大连二手房数据分析

|  |  |
| --- | --- |
| 系别 | 软件工程 |
| 姓名 | 刘政 |
| 班级 | 软测1901 |
| 学号 | 201992228 |

2022年 1月 5 日**《大连二手房数据分析》**

**一、项目概述**

近年来，随着社会的不断发展，人们对于海量数据的挖掘和运用越来越重视，互联网是面向全社会公众进行信息交流的平台，现已成为了收集信息的最佳渠道。同时，伴随着大数据技术的创新与应用，互联网为人们进行大数据统计分析进一步提供了便利。

大数据信息的统计分析可以为决策者提供充实的依据。例如，近些年国内房地产行业发展势头迅猛，二手房的需求成为了一种新热门。本次项目分析的是链家网 “二手房交易价格”的业务数据，其数据的相关分析可以为未来二手房业务的发展提供参考和指导。

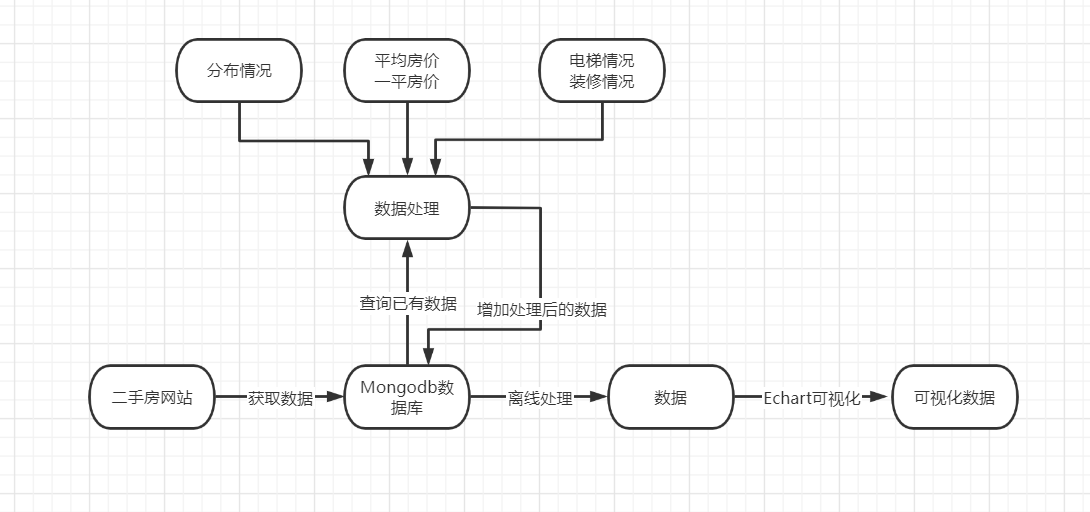
MongoDB数据库的应用场景十分广泛，本系统将利用MongoDB+Python +Echart技术开发二手房交易数据分析系统，用于对二手房交易数据进行分析并展示。

**二、系统架构**

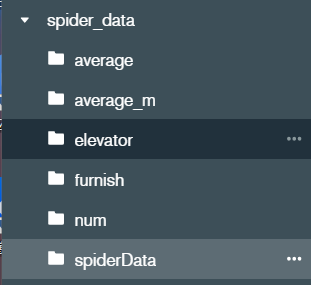
本系统所需的数据是来源于二手房交易网站，该网站中包含了众多房源的详细信息。在本案例中，我们通过Python编写网络爬虫程序，采集大连市二手房交易数据，将采集的数据存储到MongoDB数据库中。当数据采集完成后，利用函数计算框架读取MongoDB中存储的二手房交易数据，并进行离线分析，最后将分析结果存储到MongoDB数据库中。为了可以更加直观的查看分析结果，我们通过Echart工具获取由MongoDB数据库中存储数据导出的分析结果，实现数据的可视化。

**三、详细设计**

1.项目流程

****

2.数据库设计





**四、项目实现**

1. 数据采集

利用Python爬虫程序，用于采集某房屋交易网站中二手房交易数据信息，并将采集的数据存储到MongoDB数据库中。

import requests # 数据请求模块 第三方模块 pip install requests  
import parsel # 数据解析模块 第三方模块 pip install parsel  
import re # 正则表达式模块  
import csv # csv数据保存  
import pymongo  
  
client = pymongo.MongoClient('localhost', 27017)  
spider = client['spider\_data']  
spiderData = spider['spiderData']  
  
for page in range(1, 101):  
 # 1. 发送请求, 是对于房源列表页发送请求  
 print(f'正在爬取第{page}页的数据内容')  
 print('正在爬取第%s页的数据内容' % page)  
 url = f'https://dl.lianjia.com/ershoufang/pg{page}/'  
 # 请求头: 把python代码进行伪装 成浏览器 对于服务器发送请求 模拟浏览器发送请求  
 # User-Agent: 浏览器的基本标识  
 headers = {  
 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.54 Safari/537.36'  
 }  
 # 使用requests这个模块里面get方法 对于 url地址发送请求 并且携带上headers请求头  
 response = requests.get(url=url, headers=headers)  
 # <Response [200]> 响应体对象 200 状态码 表示请求成功  
 # 2. 获取数据 获取响应体的文本数据  
 # print(response.text) # 获取response.text html字符串数据内容  
 # 3. 解析数据, 提取我们想要的内容, 房源详情页url  
 # 你想要对于字符串数据内容 直接进行解析提取, 只能用re正则表达式  
 selector = parsel.Selector(response.text)  
 # css选择器, 解析方法 就是 根据标签属性内容 提取相关的数据  
 href = selector.css('li.clear a.noresultRecommend::attr(href)').getall()  
 for index in href:  
 # 4. 发送请求, 对于房源详情页url地址发送请求  
 response\_1 = requests.get(url=index, headers=headers)  
 # 5. 获取数据  
 # 6. 解析数据, 提取房源基本信息 售价 标题 单价 面积 户型....  
 selector\_1 = parsel.Selector(response\_1.text)  
 # # 标题  
 # title = selector\_1.css('div.title .main::text').get()  
 # # 售价  
 # price = selector\_1.css('.price .total::text').get() + '万元'  
 # print(title, price)  
 area = selector\_1.css('.areaName .info a:nth-child(1)::text').get() # 区域  
 community\_name = selector\_1.css('.communityName .info::text').get() # 小区  
 room = selector\_1.css('.room .mainInfo::text').get() # 户型  
 room\_type = selector\_1.css('.type .mainInfo::text').get() # 朝向  
 height = selector\_1.css('.room .subInfo::text').get() # 楼层  
 height = re.findall('共(\d+)层', height)[0]  
 sub\_info = selector\_1.css('.type .subInfo::text').get().split('/')[-1] # 装修  
 Elevator = selector\_1.css('.content li:nth-child(12)::text').get() + '电梯' # 电梯  
 if Elevator == '暂无数据电梯':  
 Elevator = '无电梯'  
 house\_area = selector\_1.css('.content li:nth-child(3)::text').get().replace('㎡', '') # 面积  
 price = selector\_1.css('.price .total::text').get() # 价格(万元)  
 date = selector\_1.css('.area .subInfo::text').get().replace('年建', '') # 年份  
 dit = {  
 '市区': area,  
 '小区': community\_name,  
 '户型': room,  
 '朝向': room\_type,  
 '楼层': height,  
 '装修情况': sub\_info,  
 '电梯': Elevator,  
 '面积(㎡)': house\_area,  
 '价格(万元)': price,  
 '年份': date,  
 '详情页': index,  
 }  
 #csv\_writer.writerow(dit)  
  
 spiderData.insert\_one(dit)  
 print(area, community\_name, room, room\_type, height, sub\_info, Elevator, house\_area, price, date, index,  
 sep='|')

2．数据分析

通过爬虫程序采集的数据内容，对采集的数据进行统计分析，将分析的数据存入到MongoDB中。

import requests # 数据请求模块 第三方模块 pip install requests  
import parsel # 数据解析模块 第三方模块 pip install parsel  
import re # 正则表达式模块  
import csv # csv数据保存  
import pymongo  
  
client = pymongo.MongoClient('localhost', 27017)  
spider = client['spider\_data']  
collection = spider['spiderData']  
f = spider['num']  
f2 = spider['average']  
f3 = spider['average\_m']  
f4 = spider['elevator']  
f5 = spider['furnish']  
  
shiqu = ['开发区', '沙河口', '高新园区', '金州', '甘井子', '普兰店', '旅顺口', '中山', '瓦房店', '西岗']  
dianti = ['有电梯', '无电梯']  
zhuangxiu = ['精装', '简装', '毛坯', '其他']  
  
for i in shiqu:  
 counts = collection.find({'市区': i})  
 print(i)  
 num = 0  
 cost = 0  
 cost\_m = 0  
 for count in counts:  
 num = num + 1  
 tx = float(count['价格(万元)'])  
 ty = float(count['面积(㎡)'])  
 tz = tx / ty  
 # print(tx)  
 # print(ty)  
 cost = cost + tx  
 cost\_m = cost\_m + tz  
#二手房分布数量  
 dit = {  
 '市区': i,  
 '数量': num  
 }  
 f.insert\_one(dit)  
 print(num)  
  
#二手房平均价格  
 avg = cost / num  
 avg2 = round(avg , 2)  
 dita = {  
 '市区': i,  
 '平均价格': avg2  
 }  
 f2.insert\_one(dita)  
 print(avg2)  
  
#二手房一平平均价格  
 avg\_m = cost\_m / num  
 avg\_m2 = round(avg\_m , 2)  
 dita\_m = {  
 '市区': i,  
 '平均价格': avg\_m2  
 }  
 f3.insert\_one(dita\_m)  
 print(avg\_m2)  
  
#二手房电梯情况  
 ditd = {  
 '市区': i,  
 }  
 for j in dianti:  
 k = 0  
 countd = collection.find({'市区': i, '电梯': j})  
 for count in countd:  
 k = k + 1  
 ditd[j] = k  
 f4.insert\_one(ditd)  
  
#二手房装修情况  
 ditz = {  
 '市区': i,  
 }  
 for j in zhuangxiu:  
 k = 0  
 countz = collection.find({'市区': i, '装修情况': j})  
 for count in countz:  
 k = k + 1  
 ditz[j] = k  
 f5.insert\_one(ditz)

3. 数据展示

采用Echart实现动态数据可视化。

function excelToECharts(obj) {

    excelToData(obj);

}

// 读取Excel转换为json

function excelToData(obj) {

    let inputId = obj.id;

    let files = obj.files;

    if (files.length) {

        let reader = new FileReader();

        let file = files[0];

        // 看下文件格式是否为xls或xlsx

        let fullName = file.name;

        let filename = fullName.substring(0, fullName.lastIndexOf("."));    // 文件名

        let fixName = fullName.substring(fullName.lastIndexOf("."), fullName.length);   // 后缀名

        // 处理excel

        if (fixName == ".xls" || fixName == ".xlsx") {

            reader.onload = function (ev) {

                let data = ev.target.result;

                // 获取excel

                let excel = XLSX.read(data, {type: 'binary'});

                // 获取第一个标签页名

                let sheetName = excel.SheetNames[0];

                // 获取第一个标签页的内容

                let sheet = excel.Sheets[sheetName];

                // 转换为JSON

                let sheetJson = XLSX.utils.sheet\_to\_json(sheet);

                // 转换成json后,转换线图格式

                if (inputId == 'inputLine') {

                    // 线图

                    getLineChartFromJson(sheetJson, filename);

                } else if (inputId == 'inputPie') {

                    // 饼图

                    getPieChartFromJson(sheetJson, filename);

                }

            }

        } else {

            alert("只支持excel")

        }

        reader.readAsBinaryString(file);

    }

}

// 获取列名，返回列名的数组

function getColName(sheetJson) {

    // 遍历json的第一行从而得到key

    let keys = [];

    for (let key in sheetJson[0]) {

        keys.push(key)

    }

    return keys;

}

// 数据封装及显示

function getLineChartFromJson(sheetJson, filename) {

    // 如果有结果，处理结果

    if (sheetJson.length) {

        // 获取所有列名

        let keys = getColName(sheetJson);

        // 处理一下作为x轴的列名和数据

        let xZhou = {};

        xZhou.name = keys.splice(0,1);

        let xDatas = [];

        for (let i in sheetJson) {

            xDatas.push(sheetJson[i][xZhou.name]);

        }

        xZhou.data = xDatas;

        // 主体数据

        let datas = [];

        for (let i in keys) {

            let one = {};       // 一组

            one.name = keys[i];

            one.type = 'line';  // 图表类型

            one.smooth = true;  // 平滑的线

            let point = [];     // 记录这一组的所有点

            for (let idx in sheetJson) {

                // 把这组的点push到数组中

                point.push(sheetJson[idx][one.name]);

            }

            one.data = point;

            // 把这组数据添加到主体数据中

            datas.push(one)

        }

        // 调用展现的方法

        dataToLineChart(filename, keys, xZhou, datas);

    }

}

// 数据可视化

function dataToLineChart(title, keys, xZhou, datas) {

    document.getElementById('ECharts\_main').removeAttribute('\_echarts\_instance\_');

    // 基于准备好的dom，初始化echarts实例

    var myChart = echarts.init(document.getElementById('ECharts\_main'));

    // 指定图表的配置项和数据

    var option = {

        title: {

            text: title,

            x: 'center',

            y: 'bottom'

        },

        tooltip: {

            trigger: 'axis'

        },

        legend: {

            data: keys,

            orient: 'vertical',

            x: 'right',

            y: 'center'

        },

        xAxis: xZhou,

        yAxis: {},

        series: datas,

        toolbox: {

            show: true,

            left: 'right',

            feature: {

                dataView: {},

                magicType: {

                    type: ['line', 'bar', 'stack', 'tiled']

                },

                saveAsImage: {}

            }

        }

    };

    // 显示图表。

    myChart.setOption(option);

}

// 饼图数据封装及显示

function getPieChartFromJson(sheetJson, filename) {

    let keys = getColName(sheetJson);

    let items = [];

    for (let i in sheetJson) {

        items.push(sheetJson[i][keys[0]]);

    }

    // 获取数据

    let sheetData = [];

    for (let i in sheetJson) {

        sheetData.push({'name': sheetJson[i][keys[0]], 'value': sheetJson[i][keys[1]]});

    }

    // 构造series要的数据

    let datas = {};

    datas.name = keys[0];

    datas.type = 'pie';

    datas.radius = '50%';

    datas.center = ['45%', '50%'];

    datas.data = sheetData;

    datas.itemStyle = {

        emphasis: {

            shadowBlur: 10,

            shadowOffsetX: 0,

            shadowColor: 'rgba(0, 0, 0, 0.5)'

        }

    };

    dataToPieChart(filename, items, datas);

}

// 饼图数据展现

function dataToPieChart(title, items, datas) {

    document.getElementById('ECharts\_main').innerHTML = "";

    document.getElementById('ECharts\_main').removeAttribute('\_echarts\_instance\_');

    // 基于准备好的dom，初始化echarts实例

    let myChart = echarts.init(document.getElementById('ECharts\_main'));

    // 指定图表的配置项和数据

    let option = {

        title: {

            text: title,

            x: 'center',

            y: 'bottom'

        },

        tooltip: {

            trigger: 'item',

            formatter: "{a} <br/>{b} : {c} ({d}%)"

        },

        legend: {

            type: 'scroll',

            orient: 'vertical',

            right: 10,

            top: 30,

            bottom: 20,

            data: items,

        },

        series: datas,

        toolbox: {

            show: true,

            left: 'right',

            feature: {

                dataView: {},

                magicType: {},

                saveAsImage: {}

            }

        }

    };

    // 使用刚指定的配置项和数据显示图表。

    myChart.setOption(option);

}

**五、项目运行测试**

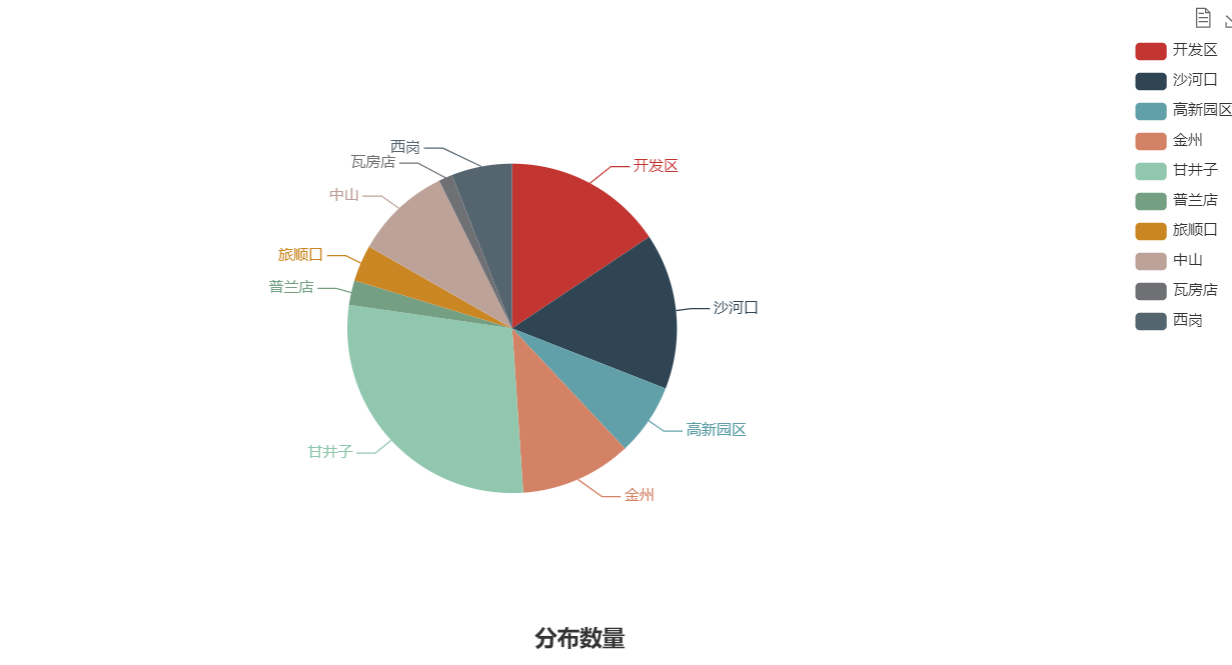
1.数据采集

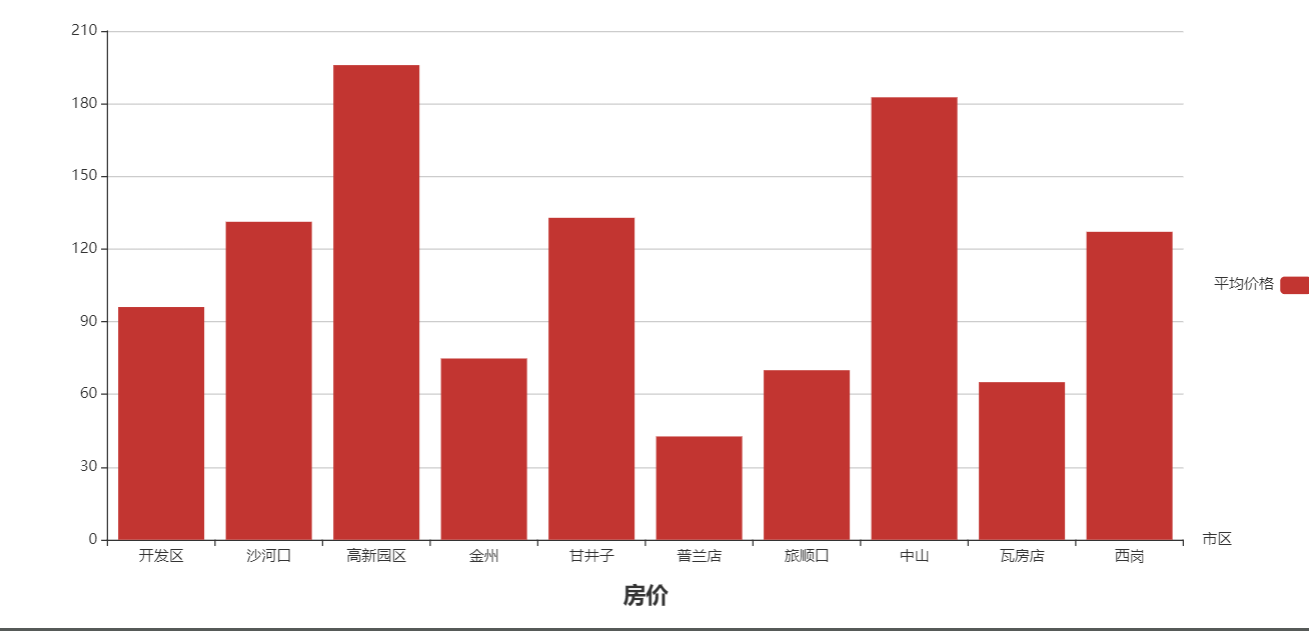
******

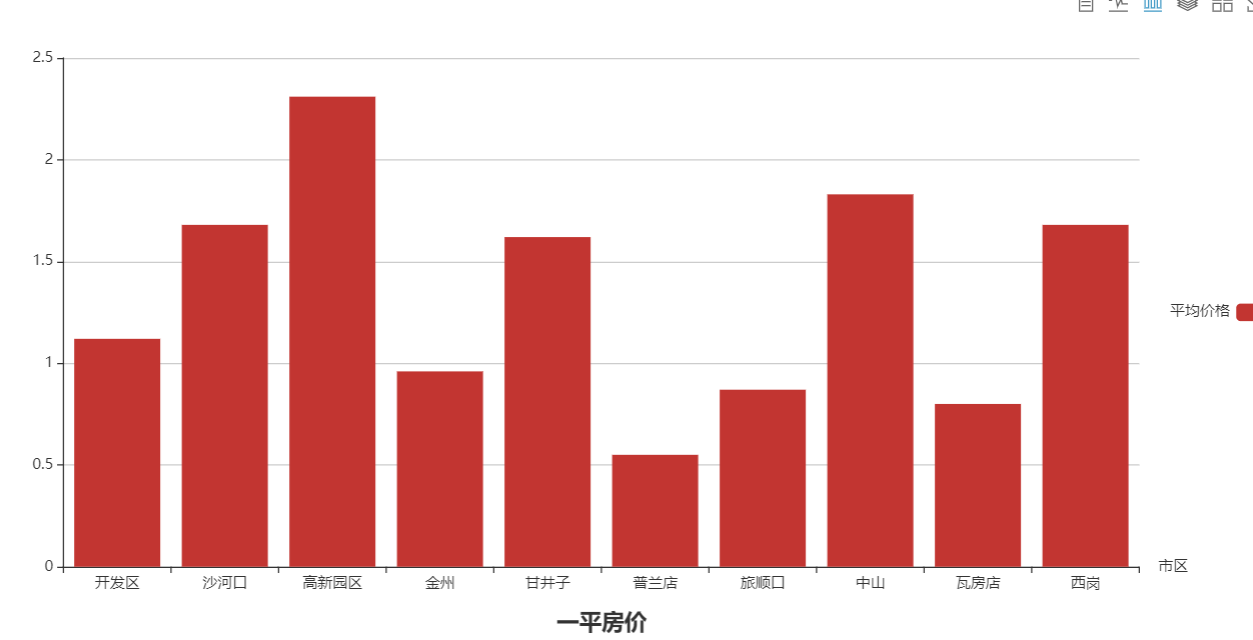
2.数据分析

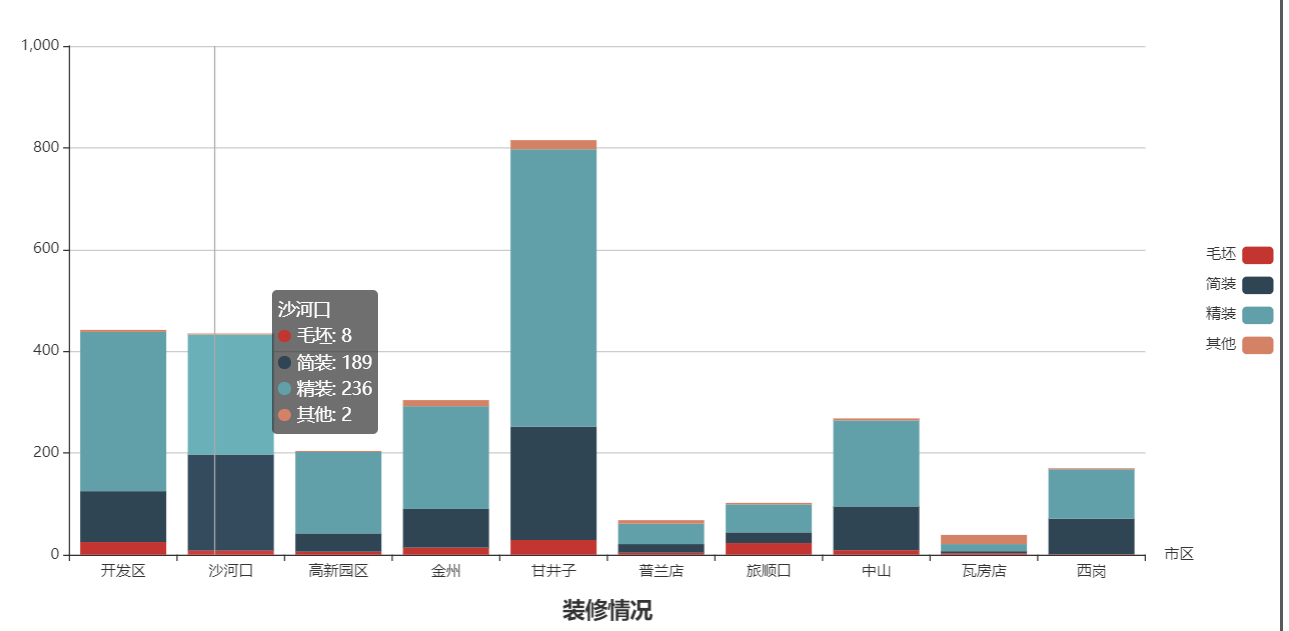
****

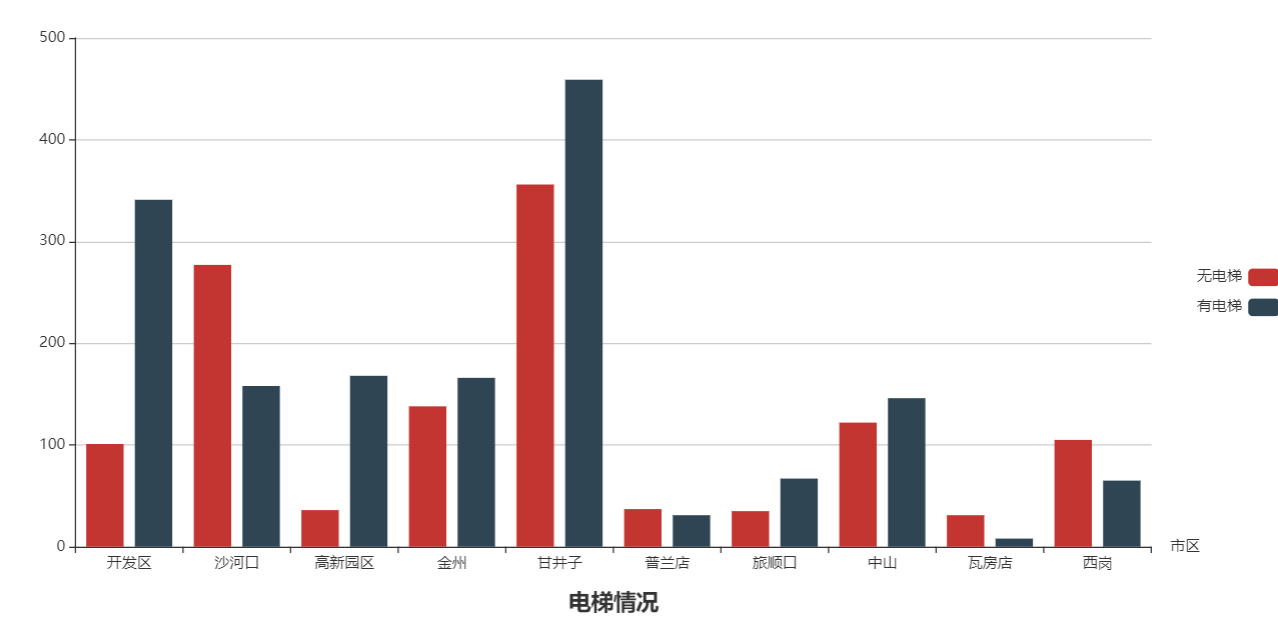
3.数据展示

****

****

****

****



**六、项目分析与总结**

此次所需的数据是来源于链家大连二手房，通过爬虫采集二手房的相关数据并存储到MongoDB数据库中，同时采用Python对采集的数据进行计算统计分析，将分析的数据存入到MongoDB中。为了可以更加直观的查看分析结果，将MongoDB数据库中存储的分析结果生成多个Echart图，实现数据的可视化。

项目还有很多不足之处，功能过于单一，没有进行扩展，只能对已经处理好的大连二手房的几类数据可视化，功能模块没有划分清楚；没有做成一个全部可视化的系统，能够选择城市，选择要查询的数据类型；数据处理方面，Python的效率较低，应该采用更优秀的计算框架；对于可视化，也只是利用了现成的Echart工具，只能对xlsx文件进行处理。

通过这门课程的学习以及大作业的实现，我学到了很多关于开源软件、项目管理以及科技大牛的相关知识，收获颇多，有很多思想值得我今后学习借鉴和应用。

**七、小组组成**

|  |  |  |
| --- | --- | --- |
| **姓名** | **角色** | **分工** |
| **刘政** | **队员** | **数据采集、分析及可视化展示** |