基于 Spark 的房产大数据分析与实现

专业：软件工程　年级:2015 级　学生：高开源　指导教师：黄风华

摘　要

随着大数据技术和数据分析技术的快速发展，大数据技术对人们的生活产生了巨大的影响。大数据来源于互联网和其他信息系统，经过分析处理框架和算法的结合得出深层次的分析结果，有利于业务的智能化运转。由于目前很少有从数据采集到数据分析的一体化平台，因此本文研究分布式爬虫和大数据分析结合的实现方法。

在系统的实现上，采用 Scrapy 框架进行爬虫程序的编写，使用 Gerapy 工具对爬虫程序进行分布式部署。在数据分析模块主要依靠 Spark 大数据平台，将 Spark 和 FPGrowth 算法和 K-means 算法相结合，最后凭借 matplotlib 和 Tableau 技术实现了数据可视化，并根据 Spark 分析结果和可视化图表进行详细分析与总结。

本文以房产交易网站数据为实例，首先阐述了选题的背景和意义，对房产行业以及相关技术进行了解。接着从软硬件的安装与部署开始，介绍了分布式爬虫、Spark 大数据分析和数据可视化实现全过程。最后，基于数据分析和可视化的结果进行了全面总结，并提出了本课题存在的问题和不足的地方，对系统的未来发展目标进行了规划和展望，有待于在未来的工作和学习中进行下一步研究。

关键词：Spark　分布式爬虫　房产　大数据　数据可视化

目　　录

1 绪论

1.1 选题的背景和意义

近几年国民经济的快速增长带动了各个行业的发展，而房产作为国民经济的重要支柱产业，发展尤为突出。借助互联网的浪潮，房产行业产生了浩瀚的交易数据，而且数量在不断快速的增长。如何了解到准确且全面的房源信息，是购房者、房产开发商、房地产商和城市管理者的重点关注问题[1]。对于购房者来说，一个精准的数据来源，有利于买到心仪的房子，即使仅仅用作投资，也可了解到何处的房产能大概率提高收益。对于城市管理者来说，了解到城市房产分布情况，对城市规划和建设以及对交通设施的部署都有决定性的作用[2]。

如今，大数据的概念蔓延，各行各业都在积极响应其发展趋势，房产行业也不例外，大量的房产交易网站出现，但是难以看到更详细的数据分析，购房者很难做出抉择，城市管理者也无法根据房价信息做出对应的政策调整[3]。但是，近年来大数据技术的快速发展，解决这些问题变得更有可能。基于 Hadoop 的一系列大数据框架，构成了一个繁荣的生态圈，其中 Spark[4]提供了一个更通用、更快的数据处理平台，其基于内存的强大计算能力，和 Hadoop 相比，Spark 可以让程序在内存中运行速度提高 100 倍，或者在磁盘上运行时速度提高 10 倍，而且 Spark 已经实现了相当多的机器学习算法。

本文通过分布式爬虫技术和 Spark 平台来实现房产的大数据分析[5]，首先利用 Scrapy 爬虫框架进行数据抓取，使用 Gerapy 分布式管理爬虫工具[6]进行爬虫的部署，通过 Gerapy 的调度功能对部署到多台服务器上的爬虫进行管理，将爬取的结果进行数据清洗，使用 Spark 的机器学习算法对数据进行分析，最后通过 matplotlib 和 Tableau 进行数据可视化的展示。

1.2 相关研究现状

目前国内对房产行业的研究主要偏向于单纯对数据进行分析，数据源主要是网络上公开的数据库，但是这些数据往往不是最新的数据，分析出来的结果并不能反馈出实时最新的数据。另外一种情况就是数据源往往太过单一，比如从单一网站获取的房产数据信息或者只有一个城市的房产数据信息，分析出来的结果也就比较偏颇。

对于网络爬虫技术，最近几年发展极其迅速，特别是和 Python 语言相结合，产生了很多网络爬虫框架，对于网络爬虫程序编写者来说，代码量也减轻了不少，不需要对爬虫原理有极其深刻的研究。比如流行的 Beautiful Soup 库，可以通过解析文档的方式

为用户提供需要抓取的数据；Scrapy 框架[7]擅长提取结构性数据，性能也十分高效。

1.3 论文主要工作

本文主要做的工作就是对房产交易网站进行数据抓取，并将爬虫爬取程序进行多主机分布式部署，爬取下来的数据进行数据清洗后，使用 Spark 大数据框架进行数据的分析，将分析出来的结果进行可视化的展示，是一个集数据收集、数据分析和数据可视化的一站式系统。

本论文的主要内容有：

（1）研究网络爬虫技术，特别是分布式爬虫的实现，通过可视化的爬虫管理工具对多主机上部署多爬虫。

（2）基于 Docker 技术对服务器上的环境进行统一化管理，保证每个爬虫服务器上的环境保持一致。

（3）基于 Spark 框架对爬取下来的数据和相关算法进行结合，分析出结果。

（4）基于 matplotlib 可视化库和 Tableau 平台进行数据的可视化分析，产出多张数据可视化图表。

本文就房产数据分析的主要功能、过程与任务，完成了以下工作：

（1）需求分析：通过对将要爬取房产交易网站的结构进行分析、分布式爬虫技术的需求、Spark 大数据分析框架和数据可视化的需求进行了分析。

（2）系统设计：就房产交易网站的分布式爬虫部署、房产交易数据的大数据分析和房产交易分析数据的可视化展示进行设计，包含房产交易网站的选取，分布式爬虫系统和可视化爬虫管理系统，大数据处理整体系统的设计，数据分析结果的可视化展示设计等。

（3）系统实现：根据前期设定好的分析需求和系统设计进行实践，使用的主要编程语言为 Python，爬虫部分主要基于 Scrapy 框架，爬虫的分布式实现主要基于 Scrapyd 框架，利用 Docker 技术进行分布式爬虫服务器的环境配置，分布式爬虫的可视化管理功能则依赖 Gerapy 工具，该工具可以对多服务器上的多个爬虫程序进行单独部署，并将爬取下来的数据通过 Spark 框架和数据分析算法的结合，最后通过 matplotlib 可视化库和 Tableau 可视化工具实现了分析结果可视化。

在系统实现的过程中，独立完成了房产交易网站的选取，独立编写了房产交易网站的网络爬虫程序，在爬取程序运行失败的时候，做出了相对应爬取策略的更新，保证了数据来源的及时；独立部署分布式爬虫服务器，使用 Docker 容器技术减少了每台服务器重复安装环境的工作量；独立部署在虚拟机中的大数据开发环境，保证了大数据分析系统的正常运行；独立完成数据可视化的实现，并选取了多个工具进行数据可视化的多样化展示。

1.4 论文的组织结构

本文主要分为五个部分进行展开：

第一部分是绪论。首先对本文所研究的房产大数据背景和意义进行阐述，接着对相关研究现状进行总结，分析本文涉及课题研究的必要性；再就是介绍本文所做的主要工作，阐述本文主要完成的为集数据收集、数据分析和数据可视化的一站式系统。

第二部分是相关技术概述。首先介绍的为本文所使用的主要编程语言 Python，接着介绍了分布式爬虫技术，作为本文的数据采集部分，分布式爬虫实现起来需要依赖很多技术，先阐述了底层依赖 Scrapy 框架，通过对 Scrapy 的架构和数据流的阐述，表现其作为网络爬虫技术的诸多优点；接着阐述了 Scrapyd 技术，这也是分布式爬虫的核心；阐述 Docker 容器技术的特点，说明其作为爬虫服务器的环境搭建技术的优点，阐述 Spark 大数据框架以及本文将用到的相关算法，最后介绍了本文所用的数据可视化技术。

第三部分是系统分析和设计。首先是对系统总体结构进行分析和设计，接着对系统业务概述与需求，对基本应用分析和非功能性分析进行了深层次的分析，然后对系统模块进行了设计，通过两张图简洁明了的介绍了各模块功能与联系，最后讲了系统的技术框架。

第四部分是系统实现。首先是对本文所用的主要编程语言 Python、分布式爬虫所依赖的各种环境和工具、大数据分析所用到的框架和算法和数据可视化所用到的平台进行了阐述，包括它们的安装过程以及配置过程。接下来分别从 Python 分布式爬虫的实现、Spark 算法实现和数据可视化的实现三个方面进行阐述。

最后是结论部分，主要包括对课题研究工作和论文内容进行总结，以及提出系统存在的问题和进一步的研究展望。

2 相关技术概述

本章主要介绍基于 Spark 房产大数据分析系统在实现过程中用到的相关技术，以及这些技术的原理。首先介绍了主要编程语言 Python，其次介绍了分布式爬虫相关的原理、Scrapy 框架、Docker 容器技术和 Gerapy 工具，最后就大数据分析相关的 Hadoop 框架、Spark 框架以及数据分析涉及到的 K-Means 算法和 FPGrowth 算法进行了介绍。

2.1 Python 语言

作为本文主要使用的编程语言，Python 编程语言起源于八十年代末和九十年代初，是由 Guido van Rossum 利用业余时间所设计出来的。当然，它的诞生也不是凭空而来，而是借助了诸多编程语言的优点而发展起来的，诸如 SmallTalk、C、C++和其他脚本编程语言。遵循 General Public License 开源协议，所以每个人都可以为其贡献自己的代码，但是仍然主要由 Guido van Rossum 和一个核心团队维护。随着这几年机器学习浪潮和大数据浪潮的来袭，因为 Python 语言其极易上手，且开源的库和工具极其多，实现同样的功能所编写的代码量更少，故 Python 编程语言也变得越来越流行起来。

不管是网络爬虫程序的编写还是数据可视化的实现，Python 都有很多现成的库可以使用，对于本文房产大数据分析系统的实现有较大的帮助。

2.2 分布式爬虫

2.2.1 分布式爬虫介绍

当爬虫程序编写后运行在一台主机上，爬取效率是有限的，为了解决这一问题就需要用到分布式爬虫技术，即将多台主机进行组合，共同完成同一个爬取任务。

2.2.2 Scrapy 架构介绍

分布式爬虫部分采取的是 Scrapy 框架[8]，这是一个使用 Python 语言编写的爬虫框架，也是一个基于 Twisted 异步处理框架。作为一个爬虫框架来说，Scrapy 有着特别清晰的架构，各个模块之间的耦合程度低，有极强的可扩展性，对于各种需求都可以灵活完成。Scrapy 架构图如图 2-1 所示。

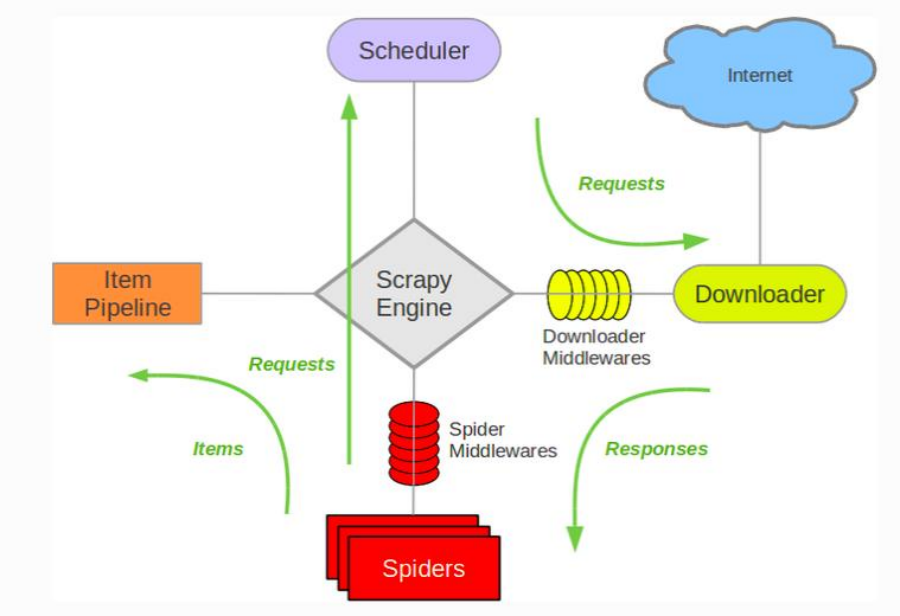

图 0-1 Scrapy 框架结构图

Scrapy 框架可以分成如下几个部分：

Engine：引擎，是 Scrapy 框架的核心部分，可以处理整个系统的数据流以及负责触发事物。

Item：项目，负责对爬虫的爬取结果定义数据结构，Item 对象即为爬虫所爬取的数据。

Scheduler：调度器，爬虫程序发送请求后对其进行接收，然后在队列中将接收的爬虫程序加入，在引擎请求调用的时候提供给引擎。

Downloader：下载器，负责将网页上的内容进行下载，并将该内容返回给爬虫程序中的蜘蛛。

Spider：蜘蛛，主要负责解析爬虫发出的响应，并生成提取出来的结果以及产生新的请求，在蜘蛛内部，定义了所有爬虫爬取的逻辑代码和网页进行解析的规则。

Item Pipeline：项目管道，主要负责的是对爬取下来的数据进行清洗、验证和存储，另外，还处理蜘蛛从网页中爬取的其他项目。

Downloader Middlewares：下载器中间件，主要负责处理引擎与下载器之间发出的请求与响应，是位于引擎和下载器之间的框架，作用类似于钩子。

Spider Middlewares：蜘蛛中间件，主要用来处理蜘蛛输入的响应和输出的结果及新的请求。

2.2.3 Scrapy 工作过程

在 Scrapy 中，各组件之间协同合作，一个完整的工作过程如下：

(1)Engine 先将网站打开，找到程序文件中可以处理该网站的 Spider，接着向该 Spider 请求第一个要爬取的 URL。

(2) Engine 从 Spider 中获取 URL 连接，这个链接就是要爬取的网站 URL，然后就通过 Scheduler 以 Request 响应头的形式调度。

(3) Engine 向 Scheduler 请求接下来要继续爬取的 URL。

(4) Scheduler 返回一个将要爬取的新的 URL 给 Engine, Engine 通过 Downloader Middlewares 将 URL 进行调度，最后通过 Downloader 进行下载。

(5)等待爬虫程序下载完所有页面后，最后通过 Downloader Middlewares 将 Response 回应头发送给 Engine。

(6) Engine 从 Downloader 中接收到 Response 回应头, 然后就将 Reaponse 通过 Spider Middlewares 发送给 Spider。

(7) Spider 处理 Response 回应头后，将爬取到的 Item 及新的 Request 请求一起返回给 Engine。

(8) Spider 返回的 Item 被 Engine 传递给 Item Pipeline，将新的 Request 给 Scheduler。

(9)重复第(2)步到第(8)步，直到 Scheduler 中没有更多的 Request， Engine 关闭该网站，爬取结束。

Scrapy 中不同的组件完成不同的工作并支持异步处理，其可通过多个组件的相互协作，最大限度地利用了网络带宽，大大提高了数据爬取和处理的效率。

2.2.4 Scrapyd 分布式爬虫部署

Scrapy 爬虫爬取程序编写后，需要一个工具进行部署，这时就需要 Scrapyd 程序了，它提供很多 HTTP 接口，可以部署、启动、停止、删除爬虫程序，简直就是 scrapy 的得力助手。Scrapyd 支持版本管理，同时还可以管理多个爬虫任务，利用它可以非常方便地完成 Scrapy 爬虫项目的部署任务调度。

2.2.5 Docker 容器技术

Docker 容器技术是最近十分流行的生产环境部署工具，是一个开源的引擎，每个开发者都可以快速入门并在自己的服务器中进行实战。一直以来，广大开发者们都被频繁的生产环境搭建所困扰，每购买新的服务器后，都要重新部署一遍开发或者生产所依赖的环境，即使有的服务器提供商可以提供镜像服务，依然无法满足很多定制化的需求，这时候 Docker 技术就应运而生。当然，它不仅仅可以轻松快捷地部署环境，启动速度

也相当快，可以在秒级实现，比很多传统的虚拟机都要快很多倍。最后要说明的就是其对系统资源的利用率很高，一台服务器上甚至可以同时运行上千个 Docker 容器，消耗也十分低，但是应用的性能却很高。

2.2.6 Gerapy 框架

Gerapy 是一个分布式爬虫管理框架，基于 Scrapyd 和 Scrapyd API 进行分布式爬虫的部署，通过基于 Python 编程语言的 Django 框架和前端流行框架 Vue 进行 web 端的实现。在 Gerapy 管理界面中，可以添加多台主机，添加完成后，这些主机就可以用于分布式爬虫部署，当然也可以在同台主机上部署多个爬虫爬取程序，在爬虫运行期间可以随时停止。

2.3 Hadoop 框架

Hadoop 框架是 Apache 公司旗下一个使用 java 编程语言实现的开源软件框架，在该平台可以开发和运行处理大规模数据，数据规模一般为 PB 以上。现在提到 Hadoop，一般指 Hadoop 框架体系，包括 MapReduce 大型数据集并行处理系统、Yarn 作业调度和集群资源管理框架和 HDFS 分布式文件系统等组件。

Hadoop 框架中有一个用于存储和处理海量结构化数据的开源仓库工具，叫做 Hive。在 Hadoop 中产生的海量数据并不适合直接存储在数据库中，Hive 的作用就是将这些数据直接存储在 Hadoop 的文件系统中，提供一整套类数据库的数据存储和处理机制。

2.4 Spark 框架

Spark 是一个开源的基于内存计算的集群框架[9]，由加州大学柏克莱分校的 AMPLab 实验室所开发。一直以来 Hadoop 都是大数据开发的主要工具，直到 Spark 的出现，打破了 Hadoop 长期霸占的地位。和 Hadoop 基于磁盘运行相比，Spark 可以直接在存储器中进行运算，因此数据还没有写入硬盘中就已经开始分析和运算，最后存入硬盘。Spark 的运算速度约为 Hadoop 的一百多倍，因此非常适合用于大数据和机器学习等对运算速度要求极高的领域中。除此以外，Spark 还有以下优点：

（1）Spark 在迭代计算过程中数据默认是保存在内存中，后续计算直接读取内存中的结果即可。而 Hadoop 每一步计算都是直接将结果存储在磁盘中，后续的计算从磁盘重新读取上次计算结果。基于内存读取数据的速度比磁盘读取的速度高出两个数量级。

（2）Spark 在实际执行任务前，将计算步骤根据依赖关系形成 DAG 图（向无环图），在执行过程中会根据 DAG 图的顺序来执行，这个过程还会对 DAG 进行计算路径的优化，

大大减少了 I/O 读取操作。而 Hadoop 需要手动或者借助 Oozie 等工具来处理这些步骤之间的关系。

（3）Spark 可以支持 Scala、Python、R、Java 等编程语言。而 Spark 中的核心编程语言 Scala，其本身就是一种高效率和可扩展的语言，让开发者可以用更简短的代码处理更为复杂的工作。

（4）Spark 还提供了一站式的大数据技术栈，包含了以下部分：提供基于内存计算框架的 Spark Core、用于实时计算领域的 Spark Streaming、用于结构化查询的 Spark SQL、用于机器学习的 MLlib、用于图计算的 GraphX 和用于数学计算的 SparkR。所以针对大数据处理的任何一场景，Spark 都为你提供了相应的组件。

Spark 提供了本地 Local 运行模式，用来学习和测试，对于集群部署模式，Spark 能够以 YARN、Mesos 和自身提供的 Standalone 作为资源管理调度框架来执行作业。对于数据源，Spark 能够读取 HDFS、Cassandra、HBase、S3、Alluxio 等数据源数据。Spark BDAS 以 Spark Core 分布式计算引擎为核心，在 Spark Core 之上扩展了用于结构化查询的 Spark SQL、用于实时数据处理的 Spark Streaming、用于机器学习的 MLlib、用于图计算的 GraphX 和用于统计分析的 SparkR 等。

2.5 k-means 算法

作为聚类算法中的一种，k-means 算法是很典型的基于距离的聚类算法。聚类，顾名思义就是根据相似性对数据进行划分，将具有较高相似度的数据对象归为同一类簇，将具有较高异同度的数据对象划分为不同的类簇。聚类和分类不用，聚类过程为无监督过程，每一个处理结果都是在没有设定分类规则的情况下进行的，而分类则是先规定好分类原则，再进行数据的处理。

k-means 算法中的 k 代表类簇个数，means 代表类簇内数据对象的均值，因此 k-means 算法又被称为 k-均值算法。在 k-means 算法中，每两个数据对象之间的距离计算方法为欧氏距离法。

2.6 FPGrowth 算法

FPGrowth 算法是基于 Apriori 原理实现的，通过将数据集存储在 FP（Frequent Pattern）树上后，可以找到频繁出现的项集，但是并不能发现数据之间的关联规则。FPGrowth 算法通常需要对数据库中的所有数据进行两次扫描。其中算法发现频繁项集的过程是先构建 FP 树，然后从 FP 树中挖掘频繁项集。过程图如图 2-2 所示。

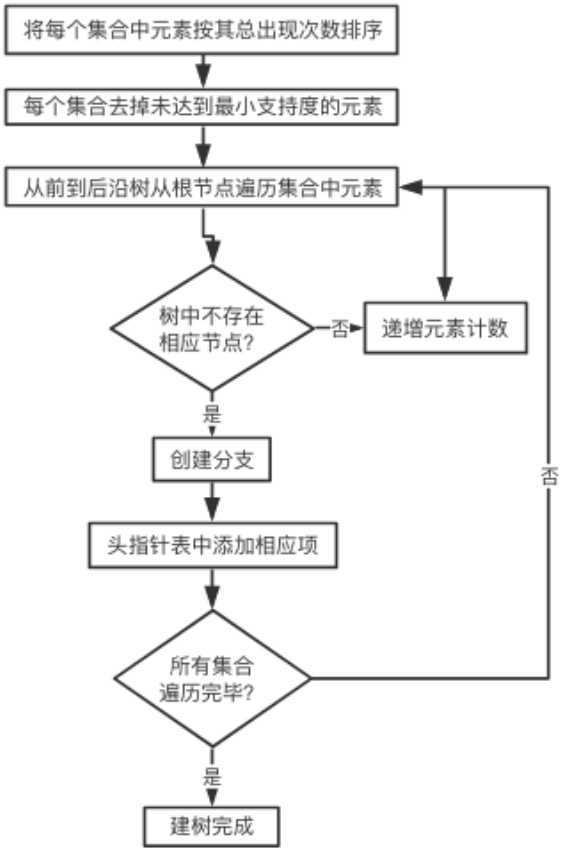

图 0-2 构建 FP 过程

从 FP 树挖掘频繁项集过程如图 2-3 所示。

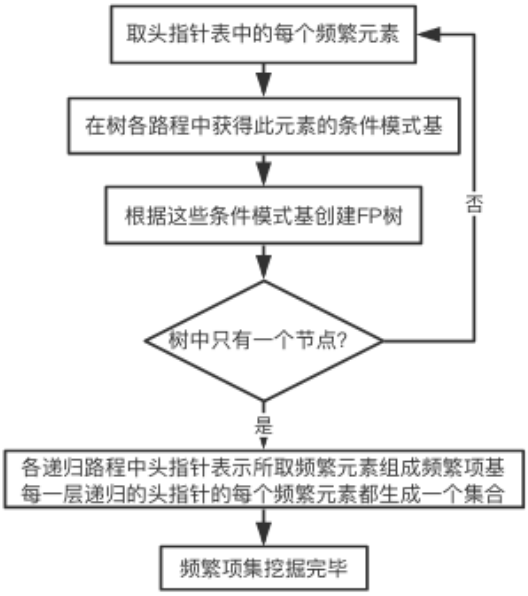

图 0-3 从 FP 树挖掘频繁项集过程

每个项集出现的频率都会被 FP 树所存储，树中的路径即为每一个项集，在这个方式上同搜索树是不同的，其中每一个元素项可以在一棵 FP 树中出现很多次。只有当集合与集合之间出现完全不同的情况的时候，FP 树才会产生分叉。在每一个 FP 树节点上，给出每个集合中的单个元素个数，以及该元素在所有元素序列中的出现次数，路径会给出该序列的出现次数。相似项之间的链接即节点链接，可以用在快速发现每两个相似项之间的位置。因此，为了构建一个完整的 FP 树，需要对所有数据集进行两次扫描。第一次就是对所有元素项的出现次数进行计数，并且会统计每一个元素项出现的频率。第二遍仅仅扫描出哪些为频繁元素。

2.7 数据可视化

数据可视化有很多工具可以使用，下面将讲解常用的几个技术：

matplotlib 是一个基于 Python 语言编写的数据可视化处理库，主要用于绘制 2D 的图形，在安装特定的工具包后，同样可以绘制 3D 的模型或者基于地图的热力图表。因为其强大的功能，而且有着极其多的扩展工具包，在数据分析领域的地位也越来越高。

Tableau 是一款数据可视化的工具，能帮助人们查看和理解数据，支持众多的可视化图表，对于数据源的支持也是多种多样的，不管是最常见的 Excel 还是各种各样的数据库都可以直接导入数据，对于在服务器上数据，同样也可以进行连接服务器并读取[10]。

Plotly 是一个用于做分析和可视化的在线平台，其功能强大，可与多个主流绘图软件对接，还可像 Excel 实现交互制表，图表种类齐全且可以在线分享。

2.8 本章小结

主要介绍了本文所涉及到的相关技术，包括 Python 编程语言、分布式爬虫、Spark 框架、k-means 算法和数据可视化技术，为后续章节打好理论基础。

3 系统分析和设计

房产大数据分析系统是一个支持网络数据抓取、分布式爬虫管理以及数据分析的平台。为了解决原来各个房产交易网站信息分布较为分散，以及无法被购房者及城市管理者们有效利用的问题，随着房产交易数量以及房产交易平台的不断增，需要针对不同房产交易平台的不同网页架构定制不同的网络爬虫，才能有效进行房产交易信息爬取，并对所获得的数据进行数据分析和可视化。本章将从系统业务概述与需求、系统技术框架和系统模块划分几个方面进行阐述。

3.1 系统业务概述与需求

作为我国的经济命脉，房地产市场的稳定发展对国家经济也有着深远的影响，房产交易也因此成为了在国家发展过程中非常重要的一环。购房者以及城市管理者们根据房源的相关信息就可以了解到真实的地区分布和价格趋势等信息，这样有利于房地产交易市场更加健康平稳的发展。传统的房产数据分析主要被一些房地产行业的巨头垄断，一般仅供公司内部使用，即使能够买到相关的数据分析服务，价格也是十分昂贵的。目前市场上也没有从数据获取到数据分析的一体化平台，为了解决这一需求，本文将实现房产数据分析从原始数据获取到数据可视化的展示。

3.2 基本应用分析

网络爬虫部分应该使用方便快捷的爬虫框架，精简爬虫程序的编写，专注于网络爬虫功能的实现。对于网络爬虫爬取下来的数据，既能保存到数据库当中，又能保存为各种表格格式；对于分布式爬虫的实现应当可以在多个主机上进行多个爬虫程序的运行；在数据分析时可选取不止一个算法进行多维度的分析，对于分析的结果能进行可视化的展示；数据可视化的类型应当涵盖柱状图、树状图和热力图等多种图表类型。

3.3 非功能性分析

除了上述的基本应用分析外，还有一些其他非功能性的分析。为了让管理爬虫更加简洁方便，应当使用可视化爬虫管理界面；为了让运行在服务器上的爬虫服务更安全，不被恶意使用，要为爬虫程序使用的端口添加认证功能。

3.4 系统功能模块设计

根据前文的系统需求分析，本文所实现的房产大数据分析系统功能模块设计图 3-1 所示。

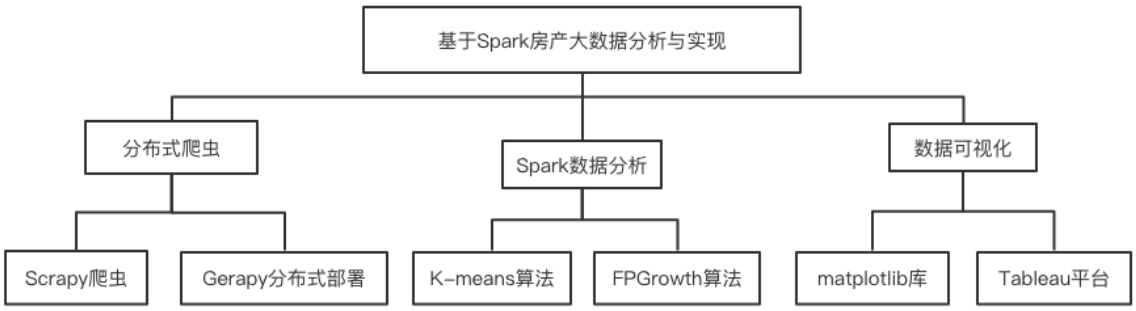

图 0-1 系统功能模块设计图

根据三大功能模块的设计，则房产大数据分析系统大致实现流程如下：

（1）通过爬虫框架进行爬虫程序的编写，再通过将爬虫程序部署到服务器上，可以在多个服务器上同时运行多个爬虫。

（2）将第（1）步爬取下来的数据和 Spark 数据分析算法相结合，得出初步的分析结果。

（3）针对第（2）步的分析结果进行数据的可视化展示。在本模块中，可以结合具体数据进行分析图表的选择，比如柱状图、树状图、漏斗图或者热力图等。

3.5 系统架构设计

由于 Python 编程语言有相当多的开源库和框架用于网络爬虫和数据分析，为了能高效率开发，将其作为本文的主要编程语言；对于爬虫程序的编写，选取了 Scrapy 框架；分布式爬虫的部署选取了 Docker 环境搭建+Scrapyd 爬虫部署+Gerapy 可视化管理；Spark 作为大数据分析的主要技术，并将 Jupyter Notebook 作为开发环境，Python 作为开发语言；matplotliblib 作为数据可视化的库，同时还使用了 Tableau 工具进行数据可视化的实现。根据系统技术框架设计系统架构如图 3-2 所示。

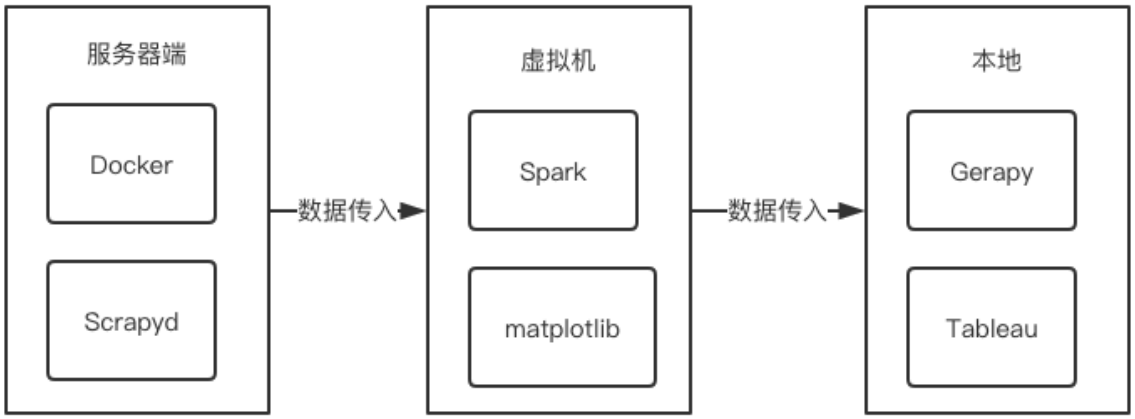

图 0-2 系统架构图

3.6 本章小结

本章主要讲了房产数据分析系统的系统需求分析和设计部分。通过系统业务概述与需求、基本应用分析、非功能性分析、系统模块设计和系统技术框架五个部分进行叙述。

4 系统的实现

本章主要介绍了房产数据分析系统的实现过程，也是本文的重点部分。介绍了从实验硬件配置和实验软件安装开始，到分布式爬虫、大数据分析和数据可视化的实现过程。

4.1 系统开发环境构建

4.1.1 实验硬件配置

本文实验环境采用的是 MacBook Air（13-inch, 2017），处理器为 1.8GHz Inter Core i5,内存为 8GB 1600MHz DDR3，图形卡为 Intel HD Graphics 6000 1536 MB。

分布式爬虫使用的服务器为阿里云学生服务器，CPU 为 1 核，内存为 2GiB，操作系统为 Ubuntu18.04 64 位，带宽为 1Mbps。

Spark 的实验环境为 VMware Fusion 虚拟机，使用版本为 CentOS7.2。

4.1.2 实验软件安装与配置

（1） Python 安装

本文所实现的各种功能的主要编程语言为 Python，所以首先要安装 Python，由于 Python3 是官方推荐的使用版本，Python2 已经逐渐不再被各种依赖库所支持，所以安装的版本为 Python3。在本文所使用的实验环境 Mac 系统中，有多种安装方式，推荐使用 Homebrew 进行安装。Homebrew 是 Mac 平台下强大的包管理工具，安装命令如下：

```
$ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install)"
```

安装完成后就可以使用 brew 命令来安装 Python3，命令如下：

```
$ brew install python3
```

（2） Scrapy 安装

对于本文所使用的实验环境 Mac OS 系统来说，系统是会引用自带 python2.x 库，所以不能删除默认安装的包，但 Scrapy 用 python2.x 来安装会报错，尝试用 python3.x 安装后仍然是报错的，直接安装 scrapy 的办法一直找不到，最后选用了另一种安装方式解决了这个问题，那就是使用 virtualenv 安装，这是一个专门用来为一个应用创建一套单独隔离出来的 Python 运行环境。命令如下：

```
$ sudo pip install virtualenv
```

$ virtualenv scrapyenv

$ cd scrapyenv

$ source bin/activate

$ pip install Scrapy

安装完成后，在终端中输入 Scrapy 检验是否安装成功，成功界面如图 4-1 所示。

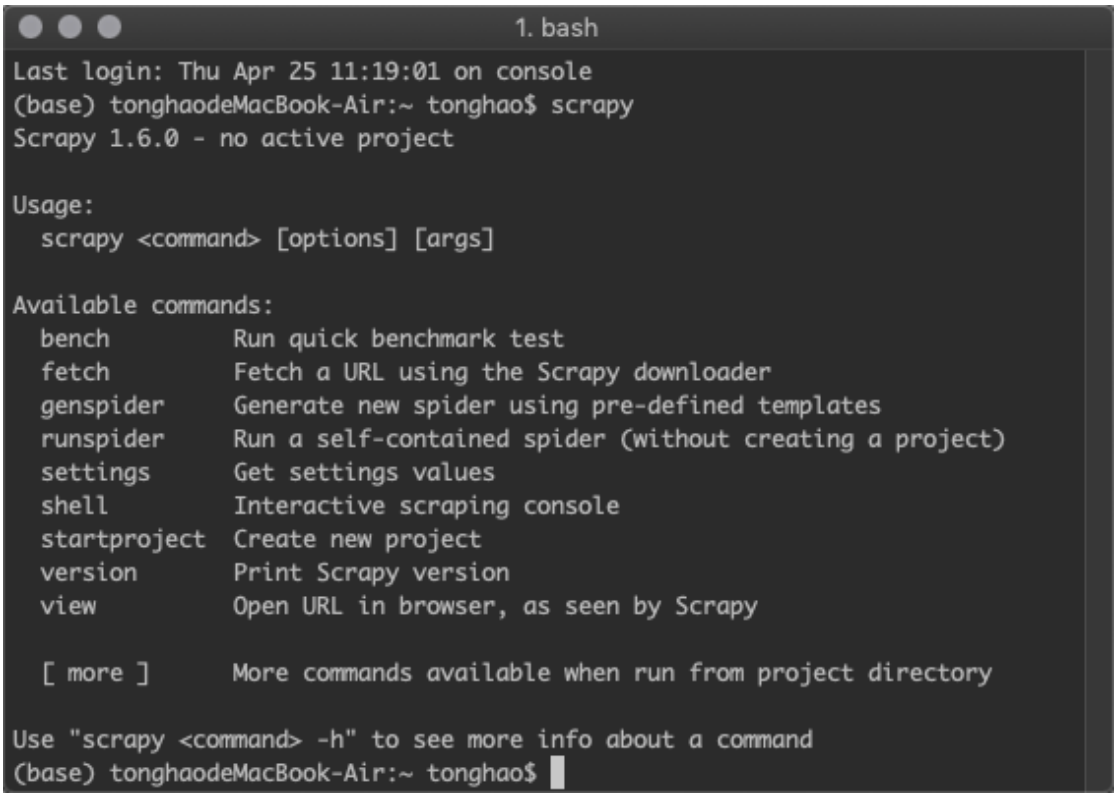

图 0-1 Scrapy 安装成功界面

（3） Scrapyd 安装

本文所实现的部署和运行 Scrapy 项目的工具为 Scrapy，通过 Scrapy，可以将写好的 Scrapy 项目上传到云服务器并通过其提供的 API 来控制爬虫程序的运行。选择 pip 作为安装方式，命令如下：

$ pip install scrapyd

安装完成后，还需要新建一个配置文件 scrapyd.conf，配置文件位置为 /etc/scrapyd，当 Scrapyd 在运行的时候，会自动读取该配置文件，创建文件夹以及新建文件命令如下：

$ sudo mkdir /etc/scrapyd

$ sudo vi /etc/scrapyd/scrapyd.conf

然后将以下内容写入该配置文件中：

```
[scrapyd]
eggs_dir       = eggs
logs_dir       = logs     #日志目录
items_dir      =
jobs_to_keep = 5           #同时运行工作数
dbs_dir        = dbs
max_proc       = 0
max_proc_per_cpu = 4
finished_to_keep = 100
poll_interval = 5.0
bind_address = 127.0.0.1       #运行主机地址
http_port      = 6800        #运行主机的 scrapyd 服务端口号
debug          = off
runner         = scrapyd.runner
application = scrapyd.app.application
launcher       = scrapyd.launcher.Launcher
webroot        = scrapyd.website.Root
[services]
schedule.json         = scrapyd.webservice.Schedule
cancel.json           = scrapyd.webservice.Cancel
addversion.json       = scrapyd.webservice.AddVersion
listprojects.json = scrapyd.webservice.ListProjects
listversions.json = scrapyd.webservice.ListVersions
listspiders.json    = scrapyd.webservice.ListSpiders
delproject.json       = scrapyd.webservice.DeleteProject
delversion.json       = scrapyd.webservice.DeleteVersion
listjobs.json         = scrapyd.webservice.ListJobs
daemonstatus.json = scrapyd.webservice.DaemonStatus
```

Scrapyd 作为一个纯 Python 项目，还需输入以下命令才能让其在后台持续运行：

```
$ (scrapyd > /dev/null &)
```

运行后，就可以打开所部属主机的 6800 端口即可访问 WebUI 了，可以从该界面看到当前 Scrapyd 运行的任务和运行日志等内容，界面如下图 4-2 所示。

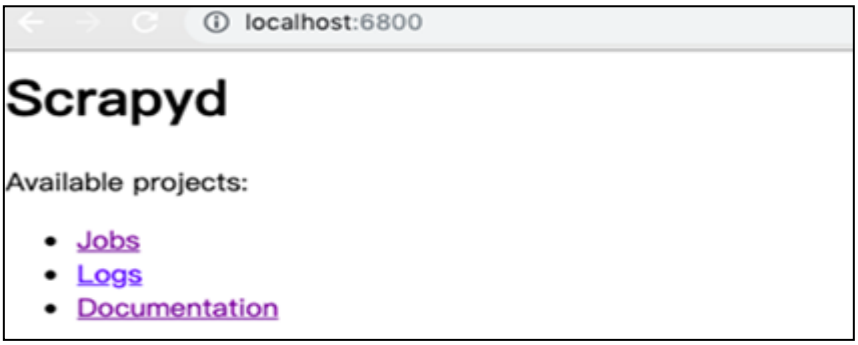

图 0-2 Scrapyd 默认界面

默认的配置完成后，Scrapyd 和它提供的接口是可以公开访问的，为了防止运行在服务器上的 Scrapyd 被他人恶意使用，还需增加一个访问认证功能，具体方法为通过 Nginx 做一个反向代理，首先需要安装 Nginx 服务器，命令如下：

```
sudo apt-get install nginx
```

然后修改 Nginx 中的配置文件 nginx.conf，增加如下配置：

```
http{
server{
listen 6801;
location / {
proxy_pass http://127.0.0.1:6800/;
auth_basic "Restricted";
auth_basic_user_file /etc/nginx/conf.d/.htpasswd;
}
}
}
```

该配置文件中使用的用户名和密码配置放置在/etc/nginx/conf.d 目录下，使用 htpasswd 命令进行创建文件，命令如下：

```
$ htpasswd -c .htpasswd admin
```

然后会提示输入密码，输入两次后就会生成密码文件。配置完成后还需重启一下 Nginx 服务，命令如下：

```
$ sudo nginx -s reload
```

这样就完成了对 Scrapyd 的访问认证功能。

（4） Gerapy 安装

本文所实现的分布式爬虫管理工具为 Gerapy，这是一个 Scrapy 分布式管理模块，可以通过匹配进行安装，安装命令如下：

```
$ pip install gerapy
```

安装完成后可以在 Python 命令行下进行测试，测试过程如下图 4-3 所示。

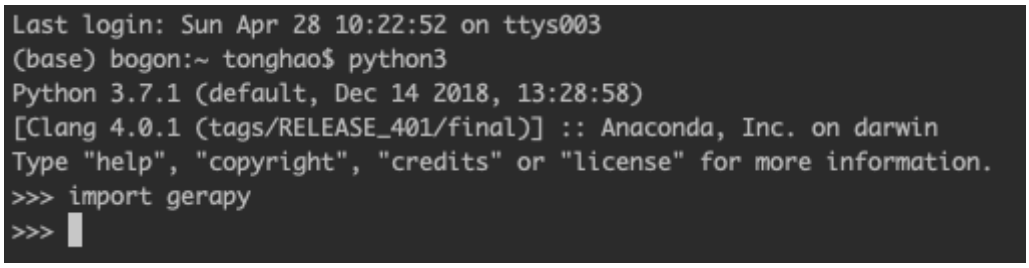

图 0-3 Gerapy 安装成功界面

没有错误报出，则说明该库安装成功。

（5） Hadoop 安装

首先从官网下载 Hadoop，选择的版本为 2.8.5，下载地址如下：

https://mirrors.tuna.tsinghua.edu.cn/apache/hadoop/common/hadoop-2.8.5/hadoop-2.8.5.tar.gz

下载完成后，进行解压缩，到/usr/local/hadoop/目录下，命令如下：

$ tar zxvf /root/download/hadoop-2.8.5.tar.gz -C /usr/local/hadoop/

将以下内容添加到/etc/profile 文件中：

export HADOOP_HOME=/usr/local/hadoop/hadoop-2.8.5

export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/sbin:$HADOOP_HOME/bin

编辑/usr/local/hadoop/hadoop-2.8.5/etc/hadoop/hadoop-env.sh

The java implementation to use. By default, this environment

variable is REQUIRED on ALL platforms except OS X!

export JAVA_HOME=

修改为：

The java implementation to use. By default, this environment

variable is REQUIRED on ALL platforms except OS X!

export JAVA_HOME=/usr/local/java/jdk1.8.0_181

创建 data 目录，命令如下：

$ mkdir /usr/local/hadoop/hadoop-2.8.5/data

编辑/usr/local/hadoop/hadoop-2.8.5/etc/hadoop/core-site.xml：

<configuration>

<property>

<name>fs.defaultFS</name>

```
        <value>hdfs://sparkvm.com: 8020</value>
    </property>
<!-- 指定 hadoop 存储数据的目录 -->
    <property>
        <name>hadoop.tmp.dir</name>
        <value> file:/home/hadoop/tools/data/tmp</value>
    </property>
</configuration>
```

编辑/usr/local/hadoop/hadoop-2.8.5/etc/hadoop/hdfs-site.xml：

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
    <property>
        <name>dfs.namenode.secondary.http-address</name>
        <value>sparkvm.com:8021</value>
  </property>
</configuration>
```

编辑/usr/local/hadoop/hadoop-2.8.5/etc/hadoop/mapred-site.xml：

```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
</configuration>
```

编辑/usr/local/hadoop/hadoop-2.8.5/etc/hadoop/yarn-site.xml

```
<configuration>
    <!-- YARN 的配置文件地址 -->
    <property>
        <name>yarn.resourcemanager.hostname</name>
        <value> sparkvm.com</value>
    </property>
    <!-- 分别指定 MapReduce 的方式 -->
```

```
    <property>
      <name>yarn.nodemanager.aux-services</name>
      <value>mapreduce_shuffle</value>
    </property>
</configuration>
```

初始化 Hadoop HDFS 文件系统，命令如下：

$ hdfs namenode -format

启动 HDFS：

./usr/local/hadoop/hadoop-2.8.5/sbin/start-dfs.sh

启动 YARN:

./usr/local/hadoop/hadoop-2.8.5/sbin/start-yarn.sh

打开 Hadoop HDFS 的管理页面，使用浏览器访问 hdfs 管理界面：

http://localhost:50070

界面如下图 4-4 所示。

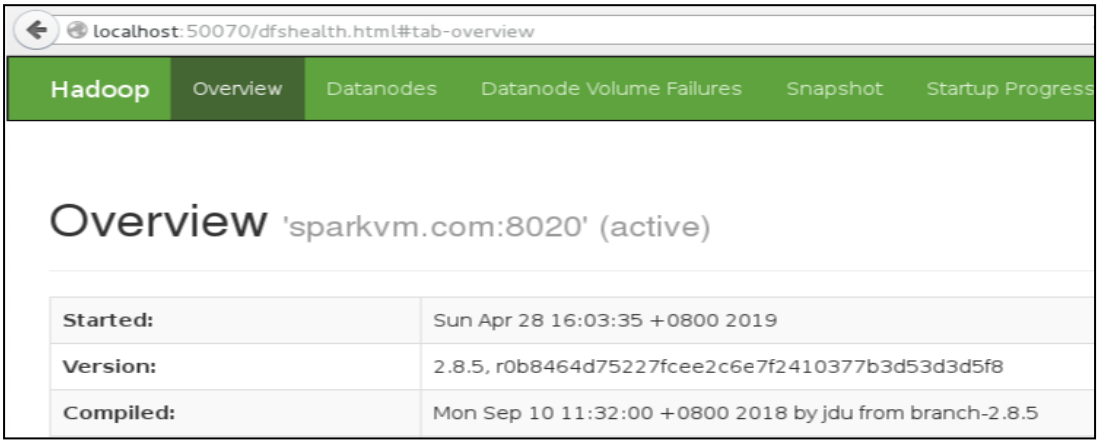

图 0-4 hdfs 管理界面

（6）pyspark 安装

直接使用 pip 进行安装，命令如下：

$ pip install pyspark

编辑/etc/profile 文件，添加下列内容：

export PYSPARK_PYTHON=/usr/local/python/bin/python3.7

直接输入 pyspark 启动，成功界面如下图 4-5 所示。

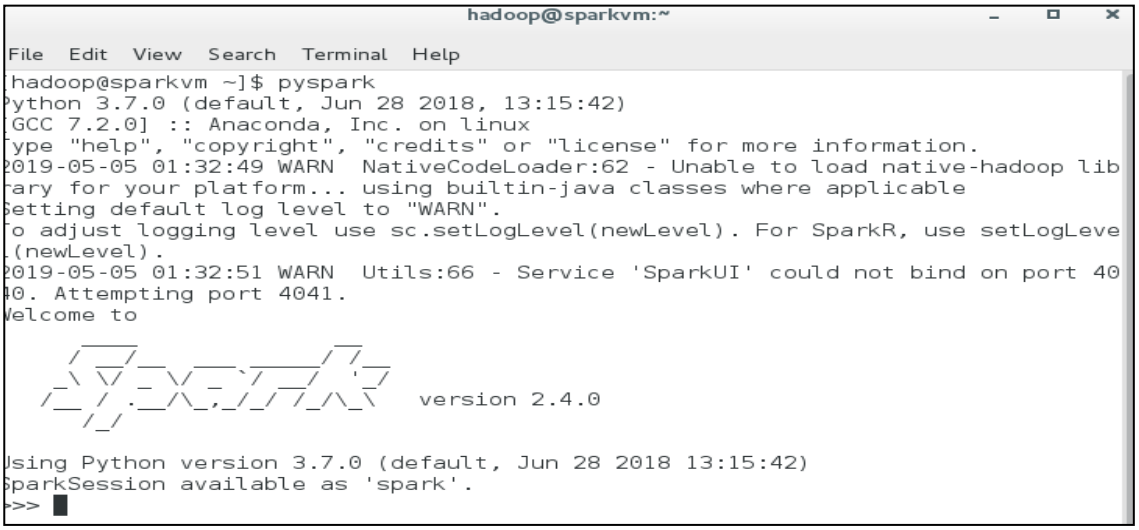

图 0-5 pyspark 启动成功界面

4.2 分布式爬虫的实现

4.2.1 Scrapy 爬虫的实现

本文所实现的网络爬虫技术框架为 Scrapy，首先要创建一个新的 scrapy 项目：

$ scrapy startproject MaitianSpider

命令中 MaitianSpider 即为项目名称，创建完成后就可以出现一个名为 MaitianSpider 的文件夹，该文件夹的大致目录如下：

```
MaitianSpider/
    scrapy.cfg  #项目的配置文件
    MaitianSpider/ #项目的 Python 模块，将会从这里引用代码
        __init__.py
        items.py  #项目的目标文件
        pipelines.py  #项目的管道文件
        settings.py #存储爬虫代码目录
        spiders/
            __init__.py
            ...
```

项目创建好后就可以确认要爬取的网站，选取的房产交易网站为麦田房产，

网址为：http://fz.maitian.cn/esfall ，网站界面如图 4-6 所示。

图 0-6 所爬取网站界面

需要爬取的信息为：房源的区域信息、详细地址、房源介绍、房源面积、房源单价和房源总价。

通过分析网站代码结构，定位到各爬取信息的 css，核心代码如下：

```
for house in response.css('div.list_title'):
        #定位到各爬取信息的 css
        location = house.css('p.house_info::text').extract()
        property = house.css('h1 a::text').extract()
        area = house.css('div.the_area ol strong span::text').extract()
        price = house.css('div.the_price ol strong span::text').extract()
        unit_price = house.css('div.the_price ol::text').extract()
        yield {
            'location':location,
            'property':property,
            'area':area,
            'price':price,
            'unit_price':unit_price,
        }
```

然后设置爬虫的逐页爬取功能，同样采用 CSS 定位功能，核心代码如下：

```
    next_url = response.css('#paging > a:nth-child(9)::attr(href)').extract_first()
     if next_url:
        next_url = response.urljoin(next_url)
```

yield scrapy.Request(next_url, callback=self.parse)

爬取下来的部分初始数据如图 4-7 所示。

location	property	area	price	unit_price
[晋安区] 国货路,王庄, 世欧王庄A区	电梯高层，东头户型，南北通透，带3个阳台	163.56㎡	585	35766元/㎡
[金山] 金山公园,浦上大道（金山万达）, 中天金海岸四期天骄苑浅水湾	金海岸四期，刚需3房，东南朝向，装修好的，带入户	93㎡	248	26666元/㎡
[金山] 浦上大道（金山万达）,闽江大道, 金山桔园2期景园	万达对面，电梯高层，南北通透，进门带入户花园，装修保持新。	117.62㎡	280	23805元/㎡
[台江区] 中亭街,台江万达, 光明港苑	光明港苑 2室2厅 230万元	90㎡	230	25555元/㎡
[金山] 浦上大道（金山万达）,金山大道, 钱隆金山	乐购对面，人车分流，环境好，标准2.5房，居家装修	79.06㎡	220	27826元/㎡
[台江区] 五一广场,茶亭, 群众新村	五一广场旁，框架3．5房，西南北通透，采光好。	111.2㎡	330	29676元/㎡

图 0-7 爬取初始数据

通过简单的 Excel 进行原始数据的处理，将区域单独设置成一列，并将一些特殊符号去除，单位去除，仅在表头处注明，添加一个数据表名称，更能直观的显示数据。处理后的部分二手房数据如图 4-8 所示。

区域	位置	楼盘名	面积(㎡)	单价(元/㎡)	总价(万元)
闽侯	南屿,福州高新区, 阳光城霸丽湾	阳光城霸丽湾 2.5室2厅 175万元	88	19886	175
金山	闽江大道,金山大道, 保利西江林语	保利西江林语,高层看双江！东端头！标准4房！带大,	104	25961	270
仓山区	南江滨,白湖亭, 闽江世纪城A区	闽江世纪城，电梯高层，看江，看湖，南北通透	123.18	31336	386
仓山区	南江滨,白湖亭, 闽江世纪城A区	前后视野很好，3房2卫，南北通透，大阳台	107	29719	318
鼓楼区	左海西湖,西江滨, 西风小区	刚需2．5房，普通居家装修，南北通透。户型方正，且	88	22727	200
鼓楼区	三坊七巷,乌山路, 杨南新城	湖滨十八 左海西湖边 电梯三房 南北通透 带车位	103.4	34816	360
仓山区	南江滨,白湖亭, 闽江世纪城B区	闽江世纪城，方正三房，二排好位置，纯毛坯如你装修	129.1	24012	310

图 0-8 简单处理后的数据

4.2.2 Gerapy 分布式爬虫的部署

为了满足同时爬取多个城市的分布式爬取需求，使用了 gerapy 工具，使用方法类似 scrapy，首先创建一个新的项目，命令如下：

$ gerapy init

生成的的 gerapy 文件夹就会出现在当前目录，进入 gerapy 文件夹后，可以看到一个空的 projects 文件夹。此时要进行数据库的初始化，命令如下：

$ gerapy migrate

这样做的目的是为了将各个主机配置信息和部署版本等信息保存到生成的 SQLite 数据库中。

启动 Gerapy 服务，命令如下：

$ gerapy runserver

由于 Gerapy 默认在 8000 端口开启，打开浏览器，输入 http://127.0.0.1:8000，即可进入 Gerapy 的管理界面，主要提供了主机管理和项目管理的功能，界面如下图 4-9 所示。

图 0-9 Gerapy 管理界面

可以点击“创建”按钮进行主机的增加，如要设置一个自定义主机名称、IP 地址和 Scrapyd 端口号（默认为 6800 端口），如果是有认证服务的服务器还需要输入用户名和密码。

将 scrapy 项目直接拖到 Gerapy 目录下的 projects 文件夹中，进入 Gerapy 中的项目管理界面，刷新即可看到名称为 MaitianSpider 的爬虫项目，如图 4-10 所示。

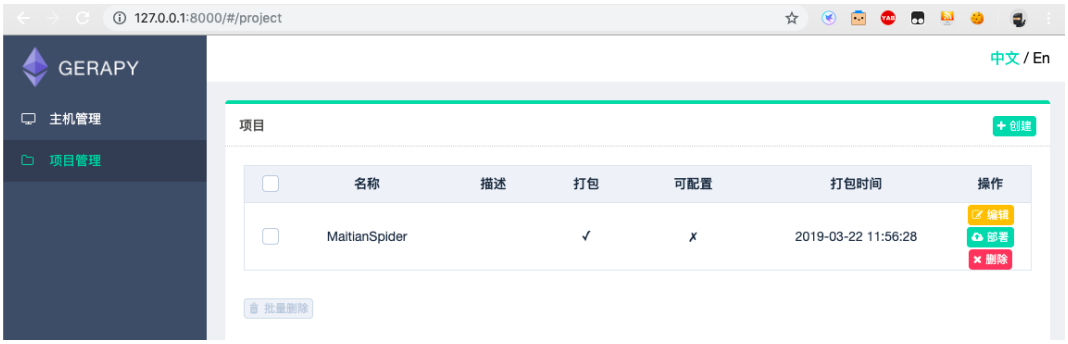

图 0-10 Gerapy 项目管理界面

使用部署功能，就可以选择服务器进行爬虫的部署。可以选择一个服务器，也可以选多个服务器同时部署，Gerapy 爬虫部署界面如图 4-11 所示。

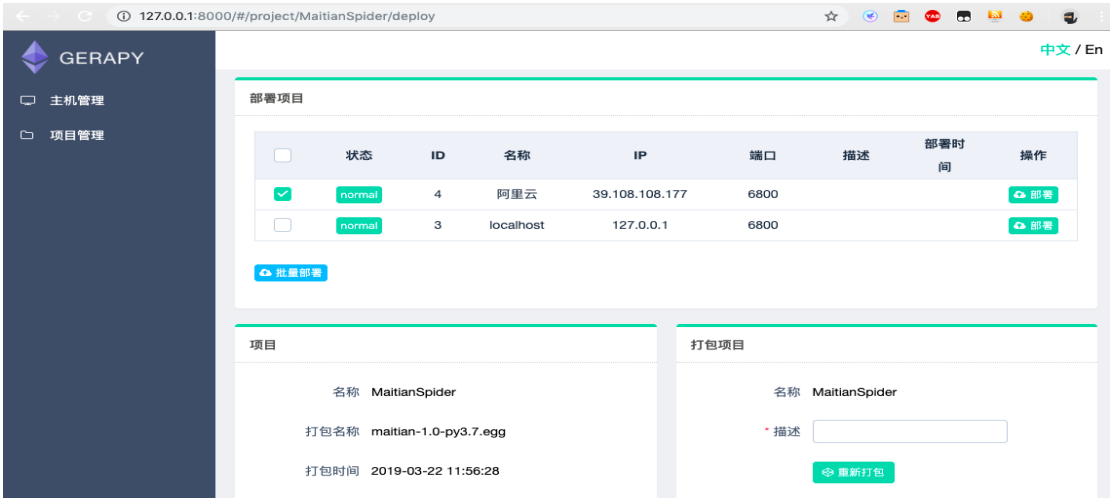

图 0-11 Gerapy 爬虫部署界面

Gerapy 爬虫运行界面如图 4-12 所示。

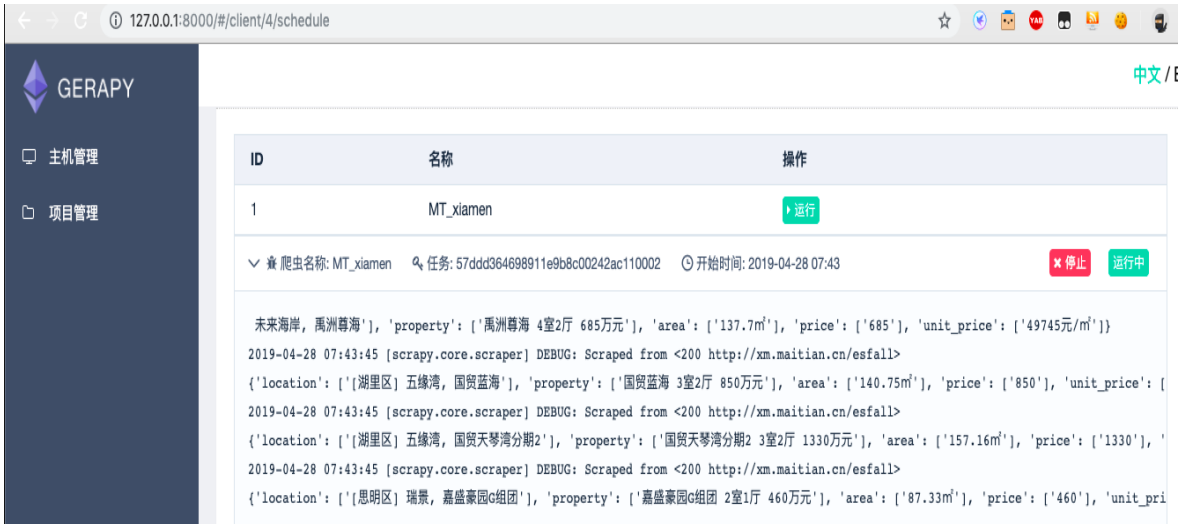

图 0-12 Gerapy 爬虫运行界面

通过主机管理界面的调度功能，就可以实现爬虫在该主机上的爬取。爬取过程中也可以随时停止。

爬取下来的数据将自动保存为设置好的格式。为了便于分析和可视化，选择了 csv（逗号分隔符文件）格式，所爬取网站福州二手房数据大概为 5372 条。

4.3 房产空间分布可视化分析

将爬取下来的数据直接与 Tableau 平台结合，便可得出简单的数据可视化图表，如下图 4-13 所示。

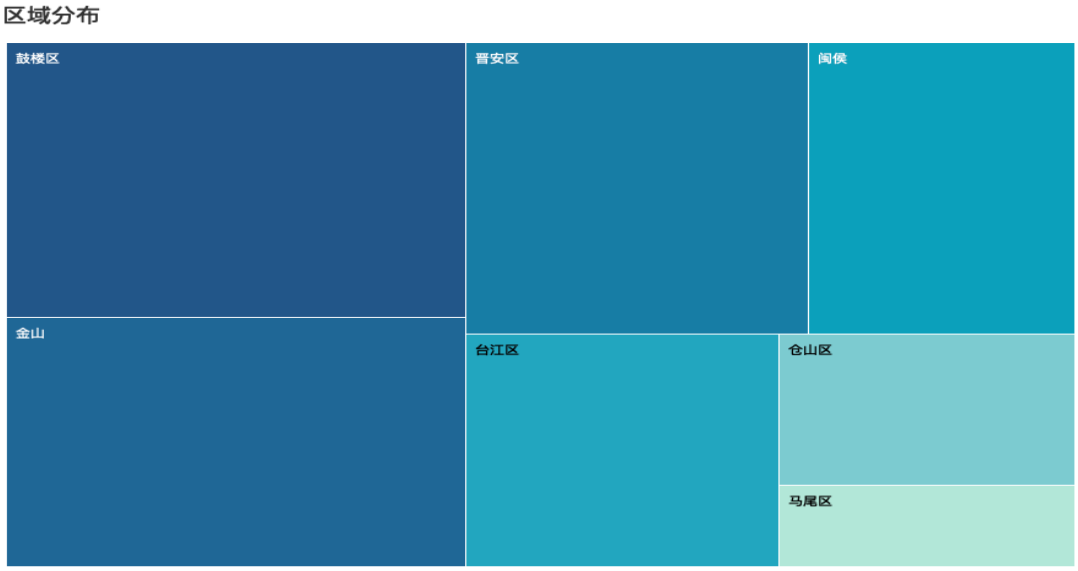

图 0-13 区域分布树状图

该图的矩形颜色和面积均代表房源数量，即矩形面积越大，颜色越深即说明该矩形所代表的区的二手房源越多。也可根据数据做出热力图，如图-14 和图-15 所示。

图 0-14 福州二手房价格热力图

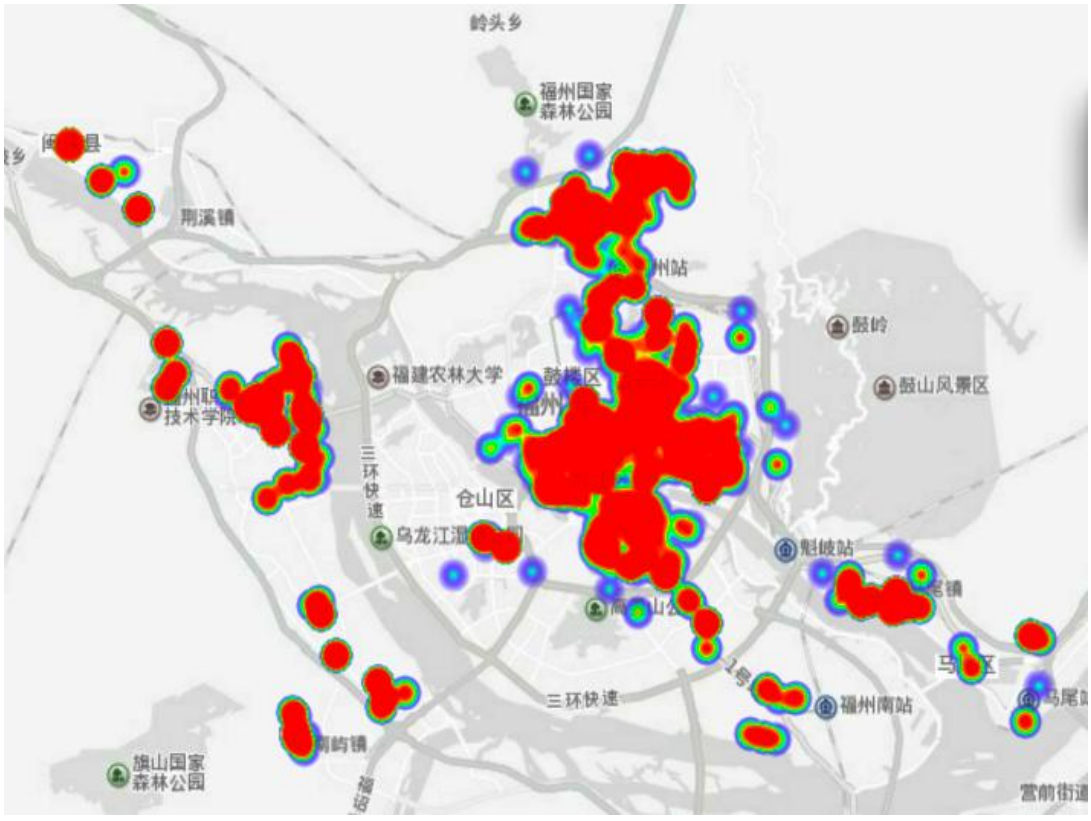

图 0-15 福州二手房价格热力图

鼓楼区为房源数量最多的区域，相反，马尾区为房源数量最少的区域。由于所爬取网站没有长乐区和福州其他县的业务，所以在图上并没有显示。

通过对福州地区的二手房交易数据进行分析，发现鼓楼区为福州市区最热门的区域，说明广大购房者更偏爱本区域，再通过查阅福州市区相关资料，鼓楼区作为福州市区的老城区同时也是市中心所在区域，各种配套设施相对比较完善，交通也是特别便利，地铁线路贯穿其中，政府办公部门也大都在此区域。所以不管是福州二手房数量热力图和价格热力图，鼓楼区都是最显眼的区域。这个分析结果同时也能给城市管理者一定的辅助管理功能，对于城市规划以及建设，可以特别注意热门的区域，车流量高峰期也应当在热门区域安排更多交通警察指挥交通，防止发生大规模道路瘫痪，对于后期地铁线路的持续建设，也应当多穿插热门区域，缓解这些区域的交通压力，让市民享受更便利更舒适的城市生活。

4.4 基于 FPGrowth 算法的房源关键词分析

本文所实现的 Spark 环境在 VMware Fushion 中，首先进行的是 Hadoop 和 Spark 的安装及配置。

启动 Hadoop：

$ hadoop-start-all.sh

启动 Spark：

$ spark-start-all.sh

Hadoop 和 Spark 启动成功界面如图 4-16 所示。

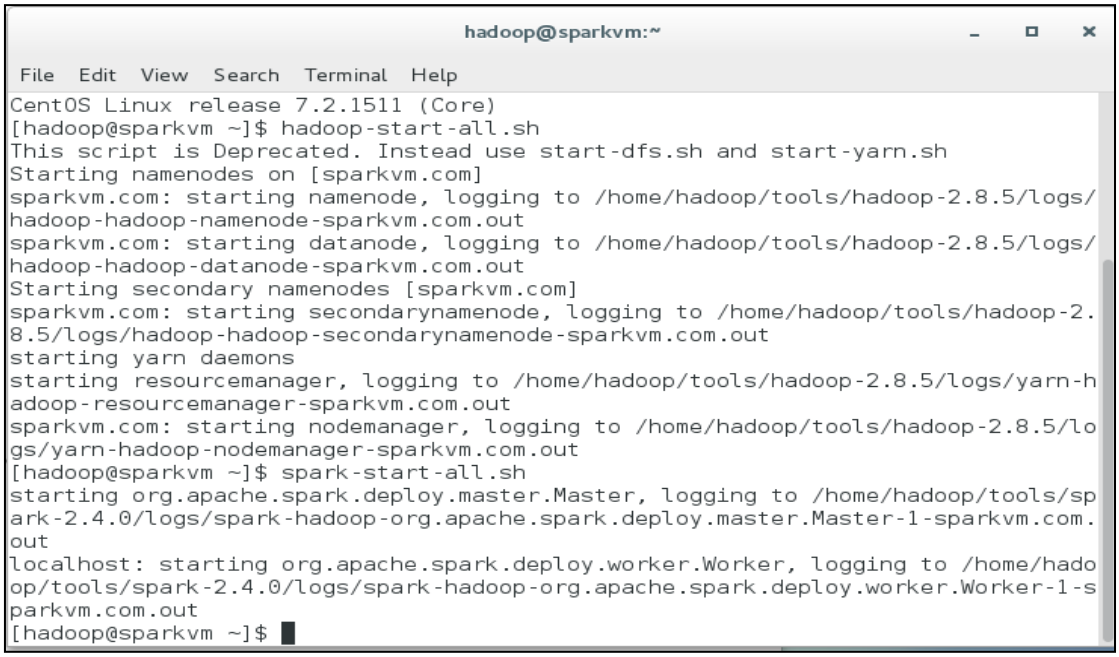

图 0-16 Hadoop 和 Spark 启动成功界面

启动开发环境 Jupyter Notebook，首先进行的是 FPGrowth 算法算法的实现，核心代码如下：

```
#读取需要分析的文件
rdd2 = sc.textFile("hdfs://192.168.72.152:8020/data/text/Fuzhou.csv")
#引入 jieba 中文分词库，并通过正则表达式库 re 进行特殊符号匹配
import jieba,re
def cutChinaword(line):
r = r'[（），。 、 □<>|“”：《》 ]'
line = line.strip()
line = re.sub(r, '', line)
s = set()
for w in jieba.cut(line):
    if len(w) > 1:
        s.add(w)
return list(s)
rdd3 = rdd2.map(lambda line :line).distinct().map(lambda line :cutChinaword(line))
#FPGrowth 的实现，并打印匹配关键词大于一个的数据
model = fmp.FPGrowth.train(rdd3, 0.001,2)
res = model.freqItemsets().collect()
for r in res:
    if len(r.items) > 1:
        print(r)
```

运行结果为：

```
FreqItemset(items=['蓝湾', '上下'], freq=12)
FreqItemset(items=['蓝湾', '上下', '北江'], freq=12)
FreqItemset(items=['蓝湾', '上下', '北江', '南北'], freq=8)
FreqItemset(items=['蓝湾', '上下', '北江', '南北', '通透'], freq=8)
FreqItemset(items=['蓝湾', '上下', '北江', 'CBD'], freq=12)
...
```

进行关联分析，按照关联度输出关联词：

```
res2 = sorted(res,key=lambda r : r.freq,reverse=True)
for r in res2:
    if len(r.items) > 1:
```

```
            print(r)
```

运行结果为：

```
FreqItemset(items=['金山', '大道'], freq=1089)
FreqItemset(items=['南北', '通透'], freq=1058)
FreqItemset(items=['高层', '电梯'], freq=834)
FreqItemset(items=['万达', '大道'], freq=750)
FreqItemset(items=['万达', '金山'], freq=666)
...
```

通过定制关键词进行关联分析，以“公园”为例：

```
for r in res2:
    if "公园" in r.items:
        print(r)
```

运行结果为：

```
FreqItemset(items=['公园'], freq=1059)
FreqItemset(items=['公园', '金山'], freq=494)
FreqItemset(items=['公园', '大道'], freq=489)
FreqItemset(items=['公园', '金山', '大道'], freq=487)
FreqItemset(items=['温泉', '公园'], freq=413)
...
```

同样可以为关键词设置特定位置，将“万达”设为第二关键词举例：

```
for r in res2:
    w = r.items
    if len(w) > 1:
        if w[1].find(str("万达"))>-1:
            print(r)
```

运行结果为：

```
FreqItemset(items=['浦上', '万达'], freq=658)
FreqItemset(items=['浦上', '万达', '金山'], freq=658)
FreqItemset(items=['浦上', '万达', '金山', '大道'], freq=658)
FreqItemset(items=['浦上', '万达', '大道'], freq=658)
FreqItemset(items=['浦上', '万达', '公园'], freq=290)
```

通过 Spark 和 FPGrowth 算法的结合，可以方便快捷的看出各关键词之间的联系，如上图运行结果所示，“公园”关键词和“金山大道”、“五四路”和“鼓楼区”关联度更高，因此当购房者对公园有需求时，可以优先考虑这些关键词附

近的房源信息。同样，当购房者对万达广场有需求时，也可直接看出与其相关联的关键词，进而重点关注关键词附近的房源信息。

4.5 基于 k-means 算法房源分布分析

核心代码如下：

```
conf = pyspark.SparkConf()
sc = pyspark.SparkContext()
cw = pd.read_csv(os.getcwd() + "/data/Fuzhou.csv",encoding='utf-8',engine = 'python')
#设置要进行分析的两列数据，分别为面积和价格
dataSet = [(m1,m2)for m1,m2 in zip(cw["area"],cw["price"])]
rdd = sc.parallelize(dataSet)
#聚类分析
clusters = cl.KMeans.train(rdd, 5, maxIterations=10, initializationMode="random")
#聚类的中心点
clusters.clusterCenters
#保存模型，predict 用训练好的模型，去进行聚类
cw["clf"] = clusters.predict(rdd).collect()
#使用 matplotlib 库进行聚类分析的可视化展示
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(14,10))
ax = fig.gca()
ax.grid(alpha= 1)
ax.scatter(cw["area"],cw["price"],c=cw["clf"])
```

代码运行截图 4-17 如下。

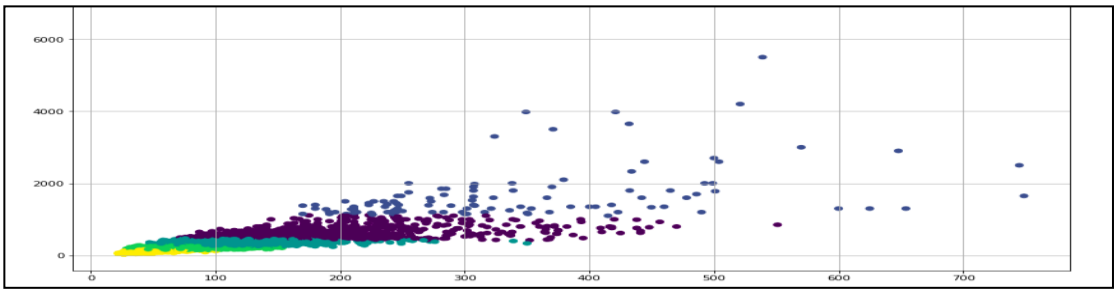

图 0-15 k-means 算法实现

```
#去除 clf 列，只分析 area 和 price
fig = plt.figure(figsize=(14,10))
ax = fig.gca()
ax.grid(alpha= 1)
ax.scatter(cw2["area"],cw2["price"])
```

代码运行结果如图 4-18 所示。

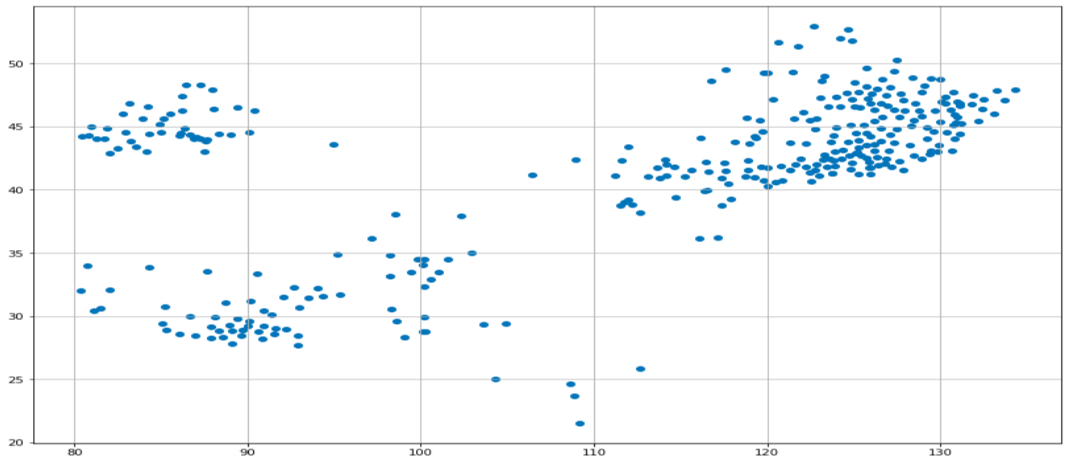

图 0-16 k-means 算法去除 clf 列运行结果

由上图可以看出，福州地区的二手房交易主要遍布在 110-130 平方米的房子，80-100 平方米的房子虽然在图上有小部分聚集的区域，但不是最主要的聚合点。因此，如果有 110-130 平方米的房源想要出售的话，将会十分抢手，同样，如果想要购买这一区间的房子，因为房源多的原因，选择也会多种多样。

4.6 本章小结

本章主要讲了房产数据分析系统的实现部分，介绍了从实验硬件配置和实验软件安装开始，然后介绍了从分布式爬虫、大数据分析到数据可视化的实现过程。

结论

本文以分布式爬虫、房产数据分析和数据可视化为主题，介绍了国内外的研究现状和相关技术，在 Scrapy 爬虫框架的基础上，结合了分布式爬虫部署的知识，最后和 Spark 大数据框架及相关算法相结合，实现了从数据获取到数据分析的完整过程。具体工作如下：

（1）了解房产行业基础知识，对相关购房政策进行解读，调研目前流行的房产交易网站并分析其网站结构，为接下来的爬虫程序编写打下基础。

（2）实现了 Scrapy 爬虫，接着实现了本文的核心模块之一，即分布式爬虫的部署，将 Docker 容器技术和服务器环境搭建相结合，减少了每启用新服务器安装环境的步骤。研究 Hadoop 和 Spark 大数据框架体系，深入学习大数据开发流程。学习使用数据可视化工具 Tableau，了解多种数据可视化图表的作用，实现了本文所要求的各种可视化图表。

（3）学习 FPGrowth 关联算法和 K-means 算法的基础知识，思考它们和房产数据的联系，进而实现这些算法与 Spark 框架有效结合。

本文通过分布式爬虫和 Spark 大数据分析框架相结合的方式，提供了一整套的从数据获取到数据分析的完整解决方案，但由于水平有限和研究时间的限制，还有许多功能不够完善，需要在未来的工作中进行改进：

（1）分布式爬虫部分仍不够完善，对于一些需要验证码登录的网站，还没有完美解决的方案，这也是未来重要的研究方向。

（2）数据库的使用问题。由于为了分析方便，本系统使用的为逗号分隔符文件 CSV，但是未来仍然会有数据库的需求，本系统虽然没有实现，但是已经有了思路，数据库也已大致确认为 MongoDB。

（3）数据深度不足。本文使用的福州二手房数据仅仅只有一个网站的数据，虽然分布式爬虫实现了，但也只是做到同一网站的不同城市，由于时间问题来不及解决其他房产交易网站的反爬虫措施，未来将对其他网站进行爬取策略分析。

参考文献

[1] 车江涛. 基于 Spark 的智慧城市房价评估系统的研究与实现[D]. 西安:西安电子科技大学计算机学院应用技术专业, 2017.

[2] 孙哲.北京二手房波动探因[J].投资北京,2016(10):14-20.

[3] 黄明宇,夏典.合肥市二手房价多元线性回归预测模型[J].合作经济与科技,2019(09):80-82.

[4] 李文栋.基于 Spark 的大数据挖掘技术的研究与实现[D].济南:山东大学软件学院软件工程系,2015.

[5] 许礼捷.基于 Spark 的大数据处理平台的搭建与研究[J].电脑知识与技术:学术交流,2016,12(15):14-16.

[6] 逄菲.基于 Python 的分布式网络爬虫系统的设计与实现[J].电子技术与软件工程 .2018(23):6-7.

[7] 李乔宇,尚明华,王富军,等.基于 Scrapy 的农业网络数据爬取[J].山东农业科学,2018,50(1):142-147.

[8] 李代祎,谢丽艳,钱慎一,等.基于 Scrapy 的分布式爬虫系统的设计与实现[J]. 湖北民族学院学报(自然科学版), 2017 (3):16-17.

[9] Gita Donkal,Gyanendra K.A multimodal fusion based framework to reinforce IDS for securing Big Data environment using Spark[J]. Verma. Journal of Information Security and Applications, 2018 (10):1-11.

[10] 蒋晓宇.基于 tableau 可视化业务报表设计与实现[J]. 数字通信世界, 2017 (2):112-113.