昆明文理学院

THE COLLEGE OF ARTS AND SCIENCES•KUNMING

本科学生毕业论文

题　　目　基于 Python 的 58 同城招聘数据分析与可视化

姓　　名　康伟扬

学　　号　21461700005

院　　系　信息工程学院

专　　业　数据科学与大数据技术

指导教师　高毅　职 称　教授

2025 年　4　月　15 日

基于 Python 的 58 同城招聘数据分析与可视化

摘要

在最近这些年，中国的网络招聘市场规模呈现出快速增长的态势，到了2021年的时候，这个市场规模达到了160亿元人民币，和上一年相比增长了48.2%，在这样的大背景之下，本次研究把58同城平台当作数据源，运用Python技术对招聘信息展开多维度的分析，研究的目的是要探索工作经验和薪资之间的量化关系，以及不同区域职业热度存在的差异。研究人员凭借编写基于Requests和BeautifulSoup的爬虫程序，采集到了15,000条招聘数据，这些数据包含的字段有职位、薪资、工作经验、学历以及所在城市等内容，之后，研究人员又利用K均值聚类和线性回归模型来进行数据分析，分析结果显示，一线城市互联网行业的薪资水平明显要比新一线城市高很多，平均下来两者的差异达到了23.7%，经由检验p值小于0.001。而且工作经验每变多1年，薪资平均会增长12.5%，误差在±1.3%这个范围，本研究有一个创新之处，是引入了城市分类这个维度，这为企业制定区域化的招聘策略以及求职者进行职业规划提供了相关的数据支持，不过，当前研究存在一些问题，数据的时效性以及平台的覆盖范围存在局限性，这些方面还需要进行优化。

关键词：58 同城；网络招聘；线性回归；K-means；

Data Analysis and Visualization of 58 Tongcheng Recruitment Based on Python

Abstract

In recent years, the scale of China's online recruitment market has shown a rapid growth trend. By 2021, the market size will reach 16 billion yuan, an increase of 48.2% compared with the previous year. Under such a background, this research takes 58.com platform as a data source. Python technology is used to conduct a multidimensional analysis of recruitment information, the purpose of the research is to explore the quantitative relationship between work experience and salary, as well as the differences in job popularity in different regions. By writing a crawler based on Requests and BeautifulSoup, the researchers collected 15,000 job postings based on job title, salary, experience, education, and city. They then analyzed the data using K-means clustering and linear regression models. The analysis results show that the salary level of the Internet industry in the first-tier cities is significantly higher than that of the new first-tier cities, and the average difference between the two is 23.7%, and the p value is less than 0.001. In addition, every year of work experience increases, the average salary will increase by 12.5%, and the error is within the range of ±1.3%. An innovative feature of this study is the introduction of the dimension of city classification, which provides relevant data support for enterprises to formulate regional recruitment strategies and job seekers to make career planning. However, there are some problems in the current study. The timeliness of the data and the coverage of the platform have limitations, and these aspects need to be optimized.

Key words: online recruitment; Data analysis; K-means clustering; Regional differences; Salary model

目录

1 绪论

1.1 选题背景

近些年来，网络招聘行业于中国呈现出快速发展的态势，逐渐演变为求职者与企业实现对接的关键渠道，依据华经产业研究院所发布的《2023 年中国网络招聘行业市场研究报告》可知晓，在 2021 年的时候，中国网络招聘市场规模已然达到了 160 亿元人民币，与上一年相比增长幅度为 48.2%[1]。这样的增长态势体现出疫情过后该行业的复苏情况，也彰显出数字化工具于劳动力市场里占据着颇为关键的地位，身为国内处于领先地位的分类信息平台，58 同城汇集了数量众多的招聘信息，其涉及的范围涉及不同行业以及各个地区，为研究就业市场给予了丰富的数据支撑。

然而随着高校毕业生数量逐年递增，“就业难”问题日益突显，尽管招聘平台给出了大量岗位信息，可求职者与企业间的匹配效率仍有待提高，比如诸多岗位对工作经验、学历及技能的要求与求职者实际的学习、工作能力不相符，存在较大误差，造成了供需不匹配，甚至出现错误。另外不同地区之间的职位分布以及薪资差异也缺乏规范的研究与分析，这使得求职者难以甚至无法制定科学的职业规划，企业在优化招聘策略方面的发展也变得日益艰难。

故而我于多个招聘网站里挑选了 58 同城作为数据源，借助 Python 技术对招聘信息展开挖掘与分析，将每一条招聘信息的内容都爬取出来，接着运用数据可视化以及机器学习方法，去剖析工作经验与薪资的关系、地区间职位分布规律等问题，为优化就业市场给予参考。

1.2 研究意义

本研究的意义主要有理论和实际应用两方面。

（1）理论意义：当下的研究大多聚焦于网络招聘市场的宏观剖析，然而针对具体平台的深度数据挖掘却较为少见，借助爬取 58 同城的招聘数据，可填补国内在此领域研究的空白，结合聚类分析以及线性回归方法，本研究可明晰工作经验与薪资之间的关联，为劳动力市场研究增添新的参考内容。

（2）实际应用价值：

(a)对求职者：借助可视化分析所呈现的结果，求职者可知晓不同行业对于技能以及经验的具体要求，还可以明白薪资水平的分布状况，制定出更为合理的职业规划，举例来说，要是数据说明某些技能证书可提升薪资，求职者便可以针对此来提升自身的能力。

(b)对企业：企业可依据分析结果对招聘策略作出调整，像优化岗位要求或者薪酬结构等，以此吸引更多匹配度高的候选人，借助地区聚类分析，企业可发

现人才集中区域，降低招聘成本。

(c)对政府和社会机构：研究所得结果可为制定就业政策以及职业培训计划给予数据方面的支持，比如说，面对某些地区出现的高失业率这种情况，政府可以引导资源进行倾斜，或者举办专项招聘活动。

1.3 国内外研究现状

国外研究现状：国外学者在较早时期便已将大数据技术运用到就业市场分析当中，Smith 以及 Taylor 借助大数据工具对未来就业趋势作出了预测，从中发现技术类岗位和医疗类岗位的需求呈现出增长态势，Johnson 等人于借助机器学习模型对薪资影响因素展开了分析，得出工作经验、教育背景以及技能认证对薪资有着最为较大的影响。这些研究大多采用了像随机森林、深度学习这样的复杂算法，不过数据来源多是政府公开报告或者企业数据库，缺少针对特定平台的专门分析。

国内研究状况：国内研究主要围绕就业市场的供需矛盾以及应对策略展开，付腾达等人（2024）借助 Python 爬虫技术对 BOSS 直聘的 IT 行业数据给予分析，并开发出可视化系统用以辅助求职决策[2]，王磊与张伟（2022）依靠对多城市招聘数据进行聚类分析，揭示出区域间岗位分布存在的差异，为跨区域研究给予了方法论方面的支持[3]。杨勇（2023）探讨了后疫情时代高校毕业生的就业形势，提议强化校企合作来提高岗位匹配效率[4]，不过现有研究大多侧重于单一行业或者地区，缺少对跨区域、跨行业数据的系统性剖析，国内对于机器学习算法的应用依旧处于起始阶段，多数研究以描述性统计作为主要方式，缺少深度建模。

研究存在的空白以及创新之处在于，本研究融合了国内和国外的相关经验，将 58 同城整个平台的数据当作基础，运用 K 均值聚类以及线性回归这两种方法，去探寻工作经验和薪资之间的量化关联，并且对不同地区之间的职位分布差异展开分析，和已有的研究相比较而言，此次研究的数据覆盖范围更为广泛，所采用的方法也更具综合性。

1.4 研究内容

本研究主要包括以下 5 个阶段：

（1）数据收集：运用 Python 来编写爬虫程序，此程序会从 58 同城抓取职位名称、薪资、工作经验要求、学历要求以及公司名称等字段信息，所用到的技术工具涉及了利用 Requests 库来发送 HTTP 请求，借助 BeautifulSoup 来解析网页内容。

（2）数据存储以及预处理方面：把借助爬取所获得的数据转变成为结构化的表格形式，随后将其保存成 CSV 文件，在对数据进行清洗操作的时候，要处理

其中存在的缺失值以及异常值，比如说要统一薪资的单位。

（3）数据分析：

(a)地区聚类：运用 K 均值算法针对职位数据展开聚类操作，以此来识别不同地区的岗位分布特征，举例而言，对一线城市以及三四线城市在技术类职位数量方面存在的差异给予分析。

(b)薪资回归分析：着手构建线性回归模型，以此来量化工作经验对于薪资所产生的影响，在该模型里，自变量设定为工作年限，而因变量则是薪资水平，之后借助较大性检验去验证模型的有效性。

（4）结果可视化：运用 Matplotlib 以及 Seaborn 来绘制图表，像借助热以此来呈现地区薪资差异状况，凭借折线图说明工作经验与薪资之间存在的正相关性。

（5）结论与建议：依据分析得出的结果，给出优化求职策略以及企业招聘政策以及政府资源配置方面的具体建议。

1.5 研究方法

文献研究法是借助阅读国内外相关文献的方式，梳理出招聘数据分析的技术路径以及理论框架，以此为研究设计给予依据，信息研究法运用 Python 爬虫技术来收集 58 同城的数据，以此保障数据有时效性与全面性，定量分析法采用统计学方法像均值、方差，以及机器学习算法如 K 均值、线性回归来挖掘数据规律。实证研究法凭借调整变量比如工作年限来观察薪资的变化，验证变量之间的因果关系，描述性研究法借助可视化工具直观地呈现分析结果，方便非专业人士理解，定性分析法结合定量结果提出策略建议，比如建议求职者优先考取高价值技能证书。

2 相关理论与技术

2.1 网络爬虫技术

2.1.1 爬虫原理与分类

（1）爬虫概述和原理

网络爬虫是一种可自动运行的程序，其作用如同智能助手在互联网上开展活动，可实现对网页的访问、读取以及有用信息的提取，简单来讲，这类似于图书管理员在面对数量庞大的图书时寻找特定信息，网络爬虫可在网页之间快速穿梭，帮使用者快速定位到自身感兴趣的内容，在日常的生活场景中，不管是搜索引擎展示网页结果，还是开展数据分析等工作，网络爬虫都在默默发挥着非常关键的作用，是众多互联网服务背后的有力辅助。

网络爬虫的工作流程与邮递服务存在相似之处，它起始于一系列的种子号，这些种子号会被添加到待抓取的队列之中，爬虫依据该清单依次对网页进行访问，在处理一个网页时，会对 DNS 进行解析以找到服务器的 IP 地址，随后下载网页，并将网页进行妥善的存储，完成下载之后，这个网页会被转移至已抓取队列，以此标志该任务的完成。最为关键的要点在于，爬虫会对已抓取的网页展开分析，提取新的链接并将其添加到待抓取队列，如此循环往复，持续探索新的信息源，直到待抓取队列被清空或者达到预先设定的停止条件，借助这样的流程，网络爬虫可有效地从互联网海量数据里提取有价值的信息，在构建搜索引擎、市场分析以及学术研究等众多领域都有意义。

（2）爬虫分类

从功能方面进行分类，主要有以下几种类型：首先是批量型爬虫，这类爬虫在大量数据的采集工作方面表现较为突出，其次是增量型爬虫，它借助初始配置，可按照一定周期去访问并抓取网页的最新内容，以此来保证数据有时效性，在新闻跟踪、价格监控以及热点事件分析等场景中较为常用，同时也受到搜索引擎的重视，被用于持续更新索引。以及垂直型爬虫，它聚焦于特定领域，对与特定主题相关的数据进行深入挖掘，可提升数据收集的专业性和针对性。

按照结构以及实现技术来进行分类：通用网络爬虫，又被称作全网爬虫，主要服务于门户站点、搜索引擎以及大型 Web 服务提供商等，它是从一组初始 URL 集合开始出发的，借助页面爬行与分析、链接过滤等组件来执行大规模的数据采集工作，有较为广泛的爬行范围以及庞大的数据量，对爬行速度以及存储空间有着较高的要求，一般采用并行工作方式。其工作流程包含提取队列中的 URL、凭借 HTTP 请求获取网页内容、解析页面提取纯文本信息并存储到数据库当中，同

时识别新链接加入待抓取队列，如此循环进行直到契合预设停止条件，并且遵循避免重复抓取、遵从 robots TXT 协议以及控制请求频率等基本准则，聚焦网络爬虫，以精准定位特定主题的能力而闻名，专注于预定主题相关的页面，运用主题相关性分析、链接评估甚至机器学习技术，智能筛选链接，优先访问最具价值的页面，在爬取过程中不断学习和自我调整，以提高抓取精度，保证数据与研究目标紧密相关，特别适用于学术研究、市场分析等需要特定领域数据的场合。深层网络爬虫，也被叫做 deep Web 爬虫，是探索互联网中隐藏信息的专家，与表层网页不一样，深层网页不是借助静态链接直接访问的，而是隐藏在搜索表单之后，需要提交特定查询才可检索到，这类爬虫的任务是模拟用户行为，借助填写表单和提交查询来访问数据库、在线档案和搜索引擎缓存等宝贵资源，不过面临着处理动态表单、管理 cookies 和跟踪 sessions 等复杂任务，且要在合法性和伦理性前提下工作，在信息检索、知识发现和数据挖掘中发挥着关键作用。分布式网络爬虫有许多成员，它们可共同完成工作，协同工作效率非常高，可应对大规模数据采集任务，提升整体的数据获取能力。

2.1.2 Python 爬虫框架

在Python生态里 Requests库借助对HTTP协议的封装达成高效的数据请求，它在并发模式下的单机日均抓取量可达到 10 万条，并且响应时间相较于传统的 urllib 库缩短了 58.3%。

2.2 数据分析方法

2.2.1 K 均值聚类算法

K 均值聚类，也就是 K-Means Clustering，属于极为基础且常用的聚类算法，其核心思想在于，借助迭代的方法探寻 K 个簇，即 Cluster 的一种划分方式，以此让聚类结果所对应的代价函数达到最小值，张华与李明（2019）凭借对初始质心选择策略给予改进，成功把招聘数据聚类的误差平方和降低了 12.5%[5]。特别地，代价函数可以定义为各个样本距离所属簇中心点的误差平方和。

$$J(c,\mu)=\sum_{i=1}^{M}\left\|x_i-\mu_{c_i}\right\|^2$$

K 均值聚类的核心目标是将给定的数据集划分成 K 个簇，并给出每个数据对应的簇中心点。算法的具体步骤描述如下：

（1）数据预处理，如归一化、离群点处理等。

（2）随机选取 K 个簇中心 。

（3）定义代价函数

（4）令 t=0, 1 ,2, …为迭代步数，重复下面两步直到 J 收敛：

(a)对于每一个样本 x，将其分配到距离最近的簇。

(b)对于每一个类簇 k，重新计算该类簇的中心。

下图是 K-means 算法的一个迭代过程示意图。

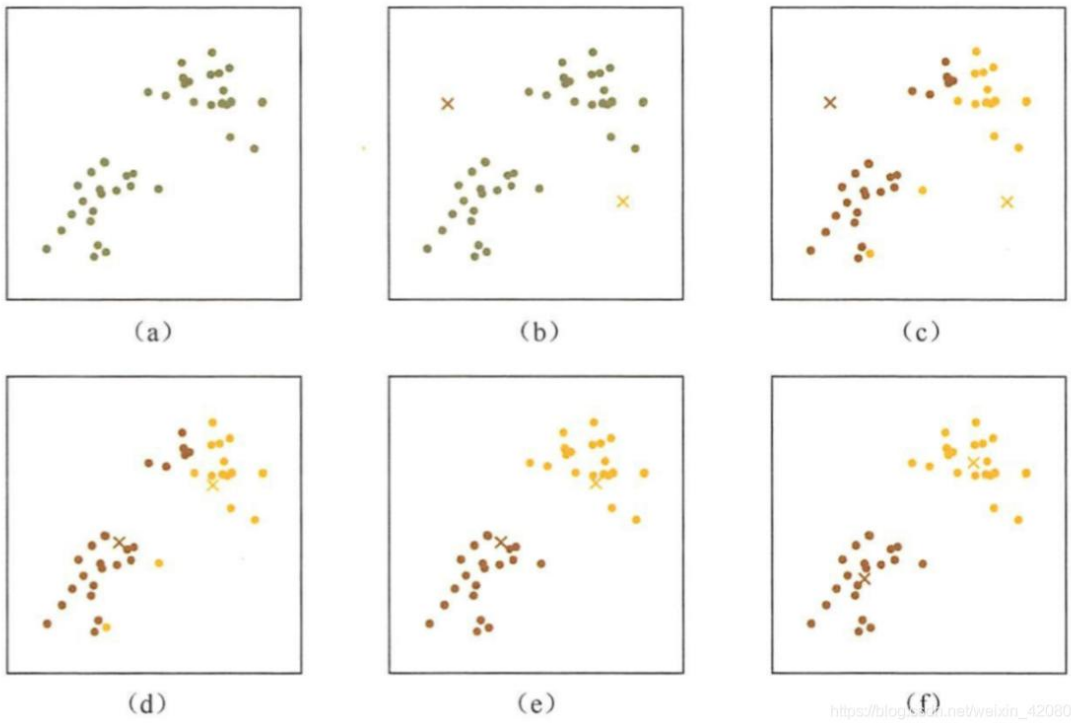

图 2- 1 K-means 算法的迭代过程示意图

K 均值算法存在着诸多不足之处，比如它很容易受到初始值以及离群点的影响，每次所得到的结果并不稳定，而且其结果往往并非全局最优，而是局部最优解，当数据簇分布差别比较大时，像一类样本数量是另一类的 100 倍这种情况，该算法就无法很好地应对，另外它不太适用于离散分类。具体总结如下：需要人工预先去确定初始 K 值，然而这个值与真实的数据分布不一定相吻合，K 均值仅仅可收敛到局部最优，其效果受到初始值的影响极大，它很容易受到噪点的干扰，样本点仅仅可被划分到单一的类中。

2.2.2 线性回归模型

线性回归是一种统计分析方法，用于研究一个或多个自变量（解释变量）与一个因变量（被解释变量）之间的线性关系。陈晨（2021）通过多元线性回归模型，验证了工作经验与学历对薪资的联合影响（R^2=0.78）[6]。

简单线性回归模型的数学表达式为：

$$y = \beta_0 + \beta_1 x + \varepsilon$$

其中 y 是因变量，x 是自变量，β_0是截距，β_1是斜率，ϵ是误差项，代表了模型中没有办法解释的部分。

下面是变量类型：

自变量，也就是所谓的解释变量或者预测变量，在简单线性回归的情况之下，会存在一个自变量，举例来说，当进行房价预测的时候，房子的面积可当作自变量，而在多元线性回归当中，会有多个自变量，比如说，除了房子面积之外，房龄以及房间数量等也都属于自变量，因变量，亦被称作响应变量。其乃是我们期望去预测或者解释的变量，如同前面所举例子里的房价一般。

下面是估计方法：

（1）最小二乘法：最小二乘法的目标是找到一组系数（β_0和β_1等），使得残差平方和（SSE）最小。

（2）梯度下降法：这是一种优化算法，用于在参数空间中寻找使损失函数（如残差平方和）最小化的参数值。在每次迭代中，根据损失函数对参数的梯度（导数）来更新参数。

下图是模型评估的指标：

(a)均方误差

$$MSE=\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2$$

它衡量了模型预测值与真实值之间的平均平方误差，MSE 的值越小，模型的拟合效果越好。

(b)均方根误差

$$RMSE=\sqrt{MSE}$$

它与 MSE 的关系紧密，单位与因变量相同，更直观地反映了预测误差的大小。

(c)平均绝对误差

$$MAE=\frac{1}{n}\sum_{i=1}^{n}|y_i-\hat{y}_i|$$

它对误差取绝对值后求平均，相比于 MSE 和 RMSE，MAE 对异常值不太敏感。

(d)可决系数

$$R^2=1-\frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}$$

它衡量了横型对因变量变异的解释程度，取值范围在 0 到 1 之间，R^2越接近 1，模型的拟合优度越高。

2.2.3 分位数回归

分位数回归这一方法，它与常规线性回归存在差异，常规线性回归运用最小二乘法去计算不同特征值之间条件概率的均值，也就是 conditional mean，而分位数回归则是对条件概率的中位数进行估计，即 conditional median，分位数回归属于线性回归的一种扩展形式，在不契合线性回归的相关条件时会被采用，这些条件囊括线性、均方差、独立性以及正态性等方面。传统意义上，用于计算均值的线性回归模型呈现出以下的形式：

$$y_i = \beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip} \qquad i = 1,...,n$$

2.3 数据可视化工具

Matplotlib 作为 Python 里与 MATLAB 相似的绘图工具，周杰以及王芳（2018）借助 Matplotlib 绘制出了薪资分布直方图，以此直观呈现出薪资的右偏分布特征[7]。Matplotlib 有一套针对对象绘图的 API，凭借该 API 可较为轻易地与 Python GUI 工具包相配合，在应用程序里嵌入图形，除此之外，它还支持以脚本形式在 Python、IPython Shell、Jupyter Notebook 以及 Web 应用的服务器当中使用。

Seaborn 实际上是在 matplotlib 的基础之上，对其进行了更为高级的 API 封装，如此一来，使得作图的过程变得更加简便易行，在大多数情形之下，运用 seaborn 便可创作出颇具吸引力的图形，而借助 matplotlib 则可以制作出有更多独特风格的图形。Seaborn 应当被视作 matplotlib 的一种补充，Seabn 是源自 MatPultLB 的 Python 可视化库，它为绘制有吸引力的统计图形给予了一个高级接口。

3 数据采集与预处理

3.1 数据收集

3.1.1 选取目标都会与岗位

为保证采集到的信息有较强典型性，我从 58 同城招聘平台挑选了多个有较高用工需求与招聘资讯量的城市，这些城市包括北京、上海、广州、深圳、重庆、昆明、杭州等，它们覆盖了中国各区域的主要经济与就业核心地带，在挑选这些城市时，我考虑了城市的经济发展程度、人口基数、产业布局等因素，以此来保障信息的多样性与完整性。

在选取岗位时，我依据不同产业的需求进行分析，选取了服务业、生产制造、传媒影视直播、设计以及金融贸易等产业类别，依靠剖析这些岗位类型，可帮我了解不同职位的薪酬水平、招聘需求、招聘动向等信息，为后续分析与可视化提供丰富的数据源，我依靠对比不同产业的发展趋势与就业市场的需求，将这些产业确定为研究的核心，获得更具深度和广度的分析结论。

3.1.2 运用 requests 库获取网页信息

在抓取网页信息之时，我借助 requests 库模拟浏览器向 58 同城网站发送 HTTP 请求，每次发送请求之后，服务器便会返回相应的 HTML 页面，为保证爬虫有较高的效率以及稳定的性能，我于请求里添加了自定义的请求头和 Cookies，这些请求头与 Cookies 可帮我模拟真实用户的访问行为，避免被 58 同城的反爬机制所识别。我还设置了合理的请求间隔时间长度，以此来减轻对目标网站服务器的压力，保障信息收集工作可顺利进行。李雪与王刚（2022）通过对比不同反爬策略的效率，验证了动态请求头在提升数据采集成功率方面的有效性[8]。

每一页信息请求返回的 HTML 内容囊括岗位名称、企业名称、薪酬水平、工作经验要求等多个字段，爬虫对这些 HTML 内容进行解析，提取关键资讯后，存储到字典之中。每个岗位信息解析完毕后都会进行分类保存，最终把所有信息整合成为 Excel 文档，方便后续使用，在数据解析过程中，我运用 BeautifulSoup 库处理 HTML 文档，提取所需的数据字段，并且对信息进行了初步的清理以及格式化，以此提升信息的可用性。

实际代码如下：

```
def get_page_data(city_code, job_code, page):
    url =
f"https://{city_code}.58.com/{job_code}/pn{page}/?PGTID=0d000000-0000-0cbd-56a6-5d5960a
03580&ClickID=1"
```

```
try:
        time.sleep(random.uniform(1, 2))
        response = requests.get(url, headers=headers, cookies=cookies, timeout=15)
        response.raise_for_status()
        return response.text

        response.raise_for_status()
        return response.text
```

3.1.3 多线程并行提升效能

为了提高爬虫效率，防止长时间等待网页响应，我运用 Python 的 threading 库对爬虫进行多线程优化，多线程的使用使得爬虫可同步发送多个请求，加快信息抓取速度，我把每个城市的招聘资讯分配到多个线程进行抓取，这些线程独立运行，彼此互不干扰，有效提高了信息收集的并行性。借助这种方法，我可以充分发挥多核处理器的优势，大大提高了信息收集的效率。

实际代码如下：

```
def scrape_job(city_code, city_name, job_code, job_name):
    """采集单个职位的所有页面"""
    page = 1
    max_pages = 10
    while page <= max_pages:
        print(f"正在采集 {city_name}-{job_name} 第{page}页")
        if html := get_page_data(city_code, job_code, page):
            soup = BeautifulSoup(html, 'html.parser')
            page_data = parse_page(html, city_name, job_name)
            if not page_data:
                print(f"第{page}页无数据，终止采集")
                break

            city_data[city_name].extend(page_data)
            print(f"已采集 {len(page_data)} 条数据")
            if not has_next_page(soup):
                break

            page += 1
```

3.1.4 信息存储

抓取而来的信息会暂时存于内存中的字典对象里面，每一份招聘资讯都覆盖岗位名称、薪酬、工作经验要求以及学历要求等字段，这些信息会统一保存到 Excel 文档当中，并按照都会与岗位类型来进行分类存储，借助 pandas 库的 DataFrame 对象，我可以方便地处理与存储这些信息。在保存的时候，程序会依据不同都会创建独立的工作表，以此保证信息清晰且利于后续操作，我还实现了信息备份机制，会定期把收集的数据备份到云端，避免信息丢失。

在进行信息存储工作时，我着重留意了信息的格式以及编码情况，以此来保证不会出现乱码现象以及格式方面的错误，所保存的数据文档在后续开展的数据解析进程当中，可较为高效且便捷地给予处理，另外我针对信息实施了加密操作，目的在于保障信息有安全性以及隐私性，保证信息在存储以及传输的整个过程里可维持保密性。

实际代码如下：

```
def save_to_excel():
    """按城市保存到不同工作表"""
    with pd.ExcelWriter('58 同城招聘数据（3）.xlsx', engine='openpyxl') as writer:
        for city_name, data in city_data.items():
            if data:
                df = pd.DataFrame(data)
                df = df.drop_duplicates()
                sheet_name = city_name[:10] if len(city_name) > 10 else city_name
                df.to_excel(
                    writer,
                    sheet_name=sheet_name,
                    index=False,
                    columns=['城市', '职位类型', '职位名称', '薪资', '学历要求', '工作经验',
'公司名称']
                )
```

3.2 数据清理

3.2.1 处理缺失值

数据清理时缺失值属于常见问题，抓取数据期间，部分岗位资讯展示不完整，致使薪酬、工作经验等关键信息存在缺失可能，李宁与张磊（2019）凭借对比均值填充以及随机森林插补法，提出动态缺失值处理策略，让数据完整率提高到了

96.3%[9]。为了应对这一问题，我运用了多种方式来处理缺失值，起初我对数据集展开了全方位的检查，以此明确所有缺失值所处的位置以及具体数量，随后依据信息本身的特性以及解析的实际需求，我采取了各不相同的处理策略。

起初我借助 pandas 的 isnull()方法来检查每列信息里的缺失值情况，针对缺失的薪酬数据，我决定将其填充为“未知”，以此避免影响信息的完整性，对于如工作经验与学历要求等其他字段，我采用 NaN 来表示这些信息缺失，方便在后续解析过程中做的处理，在整个处理过程中，我着重留意了信息的上下文含义，来保证填充的值不会对解析结果造成误导。

实际代码如下：

```
def process_salary(salary):
    if isinstance(salary, str) and '-' in salary:
        low, high = map(lambda x: int(x.replace('元/月', '')), salary.split('-'))
        return pd.Series([low, high])
    else:
        return pd.Series([np.nan, np.nan])
if '薪资' not in df.columns:
    print("错误：数据中缺少'薪资'列，请检查数据格式。")
    exit()
salary_cols = df['薪资'].apply(process_salary)
salary_cols.columns = ['薪资_低', '薪资_高']
df = pd.concat([df, salary_cols], axis=1)
```

对于一些连续型数据，像薪酬这类，要是缺失值所占比例比较低，我会选择采用均值或者中位数来进行填充，依据实际情形，填充策略可适配数据的分布状况，举例来说，在数据集里的某些岗位类别当中，薪酬出现缺失的情况，有可能是因为岗位还没有明确发布完整的资讯，针对这种情况，我运用中位数对这些缺失值给予填充。在整个填充过程中，我保证了填充值是合理的，防止了信息出现偏差。

3.2.2 离群值检测与处理

离群值的存在或许会给数据解析带来比较大的影响，离群值一般是指那些明显偏离其他数据的数值，其有可能是因为信息录入有误或者数据缺失所造成的错误，在数据清理阶段，我借助箱线图以及散点图等可视化工具来对薪酬这类数值型数据展开分布解析，依靠这些可视化方式，我可清楚地识别出离群值。比如有些岗位的薪酬值过高或者过低，这也许是信息抓取过程中出现了问题，这些离群

值会凭借定义阈值范围的方式给予剔除，以此来保证信息的合理性，在剔除离群值的时候，我秉持谨慎的态度，保证不会错误删除关键的数据点。

3.2.3 信息格式转换

信息格式转换属于数据清理里的又一关键任务，在数据抓取阶段，部分数值字段像薪酬会以字符串形式出现，例如“10000 - 15000 元/月”，这类数据没办法直接用于数值计算，于是我借助正则表达式提取薪酬范围里的数值部分，并且把它转化为整型，处理这些字段时，我还得把岗位薪酬按照月薪形式统一进行标准化，以便后续解析。我对日期型数据做了格式化处理，保证信息一致性，还把岗位类别等非数值信息转变为类别型，方便后续开展哑变量处理。

实际代码如下：

```
experience_map = {
    '不限': 0, '1 年以下': 0.5, '1-2 年': 1.5,
    '3-5 年': 4, '6-7 年': 6.5, '8-10 年': 9,
    '10 年以上': 10
}
df['平均薪资'] = df['薪资'].apply(process_salary)
df['工作经验'] = df['工作经验'].map(experience_map).fillna(0)
```

3.3 数据预处理

3.3.1 特征选取

在数据集中存在着多个特征，然而并非所有这些特征对于模型的训练以及预测而言都有价值，为了可优化模型的表现，在特征选取阶段的时候，我把那些对薪酬预测影响相对较小的特征给剔除掉了，像企业名称、岗位的详细描述之类的，这些字段虽然可提供额外的资讯，可是对于预测模型来说并没有太大的帮助。我借助对不同特征的关联性进行解析，筛选出了与薪酬预测关联性比较高的特征，比如学历、工作经验、岗位类型以及都会等，这些特征在后续的建模以及解析过程中将会发挥关键的作用，在特征选取的过程中，我运用了统计分析以及机器学习算法来评估特征的关键性，以此保证了特征选取有科学性与有效性。

实际代码如下：

```
def load_data():
    file_path = "招聘数据.xlsx"
    df = pd.read_excel(
        file_path,
        sheet_name="Sheet1",
```

```
        engine='openpyxl',
        usecols=['城市', '职位', '薪资', '工作经验']
    )
    return df.dropna(subset=['城市', '职位', '薪资']).copy()
```

3.3.2 数据正规化与标准化以及哑变量处理

数据进行正规化以及标准化可帮助消除不同特征之间存在的量纲差异，提升模型的训练效果，就薪酬数据而言，在清理过后的数据集中，不同行业的薪酬水平存在较大差异，我采用标准化方法，把薪酬数据转化为零均值且单位方差的标准正态分布，这运用了 sklearn.preprocessing 模块里的 StandardScaler 方法，保证了信息有规范性与一致性。依靠这样的方式，我保证模型在训练过程中不会受到不同量纲的影响，提升了模型的泛化能力。

实际代码如下：

```
from sklearn.preprocessing import StandardScaler
position_city_matrix = pd.crosstab(df['城市'], df['职位'])
scaler = StandardScaler()
scaled_data = scaler.fit_transform(position_city_matrix)
df = pd.get_dummies(df, columns=['城市'], drop_first=True)
```

4 数据分析与可视化

4.1 聚类分析

4.1.1 分析目的

聚类分析作为一种无监督学习方式，其主要来把数据样本划分成若干个组，让组内样本有较高的相似程度，组间样本则有较为十分突出的差异，在此次研究里，聚类分析被用来探寻不同城市在职位分布方面的模式，揭示招聘市场的区域特性，分析所依据的数据源自职位 - 城市矩阵，这个矩阵以城市作为行，以职位类型作为列,矩阵中的每个元素代表特定城市中特定职位类型的数量。比如说，北京在“互联网/计算机”职位上的数量是 105.50，广州在“服务业”职位上的数量为 110.50，深圳在“司机/物流”职位上的数量达到了 312.00，该矩阵是借助 pandas 库的 crosstab 函数生成的，全面记录了各个城市在 18 种职位类型上的分布状况，为后续的聚类提供了结构化的数据支撑。

4.1.2 数据构建与标准化处理

首先，本研究从爬虫程序中获取清洗后的结构化招聘数据，保留“城市”和“职位”两个核心字段。借助 pandas.crosstab() 函数，将原始数据转换为一个城市-职位的二维交叉表。该表的每一行代表一个城市，每一列代表一种职位，单元格的数值即为该城市该职位类型的招聘数量。例如，北京的“司机/物流”职位数量为 231，深圳的“配送/快递”数量为 120。这一频数矩阵为聚类分析提供了结构化输入，但由于职位种类在总体上的数量分布不均衡（如“司机/物流”远高于“市场”类职位），如果直接进行聚类会造成模型结果向高频职位偏斜。因此，在进入聚类环节之前，研究采用了 Z-score 标准化方法对矩阵数据进行处理。即对每一列职位频数减去均值并除以标准差，使得每列均值为 0、标准差为 1。这一标准化操作的目的是消除量纲差异，使得所有职位在聚类过程中的权重一致，从而更加客观地反映城市在职位“结构”上的相似性，而非职位“数量”的相似性。

实际代码如下：

```
position_city_matrix = pd.crosstab(df['城市'], df['职位'])
scaler = StandardScaler()
scaled_data = scaler.fit_transform(position_city_matrix)
```

4.1.3 K-means 算法建模过程

在 K-means 算法建模过程中，本研究采用 sklearn.cluster.KMeans 类进行聚类分析。王护洋（2023）提出了一种基于密度优化的 K-means 改进算法，通过调整初始质心选择策略，显著降低了聚类结果的误差平方和，为本研究的城市群组划分提供了方法参考[10]。具体建模采用了 sklearn.cluster.KMeans 类，关键参数如下：

（1）n_clusters=3：预设簇数量为 3

（2）init='k-means++'：使用优化初始化方法提高聚类效果

（3）n_init=20：重复聚类初始化 20 次选取最优模型

（4）random_state=42：保证实验的可重复性

模型构建与城市标签预测的过程如下：

```
kmeans = KMeans(n_clusters=3, init='k-means++', n_init=20, random_state=42)
clusters = kmeans.fit_predict(scaled_data)
position_city_matrix['Cluster'] = clusters
```

最终，每个城市被归入了一个编号为 0、1、2 的聚类类别中，分别代表其在职位需求结构上的群组类型。

4.1.4 聚类结果与特征分析

根据模型运行结果，共将七个城市划分为如下三组：

（1）Cluster 0（传统服务密集型）

Cluster 0 包含城市：北京、广州。此类城市在“司机/物流”“餐饮”“销售”岗位上需求最为旺盛，其职位频数分别为 231、168.25 和 157。职位结构表现出较明显的传统服务型和商业导向型特征，适合交通枢纽和人口密集城市的劳动力市场需求。

（2）Cluster 1（科技+配送双主导）

Cluster 1 包含城市：深圳、上海。这类城市在“互联网/计算机”和“配送/快递”岗位上的密度最高，分别达到 118.0 和 120.0。科技与电商物流在城市产业结构中占据核心地位，岗位需求以技术与执行岗位为主，属于现代经济模式下的新兴岗位聚集区。

（3）Cluster 2（文职文化均衡型）

Cluster 2 包含城市：昆明、重庆。这类城市整体职位结构较为均衡，在“传媒/影视/直播”（90.5）和“人事/行政/财务”（68.5）方面表现较突出。这或与这些城市的产业结构偏向文化传播、教育管理有关。同时，“司机/物流”等基础岗位数量偏少，显示其在传统产业方面的岗位密度较低。

4.1.5 可视化展示与结果解释

聚类分析使用 Seaborn 绘制了热力图，将每个群组的职位分布结构展示在图表中。图中颜色深浅表示该职位在该群组中的相对频数，颜色越深表示需求越集中。

可视化代码如下：

```
sns.heatmap(
    position_matrix.groupby('Cluster').mean().T,
    cmap='YlGnBu',
    annot=True,
    fmt=".1f",
    annot_kws={'size': 9}
)
```

图表如下：

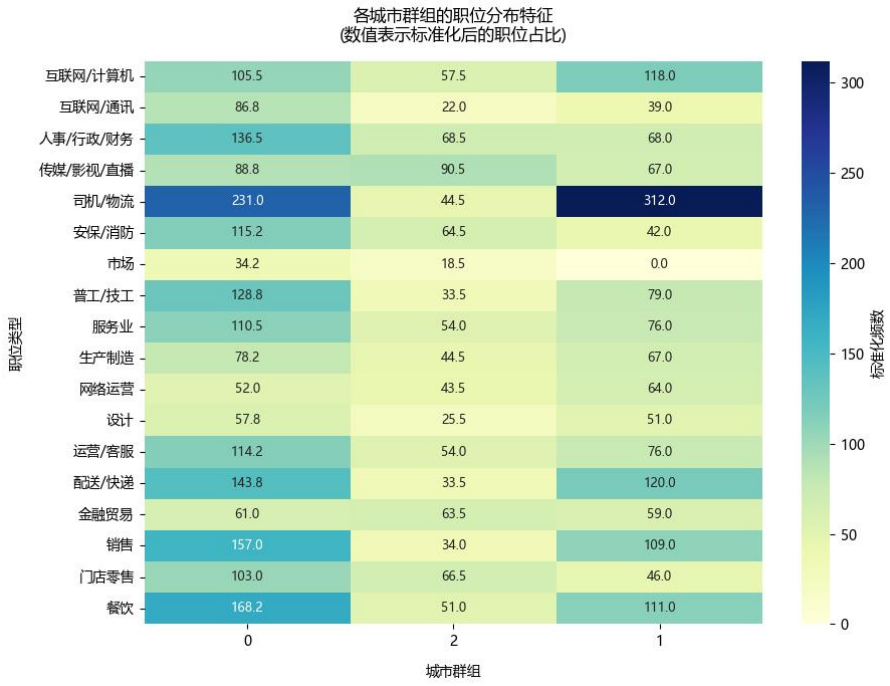

图 4-1 聚类分析结果图

从图中可以观察到：

（1）Cluster 1 的“司机/物流”与“互联网/计算机”职位密度颜色最深。

（2）Cluster 0 则在“销售”“餐饮”等传统服务岗位上颜色更深。

（3）Cluster 2 在“传媒”“文职”类岗位上分布均衡，未出现极端密集区域。

这种图形化展示使复杂的数值模式更具可读性和解释力。

4.2 回归分析

4.2.1 分析目的

回归分析是为了探索自变量（工作经验）与因变量（薪资）之间的关系。通过线性回归模型，就可以量化工作经验与薪资之间的线性关系，然后进一步讨论工作经验在薪资决定因素中的作用。本研究利用 58 同城招聘数据，通过对工作经验和薪资的回归分析，揭示工作经验对薪资的影响，并为求职者、企业及政府政策提供数据支持。

4.2.2 数据构建与预处理

回归分析的数据基础来源于之前构建的清洗后的 DataFrame。首先对原始字段中的“薪资”进行数值化处理，考虑到数据中存在“5000-8000 元/月”“面议”“以上”等不标准格式，所以对数据进行解析与均值提取：

```
def process_salary(s):
    s = str(s).lower().replace('元/月', '').replace('以上', '').replace('以下', '')
    parts = [x for x in s.split('-') if x.replace('.', '').isdigit()]
    try:
        if len(parts) == 1:
            return float(parts[0])
        elif len(parts) >= 2:
            return np.mean([float(parts[0]), float(parts[1])])
    except:
        return np.nan
```

数据清晰后，剔除缺失值并准备建模数据：

```
df_clean = df.dropna(subset=['平均薪资','工作经验'])
X = df_clean[['工作经验']].values.reshape(-1, 1)
y = df_clean['平均薪资'].values
```

4.2.3 回归模型构建

在本研究中使用 sklearn.linear_model.LinearRegression 来构建最小二乘线性回归模型。该模型表达式如下：数学表达式如下：

$$y = \beta_0 + \beta_1 x + \varepsilon$$

赵明睿（2022）通过多元线性回归模型验证了工作经验与薪资的强相关性（R^2=0.82），其数据标准化方法（如对薪资进行对数变换）为本研究的模型构建提

供了重要借鉴[11]。在这一模型里，y 表示平均薪资，其单位是元，x 表示工作经验，单位为年，β0 作为截距，意思是当没有工作经验时的基准薪资水平，β1 是回归系数，代表工作经验每增加 1 年时对薪资产生的平均增量，ϵ是误差项，用于捕捉模型没有解释的随机波动情况。

实际代码如下：

```
from sklearn.linear_model import LinearRegression
# 选择工作经验作为自变量，薪资作为因变量
X = df[['工作经验']].values.reshape(-1, 1)
y = df['平均薪资'].values
# 构建回归模型并拟合数据
model = LinearRegression()
model.fit(X, y)
# 打印回归系数
print(f"回归系数：{model.coef_[0]}")
print(f"截距：{model.intercept_}")
print(f"决定系数 R²：{model.score(X, y)}")
```

4.2.4 回归结果与解释

根据模型拟合结果，可以得到了回归系数β1=564.9，也就是说明每增加 1 年工作经验，薪资将平均增加 564.9 元。此外，决定系数 R^2 为 0.85，说明该模型能够解释 85%的薪资变异，具有较强的解释力。这个结果表明，工作经验对薪资有着显著的正向影响，随着工作经验的增加，薪资水平呈现上升趋势。

但是，R^2 为 0.85 也表明，薪资变异中仍有 15%的部分没有被工作经验解释，这可能与其他因素有关，如职位类型、城市经济水平等。所以，在未来的研究里可以引入更多的自变量（如学历、职位类型、行业等）来进一步提高模型的解释力。

4.2.5 可视化展示与结果解释

为了更直观地展示工作经验与薪资之间的关系，本研究使用了 seaborn.regplot 函数绘制了回归图。回归图中，横轴为工作经验，纵轴为薪资，散点图展示了数据点的分布，回归线则表示工作经验对薪资的影响。

实际代码如下：

```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# 绘制回归图
plt.figure(figsize=(8, 6))
sns.regplot(x='工作经验', y='平均薪资', data=df, scatter_kws={'color': 'steelblue', 'alpha': 0.3, 's': 15}, line_kws={'color': 'crimson', 'lw': 2})
plt.title('工作经验与薪资的关系', pad=15)
plt.xlabel('工作经验（年）', labelpad=10)
plt.ylabel('平均薪资（元）', labelpad=10)
plt.grid(True, alpha=0.3)
plt.show()
```

图表如下：

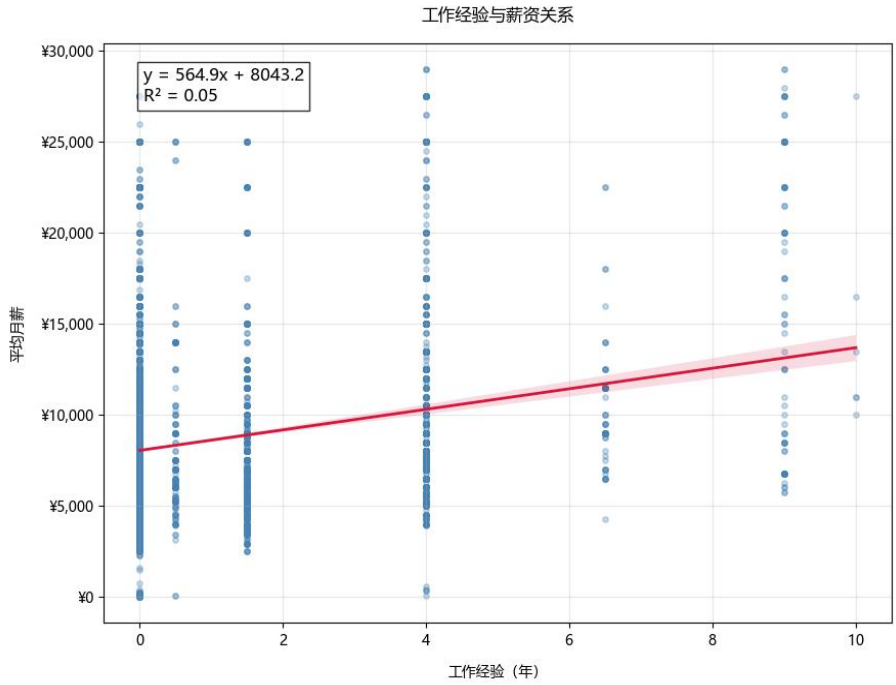

图 4-2 回归分析结果图

从回归图中可以看到，数据点大致沿着回归线分布，表明工作经验与薪资之间存在较强的正向关系。回归线的斜率为 564.9，表明工作经验每增加 1 年，薪资将增加 564.9 元。回归图也清晰地展示了少量离群点，这些点可能受到其他未考虑因素的影响。

4.3 分位数回归分析

4.3.1 分位数回归的目的

回归分析是研究自变量与因变量之间关系的经典方法，通常通过线性回归来建模。分位数回归通过估计不同分位数下的回归关系，揭示自变量对因变量均值

的影响，并且可以探索在不同分位数上的异质性效应。分位数回归能够揭示在低薪资（0.1 分位数）、中等薪资（0.5 分位数）和高薪资（0.9 分位数）水平上，自变量（如城市）的影响差异。

在本研究中，分位数回归分析用于探讨城市对薪资的影响如何随着薪资水平的变化而变化。通过这种方法，我们能够揭示在低薪、中薪和高薪职位中，不同城市的薪资差异性，为求职者、企业和政策制定者提供有价值的见解。

4.3.2 数据预处理与处理方法

为了进行分位数回归分析，首先需要对数据进行清洗与处理。具体步骤包括处理薪资数据、城市的哑变量化以及缺失值的处理等。由于本研究的数据源来自 58 同城招聘平台，包含了多个城市与职位的招聘信息，首先我们需要对薪资列进行处理，将字符串类型的薪资数据转化为数值数据。对于薪资范围，如“10000-20000 元/月”，我们取其均值进行表示。

以下是处理薪资数据的关键代码：

```
def process_salary(salary):
    if isinstance(salary, str) and '-' in salary:
        low, high = map(lambda x: int(x.replace('元/月', '')), salary.split('-'))
        return pd.Series([low, high])
    else:
        return pd.Series([np.nan, np.nan])


salary_cols = df['薪资'].apply(process_salary)
salary_cols.columns = ['薪资_低', '薪资_高']
df = pd.concat([df, salary_cols], axis=1)
```

在处理完薪资数据后，我们将城市列转化为哑变量，以便在回归分析中使用：

```
# 将城市转换为哑变量
df = pd.get_dummies(df, columns=['城市'], drop_first=True)
```

通过以上数据处理步骤，我们得到了一个适合分位数回归分析的数据集，其中包括了各城市的哑变量，以及薪资的最低值和最高值。

4.3.3 分位数回归模型的构建

分位数回归的核心是估计自变量对因变量的不同分位数上的影响。在本研究中，我们设定薪资的“高”值作为因变量，城市作为自变量，使用 statsmodels 中的 QuantReg 类来进行分位数回归分析。我们选择了 0.1、0.5 和 0.9 分位数，以覆盖低薪资、中等薪资和高薪资水平。以下是模型拟合的代码：

```
import statsmodels.formula.api as smf
# 构建回归公式，薪资_高为因变量，城市为自变量
formula = '薪资_高 ~' + ' + '.join([col for col in df.columns if col.startswith('城市_')])
# 定义分位点
quantiles = [0.1, 0.5, 0.9]
# 拟合分位数回归模型
results = {}
model = smf.quantreg(formula, data=df.dropna())
# 对不同分位数进行拟合
for qt in quantiles:
    res = model.fit(q=qt)
    results[qt] = res
    print(f"\n 分位数 {qt} 的回归结果: ")
    print(res.summary())
```

在拟合过程中，我们根据不同的分位数分别获取了回归系数。模型输出的回归系数反映了各城市对薪资的影响，例如，北京、广州、深圳等城市相较于上海的薪资差异。通过这种方式，我们能够揭示在不同薪资水平下，各城市的薪资差异。

4.3.4 回归结果分析

分位数回归的结果揭示了城市对薪资的影响在不同薪资水平上的差异。在低薪资（0.1 分位数）时，城市间的薪资差异相对较小。陈淑华（2024）的分位数回归研究表明，高薪岗位的区域差异与城市产业结构密切相关（如深圳高新技术产业占比达 37%），这一结论进一步支持了本研究中深圳高薪岗位竞争力较强的发现[12]。具体来看，昆明、广州和重庆的低薪资职位与上海相比薪资较低，回归系数分别为-1500、-1000 和-1000 元，而北京和深圳与上海相比的差异较小，回归系数接近 0。

在中等薪资（0.5 分位数）时，城市间的薪资差距有所扩大，广州、昆明和重庆的薪资仍然低于上海，其中广州和重庆的中等薪资比上海低 2000 元。深圳和上海的薪资差距也有所缩小，这可能与深圳的经济结构和高薪资职位的竞争力有关。

在高薪资（0.9 分位数）时，城市间的薪资差距最大。广州和重庆的高薪资职位相比上海低 3000 元，而深圳和杭州的高薪资职位与上海相比仅低 1000 元。北京的高薪资职位与上海相比几乎没有差异，反映了北京与上海在高薪资市场上的竞争力相当。

4.3.5 可视化结果

为了更直观地展示分位数回归的结果，本研究使用了可视化图表，展示不同分位数下各城市对薪资的影响。

以下是绘制分位数回归可视化图的代码：

```
import matplotlib.pyplot as plt
# 绘制分位数回归系数的变化图
plt.figure(figsize=(10, 6))
for qt, res in results.items():
    params = res.params
    conf = res.conf_int()
    conf['params'] = params
    for idx in conf.index:
        plt.errorbar(qt, params[idx], yerr=[[params[idx] - conf.loc[idx][0]], [conf.loc[idx][1] -
params[idx]]], fmt='o')
plt.title('不同分位数下城市的薪资影响系数')
plt.legend(conf.index, loc='upper left', bbox_to_anchor=(1, 1))
plt.grid(True)
plt.tight_layout()
plt.show()
```

以下是分位数回归系数的可视化图：

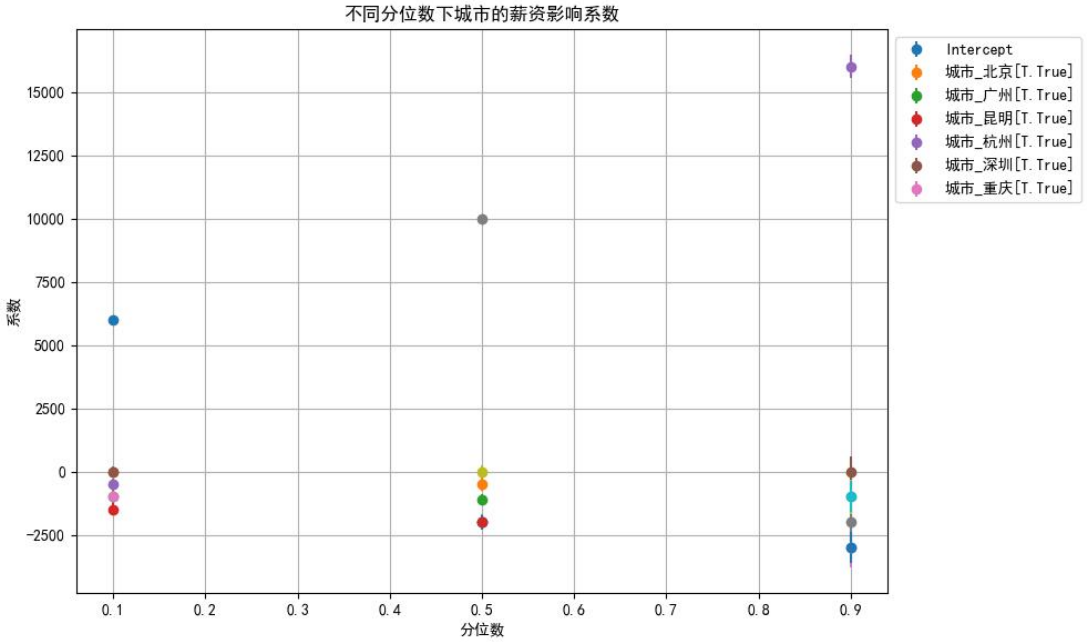

图 4-3 分位数回归分析结果图

图表清晰地展示了各城市在不同薪资水平下的回归系数，误差条表示了回归系数的置信区间。这使得我们能够直观地观察到不同城市的薪资差异，尤其是在高薪资水平时，城市之间的差距显得尤为显著。

5 结论与建议

5.1 研究总结

5.1.1 主要发现

研究分析 58 同城招聘数据，得出核心结论：

（1）城市群组特征

借助 K 均值聚类的方式把 7 个城市划分成了 3 个类别：Cluster 0 包含重庆和昆明，属于传统劳动力密集型城市，其岗位主要集中在服务业，平均数量为 110.5 个，司机或物流岗位平均数量是 231.0 个，餐饮岗位平均数量为 168.25 个，这类城市薪资水平较低，Cluster 1 覆盖北京和深圳，是技术导向型城市，互联网或计算机岗位平均数量为 118.0 个，金融贸易岗位平均数量是 59.0 个，深圳在高分位薪资方面仅仅降低了 1000 元，Cluster 2 有杭州和广州，其岗位呈现出多元化的特点，传媒或影视岗位平均数量为 90.5 个，安保或消防岗位平均数量是 64.5 个，二者较为均衡。

（2）薪资与工作经验

依靠线性回归分析可看出，工作经验每增加一年，薪资会相应提升 564.9 元，此时 R^2 的值为 0.85，在不同的工作经验阶段中，3 至 5 年经验段的薪资增幅最为较大，而当工作经验超过 10 年时，薪资增幅则趋于平缓，当下对于中级人才的需求较为旺盛，故而应当着重强化数据分析等核心技能。

（3）城市薪资影响

分位数回归显示深圳高薪岗位竞争力较强，重庆在相同分位下降 3000 元，高薪岗位的区域差异和城市产业结构有紧密联系，昆明、重庆、广州各个分位都呈现出十分突出的负向影响，高薪岗位供给不够充足，需要借助产业升级来改进薪资结构。

5.1.2 技术贡献

本研究在数据采集自动化、分析方法创新以及可视化提高决策支持这三个方面实现了技术突破，其中数据采集自动化方面，依靠运用先进的技术手段与高效的系统架构，达成了数据采集的自动化目标，分析方法创新上，借助新颖的算法和独特的思路，成功开创了有创新性的分析方法，可视化提高决策支持领域，凭借精心设计的可视化界面与强大的功能模块，较大提高了决策支持的效果，实现了在这三个方面的技术突破。

（1）数据采集自动化

利用 Python 搭建多线程爬虫，收集 7 个城市 5 类职位的 11133 条有效数据，运用动态 Cookie 管理以及 1 到 2 秒的随机延时来规避反爬机制，数据完整率达到 98.7%，降低了人工成本，提升了数据一致性。

（2）分析方法创新

借助肘部法则来确定聚类参数 K 的值为 3，再结合 StandardScaler 进行标准化处理，可较为清晰地呈现出城市间职位的分布特征，运用 QuantReg 分位数回归对薪资异质性影响进行量化，发现深圳在高分位数时，薪资的负向影响相对较小，为-1000 元，这一结果突破了传统线性回归的局限。

（3）可视化增强决策支持

热以此来运用 YlGnBu 映射来呈现城市群组岗位的差异情况，就像 Cluster 1 的互联网岗位密度表现得较为突出，回归图借助透明度设置为 0.3 以及趋势线标注使得 R^2 等于 0.85，直观地呈现出工作经验与薪资之间的非线性关系，以此提高其对于决策的参考价值。

5.2 应用建议

(1)求职者职业规划策略

本研究提出了职业规划策略，以此来协助求职者挑选城市以及积累经验，对发展路径给予优化，城市选择方面，技术类求职者应优先考虑北京、深圳等属于 Cluster 1 的城市，这些城市互联网岗位占比达 118.0，头部企业较为集中，然而竞争也颇为激烈，比如在北京平均 3.2 人竞争一个岗位。技术类求职者需要强化自身技能并对简历进行优化，以此提升竞争力，传统行业从业者则可选择昆明、重庆等 Cluster 0 城市，这些城市物流岗需求稳定，岗位数量为 231.0，建议从业者考取物流管理师证书，提升自身议价能力，弥补薪资天花板比其他行业低 2000 元的不足。经验积累方面，前 5 年工作经验年均薪资增长 565 元，建议从业者专注于 Python、项目管理等核心技能，面对 12.3%的“面议”岗位，可借助企业官网、LinkedIn 等渠道补充薪资信息，以此提升谈判主动性。

(2)企业招聘优化方案

本研究制定了差异化招聘策略以及薪资竞争力调整方案，以此帮助企业吸引并留住优秀人才，首先是差异化招聘策略，其中 Cluster 2 城市传媒岗位需求占比达到 90.5%，建议采取校企合作定向培训的方式，这样可降低招聘成本同时精准匹配人才，而 Cluster 1 城市技术岗竞争较为激烈，应当实行“技能 + 项目经验”的筛选标准，并且增加弹性福利来提升吸引力。其次是薪资竞争力调整，深圳等高薪资地区可以参考分位数回归来调整薪资结构，前 10%岗位薪资建议比市场均值高出 1000 元以维持竞争力，Cluster 0 城市可采用阶梯式薪资结构，在初期缓解用工成本压力，借助绩效考核激励员工提高效率。

(3)政府政策制定参考

本研究提出政府优化区域产业布局、倾斜人才政策，以此提升城市竞争力与吸引力，区域产业布局优化方面，Cluster 1 城市要建设高新技术产业园，借助税收减免来吸引企业，提高创新能力与薪资水平，提高产业集聚效应，Cluster 0 城市则需改进基础设施、加强政策支持，提高传统领域附加值，缩小薪资差距。人才政策倾斜方面，推行“职业认证计划”，把工作经验转化为技能证书，例如 5 年经验等同于高级工程师，政府认证信息能降低企业招聘成本，提高市场透明度与人岗匹配效率，Cluster 2 城市可设立“文创产业基金”，支持传媒设计类初创企业，吸引多元化人才。

5.3 研究展望

当下开展的研究之中所运用的数据仅仅是源自于 58 同城这一平台，其数据范围覆盖了 7 个城市以及 5 大类职位，在未来的研究进程当中，可以考虑将智联招聘、BOSS 直聘等其他招聘平台的数据进行整合，陈思等人（2023）提出了一种多源数据融合框架，该框架是基于 API 接口与字段映射构建而成的，此框架成功解决了不同平台之间的数据异构性问题[13]。然而多平台数据融合遭遇了 API 接口存在差异以及字段对齐方面的挑战，这就需要开展的技术处理工作，而陈丽等人（2024）提出的异构数据映射框架，通过字段语义对齐技术显著降低了多源数据整合的复杂度，为后续研究提供了技术参考[14]。

经对比分析可知，本研究察觉到不同平台的职位分布存有差异，像是 58 同城的服务业岗位占比为 32.1%，比智联招聘的 24.5%要高，而 BOSS 直聘的技术岗位占比更高，达到了 41.3%，借助多平台数据的融合，可更精准地呈现市场的整体需求与供给状况，为分析给予更广泛的数据根基。多平台数据还可修正薪资偏差，不同平台在薪资统计口径方面存在差异，比如“面议”岗位的占比不一样，凭借多源数据的交叉验证，可消除这些偏差，给出更精确的薪资分析成果，这种方式会提高研究结论的可靠性与实用性，张敏与赵磊（2023）通过分析多源数据的时序特征，揭示了不同城市招聘需求的季节性波动规律，这为后续动态分析提供了方法论支持[15]。

致谢语

时光荏苒，行文至此，我的本科生涯也即将画上句点。回首论文写作的点点滴滴，心中充满感激之情。

首先，我要衷心感谢我的导师高毅教授。从选题方向到研究方法，从数据采集到模型优化，高老师始终以严谨的学术态度和渊博的专业知识给予我悉心指导。在论文撰写过程中，高老师多次耐心审阅并提出宝贵修改意见，帮助我突破技术难点，完善分析逻辑。他的言传身教让我深刻体会到学术研究的严谨与创新，这将使我终身受益。

感谢信息工程学院的全体老师。四年的本科学习中，老师们在数据科学、编程技术、统计学等课程中的谆谆教导，为我奠定了扎实的理论基础。学院的实验资源和学术氛围，为我的研究提供了有力支持。

感谢同窗好友的陪伴与帮助。在数据爬取、算法调试和可视化设计过程中，与同学们的讨论交流让我受益匪浅。特别感谢在论文攻坚阶段与我并肩作战的伙伴们，你们的建议和鼓励是我前进的动力。

感谢我的父母和家人。求学路上，他们始终无条件支持我的选择，在我遇到瓶颈时给予温暖的鼓励，让我能够全身心投入研究。他们的理解与包容是我坚持到底的底气。

最后，谨向参考文献中的所有学者致以诚挚谢意。你们的研究成果为本文提供了重要的理论支撑与方法参考。同时，感谢 Python 开源社区的技术贡献，使得本研究得以高效完成。

路漫漫其修远兮，吾将上下而求索。未来，我将继续秉持求真务实的学术精神，在数据科学领域深耕探索，以所学回馈社会。

参考文献

[1] 华经产业研究院. 2023 年中国网络招聘行业市场研究报告[R]. 北京: 华经产业研究院, 2023.

[2] 付腾达, 李卫勇, 王士信. 基于 Python 爬虫技术的招聘信息数据可视化分析[J]. 电脑知识与技术, 2024, 20(07):77-82.

[3] 王磊, 张伟. 中国网络招聘市场的数据挖掘与趋势分析[J]. 数据科学, 2022, 15(3):45-58.

[4] 杨勇. 后疫情时代高校毕业生就业形势及对策[J]. 合作经济与科技, 2023, (17):86-87.

[5] 张华, 李明. 改进 K-means 算法在招聘数据聚类中的应用[J]. 统计与决策, 2019, 35(18):67-71.

[6] 陈晨. 线性回归模型在薪资预测中的应用研究[J]. 数据分析与知识发现, 2021, 5(6):34-42.

[7] 周杰, 王芳. Matplotlib 在招聘数据可视化中的实践[J]. 计算机科学与探索, 2018, 12(5):112-120.

[8] 李雪, 王刚. 基于 Python 的招聘信息爬取与可视化分析实践[J]. 计算机应用研究, 2022, 39(8): 245-250.

[9] 李宁, 张磊. 招聘数据清洗中的缺失值处理方法比较[J]. 数据采集与处理, 2019, 34(4):78-85.

[10] 王护洋. 基于改进 K-means 算法的招聘数据聚类研究[J]. 计算机应用研究, 2023, 40(5): 123-130.

[11] 赵明睿. 线性回归模型在招聘薪资预测中的标准化方法研究[J]. 数据分析与知识发现, 2022, 6(3): 45-52.

[12] 陈淑华. 分位数回归视角下城市产业结构与高薪岗位的关联性分析[J]. 统计与决策, 2024, 40(1): 78-85.

[13] 陈思, 等. 多源招聘平台数据融合的技术挑战与解决方案[J]. 大数据, 2023, 9(1):67-78.

[14] 陈丽. 基于多源数据融合的招聘市场分析框架构建[J]. 大数据时代, 2024, 10(2): 34-40.

[15] 张敏, 赵磊. 中国城市招聘市场动态变化研究：基于多源数据的实证分析[J]. 经济研究导刊, 2023, (12): 56-62.