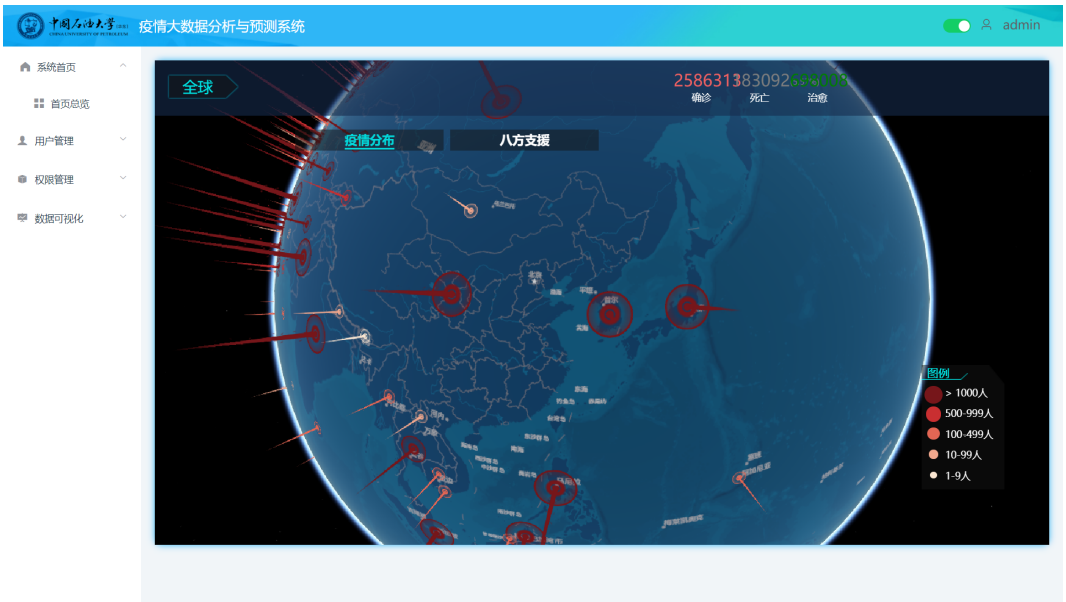
```
cd Server
node app.js

cd Global-3D
ndoe app.js
```

（2）前端运行

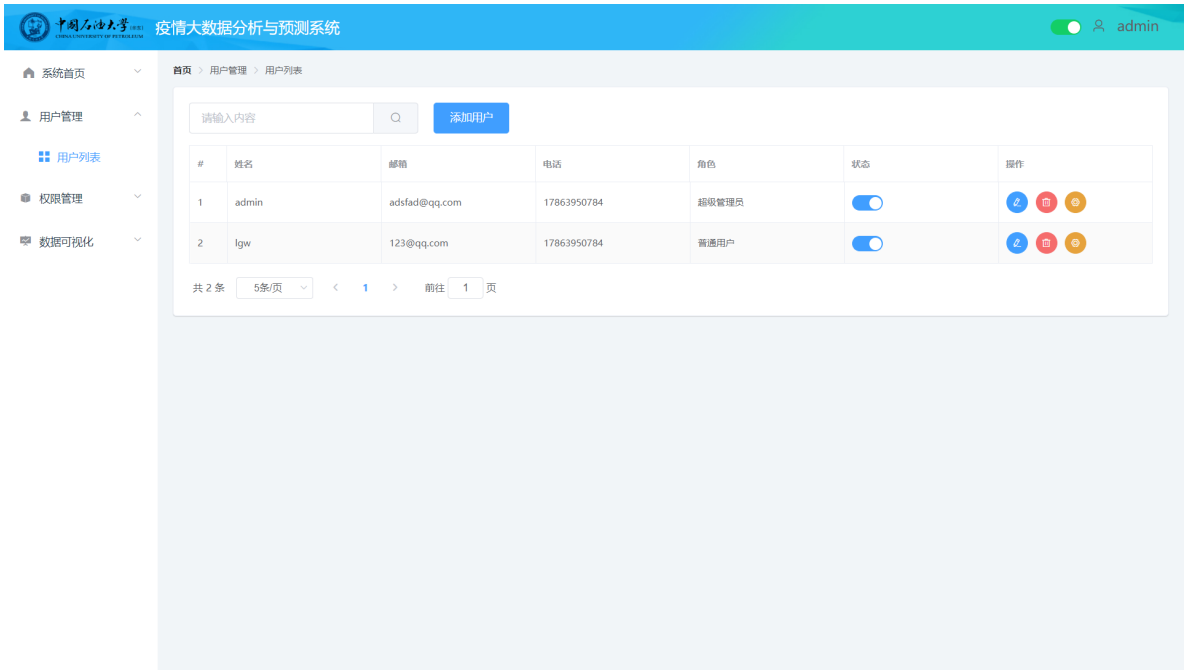
```
vue ui
```
之后将前端项目Client导入。点击运行

5. 项目截图

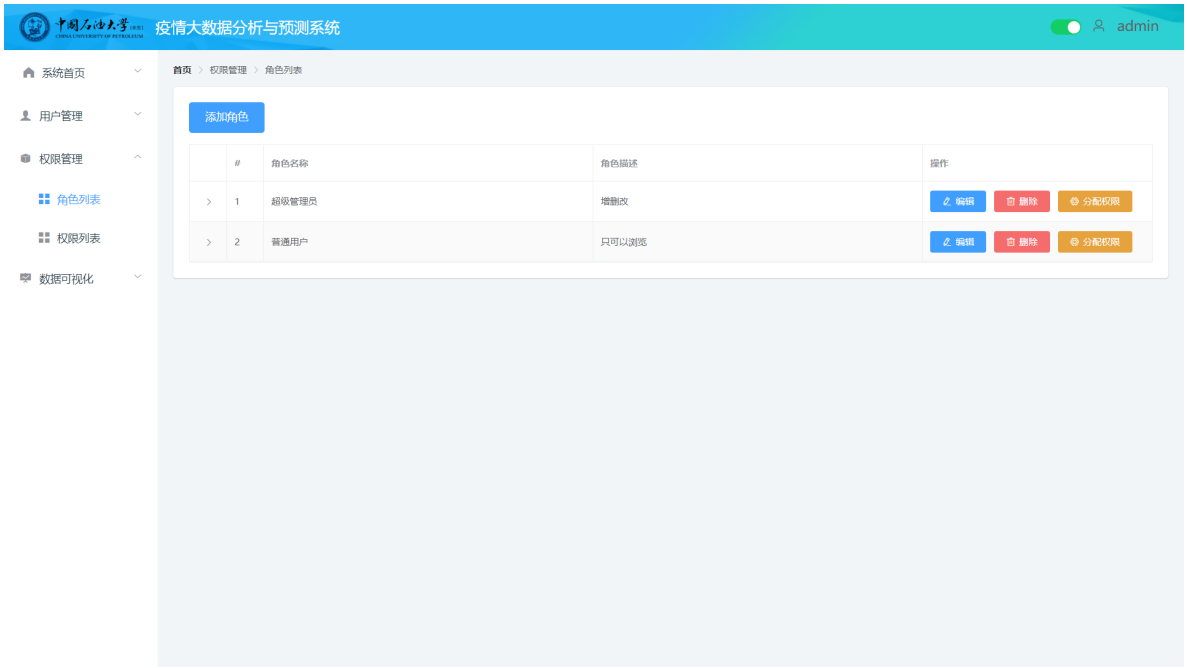
(1) 首页总览



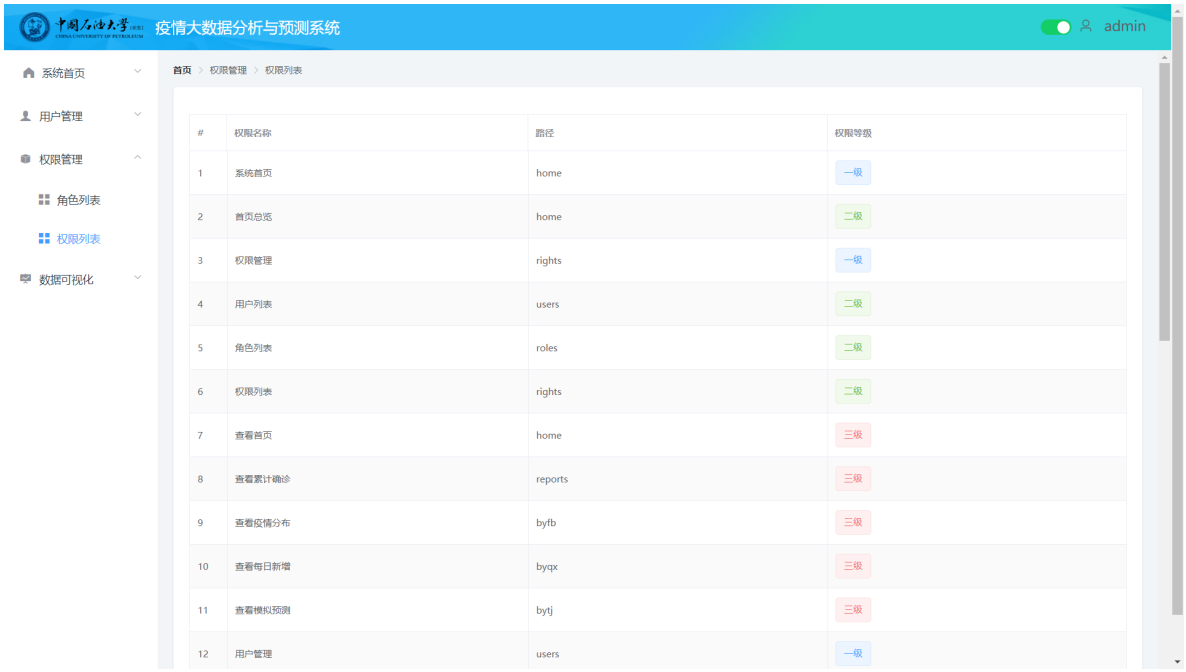
(2) 用户管理



(3) 权限管理 - 角色列表

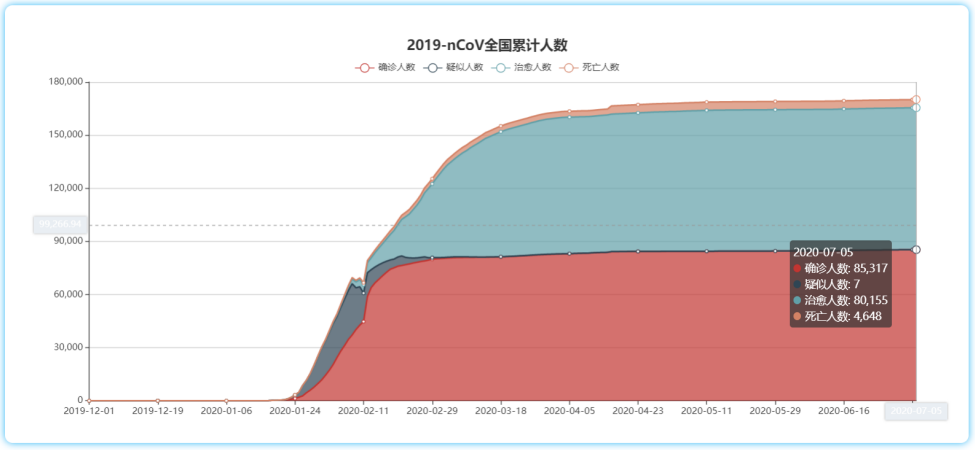


权限管理-权限列表

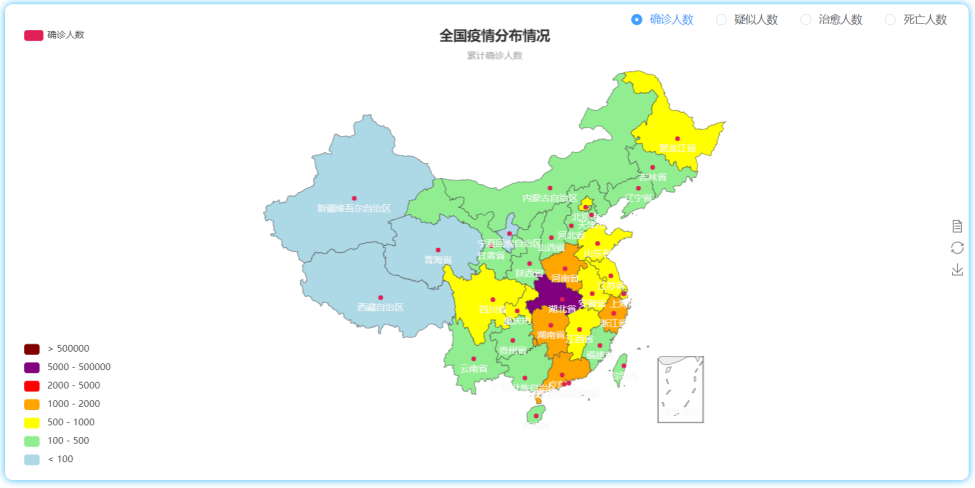


（4）数据可视化

系统首页
用户管理
权限管理
数据可视化
累计确诊
疫情分布
每日新增
模拟预测

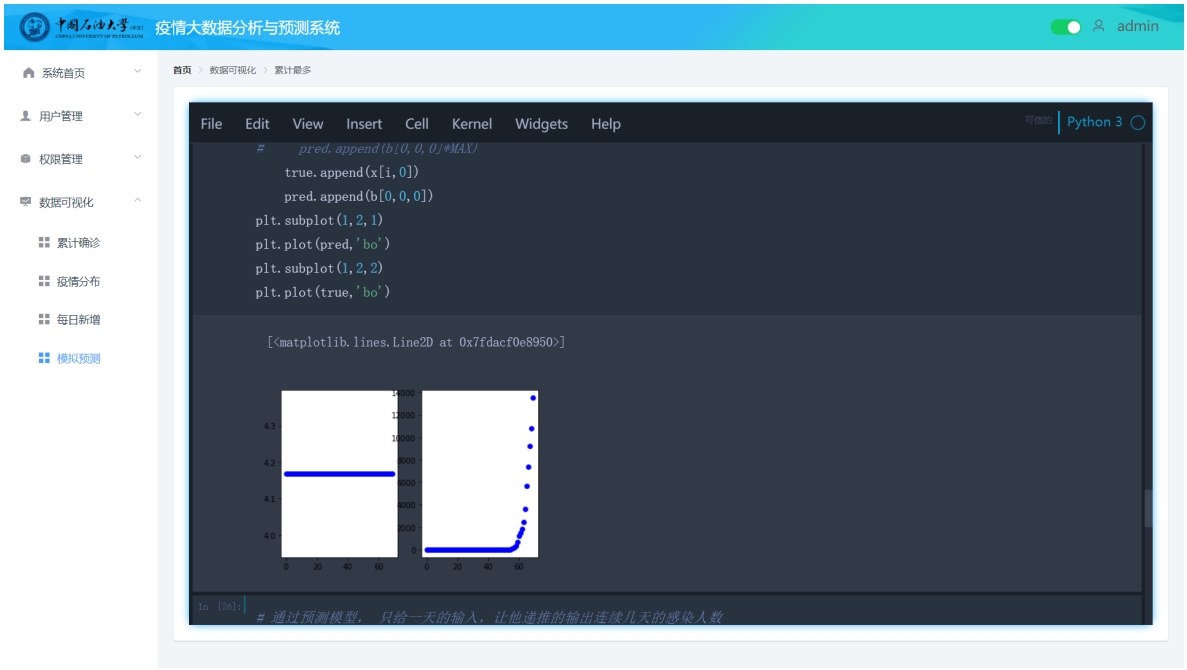


系统首页
用户管理
权限管理
数据可视化
累计确诊
疫情分布
每日新增
模拟预测



系统首页
用户管理
权限管理
数据可视化
累计确诊
疫情分布
每日新增
模拟预测

2.4 数据预测


疫情大数据分析与预测系统
admin
系统首页
用户管理
权限管理
数据可视化
累计确诊
疫情分布
每日新增
模拟预测
首页 > 数据可视化 > 累计最多
File Edit View Insert Cell Kernel Widgets Help
Python 3
# pred.append(b[0,0,0]*MAX)
true.append(x[i,0])
pred.append(b[0,0,0])
plt.subplot(1,2,1)
plt.plot(pred,'bo')
plt.subplot(1,2,2)
plt.plot(true,'bo')
[<matplotlib.lines.Line2D at 0x7fdacf0e8950>]
In [26]:
通过预测模型， 只给一天的输入，让他递推的输出连续几天的感染人数