大数据分析实训大作业报告

1. ***功能描述***

本作业从MongoDB取出中国和世界疫情的相关数据进行处理和分析，然后将处理好的数据以Json的形式分别加载到相应的网站，基于ajax异步加载实现轮询，设定每一个小时发起一次请求，从相应的网站获取数据。基于flask实现了在大屏上展示中国&世界疫情实时追踪，使用flask_apscheduler实现每天的十点钟爬取最新的疫情数据（每天九点多数据源网站更新）。共实现了全国现有确诊、累计确诊总数，现有确诊排名前八的国家，近2个月全国疫情走势图，基于前两月新增确诊数据预测未来一周新增确诊，当日新增确诊、累计确诊、本土无症状的各区间累计情况，现有确诊排名前八的省份，直辖市现有确诊情况，中国疫情地图八个模块，以及在左上角显示当前展示的数据是何时更新的，右上角展示当前时间。

static存储的大部分为静态元素，例如一些js文件，图片，字体，文字样式等等。

static\css\comon0.css文件是css文件，是对大屏展示网站进行美化页面，修饰标签

static\font\DS-DIGIT.TTF 是css的一种字体，数字液晶字体

static\images 储存的是大屏的背景照片

static\js\china.js 中国地图的函数库,使数据中的省份和地图能够对应起来

static\js\echarts.min.js ECharts纯Javascript的图表库,支持多类图表，同时提供标题等7 个可交互组件，支持多图表、组件的联动和混搭展现。

static\js\index.html 网页的背景，实现一个动态粒子的效果

static\js\jquery.js Jquery是一个优秀的Javascrīpt框架,Jquery能更方便地处理HTML documents、events、实现动画效果，并且方便地为网站提供AJAX交互。

static\js\loading.js 是一个加载动画，这个加载动画使得网页加载具有淡入的效果，并且确保了网页在拉伸的时候能自适应调整布局。

static\js\showtime.js 实现时间的显示，时间每秒刷新。

templates\index.html 网页源码，控制网页整体布局及数据的显示，echarts的一些参数设置，设置要实现的功能模块option，使用ajax异步加载数据，控制网站轮询更新数据。

app.py 定义了每个模块ajax跳转的网站，网站加载每个模块需要的数据，并且采用了flask ApScheduler定时器功能，定时在每天的十点运行数据爬取模块，更新数据，启动flask网页服务器。

data.py 记载MongoDB中存储的疫情数据，对数据按照实现模块的需求进行处理。

dataCrawler.py 爬取最新的国内外疫情数据，存储到MongoDB中，并更新更新时间。

model.py 利用前两个月的新增确诊数据，训练出一个Holt-Winters季节性预测模型，并且利用这个模型对未来七天的新增确诊人数进行预测。

2. ***数据存**储***

本作业使用的是MongoDB数据库存储数据，MongoDB是一个基于分布式文件存储 的数据库。MongoDB 文档数据库的存储结构分为四个层次，从小到大依次是：键值对、文档（document）、集合（collection）、数据库（database）。

本作业使用了一个数据库coronavirus三个集合chinaTotal、chinahistory、chinaprovince、worldcountry来存储数据。

chinaTotal中只存了一个文档，文档中name是存储的“中国”，nowConfirm存储的是中国现有确诊人数，confirm存储的是中国累计确诊人数，upgradetime存储的是当前数据库中存储的数据的更新时间。

chinahistory中存储的是中国两个月时间内的当日新增确诊人数和当日现有确诊人数，date存储的是时间日期，confirm存储的是当日现有确诊人数、confirmAdd存储的是当日新增确诊人数。

chinaprovince中存储的是中国各省份现有确诊人数、累计确诊人数、现有无症状人数、累计死亡人数、本地新增确诊人数。name存储的是省份名称，nowConfirm存储的是该省份现有确诊人数，confirm存储的是该省份累计确诊人数，wzz存储的是该省份无症状感染者人数、dead存储的是该省份累计死亡人数、localAdd存储的是该省份本地新增确诊人数。

worldcountry中存储的是世界各个国家的疫情数据，name存储的是国家名称，nowConfirm存储的是该国家现有确诊人数，confirm存储的是该国家累计确诊人数，dead存储的是该国家累计死亡人数。

3. *大屏展示代码*

3.1 *python代码*

dataCrawler.py在这个文件中写了一个类crawlData（数据爬虫），这个类一共有五个属性

```
def __init__(self):
    self.worldUrl = 'https://api.inews.qq.com/newsqa/v1/automation/modules/list?modules=FAutoCountryConfirmAdd,WomWorld,WomAboard'
    self.historyurl = 'https://api.inews.qq.com/newsqa/v1/query/inner/publish/modules/list?modules=chinaDayList,chinaDayAddList,nowConfirmStatis,provinceCompare
    self.provinceurl = 'https://api.inews.qq.com/newsqa/v1/query/inner/publish/modules/list?modules=statisGradeCityDetail,diseaseh5Shelf'
    self.client = MongoClient(
        "mongodb+srv://dbh:dbh04051204@cluster0.uspgg.mongodb.net/myFirstDatabase?retryWrites=true&w=majority")
    self.db = self.client.coronavirus
```

worldUrl是要爬取的世界疫情数据的网址，historyUrl是爬取中国两个月内历史数据的网址，provinceUrl是爬取中国整体及各省份疫情数据的网址，client是连接我的MongoDB数据库，db是储存所需要用的数据的数据库coronavirus。

类中还写了get_page_world_data、save_MongoDB_world、get_page_history_data、save_MongoDB_history、get_page_province_data、save_MongoDB_province，分别用于从相应的网站中爬取中国近两月历史疫情数据、全国整体及各省份疫情数据、世界各国疫情数据，其中save_MongoDB_province方法还将更新疫情数据的时间写入了chinaTotal集合中。

run（）方法是用来运行上述爬取疫情数据的方法，所有方法运行完成后，会输出“数据更新完毕”

```
def run(self):
    worldData = self.get_page_world_data()
    self.save_MongoDB_world(worldData)
    historyData = self.get_page_history_data()
    self.save_MongoDB_history(historyData)
    provinceData = self.get_page_province_data()
    self.save_MongoDB_province(provinceData)
    print("数据更新完毕")
```

data.py在这个文件中写了一个类getData，这个类一共有六个属性

```
def __init__(self):
    self.client = MongoClient(
        "mongodb+srv://dbh:dbh04051204@cluster0.uspgg.mongodb.net/myFirstDatabase?retryWrites=true&w=majority")
    self.db = self.client.coronavirus
    self.chinahistory = self.db.chinahistory
    self.worldcountry = self.db.worldcountry
    self.chinaprovince = self.db.chinaprovince
    self.chinaTotal = self.db.chinaTotal
```

client是连接我的MongoDB数据库，db是储存所需要用的数据的数据库coronavirus，chinahistory是coronavirus数据库中国近两个月历史疫情数据集合，worldcountry是coronavirus数据库中世界各国疫情数据集合，chinaprovince是coronavirus数据库中国各个省份疫情数据集合，chinaTotal是coronavirus数据库中国整体数据集合。

Jsonmongodb（）方法是获取我的数据库中chinaprovince集合中的数据，即各个省份的疫情数据，将集合中的各个文档中的数据加入到output列表中，将列表返回。列表中的元素是一个字典，字典中包括各个省份的名字、现存感染人数、累计感染人数、无症状感染人数、累计死亡人数，这个方法返回的数据是大屏展示的中国疫情地图所需要的数据。

echart1（）方法返回现有确诊排名前八的国家模块所需要的数据，该方法首先从数据库的worldcountry集合中取出所有数据 ，然后将从chinaTotal中中国的整体数据加入，转换为列表类型，按照现有确诊人数进行降序排序后取出排名前八的国家，将该数据返回。

echart2（）方法返回基于前两月新增确诊数据预测未来一周新增确诊模块需要的数据，该方法从数据库中chinahistory集合中取出数据的日期和当日新增确诊两列，然后初始化一个Model类，使用model的预测方法，将前两个月的新增确诊数据传入，然后得到预测的未来七天的新增确诊人数，将近一个月的新增确诊人数和预测的未来七天的新增确诊人数结合到一起后，以字典的形式返回，Date的值是这整个周期的日期，add1是预测的未来七天的新增确诊人数，add2是中国近两个月的新增确诊人数。

echart3（）方法返回当日新增确诊、累计确诊、本土无症状的各区间累计情况模块的数据，该方法首先从数据库的chinaprovince集合中获取数据，然后计算当日新增确诊、累计确诊、本土无症状处于0", "1-99", "100-999", "1000-9999","10000+"区间的省份数量。将得到的数据以列表的形式返回，列表中的元素为字典。

```
    data['data1'] = [{"name": "0", "value": x11}, {"name": "1-99", "value": x12}, {"name": "100-999", "value": x13},
                     {"name": "1000-9999", "value": x14}, {"name": "10000+", "value": x15}]
    data['data2'] = [{"name": "0", "value": x21}, {"name": "1-99", "value": x22}, {"name": "100-999", "value": x23},
                     {"name": "1000-9999", "value": x24}, {"name": "10000+", "value": x25}]
    data['data3'] = [{"name": "0", "value": x31}, {"name": "1-99", "value": x32}, {"name": "100-999", "value": x33},
                     {"name": "1000-9999", "value": x34}, {"name": "10000+", "value": x35}]
    data['xAxis'] = ["0", "1-99", "100-999", "1000-9999", "10000+"]
    data['title']=['当日新增确诊','累计确诊','本土无症状']
    return data
```

echart4（）方法返回近2个月全国疫情走势图模块需要的数据，该方法从数据库中chinahistory集合中取出数据的日期、本地新增确诊、本地现有确诊三列数据，然后将date作为xAxis，返回字典类型的数据。

echart5（）方法返回现有确诊排名前8的省份模块的数据，该方法首先从数据库的chinaprovince集合中取出所有数据，转换为列表类型，按照现有确诊人数进行降序排序后取出排名前八的省份，将该数据返回。

echart6（）方法返回直辖市现有确诊情况模块的数据，从数据库的chinaprovince集合中找到“北京”，“上海”，“天津”，“重庆”四个直辖市的现有确诊数据，方法中的color 是圆环的颜色，radius是每个圆环的半径，将四个直辖市按照现有确诊人数的多少降序排序，人数越多的市圆环的半径越大。将直辖市的确诊人数数据和颜色、半径对应后，以字典类型返回。

title（）方法 以字典类型返回全国现有确诊、累计确诊总数的值，数据库更新数据的时间，并返回各个模块展示的标题。

```
def Title(self):
    data = {}
    data["time"] = "数据更新于:" + self.db.chinaTotal.find_one({"name": "中国"})['upgradetime']
    data['counter'] = {
        "counter1": {'name': "全国现有确诊", 'value': self.db.chinaTotal.find_one({"name": "中国"})['nowConfirm']},
        "counter2": {'name': '累计确诊总数', 'value': self.db.chinaTotal.find_one({"name": "中国"})['confirm']}}
    data['title']=['现有确诊排名前8的国家','预测未来7天趋势','近2个月全国疫情走势图','现有确诊排名前8的省份','直辖市现有确诊情况']
    return data
```

model.py 该文件主要实现了用近两个月中国的历史新增确诊人数构建并训练时序模型，对未来七天的新增确诊人数展开预测。

该文件中构建了一个类predictModel，其中

model（）方法需要一个data参数，这个参数就是data.py文件中echart2（）方法传进来的从数据库中chinahistory集合中取出数据的日期和当日新增确诊两列数据。然后使用pandas将日期转为datetime类型，然后将date作为value的index，从 statsmodels.tsa.api 导入 ExponentialSmoothing，这种模型叫做Holt-Winters季节性预测模型，它是一种三次指数平滑预测，其背后的理念就是除了水平和趋势外，还将指数平滑应用到季节分量上。我经过很多次试验后选择了 seasonal_period = 25作为每周重复的数据。也可以调整其它其它参数，我将trend，seasonal都调成了'mul'模式。然后使用timedelta函数，计算出未来七天起始的日期和结束的日期，将起始日期和结束日期传入预测函数，得到预测结果，将结果返回。

```
ts=trainData['value']
model_fit=ExponentialSmoothing(ts,seasonal_periods=25,trend='mul',seasonal='mul').fit()
startstart=str(date_index[-1]+timedelta(days=1)).split(" ")[0]
endend=str(date_index[-1]+timedelta(days=7)).split(" ")[0]
predicts=model_fit.predict(start=startstart,end=endend)
```

predict（）方法，接收传进来的数据，然后运行model方法，训练模型并预测。

app.py中导入了crawlData类，并新建了此类的实例，运行类的run（）方法，在上边写了，crawlData的run（）方法会运行爬取疫情数据的一些方法，写在app.py文件的最开头是为了每次启动flask服务之前，先更新数据，以保证每次大屏展示的数据都是最新的。

Config这个类继承了object类，这个类使用来设定定时器任务，即设定任务要进行的操作即运行run（）方法，这个run（）方法就是运行上边crawlData的实例的run（）方法，触发方式是corn，即在特定时间运行，我设置的是每天十点，并且设定可以访问定时器页面，在启动flask任务后，访问http://127.0.0.1:5000/scheduler/jobs，就可以看到定时任务。初始化flask app，并且将设定好的config导入，然后初始化scheduler （任务调度器），并将调度器和app连接起来，启动调度器。

```
class Config(object):    # 定时任务
    JOBS = [{
        'id': 'job1',
        'func': 'app:run',
        'args': (),
        'trigger': 'cron',
        'day': '*',
        'hour': '10',
        'minute': '00'
    }
    ]
    SCHEDULER_API_ENABLED = True
    SCHEDULER_API_PREFIX = '/scheduler'

scheduler = APScheduler()
scheduler.init_app(app)
scheduler.start()
```

导入getData类，并且初始化一个实例getDataSource，这个类是data.py文件中定义处理并返回所需要数据的类。

jsonmongodb（）　，使用 route() 装饰器把函数绑定到http://127.0.0.1:5000/province，调用getDataSource.jsonmongodb()函数，使得该网站返回的是json形式的中国疫情地图所需要的数据。

echart1（）,使用 route() 装饰器把函数绑定到http://127.0.0.1:5000/echart1,

调用getDataSource.echart1()函数,获得确诊排名前八的国家模块所需要的数据，并且将数据化为json形式传到网页中。

echart2（）,使用 route() 装饰器把函数绑定到http://127.0.0.1:5000/echart2,

调用getDataSource.echart2()函数,获得基于前两月新增确诊数据预测未来一周新增确诊模块需要的数据，并且将数据化为json形式传到网页中。

echart3（）,使用 route() 装饰器把函数绑定到http://127.0.0.1:5000/echart3,

调用getDataSource.echart3()函数,获得当日新增确诊、累计确诊、本土无症状的各区间累计情况模块的数据，并且将数据化为json形式传到网页中。

echart4（）,使用 route() 装饰器把函数绑定到http://127.0.0.1:5000/echart4,

调用getDataSource.echart4()函数,获得近2个月全国疫情走势图模块需要的数据，并且将数据化为json形式传到网页中。

echart5（）,使用 route() 装饰器把函数绑定到http://127.0.0.1:5000/echart5,

调用getDataSource.echart5()函数,获得现有确诊排名前8的省份模块的数据，并且将数据化为json形式传到网页中。

echart6（）,使用 route() 装饰器把函数绑定到http://127.0.0.1:5000/echart6,

调用getDataSource.echart6()函数,获得直辖市现有确诊情况模块的数据，并且将数据化为json形式传到网页中。

Title（），使用 route() 装饰器把函数绑定到http://127.0.0.1:5000/Title，

调用getDataSource.Title（）函数，获得全国现有确诊、累计确诊总数的值，数据库更新数据的时间，各个模块展示的标题。并且将数据json化，传入网页中。

index（），使用 route() 装饰器把函数绑定到http://127.0.0.1:5000/，

设置大屏的标题为'中国&国际疫情实时追踪'，并且使用render_template（）方法，将网页跳转到index.html。

运行app.run（）方法，启动flask服务。

3.2 *外部S代码*

static\js\china.js 中国疫情地图是基于echarts.js和china.js绘制图像。中国地图的函数库,使数据中的省份和地图能够对应起来

static\js\echarts.min.js 是ECharts一个Javascript的图表库,支持多类图表，同时提供标题等7 个可交互组件，支持多图表、组件的联动和混搭展现，使得我们在html文件中写的echart图表能够展示出来。

static\js\jquery.js Jquery是一个优秀的Javascrīpt框架,Jquery能更方便地处理HTML documents、events、实现动画效果，并且方便地为网站提供ajax交互。

static\js\loading.js 是一个加载动画，这个加载动画使得网页加载具有淡入的效果，并且确保了网页在拉伸的时候能自适应调整布局。

static\js\showtime.js 实现时间的显示，时间每秒刷新。

3.3 *HTML文件*

本作业用到了两个HTML文件，分别是static\js\index.html和templates\index.html。

static\js\index.html 加载了网页的背景，实现一个动态粒子的效果

templates\index.html 是app.py中index（）函数跳转的网页，也是整个大屏展示的HTML文件。

```
<head>
    <meta charset="utf-8">
    <title>段博涵的大屏幕</title>
    <script type="text/javascript" src="../static/js/jquery.js"></script>
    <link rel="stylesheet" href="../static/css/comon0.css">
</head>
<script type="text/javascript" src="../static/js/loading.js"></script>
<script type="text/javascript" src="../static/js/echarts.min.js"></script>
<script type="text/javascript" src="../static/js/china.js"></script>
```

这段HTML代码导入了外部js文件，和css文件，并且定义了网站的名字为段博涵的大屏幕。

```
<!--这是一个动态的粒子效果图，设置了透明度为0.2-->
<div class="canvas" style="...">
    <iframe frameborder="0" src="../static/js/index.html" style="..."></iframe>
</div>
```

导入static\js\index.html文件，实现动态的粒子效果图。

```
<!--这是一个在正式页面加载进来之前显示的加载小动画，文字可以修改-->
<div class="loading">
    <div class="loadbox"><img src="../static/picture/loading.gif"> 我正在加载中...</div>
</div>
```

实现了等待加载时等待小动画效果

```
<div class="head">

    <h1>{{ title }}</h1>
    <div class='upgradeTime' id="UpGrade"></div>
    <div class="weather">
        <img src="../static/picture/weather.png"><span>天气</span>
        <span id="showTime"></span>
    </div>

    <!--    尝试放到js里面-->
    <script type="text/javascript" src="../static/js/showtime.js"></script>

</div>
```

导入外部的showtime.js文件，实现了在大屏右上角展示时间和天气的功能。并且定义了更新时间模块，设置“id=UpGrade”。

```
<div class="mainbox">
    <ul class="clearfix">

        <!--          第1栏 -->
        <li>
            <div class="boxall" style="...">
                <div class="alltitle" id="echart1_title"></div>
                <div class="allnav" id="echart1"></div>
                <div class="boxfoot"></div>
            </div>
            <div class="boxall" style="...">
                <div class="alltitle" id="echart2_title"></div>
                <div class="allnav" id="echart2"></div>
                <div class="boxfoot"></div>
            </div>
            <div class="boxall" style="...">
                <div style="...">
                    <div class="sy" id="fb1"></div>
                    <div class="sy" id="fb2"></div>
                    <div class="sy" id="fb3"></div>
                </div>
                <div class="boxfoot">
```

将大屏划分为3栏，这是最左边的一栏，这一栏一共放了现有确诊排名前八的国家，基于前两月新增确诊数据预测未来一周新增确诊，当日新增确诊、累计确诊、本土无症状的各区间累计情况三个模块。定义了这三个模块的id，以便在后边进行引用。

```
<div class="bar">
    <!--                放数字显示区域 -->
    <div class="barbox">
        <ul class="clearfix">
            <li class="pulll_left counter" id="counterValue1"></li>
            <li class="pulll_left counter" id="counterValue2"></li>
        </ul>
    </div>
    <!--                放文字显示区域 -->
    <div class="barbox2">
        <ul class="clearfix">
            <li class="pulll_left" id="counter1name"></li>
            <li class="pulll_left" id="counter2name"></li>
        </ul>
    </div>
</div>
<div class="map">
    <!--                lbx 三角形组成的网 -->
    <div class="map1"><img src="../static/picture/lbx.png"></div>
    <!--                jt 围绕着地球旋转的一个光环 -->
    <div class="map2"><img src="../static/picture/jt.png"></div>
    <!--                map 是一个小地球 -->
    <div class="map3"><img src="../static/picture/map.png"></div>
    <!--                这里估计是一个中国地图-->
```

第二栏，放置全国现有确诊、累计确诊总数和中国疫情地图模块

```
        <div class="boxall" style="...">
            <div class="alltitle" id="echart4_title"></div>
            <div class="allnav" id="echart4"></div>
            <div class="boxfoot"></div>
        </div>
        <div class="boxall" style="...">
            <div class="alltitle" id="echart5_title"></div>
            <div class="allnav" id="echart5"></div>
            <div class="boxfoot"></div>
        </div>
        <div class="boxall" style="...">
            <div class="alltitle" id="echart6_title"></div>
            <div class="allnav" id="echart6"></div>
            <div class="boxfoot"></div>
        </div>
```

第三栏，放置了近2个月全国疫情走势图、现有确诊排名前八的省份、直辖市现有确诊情况三个模块。

function refreshTime()，这个是更新时间模块的方法，使用ajax技术，从"http://127.0.0.1:5000/Title" + "?n=" + Math.random()获取更新数据的时间，在网站后面加上一个随机数字是为了避免浏览器直接读取缓存，让浏览器每次都以为是一个新的请求，使得数据在更换之后重新加载可以加载新的数据。在这个函数中，我直接使用css selector找到更新时间模块的id，将时间传入模块的text。最后使用轮询技术，每隔一个小时，重新加载一次这个function。

```
<script>
    function refreshTime() {
        $.ajax({
            type: 'GET',
            url: "http://127.0.0.1:5000/Title" + "?n=" + Math.random(),
            dataType: 'json',
            success: function (data) {
                console.log(data);
                $('#UpGrade').text(data['time']);
            }
        })
    }

    refreshTime()
    var processTime = window.setInterval('refreshTime()', 3600000)
</script>
```

***function refreshCounter()* *，*这个是给全国现有确诊、累计确诊总数模块传输数据的方法，同样使用了ajax技术，从 "http://127.0.0.1:5000/Title" + "?n=" + Math.random()中加载数据，

使用css selector找到模块的id，将数据传入模块。使用轮询技术，每隔一个小时，重新加载一次这个function。

function refreshTitle(),***这个函数使用了ajax技术，从"http://127.0.0.1:5000/Title" + "?n=" + Math.random()获取各个模块的标题，使用css selector找到各个模块的标题的id，将标题传入，使用轮询技术，每隔一个小时，重新加载一次这个function。

function refresh1()，***这个函数实现现有确诊排名前八的国家模块，使用ajax技术，"http://127.0.0.1:5000/echart1" + "?n=" + Math.random()在控制台加载数据，然后使用echart，通过查找ID，初始化模块，将背景设为蓝色，用data中的“name”即国家名称作为图表的x轴，“value”即现有确诊人数的值作为y轴，grid是位置参数，画出柱状图。使用tooltip，将鼠标放置在柱子上，会显示国家名称和确诊人数。使用轮询技术，每隔一个小时，重新加载一次这个function。

function refresh2*****()*****，***这个函数实现基于前两月新增确诊数据预测未来一周新增确诊模块，函数通过ajax技术，从 "http://127.0.0.1:5000/echart2" + "?n=" + Math.random()获得数据，然后使用echart，通过查找ID，初始化模块，option中的grid是图表位置的参数，其中蓝色的实线部分是历史数据，通过data的“add2”获取，绿色的虚线部分是预测数据，每个点代表未来通过data的“add1”获取。使用轮询技术，每隔一个小时，重新加载一次这个function。

function refresh3_1*****()*****,*****function refresh*****3_2*****()*****，*****function refresh*****3_3*****()*****，***这三个个函数实现当日新增确诊、累计确诊、本土无症状的各区间累计情况模块。他们都使用ajax从网址"http://127.0.0.1:5000/echart3" + "?n=" + Math.random()获取数据，然后使用echart，通过查找ID，初始化模块，数据分别为data[‘data1’],data[‘data2’],data[‘data3’]，各个区间由不同的颜色区分，鼠标放上去之后浮标显示为各个区间的数量和比例，图表中占据圆环的大小代表数量的多少。三个函数都通过轮询技术，每隔一小时重新记载一次这个function。

function refresh4()，***这个函数实现了近2个月全国疫情走势图模块， 从"http://127.0.0.1:5000/echart4" + "?n=" + Math.random(),然后使用echart，通过查找ID，初始化模块，绿色的线代表本地现有确诊，蓝色的线代表本地新增确诊，分别传入data[‘data0’],data[‘data1’]。将鼠标放上去，就会显示日期和当日的本地新增确诊和现有确诊人数。通过轮询技术，每隔一小时重新记载一次这个function。

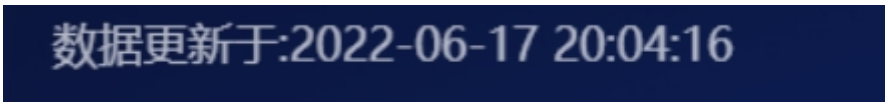
function refresh*5*****()*****，***这个函数实现了现有确诊人数排名前八的省份模块，通过获取"http://127.0.0.1:5000/echart5" + "?n=" + Math.random()的数据，然后使用echart，通过查找ID，初始化模块，将省份名称作为x轴，现有确诊人数作为y轴，展示为柱状图。通过轮询技术，每隔一小时重新记载一次这个function。

function refresh6()，这个函数实现了直辖市现有确诊情况模块，通过获取"http://127.0.0.1:5000/echart6" + "?n=" + Math.random()的数据，然后使用echart，通过查找ID，初始化模块，将排序好的数据传入环形图，有颜色的部分也大表示确诊人数越多，最外层圆环表示的直辖市是确诊人数最多的直辖市，越往里越少。通过轮询技术，每隔一小时重新记载一次这个function。

function refresh_map() ，这个函数实现了中国疫情地图的模块，模块的数据加载自"http://127.0.0.1:5000/province" + "?n=" + Math.random(),然后使用echart，通过查找ID，初始化模块。地图按照确诊人数的多少区分，确诊颜色越红代表该省份确诊人数越多，若没有确诊，则地图显示为蓝色，将鼠标放置在省份上，会显示该省份的名称、现有确诊人数、累计确诊人数、累计死亡人数、无症状感染者人数。地图的右下角，显示各个颜色代表的人数区间，这是通过option的pieces模块，将人数分级，并且对应各自的颜色实现的。通过轮询技术，每隔一小时重新记载一次这个function。

4. *系统实现及部署*

4.1 *系统实现*



数据库更新数据时间

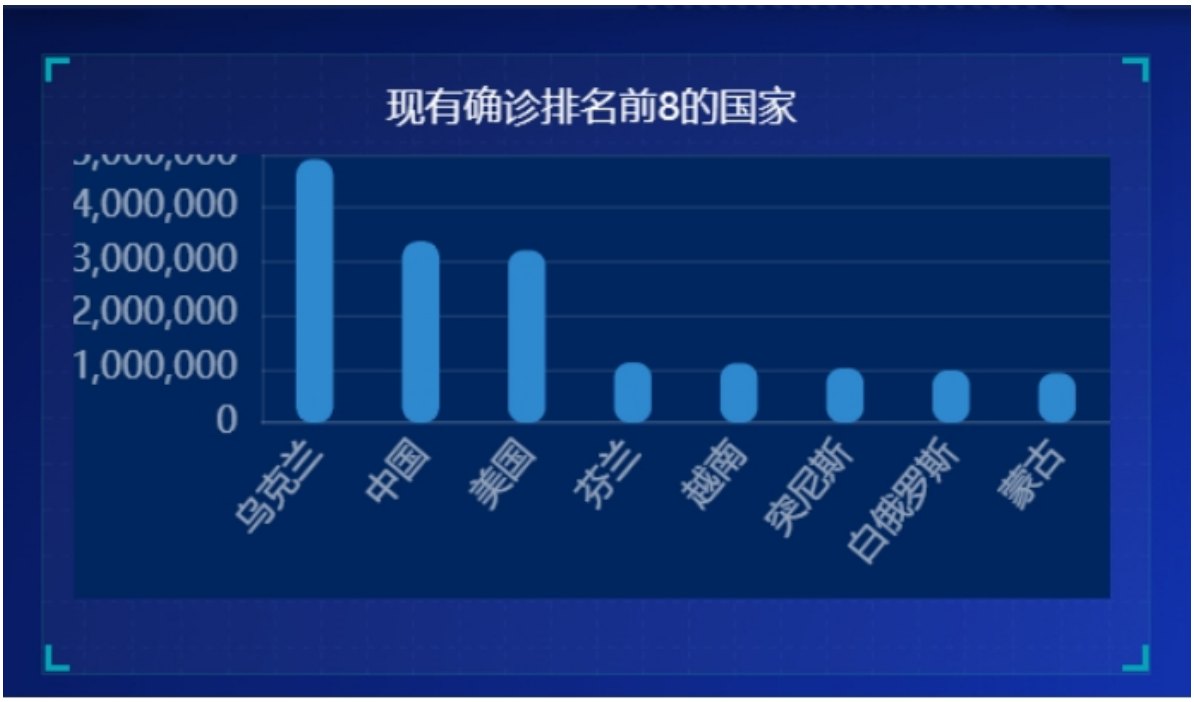
通过每次更新数据库数据时更新时间戳，然后在html文件中创建该模块，在css文件中设置模块的位置及属性，通过ajax技术，获取时间戳，然后通过id定位到该模块，将时间传入该模块的text属性。

当前时间

通过导入外部的showtime.js，然后在html文件中创建该模块，在css文件中设置模块的位置及属性，实时更新时间。

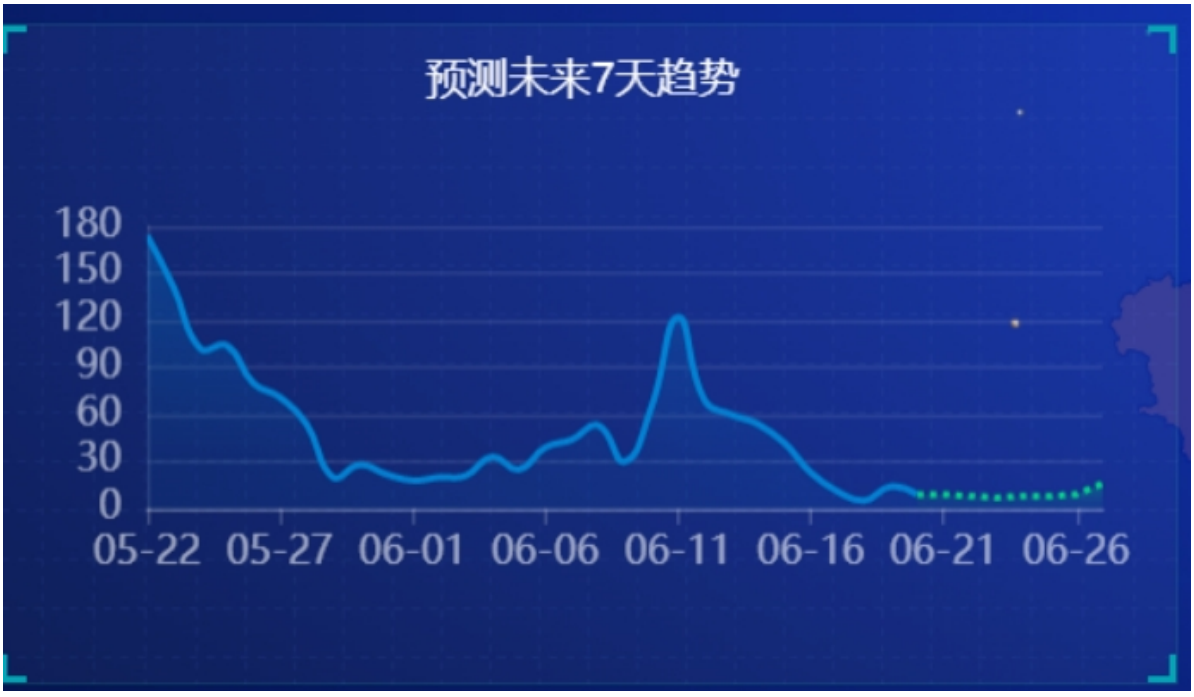
全国现有确诊、累计确诊总数

从render_template方法传入title，在html文件中创建该模块，在css文件中设置模块的位置及属性，通过ajax技术，获取全国现有确诊人数和累计确诊人数，然后通过css selector查找id定位到模块，将数字和标题传入text属性。



现有确诊排名前八的国家

在html文件中创建该模块，在css文件中设置模块的位置及属性，通过ajax技术，获取模块需要的数据，然后通过echart技术设置图表的属性、交互、显示效果，将数据插入，横坐标代表省份，纵坐标代表人数。轮询更新。

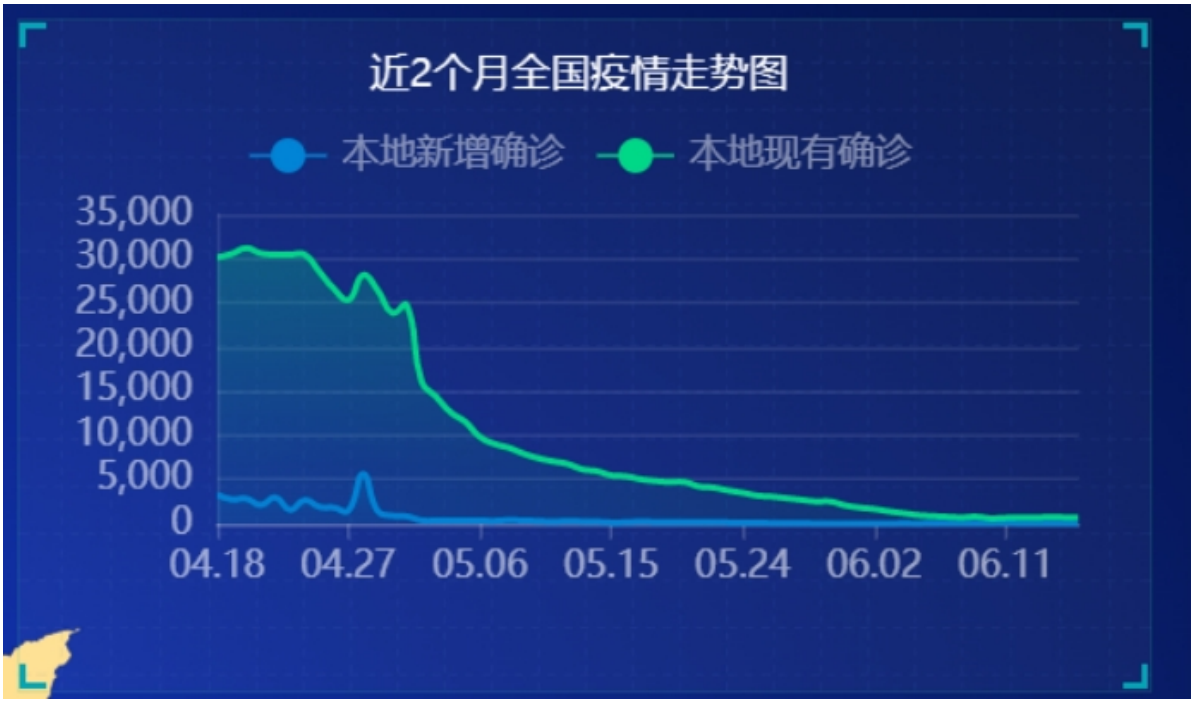


基于前两月新增确诊数据预测未来一周新增确诊

在html文件中创建该模块，在css文件中设置模块的位置及属性，爬取的前两月新增确诊数据通过在时序模型中预训练之后，预测未来七天的数据，通过ajax技术传入模块需要的数据，然后通过echart技术设置图表的属性、交互、显示效果，将数据插入，横坐标代表日期，纵坐标代表人数。轮询更新。

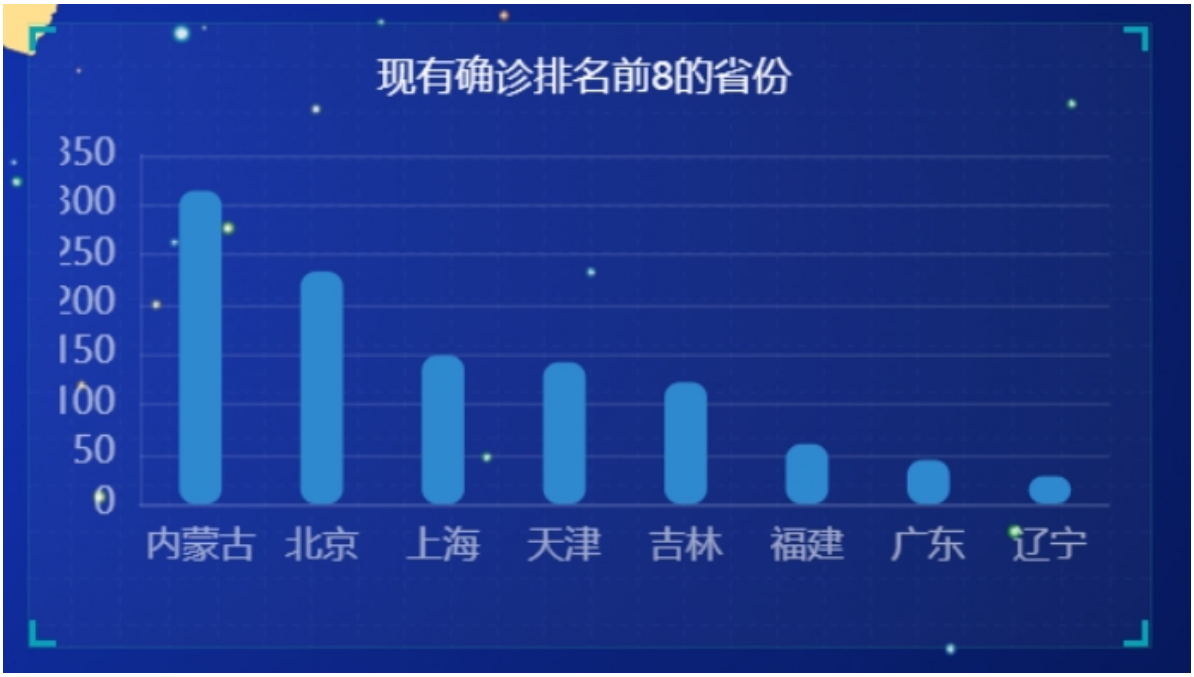
当日新增确诊、累计确诊、本土无症状的各区间累计情况

在html文件中创建该模块，在css文件中设置模块的位置及属性，将数据按照类别和区间分好，通过ajax技术，获取模块需要的数据，然后通过echart技术设置图表的属性、交互、显示效果，将区间和颜色对应起来，将数据插入。轮询更新。



近2个月全国疫情走势图

在html文件中创建该模块，在css文件中设置模块的位置及属性，通过ajax技术，获取模块需要的数据，然后通过echart技术设置图表的属性、交互、显示效果，蓝色的线是本地新增确诊人数、绿色的线是本地西安确诊人数，横坐标代表日期，纵坐标是人数，将数据插入。轮询更新。



现有确诊排名前八的省份

在html文件中创建该模块，在css文件中设置模块的位置及属性，通过ajax技术，获取模块需要的数据，然后通过echart技术设置图表的属性、交互、显示效果，横坐标代表省份，纵坐标是人数，将数据插入。轮询更新。

直辖市现有确诊情况

在html文件中创建该模块，在css文件中设置模块的位置及属性，通过ajax技术，获取模块需要的数据，然后通过echart技术设置图表的属性、交互、显示效果，越往外的直辖市现有确诊人数越多，将数据插入。轮询更新。

中国疫情地图

在html文件中创建该模块，在css文件中设置模块的位置及属性，导入地图需要的外部js文件china.js，将地图和数据对应起来。通过ajax技术，获取模块需要的数据，然后通过echart技术设置图表的属性、交互、显示效果，将数据插入。轮询更新。

4.2 *系统部署*

本作业需要安装flask库、 flask_apscheduler库、warnings库、json库、pandas库、requests库、pymongo库、datetime库、statsmodels.tsa.api库

运行app.py文件，文件会运行app.run()方法，flask服务会将网站部署在默认的网站端口上。

访问网址：http://127.0.0.1:5000/

整体效果

5. *实训问题及总结*

问题

1. MongoDB数据库无法访问，连接总是超时

起先以为是换了网络IP没有访问权限，但是一直设置为0.0.0.0，后来查找了很多帖子，再将网络的DNS设为8.8.8.8之后解决问题

2. 数据库中的数据更新了，不重启flask服务则不会在浏览器中更新数据

通过翻看ajax的文档，发现了通过定时器实现轮询可以实现在数据库中更新数据自动更新到网页上，设置每一个小时请求一次

3. 想要实现爬虫自动爬取数据，但是之前的爬虫写在很多文件中，每次爬取都需要很繁琐的步骤

我将所有的爬虫集合成一个爬虫类，然后通过查flask的文档找到了flask apschedule，可以实现定时运行某个任务，设置每天十点定时爬取数据。

4. 时序预测模型预测的数据一直为负数

爬取到的数据中，缺少一天的数据，导致数据产生错位。我将数据单独查取后补齐，然后更换时序预测模型得到了比较好的结果。

总结

感觉这次的大作业，让我对所学到的技术的应用有了更深次的理解。我明白了真正的项目不是那么简单的对一个程序或者算法的应用，而是不同技术、不同思想甚至是不同学科的复杂融合，需求也不是简单的做个计算画个图就能解决的。对我使用计算机技术解决问题有了极大的锻炼和提高。这次的实验，我对时间的分配也比较不合理，在实验报告上分配的时间比较少，导致实验报告写的有些粗糙。做完了这次的大作业，感觉一身轻，我觉得最重要的收获是对一个比较完整的项目的理解，从需求到框架到方法再到最后的实现，受益良多。