课程编号：A0801051420

《Python 数据分析》实验报告

姓　　名		学　　号	20227091
班　　级	软件 2203	指　导　教　师	于鲲鹏
实　验　名　称	《Python 数据分析》		
开　设　学　期	2023-2024 春季学期		
开　设　时　间	第 1 周——第 12 周		
报　告　日　期	2025 年 5 月 1 日		
评　定　成　绩		评　定　人	于鲲鹏
		评　定　日　期	2025 年 5 月 15 日

东北大学软件学院

一、实验目的

1、掌握 Python 数据分析工具的使用方法：

熟练运用 Python 编程语言及相关数据分析库（如 NumPy、pandas、Matplotlib、Seaborn、Scikit-learn 等），提升编程实践能力；

学会使用集成开发工具（如 PyCharm、Visual Studio Code），提高代码编写和调试效率。

2、培养处理实际数据的能力：

数据采集：学习使用网络爬虫技术（如 Selenium、Scrapy）和 API 接口获取数据，能够从不同来源收集与 Python 岗位相关的信息，如招聘网站、证券之星等；

数据清洗：对采集到的数据进行预处理，包括去除重复值、处理缺失值、纠正错误数据、异常值检测与处理等，提高数据质量；

数据转换：对数据进行标准化、归一化、特征编码等操作，使其满足后续分析和建模的要求；

数据集成：将来自不同源的数据进行合并、连接和组合，形成统一的数据集，为综合分析提供基础。

3、提升数据挖掘和机器学习模型的应用能力：

学习常用的数据挖掘算法，如回归分析（线性回归、多项式回归等）、分类算法（KNN、决策树、随机森林等）、聚类算法（KMeans、DBSCAN 等）等，掌握其原理和应用场景；

能够根据实际问题选择合适的算法，构建数据挖掘模型，如建立薪资预测模型、岗位推荐模型、上市公司聚类模型等；

对模型进行训练、评估和优化，通过调整参数、改进算法等手段提高模型的性能和准确性，学会使用交叉验证、网格搜索等技术选择最佳模型。

4、提高数据分析报告的撰写能力：

学会将实验过程、分析结果和结论以报告的形式清晰、准确地表达出来；

掌握数据分析报告的结构和写作规范，包括实验目的、内容、环境、过程与分析、创新点、总结等部分；

能够运用图表（如折线图、柱状图、饼图、散点图等）直观地展示数据分析结果，增强报告的可读性和说服力；

提高对数据分析结果的解读和总结能力，能够从中提炼出有价值的信息和见解，为实际决策提供支持。

5、增强解决复杂实际问题的能力，形成对 python 岗位就业环境的认识：

通过对 Python 岗位数据的采集和分析，了解当前 Python 岗位的市场需求、薪资水平、技能要求等，为求职者提供参考，使其更好地规划职业发展；

结合股票市场数据，探索岗位与股票市场的关系，培养从多角度分析问题的能力；

通过对上市公司进行聚类分析，帮助求职者选择适合的公司，提高就业成功率；

二、实验内容

实验 1：数据采集

1、确定数据分析方向，选择与 Python 岗位相关的信息作为研究目标；

2、使用 Selenium 和 BeautifulSoup 编写爬虫，爬取拉勾网的 Python 岗位信息；

3、使用 Scrapy 框架爬取证券之星的股票信息；

4、利用 AKShare API 获取股票的日际数据；

5、将爬取的数据存储到 MySQL 数据库和本地文件（CSV、Excel、JSON、XML、HTML）中。

实验 2：数据清洗与预处理
1、合并、连接和组合不同来源的数据集；
2、清洗无效和错误数据，处理缺失值、重复值和异常值；
3、对数据进行标准化、缩放、归一化和特征编码等预处理；
4、根据数据挖掘算法的需求，对数据集进行必要的变换操作。

实验 3：数据挖掘
1、对数据集进行统计分析，计算求和、均值、分组等指标；
2、构建回归、分类和聚类等数据挖掘模型；
3、使用训练数据集训练模型，并通过测试数据集验证模型；
4、根据模型评估结果优化算法参数，提高模型性能。

实验 4：数据可视化
1、选择合适的可视化方式，如折线图、柱状图、饼图、散点图等，展示数据分析和挖掘结果；
2、利用 Matplotlib、Seaborn 等库创建可视化图表，直观呈现数据特征和分析结果；
3、通过可视化结果解读数据，提炼有价值的信息。

三、实验环境

处理器：AMD Ryzen 7 6800H with Radeon Graphics
机带 RAM：16.0 GB (15.2 GB 可用)
操作系统：Windows 11 24h2
IDE：Visual Studio Code 1.100.1

爬虫部分 python 环境：
python：3.10
selenium：4.32.0
scrapy：2.12.0
scrapy-selenium：0.0.7
numpy：2.2.5

数据分析部分 python 环境：
python：3.9
numpy：1.26.4
pandas：2.2.3
scikit-learn：1.3.1
scipy：1.13.1
seaborn：0.13.2

四、实验过程与分析

数据获取：

在本次实验中，我的数据来源主要有三：

1、使用爬虫爬取来自拉勾网（https://www.lagou.com/wn/）对于“python”的搜索所搜索到的岗位及其信息。

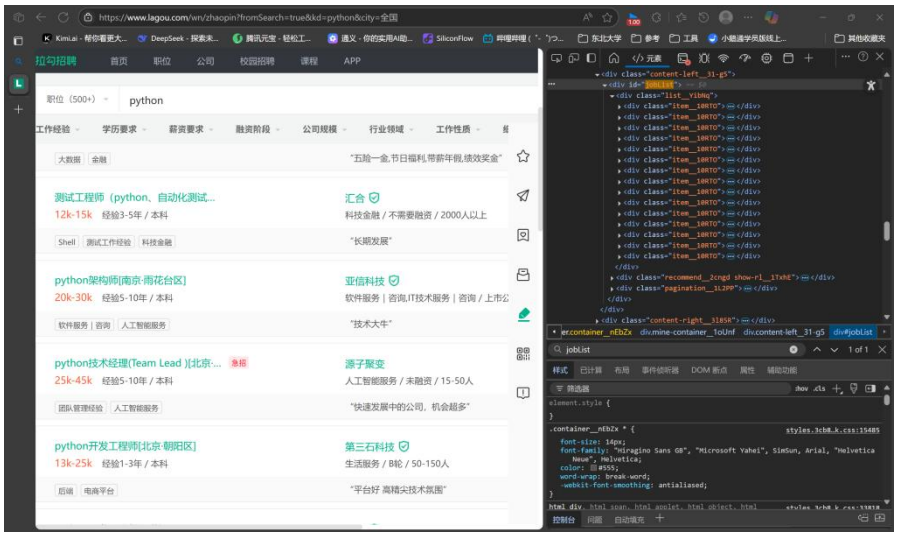
2、使用 scrapy 爬虫爬取来自证券之星（https://www.stockstar.com/）的 2025 年 5 月 9 日的所有股票的流通市值、总市值、流通股本、总股本等信息。

3、使用 API 获取 AKShare（https://www.akshare.xyz/index.html）部分股票的日际数据。

首先介绍爬虫 1。

我使用 selenium 来模拟人类用户的行为，使用 BeautifulSoup 解析 html 信息。

使用浏览器的检查功能查看网页，发现需要爬取的数据在一个 id 为 jobList 的元素下。



jobList 里面，class 为 list__YibNq 的元素存储了这页工作的信息，class 为 item__10RTO 的元素存储了某一项工作的信息。除此之外，jobList 还存储了推荐公司、推荐城市等其他信息，这些信息我并不需要。

```
<div id="jobList">
    <div class="list__YibNq">
        <div class="item__10RTO">
            <div class="item-top__1Z3Zo">
                <div class="position__21iOS">
                    <div class="p-top__1F7CL"><a
                            id="openWinPostion">Python开发工程师<!-- -->[天津·南开区]</a></div>
                    <div class="p-bom__JlNur"><span class="money__3Lkgq">11k-20k</span>经验1-3年 /
                        本科</div>
                </div>
                <div class="company__2EsC8">
                    <div class="company-name__2-SjF"><a>帮伯乐</a></div>
                    <div class="industry__1HBkr">专业服务 | 咨询,营销服务 | 咨询 / 不需要融资 / 15-50人</div>
                </div>
                <div class="com-logo__1QOwC"><img
                        src="https://www.lgstatic.com/thumbnail_120x120/10/93c0f4637f0b4714948a3017418ff042.png"
                        alt="帮伯乐" /></div>
            </div>
            <div class="item-bom__cTJhu">
                <div class="ir___QwEG"><span>营销服务 | 咨询</span><span>专业服务 | 咨询</span></div>
                <div class="il__3lk85">“方案”</div>
            </div>
        </div>
        <div class="item__10RTO">
            <div class="item-top__1Z3Zo">
                <div class="position__21iOS">
                    <div class="p-top__1F7CL"><a
                            id="openWinPostion">Python开发工程师<!-- -->[武汉·东西湖区]</a></div>
                    <div class="p-bom__JlNur"><span class="money__3Lkgq">10k-20k</span>经验3-5年 /
                        硕士</div>
                </div>
                <div class="company__2EsC8">
```

所以，爬取拉勾网的思路是：

1、登录，避免在翻页搜索时要求登录，打断自动化爬虫。

2、请求某一页。

3、根据 id 解析 jobList 元素。

4、解析每一个 class 为 list__YibNq 的元素。得到的信息是每个工作信息，解析这些信息，并保存。

首先进行一些基本的配置，比如禁用 GPU 加速、隐藏自动化特征等。

```
options = webdriver.EdgeOptions()
options.add_argument("--disable-gpu")  # 禁用 GPU 加速
options.add_argument("--disable-blink-features=AutomationControlled")  # 隐藏自动化特征
```

随后，请求拉勾网首页，并在这里程序暂停，此时，浏览器会弹出，全屏，并跳转到拉勾网首页，在这里，我会手动进行登录，避免之后分页爬取时，会被网站要求进行登录。

```
driver = webdriver.Edge(options=options)
driver.maximize_window()
# 请求拉勾网首页
driver.get("https://www.lagou.com/")
# 暂停，进行手动登录。
input("手动登录完成按回车")
```

下一步，请求每一页并进行数据解析，使用 get 方法请求对应的页，尝试获取 id 为 jobList 的元素，最多等待 10 秒，否则进行刷新。获取上述元素后，获取 jobList 的 html 源码，使用 BeautifulSoup 对 html 源码进行解析。在 jobList 中寻找所有 class 为 item__10RTO 的元素，这些是该页某个工作的信息。遍历这些信息，通过 class 进行解析：

描述	class	变量名和表头
岗位名和工作地址	.p-top__1F7CL a	job_address
薪资	.money__3Lkgq	salary
要求	.p-bom__JlNur	requirements
公司名称	.company-name__2-SjF a	company
公司信息	.industry__1HBkr	company_info
岗位信息	.ir___QwEG span	job_info
亮点	.il__3lk85	highlights

在解析时，因为会遇到部分字段为空的情况，所以使用 try-except 代码块进行包裹，并在发生异常时，手动置空。

最后将数据存储到数据库中。休眠一段时间，爬取下一页，避免被反爬机制阻止。分页爬虫的主要代码如下：

```
while page <= all_page:
    while retry_count < max_retries:
driver.get(f"https://www.lagou.com/wn/jobs?cl=false&fromSearch=true&kd=python&pn={page}")
        try:
            # 显式等待元素加载，最多 10 秒
            job_list = WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.ID,
"jobList")))
            break  # 成功获取元素后退出循环
        except TimeoutException:
            print(f"第{retry_count+1}次尝试超时，刷新页面重试...")
            driver.refresh()
            retry_count += 1
            time.sleep(10)  # 避免连续刷新过快
    # 检查是否成功获取元素
    if retry_count < max_retries:
        html = driver.page_source
        soup = BeautifulSoup(html, "html.parser")
```

```
        for item in soup.select("div.item__10RTO"):
            try:
                # 岗位名和工作地址
                job_address = item.select_one(".p-top__1F7CL a").text.strip()
            except Exception as e:
                print('异常 ', e)
                job_address = ''
            try:
                # 薪资
                salary = item.select_one(".money__3Lkgq").text.strip()
            except Exception as e:
                print('异常 ', e)
                salary = ''

            ……

            cur.execute(
                sql,
                (
                    job_address,
                    salary,
                    requirements,
                    company,
                    company_info,
                    job_info,
                    highlights,
                ),
            )
    else:
        print("已达到最大重试次数，元素仍未加载")
    conn.commit()
    time.sleep(50 + random.uniform(0, 20))
    page += 1
```

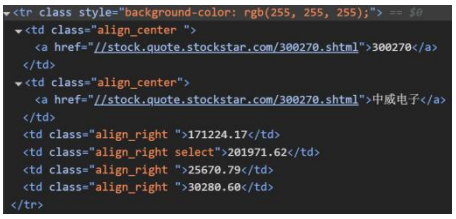
展示爬虫结果如下：

id	job_address	salary	requirements	company	company_info	job_info	highlights
# int	varchar(255)	varchar(255)	varchar(255)	varchar(255)	varchar(255)	varchar(255)	varchar(255)
330	Python软件开发工	6k-8k	6k-8k经验不限 / 不	河南端琪实业发展	新零售 / 不需要融资	新零售	"五险一金,包住,绩
331	python开发工程师	10k-18k	10k-18k经验1-3年	帮伯乐	专业服务｜咨询,营销	营销服务｜咨询/专	"方案"
332	python开发工程师	15k-30k	15k-30k经验3-5年	浙江巨能环境工程	能源｜矿产｜环保 /	能源｜矿产｜环保	"五险一金,绩效奖金
333	软件工程师（C++,	15k-25k	15k-25k经验3-5年	上海全晓信息技术			"12天带薪年假,免费
334	对日JAVA、Pytho	18k-35k	18k-35k经验5-10年	上海网擎信息系统	网络通信 / 不需要融	网络通信	"五险一金,定期体检
335	Python工程师[青	6k-8k	6k-8k经验3-5年 / ;	青岛启环国际贸易	贸易｜进出口 / 不需	贸易｜进出口	"五险一金,专业培训
336	python开发工程师	9k-14k	9k-14k经验1-3年 /	维普资讯	电商平台,内容社区 /	内容社区/电商平台	"五险一金,绩效奖金
337	Python开发工程师	12k-20k	12k-20k经验1-3年	辰而将行软件	软件服务｜咨询,数据	软件服务｜咨询/数	"五险一金,弹性工作
338	python开发[深圳-i	15k-18k	15k-18k经验3-5年	紫川软件	其他 / 不需要融资 /		"五险一金,定期体检
339	Python开发工程师	11k-20k	11k-20k经验1-3年	帮伯乐	专业服务｜咨询,营销	营销服务｜咨询/专	"方案"
340	python自动化测试	20k-25k	20k-25k经验5-10年	万宝盛华企业管理	服务业 / 不需要融资	服务业	"五险一金,定期体检
341	python后端开发(8	10k-15k	10k-15k经验1-3年	深圳市科雅智能数	其他 / 不需要融资 /		"购买社保,方案,技
342	Python高级工程师	18k-25k	18k-25k经验5-10年	兰台信息	专业服务｜咨询 / 不	专业服务｜咨询	"五险一金,绩效奖金
343	Python开发工程师	17k-19k	17k-19k经验5-10年	博为峰			"五险一金,周末双休
344	python开发工程师	15k-18k	15k-18k经验3-5年	深智无限	人工智能服务 / 未融	人工智能服务	"办公环境优，氛围
345	python自动化测试	20k-25k	20k-25k经验5-10年	万宝盛华企业管理	服务业 / 不需要融资	服务业	"五险一金,定期体检
346	python后端开发(8	10k-15k	10k-15k经验1-3年	深圳市科雅智能数	其他 / 不需要融资 /		"购买社保,方案,技
347	Python高级工程师	18k-25k	18k-25k经验5-10年	兰台信息	专业服务｜咨询 / 不	专业服务｜咨询	"五险一金,绩效奖金
348	python开发工程师	28k-35k	28k-35k经验5-10年	岩山科技	金融业 / 不需要融资	金融业	"五险一金,绩效奖金
349	python高级开发工	30k-40k	30k-40k经验5-10年	Veeva System维	软件服务｜咨询,IT技	JAVA/软件服务｜i	"世界前 50 强制药
350	Python开发工程师	17k-19k	17k-19k经验5-10年	博为峰			"五险一金,周末双休
351	python开发实习生	3k-4k	3k-4k经验不限 / 本	玖乾奕软件	IT技术服务｜咨询 /	C语言/科技金融/金	"实习可转正机会"
352	python开发工程师	15k-18k	15k-18k经验3-5年	深智无限	人工智能服务 / 未融	人工智能服务	"办公环境优，氛围
353	海外爬虫开发工程i	15k-23k	15k-23k经验1-3年	特易资讯	其他 / 不需要融资 /		"五险一金,绩效奖金
354	python开发工程师	6k-10k	6k-10k经验3-5年 /	瑞丰科技	农林牧渔 / 不需要融	农林牧渔	"五险一金,绩效奖金
355	python开发[北京-i	14k-18k	14k-18k经验1-3年	上海韦创立企业管	专业服务｜咨询,营销	营销服务｜咨询/专	"发展前景佳、团队
356	python数据开发工	15k-30k	15k-30k经验1-3年	北京指南针	科技金融 / 上市公司	科技金融	"上市公司"
357	web开发工程师（p	10k-18k	10k-18k经验不限 /	担路网	移动互联网,企业服务	软件服务｜咨询	"弹性工作 节假日补

接着介绍爬虫 2。

我使用 scrapy 框架编写爬虫，对证券之星（https://quote.stockstar.com/stock）中的各个交易所的 5 月 9 日的股票信息进行爬取。

使用浏览器的检查功能，发现表格数据是通过下面的 html 实现的。每一行数据在 tr 标签下，每一个数据在 td 标签下的 a 标签下或者是 td 标签下。



于是，可以将 items.py 写作：

```
import scrapy
from scrapy.loader import ItemLoader
from itemloaders.processors import TakeFirst

class StockstarItemLoader(ItemLoader):
    #自定义 itemloader，用于存储爬虫所抓取的字段内容 de
    default_output_processor = TakeFirst()

class StockstarItem(scrapy.Item): #建立相应的字段
    # define the fields for your item here like:
    # name = scrapy.Field()
    code = scrapy.Field()        #股票代码
    abbr = scrapy.Field()        #股票简称
    traded_market_value = scrapy.Field()   #流通市值
    aggregate_market_value = scrapy.Field()    #总市值
    capital_stock_in_circulation = scrapy.Field()      #流通股本
    total_stock_issue = scrapy.Field()   #总股本
```

其中，code 表示股票代码，abbr 股票表示简称，traded_market_value 表示流通市值，aggregate_market_value 表示总市值，capital_stock_in_circulation 表示流通股本，total_stock_issue 表

示总股本。

将 stock.py 写作：

```
import scrapy
from spider.items import StockstarItem,StockstarItemLoader

class StockSpider(scrapy.Spider):
    name = 'stock'                                          #定义爬虫名称
    allowed_domains = ['quote.stockstar.com']       #定义爬虫域
    start_urls = ['http://quote.stockstar.com/stock/ranklist_a_3_1_1.html']
    #定义开始爬虫链接
    def parse(self,response):       #撰写爬虫逻辑
        page = int(response.url.split("_")[-1].split(".")[0])   #抓取页码
        item_nodes = response.css('#datalist tr')
        for item_node in item_nodes:
            #根据 item 文件中所定义的字段内容，进行字段内容的抓取
            item_loader = StockstarItemLoader(item=StockstarItem(),selector= item_node)
            item_loader.add_css("code","td:nth-child(1) a::text")
            item_loader.add_css("abbr", "td:nth-child(2) a::text")
            item_loader.add_css("traded_market_value", "td:nth-child(3)::text")
            item_loader.add_css("aggregate_market_value", "td:nth-child(4)::text")
            item_loader.add_css("capital_stock_in_circulation", "td:nth-child(5)::text")
            item_loader.add_css("total_stock_issue", "td:nth-child(6)::text")
            stock_item = item_loader.load_item()
            yield stock_item
        if item_nodes:
            next_page = page + 1
            next_url = response.url.replace("{0}.html".format(page),"{0}.html".format(next_page))
            yield scrapy.Request(url=next_url,callback=self.parse)
```

这是爬虫的主要逻辑，通过 add_css 函数分别找到 code、abbr、traded_market_value、aggregate_market_value、capital_stock_in_circulation、total_stock_issue 自动，并通过 replace，不断对 url 最后一个数字，也就是表示页数的数字进行替换，并使用 Request 请求，以实现翻页功能。

随后对 pipelines.py 进行编码，实现 StockstarPipeline 类，编写存储数据库的相关业务。

from_crawler 方法从项目的 setting.py 中获取配置，初始化数据库的有关参数。

```
    @classmethod
    def from_crawler(cls, crawler):
        return cls(
            host=crawler.settings.get('MYSQL_HOST'),
            port=crawler.settings.get('MYSQL_PORT'),
            user=crawler.settings.get('MYSQL_USER'),
            password=crawler.settings.get('MYSQL_PASSWORD'),
            db=crawler.settings.get('MYSQL_DB'),
        )
```

open_spider 方法根据类保存的数据库参数，初始化数据库连接，这将在打开爬虫时进行。

```
    def open_spider(self, spider):
```

```
        self.connection = pymysql.connect(
            host=self.host,
            port=self.port,
            user=self.user,
            password=self.password,
            db=self.db,
            charset='utf8mb4',
            cursorclass=cursors.DictCursor
        )
        self.cursor = self.connection.cursor()
```

close_spider 方法用于关闭数据库连接，这将在关闭爬虫时进行。

```
    def close_spider(self, spider):
        self.connection.close()
```

process_item 方法在获取到数据时，将每一个数据插入数据库。先定义了 SQL 语句，这是一个 INSERT 的插入语句。使用 execute 方法执行语句，最后使用 commit 方法确认数据提交到数据库。

```
    def process_item(self, item, spider):
        # 插入数据到数据库
        try:
            sql = '''
                INSERT INTO stock (code, abbr, traded_market_value, aggregate_market_value, capital_stock_in_circulation, total_stock_issue)
                VALUES (%s, %s, %s, %s, %s, %s)
            '''
            self.cursor.execute(sql, (
                item['code'],
                item['abbr'],
                item['traded_market_value'],
                item['aggregate_market_value'],
                item['capital_stock_in_circulation'],
                item['total_stock_issue']
            ))
            self.connection.commit()
        except Exception as e:
            spider.logger.error(f"Error inserting item into MySQL: {e}")
            self.connection.rollback()
            raise DropItem(f"Failed to insert item into MySQL: {item}")
```

在 setting.py 中进行设置。包括设置爬虫时间和协议，数据库配置等。

```
ROBOTSTXT_OBEY = False
DOWNLOAD_DELAY = 0.25
ITEM_PIPELINES = {
    'spider.pipelines.StockstarPipeline': 300,
}
MYSQL_HOST = 'localhost'
MYSQL_PORT = 3306
```

```
MYSQL_USER = 'root'
MYSQL_PASSWORD = '123456'
MYSQL_DB = 'spider'
```

在项目的根目录/spider 下，在配置好 scrapy 的 python 环境中，执行命令开始爬虫。

```
scrapy crawl stock
```

爬虫结果展示如下：

id # int	code varchar(255)	abbr varchar(255)	traded_market_value varchar(255)	aggregate_market_val varchar(255)	capital_stock_in_circul varchar(255)	total_stock_issue varchar(255)
1	000001	平安银行	21637212.61	21637598.79	1940557.18	1940591.82
2	000002	万 科A	6588082.52	8089021.02	971693.59	1193070.95
3	000004	*ST国华	97620.70	102329.96	12628.81	13238.03
4	000006	深振业A	869391.88	869396.81	134998.74	134999.50
5	000007	全新好	245978.11	245978.11	34644.80	34644.80
6	000008	神州高铁	746964.33	747003.86	271623.39	271637.77
7	000009	中国宝安	2048798.55	2071108.81	255143.03	257921.40
8	000010	美丽生态	150422.09	328800.56	52595.14	114965.23
9	000011	深物业A	441712.98	500026.46	52647.55	59597.91
10	000012	南 玻A	938489.05	1470861.52	195926.73	307069.21
11	000014	沙河股份	265766.75	265766.75	24204.62	24204.62
12	000016	深康佳A	860564.06	1297882.57	159659.38	240794.54
13	000017	深中华A	197243.21	448659.39	30298.50	68918.49
14	000019	深粮控股	281778.51	780266.37	41621.64	115253.53
15	000020	深华发A	217036.14	339227.15	18116.54	28316.12
16	000021	深科技	2836850.31	2837148.24	156042.37	156058.76
17	000025	特 力A	673222.04	738833.96	39277.83	43105.83
18	000026	飞亚达	423861.95	471497.78	36476.93	40576.40
19	000027	深圳能源	3135119.95	3135119.95	475738.99	475738.99
20	000028	国药一致	1189397.64	1384733.91	47805.37	55656.51
21	000029	深深房A	1497988.80	1699588.80	89166.00	101166.00
22	000030	富奥股份	1037448.47	1067627.21	169241.19	174164.31
23	000031	大悦城	1048756.44	1123014.09	400288.72	428631.33
24	000032	深桑达A	1452008.10	2562684.19	64476.38	113795.92
25	000034	神州数码	2441663.92	2921858.85	59436.80	71126.07
26	000035	中国天楹	1072200.99	1105346.17	242579.41	250078.32
27	000036	华联控股	536218.39	537184.12	148126.63	148393.40
28	000037	深南电A	291122.10	517773.07	33890.82	60276.26

最后介绍，通过 AKShare 的 API 获取我想要分析的股票的日际数据。这主要通过 akshare 包实现的，其中 stock_zh_a_hist 方法可以从网络中获取相应股票的 dataframe。symbol 参数表示股票代码；period 参数表示时间粒度，比如日、周、月、季度、年等；start_date 参数表示开始时间；end_date 参数表示结束时间；

```
import akshare as ak
for stock in lst_stock:
    print(stock)
    stock_df = ak.stock_zh_a_hist(
        symbol=stock,
        period="daily",
        start_date="20150430",
        end_date="20250430",
    )
    stock_df.to_csv(stock + '.csv')
    print(stock_df.head())
```

获取的信息展示如下例：

```
,日期,股票代码,开盘,收盘,最高,最低,成交量,成交额,振幅,涨跌幅,涨跌额,换手率
0,2015-04-30,002195,54.57,56.45,58.98,54.53,220419,1256763776.0,8.15,3.41,1.86,15.01
1,2015-05-04,002195,55.8,53.69,55.8,52.1,120101,645072352.0,6.55,-4.89,-2.76,8.18
2,2015-05-05,002195,53.85,52.3,53.85,52.0,84514,446226240.0,3.45,-2.59,-1.39,5.76
3,2015-05-06,002195,52.42,53.05,55.25,52.42,111473,600553568.0,5.41,1.43,0.75,7.59
4,2015-05-07,002195,52.0,51.28,52.77,51.0,79110,410317552.0,3.34,-3.34,-1.77,5.39
5,2015-05-08,002195,52.21,56.41,56.41,52.21,291370,1619969104.0,8.19,10.0,5.13,19.84
6,2015-05-11,002195,58.65,62.05,62.05,57.02,370773,2218886848.0,8.92,10.0,5.64,18.07
7,2015-05-12,002195,62.47,60.81,63.86,59.0,346048,2116798176.0,7.83,-2.0,-1.24,16.86
8,2015-05-13,002195,58.6,59.09,61.15,56.55,233884,1376787568.0,7.56,-2.83,-1.72,8.99
9,2015-05-14,002195,58.7,55.53,58.7,53.99,359636,2003110528.0,7.97,-6.02,-3.56,13.83
10,2015-05-15,002195,54.75,56.6,58.24,53.36,228883,1281289248.0,8.79,1.93,1.07,8.8
11,2015-05-18,002195,55.79,58.45,58.89,55.15,230509,1331591072.0,6.61,3.27,1.85,8.86
12,2015-05-19,002195,58.1,64.3,64.3,58.1,425333,2665856464.0,10.61,10.01,5.85,16.35
13,2015-05-20,002195,69.0,70.73,70.73,67.6,510561,3582187680.0,4.87,10.0,6.43,19.63
14,2015-05-21,002195,70.0,74.1,77.1,66.78,543046,3889242416.0,14.59,4.76,3.37,20.88
15,2015-05-22,002195,73.77,69.6,73.77,69.3,355438,2520740448.0,6.03,-6.07,-4.5,13.67
16,2015-05-25,002195,62.64,62.64,62.64,62.64,139399,873195344.0,0.0,-10.0,-6.96,5.36
17,2015-05-26,002195,61.0,63.99,65.5,61.0,387647,2467391680.0,7.18,2.16,1.35,14.91
18,2015-05-27,002195,63.99,61.87,65.49,61.1,273233,1710674976.0,6.86,-3.31,-2.12,10.51
19,2015-05-28,002195,61.4,64.0,67.7,60.28,364730,2315782592.0,11.99,3.44,2.13,14.02
20,2015-05-29,002195,62.29,62.87,66.3,57.6,252532,1591264160.0,13.59,-1.77,-1.13,9.71
21,2015-06-01,002195,62.7,67.87,68.47,62.5,314424,2074869088.0,9.5,7.95,5.0,12.09
22,2015-06-02,002195,67.0,70.17,72.2,65.2,330203,2253053696.0,10.31,3.39,2.3,12.7
```

数据存储：

数据库（如 MySql）和文件（如 csv 文件、html 文件、json 文件、xlsx 文件、xml 文件等）都是存储数据的有效方式。

pandas 库提供了各种存储方式的存储和保存功能。所以可以用于进行各种存储方式的转换。比如我爬取的数据存储在数据库里，使用下面的代码将数据库里面的数据加载为 dataframe：

```
conn = mysql.connector.connect(
    host="127.0.0.1",
    user="root",
    password="123456",
    database="spider",
)
df = pd.read_sql("SELECT * FROM lagou_jobs", conn)
```

随后，保存为 csv 文件：

```
df.to_csv("lagou_jobs.csv", index = False)
```

保存为 xlsx 文件：

```
df.to_excel("lagou_jobs.xlsx", index = False)
```

保存为 json 文件：

```
df.to_json("lagou_jobs.json", orient = "records")
```

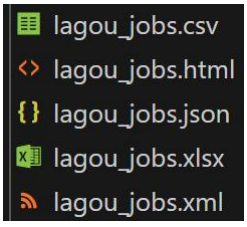
保存为 xml 文件：

```
df.to_xml("lagou_jobs.xml")
```

保存为 html 文件：

```
df.to_html("lagou_jobs.html")
```

得到各种存储形式：



数据清洗：

首先加载爬虫 1 获取的数据：

```
df = pd.read_csv('lagou_jobs.csv')
```

	id	job_address	salary	requirements	company	company_info	job_info	highlights
0	330	Python软件开发工程师[郑州·管城回族区]	6k-8k	6k-8k经验不限 / 不限	河南端琪实业发展有限公司	新零售 / 不需要融资 / 150-500人	新零售	"五险一金,包住,绩效奖金"
1	331	python开发工程师[上海·闵行区]	10k-18k	10k-18k经验1-3年 / 本科	帮伯乐	专业服务｜咨询,营销服务｜咨询 / 不需要融资 / 15-50人	营销服务｜咨询/专业服务｜咨询	"方案"
2	332	python开发工程师[杭州·滨江区]	15k-30k	15k-30k经验3-5年 / 本科	浙江巨能环境工程有限公司	能源｜矿产｜环保 / 不需要融资 / 50-150人	能源｜矿产｜环保	"五险一金,绩效奖金,定期体检"
3	333	软件工程师（C++/C, Java/C#, python）[南京·溧水区]	15k-25k	15k-25k经验3-5年 / 本科	上海全晓信息技术有限公司	NaN	NaN	"12天带薪年假,免费三餐,定期体检"
4	334	对日JAVA、Python高级工程师[上海·杨浦区]	18k-35k	18k-35k经验5-10年 / 本科	上海网擎信息系统有限公司	网络通信 / 不需要融资 / 少于15人	网络通信	"五险一金,定期体检,员工旅游"

要进行的数据处理有：

1、初步特征工程，删除绝对不会用到的列。

2、重复值处理

3、缺失值处理

4、异常值处理

5、数据拆分

（1）对“job_address”进行数据分割，分割为工作列和地址列。

（2）对“salary”进行数据分割，分割为最小薪资“min_salary”和“max_salary”。

（3）对“requirements”职位要求列分割为经验要求“experience”和学历要求“education”。

（4）对“company_info”进行数据分割，取最后的两个要素融资情况“financing”和公司规模“size”。

（5）统计职位亮点“highlights”中有几种，并对最常见的几种亮点进行独热编码，包括五险一金“insurance”，奖金“bonus”，体检“examination”，周末双休“weekend”，发展前景“prospect”，团队“team”，旅游“travel”，培训“train”。

首先，删除绝对不会用到的列：

```
df1 = df.drop('id', axis=1)
df1 = df1.drop('job_info', axis=1)
```

然后，进行重复值处理：

```
df2 = df1.drop_duplicates()
```

下面先介绍数据拆分。

拆分“job_address”列，并提取·前面的市，得到“city”列，将“job”和“city”合并到原 dataframe，并删除“job_address”列。

```
# 拆分 job_address 列，并提取 · 前面的市
tmp = df2['job_address'].str.extract(r'(.*)\[(.*)\]')
tmp.columns = ['job', 'address']
tmp['city'] = tmp['address'].str.split('·').str[0]
tmp = tmp[['job', 'city']]
df3 = pd.concat([df2, tmp], axis=1)
df3 = df3.drop('job_address', axis=1)
```

拆分“salary”，这个列形如“1k-4k”，以连字符“-”为标志来进行拆分，得到“min_salary”和“max_salary”，接着去掉表示数字单位千的字符“k”，并转为数字类型。

```
df3[['min_salary', 'max_salary']] = df3['salary'].str.split('-', expand=True)
df3['min_salary'] = df3['min_salary'].str.replace('k', '').astype(float) * 1000
df3['max_salary'] = df3['max_salary'].str.replace('k', '').astype(float) * 1000
df4 = df3.drop('salary', axis=1)
```

拆分“requirements”，这一列形如“10k-18k 经验 1-3 年 / 本科”，从字符“经验”到字符“/”是经验要求“experience”，从字符“/”到最后是学历要求“education”。可以通过正则表达式来进行匹配并分割。

```
df4[['experience', 'education']] = df4['requirements'].str.extract(r'(.*)经验(.*) / (.*)', expand=True).iloc[:, [1, 2]].apply(lambda x: x.str.strip())
df5 = df4.drop('requirements', axis=1)
df5.head()
```

拆分“company_info”，通过字符“/”进行分割，并取分割得到的列表的最后两个元素，倒数第二个是“financing”，倒数第一个是“size”。

```
df5[['financing', 'size']] = df5['company_info'].str.split('/', expand=True).iloc[:, -2:]
df6 = df5.drop('company_info', axis=1)
# 去掉空格
df6['financing'] = df6['financing'].str.strip()
df6['size'] = df6['size'].str.strip()
```

先统计一下“highlights”里面有几种可能，以及它们的数量，输出前八种“highlights”。

```
tmp = df6['highlights'].str.strip("“”").str.split(',')
all_highlights = tmp.explode()
highlight_counts = all_highlights.value_counts()
highlight_counts.head(8)
```

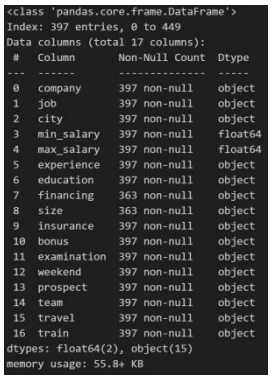
```
highlights
五险一金                    62
绩效奖金                    35
定期体检                    27
方案                      15
周末双休                    13
发展前景佳、团队实力强、老板nice    13
员工旅游                    10
专业培训                     8
Name: count, dtype: int64
```

展开主要的 highlights，这一步同时也处理了 highlights 的缺失值。列举主要的 highlights 的关键词 columns，循环遍历 highlight 和 columns 的笛卡尔积，并做字符串匹配，如果匹配得上，对对应的 columns 的相应行填上“Y”，表示该岗位有该亮点，否则填上“N”，从而完成独热编码。

```
# 提取主要的 highlights
main_highlights = ['五险一金', '奖金', '体检', '双休', '前景', '团队', '旅游', '培训']
columns = ['insurance', 'bonus', 'examination', 'weekend', 'prospect', 'team', 'travel', 'train']
# 去掉引号
df6['highlights'] = df6['highlights'].str.strip("“”")
# 遍历 main_highlights 和对应的列名
for highlight, col in zip(main_highlights, columns):
    # 检查 highlights 列中是否包含关键词
    df6[col] = df6['highlights'].apply(lambda x: 'Y' if highlight in x else 'N')
df7 = df6.drop('highlights', axis=1)
```

查看数据分割后的 dataframe 各个数据列的信息：

```
df7.info()
```



```
<class 'pandas.core.frame.DataFrame'>
Index: 397 entries, 0 to 449
Data columns (total 17 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   company      397 non-null    object
 1   job          397 non-null    object
 2   city         397 non-null    object
 3   min_salary   397 non-null    float64
 4   max_salary   397 non-null    float64
 5   experience   397 non-null    object
 6   education    397 non-null    object
 7   financing    363 non-null    object
 8   size         363 non-null    object
 9   insurance    397 non-null    object
 10  bonus        397 non-null    object
 11  examination  397 non-null    object
 12  weekend      397 non-null    object
 13  prospect     397 non-null    object
 14  team         397 non-null    object
 15  travel       397 non-null    object
 16  train        397 non-null    object
dtypes: float64(2), object(15)
memory usage: 55.8+ KB
```

发现 financing 和 size 有缺失值，考虑到缺失行是字符变量，所以先使用哑变量“未知”进行填充。哑变量填充的意义在于，给予“未知”的语义信息，并且，相对“NAN”可以进行一些运算。

```
df8 = df7.fillna("未知")
df8.info()
df8.head()
```

为了更方便地进行数据统计分析和挖掘任务，进一步进行数据拆分操作。

对 experience 进行拆分，首先查看有几种'experience'。

```
df['experience'].unique()
```

```
array(['不限', '1-3年', '3-5年', '5-10年', '在校/应届', '1年以下', '10年以上'],
      dtype=object)
```

对于不是形如“1-3 年”这样的描述，按照这样的策略对 experience 进行处理：

1、对于在校/应届，最小经验和最大经验都填 0；

2、对于 1 年以下，填 0-1；

3、对于 10 年以上，填 10-40（假设一个人 65 岁退休，25 岁开始工作，大概是 40 年工作经验）；

4、对于不限，填 0-40；

```
df['experience'] = df['experience'].str.replace('在校/应届', '0-0')
df['experience'] = df['experience'].str.replace('1 年以下', '0-1')
df['experience'] = df['experience'].str.replace('10 年以上', '0-1')
df['experience'] = df['experience'].str.replace('不限', '0-40')
# 移除 年 字符
df['experience'] = df['experience'].str.replace('年', '')
df[['min_experience', 'max_experience']] = df['experience'].str.split('-', expand=True).astype(int)
df1 = df.drop('experience', axis=1)
```

接着拆分 size，先看看有几种 size。

```
print(df1['size'].unique())
```

```
['150-500人' '15-50人' '50-150人' '未知' '少于15人' '500-2000人' '2000人以上']
```

对于不是形如“150-500 人”这样的描述，按照这样的策略对 size 进行处理：

1、对于少于 15 人，替换为 0-15；

2、对于 2000 人以上，替换为 2000-208000（查阅资料，数据表中最大公司应该是华为，在 2025 年 5 月 11 日大概 208000 人）；

3、未知，替换为 0-208000；

```
df1['size'] = df1['size'].replace({
    '少于 15 人': '0-15',
```

```
    '2000 人以上': '2000-208000',
    '未知': '0-208000'
})
# 移除 人 字符
df1['size'] = df1['size'].str.replace('人', '')
df1[['min_size', 'max_size']] = df1['size'].str.split('-', expand=True).astype(int)
df2 = df1.drop('size', axis=1)
```

进行异常值处理，首先进行异常值检测，使用 describe 函数打印数据的基本信息如下：

	min_salary	max_salary
count	397.000000	397.000000
mean	12335.012594	20037.783375
std	6081.608997	10247.497015
min	1000.000000	2000.000000
25%	8000.000000	13000.000000
50%	11000.000000	18000.000000
75%	15000.000000	25000.000000
max	40000.000000	60000.000000

发现最低的最低工资低的不正常，查阅资料发现，根据人力资源和社会保障部 2025 年 4 月 3 日发布的全国各省、自治区、直辖市最低工资标准情况，目前中国最低的月最低工资标准是青海省的 1880 元。

所以，对 min_salary 或 max_salary 小于 1880 的行进行删除操作。

```
df9 = df8[(df8['min_salary'] >= 1880) & (df8['max_salary'] >= 1880)]
```

数据合并：

先对 2025 年 5 月 9 日的各个交易所的股票数据进行以列为维度的合并。这是通过 concat 函数实现的。然后，使用 drop_duplicates 去除重复值。

```
df1 = pd.read_csv('1_A 股市场.csv')
df2 = pd.read_csv('2_沪市 A 股.csv')
df3 = pd.read_csv('3_深市 A 股.csv')
df4 = pd.read_csv('4_B 股市场.csv')
df5 = pd.read_csv('5_沪市 B 股.csv')
df6 = pd.read_csv('6_深市 B 股.csv')
df7 = pd.read_csv('7_创业板.csv')
df8 = pd.read_csv('8_科创板.csv')
df9 = pd.read_csv('9_新股.csv')
df10 = pd.read_csv('10_沪港通（沪股通）.csv')
df11 = pd.read_csv('11_深港通（深股通）.csv')
df12 = pd.read_csv('12_沪市风险警示板.csv')
df13 = pd.read_csv('13_北证 A 股.csv')
dfs = [df1, df2, df3, df4, df5, df6, df7, df8, df9, df10, df11, df12, df13]
merged_df = pd.concat(dfs, ignore_index=True)

df_unique = merged_df.drop_duplicates()
```

下面进行数据的连接操作，将爬虫 1 得到的数据（以 company 为键）和上述合并数据（以 abbr 为键）进行内连接的操作，可以得到在两个数据中都出现的公司的数据。

```
# 重命名列名以便连接操作
df_unique_rename = df_unique.rename(columns={'abbr': 'company'})

# 合并两个表格，以 'company' 列为键
df_merge = pd.merge(df, df_unique_rename, on='company', how='inner')
```

上述出现的公司为上市公司，为这些公司在爬虫 1 获得的数据中打上标签，方便之后进行数据挖掘使用。

```
# 在 df 中新建一列 'market'，初始值为 'N'
df['market'] = 'N'

# 检查 df 中的公司是否存在于 df_merge 中
df.loc[df['company'].isin(df_merge['company']), 'market'] = 'Y'
```

数据统计分析：

对数据的每一个数字列，进行统计基本统计量的计算：

```
numeric_df = df2[['min_salary','max_salary','min_experience','max_experience','min_size','max_size']]
print('最小值')
print(numeric_df.min())
print('最大值')
print(numeric_df.max())
print('平均值')
print(numeric_df.mean())
print('求和')
print(numeric_df.sum())
print('最小值')
print(numeric_df.min())
print('标准差')
print(numeric_df.std())
```

```
最小值
min_salary        2000.0
max_salary        3000.0
min_experience       0.0
max_experience       0.0
min_size             0.0
max_size            15.0
dtype: float64
最大值
min_salary        40000.0
max_salary        60000.0
min_experience        5.0
max_experience       40.0
min_size           2000.0
max_size         208000.0
dtype: float64
```

```
平均值
min_salary        12421.319797
max_salary        20175.126904
min_experience        1.822335
max_experience       10.406091
min_size            437.550761
max_size          55208.299492
dtype: float64
求和
min_salary         4894000.0
max_salary         7949000.0
min_experience         718.0
max_experience        4100.0
min_size            172395.0
max_size          21752070.0
dtype: float64
```

```
标准差
min_salary         6023.292556
max_salary        10164.163490
min_experience        1.529659
max_experience       13.579755
min_size            740.786257
max_size          91617.244280
dtype: float64
```

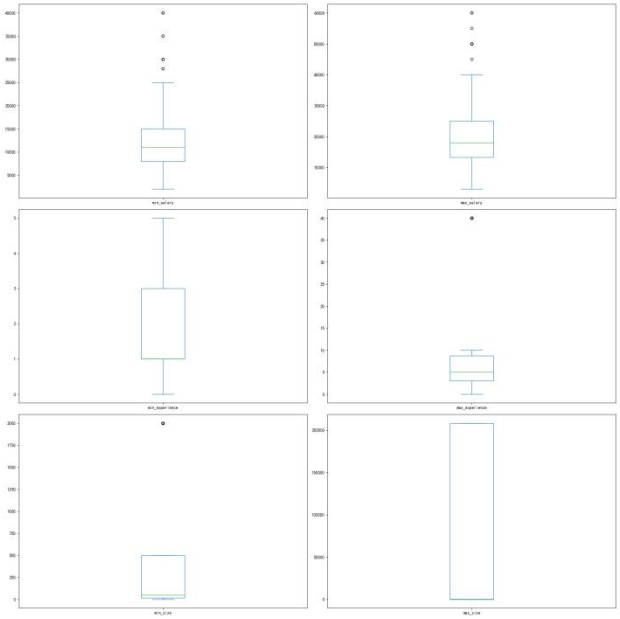
基本分析可以得到，python 岗位的薪资的平均值较高，在万元以上，但两级分化较大，最低薪资可能甚至 2000 元，岗位普遍要求较长的工作经验，并且大多数情况是大公司在招收 python 岗位的员工。

接着，按照地区进行分组，计算基本统计量。

```
grouped_city = df2.groupby("city")[['min_salary','max_salary','min_experience','max_experience','min_size','max_size']].agg(["mean", "max", "min", "sum"])
```

	min_salary				max_salary				min_experience	...	max_experience		min_size
	mean	max	min	sum	mean	max	min	sum	mean	max ...	min	sum	mean
city													
上海	14030.303030	35000.0	3000.0	926000.0	22090.909091	55000.0	4000.0	1458000.0	1.954545	5 ...	0	589	307.954545
东莞	9500.000000	15000.0	4000.0	57000.0	14333.333333	25000.0	5000.0	86000.0	2.500000	5 ...	0	28	835.833333
中山	10000.000000	10000.0	10000.0	10000.0	16000.000000	16000.0	16000.0	16000.0	3.000000	3 ...	5	5	0.000000
佛山	8500.000000	9000.0	8000.0	17000.0	14500.000000	15000.0	14000.0	29000.0	2.000000	3 ...	3	8	1025.000000
兰州	6000.000000	6000.0	6000.0	6000.0	8000.000000	8000.0	8000.0	8000.0	1.000000	1 ...	3	3	0.000000
北京	16569.230769	40000.0	2000.0	1077000.0	26476.923077	60000.0	3000.0	1721000.0	1.676923	5 ...	0	851	451.692308
南京	11315.789474	20000.0	4000.0	215000.0	19000.000000	30000.0	5000.0	361000.0	2.157895	5 ...	0	121	586.578947
厦门	9000.000000	15000.0	3000.0	18000.0	12000.000000	20000.0	4000.0	24000.0	1.500000	3 ...	0	5	75.000000
合肥	9333.333333	10000.0	8000.0	28000.0	15333.333333	16000.0	15000.0	46000.0	2.333333	3 ...	3	13	100.000000

绘制所有数值型列的箱线图如下：

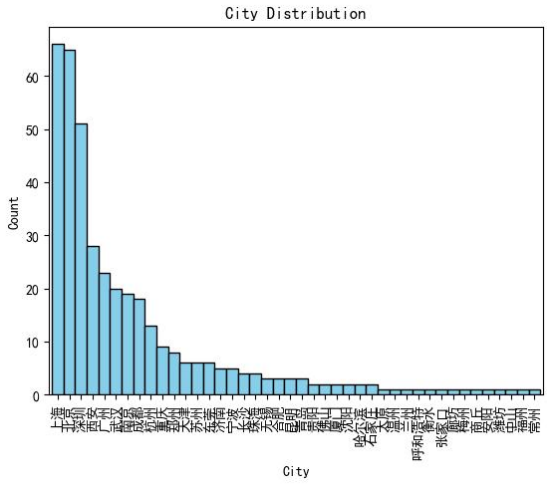


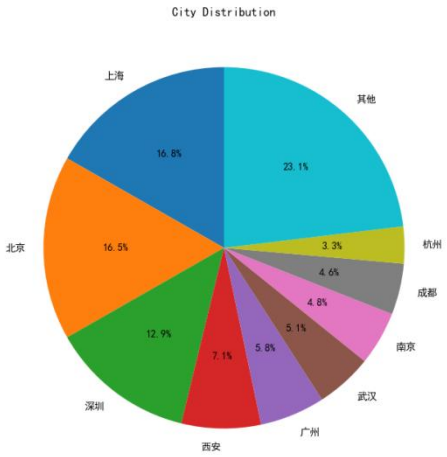
箱线图不仅能大致展示数据的基本分布情况，而且能进行异常值的检验，这里虽然检验出了较多的异常值，但我认为这些异常值是事物的客观规律，所以没有处理。

对城市出现的频数进行统计，并绘制频数直方图和饼图。

```
city_counts = df2["city"].value_counts()
city_counts.plot(kind='bar', color='skyblue', edgecolor='black', width=1)
plt.title('City Distribution')
plt.xlabel('City')
plt.ylabel('Count')
plt.show()
# 只保留前 9 个城市，其余合并为 其他
top_9 = city_counts.head(9)
other = city_counts[9:].sum()
city_counts = pd.concat([top_9, pd.Series({"其他": other})])
plt.figure(figsize=(8, 8))
plt.pie(city_counts, labels=city_counts.index, autopct="%1.1f%%", startangle=90)
```

```
plt.title("City Distribution")
plt.show()
```



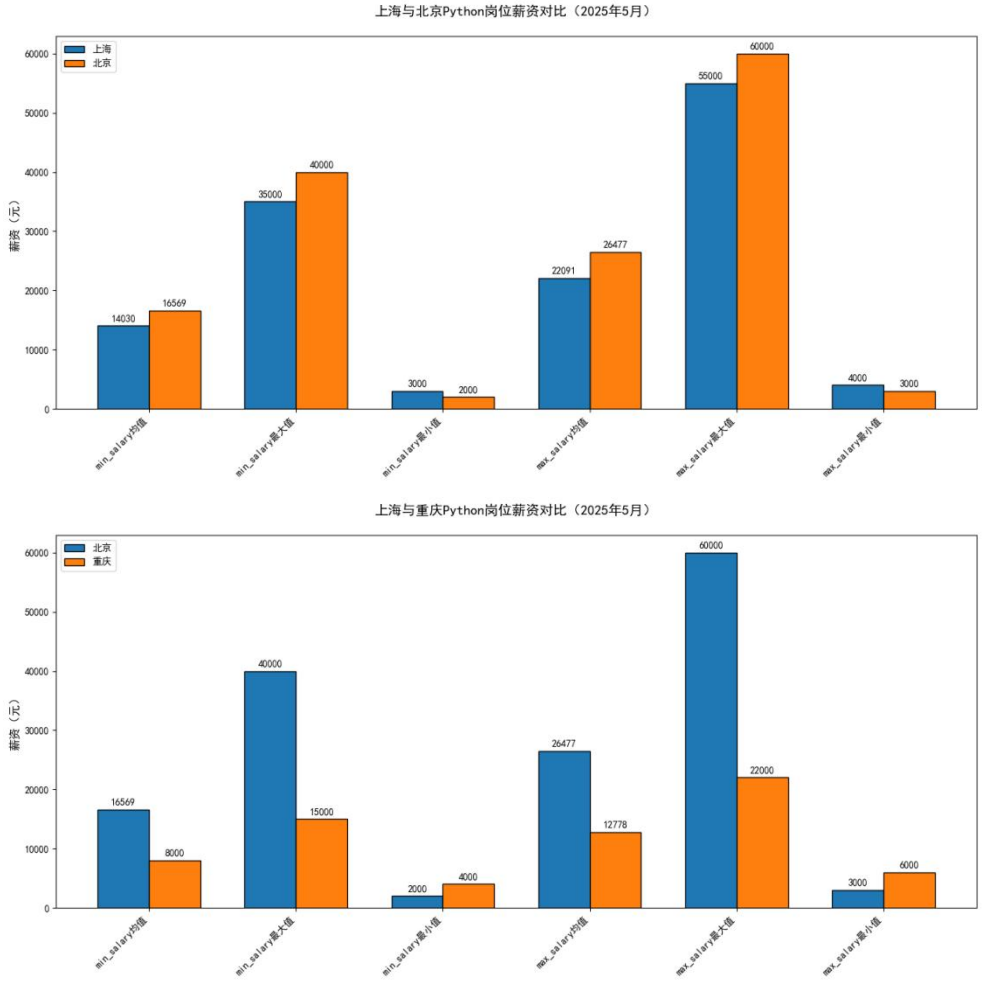


我做出结论，经济和软件产业发达的城市（如北京、上海、深圳等）占比较多，这些地区对 python 岗位员工的需求旺盛。

下面的代码用于对两个城市的薪资通过绘制柱状图的形式进行对比。

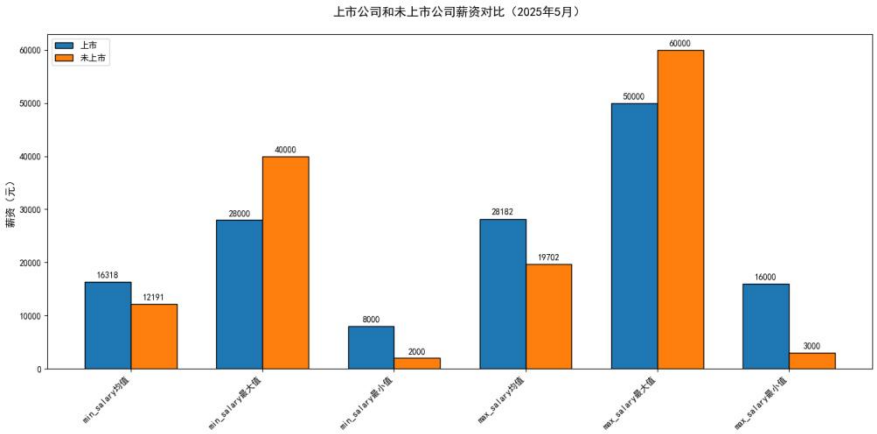
```
import numpy as np
df_filtered = df2[df2['city'].isin(['北京', '上海'])]
# 统计薪资指标
salary_stats = df_filtered.groupby("city")[['min_salary','max_salary']].agg(["mean", "max", "min"])
print(salary_stats)
cities = ['上海', '北京']
metrics = ['min_salary 均值', 'min_salary 最大值', 'min_salary 最小值',
           'max_salary 均值', 'max_salary 最大值', 'max_salary 最小值']
shanghai = [14030.30, 35000.0, 3000.0, 22090.91, 55000.0, 4000.0]
beijing = [16569.23, 40000.0, 2000.0, 26476.92, 60000.0, 3000.0]
# 图表参数设置
x = np.arange(len(metrics))  # 指标位置
width = 0.35  # 柱体宽度
colors = ['#1f77b4', '#ff7f0e']  # 城市配色
# 创建画布
plt.figure(figsize=(14, 7))
ax = plt.subplot()
rects1 = ax.bar(x - width/2, shanghai, width, label='上海', color=colors[0], edgecolor='black')
rects2 = ax.bar(x + width/2, beijing, width, label='北京', color=colors[1], edgecolor='black')
ax.set_ylabel('薪资（元）', fontsize=12)
ax.set_title('上海与北京 Python 岗位薪资对比（2025 年 5 月）', fontsize=14, pad=20)
ax.set_xticks(x)
ax.set_xticklabels(metrics, rotation=45, ha='right')
ax.legend(loc='upper left')
def autolabel(rects):
    for rect in rects:
        height = rect.get_height()
```

```
            ax.annotate(f'{height:.0f}',
                        xy=(rect.get_x() + rect.get_width()/2, height),
                        xytext=(0, 3),
                        textcoords="offset points",
                        ha='center', va='bottom')
autolabel(rects1)
autolabel(rects2)
plt.tight_layout()
plt.show()
```



以北京、上海和重庆为例，北京的 python 岗位的工资在统计意义上略高于上海，显著高于重庆。这说明在经济发达、IT 行业繁荣的城市，python 岗位的工资可能会更高。

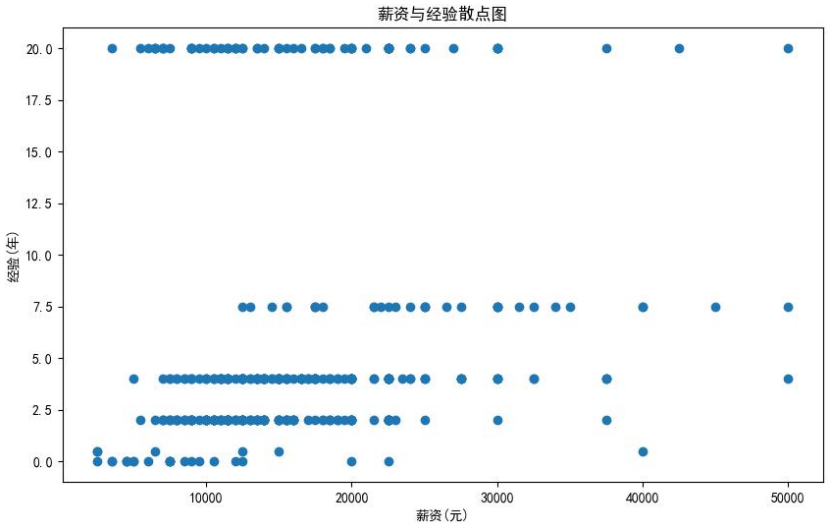
同理，可以以是否上市为分类标准，绘制柱状图，对比上市公司和未上市公司的薪资情况。



上市公司的 min_salary 的均值和最小值以及 max_salary 的均值和最小值显著地高于未上市公司；上市公司的 min_salary 的最大值以及 max_salary 的最大值低于未上市公司，这可能是因为极端数据的影响。总体来说，上市公司能给 python 求职者更多的薪资。

散点图擅长于发掘两个指标之间的相关性，绘制薪资与经验的散点图如下：

```
plt.figure(figsize=(10, 6))
plt.scatter((df2['min_salary'] + df2['max_salary']) / 2, (df2['min_experience'] + df2['max_experience']) / 2)
plt.title('薪资与经验散点图')
plt.xlabel('薪资(元)')
plt.ylabel('经验(年)')
plt.show()
```

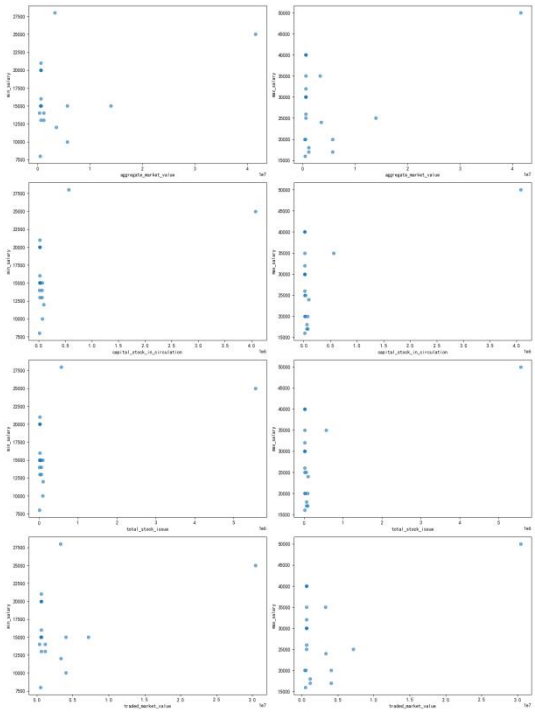


由上图，python 岗位的薪资和经验近似地呈现少许相关性，尤其对于经验在 10 年以内，薪资在 30000 元以内的 python 岗位。

下面绘制上市公司的总市值、流通股本、总股本、流通市值与最低薪资和最高薪资的散点图。

```
import matplotlib.pyplot as plt
x_list = ['aggregate_market_value', 'capital_stock_in_circulation', 'total_stock_issue', 'traded_market_value']
y_list = ['min_salary', 'max_salary']
fig, axes = plt.subplots(nrows=4, ncols=2, figsize=(15, 20))
for i, x in enumerate(x_list):
    for j, y in enumerate(y_list):
```

```
            # 计算当前子图的位置
            ax = axes[i, j]
            ax.scatter(df_merge[x], df_merge[y], alpha=0.6)
            ax.set_xlabel(x)
            ax.set_ylabel(y)
plt.tight_layout()
plt.show()
```

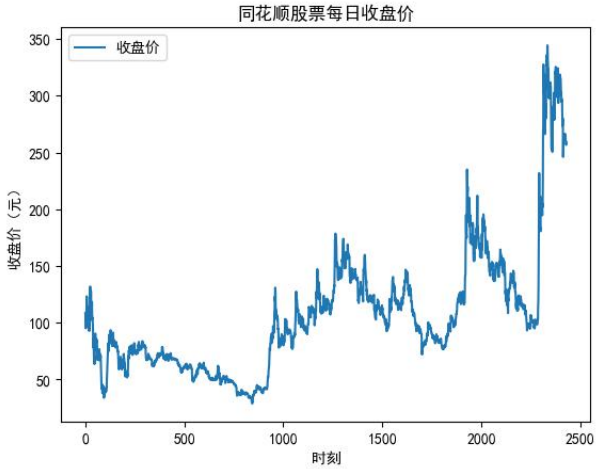


可以认为公司股票的总市值和流通市值与薪资呈现更高的相关性。

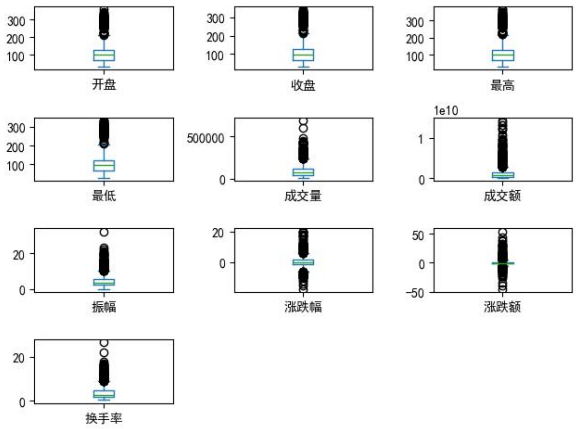
数据挖掘分析：

回归模型 1：股票收盘价的时序预测模型

以同花顺公司的股票时序信息为例，进行回归分析。其股票收盘价走势如下：



绘制数据的箱线图如下：



虽然箱线图识别出来较多异常值，但我认为这是事物的客观规律，所以不进行处理。

以收盘价作为因变量，使用时间作为自变量，建立一个多项式回归模型。将数据集的前 80%作为多项式回归的训练集，将后 20%作为测试集。使用 MSE 和 R2 参数作为评价指标，使用一个循环搜索多项式的最高次。多项式的最高次是个重要的参数，太小模型过于简单，效果不好；过大会产生过拟合或者预测值跳变的现象。

```
for degree in range(1, 8):
    poly_features = PolynomialFeatures(degree=degree, include_bias=False)
    X_train_poly = poly_features.fit_transform(X_train)
    X_test_poly = poly_features.transform(X_test)
    # 拟合线性回归模型
    model = LinearRegression()
    model.fit(X_train_poly, y_train)
    # 预测
    y_train_pred = model.predict(X_train_poly)
    y_test_pred = model.predict(X_test_poly)
    # 计算并输出误差指标
    print(degree)
    train_mse = mean_squared_error(y_train, y_train_pred)
    test_mse = mean_squared_error(y_test, y_test_pred)
    train_r2 = r2_score(y_train, y_train_pred)
    test_r2 = r2_score(y_test, y_test_pred)
    print(f"训练集均方误差 (MSE): {train_mse}")
    print(f"测试集均方误差 (MSE): {test_mse}")
    print(f"训练集 R^2 得分: {train_r2}")
    print(f"测试集 R^2 得分: {test_r2}")
```

```
1
训练集均方误差 (MSE): 750.1115255209922
测试集均方误差 (MSE): 7024.26954786016
训练集R^2得分: 0.3476243247763673
测试集R^2得分: -0.3740458057312608
2
训练集均方误差 (MSE): 727.2413152721589
测试集均方误差 (MSE): 5132.236035448482
训练集R^2得分: 0.3675146588746555
测试集R^2得分: -0.003937470007694399
3
训练集均方误差 (MSE): 535.8710207239543
测试集均方误差 (MSE): 32030.717210927414
训练集R^2得分: 0.533950343270944
测试集R^2得分: -5.265658277827113
```

```
3
训练集均方误差 (MSE): 535.8710207239543
测试集均方误差 (MSE): 32030.717210927414
训练集R^2得分: 0.533950343270944
测试集R^2得分: -5.265658277827113
4
训练集均方误差 (MSE): 533.9264604269423
测试集均方误差 (MSE): 22780.787220054935
训练集R^2得分: 0.5356415369049776
测试集R^2得分: -3.456242021707216
5
训练集均方误差 (MSE): 330.14655379152106
测试集均方误差 (MSE): 327096.3601689417
训练集R^2得分: 0.7128699218387474
测试集R^2得分: -62.98464334231209
```

```
6
训练集均方误差 (MSE): 222.37692542044613
测试集均方误差 (MSE): 2588462.808915002
训练集R^2得分: 0.806597696556445
测试集R^2得分: -505.3396900770275
7
训练集均方误差 (MSE): 220.07749595956986
测试集均方误差 (MSE): 4447248.952246223
训练集R^2得分: 0.8085975216439563
测试集R^2得分: -868.9443733246551
```

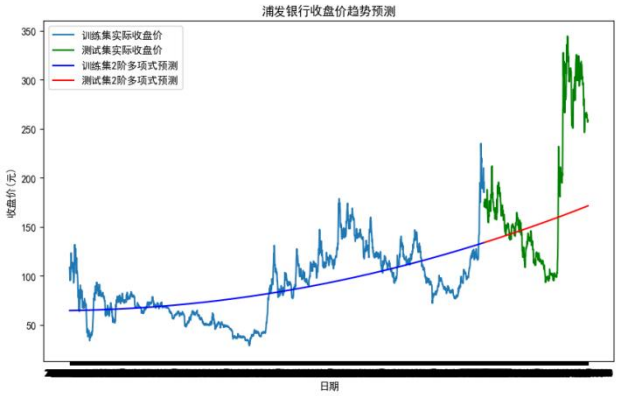
可以认为最好的最高次数是 2。

以 2 为参数，进行模型的训练和可视化：

```
degree = 2
poly_features = PolynomialFeatures(degree=degree, include_bias=False)
X_train_poly = poly_features.fit_transform(X_train)
X_test_poly = poly_features.transform(X_test)
# 拟合线性回归模型
model = LinearRegression()
model.fit(X_train_poly, y_train)
# 预测
y_train_pred = model.predict(X_train_poly)
y_test_pred = model.predict(X_test_poly)
# 计算并输出误差指标
print(degree)
train_mse = mean_squared_error(y_train, y_train_pred)
test_mse = mean_squared_error(y_test, y_test_pred)
train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2_score(y_test, y_test_pred)
print(f"训练集均方误差 (MSE): {train_mse}")
print(f"测试集均方误差 (MSE): {test_mse}")
print(f"训练集 R^2 得分: {train_r2}")
print(f"测试集 R^2 得分: {test_r2}")
plt.figure(figsize=(10, 6))
plt.plot(df.index[X_train.flatten()], y_train, label='训练集实际收盘价')
plt.plot(df.index[X_test.flatten()], y_test, label='测试集实际收盘价', color='green')
plt.plot(df.index[X_train.flatten()], y_train_pred, label=f'训练集{degree}阶多项式预测', color='blue')
plt.plot(df.index[X_test.flatten()], y_test_pred, label=f'测试集{degree}阶多项式预测', color='red')
plt.title('浦发银行收盘价趋势预测')
plt.xlabel('日期')
plt.ylabel('收盘价(元)')
plt.legend()
plt.show()
# 输出模型系数
print("Model Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
```

```
2
训练集均方误差 (MSE): 727.2413152721589
测试集均方误差 (MSE): 5132.236035448482
训练集R^2得分: 0.3675146588746555
测试集R^2得分: -0.003937470007694399
```

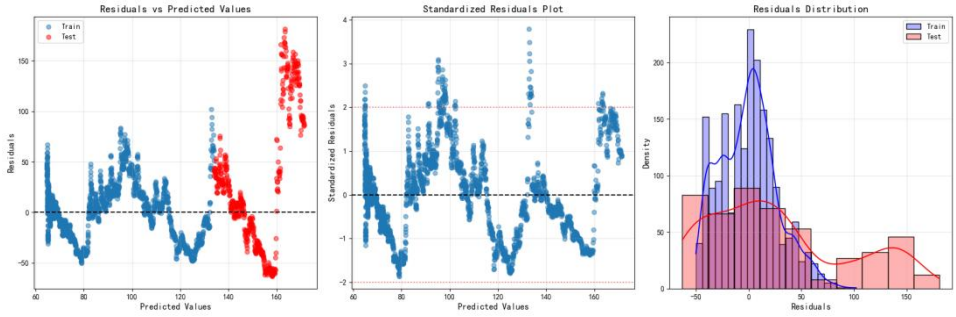
```
Model Coefficients: [2.63802136e-03 1.69777095e-05]
Intercept: 64.69627733120964
```



下面绘制预测值-残差散点图，标准化残差图，残差分布直方图。

```
import seaborn as sns
# 计算训练集和测试集残差
residuals_train = y_train - y_train_pred
residuals_test = y_test - y_test_pred
# 创建子图布局
plt.figure(figsize=(18, 6))
# 子图 1：预测值-残差散点图
plt.subplot(1, 3, 1)
plt.scatter(y_train_pred, residuals_train, alpha=0.5, label='Train')
plt.scatter(y_test_pred, residuals_test, alpha=0.5, color='red', label='Test')
plt.axhline(y=0, color='black', linestyle='--')
plt.xlabel('Predicted Values', fontsize=12)
plt.ylabel('Residuals', fontsize=12)
plt.title('Residuals vs Predicted Values', fontsize=14)
plt.legend()
plt.grid(alpha=0.3)
# 子图 2：标准化残差图
plt.subplot(1, 3, 2)
std_residuals = np.concatenate([(residuals_train - np.mean(residuals_train))/np.std(residuals_train),
                                (residuals_test - np.mean(residuals_test))/np.std(residuals_test)])
plt.scatter(np.concatenate([y_train_pred, y_test_pred]), std_residuals, alpha=0.5)
plt.axhline(y=0, color='black', linestyle='--')
plt.axhline(y=2, color='red', linestyle=':', alpha=0.5)
plt.axhline(y=-2, color='red', linestyle=':', alpha=0.5)
plt.xlabel('Predicted Values', fontsize=12)
plt.ylabel('Standardized Residuals', fontsize=12)
plt.title('Standardized Residuals Plot', fontsize=14)
plt.grid(alpha=0.3)
# 子图 3：残差分布直方图
plt.subplot(1, 3, 3)
sns.histplot(residuals_train, kde=True, color='blue', label='Train', alpha=0.3)
sns.histplot(residuals_test, kde=True, color='red', label='Test', alpha=0.3)
plt.xlabel('Residuals', fontsize=12)
```

```
plt.ylabel('Density', fontsize=12)
plt.title('Residuals Distribution', fontsize=14)
plt.legend()
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```



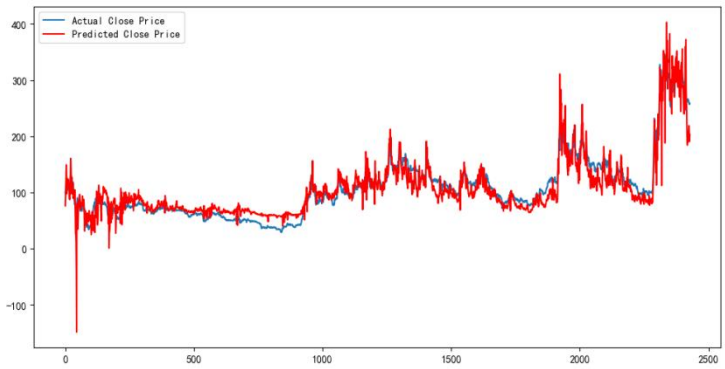
自回归的意思是，对于一个时间序列，使用一个时刻的前几个时刻的数据，预测这一个时刻的数据。

下面的方案使用自回归的思想，使用多元多项式回归和数据挖掘算法构建回归模型。

首先是时间窗口为 3，使用多元三项式回归，按照 80%的比例随机选择训练集的情况。

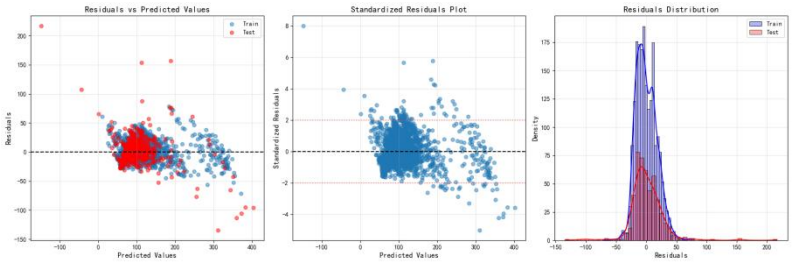
```
# 选择需要的特征
features = ['开盘','最高','最低','成交量','成交额','振幅', '换手率']
target = '收盘'
# 确保没有缺失值
df_selected = df[features + [target]].dropna()
# 构造滞后特征
lag_features = {}
lags = 3  # 滞后期数
for feature in features + [target]:
    for lag in range(1, lags + 1):
        col_name = f'{feature}_lag_{lag}'
        df_selected[col_name] = df_selected[feature].shift(lag)

# 删除含有 NaN 的行（由滞后导致）
df_selected.dropna(inplace=True)
# 划分特征和目标变量
X = df_selected[[col for col in df_selected.columns if col != target]]
y = df_selected[target]
# 使用 train_test_split 进行随机划分，保留时间索引用于后续绘图
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=True, random_state=42)
print(f"训练集大小: {len(X_train)}")
print(f"测试集大小: {len(X_test)}")
# 创建多项式特征
degree = 2  # 设定多项式的阶数
poly = PolynomialFeatures(degree=degree, include_bias=False)
X_train_poly = poly.fit_transform(X_train)
```

```
X_test_poly = poly.transform(X_test)
# 创建并训练线性回归模型
model = LinearRegression()
model.fit(X_train_poly, y_train)
# 预测
y_pred_train = model.predict(X_train_poly)
y_pred_test = model.predict(X_test_poly)
# 评估模型
train_rmse = np.sqrt(mean_squared_error(y_train, y_pred_train))
test_rmse = np.sqrt(mean_squared_error(y_test, y_pred_test))
print(f"Train RMSE: {train_rmse}")
print(f"Test RMSE: {test_rmse}")
# 为了可视化，我们需要将预测结果与原始索引对齐
# 将预测结果放到 DataFrame 中，并用原始索引排序
predictions_df = pd.DataFrame({
    'Actual': np.concatenate([y_train.values, y_test.values]),
    'Predicted': np.concatenate([y_pred_train, y_pred_test])
}, index=np.concatenate([y_train.index.values, y_test.index.values]))
predictions_df.sort_index(inplace=True)
# 可视化收盘价预测结果与实际值
plt.figure(figsize=(12, 6))
plt.plot(np.arange(predictions_df.shape[0]), predictions_df['Actual'], label='Actual Close Price')
plt.plot(np.arange(predictions_df.shape[0]), predictions_df['Predicted'], label='Predicted Close Price', color='red')
plt.legend()
plt.show()
```



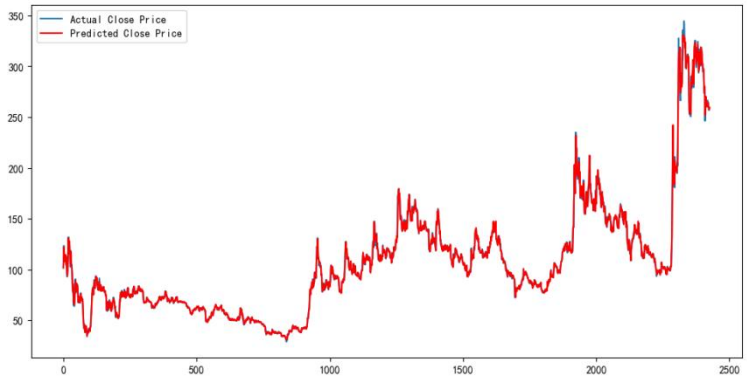

```
Train RMSE: 17.13177649028624
Test RMSE: 27.04550456961892
```

绘制预测值-残差散点图，标准化残差图，残差分布直方图如下：

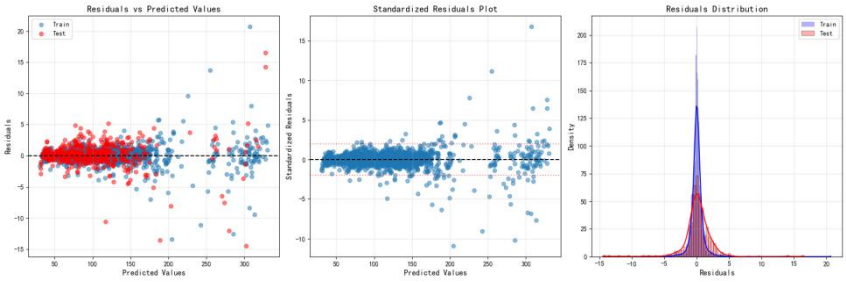


下面是使用随机森林回归模型，时间窗口大小为 5，使用 5 则交叉验证和以均方误差为标准网格搜索来求取最佳参数，训练结果如下：

```
rf = RandomForestRegressor(random_state=137)
param_grid = {
    'n_estimators': [100, 150, 200],  # 决策树的数量
    'max_depth': [None, 5, 10],  # 树的最大深度
}
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)
print("最佳参数组合：", grid_search.best_params_)
print("最佳模型的交叉验证 R²分数：", grid_search.best_score_)
# 使用最佳模型对测试集进行预测
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
# 计算并输出测试集的均方误差
mse = mean_squared_error(y_test, y_pred)
print("测试集的均方误差：", mse)
# 下面进行可视化
y_pred_train = best_model.predict(X_train)
y_pred_test = y_pred
# 为了可视化，我们需要将预测结果与原始索引对齐
predictions_df = pd.DataFrame({
    'Actual': np.concatenate([y_train.values, y_test.values]),
    'Predicted': np.concatenate([y_pred_train, y_pred_test])
}, index=np.concatenate([y_train.index.values, y_test.index.values]))
predictions_df.sort_index(inplace=True)
plt.figure(figsize=(12, 6))
plt.plot(np.arange(predictions_df.shape[0]), predictions_df['Actual'], label='Actual Close Price')
plt.plot(np.arange(predictions_df.shape[0]), predictions_df['Predicted'], label='Predicted Close Price', color='red')
plt.legend()
plt.show()
```

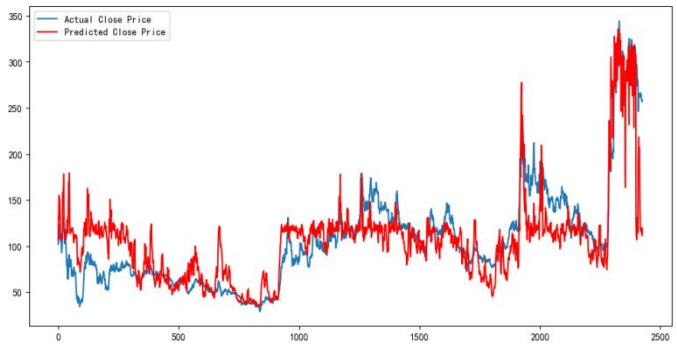


```
最佳参数组合： {'max_depth': None, 'n_estimators': 150}
最佳模型的交叉验证R²分数： -10.71169186449254
测试集的均方误差： 4.757800603539139
```

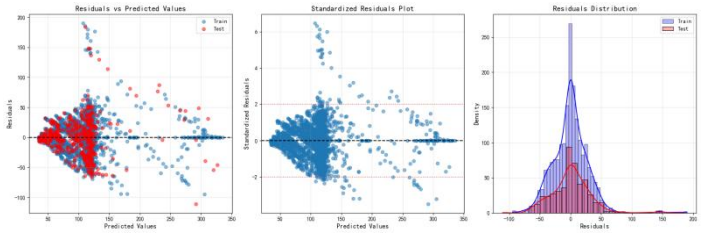


同理，除了随机森林，我还使用支持向量机、多层感知机、KNN 使用相同的训练手法进行了训练。结果展示如下。

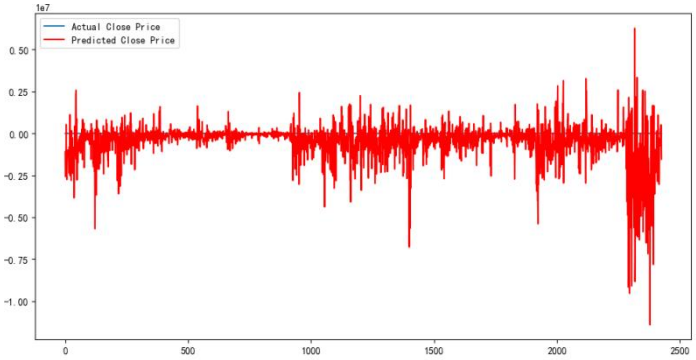
支持向量机：



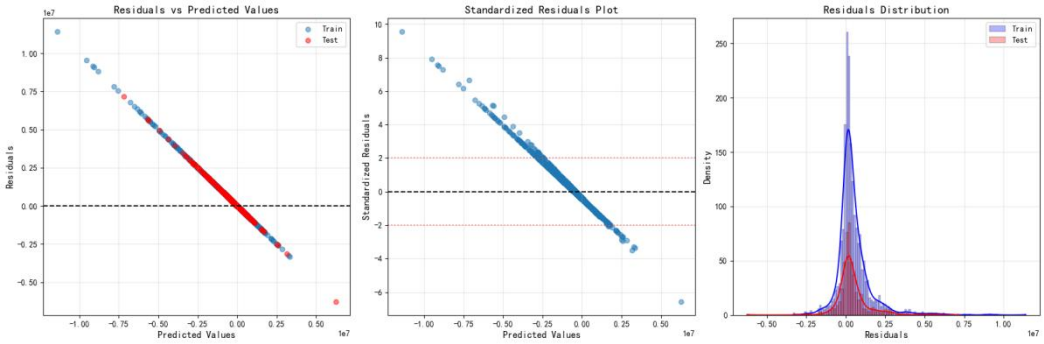

```
最佳参数组合： {'C': 100}
最佳模型的交叉验证R²分数： -1007.2891232350603
测试集的均方误差： 1023.2263818677548
```



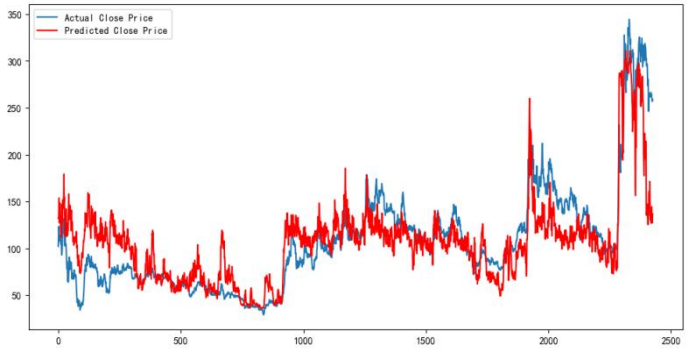
多层感知机：



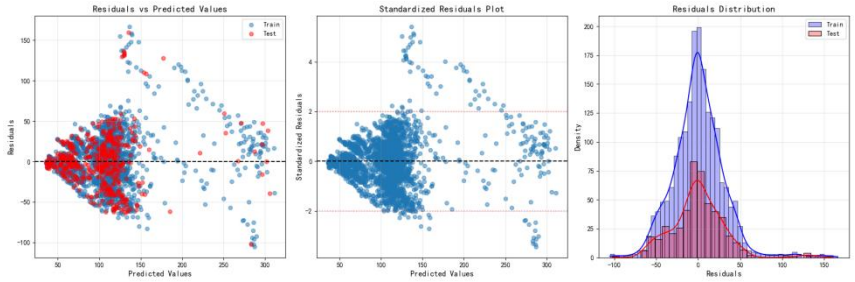

```
最佳参数组合： {'hidden_layer_sizes': (50, 100, 50), 'learning_rate_init': 0.01}
最佳模型的交叉验证R²分数： -1011003542607.9563
测试集的均方误差： 1196535683218.0005
```



KNN：




```
最佳参数组合： {'n_neighbors': 22}
最佳模型的交叉验证R²分数： -1034.5248323025476
测试集的均方误差： 1020.143921147927
```



在上面的回归模型中，随机森林模型取得了最好的预测效果。

回归模型 2：薪资的预测模型

模型通过一个工作的工作城市、学历要求、公司的融资情况、公司是否提供五险一金、体检等福利、经验要求、公司规模等来预测工作的薪资。

考虑到城市、学历等这些信息是离散信息，难以直接作为回归模型的输入，所以，我首先对非数字列进行标签编码，并随机取 80%的数据为训练集，从而由爬虫 1 数据构造出数据集。

```
y = df['salary']
X = df.drop('salary', axis=1)
#  对非数字列进行标签编码
for column in X.columns:
    if X[column].dtype == 'object':
        label_encoder = LabelEncoder()
        X[column] = label_encoder.fit_transform(X[column])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

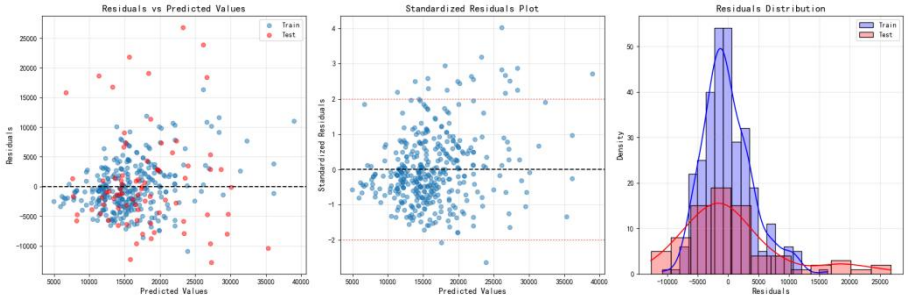
构建随机森林模型，使用五折交叉验证，以 R2 分数为标准进行最佳参数的网格搜索，训练得到

模型的评估指标如下：

```
rf = RandomForestRegressor(random_state=137)
param_grid = {
    'n_estimators': [100, 150, 200],  # 决策树的数量
    'max_depth': [None, 3, 4, 5, 6, 10],  # 树的最大深度
}
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, scoring='r2')
grid_search.fit(X_train, y_train)
print("最佳参数组合：", grid_search.best_params_)
print("最佳模型的交叉验证 R²分数：", grid_search.best_score_)
# 使用最佳模型对测试集进行预测
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
# 计算并输出测试集的均方误差
mse = mean_squared_error(y_test, y_pred)
print("测试集的均方误差：", mse)
r2 = r2_score(y_test, y_pred)
print("测试集的 r2 分数：", r2)
```

```
最佳参数组合： {'max_depth': 6, 'n_estimators': 100}
最佳模型的交叉验证R²分数： 0.25951913066047994
测试集的均方误差： 67312891.60257225
测试集的r2分数： 0.31975692510577136
```

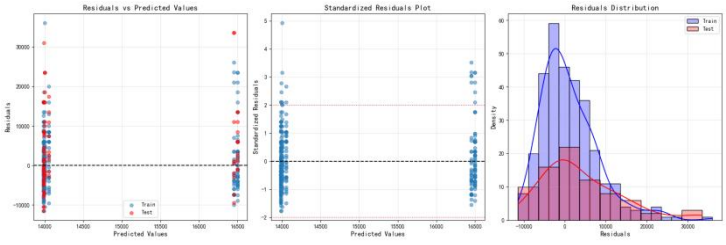
绘制预测值-残差散点图，标准化残差图，残差分布直方图如下：



以相同的训练手法，我还构建了支持向量机模型、多层感知机、KNN 的回归模型，结果展示如下。

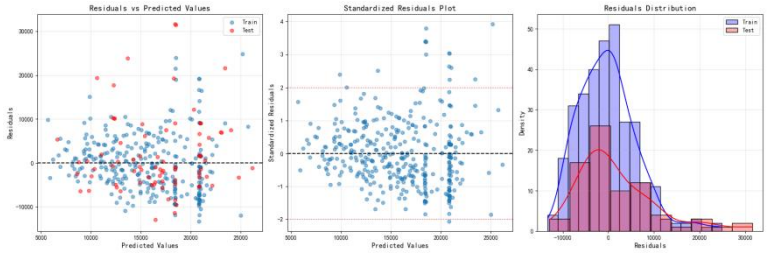
支持向量机：

```
最佳参数组合： {'C': 10000}
最佳模型的交叉验证R²分数： -0.0005089448965880594
测试集的均方误差： 110388092.77319816
测试集的r2分数： -0.1155476146099812
```



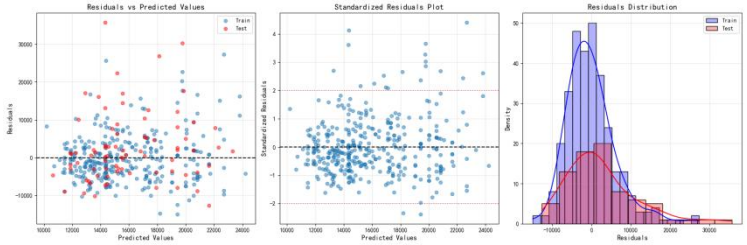
多层感知机：

```
最佳参数组合： {'hidden_layer_sizes': (50, 50), 'learning_rate_init': 0.01}
最佳模型的交叉验证R²分数： 0.11071608784897513
测试集的均方误差： 80282994.44186698
测试集的r2分数： 0.1886851136437443
```



KNN：

```
最佳参数组合： {'n_neighbors': 11}
最佳模型的交叉验证R²分数： 0.08314789114357735
测试集的均方误差： 89902421.80144367
测试集的r2分数： 0.09147418286937736
```



对比发现，首先效果都不算很好，可能是因为数据集的随机性较强，任何模型都不能很好的捕捉其中的规律。其中，随机森林模型效果相对较好，尤其 R2 分数较高，可以用于预测薪资高低的大致走向。

分类模型：以城市为标签进行分类。

由上面的统计分析可知，城市的数据的分布是极度不均的，所以，首先，我使用数据合成的手段，对数据进行数据均衡。

本次实验，我使用 SMOTE-NC 算法进行过采样，这个算法的优势在于不仅能对数值型数据进行处理，而且可以对非数值类数据处理，非常符合我这个数据集的特点。

```
# 加载数据并预处理
df = pd.read_csv('output4.csv')
top_10_cities = df["city"].value_counts().head(10).index
df['city'] = df['city'].where(df['city'].isin(top_10_cities), '其他')
df = df.drop(['job', 'company'], axis=1)
# 定义目标变量 y 和特征 X
y = df['city']
X = df.drop('city', axis=1)
# 定义分类特征列名（排除数值型列）
categorical_features = [
    'education', 'financing', 'insurance', 'bonus',
    'examination', 'weekend', 'prospect', 'team',
    'travel', 'train'
]
```

```
# 获取分类特征在 X 中的索引位置
categorical_indices = [X.columns.get_loc(col) for col in categorical_features]
# 初始化 SMOTE-NC，明确指定分类特征索引
smote_nc = SMOTENC(
    categorical_features=categorical_indices,
    random_state=42
)
# 应用过采样
X_res, y_res = smote_nc.fit_resample(X, y)
```

数据均衡后，使用标签编码对非数值数据进行标签编码，并且按 80%的比例，随机选择训练集。

构建 KNN 的分类模型，使用五折交叉验证，以准确率为标准进行参数的网格搜索。

对 KNN 模型等以距离为基石的模型所对应的数据使用标签编码的合理性在于，我的非数值变量要么是分等级的，比如学历，有“不限”、“本科”、“硕士”、“博士”等，用标签编码可以表示它们之间的差异程度，符合距离的语义信息；要么就是该变量是二元的，比如是否有五险一金、是否有双休等，也符合距离的语义信息。

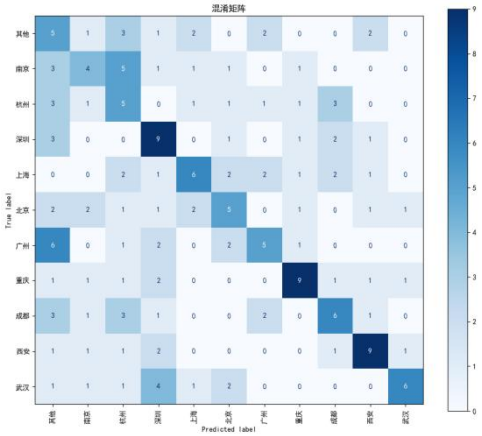
```
rf = KNeighborsClassifier()
param_grid = {
    'n_neighbors': range(3, 50)
}
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)
print("最佳参数组合：", grid_search.best_params_)
```

混淆矩阵如下，各种评估参数使用 classification_report 打印如下。

```
# 使用最佳模型对测试集进行预测
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
print(classification_report(y_test, y_pred))
# 绘制混淆矩阵
fig, ax = plt.subplots(figsize=(12, 10))
ConfusionMatrixDisplay.from_predictions(
    y_test,
    y_pred,
    display_labels=y_test.unique(),
    cmap='Blues',
    ax=ax,
    xticks_rotation='vertical'
)
plt.title(f'混淆矩阵')
plt.show()
```

最佳参数组合： {'n_neighbors': 4}

	precision	recall	f1-score	support
上海	0.18	0.31	0.23	16
其他	0.33	0.25	0.29	16
北京	0.22	0.31	0.26	16
南京	0.38	0.53	0.44	17
广州	0.46	0.35	0.40	17
成都	0.36	0.31	0.33	16
杭州	0.42	0.29	0.34	17
武汉	0.60	0.53	0.56	17
深圳	0.40	0.35	0.38	17
西安	0.56	0.56	0.56	16
重庆	0.67	0.38	0.48	16
accuracy			0.38	181
macro avg	0.42	0.38	0.39	181
weighted avg	0.42	0.38	0.39	181

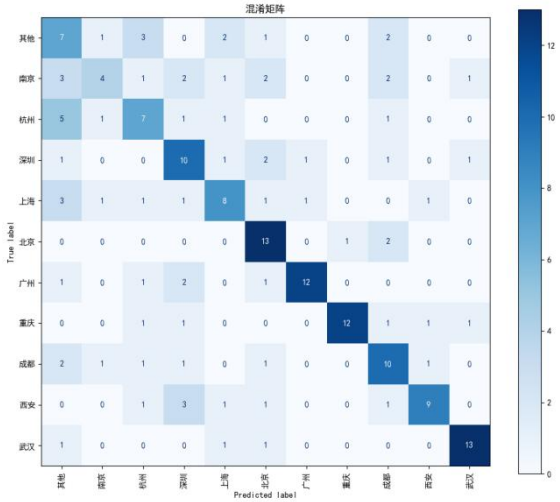


以相同的训练手法，我构建了随机森林、多层感知机、支持向量机的分类模型，结果如下。

随机森林：

最佳参数组合： {'max_depth': 20, 'n_estimators': 200}

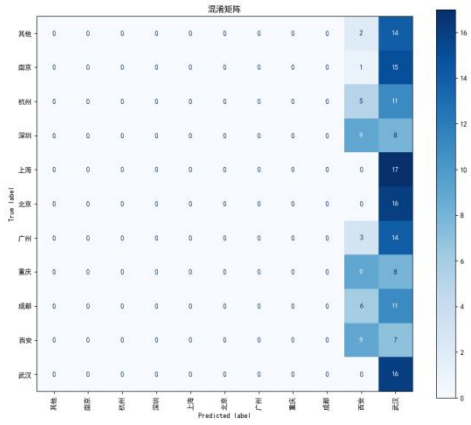
	precision	recall	f1-score	support
上海	0.30	0.44	0.36	16
其他	0.50	0.25	0.33	16
北京	0.44	0.44	0.44	16
南京	0.48	0.59	0.53	17
广州	0.53	0.47	0.50	17
成都	0.57	0.81	0.67	16
杭州	0.86	0.71	0.77	17
武汉	0.92	0.71	0.80	17
深圳	0.50	0.59	0.54	17
西安	0.75	0.56	0.64	16
重庆	0.81	0.81	0.81	16
accuracy			0.58	181
macro avg	0.61	0.58	0.58	181
weighted avg	0.61	0.58	0.58	181



多层感知机：

最佳参数组合： {'hidden_layer_sizes': (16, 16, 16), 'learning_rate_init': 0.01}

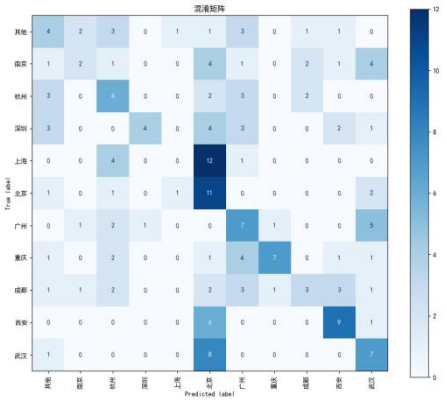
	precision	recall	f1-score	support
上海	0.00	0.00	0.00	16
其他	0.00	0.00	0.00	16
北京	0.00	0.00	0.00	16
南京	0.00	0.00	0.00	17
广州	0.00	0.00	0.00	17
成都	0.00	0.00	0.00	16
杭州	0.00	0.00	0.00	17
武汉	0.00	0.00	0.00	17
深圳	0.00	0.00	0.00	17
西安	0.20	0.56	0.30	16
重庆	0.12	1.00	0.21	16
accuracy			0.14	181
macro avg	0.03	0.14	0.05	181
weighted avg	0.03	0.14	0.05	181



支持向量机：

最佳参数组合： {'C': 1000}

	precision	recall	f1-score	support
上海	0.27	0.25	0.26	16
其他	0.33	0.12	0.18	16
北京	0.29	0.38	0.32	16
南京	0.80	0.24	0.36	17
广州	0.00	0.00	0.00	17
成都	0.22	0.69	0.33	16
杭州	0.28	0.41	0.33	17
武汉	0.78	0.41	0.54	17
深圳	0.38	0.18	0.24	17
西安	0.53	0.56	0.55	16
重庆	0.32	0.44	0.37	16
accuracy			0.33	181
macro avg	0.38	0.33	0.32	181
weighted avg	0.38	0.33	0.32	181



对比发现，首先效果都不算很好，可能是因为数据集的随机性较强，任何模型都不能很好的捕捉其中的规律。其中，随机森林模型效果相对较好，但对每一类别的效果不平均，我认为这是因为数据层面对一些类别的规律性较强，对另一些类别难以区分，如果能更多维度地采集更多的信息，模型的效果可能会更好。

聚类模型：对上市公司进行聚类分析

除了爬虫 1 和爬虫 2 的数据的内连接表以外，我还通过时序数据统计公司股票的“开盘价”、“收盘价”、“最高价”和“最低价”的统计量数据，以提供更多的语义信息。

```
stats_lis = []
for code in stock_code_lis:
    file_path = f"{code}.csv"
    stock_df = pd.read_csv(file_path)
    stock_df = stock_df.dropna(subset=['开盘', '收盘', '最高', '最低'])
    open_avg = stock_df['开盘'].mean()
    close_avg = stock_df['收盘'].mean()
    high_avg = stock_df['最高'].mean()
    low_avg = stock_df['最低'].mean()
    open_max = stock_df['开盘'].max()
    close_max = stock_df['收盘'].max()
    high_max = stock_df['最高'].max()
    low_max = stock_df['最低'].max()
    open_min = stock_df['开盘'].min()
    close_min = stock_df['收盘'].min()
    high_min = stock_df['最高'].min()
    low_min = stock_df['最低'].min()
    stats_lis.append({
        'code': code,
        'company': df[df['code'] == code]['company'].values[0],
        'open_avg': open_avg,
        'close_avg': close_avg,
        'high_avg': high_avg,
        'low_avg': low_avg,
        'open_max': open_max,
        'close_max': close_max,
        'high_max': high_max,
        'low_max': low_max,
        'open_min': open_min,
        'close_min': close_min,
        'high_min': high_min,
        'low_min': low_min
    })
stats_df = pd.DataFrame(stats_lis)
```

我首先对数据进行标准化处理，随后声明并训练了一个簇为 3 的 KMeans 模型，并最后打印轮廓系数和 Davies-Bouldin 指数来评估模型效果。

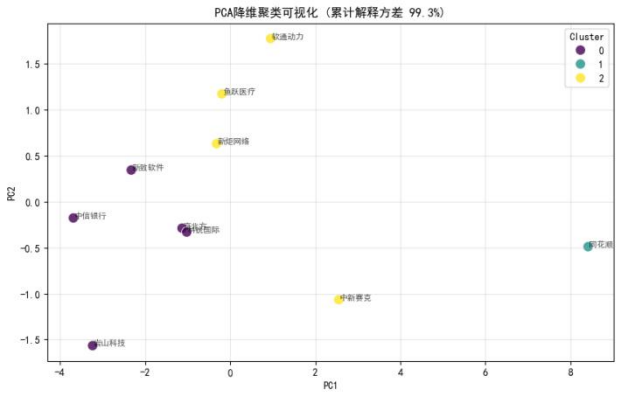
```
scaler = StandardScaler()
scaled_features = scaler.fit_transform(stats_df[features])
K = 3
kmeans = KMeans(n_clusters=K, n_init='auto', random_state=42)
```

```
stats_df['cluster'] = kmeans.fit_predict(scaled_features)
print("聚类结果：")
print(stats_df[['code', 'company', 'cluster']])
# 评估聚类效果
silhouette_avg = silhouette_score(scaled_features, stats_df['cluster'])
davies_bouldin = davies_bouldin_score(scaled_features, stats_df['cluster'])
print("\n 聚类效果评估：")
print(f"轮廓系数（Silhouette Score）: {silhouette_avg:.2f}")
print(f"Davies-Bouldin 指数: {davies_bouldin:.2f}")
```

```
聚类结果:
     code company  cluster
0  002195    岩山科技        0
1  002223    鱼跃医疗        2
2  002912    中新赛克        2
3  002987     京北方         0
4  300033     同花顺         1
5  300662    科锐国际         0
6  301236    软通动力        2
7  601998    中信银行        0
8  605398    新炬网络        2
9  688590    新致软件        0

聚类效果评估:
轮廓系数（Silhouette Score）: 0.34
Davies-Bouldin 指数: 0.63
```

原生的聚类结果难以可视化，因为数据是高维的，难以画在二维平面上，对数据进行 PCA 降维为二维，并进行可视化展示。

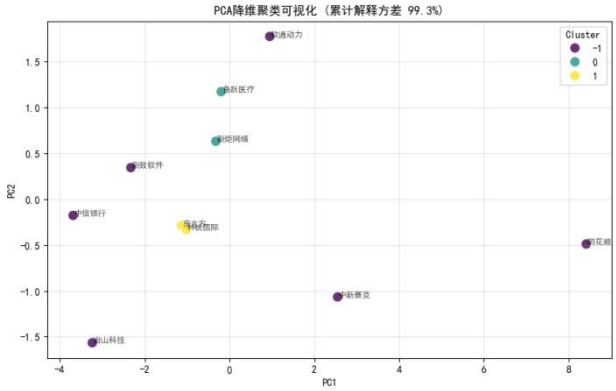
```
# 执行 PCA 降维
pca = PCA(n_components=2)
pca_features = pca.fit_transform(scaled_features)
# 创建可视化 DataFrame
plot_df = pd.DataFrame({
    'PC1': pca_features[:,0],
    'PC2': pca_features[:,1],
    'Cluster': stats_df['cluster'],
    'Company': stats_df['company']
})
# 绘制 PCA 散点图
plt.figure(figsize=(10,6))
sns.scatterplot(data=plot_df, x='PC1', y='PC2', hue='Cluster',
                palette='viridis', s=100, alpha=0.8)
plt.title("PCA 降维聚类可视化（累计解释方差 {:.1%}）".format(pca.explained_variance_ratio_.sum()))
plt.grid(alpha=0.3)
# 添加公司标签
for line in range(0, plot_df.shape[0]):
    plt.text(plot_df.PC1[line]+0.02, plot_df.PC2[line],
             plot_df.Company[line], horizontalalignment='left',
             fontsize=8, color='black', alpha=0.7)
plt.show()
```



同理，我还使用了 DBSCAN 和层次聚类进行了聚类分析，参数和结果如下。

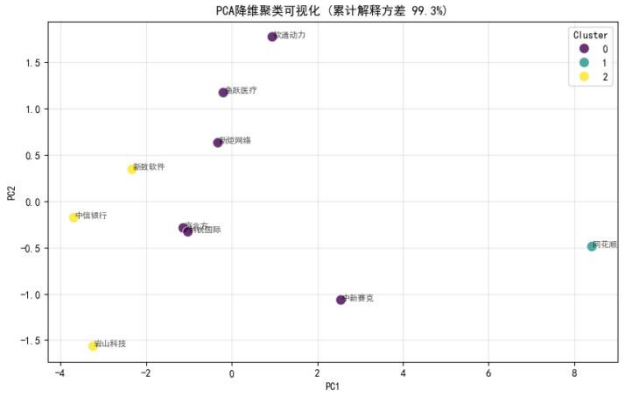
DBSCAN（eps=1.1, min_samples=2）：

```
聚类结果:
     code company  cluster
0  002195    岩山科技       -1
1  002223    鱼跃医疗        0
2  002912    中新赛克       -1
3  002987     京北方        1
4  300033     同花顺       -1
5  300662    科锐国际        1
6  301236    软通动力       -1
7  601998    中信银行       -1
8  605398    新炬网络        0
9  688590    新致软件       -1

聚类效果评估:
轮廓系数（Silhouette Score）: 0.62
Davies-Bouldin 指数: 0.38
```



层次聚类（n_clusters=3）：

```
聚类结果:
     code company  cluster
0  002195    岩山科技        2
1  002223    鱼跃医疗        0
2  002912    中新赛克        0
3  002987     京北方        0
4  300033     同花顺        1
5  300662    科锐国际        0
6  301236    软通动力        0
7  601998    中信银行        2
8  605398    新炬网络        0
9  688590    新致软件        2

聚类效果评估:
轮廓系数（Silhouette Score）: 0.34
Davies-Bouldin 指数: 0.55
```



从轮廓系数的角度来讲，DBSCAN 算法的效果最好，DBSCAN 将京东方和科瑞国际分为一类，鱼跃医疗和新矩网络分为另一类，可能这些公司对 python 岗位的招收用人标准相似。

结论：

依托实验结果，我对 python 岗位求职者的建议如下：

1、选择经济发达、IT 产业发达地区的公司和工作，这些工作统计上能提供更多的薪资和更好的福利待遇。

2、选择上市公司或大公司，上市公司统计上能提供更多的薪资和更好的福利待遇。

3、选择股票总市值和流通市值并且呈上升趋势的公司，这些公司统计上能提供更多的薪资和更好的福利待遇。

4、当准备一个公司的求职但被拒绝时，可以考虑被聚为同一类的其他公司，这些公司可能有相似的求职要求和员工待遇。

5、提高学历以保证竞争力。

6、经验是能力的一大方面，要发扬自己的经验优势，弥补经验上的劣势。

四、实验创新点

1、爬虫技术的创新应用：

本次实验在爬虫阶段采用了多种技术手段以获取全面且准确的数据。首先，针对需要用户登录才能访问数据的网站，如拉勾网，实验中巧妙地运用了 Selenium 技术模拟用户登录行为。通过手动登录的方式，成功突破了网站的访问限制，从而能够爬取到高质量的岗位信息数据。

在爬取证券之星股票信息时，运用了 Scrapy 框架。该框架具有高效、可扩展性强的特点，能够轻松应对大规模数据的爬取任务，为后续的数据分析提供了丰富的数据基础。

对于股票的日际数据获取，则借助了 AKShare API。这一 API 接口提供了便捷的数据获取方式，能够快速获取股票的历史数据，大大提高了数据采集的效率和准确性。

2、数据分析的多维度：

实验跳出了单一数据类型分析的局限，将 Python 岗位信息与股票市场数据相结合，开创性地从多个维度挖掘了两者之间的潜在关系。通过对岗位数据的深入分析，不仅能够了解 Python 岗位的市场需求、薪资水平和技能要求等基本信息，还能借助股票市场数据，进一步探讨不同公司在股票市场上的表现与 Python 岗位招聘之间的关联性。

在分析过程中，巧妙地运用了数据可视化技术，通过绘制各种图表（如柱状图、折线图、散点图等）直观地展示了复杂的数据关系，使分析结果更加易于理解和解释。

3、聚类分析的应用：

实验运用聚类分析方法对上市公司进行了详细的分类，从多个角度揭示了不同公司在 Python 岗

位招聘方面的特点和规律。通过对公司的聚类分析，能够将具有相似招聘模式和岗位要求的公司归为一类，为求职者提供了更清晰的公司选择依据。

在聚类过程中，综合考虑了公司的薪资水平、岗位数量、公司规模、融资情况以及福利待遇等多个因素，确保了聚类结果的全面性和准确性。同时，还运用了多种聚类算法（如 K-Means、DBSCAN 和层次聚类）进行实验对比，通过评估指标（如轮廓系数和 Davies-Bouldin 指数）来选择最优的聚类算法，从而进一步提高了聚类分析的质量和可靠性。

3、机器学习模型优化：

在数据挖掘阶段，实验采用了多种机器学习算法（如线性回归、多项式回归、随机森林、支持向量机、KNN 等）。在模型训练过程中，运用了网格搜索和交叉验证等技术，对模型的参数进行了系统的调整和优化，从而找到了最优的模型参数组合。通过对比不同算法的预测结果，能够更全面地了解各算法在不同数据集上的表现特点，为实际问题选择最合适的算法提供了有力的支持。

六、实验总结

本次实验涵盖了 Python 数据分析的完整流程，从数据采集到可视化展示，全面提升了数据分析能力。通过实验，成功采集并清洗了大量岗位和股票数据，为后续的分析和建模提供了坚实的数据基础。在数据采集阶段，熟练掌握了 Selenium、Scrapy 和 AKShare 等工具的使用方法，能够从多种来源获取数据。数据清洗阶段，学会了处理缺失值、重复值和异常值等常见问题，提高了数据质量。

在数据挖掘阶段，构建了多种数据挖掘模型，并通过可视化手段呈现了分析结果。实验创新性地结合了爬虫技术、机器学习算法和数据可视化，为求职者提供了有价值的参考和建议。

在实验过程中，遇到了各种技术难题，如爬虫被反爬虫机制阻止、数据清洗过程中的数据不一致性问题、模型训练中的过拟合问题等。通过查阅大量资料和不断尝试，成功找到了解决方案，克服了这些困难。这一过程不仅提高了技术能力，还培养了分析和解决问题的能力，学会了如何在面对复杂问题时保持冷静和耐心，逐步找到突破口。

实验中运用了多种技术和工具，如 Python 编程语言及其数据分析库（NumPy、pandas、Matplotlib、Seaborn、Scikit-learn 等）、Selenium 和 Scrapy 爬虫框架、MySQL 数据库等，深刻体会到了不同技术的特点和应用场景。

实验的成果为 Python 求职者提供了宝贵的参考信息。通过对岗位数据的分析，求职者可以了解当前市场的薪资水平、技能要求和热门岗位等信息，从而更好地规划自己的职业发展路径。个性化的岗位推荐模型能够为求职者量身定制推荐岗位，提高求职效率。同时，实验也为招聘者提供了有价值的 insights，帮助其了解市场供需关系和竞争对手的招聘策略，从而优化自身的招聘流程和岗位设置。

依托实验结果，我对 python 岗位求职者的建议如下：

7、选择经济发达、IT 产业发达地区的公司和工作，这些工作统计上能提供更多的薪资和更好的福利待遇。

8、选择上市公司或大公司，上市公司统计上能提供更多的薪资和更好的福利待遇。

9、选择股票总市值和流通市值并且呈上升趋势的公司，这些公司统计上能提供更多的薪资和更好的福利待遇。

10、当准备一个公司的求职但被拒绝时，可以考虑被聚为同一类的其他公司，这些公司可能有相似的求职要求和员工待遇。

11、提高学历以保证竞争力。

12、经验是能力的一大方面，要发扬自己的经验优势，弥补经验上的劣势。

尽管实验取得了一定的成果，但也存在一些不足之处。例如，数据采集的范围和规模有限，可

能无法完全反映整个 Python 岗位市场的全貌；部分模型的预测精度还有待提高，需要进一步优化算法和参数。在进一步的工作中，将继续扩大数据采集的范围和规模，引入更多维度的数据进行分析。同时，将进一步深入研究机器学习算法，探索新的模型架构和优化方法，提高模型的性能和预测精度。

附录：

《Python 数据分析》实验成绩评定表

考察点	具 体 要 求	满分	得分
基本内容	能够按照实验要求合理设计并开发出程序，实验态度认真，能够完整记录实验数据、原理及实验结果。	20	
分析	经过分析，除了基本内容以外，设计并实现额外功能，使系统功能更加完善。	20	
创新	实验设计、结果分析具有创新性。	10	
报告质量	报告内容充实、正确，实验内容记录完整，实验报告格式正确，文档提交完整、正确。	10	
总计			