中国石油大学（华东）
CHINA UNIVERSITY OF PETROLEUM

《数据分析（Python）》课程报告

学生姓名：章震豪

学　　号：2014020228

专业班级：计算2003

学　　院：计算机科学与技术学院

课程认知 10%	代码规范 5%	工作量 60%	项目立意 5%	创新性 20%	总分	阅卷人

2021年 6月 15日

2011-2022年青岛地区历史天气数据的获取与分析

1　数据分析认知

数据分析指用适当的统计、分析方法对收集来的大量数据进行分析，将它们加以汇总和理解并消化，以求最大化地开发数据的功能，发挥数据的作用。数据分析是为了提取有用信息和形成结论而对数据加以详细研究和概括总结的过程[1]。

数据分析的目的是把隐藏在一大批看来杂乱无章的数据中的信息集中和提炼出来，从而找出所研究对象的内在规律。在实际应用中，数据分析可帮助人们做出判断，以便采取适当行动。数据分析是有组织有目的地收集数据、分析数据，使之成为信息的过程。这一过程是质量管理体系的支持过程。在产品的整个寿命周期，包括从市场调研到售后服务和最终处置的各个过程都需要适当运用数据分析过程，以提升有效性。例如设计人员在开始一个新的设计以前，要通过广泛的设计调查，分析所得数据以判定设计方向，因此数据分析在工业设计中具有极其重要的地位[2]。

数据也称为观测值，是实验、测量、观察、调查等的结果。数据分析中所处理的数据分为定性数据和定量数据。只能归入某一类而不能用数值进行测度的数据称为定性数据。定性数据中表现为类别，但不区分顺序的，是定类数据，如性别、品牌等；定性数据中表现为类别，但区分顺序的，是定序数据，如学历、商品的质量等级等。

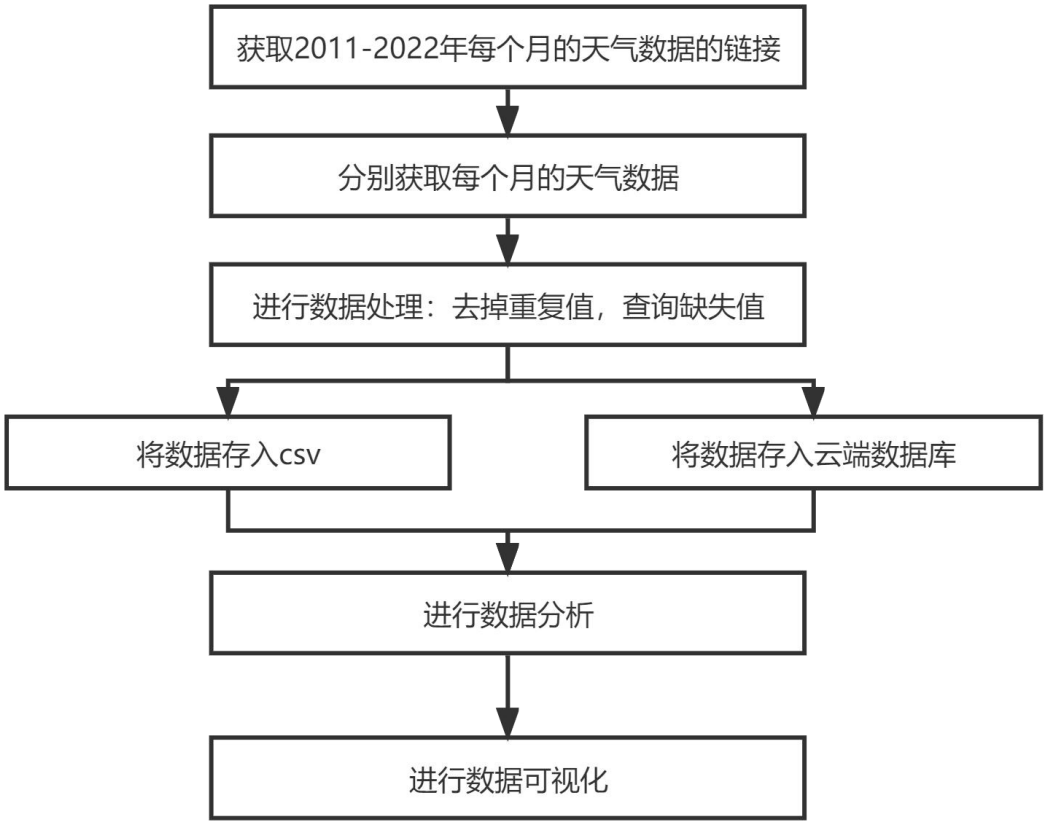
数据分析也可以与数据可视化相结合，通过图形更加直观的展现出数据处理的结果，使得数据展现的更为直观；数据分析可以与数据的预测相结合，根据历史数据来估计后面一段时间的数据。

2　项目概述

本项目使用bs4，urllib，requests，正则表达式，使用了两种匹配方式来完成青岛2011-2022年的天气数据获取过程，将数据存入csv中；使用numpy和pandas库进行数据处理与分析，使用pymysql连接云端数据库，将获取到的数据完成预处理后永久保存到数据库中；使用sklearn对6月上旬的温度数据进行了回归，预测了6月下旬的最高温度与最低温度。使用matplotlib，pylab，pyecharts[3]对数据进行可视化，主要绘制了折线图，曲线图（先将折线图中的数据进行多项式拟合，再绘图），饼图，散点图，柱状图，动态gif图。最后，使用wordcloud生成词云。

本项目的意义在于，通过掌握过去12年的青岛天气数据，可以统计青岛具体的天气情况，青岛的平均温度等天气因素，通过数据可视化使这些数据战线的更为直观，同时进行温度预测，有助于居民后一段时间的出行决定。同时也提升了我使用Python进行数据获取，处理，分析的技术。传统的搜索引擎固然方便，但是在当前大数据的背景下，利用python进行网络爬虫技术的发展是大势所趋[4],爬虫可以大幅度提升数据的获取速度。可视化是借助图形化的方法,清晰有效的将数据展示出来。在大数据快速发展的今天,使用正确有效的工具来实现数据可视化变得至关重要。Python其丰富的第三方库和工具包在文件处理、科学计算及数据可视化领域越来越凸显其地位及价值[5]。

3 项目流程图



4 项目具体内容

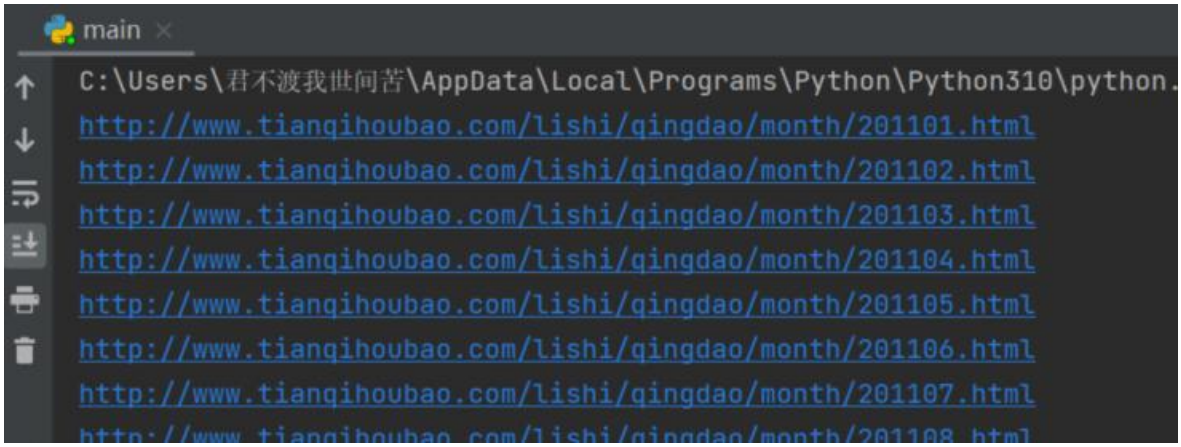
4.1 数据获取

通过爬虫获取数据，分为两步，使用了两种不同的方式进行匹配

第一步：从主页上获取2011-2022年每个月天气的url，使用headers来解决反爬问题[6]，使用**正则表达式**来匹配需要的信息

```
    def get_text1(url):  # 从首页上获得每个月的天气链接
        a_list = np.array([])  # 此处使用numpy
        url_list = np.array([])  # 此处使用numpy
        headers = {
            "User-Agent": 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.124 Safari/537.36 Edg/102.0.1245.41'
        }
        req = request.Request(url=url, headers=headers, method='POST')
        req = request.urlopen(req)
        req = req.read().decode('gbk')
        div_list = re.findall(r'<DIV class="box pcity">(.*?)</div>', req, re.S | re.I)  # 包含新功能-正则表达式
        for i in div_list:
            a_list1 = re.findall(r'<a.*?</a>', i)  # 包含新功能-正则表达式
            a_list = np.append(a_list, a_list1)  # 此处使用numpy
        for i in a_list:
            url_list1 = re.findall(r'href=\'(.*?)\'', i)  # 包含新功能-正则表达式
            if 'qingdao' not in url_list1[0]:
                break
            url_list = np.append(url_list, url_list1[0])  # 此处使用numpy
        url_list = url_list[:-12]
        return url_list
```

运行结果截图：



第二步，从每个月份的url中爬取天气信息，使用**bs4内置的函数**进行匹配[7]（以下为函数的部分代码）

```
    def get_text2(url):  # 获取每个月的天气数据
        global date, weather_m, wind_m, temperature_m, weather_e, wind_e, temperatur
e_e
        headers = {
            "User-
Agent": 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/100.0.4896.127 Safari/537.36 Edg/100.0.1185.44'
        }
        req = requests.get(url, headers=headers,timeout=60)
        req.encoding = req.apparent_encoding
        soup = BeautifulSoup(req.text, 'lxml')
        weather1 = soup.find_all('tr')
        weather1 = weather1[2:]
        weather = np.array([])  # 此处使用numpy
        for i in weather1:
            i1 = i.text
            i1 = i1.replace('\n', ',').replace('\xa0', '').replace('\r','').replace(
' ','')
            weather = np.append(weather, i1[1:])  # 此处使用numpy
```

4.2 数据预处理与储存

第一步：使用pandas和numpy将爬取到的天气数据储存到dataFrame中：

```
    #此处使用pandas
    data = pd.DataFrame()
    data['日期'] = date
    data['白天天气状况'] = weather_m
    data['白天风力方向'] = wind_m
    data['最高温度'] = temperature_m
    data['夜晚天气状况'] = weather_e
    data['夜晚风力方向'] = wind_e
    data['最低温度'] = temperature_e

    #数据存入dataFrame
    for data in weather:
            data1 = data.split(',')
            data1 = data1[1:-1]
            # 此处使用numpy
            date = np.append(date, data1[1])
            weather_m = np.append(weather_m, data1[6])
            wind_m = np.append(wind_m, data1[-2])
            temperature_m = np.append(temperature_m, data1[9])
            weather_e = np.append(weather_e, data1[7][1:])
            wind_e = np.append(wind_e, data1[-1][1:])
```

```
            temperature_e = np.append(temperature_e, data1[11])
```

第二步：进行数据预处理，删除**重复数据与有缺失项**的数据

```
#此处使用pandas
#数据预处理
print(data[data.duplicated()==True])#打印重复值
data.drop_duplicates(inplace=True)#删除重复记录
data=data.dropna()#删除缺失值
data.reset_index()#重置索引
```

运行结果截图：

```
Empty DataFrame
Columns: [Unnamed: 0, 日期, 白天天气状况, 白天风力方向, 白天最高温度, 夜晚天气状况, 夜晚风力方向, 夜晚最高温度]
Index: []
```

第三步：保存数据到**csv文件**中与**数据库**中[8]

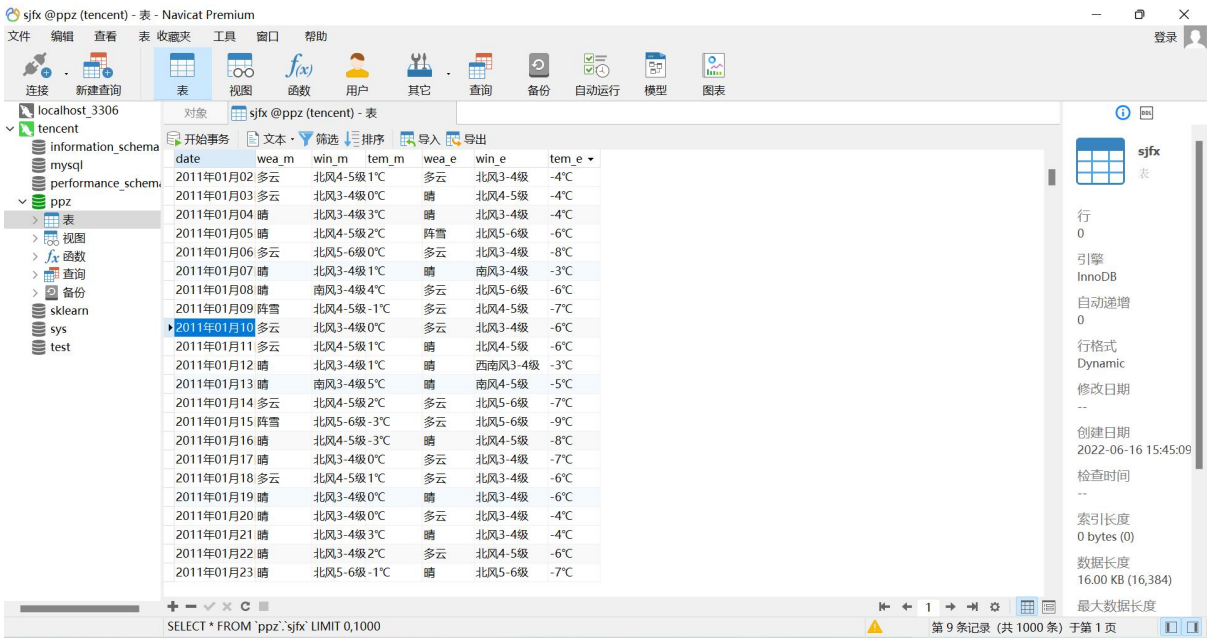
```
#保存到csv
with open('data.csv','w+',encoding='utf-8',newline='') as f:
    data.to_csv(f,encoding='utf-8')

#保存到数据库（mysql8）
conn = pymysql.connect( #包含新功能-数据库技术
    host = [redacted]
    user = [redacted]
    passwd = [redacted]
[redacted]
    [redacted]
[redacted]
)

cursor = conn.cursor()
for index, row in data.iterrows():#此处使用pandas
    sql = 'INSERT INTO sjfx VALUES '
    tp=tuple(row[1:])
    sql+=str(tp)
    cursor.execute(sql)
conn.commit()
cursor.close()
conn.close()
```

运行结果截图：

CSV：

		日期	白天天气状况	白天风力方向	最高温度	夜晚天气状况	夜晚风力方向	最低温度
2	0	2011年1月2日	多云	北风4-5级	1℃	多云	北风3-4级	-4℃
3	1	2011年1月3日	多云	北风3-4级	0℃	晴	北风4-5级	-4℃
4	2	2011年1月4日	晴	北风3-4级	3℃	晴	北风3-4级	-4℃
5	3	2011年1月5日	晴	北风4-5级	2℃	阵雪	北风5-6级	-6℃
6	4	2011年1月6日	多云	北风5-6级	0℃	多云	北风3-4级	-8℃
7	5	2011年1月7日	晴	北风3-4级	1℃	晴	南风3-4级	-3℃
8	6	2011年1月8日	晴	南风3-4级	4℃	多云	北风5-6级	-6℃
9	7	2011年1月9日	阵雪	北风4-5级	-1℃	多云	北风4-5级	-7℃
10	8	2011年1月10日	多云	北风3-4级	0℃	多云	北风3-4级	-6℃
11	9	2011年1月11日	多云	北风4-5级	1℃	晴	北风4-5级	-6℃
12	10	2011年1月12日	晴	北风3-4级	1℃	晴	西南风3-4级	-3℃
13	11	2011年1月13日	晴	南风3-4级	5℃	晴	南风4-5级	-5℃
14	12	2011年1月14日	多云	北风4-5级	2℃	多云	北风5-6级	-7℃
15	13	2011年1月15日	阵雪	北风5-6级	-3℃	多云	北风5-6级	-9℃
16	14	2011年1月16日	晴	北风4-5级	-3℃	晴	北风4-5级	-8℃
17	15	2011年1月17日	晴	北风3-4级	0℃	多云	北风3-4级	-7℃
18	16	2011年1月18日	多云	北风4-5级	1℃	多云	北风3-4级	-6℃

数据库：



4.3 数据分析

第一步：使用pandas查看简要消息

```
1.    print(data.info())#此处使用pandas
```

运行结果截图：

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3662 entries, 0 to 3661
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  3662 non-null   int64
 1   日期          3662 non-null   object
 2   白天天气状况      3662 non-null   object
 3   白天风力方向      3662 non-null   object
 4   最高温度        3662 non-null   object
 5   夜晚天气状况      3662 non-null   object
 6   夜晚风力方向      3662 non-null   object
 7   最低温度        3662 non-null   object
dtypes: int64(1), object(7)
memory usage: 229.0+ KB
```

第二步：使用词典进行数据统计

1）统计2011-2022年整体的数据

```
#2011-2022总体数据处理
for index, row in data.iterrows():#此处使用pandas
    tp=tuple(row[2:])
    dict_weather[tp[0]]=dict_weather.get(tp[0],0)+1
    dict_weather[tp[3]]=dict_weather.get(tp[3],0)+1
    dict_weather_m[tp[0]]=dict_weather_m.get(tp[0],0)+1
    dict_weather_e[tp[3]]=dict_weather_e.get(tp[3],0)+1
    dict_tem[tp[2]]= dict_tem.get(tp[2],0)+1
    dict_tem[tp[5]]= dict_tem.get(tp[5],0)+1
    dict_tem_m[tp[2]]= dict_tem.get(tp[2],0)+1
    dict_tem_e[tp[2]]= dict_tem.get(tp[5],0)+1
    dict_win[tp[1]]=dict_win.get(tp[1],0)+1
    dict_win[tp[4]]=dict_win.get(tp[4],0)+1
    dict_win_m[tp[1]]=dict_win_m.get(tp[1],0)+1
    dict_win_e[tp[4]]=dict_win_e.get(tp[4],0)+1
```

2）统计某一年的数据：以2011年为例（部分代码）

```
for i,row in data.iterrows():
    row=row[1:]
    if '2011' in row['日期']:
        date2=np.append(date2,row[0])
        weather_m2=np.append(weather_m2,row[1])
        wind_m2=np.append(wind_m2,row[2])
        temperature_m2=np.append(temperature_m2,row[3])
        weather_e2=np.append(weather_e2,row[4])
        wind_e2=np.append(wind_e2,row[5])
        temperature_e2=np.append(temperature_e2,row[6])
```

第三步：进行数据分析

1）可以统计2011-2022年整体的数据并排名：

例：2011-2022年各种天气出现的次数排名

```
dict_weather1=sorted(dict_weather.items(),key=lambda x:x[1],reverse=True)
pd1_t=np.array([])#此处使用numpy
pd1_n=np.array([])
#此处使用pandas
pd1=pd.DataFrame()
for i in dict_weather1:
    pd1_t=np.append(pd1_t,i[0])
    pd1_n=np.append(pd1_n,i[1])
pd1['天气']=pd1_t
pd1['次数']=pd1_n
print(pd1.info)
```

运行结果截图：

```
2011-2022年各种天气出现次数排行
<bound method DataFrame.info of        天气      次数
0        晴  2801.0
1       多云  2703.0
2        阴   498.0
3       小雨   374.0
4       阵雨   337.0
5        雾   228.0
6     雷阵雨   131.0
7       中雨    84.0
8     雨夹雪    49.0
9    小到中雨    37.0
10     阵雪    18.0
```

2）可以统计2011-2022年某个时间段的数据并排名

例：2011-2022年夜晚时间风力数据出现次数排行（部分代码）

```
dict_win_m1=sorted(dict_win_m.items(),key=lambda x:x[1],reverse=True)
print(pd2.info)
```

运行结果截图：

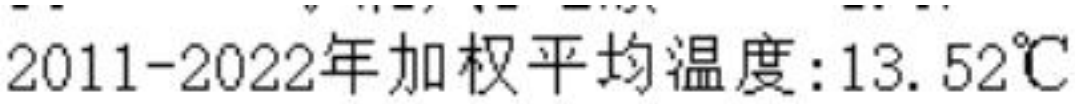
```
2011-2022年夜晚时间风力数据出现次数排行
<bound method DataFrame.info of              风力      次数
0         南风3-4级  1068.0
1         北风3-4级   835.0
2        东南风3-4级   327.0
3         北风4-5级   263.0
4         南风4-5级   151.0
5         北风5-6级   135.0
6        西南风3-4级   133.0
7        东北风3-4级   122.0
8        西北风3-4级    94.0
9         南风3～4级    67.0
10       东南风4-5级    45.0
11       西北风4-5级    42.0
12       东北风4-5级    39.0
13       西北风5-6级    37.0
```

3）可以计算平均值

例：计算2011-2022年加权平均温度

```
1.    dict_tem=sorted(dict_tem.items(),key=lambda x:x[1],reverse=True)
2.    ans=0
3.    count=0
4.    for i in dict_tem:
5.        ans+=float(i[0][:-1])*i[1]
6.        count+=int(i[1])
7.    print("{:.2f}{}".format(ans/count,'℃'),)
```

运行结果截图：



4）可以对于某一年，统计数据并排名，计算某个时间段的数据，计算平均数：

例：2011年各种风力情况出现次数排行（部分代码）

```
1.    dict_weather3=sorted(dict_weather2.items(),key=lambda x:x[1],reverse=True)
2.    print(pd3.info)
```

运行结果截图：

```
2011年各种风力情况出现次数排行
<bound method DataFrame.info of        天气     次数
0        晴  2801.0
1       多云  2703.0
2        阴   498.0
3       小雨   374.0
4       阵雨   337.0
5        雾   228.0
6      雷阵雨   131.0
7       中雨    84.0
8      雨夹雪    49.0
9     小到中雨    37.0
10      阵雪    18.0
11    中到大雨    15.0
12      小雪    12.0
13      大雨    11.0
```

例：2011年夜晚时间天气数据出现次数排行（部分代码）

```
1.    dict_win_e3=sorted(dict_win_e2.items(),key=lambda x:x[1],reverse=True)
2.    print(pd4.info)
```

运行结果截图：

```
2011年夜晚时间天气数据出现次数排行
<bound method DataFrame.info of          风力     次数
0     南风3-4级  111.0
1     北风3-4级   78.0
2     北风4-5级   39.0
3    东南风3-4级   28.0
4     南风4-5级   27.0
5     北风5-6级   21.0
6    西南风3-4级    8.0
7    东南风4-5级    8.0
8    东北风3-4级    7.0
9     东风3-4级    5.0
10   东北风4-5级    3.0
```

例：2011年加权平均温度（部分代码）

```
1.    dict_tem3=sorted(dict_tem2.items(),key=lambda x:x[1],reverse=True)
2.    print("{:.2f}{}".format(ans1/count1,'℃'),)
```

运行结果截图：

```
2011年加权平均温度:13.15℃
```

5）可以**预测**后续天气

例：预测2022年六月下旬温度情况

```
    t_h=np.array([])#包含新功能-机器学习Lasso回归
    t_l=np.array([])
    for i in range(len(data)-15,len(data)):
        str3=data.iloc[i]
        t_h=np.append(t_h,float(str3['最高温度'][:-1]))
        t_l=np.append(t_l,float(str3['最低温度'][:-1]))
    t_x=[]
    t_p=[]
    for i in range(1,16):
        l1=[]
        l1.append(i)
        t_x.append(l1)
    for i in range(16,31):
        l2=[]
        l2.append(i)
        t_p.append(l2)
    reg1 = linear_model.Lasso(alpha = 0.1)
    reg1.fit(t_x,t_h)
    reg2 = linear_model.Lasso(alpha = 0.1)
    reg2.fit(t_x,t_l)
```

运行结果截图：

```
最高温度预测结果
[22.22380952 22.14345238 22.06309524 21.9827381  21.90238095 21.82202381
 21.74166667 21.66130952 21.58095238 21.50059524 21.4202381  21.33988095
 21.25952381 21.17916667 21.09880952]
最低温度预测结果
[19.17619048 19.25654762 19.33690476 19.4172619  19.49761905 19.57797619
 19.65833333 19.73869048 19.81904762 19.89940476 19.9797619  20.06011905
 20.14047619 20.22083333 20.30119048]
```

4.4 数据可视化

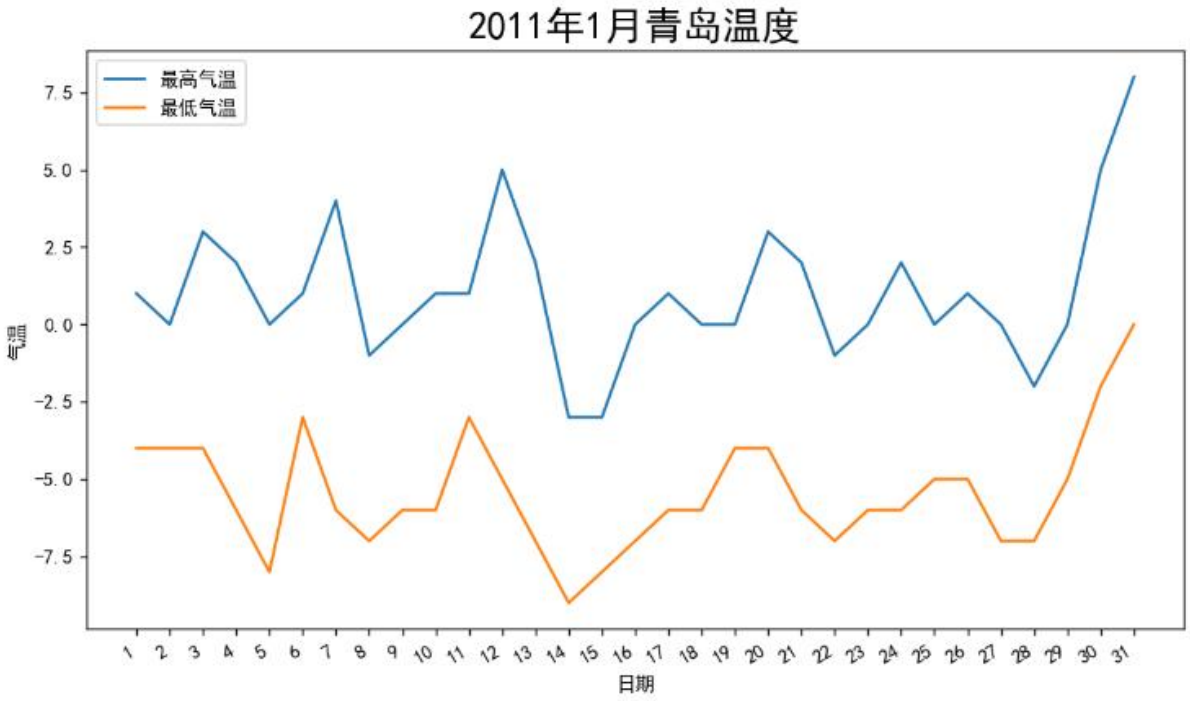
1）绘制了折线图

例：以2011年1月每日最高气温与最低气温为例

```
    plt.rcParams['font.sans-serif'] = ['SimHei']#中文字体
    plt.rcParams['axes.unicode_minus'] =False#显示负号
    fig = plt.figure(dpi=128,figsize=(10,6))
    L1,=plt.plot(range(1,32),t_high,label='最低气温')
    L2,=plt.plot(range(1,32),t_low,label='最高气温')
    plt.legend(handles=[L1,L2],labels=['最高气温','最低气温'], loc='best')
```

```
    plt.title('2011年1月青岛温度',fontsize=20)
    plt.xlabel('日期',fontsize=10)
    fig.autofmt_xdate()
    plt.ylabel('气温',fontsize=10)
    plt.tick_params(axis='both',which='both',labelsize=10)
    plt.xticks(range(1,32))
    plt.show()
```

运行结果截图：



2）绘制了曲线图

例：以2012年1月每日最高气温与最低气温为例（部分代码）

先对散点进行拟合：

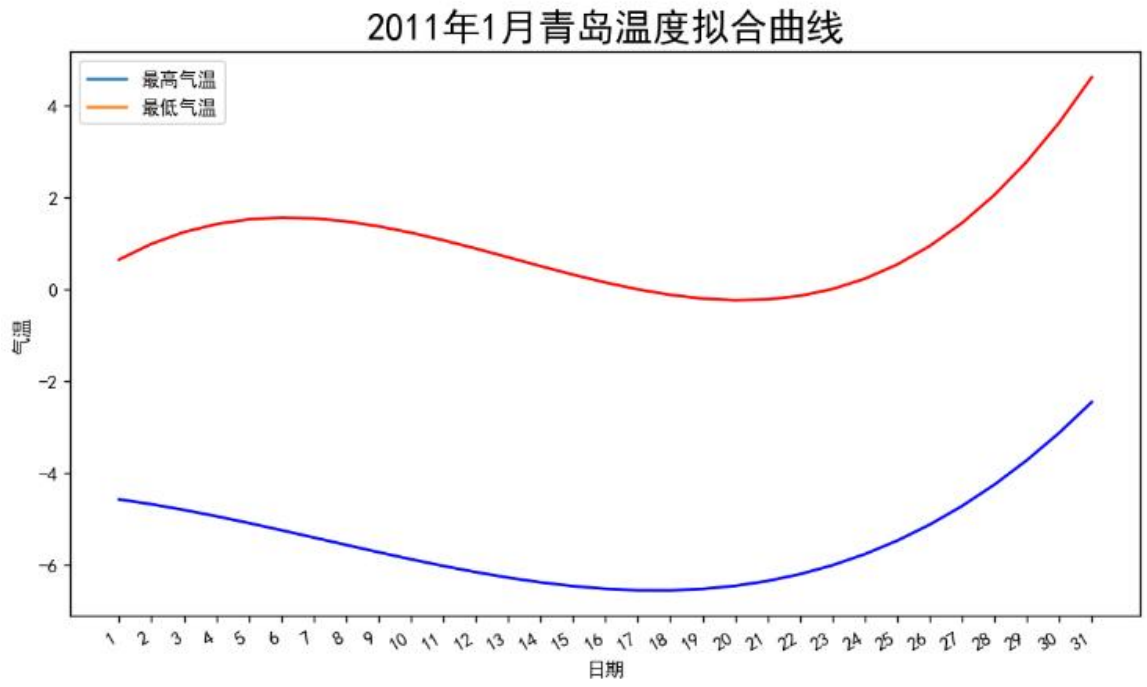
```
    t1 = np.polyfit(x, y, 3)
    p1 = np.poly1d(t1)
    t2 = np.polyfit(x, z, 3)
    p2 = np.poly1d(t2)
    y_pred1 = p1(x)
    y_pred2 = p2(x)
```

绘制曲线图：

```
    plot1 = pylab.plot(x, y_pred1, 'r', label='fit values')
```

```
    plot2 = pylab.plot(x, y_pred2, 'b', label='fit values')
    pylab.legend(handles=[L1,L2],labels=['最高气温','最低气温'], loc='best')
    pylab.show()
```

运行结果截图：

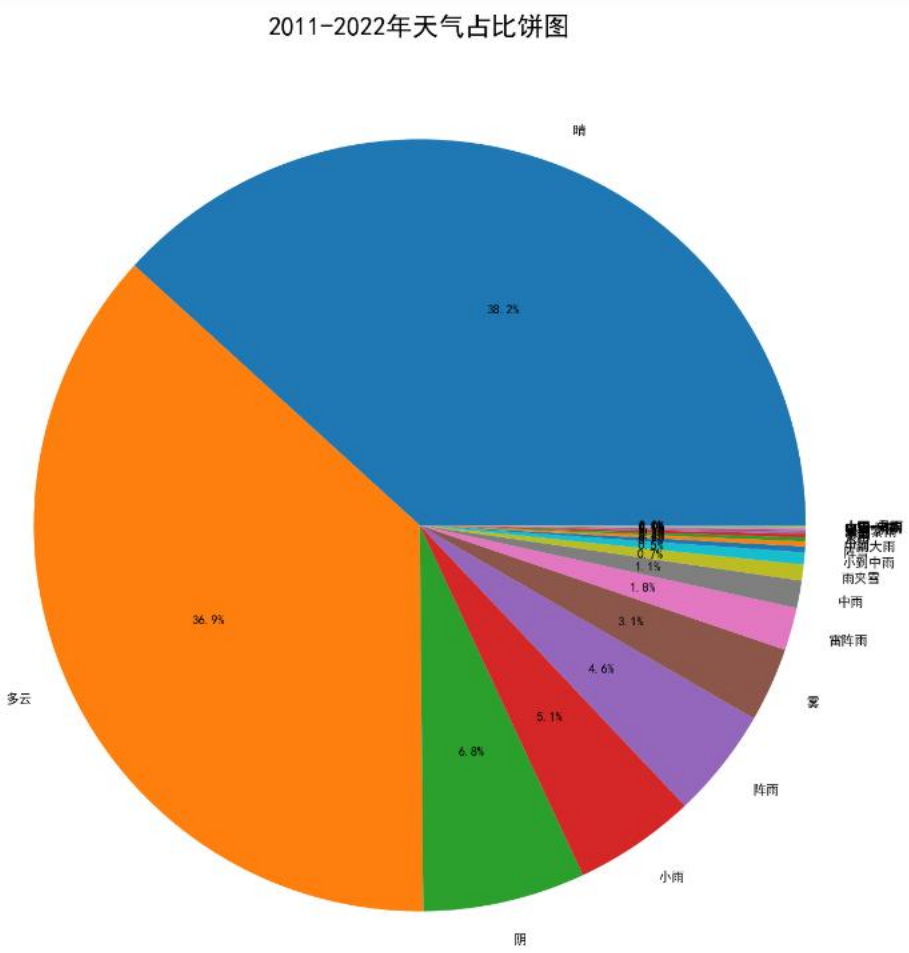


3）绘制了饼图

例：以2011-2022年天气占比情况为例（部分代码）

```
    plt.pie(x_b,labels=y_b, autopct="%.1f%%")
    plt.show()
```

运行结果截图：

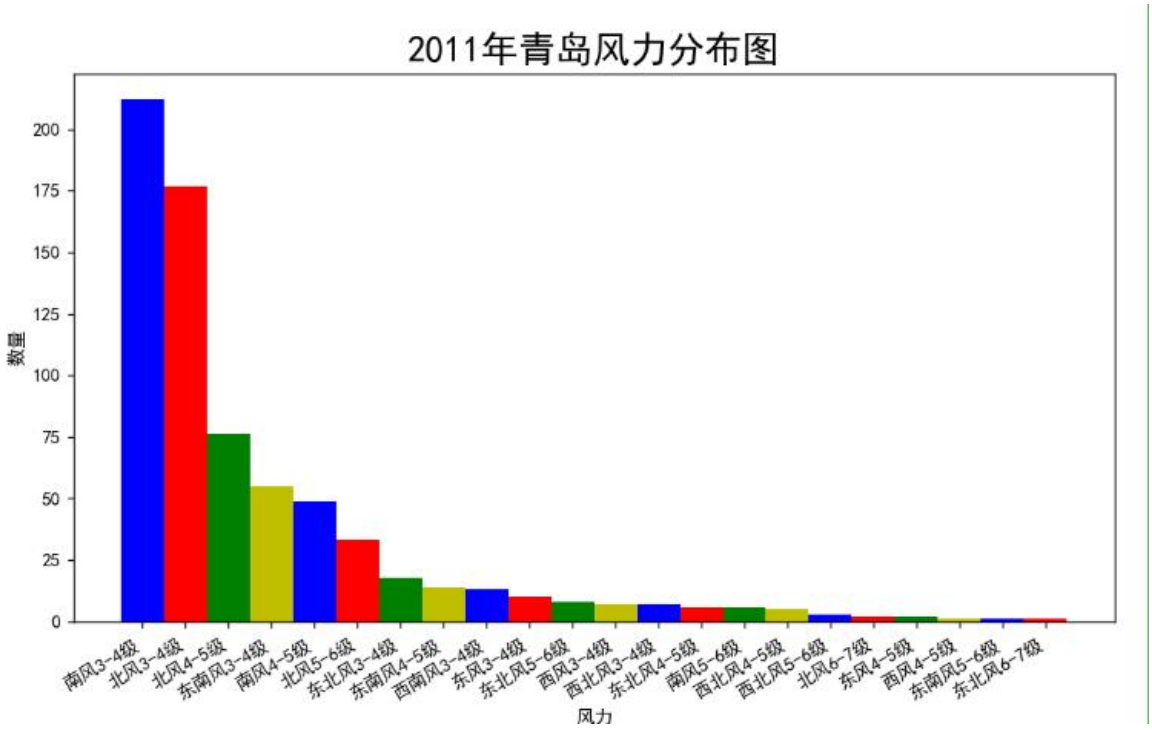


4）使用matplotlib绘制了柱状图

例：2011年青岛风力分布情况（部分代码）

```
    plt.bar(x_w, y_w, width=1, color=["b", "r", "g", "y"] * 2)
    plt.show()
```

运行结果截图：



例：以2011年的风力为例

5）使用pyecharts绘制了柱状图

例：以2011年的风力为例

```
CurrentConfig.NOTEBOOK_TYPE = NotebookType.JUPYTER_LAB#包含新功能-pyecharts
bar=(
    Bar()
    .add_xaxis(x_w)
    .add_yaxis('数量',y_)
    .reversal_axis()
    .set_series_opts(label_opts=opts.LabelOpts(position="best"))
    .set_global_opts(title_opts=opts.TitleOpts(title="2011年青岛风力分布图"))
)
bar.render('pic.html')
```

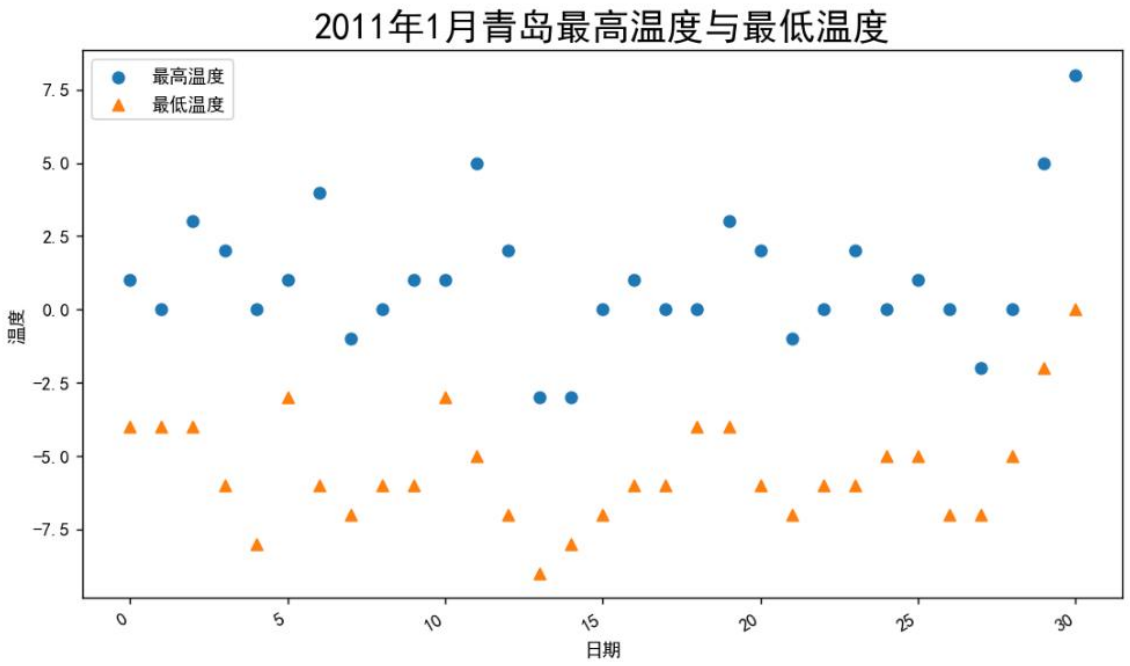
运行结果截图：

6）绘制了散点图

例：以2011年1月最低和最该的温度情况为例（部分代码）

```
1.    plt.scatter(x_, t_high, marker='o', label="最高温度")
2.    plt.scatter(x_, t_low, marker='^', label="最低温度")
3.    plt.legend(loc='best')
4.    plt.show()
```

运行结果截图：



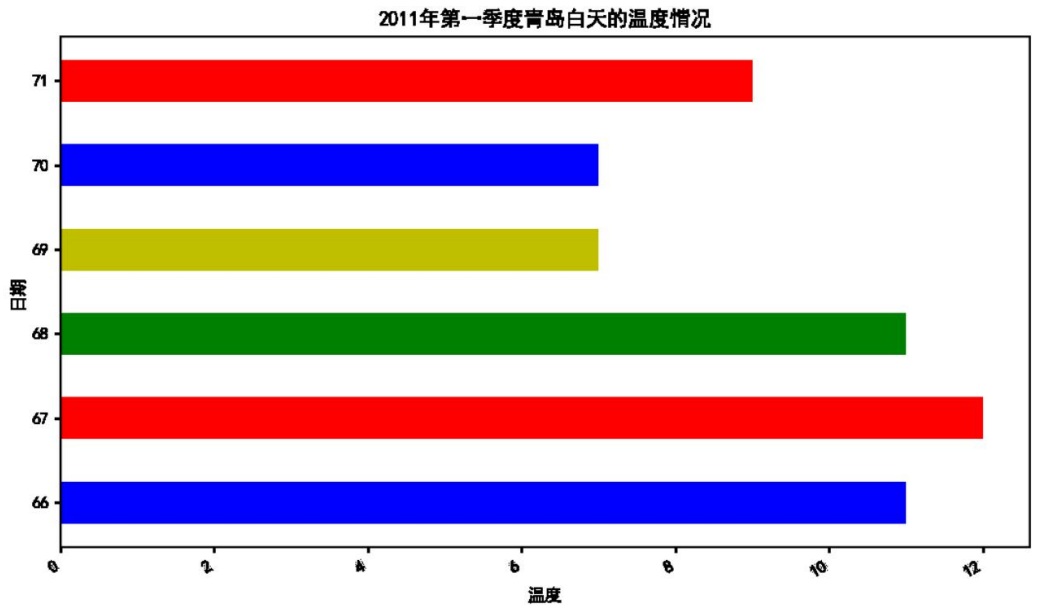
7）绘制了gif动态图

例：以2011年第一季度青岛白天的温度情况为例

```
1.    images = []
```

```
    buf = io.BytesIO()
    fig4 = plt.figure(dpi=128,figsize=(10,6))
    for i in range(0,120,6):
        plt.cla()
        data1_=[]
        for j in range(i,i+6):
            str4=data2011.iloc[j][3][:-1]
            data1_.append(float(str4))
        data2_=[i,i+1,i+2,i+3,i+4,i+5]
        plt.barh(data2_, data1_, height=0.5, color=["b", "r", "g", "y"] * 2, alpha=1)
        plt.xlabel("温度")
        plt.ylabel("日期",fontsize=10)
        plt.title('2011年第一季度青岛白天的温度情况')
        fig4.autofmt_xdate()
        pickle.dump(plt.gca(), buf)
        plt.savefig('tem.png')
        images.append(imageio.imread('tem.png'))
imageio.mimsave("pic.gif", images, fps=5)
```

运行结果截图：



Gif链接：

http://ai.computer.upc.edu.cn:7001/6921a2e7db8f32a9cb26bf6cf1c33fa8/view/pic.gif

8）生成了词云

例：以2011-2022年的天气数据为例

```
text=""#包含新功能-词云
for i in dict_weather.keys():
    dict_weather[i]*=20
    text+=int(dict_weather[i])*str(i)

ls=jieba.lcut(text)
txt=" ".join(ls)
w=wordcloud.WordCloud(
    font_path='STXingkai.ttf',
)
w.generate(txt)
w.to_file("wordcloud.png")
```

运行结果截图：

5 项目亮点

5.1 使用正则表达式进行匹配

5.2 使用sklearn对数据进行处理，预测下一周的天气情况

5.3 使用pyecharts进行可视化

5.4 使用数据库技术

5.5 将折线图进行多项式拟合，拟合成了光滑的曲线

5.6 生成了词云

6 课程建议

建议可以在课堂时间设置一些随堂测试或者课堂弹题，让我们更好的融入课堂中去，也可以让我们自查我们这节课的学习效果。

7 总结

在转专业到计算机学院之前，在原专业的通识课上了解过一些Python的基础知识，一直对这门神奇的语言充满着兴趣与向往。刚开始学这门课程的时候，由于计算机基础较为薄弱，加上数据分析作为一门计算机的专业课程，难度远大于之前学习的通识课，我学习的很吃力。但是通过日积月累的学习，认真完成每个单元的作业，遇到问题主动提问，我也逐渐能够跟上进度。

这次课程报告，无疑是对之前的课程学习的凝练与提升。自己设计一个爬虫来进行数据获取，使用numpy和pandas对数据进行处理与分析，使用matplotlib,pylab,pyecharts来进行可视化，自学了解了一些机器学习的内容并进行了简单的数据预测。

完成这次课程报告的过程并不容易：爬虫一开始没有爬取到想要的内容，储存文件的路径设置错误，numpy和pandas使用遇到了种种问题，调整matplotlib,pylab,pyecharts,wordcloud的各种参数让我伤透了脑筋。

做这次课程报告这些花费了我很长一段时间。但是我认为这是值得的，做这次课程报告的过程让我真正意义上，第一次不在任何外界帮助下，能够用Python这门语言去做一些东西，去实现一些以前看起来遥不可及的功能。

在以后的学习生活中，我也会继续对于Python这门语言的学习，努力在Python和数据分析这个方面有更大的收获！

以上就是我本次课程报告的全部内容。

参考文献

[1] 刘春辉.大数据分析与情报分析关系辨析[J].科技视界,2022(12):8-10.DOI:10.19694/j.cnki.issn2095-2457.2022.12.02.

[2] 赵惠芹.大数据分析技术在企业档案管理中的应用[J].办公室业务,2022(09):128-130.

[3] 张玉叶.基于PyEcharts的数据可视化[J].电脑知识与技术,2022,18(02):24-27.DOI:10.14004/j.cnki.ckt.2022.0006.

[4] 毕森,杨昱昺.基于python的网络爬虫技术研究[J].数字通信世界,2019(12):107-108.

[5] 李俊华.基于Python的数据可视化[J].新型工业化,2021,11(03):69-70.DOI:10.19335/j.cnki.2095-6649.2021.3.031.

[6] Zhang Qi,and Xiong Wei."Research on Online Game Traffic Classification Based on Machine Learning".*Proceedings of 2011 International Conference on Engineering and Information Management (ICEIM 2011)*.Ed.. , 2011, 58-63.

[7] Liu Jingfa,Li Xin,Zhang Qiansheng,Zhong Guo. A novel focused crawler combining Web space evolution and domain ontology[J]. Knowledge-Based Systems,2022,243.

[8] Zhuoxi Zhang,Ming Yuan,Hanwei Qian. Research on MySQL Database Recove ry and Forensics Based on Binlog[C]//.Proceedings of the 11th International Confere nce on Computer Engineering and Networks(CENet2021)Part I.,2021:750-759.DOI: 10.26914/c.cnkihy.2021.045076.