

**毕业设计**

|  |  |
| --- | --- |
| **中文题目** | **基于电商网络的数据采集与分析** |
| **英文题目** | **Data acquisition and analysis based on e-commerce network** |

|  |  |
| --- | --- |
| 院系： | 光电与通信工程学院 |
| 年级专业： | 2016级通信工程 |
| 姓名： | 卓越 |
| 学号： | 1633042446 |
| 指导教师： | 张璘 |
| 职称： | 讲师 |

2020年 6 月 7日

**毕业设计（论文）诚信声明书**

本人郑重声明：在毕业设计（论文）工作中严格遵守学校有关规定，恪守学术规范；我所提交的毕业设计（论文）是本人在 指导教师的指导下独立研究、撰写的成果，设计（论文）中所引用他人的文字、研究成果，均已在设计（论文）中加以说明；在本人的毕业设计（论文）中未剽窃、抄袭他人的学术观点、思想和成果，未篡改实验数据。

本设计（论文）和资料若有不实之处，本人愿承担一切相关责任。

学生签名：

年 月 日

基于电商网络的数据采集与分析

**摘要：**随着人类社会以及互联网的迅速发展，越来越多的人投身到网购大军中，但如何有效地获取商品的数据，进行分析归纳总结，从而减少学习成本和金钱成本，已经成为了热门话题。本文提出了基于Python语言的淘宝数据爬取系统，从指定网站上抓取商品信息，经处理后以特定格式存储到数据库，再对数据进行分析，实现对影响商品数据的各种因素的可视化，以帮助人们高效地了解该做哪些产品，该如何去做。

本文基于Python语言在PyCharm开发平台设计了淘宝数据爬虫系统，对获得的数据分析总结并利用pandas/numpy/matplot库进行可视化。最后实现了对淘宝网站数据的实时爬取，通过柱状图与散点图总结数据特点，并构建了两者的线性回归方式，使内容更加清晰明了。

**关键词：**Python；网络爬虫；数据库；商品数据；数据分析；数据可视化；

Data acquisition and analysis based on e-commerce network

Abstract：With the rapid development of human society and the Internet, more and more people are engaged in online shopping. However, it has become a hot topic how to effectively obtain the data of commodities, analyze and summarize them, so as to reduce the cost of learning and money. This paper proposes a Python based taobao data crawling system, which captures commodity information from designated websites, stores it in a specific format to the database after processing, and then analyzes the data to realize the visualization of various factors affecting commodity data, so as to help people effectively understand which products to make and how to make them.

Based on Python language, this paper designed taobao data crawler system on PyCharm development platform, analyzed and summarized the obtained data and visualized the pandas,numpy and matplot libraries. Finally, it realizes the real-time crawling of the data on taobao website, summarizes the characteristics of the data through the bar chart and the scatter chart, and constructs the linear regression method of the two to make the content more clear.

KeyWords: Python; Web Crawler; Database; Commodity data; Data analysis; Data Visualization;

目录

[第一章 前言 - 1 -](#_Toc43929858)

[1.1课题背景 - 1 -](#_Toc43929859)

[1.2设计目的 - 2 -](#_Toc43929860)

[1.3设计方案 - 2 -](#_Toc43929861)

[第二章 Python及其环境搭建 - 3 -](#_Toc43929862)

[2.1 Python简介 - 3 -](#_Toc43929863)

[2.2 Windows系统下安装Python 3.8 - 4 -](#_Toc43929864)

[2.3 PyCharm集成开发环境 - 5 -](#_Toc43929865)

[2.4 MongoDB数据库介绍 - 5 -](#_Toc43929866)

[2.5 本章小结 - 5 -](#_Toc43929867)

[第三章 淘宝产品数据爬虫 - 6 -](#_Toc43929868)

[3.1 爬取网站介绍 - 6 -](#_Toc43929869)

[3.2 各模块设计 - 6 -](#_Toc43929870)

[3.2.1 信息模块 - 6 -](#_Toc43929871)

[3.2.2解析模块 - 10 -](#_Toc43929872)

[3.2.3 存储模块 - 12 -](#_Toc43929873)

[3.3 本章小结 - 14 -](#_Toc43929874)

[第四章 数据处理 - 15 -](#_Toc43929875)

[4.1 数据拆分 - 15 -](#_Toc43929876)

[4.2 数据清洗 - 16 -](#_Toc43929877)

[4.2.1 分词 - 16 -](#_Toc43929878)

[4.2.2剔除停用词 - 16 -](#_Toc43929879)

[4.2.3 去重 - 17 -](#_Toc43929880)

[4.3 本章总结 - 18 -](#_Toc43929881)

[第五章 商品信息可视化分析 - 19 -](#_Toc43929882)

[5.1 词云可视化 - 19 -](#_Toc43929883)

[5.2 不同词组对于销量的影响 - 20 -](#_Toc43929884)

[5.3 商品价格与销量的关系 - 21 -](#_Toc43929885)

[5.4产地对商品的影响 - 23 -](#_Toc43929886)

[5.5总体设计 - 24 -](#_Toc43929887)

[5.6 本章总结 - 25 -](#_Toc43929888)

[第六章 总结与展望 - 26 -](#_Toc43929889)

[致谢 - 27 -](#_Toc43929890)

[参考文献: - 28 -](#_Toc43929891)

# 第一章 前言

## 1.1课题背景

信息是维持人类生存交流的根本，是人类在世界上无时无刻都在接触的因素。信息是丰富的，繁杂的，我们平时接触到的信息非常多，但是我们无法知晓我们获取了怎么样的信息，我们又传递了哪些信息，互联网给予了我们解决这个问题的可能[[[[1]](#endnote-1)]]。

现如今4G技术已基本普及在各大地区，应用但不仅限于生活、教育、科技等领域，可以说国内目前在世界上已经走到了互联网生活的前沿[[[2]](#endnote-2)]。不仅如此，在4G之外，5G技术也在蓬勃发展中，可以说在未来人们所接收到的信息会更加迅速、丰富和快捷，人们可以尽情地在互联网上使用各种搜索引擎去查找自己感兴趣的内容；可以在各大视频网站上去观看，浏览他人对其的评价；可以在购物网站上去购买自己喜爱但又出于种种原因无法在线下购买的物品等等，因此人们生活于一个互联网串通彼此，信息大爆炸的时代。

但是信息爆炸会带来一连串的问题，人们经常会在使用互联网的时候遇到这些问题并因此而感到困扰，其中最引人注目的便是：如果我想要批量获取信息，该如何去做[[[3]](#endnote-3)]。互联网会提供给人们各种各样的web页面，其中以文本、图像、音频和视频的形式存储的信息是最常见的，从像互联网这样的大型存储库中检索正确的图像是困难的。为了支持检索正确的信息，人们使用到了搜索引擎，最受欢迎的是谷歌，百度，Bing，Yahoo等。这些搜索引擎使用浏览整个互联网的web爬虫程序从相应的URL收集相关信息并将其存储在数据库中，网络爬虫增强了快速搜索终端用户的范围[[[4]](#endnote-4)]。

基于内容的图像检索系统是需要从大型数据集或互联网检索图像的领域之一。对该系统来说，正确的图像检索是一个重要的挑战。在网络爬虫中，程序可以递归地检索由这些链接在提供的URL下标识的所有web页面[[[5]](#endnote-5)]；网络爬虫程序搜索也是基于一个关键字，关键字可以与作为文件名称或网页中文件的标题或描述中检索到的图像相关联。这些可以在爬虫下载的网页链接中找到。爬虫的设计必须包含一些社会责任，如道德，在爬行网站经常有网站警告机器人不要爬进它们，这样的服务器将在一个名为Robots .txt的文件中实现机器人排除协议（当人们使用部分爬虫框架时可以更改该文件）[[[6]](#endnote-6)]。因此爬虫需要尊重web上采用的爬虫规则标准所设定的协议，允许最小的爬虫伦理性，从而避免网络流量到这样的服务器。

现如今大多数的网络爬虫都会使用Python语言来编写，这是因为相比与其他静态编程语言，如java，c#，C++，对于获取所爬取的网页的文档接口，Python更简单清晰[[[7]](#endnote-7)]；相比其他动态脚本语言，在给予用户用于访问网页文档的API方面上，Python的urllib3包相比于perl,shell等动态脚本语言，有着更为显著的优势，比如提供的API就更加完整。

## 1.2设计目的

通过Python编写爬虫程序从淘宝网站上循环抓取产品的各项数据，并对数据进行存储与提取，然后用图表对其进行展示，直观了解产品价格，销量，产地等详细情况，并对这些数据进行分析与可视化。

## 1.3设计方案

本论文将使用Python语言，在PyCharm开发环境中去编写爬虫程序，通过selenium启动代理浏览器，在搜索文本框中输入搜索关键字，从Web服务器调用Web爬虫模块，接着爬虫模块检查Internet连接，如果网络连接成功，就处理文本搜索引擎的关键字，然后将搜索引擎结果发送给解析器，直到所有链接都被处理。通过requests库来提交网络请求，爬取HTML页面，获得网页源码。通过正则表达式解析源码，即解析页面，然后提取相关数据。最后使用pymongo库连接MongoDB数据库和Python将爬取的数据存入该数据库。通过这几个步骤我们就能实现基本的爬虫功能并将数据保存。

爬取完成产品信息后，我们就能使用numpy、pandas、matplotlib三个库对数据进行填补缺失值，去重，清洗，分类等操作，然后进行数据分析，最后实现数据可视化。

# 第二章 Python及其环境搭建

## 2.1 Python简介

Python是一种可在多平台使用的程序设计语言，是一个从多个层次将解释性、编译性、互动性以及面向对象结合起来的脚本语言。其最开始被设计用作编写自动化脚本(shell)，随着时代的进步及计算机科学的发展，在版本的不断更新中Python的语言新功能也在不断地添加，Python逐渐被更多的程序员，学生，爱好者用来编写独立、大型的项目。

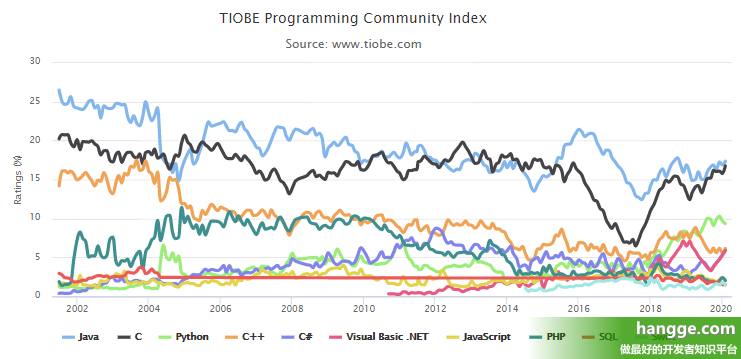


图2-1 2020年2月TIOBE长期走势图

由上图可见，Python虽不如Java及C语言，但是在18年后使用量有一个较大的提升，反映出Python越来越受使用者的喜爱，业内对其的使用率及评价也有一个较大的上升

由Python语言的三种特性，即使代码编写更为方便的简洁性，可以使使用者及后来者阅读更为容易，便于对代码进行重构的易读性，以及可由使用者自行编写扩展库并给予他人自由调用权限的可扩展性，世界上偏向于科学计算的机构纷纷开始使用Python。不仅如此，Python也走进了大多数大学的课堂，许多学生以及教授编写课设作业时相比于C语言，JAVA语言，会更加倾向于使用Python语言。Python能取得如今这样的成就，归功于它有着对编写程序有着较大帮助的扩展标识库。本次设计将会在Python上进行数据处理、运算、作图等操作，而这些操作都可以使用经典的扩展库来协助我们，比如使用numpy,pandas。

Python有一个与其他语言相比非常显著的区别，那就是Python在使用如if,for等函数模块时，会强制用户进行缩进操作。而C语言对于模块界限的定义，则是将语言的语法与字符的排列分割，即C语言是由首字符的位置决定界限，这曾被认为是时代的进步，但也曾引起过争议。不可否认的是，Python与其他语言的这种区别，使得Python的编写更加简洁明了[[[8]](#endnote-8)]。

## 2.2 Windows系统下安装Python 3.8

Python3.8提供了4个安装程序供用户自行下载，用户可以根据自己的需求以及系统配置来选择安装包。安装完成后，可通过Win+R进入运行面板，通过cmd指令打开控制台，输入Python后可查看Python的安装信息。

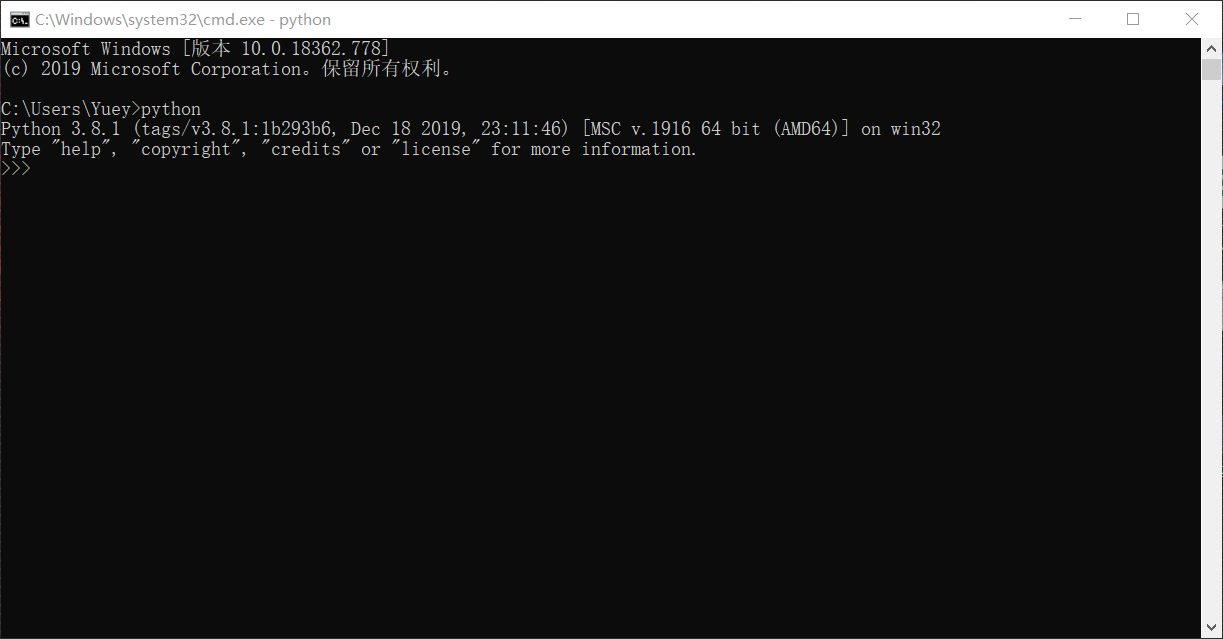


图2-2 命令行执行图

安装完Python后，我们在编写项目的时候常常需要调用功能强大的库，而这些库Python是不会默认安装的，这时候需要我们自己去查找库的文件并下载安装。通常在<https://pypi.org/>上可以查找到所需的大部分Python库，如下图所示：

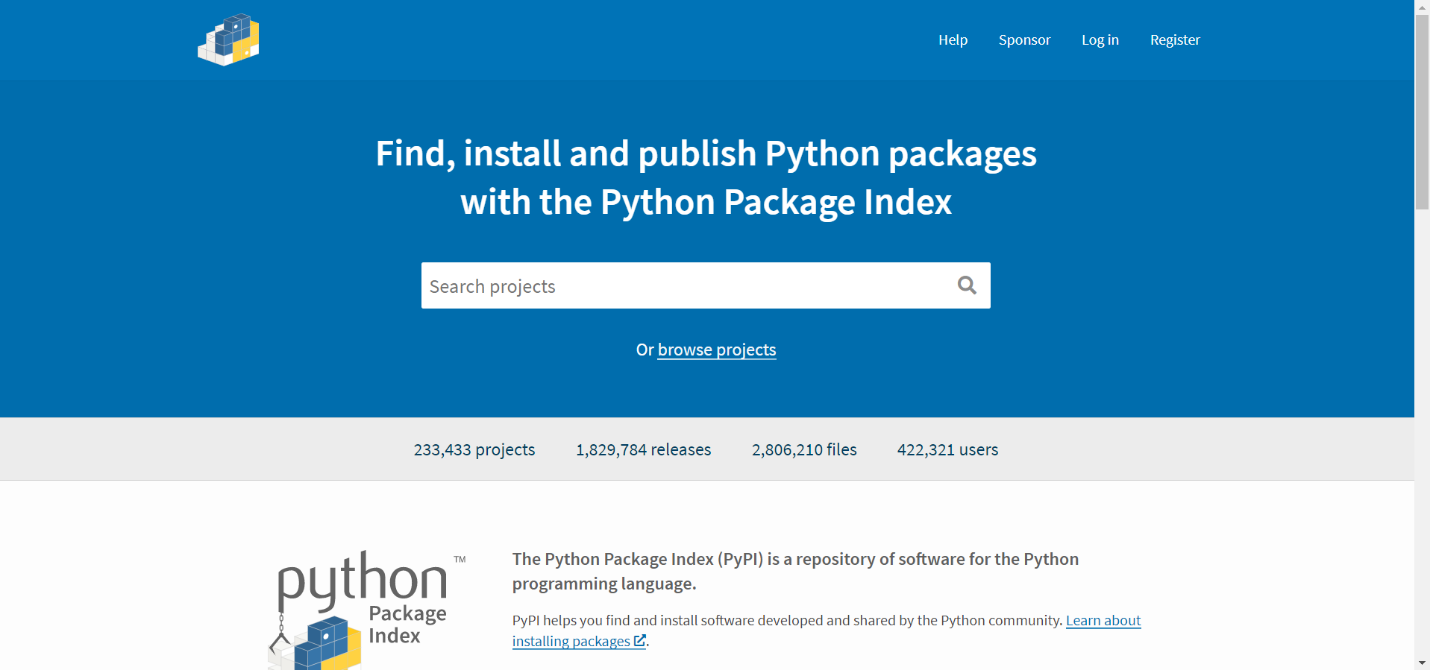


图2-3 PYPI界面

或者可以直接通过命令行安装，如pip install scipy，安装完成后如果想确认自己安装的库可以通过输入pip list查看。

2.3 PyCharm集成开发环境

PyCharm是一种提供了多种高级功能，支持在Django框架下进行开发的用于提供高效Python开发所需的Python IDE。它有着以下特点：（1）使编程更有效率，PyCharm可以帮你节省时间，PyCharm可以帮你处理日常事务，专心于其他更加重要的事，你可以仅仅通过键盘而非使用过多辅助工具，就能够使PyCharm的大部分特性充分发挥；（2）PyCharm对代码检查非常精细，它能给用户带来非常智能的帮助，使用户能够在PyCharm的帮助下快速检查代码、调试错误，对自己的代码有一个非常直观的认知；（3）PyCharm会通过项目导航以及自身内置的各种检查来检测用户代码的编写，当用户的代码编写不尽人意时，PyCharm能够及时地通过检查对用户做出提醒，提高用户的代码质量。

2.4 MongoDB数据库介绍

MongoDB是一个数据库，与市面上使用较多的MySQL不同的是，MongoDB更像是介于关系和非关系数据库之间。MongoDB使用的是分布式文件存储，这使得MongoDB在目前市面上比较流行的非关系数据库中显得独树一帜，也使得它功能非常丰富。MongoDB支持的查询语言种类丰富，也归功于它支持的数据结构相比于其他非关系数据库较为松散，这使得MongoDB非常像关系数据库。MongoDB的数据结构大多数为json和bson格式，语法更多的是面向对象，不仅可以存储复杂的数据类型，也能够对数据进行索引建立。因此关系数据库能实现的功能，大部分MongoDB也能做到。

2.5 本章小结

本章介绍了Python环境的搭建以及库的安装，还介绍了目前在市面上较为好用的Python集成IDE，PyCharm。Python用来搜集数据，而MongoDB则能将搜集来的数据存储到数据库中。

# 第三章 淘宝产品数据爬虫

## 3.1 爬取网站介绍

现如今对于大部分用户来说，提到网络购物便不可避免地会联想到淘宝网，淘宝网也毋庸置疑的是国内最受人们欢迎及使用率最高的商品网站，相比于京东、亚马逊等网站，人们更倾向于在淘宝网上购物。不仅如此，纵观整个亚太地区的网络销售圈，淘宝网也是名副其实的巨头。它拥有着数额庞大且固定的用户群体，随着时间的增加该群体还会不断壮大。淘宝网每天销售商品数已经接近10亿，平均每分钟的售量可以让大多数商家感到惊讶。

本系统将根据输入关键词，从淘宝爬取相关产品信息，包括产品名，售价，销售量，产地，并将对产品信息进行分析及可视化。



图3-1 抓取页面

3.2 各模块设计

3.2.1 信息模块

淘宝对于反爬虫做得非常好，在进行模拟登录的时候会不断的跳出滑块验证来进行干扰，request库目前没有较好的解决办法，而在输入用户名与密码后，淘宝会验证登录是否可行，返回True之后会获取一个token，之后浏览器会拿着token去阿里巴巴（alibaba.com）交换一个st码，浏览器拿着st码获取cookies，这时候才能登录成功，这期间将会返回大约30个参数到淘宝去验证，非常容易登录失败，登录的过程如下[[[9]](#endnote-9)]：

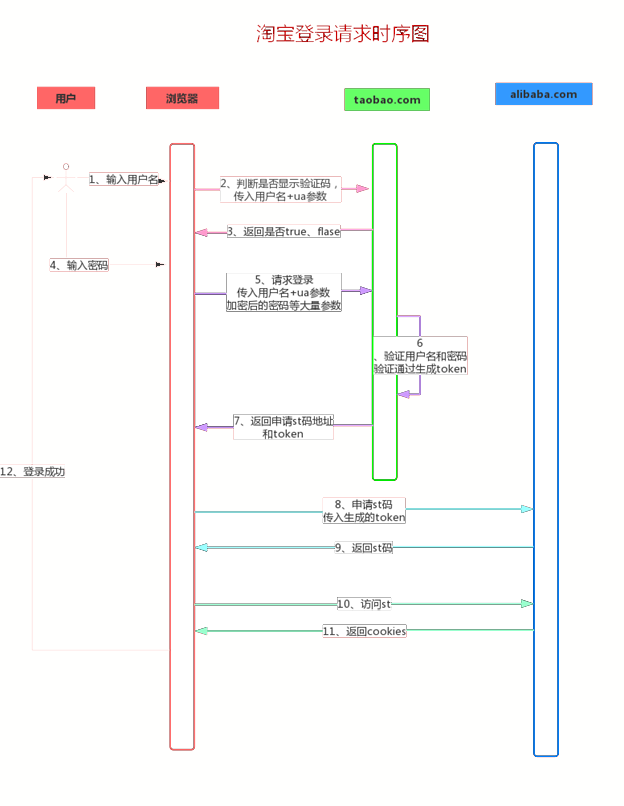


图3-2 淘宝登录请求时序图

因此为了登录更加方便，该模块将采取模拟浏览器来进行模仿真人登录，为了达到这一目标，我将借助selenium来协助完成。

selenium可以做到像真实人类对浏览器进行操作，这得归功于他有着一套完整的web系统，而且selenium不需要用户再进行任何多余操作，仅仅需通过Python启动，selenium就能对浏览器进行自动化操作，在此过程中还能测试调试，做到自适配[[[10]](#endnote-10)]。

要使用selenium，首先我们要调用它的库并定义一个浏览器驱动：from selenium import webdriver，这里我们选择的是Chrome浏览器，定义为driver=webdriver.Chrome()，然后用库中的get函数获取淘宝网址：[www.taobao.com](http://www.taobao.com)。

想要自由地搜索想要的商品，得先获取到搜索栏的地址，这里我们使用find\_element\_by\_xpath()来获取，XPath (XML Path Language)，是一种在XML文档中搜素和定位元素的查询语言，通过这个函数，我们能准确地定位搜索栏在该html中的位置，用sendkey()将我们input()的商品名字输入进去，通过相同的方式查找到搜索按钮的位置，用click()键入。

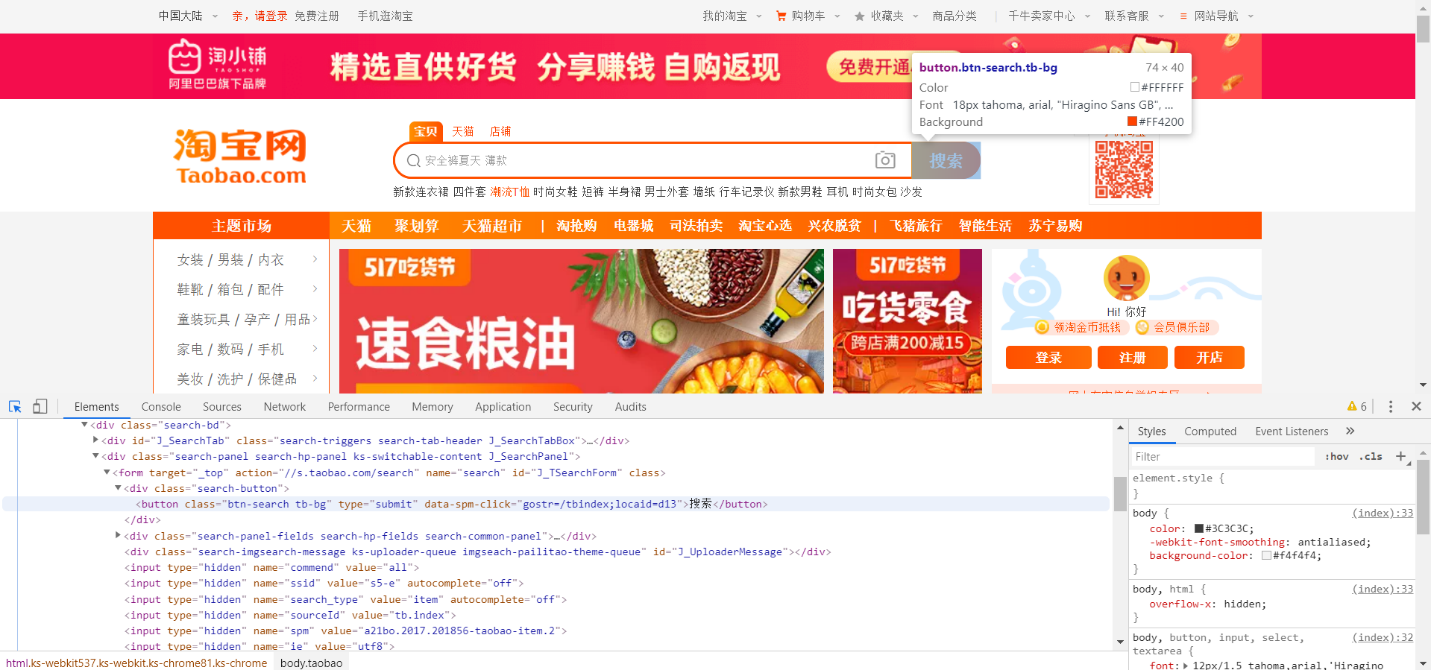


图3-3 搜索地址图

通过登录后进入到商品页面，首先要确认该商品的最大页码，同样的，寻找到最大页码的位置后将其XPath复制到函数中，并获取其信息。通过寻找其地址，我们可以发现我们获取到的是“共xx页”这样的文本，然而我们需要的是其中的数字，这里我们可以使用正则表达式即re库来处理。正则表达式的作用就是在一段长文本中来搜索我们需要的字段，当用户使用re库中的特定字符对文本进行操作处理的时候，就能对该文本中其他不需要的信息实现过滤效果，正则表达式不止只能匹配单个字符串，对于多个字符串正则表达式也有着同样的效果[[[11]](#endnote-11)]。我们便可以通过正则表达式这一特性匹配到文本中的数字，并定义该值为Int类型：

top = int(re.compile('(\d+)').search(top).group(1))。

想要爬取商品的图片，必须得等图片信息全部加载出来，但如果不把滑动条拉到相应位置，可以发现图片不会加载出来，显示为一片空白，当我们查看源代码的信息也可以发现src文件是不会显示图片信息，因此我们需要将滑动条拉动到页面的最下方，使得图片可以全部加载，下拉滑动条代码如下：

for x in range(1, 11, 2): # x按13579进位

time.sleep(0.5) # 停止时间

j = x / 10 # 滑动条位置

js = 'document.documentElement.scrollTop = document.documentElement.scrollHeight \* %f' % j

driver.execute\_script(js)

document.documentElement.scrollTop代表着浏览器滚动部分高度，document.documentElement.scrollHeight代表着浏览器所有内容高度，通过对滑动条位置的控制可将页面全部加载。

抓取完这一页的信息后，下一步就要进入下一页将以上步骤重复一遍，常规的操作是抓取到“下一页”的位置并click，但当执行该操作次数过多时，淘宝会检测到异常返回值并向该用户发送验证，于是我们不采用该操作。我们检查淘宝每一页的url信息：第二页为<https://s.taobao.com/search?q=%E6%89%8B%E8%A1%A8&imgfile=&commend=all&ssid=s5-e&search_type=item&sourceId=tb.index&spm=a21bo.2017.201856-taobao-item.1&ie=utf8&initiative_id=tbindexz_20170306&bcoffset=1&ntoffset=1&p4ppushleft=2%2C48&s=44>

第三页为

<https://s.taobao.com/search?q=%E6%89%8B%E8%A1%A8&imgfile=&commend=all&ssid=s5-e&search_type=item&sourceId=tb.index&spm=a21bo.2017.201856-taobao-item.1&ie=utf8&initiative_id=tbindexz_20170306&bcoffset=-2&ntoffset=-2&p4ppushleft=2%2C48&s=88>

…

第十五页为

<https://s.taobao.com/search?q=%E6%89%8B%E8%A1%A8&imgfile=&commend=all&ssid=s5-e&search_type=item&sourceId=tb.index&spm=a21bo.2017.201856-taobao-item.1&ie=utf8&initiative_id=tbindexz_20170306&bcoffset=-38&ntoffset=-38&p4ppushleft=2%2C48&s=616>

不难看出，每个url中的s的值跟页数有着线性关系，为s = （页数-1）\*44，我们可以通过s的值直接跳转到需要的页数。除此之外，由于网页的url太长，经过试验后发现除了q的值与s的值其他都可以忽略不计，大大简化了代码量，使代码更加简洁美观，代码为：driver.get('https://s.taobao.com/search?q={}&s={}'.format(key, 44 \* num))（这里num初始值为1，即第二页开始）。

此外，由于不同网络环境下的网页加载速度不同，如果网页因网速原因未能加载完成，此时无法下拉滑动条，更无法获取商品信息，所以我们需要添加一行代码：

driver.implicitly\_wait(10)

该代码能让浏览器最多等待10秒，使其能适应大多数网络环境。若网络环境不同，可更改函数内的数字。

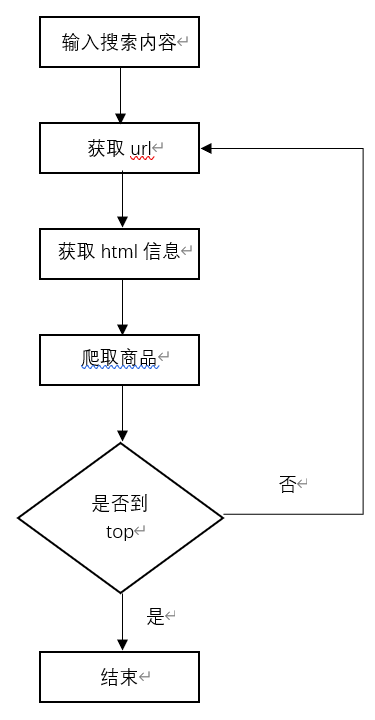


图3-4 获取信息流程图

3.2.2解析模块

通过查看网页源代码可以发现，商品的信息都包含在了一个名为[class= item]的大类下，其中又有着名为[class=item J\_MouserOnverReq]的小类，于是我们可以知道如果想有序的爬取商品信息，只需要规定在[class=item J\_MouserOnverReq]的范围内就可以了，因此可以编写代码：

divs = driver.find\_elements\_by\_xpath('//div[@class="items"]/div[@class="item J\_MouserOnverReq "]')

规定爬取范围在大的范围divs内，这样可以防止爬取到其他无关内容，并提高了爬取的准确度与效率，也便于调试与观察爬取信息。

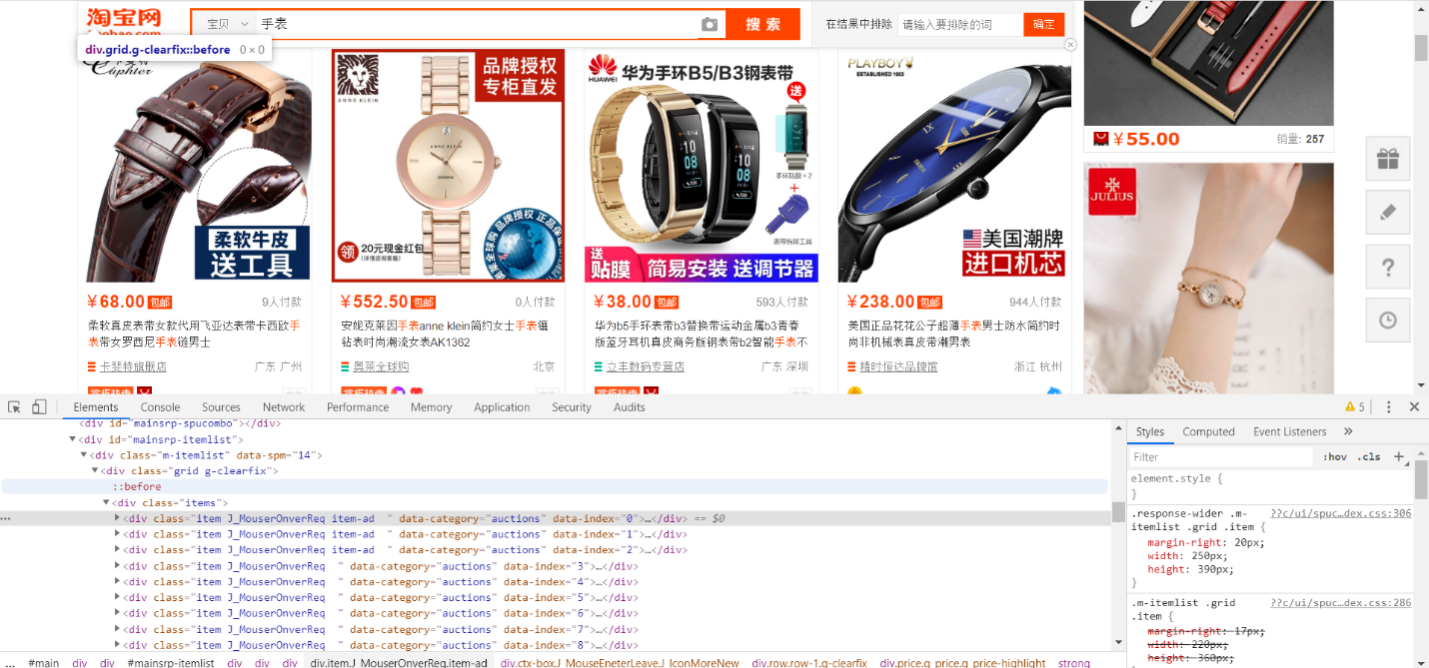


图3-5商品信息地址

查看网页源代码，可以发现商品的每种信息都存储在特定的类下。图片存储在名为pic的大类中的id为img的小类中，图片信息保存为src格式，当鼠标移动到src链接上时可以发现图片被显示出。宣传标语直接保存在名为row row-2 title的类下，这里直接提取他的文本信息就可以了。价格存储在名为row row-1 g-clearfix的大类中的名为price g\_price g\_price-highlight的小类下，这里不可以直接提取price g\_price g\_price-highlight中的文本信息，因为该部分由两部分组成：一部分为符号形式的￥，存储在<span>中，另一部分为数字形式的价格，存储在<strong>中，这里我们要提取的是<strong>中的文本。销量就是简单地存储在deal-cnt类中，直接提取文本就行了。店铺名也是简单地放在名为shop的类中，提取<span>的文本。产地就直接放在location类中，于是我们可以得到了完整代码为：

image = div.find\_element\_by\_xpath('.//div[@class="pic"]/a/img').get\_attribute('src') # 图片

info = div.find\_element\_by\_xpath('.//div[@class="row row-2 title"]').text

# 标语

price = div.find\_element\_by\_xpath('.//div[@class="row row-1 g-clearfix"]/div[@class="price g\_price g\_price-highlight"]').text # 价格

deal = div.find\_element\_by\_xpath('.//div[@class="deal-cnt"]').text

# 交易人数

name = div.find\_element\_by\_xpath('.//div[@class="shop"]/a/span[2]').text

# 店铺名

position = div.find\_element\_by\_xpath('.//div[@class="location"]').text

# 店铺所在地

但当我们提取完信息后可以发现一个问题，淘宝的销量在数额过多的时候，会以如“2万+人付款”，“8000+人付款”这样的形式保存，而当我们需要使用这些数据进行分析的时候，程序是不能识别“万”“+”这样的字符，于是我们需要将汉字的“万”转化为阿拉伯数字的“10000”，具体实现代码如下：

if '万' in deal:

deal = deal.replace('万+人付款','')

deal = float(deal) \* 10000

deal = int(deal)

deal = str(deal) + '人付款'

if '+' in deal:

deal = deal.replace('+人付款','人付款')

deal = str(deal)

这串代码实现了当识别到deal中含有“万”或者“+”的时候执行replace函数操作，由但于我们保存的deal为str即字符串形式，不能直接将文本内的数字与10000相乘，所以我们替换掉文字信息后需先将deal转化为浮点数形式再赋值，最后再将deal转回文本格式便于保存。

3.2.3 存储模块

在Python3中我们使用pymongo与MongoDB进行连接，在使用MongoDB前我们首先要创建一个数据库，这里使用MongoClient对象，并且通过指定的地址创建链接，再创建一个数据库，将其名字与URL连接起来[[[12]](#endnote-12)]，这时我们就能得到一个指定到MongoDB的链接对象，创建后的对象命名为：connection=pymongo.MongoClient()，这里由于之前已经在MongoDB中声明了localhost为27017，故不再声明。因为我们没有在MongoDB中创建集合，也没用在集合中插入文档，所以当我们在cmd查看MongoDB状态的时候，我们会发现我们所创建的数据库并不存在。其实这个时候数据库已经建立好了，但因为库中没有内容，所以无法查看。因此我们创建一个名为taobao的数据集，命名为tdb，再创建一个文档test，并将其赋值给post\_info，代表着post\_info指向test。当我们爬取到想要的数据信息后，我们将其保存在一个dict内：product = {'图片':image,'标语':info,'价格':price,'交易人数':deal,'店铺名':name,'店铺所在地':position}，然后通过pymongo的insert\_one()函数，将product逐条存储到数据库，即test文档中。这里我们还使用了MongoDB的可视化工具Robo 3T来帮助我们更直观的查看数据：

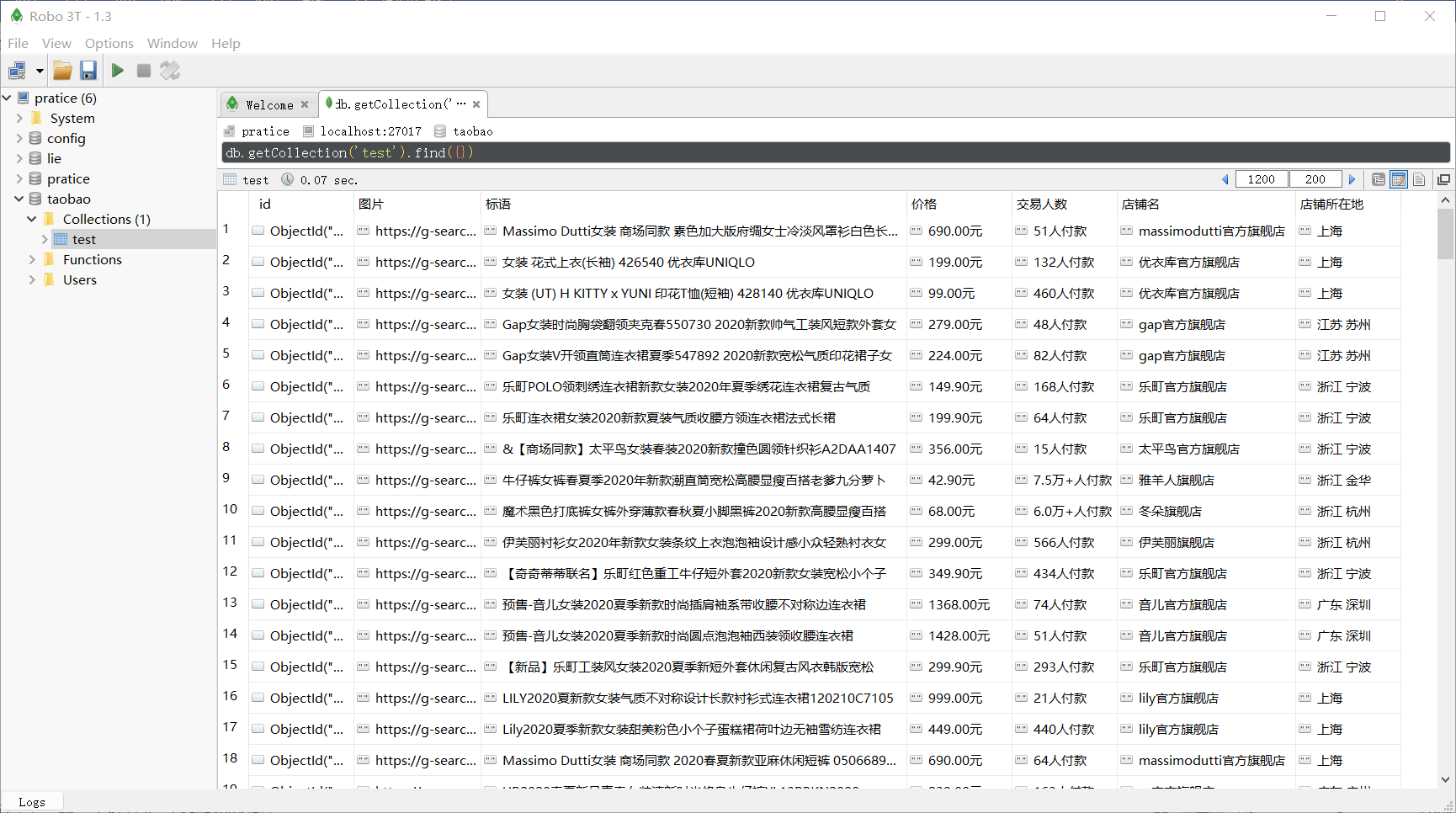


图3-6数据库

为了更好地进行数据分析，我们需要将数据导入到csv文件中，由于爬取数据量过多，普通地将数据逐条存入csv文件会耗时非常久，于是我们引用pandas库来帮助我们，这里我们使用的是pandas的DataFrame格式，。

DataFrame是一种表格型数据结构，用户可以根据索引搜索到自己想要的数据。由于信息存储到csv中为二维数组，因此在pandas的帮助下效率会提高[[[13]](#endnote-13)]。

我们将爬取到的数据集合product通过pd.DataFrame()赋值给data，然后执行下列代码：

data.to\_csv(file\_name,header=csv\_head,index=False,mode='a+',encoding='utf\_8\_sig')

要注意的是，如果这里encoding=’utf-8’将会造成数据乱码，此时需要对csv文件进行重新编码。

将爬取到的图片保存到本地也是非常简单，只需要借助requests库的帮助建立函数，获取到爬取到的图片url，然后将图片以“wb”的格式通过write函数下载到设定好的文件夹内：

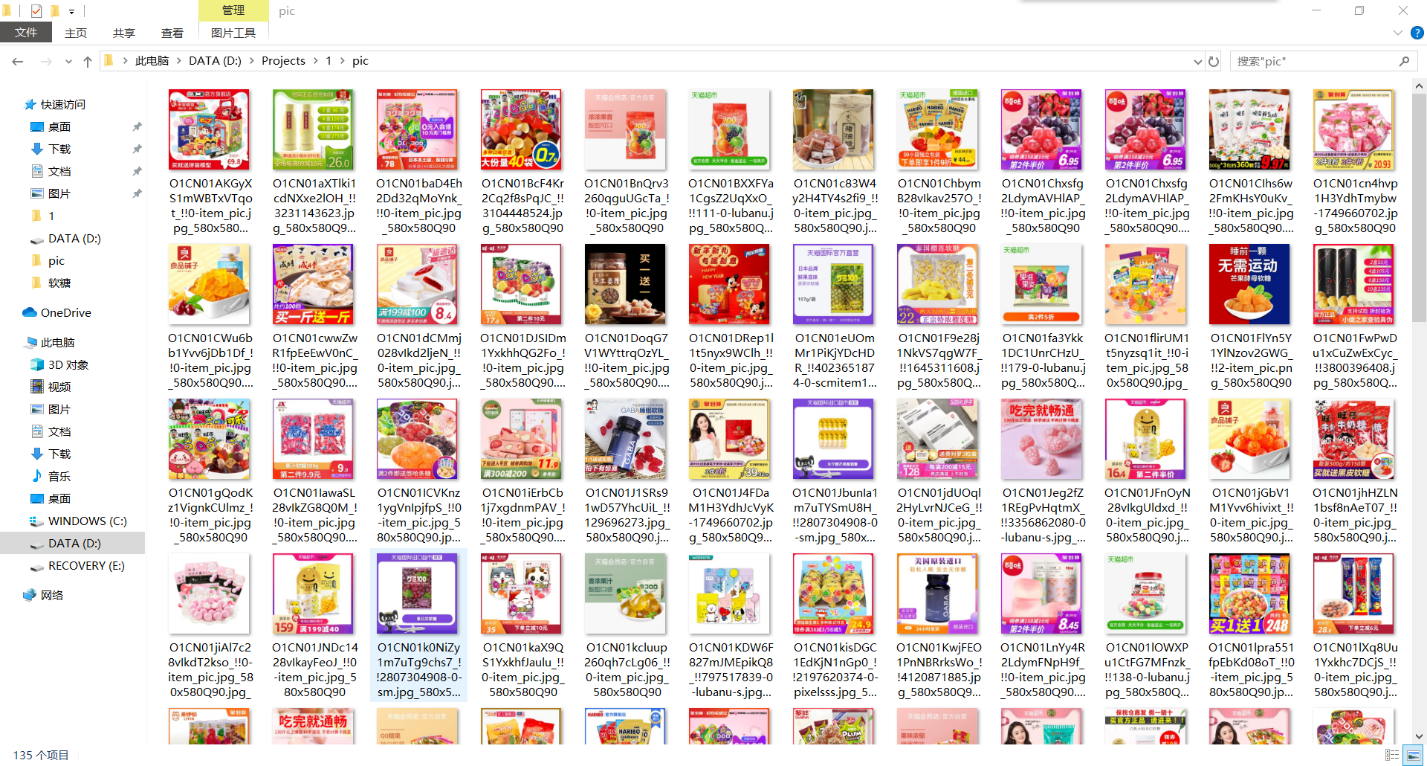


图3-7 图片保存

3.3 本章小结

本章主要介绍了被爬取的淘宝网的基本信息及其登录流程，展示了如何通过自动化浏览器绕过淘宝的反爬虫机制取获取内部的商品信息，接着分模块说明了如何获取各商品信息及解析后爬取到项目中，继而通过连接将获取的信息存储在数据库中。要注意的是，不同的内容存储的方式不同，根据目标信息的不同要对爬取方式做出相应的修改，以免造成数据混乱或者数据缺失。

# 第四章 数据处理

4.1 数据拆分

为了使数据分析及可视化更加方便，我们需要对数据进行一系列的处理。比如将中文去除，转变数据格式，合并数据等，在此过程中，我们将借用一系列Python库来协助。

首先导入我们之前保存好的csv文件，我们选择的例子是软糖商品，然后将其数据集赋予到data内，这里同样的是借助了pandas库的帮助。然后将数据集的列表名更改为'raw\_title','view\_price'，'view\_sales','item\_store','item\_loc'以便操作。由于'view\_sales'为float格式，这里我们执行data.view\_sales.astype(str)，这样我们可以得到一个可拆分的，以文本格式存储的'view\_sales'。接着进行以下步骤：

data['province'] = data.item\_loc.apply(lambda x: x.split()[0])

data['city'] = data.item\_loc.apply(lambda x: x.split()[0] if len(x) < 4 else x.split()[1])

data['sales'] = data.view\_sales.apply(lambda x: x.split('人')[0])

apply函数能对dataframe进行转换，转换成一个新的dataframe。在这里先使用split函数及lambda对数据集进行操作：对'item\_loc'进行split，然后取第一个分片保存到data['province']中，再进行重复操作，保存到data['city']中，但是考虑到有直辖市的情况，这里要进行一个判断：如果数据的长度小于4则保留第一个分片，否则取第二个分片，然后将'view\_sales'以“人“为标志进行分割取数字，但在这里要注意，考虑到数据缺失的因素，split函数无法对Nan即空的数据进行分割，所以在前面我们仍需要添加一个函数fillna()保证数据结构的完整性。

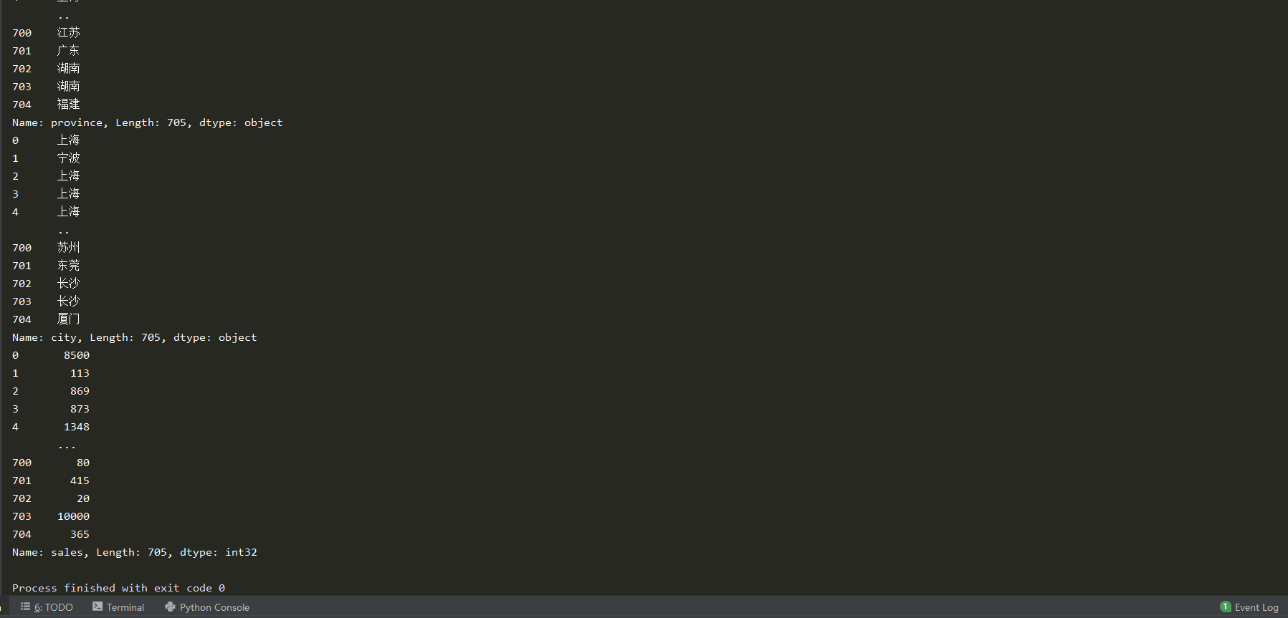


图4-1 省份、城市、销量数据集

这样我们就能得到三个新的数据集，分别代表着省份、城市、销量。我们还需要将省份和城市的数据集转化为category类型。将这两个数据集转化为该类型意味着对于其他被统计的变量都打上了特定的地域标记，这能使数据分析更方便快捷。

4.2 数据清洗

4.2.1 分词

分词模块要引用的是Jieba库。Jieba库能做到对较长的中文文本进行分词操作，我们知道淘宝上大部分店家会将各种能搜索到本店商品的词语进行排列组合并放到标语上，这样用户可以通过搜索不同的关键词搜到店铺。同样的，Jieba能将这些无关联的词组拆分开，且能按照汉字的组合概率将字与字进行组合分类，形成分词结果[[[14]](#endnote-14)]。由于Jieba具有着较为丰富且严谨的算法，这使得其分词结果有较大的准确性。

Jieba的功能使用非常简单，该模块仅使用lcut()来完成分词功能。先将商品的标语‘raw\_title’转为列表：title = data.raw\_title.values.tolist()，然后对所有在title内的句子line进行分词，将分词结果放入数组title\_s内，具体代码为：

title\_s = []

for line in title:

title\_cut = jieba.lcut(line)

title\_s.append(title\_cut)

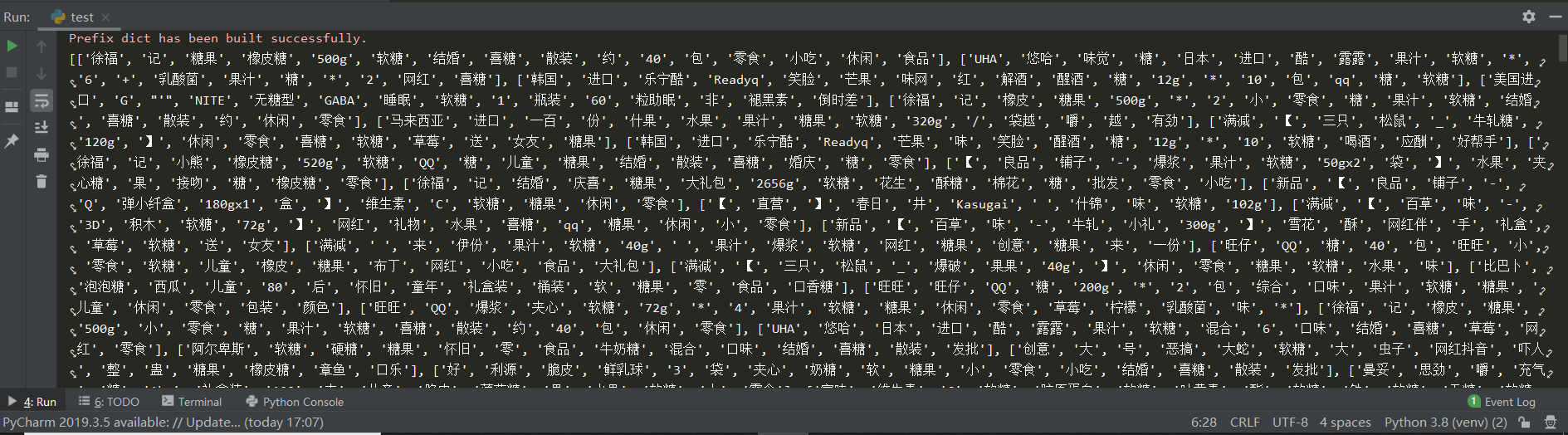


图4-2 分词结果

4.2.2剔除停用词

在分词过程中会有部分不相关的词组被放入列表，或者是无意义的空格，符号如“[”之类，会影响到数据的分析与统计，于是我们还需要将这部分不需要的词组剔除[[[15]](#endnote-15)]。

我们先创建名为stopwords的对象，然后读取我们编写的停用词表的xlsx文档，通过tolist()转化为list格式。接着对于既存在于line（line存在于title\_s）中，又不属于stopwords的词组word，将其放入到title\_clean中，title\_clean代表着剔除后的词组，具体代码为：

title\_clean = []

for line in title\_s:

line\_clean = []

for word in line:

if word not in stopwords:

line\_clean.append(word)

title\_clean.append(line\_clean)

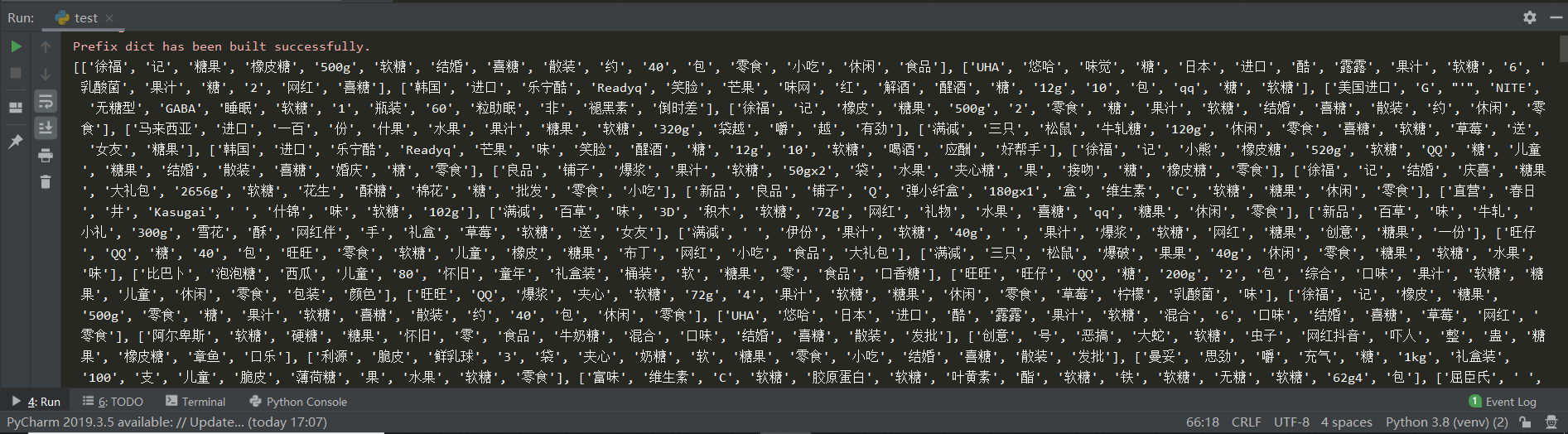


图4-3 剔除结果

很明显可以看出，相比分词后的输出结果，这次输出的无用字符基本已经消失了。

4.2.3 去重

该模块将对title\_clean中的元素进行分析，将重复的list删除，防止同一长文本中存在同样的词组干扰对每个词语的个数的统计。

首先是取存在于过滤后的数据 title\_clean中的词组line，创建一个数组line\_dist，对于存在于line中但不存在于line\_dist中的词word，将其放入line\_dist中，这样可以保证word的唯一。最后将去重后的line\_dist放入title\_clean\_dist中。

具体实现代码为：

title\_clean\_dist = []

for line in title\_clean:

line\_dist = []

for word in line:

if word not in line\_dist:

line\_dist.append(word)

title\_clean\_dist.append(line\_dist)

4.3 本章总结

本章介绍了对于上一章爬取到的商品数据的后续处理，包括将数据集的列表头进行拆解转化类型，将长标语进行智能词组拆分，将无意义的，分解错的词组从列表中剔除，并去除重复词组，这对于后续的数据分析及可视化的效率有着较大的提升，不仅如此，还大大地提高了数据地准确性，也方便对于错误数据的处理。

# 第五章 商品信息可视化分析

## 5.1 词云可视化

Word可以显示大部分文本中的高频词汇，以使用户了解关键信息，免于长阅读的烦恼。在这个模块我们同样的想更直观地了解各商品中哪种信息比重更大，于是我们采用wordcloud库来帮助我们[[[16]](#endnote-16)]。

为了达成这个目的，我们需要将清洗后，存在于每个数组line中的词组word全部提取出来，并合并成一个数据组allwords\_clean\_dist，这样才能统计各词组数量。接着我们将这个数据组转化成一个数据框，再对该数据框执行pandas库中的value\_counts函数。

value\_counts函数能够帮我们计算在我们生成的数据框内，每个词组存在多少重复值。接着我们重新构建索引，有如下结果 ：

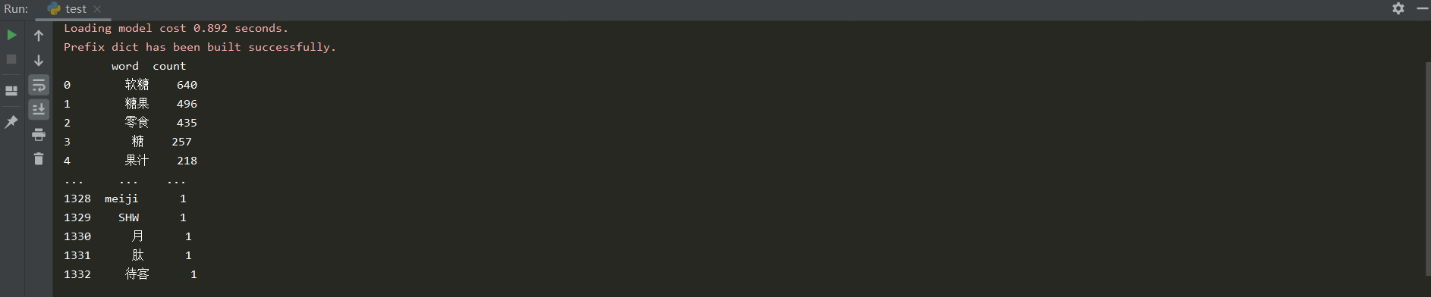


图5-1 分频结果

在使用wordcloud来作图的时候，我们同样的需要使用另外一个用于作图的Python库来协助matplotlib。matplotlib是受MATLAB的启发构建的。在数据绘图方面，人们通常会选择去使用MATLAB[[[17]](#endnote-17)]。MATLAB语言是面向过程的，Matlab可以让用户通过简单的代码就能绘制复杂的图形，再通过几行代码就能对该图形进行各种调整。

Matplotlib.pyplot能使用户实现在Python上进行MATLAB的绘图工作，完整作图过程如下：

plt.figure(figsize=(20,10))

w\_c = WordCloud(font\_path="./1/simhei.ttf",

background\_color='white',

max\_font\_size=60,

margin=1)

wc = w\_c.fit\_words({x[0]:x[1] for x in word\_count.head(100).values})

plt.imshow(wc, interpolation='bilinear')

plt.axis("off")

首先设置图片大小为20\*10，然后设置词云的格式，具体数据不再赘述。接着使用fit\_words函数输入前100个词组的值，根据词频生成词云。在输出过程中对matplotlib中的interpolation参数进行了调整，使得输出结果更清晰。



图5-2 词云可视化

对词云进行分析，可以很明显的看出水果味、橡皮糖、喜糖受欢迎程度更高；相比于传统的旺旺、徐福记等品牌，进口的和网红糖销售量更多，人们对于口味的影响不是很在意，对于产地的要求也没那么高。

## 5.2 不同词组对于销量的影响

词云可视化只能给我们一个大概的概念，而不能准确地说明哪种词组应用时销量更高，因此我们需要对词组进行进一步分析其与销量的关系。

首先我们对于每个存在于title\_clean\_dist中的词组，将其销量数据统计求和放入数组中，然后将其转化为数据框中，并将其与word\_count合并成一个新表，该表的各列为词组，数量，销量统计。接着使用sort\_value函数对其使用升序操作并取其前30组数据进行作图。具体作图参数为：

index = np.arange(df\_w\_s.word.size)

plt.figure(figsize = (6,12))

plt.barh(index,

df\_w\_s.w\_s\_sum,

color = 'blue',

align = 'center',

alpha = 0.8)

plt.yticks(index, df\_w\_s.word, fontsize = 11)

for y,x in zip(index, df\_w\_s.w\_s\_sum):

plt.text(x,y,"%.0f" %x, ha='left', va='center',fontsize = 11)

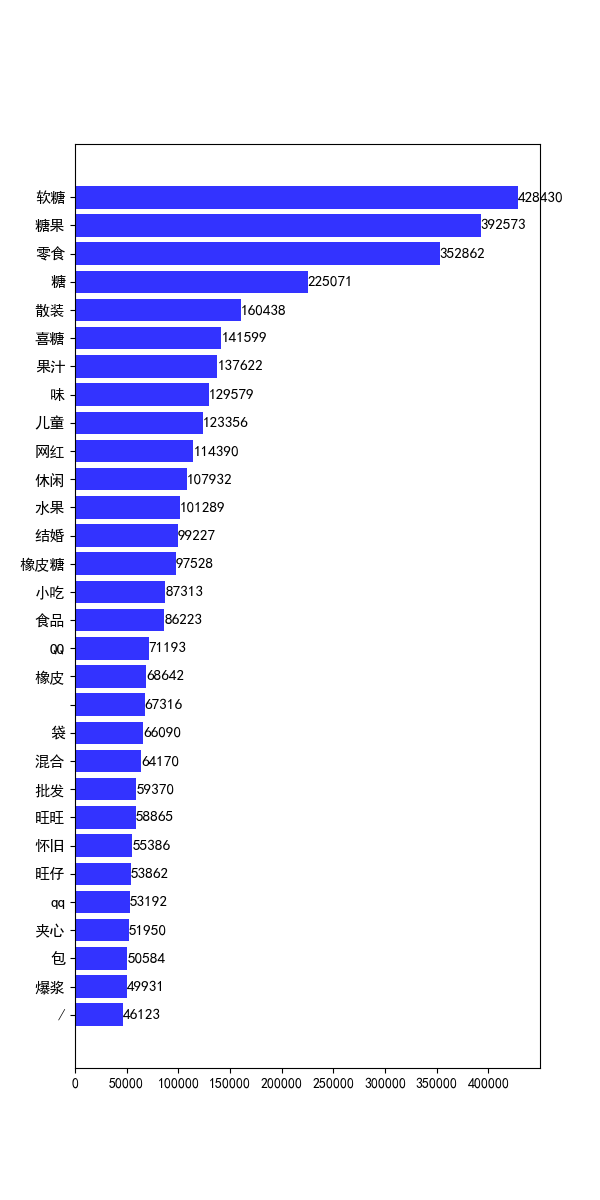


图5-3影响销量的词组可视化

从相关数据统计可以看出来：软糖是卖得最多的；散装销量远大于袋装与混合及批发；喜糖位居第六，说明有一批固定的喜宴会选择从网上购买喜糖而不是线下；人们在选择购买糖果时更多的是为了零食和小孩子，而非关注是否网红和其品牌。

## 5.3 商品价格与销量的关系

我们分析了词组对于销量的影响，但是价格同样会影响到整个产品的销售额，该模块就将对产品的价格与销量进行分析。从之前的输出结果就能看出，商品的各个信息是一个完整的数据框，每个商品有着它所对应的价格以及销量，因此当我们在作图的过程中不需要再将两个因素提取出来构成新的数据框。

为了减小特殊价格对于商品统计的影响，为了可视化更加直观清晰，这里我们选择将商品的价格规定在20000元以内。为了能对商品的各项数据都进行数学分析，我们需要将数据组内的价格以及销量列转化为整数格式。

为了数据的直观明了，这里我们调用matplotlib库中的scatter来帮助我们绘制散点图，将x轴设置为商品价格，y轴设置为销量，为了获得价格对销售额的影响，我们需要将价格与销量相乘并调用Seaborn库。该模块将会调用Seaborn库中的regplot函数来展示线性回归关系：

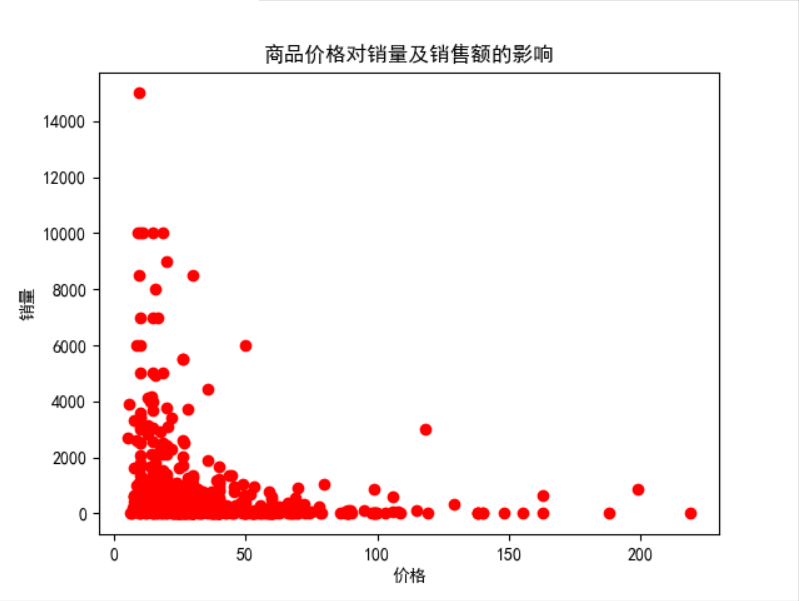


图5-4 价格对销量的影响

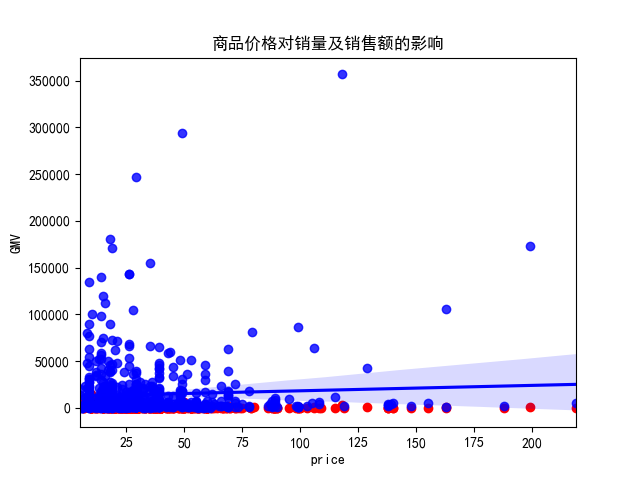


图5-5 价格对销售额的影响

通过对比这两张图，我们可以得知：随着价格的增加，销量会呈现断崖式的下跌，位于[0,50]区间的商品销售量相比于其他区间有着明显的优势；当产品价格处于100以上的时候，销量会非常低且大部分无限接近于0；由线性回归拟合线可以看出，商品销售额随着价格增长呈现上升趋势，但不是很明显；多数商品的价格偏低，销售额也偏低；[25,50],[50,75],[100,125]的价格区间各有一个销售额显著偏高的商品，其中位于125价位的商品销售额最高；[125,200]区间有三个明显的销售额与价格成正比的商品，可以得知高端糖果依旧有市场。

## 5.4产地对商品的影响

在之前我们已经将省份及城市转化为了category格式，于是在该模块我们可以直接调用data.province.value\_counts().plot()来计算在该列同一省份有多少个重复值，并根据计算结果绘制柱状图：

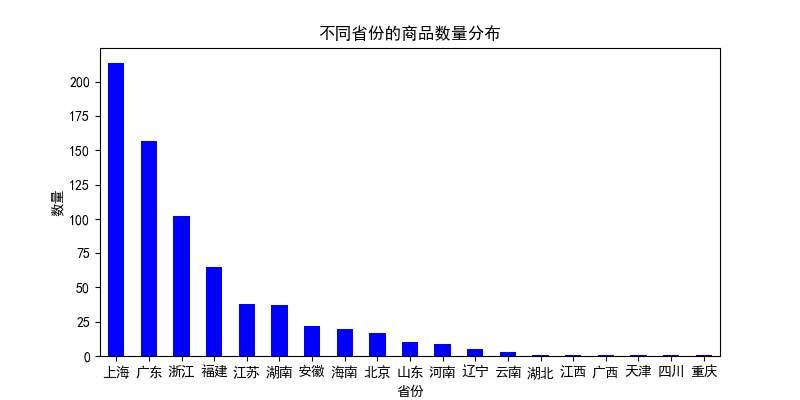


图5-6 不同省份对商品数量的影响

由图可知：上海、广东、浙江的商品数量远超其他省份，而上海的数量最高，说明在糖果这一产业，上海的店铺占主导地位。

Pivot\_table具有灵活性高，脉络清晰，操作性强的特点，所以我们使用它来对产地对销量的影响构建图表：

data.pivot\_table(index='province',values='sales',aggfunc=np.mean)

index和value能够很好地构建省份与销量之间的关系，接着通过aggfunc=npmean计算销量的平均值，然后将其通过sort\_values按照从大到小排序，接着重设索引。最后我们就可以在plt.bar函数的帮助下创建图表了：

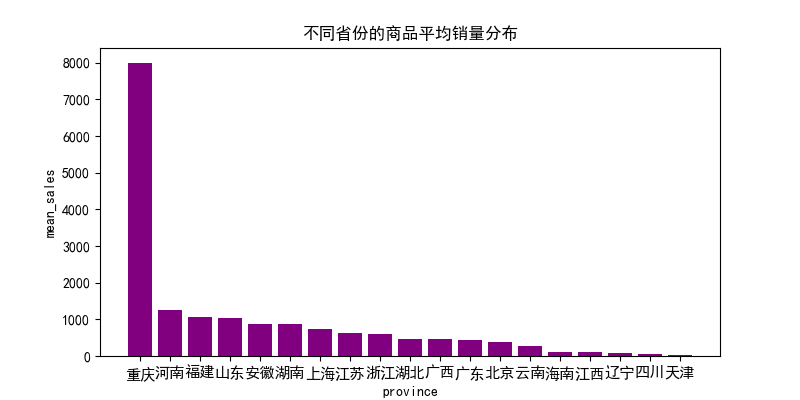


图5-7 不同省份对商品平均销量的影响

从两张图表可以看出，重庆虽然店铺数量很少，处于垫底的位置，但是它的销量却遥遥领先。反而是店铺数量较多的上海、广东、浙江的平均销量非常低，可以得知有很大一部分的店铺销量非常差。

## 5.5总体设计

在对各种数据进行处理之后，为了使用户能够更直观地了解自身需求，因此设置一个选项单来由用户自行选择。

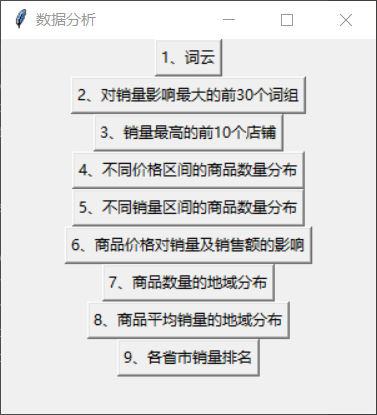


图5-8 选单

该选单是通过tkinter与py文件内构建选单界面，之后用pyinstaller将其打包成可用程序。

用户可以根据自身需求，点击相关选项来获取相关内容，比如想获取销量排名，就可以点击第9项，即可得到结果：

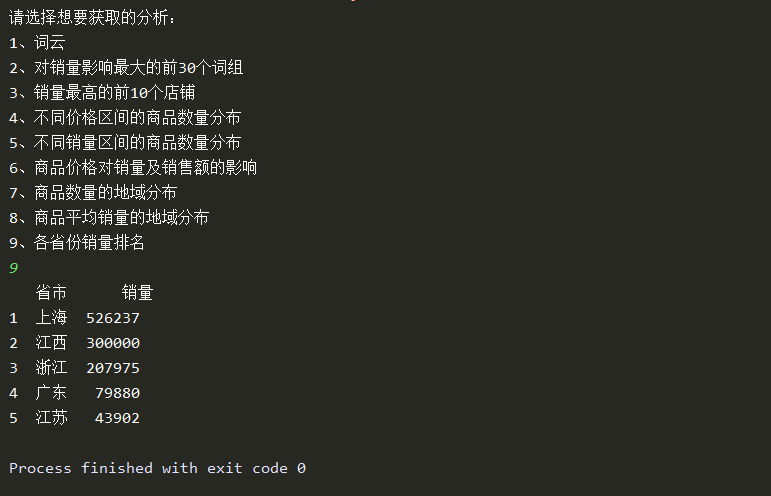


图5-9 排名结果

亦或是想获得地域分布，即可点击选项7，可以很方便地获取信息：

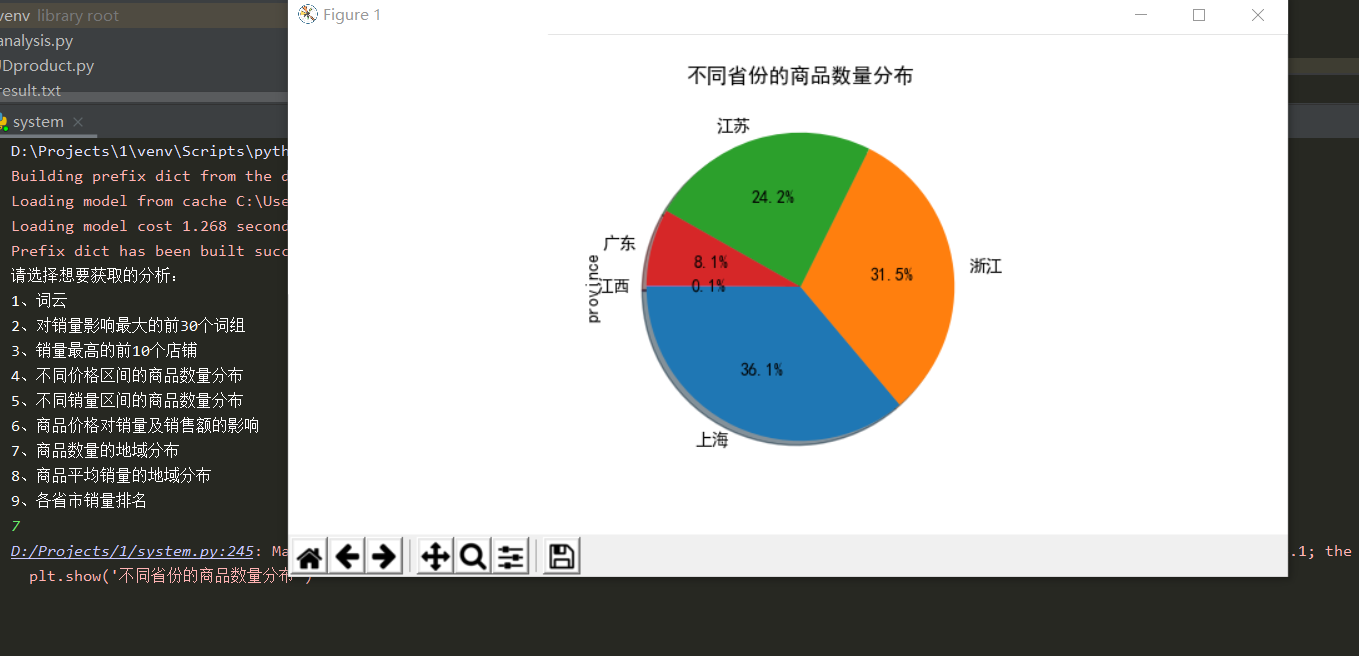


图5-10 不同省份商品数量分布

## 5.6 本章总结

从这一章我们能够很直观地了解到各种因素对于商品的影响，包括地理位置，宣传标语，价格等，虽然还有一些影响因素无法统计，但根据这些分析用户已经能够大概的了解影响到商品的因素并对商品的评估做出调整。

# 第六章 总结与展望

随着时代的进步，网络越来越普及到人们生活的方方面面，过去需要出门花费过多时间才能购买的物品，现如今在网上或者在手机上动动手就能轻松愉快地下单，免去了大量的时间浪费，可以毫不夸张的说，网购改变了人们的生活。因此越来越多的人发现了财富密码，发现了自己身边具有的商机，他们也会将自己的产品在网上售卖。

但是随着与日俱增的店家数量，大多数店家并不知道该卖什么，怎么卖，他们没有任何销售经验，仅凭观察别人的方式就一股脑地投入到网售的热潮，难免会赔得血本无归。但随着科技的发展，大数据概念的成熟，人们可以通过编写软件爬取商品相关信息，并对其做分析总结，这就大大减少了店家的学习成本和金钱成本，也对于想了解网购的他人有了更清晰直观的展示，科技改变生活。

本次设计能够很好地解决以上问题，通过使用Python语言，在PyCharm开发环境中去编写爬虫程序，通过selenium启动代理浏览器，在搜索文本框中输入搜索关键字，从Web服务器调用Web爬虫模块，接着爬虫模块检查Internet连接，如果网络连接成功，就处理文本搜索引擎的关键字，然后将搜索引擎结果发送给解析器，直到所有链接都被处理。通过requests库来提交网络请求，爬取HTML页面，获得网页源码。

通过正则表达式解析源码，即解析页面，然后提取相关数据。最后使用pymongo库连接MongoDB数据库和Python将爬取的数据存入该数据库。之后再调用requests库将获取到的图片信息保存到我们设置好的文件夹内。通过这几个步骤我们就能实现基本的爬虫功能并将数据保存。

爬取完成产品信息后，立即将数据导出，并将其分类保存以便于数据处理。将每个长标语进行分词封存于列表中，然后进行一系列清洗，去重，统计工作，成功地将商品的各类信息进行归类整理并保存于各个列表内。

对每一个不同关系的数据集合进行分析，即可得到不同因素对彼此的影响，从而能让使用者对于商品的数据有着更进一步的认知。通过对商品数据的比较及分析，使用者可以对不同商品的市场环境有更深入的理解。

总的来说，虽然在设计过程中会遇到各种各样奇奇怪怪的问题，但是都能通过大部分常规方式去解决，在解决过程中也提高了自身对于Python以及程序设计的理解。

# 致谢

四年大学时光转瞬即逝，从开始的刚入学报到时对大学生活的期待与憧憬，到现在大学生活即将结束时内心的满怀感慨，感叹那永远消逝的时间。现在即将踏入社会，开始工作，而这段大学四年时光将永远铭记。感谢老师们，让我见识了知识的魅力，课堂的精彩。感谢身边的每一个人，是他们帮助我学习成长，让我拥有了充实的大学时光。

感谢我的毕设指导老师张璘老师，是她耐心细致的讲解，让我逐步对课题有了了解和认识。感谢老师的认真负责，指导我开始着手毕业设计，帮助我一步步解决问题，使我从开始的不知如何下手，到后来明白设计思路，找到一往无前完成毕设的信心。同时也是老师细致入微的关心鼓励让我不断完善论文，追求极致。

还要感谢我的家人对我的谆谆教诲和关爱，让我懂得想要做成任何事都需要坚持，不断地努力拼搏，唯有靠自己才是最稳妥，最有保障的。

最后感谢所有给我提供帮助的人，谢谢你们对我的帮助和支持。

# 参考文献:

[1] 刘金红，陆余良.主题网络爬虫研究综述[J].计算机应用研究，2007,(10):70-79.

[2] 罗刚，王振东.自己动手写网络爬虫[M].清华大学出版社，2010,(10).

[3] 大连海事大学计算机科学与技术学院.主题搜索引擎中网络爬虫的搜索策略研究计算机工程与科学，2008，(03):44-46.

[4] 罗刚.自己动手写搜索引擎[M].电子工业出版社，2009，(11).

[5] 郑志高，刘庆圣，陈立彬.基于主题网络爬虫的网络学习资源收集平台的设计[J].中国教育信息化，2010，(01):55-67.

[6] 汪涛，樊孝忠.主题爬虫的设计与实现[J].计算机应用，2004，(S1):110-121.

[7] 汪涛，樊孝忠.链接分析对主题爬虫的改进[J].计算机应用，2004，(S2):55-70.

[8] 尹江，尹治本，黄洪.网络爬虫效率瓶颈的分析与解决方案[I].计算机应用, 2008, (5):20-31.

[9] 汪涛,樊孝忠,顾益军,刘林.基于概念分析的主题爬虫设计[J].北京理工大学学报，2004,(10):33-41.

[10]李蕾，王楠，张剑，钟义信，郭祥昊，贾自燕.中文搜索引擎概念检索初探[J].计算机工程与应用，2000，(06):66-75.

[11] 潘春华，冯太明，武港山.基于移动爬虫的专用Web信息收集系统的设计[J].计算机工程与应用，2003，(36):99-109.

[12]赫枫龄，左万利.利用超链接信息改进网页爬行器的搜索策略[J].吉林大学学报(信息科学版)，2005，(01):100-108.

[13]J.E.Tomlinson,J.H.Arnott,J.J.Harou.A water resource simulator in Python[J]. Environmental Modelling and Software,2020,126.

[14]J.Alex Brashears,H.Bobby Fokidis,Dale F.DeNardo.Fear-based aggression and its relationship to corticosterone responsiveness in three species of python[J]. General and Comparative Endocrinology,2020,289.

[15] Mikael Yamanee-Nolin, Niklas Andersson, Bernt Nilsson, Mark Max-Hansen, Oleg Pajalic. Trajectory optimization of an oscillating industrial two-stage evaporator utilizing a Python-Aspen Plus Dynamics toolchain[J]. Chemical Engineering Research and Design,2020,155.

[16] Jari Torniainen,Isaac O.Afara, Mithilesh Prakash, Jaakko K. Sarin,Lauri Stenroth, Juha Töyräs. Open-Source Python Module for Automated Preprocessing of Near Infrared Spectroscopic Data[J]. Analytica Chimica Acta,2020.

[17] Tanaka Hiroki, Okuda Katsuhiro, Ohtani Seiji, Asari Masaru, Horioka Kie, Isozaki Shotaro, Hayakawa Akira, Ogawa Katsuhiro, Hiroshi Shiono, Shimizu Keiko. Chained nuclei and python pattern in skeletal muscle cells as histological markers for electrical injury.[J]. Legal medicine (Tokyo, Japan),2017,32.

1. [↑](#endnote-ref-1)
2. [↑](#endnote-ref-2)
3. [↑](#endnote-ref-3)
4. [↑](#endnote-ref-4)
5. [↑](#endnote-ref-5)
6. [↑](#endnote-ref-6)
7. [↑](#endnote-ref-7)
8. [↑](#endnote-ref-8)
9. [↑](#endnote-ref-9)
10. [↑](#endnote-ref-10)
11. [↑](#endnote-ref-11)
12. [↑](#endnote-ref-12)
13. [↑](#endnote-ref-13)
14. [↑](#endnote-ref-14)
15. [↑](#endnote-ref-15)
16. [↑](#endnote-ref-16)
17. [↑](#endnote-ref-17)