上海交通大學

SHANGHAI JIAO TONG UNIVERRSITY

数据挖掘

课程设计报告

题目：前程无忧网站数据爬取分析

组长：沈琪 518021911196
组员：唐铭江 518021911198
组员：许阳帆 518021910866
组员：胡定伟 518021910071
专业：网络空间安全
学院：电子信息与电气工程学院

目录

1. 项目内容

- 互联网数据爬取：从前程无忧网站爬取上海地区职位薪资，具体分组任务见下一节；
- 中文分词和预处理：如爬取内容中包含自然语言，则需做分词数据转换为结构化数据，还需要进行缺失值填充和数据清洗等操作，自建变量，如职位名关键词、公司名关键词、工作地点等等；
- 数据建模、调优和测试：建立建立薪资预测模型（预测薪资下限或上限），调优超参数，并给出模型的性能；
- 报告撰写：将主要流程撰写成文档，并上传代码和爬取的数据集。

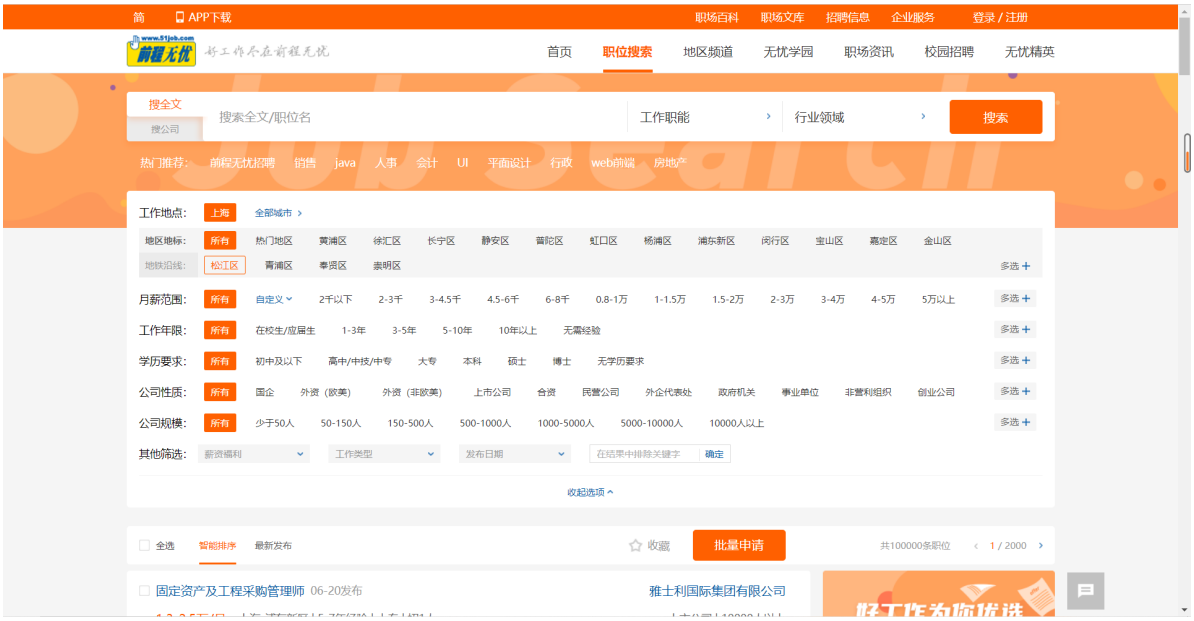
2. 数据爬取

2.1 环境配置

`PyCharm` 2021.1.1 (Professional Edition)； Python 3.8

2.2 爬取对象

前程无忧网站：



2.3 具体实现

2.3.1 目标URL

分析网站URL格式https://search.51job.com/list/000000,000000,0000,00,9,99,%2520,2,1.html。当在网站中选择目标地区为上海地区时，URL格式为https://search.51job.com/list/020000,000000,0000,00,9,99,%2520,2,1.html。可以看到，爬取上海地区薪资时，只需要将第一段000000改为020000。

为了方便数据处理，在爬取内容时，需要通过网站自带的工作职能分类分别进行爬取。比如在工作职能中选择机械机床+技工普工：

可以看到此时的URL为https://search.51job.com/list/020000,000000,6200%2c3700,00,9,99,%20,2,1.html，也就是只有0000段发生了变化。所以在爬取不同职位的信息时只需要更改0000段的内容即可。

2.3.2 换页爬取

浏览网站可以发现网站每页爬取的信息有限，如果要爬取大量内容需要换页爬取。

换至第二页，网站URL为https://search.51job.com/list/020000,000000,0000,00,9,99,+,2,2.html

换至第三页，网站URL为https://search.51job.com/list/020000,000000,0000,00,9,99,+,2,3.html

可以看到，换页只会变化URL中的x.html。其中URL中出现了2.3.1中URL没有的“+”，这表示网站自带的搜索框中内容为空。

2.3.3 `getfront`函数

该函数用于构建访问的URL并返回html原码。

```
def getfront(page,item):        #page是页数，item是输入的字符串
     result = urllib.parse.quote(item)                    #先把字符串转成十六进制编码
     url = result+',2,'+ str(page)+'.html'
     url2 = 'https://search.51job.com/list/020000,000000,6200%252c3700,00,9,99,'
        #6200%252c3700指查询机械机床+技工普工的页面，查询不同目标需要更改此处
     res = url2+url1     #拼接网址
     a = urllib.request.urlopen(res)
     html = a.read().decode('gbk')       # 读取源代码并转为unicode
     html = html.replace('\\', '')        # 将用于转义的"\"替换为空
     html = html.replace('[', '')
     html = html.replace(']', '')
     return html
```

该函数传入参数page、item，在main函数中page不断增加，用来不断换页爬取；item是接收用户输入的字符串，设计它的目的是让用户直接在控制台输入想要查询的内容，程序会自动定位对应URL，但在实际操作中由于靠用户输入的关键词搜索到的内容分类混杂，所以采用了手动修改URL的方式，在运行爬虫时程序会等待用户输入，此时输入空格即可让程序爬取已经指定的URL。接收用户输入字符串的功能仍作保留是考虑到用户仍然有可能自定义关键词爬取。

直接指定的URL是根据2.3.1和2.3.2的分析设定的。

2.3.4 `getInformation`函数

此函数用于从`getfront`返回的html中提取所需内容。

以下是返回的html中关键信息的示例：

```
window.__SEARCH_RESULT__ = {"top_ads":[],"auction_ads":[],"market_ads":
[],"engine_search_result":[{"type":"engine_search_result",
"jt":"0_0","tags":[],
"ad_track":"",
"jobid":"132934338",
"coid":"116614",
"effect":"1",
"is_special_job":"",
"job_href":"https:\/\/jobs.51job.com\/shanghai-pdxq\/132934338.html?
s=sou_sou_soulb&t=0_0",
"job_name":"固定资产及工程采购管理师",
"job_title":"固定资产及工程采购管理师",
"company_href":"https:\/\/jobs.51job.com\/all\/co116614.html",
"company_name":"雅士利国际集团有限公司",
"providesalary_text":"1.3-2.5万\/月",
"workarea":"021000",
"workarea_text":"上海-浦东新区",
"updatedate":"06-20",
"iscommunicate":"",
"companytype_text":"上市公司",
"degreefrom":"5",
"workyear":"6",
"issuedate":"2021-06-20 10:03:31",
"isFromXyz":"",
"isIntern":"",
"jobwelf":"做五休二 周末双休 带薪年假 五险一金 绩效奖金 节日福利 专业培训",
"jobwelf_list":["做五休二","周末双休","带薪年假","五险一金","绩效奖金","节日福利","专业培
训"],
"isdiffcity":"",
"attribute_text":["上海-浦东新区","5-7年经验","大专","招1
人"],"companysize_text":"10000人以上",
"companyind_text":"快速消费品(食品、饮料、化妆品)","adid":""}
,......]
```


根据上面的内容，`getInformation`函数设计如下：

```
def getInformation(html):
    reg = re.compile(r'\{"type":"engine_search_result","jt":"0".*?"job_href":"
(.*?)","job_name":"(.*?)".*?"company_href":"(.*?)","company_name":"
(.*?)","providesalary_text":"(.*?)".*?"updatedate":"
(.*?)".*?,'r'"companytype_text":"(.*?)".*?"jobwelf":"(.*?)".*?"attribute_text":"
(.*?)","(.*?)","(.*?)","(.*?)","companysize_text":"(.*?)","companyind_text":"
(.*?)","adid":""},',re.S)#匹配换行符
    items=re.findall(reg,html)
    print(items)
    return items
```

从当中提取`job_href`、`job_name`、`company_href`、`company_name`、`providesalary_text`、`updatedate`、`companytype_text`、`jobwelf`、`attribute_text`、`companysize_text`、`companyind_text`中的信息并存放到数组中。

2.3.5 写入到excel中

新建excel表

```
excel1 = xlwt.Workbook()
# 设置单元格格式
sheet1 = excel1.add_sheet('Job', cell_overwrite_ok=True)
sheet1.write(0, 0, '序号')
sheet1.write(0, 1, '职位')
sheet1.write(0, 2, '公司名称')
sheet1.write(0, 3, '公司地点')
sheet1.write(0, 4, '公司性质')
sheet1.write(0, 5, '薪资')
sheet1.write(0, 6, '学历要求')
sheet1.write(0, 7, '工作经验')
sheet1.write(0, 8, '公司规模')
sheet1.write(0, 9, '公司福利')
sheet1.write(0, 10, '发布时间')
sheet1.write(0, 11, '公司信息链接')
sheet1.write(0, 12, '分类')
```

在主函数中写入

```
sheet1.write(number, 0, number)
sheet1.write(number, 1, i[1])
sheet1.write(number, 2, i[3])
sheet1.write(number, 3, i[8])
sheet1.write(number, 4, i[6])
sheet1.write(number, 5, i[4])
sheet1.write(number, 6, i[10])
sheet1.write(number, 7, i[9])
sheet1.write(number, 8, i[12])
sheet1.write(number, 9, i[7])
sheet1.write(number, 10, i[5])
sheet1.write(number, 11, i[2])
sheet1.write(number, 12, '技工')  #这一项对应URL定义的目标，'技工'对应机械机床+技工普工的页面查询到的内容
```

2.3.6 main函数

```
number = 1
item = input()
for j in range(1, 100):   #页数自己随便改
    try:
        print("正在爬取第"+str(j)+"页数据...")
        html = getfront(j, item)      #调用获取网页原码
        for i in getInformation(html):
            try:
                sheet1.write(number, 0, number)
                sheet1.write(number, 1, i[1])
                sheet1.write(number, 2, i[3])
                sheet1.write(number, 3, i[8])
                sheet1.write(number, 4, i[6])
                sheet1.write(number, 5, i[4])
                sheet1.write(number, 6, i[10])
                sheet1.write(number, 7, i[9])
```

```
                sheet1.write(number, 8, i[12])
                sheet1.write(number, 9, i[7])
                sheet1.write(number, 10, i[5])
                sheet1.write(number, 11, i[2])
                sheet1.write(number, 12, '技工')
                number+=1
            except:
                pass
        excel1.save("技工.xls")
        time.sleep(3)  # 休息间隔，避免爬取海量数据时被误判为攻击，IP遭到封禁
    except:
        pass
```

用户输入空格，爬取对应程序中已定义的URL下1到100页的内容。

爬取的6万余条结果在data文件夹下，代码原码见代码附录。

3. 数据预处理

数据预处理的目的是处理我们在前程无忧网上爬取到的原始数据，对缺失值补全，整合文本数据等，从而能让后续数据可视化与数据分析的操作更加方便与准确。我们采用了 python 的 pandas 库提供函数进行处理。

3.1 公司种类预处理

在前程无忧网上，公司种类指的是公司的一栏共有"外资（欧美）"、"民营公司"、"上市公司"等 11 类，但同时也有部分数据缺失，这里我们将这些缺失的数据作为垃圾数据剔除，以提高模型的准确率。这些公司种类与薪资并非直接挂钩，因此并不需要进行什么数字化。我们直接用 pandas 提供的 get_dummies 函数进行了独热码化。

['外资（欧美）', '民营公司', '上市公司', '合资', '外资（非欧美）', '创业公司', '国企', '事业单位', '非营利组织', '外企代表处', '政府机关']

3.2 薪资预处理

从网站上获得的薪资数据只有一个特征，但是往往包括最大薪水和最小薪水，这里我们将其处理成最大薪水、最小薪水、平均薪水三个特征，显然这三个特征并非独立，这里这样处理只是为了后续处理的机动性和方便性。同样，就算是薪水一样有缺失值，毫无疑问，这类数据我们将直接剔除。薪水处理的复杂在于，这个特征理论上应当是一个整数值，但实际前程无忧网上提供了许多种表示方式，如元/日、千/月、万/年、千/年等，这就需要对不同的格式进行不同的处理，需要考虑的因素非常多。在实际的数字化中，我们将其全部化为年薪。如100元/日转化成 100 * 365 = 36500元/年，1000元/月转化成 1000 * 12 = 12000元/年。

3.3 学历要求预处理

对于学历要求，原本爬取到的数据掺杂了一些错误数据，信息中将招几人放到了学历一栏，这些数据作为缺失值处理。

['本科', '大专', '招2人', '硕士', 'companysize_text":"50-150人', '招若干人', 'companysize_text":"10000人以上', '高中', '中专', '招3人', '中技', 'companysize_text":"150-500人', '博士', '招1人', 'companysize_text":"少于50人', '初中及以下', 'companysize_text":"500-1000人', 'companysize_text":"1000-5000人', '招5人', '招4人', '招14人', '招10人', 'companysize_text":"5000-10000人', '招20人', '招15人', '招11人', '招7人', '招30人', '招55人', '招50人', '招9人', '招6人', '招100人', '招8人', '招25人', '招13人', '招66人', '招99人', '招31人', '招22人', '招12人', '招40人', '招39人', 'companysize_text":"', '招49人', '招59人', '招16人']

处理完后的学历要求共有 9 类，分别是"初中及以下"，"中技"，"中专"，"高中"，"大专"，"本科"，"硕士"，"博士"和缺失值。由于学历存在高低之分，显然学历越高，薪资的平均水平也会提高。因此，我们根据学历的高低，将学历进行细化的数值化，越高的学历数字越高。这里我们设定"博士"对应 8，"初中及以下"对应 1，而缺失则取中位数 3。所以最终的取值列表为 [1, 2, 3, 4, 5, 6, 7, 8]。

3.4 工作经验预处理

工作经验也同样有一些错误数据，信息中存在部分学历和招聘人数当作工作经验的情况，这里我们将其作为缺失值处理。

```
['5-7年经验', '3-4年经验', '无需经验', '2年经验', '1年经验', '本科', '8-9年经验', '招1人', '招若干人', '大专', '10年以上经验', '招2人', '硕士', '博士', '招5人', '在校生/应届生', '招3人', '招10人', '中专', '初中及以下', '招6人', '招4人', '招8人', '高中', '招7人', '招20人', '中技']
```

处理完后的工作经验共有7类，分别是“无需经验”（包括应届生和缺失值），“1年经验”，“2年经验”，“3-4年经验”，“5-7年经验”，“8-9年经验”，“10年以上经验”。同样的，工作经验往往是与薪资直接挂钩的，我们也根据工作经验的高低，将工作经验进行细化的数值化，越高的工作经验数字越高。这里我们直接取要求的年限均值作为数值。例如“5-7年经验”，对应 （5+7）/ 2 = 6；1年对应1。所以最后的数值取值列表为 [0.0, 1.0, 2.0, 3.5, 6.0，8.5, 10.0]。

3.5 公司规模预处理

公司规模主要包括 '少于50人'，'50-150人'，'150-500人'，'500-1000人'，'1000-5000人'，'5000-10000人'，'10000人以上' 这7类，同样，我们认为一家企业的人数与薪资有着一定的正比关系，所以使用人数范围的均值作为处理后的数值。最后的数值取值列表为 [25, 100, 325, 750, 10000, 3000, 7500]。

3.6 岗位地区预处理

因为我们是对网站上上海地区的岗位进行爬取，所以基本所有岗位都在上海。但依旧有一些在上海招聘但是岗位实际在外地的情况，这一类被统称为异地招聘。其余的地区包括 '上海-浦东新区', '上海-徐汇区', '上海-闵行区', '上海-嘉定区', '上海-松江区', '上海-杨浦区', '上海-宝山区', '上海-静安区', '上海-黄浦区', '上海-青浦区', '上海-长宁区', '上海-奉贤区', '上海-普陀区', '上海-金山区', '上海-虹口区', '上海-崇明区'，包括了上海共16个区。但是还是存在一些岗位只填了上海，并没有写具体的区。剔除这些数据难免可惜，所以这里我们将这类岗位也作为一个类别。所以综上一共有18类。显然，将地区分成市区郊区这样与薪资挂钩是可笑而没有意义的，所以，我们依旧使用了 pandas 提供的 get_dummies 函数对其进行了独热码化。

3.7 岗位行业预处理

我们爬取的数据中，岗位行业包括种类有，'半导体/芯片', '财务/会计', '餐饮服务', '电子商务', '后端开发', '建筑设计施工', '教育行业', '律师行业', '汽车设计制作', '前端开发', '生产运营', '市场营销', '销售管理', '游戏开发', '制药医疗器械' 共16类。这里我们也一样对其进行了独热码处理。

3.8 公司福利预处理

职员福利是较为难以处理的数据。在原始的数据中，一个职位可能存在任意多的员工福利，具体如'五险一金', '餐饮补贴', '股票期权'等。并且，福利的细分种类相当多，极为难以操作。 因此，我们首要需要做的是找出最主要的福利种类。这里我们搜索了所有的数据，将每个职位的福利用 split 函数切割（每个元福利间都会用空格分开），分成一个个元福利，再进行统计技术， 仅留下其中出现次数超过2000的数据，共 24 类，我们以字典的形式给出具体元福利和出现的次数：

```
{'五险一金': 46437, '餐饮补贴': 19488, '股票期权': 3028, '弹性工作': 11153, '年终奖金': 27246, '补充医疗保险': 7118, '带薪年假': 14525, '绩效奖金': 28726, '专业培训': 22069, '节日福利': 12440, '免费班车': 5185, '定期体检': 20321, '交通补贴': 12529, '通讯补贴': 12129, '周末双休': 7991, '做五休二': 7163, '员工旅游': 18060, '包吃': 2459, '补充公积金': 2604, '出国机会': 3537, '包住宿': 2011, '全勤奖': 3478, '高温补贴': 2784, '加班补贴': 2447}
```

之后，我们为每一类福利分配一列 0-1 编码的数值化属性。若某职位的福利经过 re 的正则匹配存在该关键福利词子串，则该项设置为 1。

3.9 职位预处理

职位预处理是非常难处理的数据。原始数据中的职位名称五花八门，甚至一样的职位也能是不同的名字。所以对其的特征提取相当困难。我们认为神经网络或者有监督的机器学习是最好的办法，但是因为时间有限，这里我们简单处理为关键词提取。首先我们需要找到所有职位名称中出现次数较多的子字符串。所以我们设计了 get_all_substrings 函数用于获取一个字符串的所有子字符串（具体代码见代码附录）。再进行遍历计数，找出出现次数最多的有意义的16个子字符串作为特征，然后进行独热编码。我们以字典的形式给出具体提取的特征和出现的次数：

```
{'工程师': 16069,'设计': 3511,'经理': 9212, '高级': 2298,'销售': 4771,'开发': 6311, '前端': 2265,'市场': 3546, '总': 2359, '主管': 4018,'助理': 2503, '专员': 5424, '项目': 2023, '运营': 4185,'电商': 2396, '法务': 2182}
```

之后，我们为每一类职位名称特征分配一列 0-1 编码的数值化属性。若某职位名称特征经过 re 的正则匹配存在该职位名称子串，则该项设置为 1。

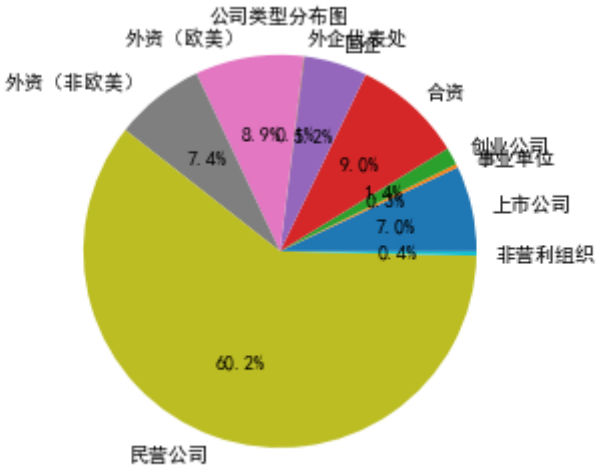
4.特征分析

特征分析将对数据进行可视化的处理，使其表达直观，并利用一些不同的相关系数探讨不同变量之间的联系。文件夹中的csv文件只是便于读取，没有进行更改。

4.1 总体情况分布

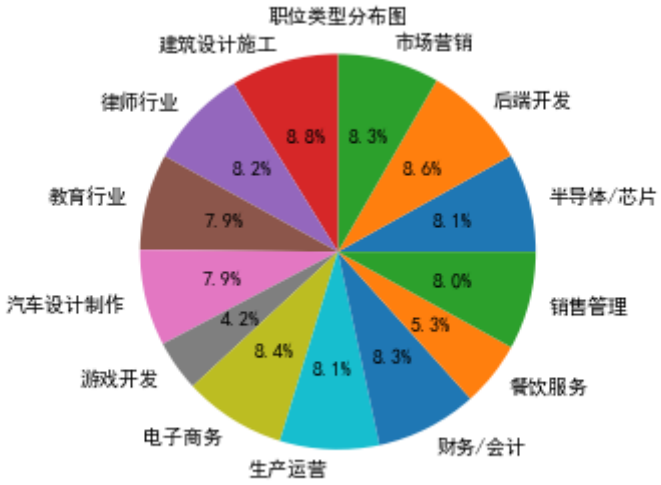
4.1.1公司类型分布

按照先前的划分查看岗位所属公司的分布。可以看出民营公司占据了一半以上的岗位数额，拥有着大量需求，除此之外，外企以及合资企业也有相当一部分，国企则可能由于发布岗位会通过官方途径，所以在网站上的较少。



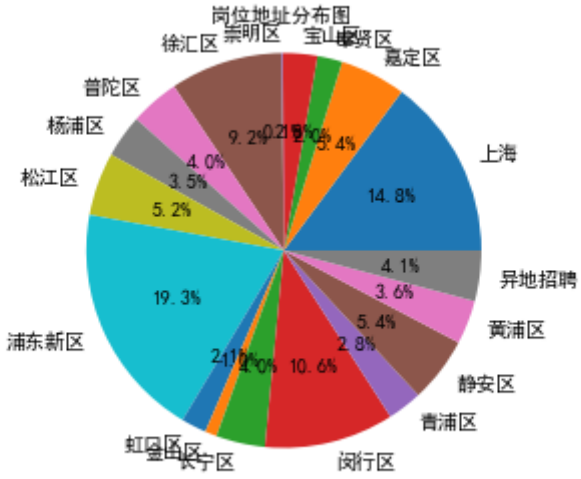
4.1.2 职位类型分布

在现有的分类下，岗位显得近乎完美地平均，但是仔细观察后可以发现计算机相关行业包括前后端以及游戏开发等内容，因此总占比会大一些，也体现了目前IT行业的岗位需求大。



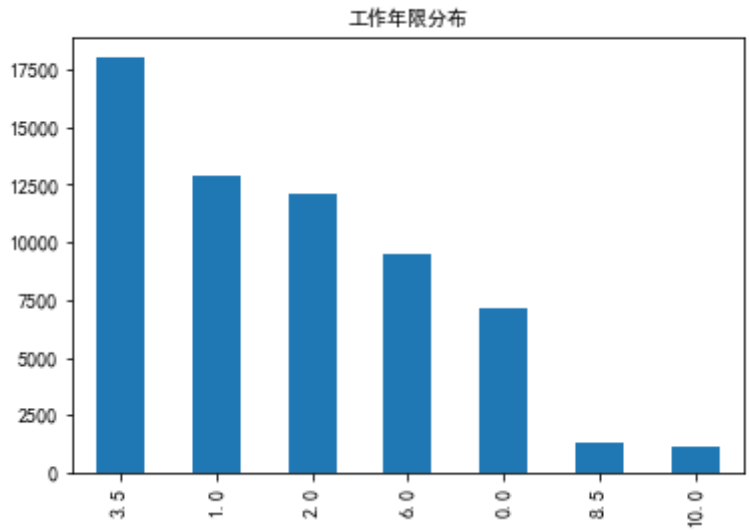
4.1.3 岗位地址分布

浦东新区、徐汇、闵行和没有详细位置的上海占比较大，原因也是比较明显的，浦东和徐汇都是中心地带也就是市区，闵行则在大学附近有大量的科学产业园区。



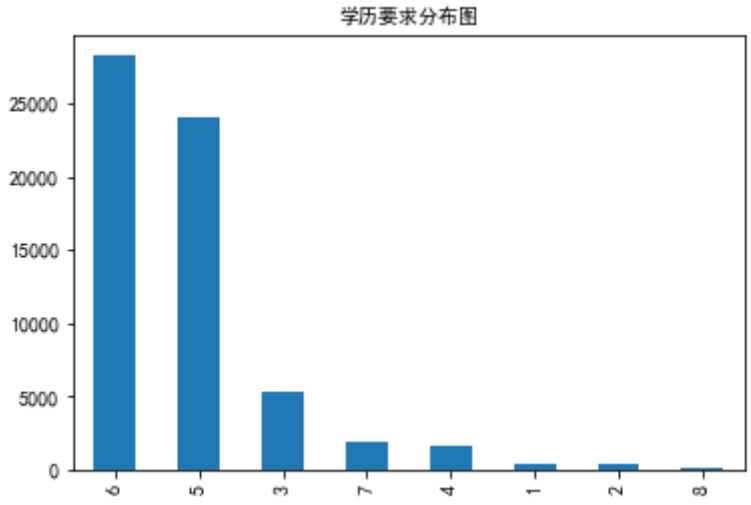
4.1.4 工作年限分布

工作经验要求整体上呈现正态分布，以3-4年最多。



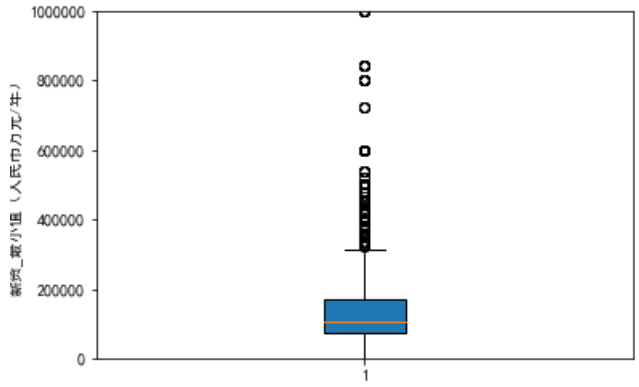
4.1.5 学历要求分布

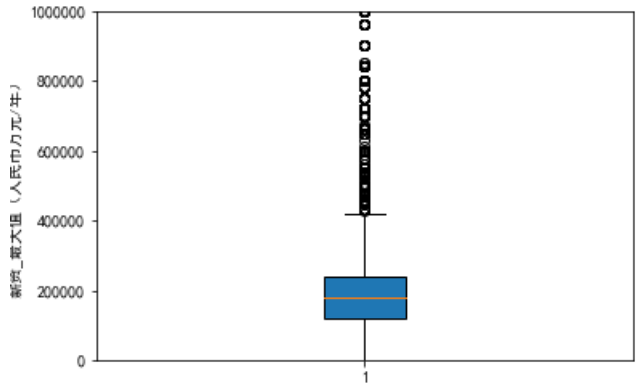
同样地，我们可以看出，本科和大专学历的要求最多，本科生学历足以满足大多数岗位的学历要求，有着硕士和博士要求的极少。

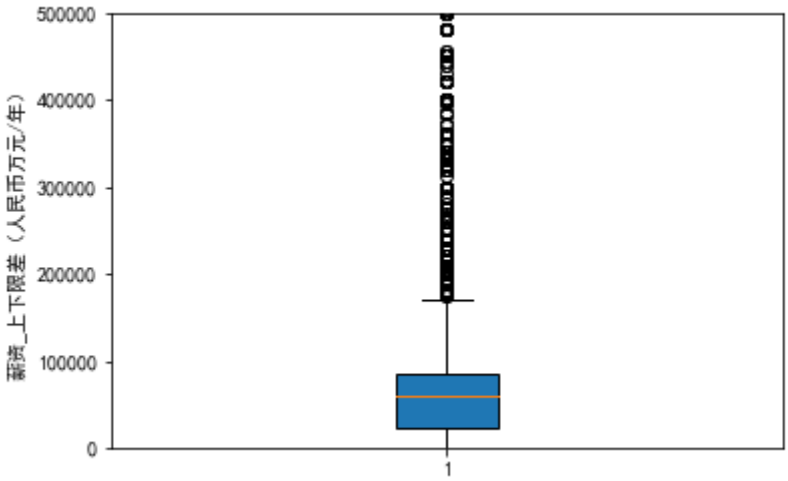


4.1.6 薪资分布

采用箱线图将薪资进行可视化，并标出中位数，大部分薪资都在10-20万的区间内，但是也有很多岗位远远超出了这个范围，被视为"异常数据"，同一个应聘岗位的上下限之差也有近10万。（三张图分别为薪资最小值，最大值和上下限差值）





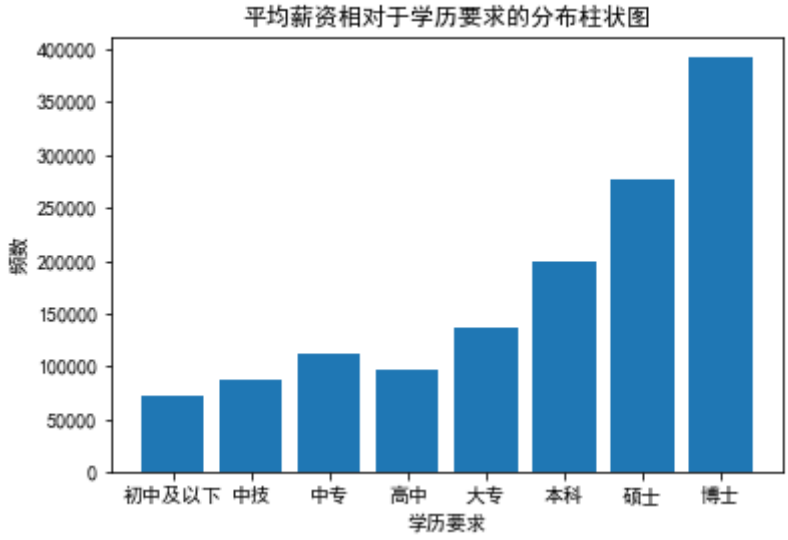


4.2相关性分析——可视化

在相关性分析中，我们首先将直观地进行平均薪资与不同要求的对应关系，然后再采用一些数值方法进行分析。

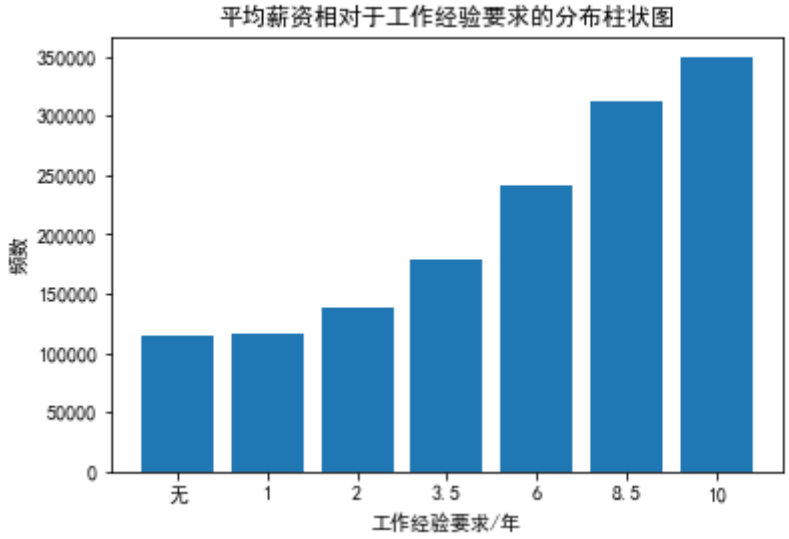
4.2.1 平均薪资与学历要求

可以看出两者是明显的正相关关系，即学历越高，工资的期望也就越高，这点是符合认知的，有一点需要注意的是由于将缺省学历要求设置为中位数——中专，所以图中中专的平均薪资略高于下一个学历。



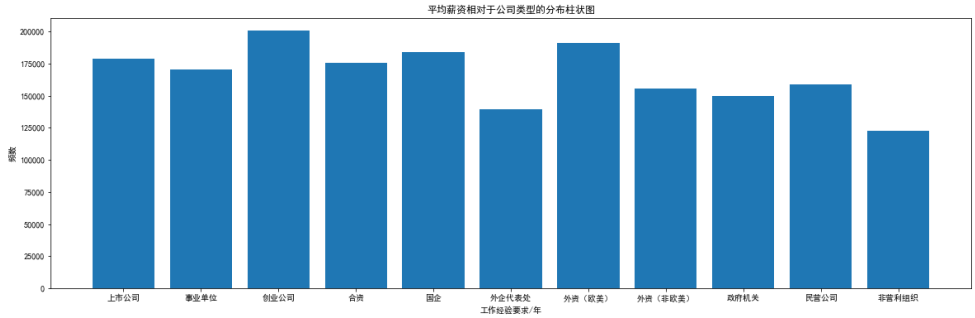
4.2.2 平均薪资与工作经验

同样地，”越老越吃香“的道理也是宏观上正确的，需要的工作经验越长，平均薪资就越高。



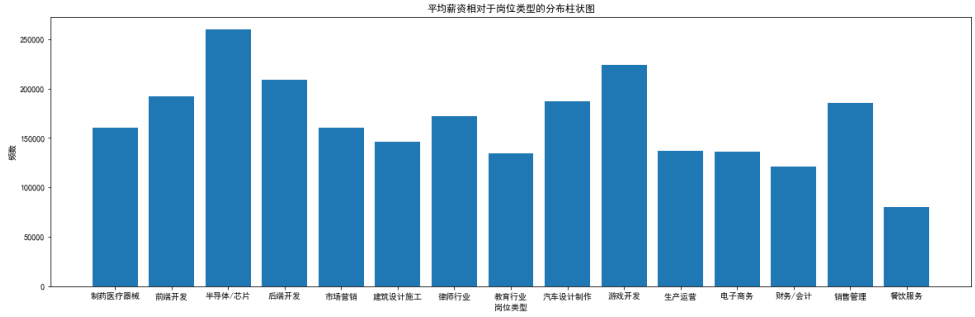
4.2.3 平均薪资与公司类型

可以看出公司类型整体与工资水平关系不大，除了外资代表处和非营利组织明显稍少。



4.2.4 平均薪资与公司类型

不同岗位地平均薪资差距较大，也不好进行概述，但是仍然可以看出前面归纳的IT行业薪资较高，餐饮服务业明显较低。



4.3 相关性分析——数值方法

Pearson, Spearman, Kendall 三类相关系数是统计学上的三大重要相关系数，表示两个变量之间变化的趋势方向和趋势程度，将采用这三种系数分别对一些和薪资相关性较强的属性进行分析，并以热图（heatmap）的形式进行呈现。

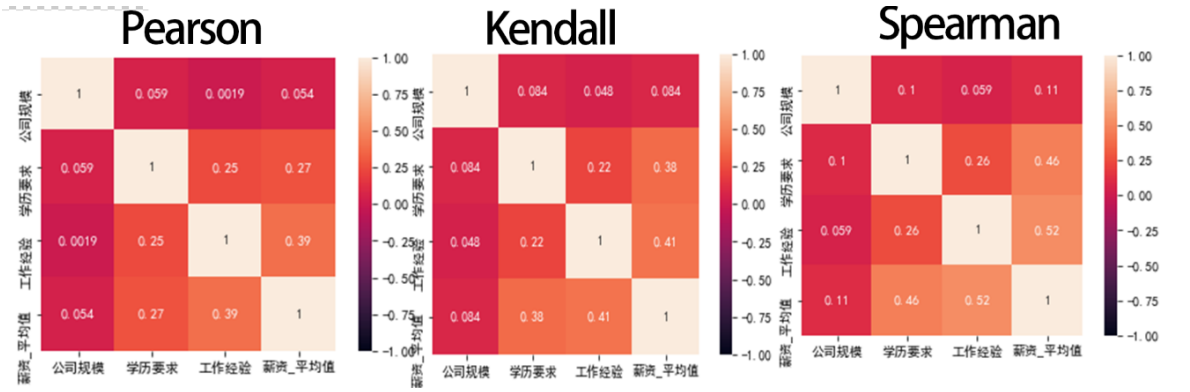
Pearson相关性系数的值等于它们之间的协方差cov(X,Y)除以它们各自标准差的乘积(σX, σY)，描述的是线性相关关系，取值[-1, 1]。负数表示负相关，正数表示正相关。在显著性的前提下，绝对值越大，相关性越强。绝对值为0， 无线性关系；绝对值为1表示完全线性相关。

Spearman相关性系数是无参数的等级相关系数，亦即其值与两个相关变量的具体值无关，而仅仅与其值之间的大小关系有关。di表示两个变量分别排序后成对的变量位置差，N表示N个样本，减少异常值的影响。

Kendall 相关性系数R=（P-(n*(n-1)/2-P))/(*n*(n-1)/2)=(4P/(n*(n-1)))-1，属于等级相关系数。排序一致，则为1， 排序完全相反则为-1。

将平均薪资、工作经验、学历要求和公司规模四个参数进行分析，三种方法的结果如下，分别是Pearson，Spearman和Kendall，颜色越浅表明相关性越强。

总体可以得出公司规模与平均薪资关系不大，学历和工作经验与薪资成正相关的结论，这也与先前的作图分析是相吻合的。不过由于学历的量化与实际情况存在一定偏差，从1到2和到7到8显然是完全不同的，因此不能很贴合地体现相关性。同样地，平均薪资、学历要求和工作经验三者之间都是正相关的，而公司规模与剩下三者关系不大，可以看出公司人数的多少与岗位情况没有什么相关性。



从不同的相关系数来看，将三种方式得到的结果放入同一个表格进行比对，数据也都是相近的，不过根据它们不同的使用场景，对不满足双联合正态分布，连续的参数来说Spearman系数最合适，Spearman系数也更接近1，合理地体现了工资与学历以及工作经验的正相关性。

	pearson	kendall	spearman
公司规模	0.053907760934969 07	0.0843522834970424	0.1134531656507696
学历要求	0.27407614769771466	0.3761607858781567	0.46122437905981 14
工作经验	0.39159885513065684	0.41049990898376404	0.523751939764013
薪资_平均值	1.0	1.0	1.0

5. 模型预测

模型预测在数据预处理的基础上，利用处理好的all_final.csv中的数据，主要针对职位酬薪与职位类型两项进行预测。

5.1 职位酬薪预测模型

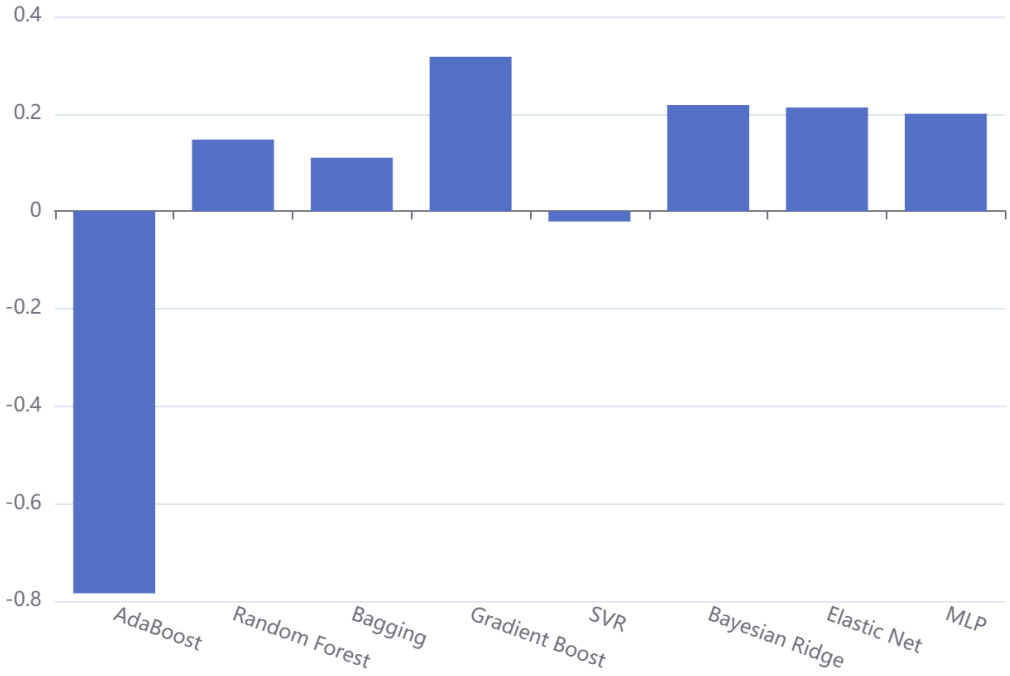
本次项目的主要任务为对进行统一处理后的职位薪酬的预测，根据预处理结果我们选取了九种数据，包括公司种类、薪资、学历要求、工作经验、公司规模、岗位地区、岗位行业、公司福利、职位等。

在该任务中，我们采取了几种回归算法，包括 AdaBoost, Random Forest, Bagging, Gradient Boost, SVR, Bayesian Ridge, Elastic Net 和 MLP回归。通过实现分别测试了它们的回归准确率和运行时间。

以下是AdaBoost, Random Forest, Bagging, Gradient Boost, SVR, Bayesian Ridge, Elastic Net 和 MLP回归的处理运行结果。

	AdaBoost	**Random Forest**	**Bagging**	**Gradient Boost**	**SVR**	**Bayesian Ridge**	**Elastic Net**	**MLP**
R2	-0.7841	0.1471	0.1099	**0.3171**	-0.021	0.2181	0.2130	0.2002

可在图表中更清晰展示：



重要函数代码包括两个预测模型共同调用的载入数据函数provider，薪资预测的载入函数salary_provider和薪资预测训练函数train，具体见代码附录。

多种回归算法中，Gradient Boost 回归算法效果最好，R2 达到 0.3171，而 AdaBoost 效果最差，R2为 -0.7841。

5.2 职位类型预测模型

本次项目的主要任务为对进行统一处理后的职位类型的预测，根据预处理结果我们选取了九种数据，包括公司种类、薪资、学历要求、工作经验、公司规模、岗位地区、岗位行业、公司福利、职位等。

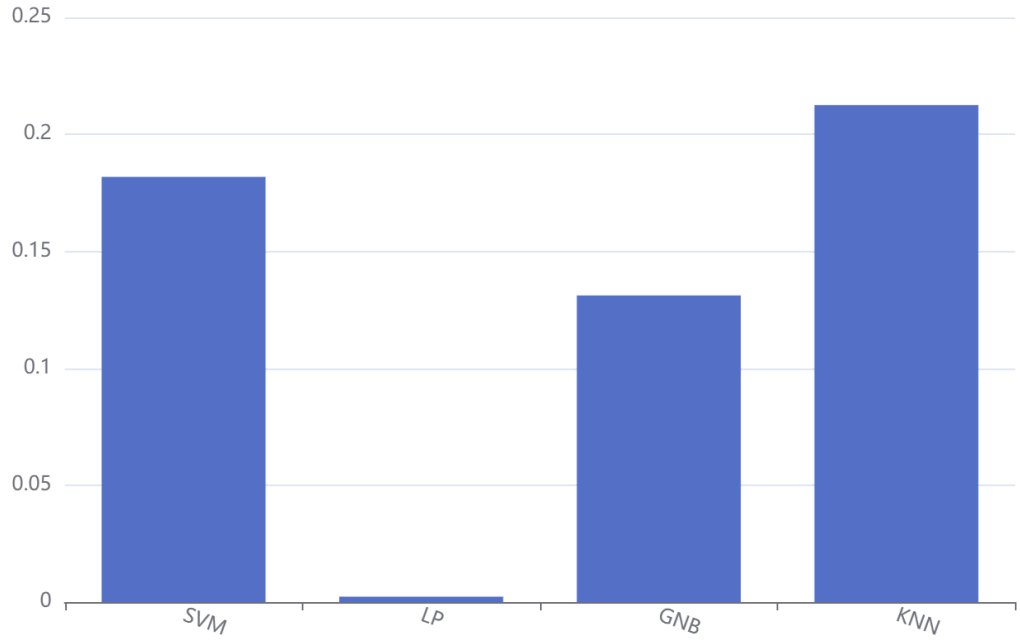
在该任务中，同样地，我们使用了几种最具代表性的分类算法，包括支持向量机 SVM，感知器，高斯贝叶斯，K 近邻、随机森林、XGBoost和神级网络多层感知器。

重要函数代码包括两个预测模型共同调用的载入数据函数provider，类型预测的载入函数job_provider和类型预测训练函数train，具体见代码附录。

其中，SVM，LP，GNB，KNN 四种算法的结果如下：

	SVM	**LP**	**GNB**	**KNN**
Accuracy	0.1818	0.0023	0.1311	**0.2125**

可在图表中更清晰展示：



多种算法中，KNN 算法效果最好，Accuracy 达到 0.3171，而 LP 效果最差，Accuracy 仅为 0.0023。

6. 个人总结

- **沈琪**

  本项目中我负责数据的预处理。经过上一次聚类作业，我已经对数据的预处理有了一些了解。本次大作业的数据预处理与上次作业又有很大的不同。前一次主要是离群点判断剔除、相关性分析等，这次主要是对于一些中文字符串的复杂处理，包括数值转化、关键词提取等。使用的主要工具是 python 的 pandas、re 库，用于对 csv 文件进行操作。

  在整个数据处理过程中，数据的不规整性异常严重，预处理工作量非常之大。譬如年薪这个数据，理论上应当是一个 int 值，但实际前程无忧网上提供了许多种表示方式，如元/日、千/月、万/年等，这就需要对不同的格式进行不同的处理，需要考虑的因素非常多。而且，还会有很多 nan 等异常数据需要处理，要考虑的地方相当多。此外，还有如职位描述与职位种类等数据，很多都是前程无忧网的用户自定义的字符串，完全无法通过直接的分类或独热等方法进行预处理，因此只能用 split, re 等字符串处理手段提取特征后再进行预处理。

  虽然本项目中遇到了很多苦难，但好在一一解决，收获颇丰！

  最后，感谢范磊老师的指导！

- **胡定伟**

  本项目中我负责数据的模型预测。大作业期间研究了很多开源代码的预测模型方法，针对csv数据处理后的或独热或分类的数据，进行预测模型搭建，调用python所带的预测模型库直接进行调用与使用，使用的主要工具是 python 的 sklearn、xgboost 库中所带的模型分析库，用于构建预测模型对相应数据属性进行预测分析。

  选取分类算法时查阅了很多相关资料，主要是了解如何调用python库的现有分类算法以及各个算法之间的差异、优势，实际调用不需要自己搭建。

  遇到了一些困难，但好在有同组同学的帮助和解惑，最后完成了该部分的任务，学到了很多，对数据挖掘尤其是对模型预测有了更深的了解和实践基础。

  同样，十分感谢范磊老师的指导！

- **许阳帆**

本次大作业我负责的是数据的特征分析，主要将数据进行了可视化处理以及相关性分析，总体来说难度不大，而且可以从中发现很多与我们息息相关的岗位数据。

在数据的处理、分析过程中，虽然画出相应的图并不困难，但是要选择合适的表现形式以及处理其中的细节却不容易。比如对于岗位的分布，既可以采用饼状图也可以采用柱状图，同时需要注意各项属性名过长的重叠等显示问题。

整体来说我负责的内容并不是特别困难，在编写代码和报告的时候，大家也一起讨论整个项目的思路以及一些细节的实现方法，让我在团队协作中锻炼了自己的数据分析和代码能力，感谢我们组其他三位成员以及范老师！

- **唐铭江**

本次大作页我负责的是数据的爬取，主要是从前程无忧网站上爬取数据。因为之前有爬虫的小作业，所以有了一定的基础，实现起来比上一次轻松。但是过程中还是遇到了不少问题。

首要的是爬取的内容选择。因为如果直接在主页面进行爬取，尽管也可以爬取到大量内容，但是对数据的分类预处理将会十分麻烦。所以在选择爬取内容的时候，先通过前程无忧网站自己提供的分类功能筛选，最终从中选定十几个大类，再分别对各个大类进行爬取。

另一个问题是网站的反爬功能。当爬取了一定数据后就会无法再爬取内容，为了解决这个问题尝试了模拟用户登陆的方式，但在实践操作中发现尽管登录了还是会被反掉。所以最后只保留了基础的设置时间间隔，设置代理头的方式，并没有模拟登录，当被反掉时重新手动操作，好在出现问题的次数不多，工作量并不大。

这次能够顺利完成项目也要感谢小组成员对我的帮助和范磊老师的指导。

7. 代码附录

前程无忧爬虫

```
# -*- coding:utf-8 -*-
import urllib.request
import xlwt
import re
import urllib.parse
import time
header={
    'Host':'search.51job.com',
    'Upgrade-Insecure-Requests':'1',
    'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36'
}
def getfront(page,item):       #page是页数，item是输入的字符串
     result = urllib.parse.quote(item)                    #先把字符串转成十六进制编码
     ur1 = result+',2,'+ str(page)+'.html'
     ur2 = 'https://search.51job.com/list/020000,000000,6200%252c3700,00,9,99,'
     res = ur2+ur1    #拼接网址
     a = urllib.request.urlopen(res)
     html = a.read().decode('gbk')      # 读取源代码并转为unicode
     html = html.replace('\\', '')       # 将用于转义的"\"替换为空
     html = html.replace('[', '')
     html = html.replace(']', '')
     return html

def getInformation(html):
    reg = re.compile(r'\{"type":"engine_search_result","jt":"0".*?"job_href":"
(.*?)","job_name":"(.*?)".*?"company_href":"(.*?)","company_name":"
(.*?)","providesalary_text":"(.*?)".*?"updatedate":"(.*?)".*?,'
```

```
                    r'"companytype_text":"(.*?)".*?"jobwelf":"
(.*?)".*?"attribute_text":"(.*?)","(.*?)","(.*?)","(.*?)","companysize_text":"
(.*?)","companyind_text":"(.*?)","adid":""},',re.S)#匹配换行符
    items=re.findall(reg,html)
    print(items)
    return items

#新建表格空间
excel1 = xlwt.Workbook()
# 设置单元格格式
sheet1 = excel1.add_sheet('Job', cell_overwrite_ok=True)
sheet1.write(0, 0, '序号')
sheet1.write(0, 1, '职位')
sheet1.write(0, 2, '公司名称')
sheet1.write(0, 3, '公司地点')
sheet1.write(0, 4, '公司性质')
sheet1.write(0, 5, '薪资')
sheet1.write(0, 6, '学历要求')
sheet1.write(0, 7, '工作经验')
sheet1.write(0, 8, '公司规模')
sheet1.write(0, 9, '公司福利')
sheet1.write(0, 10, '发布时间')
sheet1.write(0, 11, '公司信息链接')
sheet1.write(0, 12, '分类')
number = 1
item = input()

for j in range(1, 100):   #页数自己随便改
    try:
        print("正在爬取第"+str(j)+"页数据...")
        html = getfront(j, item)      #调用获取网页原码
        for i in getInformation(html):
            try:
                sheet1.write(number, 0, number)
                sheet1.write(number, 1, i[1])
                sheet1.write(number, 2, i[3])
                sheet1.write(number, 3, i[8])
                sheet1.write(number, 4, i[6])
                sheet1.write(number, 5, i[4])
                sheet1.write(number, 6, i[10])
                sheet1.write(number, 7, i[9])
                sheet1.write(number, 8, i[12])
                sheet1.write(number, 9, i[7])
                sheet1.write(number, 10, i[5])
                sheet1.write(number, 11, i[2])
                sheet1.write(number, 12, '技工')
                number+=1
            except:
                pass
        excel1.save("技工.xls")
        time.sleep(3)  # 休息间隔，避免爬取海量数据时被误判为攻击，IP遭到封禁
    except:
        pass
```

公司种类预处理

```
df = pd.read_csv("./all_salary.csv")
company_xinzhi = list(df["公司类别"].unique())
print(company_xinzhi)
onehot_leibie = pd.get_dummies(df["公司类别"], prefix ='公司类别')
print(onehot_leibie.head(5))
df = pd.concat([df,onehot_leibie],axis=1)
df = df.drop("公司类别", axis=1)
df.to_csv('./all_leibie.csv', encoding='utf_8_sig')
print(df.head(5))
```

薪资预处理

```
df = pd.read_csv("./all_leibie.csv")
minimum = []
maximum = []
average = []
for i in range(len(df['薪资'])):
    tmp = df['薪资'][i]
    if re.compile('年').search(tmp):
        yearTag = 1 # year salary
    elif re.compile('天').search(tmp):
        yearTag = 300 # day salary
    else:
        yearTag = 12 # month salary

    if re.compile('千').search(tmp):
        multiTag = 1000 # count as 1000
    elif re.compile('元').search(tmp):
        multiTag = 1 # count as 1
    else:
        multiTag = 10000 # count as 10000

    numberIndex = max(tmp.find("千"), tmp.find("元"), tmp.find("万"))
    numberString = tmp[0:numberIndex]
    numberList = numberString.split("-")
    if len(numberList) == 1:
        minimum.append(int(float(numberList[0]) * multiTag * yearTag))
        maximum.append(int(float(numberList[0]) * multiTag * yearTag))
        average.append(int(float(numberList[0]) * multiTag * yearTag))
    else:
        minimum.append(int(float(numberList[0]) * multiTag * yearTag))
        maximum.append(int(float(numberList[1]) * multiTag * yearTag))
        average.append(int((float(numberList[0])+float(numberList[1]))/2 *
multiTag * yearTag))
# append the maxinum and minimum salary in newDF
df.insert(loc=len(df.columns), column="薪资_最大值", value=maximum)
df.insert(loc=len(df.columns), column="薪资_最小值", value=minimum)
df.insert(loc=len(df.columns), column="薪资_平均值", value=average)
df = df.drop("薪资", axis=1)
print(df.head(5))
df.to_csv('./all_salary.csv', encoding='utf_8_sig')
```

学历要求预处理

```
df = pd.read_csv("./all_salary.csv")
print(list(df["学历要求"].unique()))
newframe = []
for i in range(len(df['学历要求'])):
    tmp = df['学历要求'][i]
    if tmp == '初中及以下':
        newframe.append(1)
    elif tmp == '中技':
        newframe.append(2)
    elif tmp == '中专':
        newframe.append(3)
    elif tmp == '高中':
        newframe.append(4)
    elif tmp == '大专':
        newframe.append(5)
    elif tmp == '本科':
        newframe.append(6)
    elif tmp == '硕士':
        newframe.append(7)
    elif tmp == '博士':
        newframe.append(8)
    else:
        newframe.append(3)
df.drop("学历要求",axis=1)
df.insert(loc=len(df.columns), column="学历", value=newframe)
print(list(df["学历"].unique()))
df.to_csv('./all_edu.csv', encoding='utf_8_sig')
```

工作经验预处理

```
df = pd.read_csv("./all_edu.csv")
newframe = []
print(list(df["工作经验"].unique()))
for i in range(len(df['工作经验'])):
    tmp = df['工作经验'][i]
    if re.compile("3-4年经验").search(tmp):
        newframe.append(3.5)
    elif re.compile("5-7年经验").search(tmp):
        newframe.append(6)
    elif re.compile("8-9年经验").search(tmp):
        newframe.append(8.5)
    elif re.compile("10年以上经验").search(tmp):
        newframe.append(10)
    elif re.compile("1年经验").search(tmp):
        newframe.append(1)
    elif re.compile("2年经验").search(tmp):
        newframe.append(2)
    elif re.compile('在校生/应届生').search(tmp):
        newframe.append(0)
    elif re.compile('无需经验').search(tmp):
        newframe.append(0)
    else:
        newframe.append(0)
```

```
df = df.drop("工作经验",axis=1)
df.insert(loc=len(df.columns), column="工作经验", value=newframe)
print(list(df["工作经验"].unique()))
df.to_csv('./all_experience.csv', encoding='utf_8_sig')
```

公司规模预处理

```
df = pd.read_csv("./all_experience.csv")
newframe = []
print(list(df["公司规模"].unique()))
for i in range(len(df['公司规模'])):
    tmp = df['公司规模'][i]
    if re.compile("少于50").search(tmp):
        newframe.append(25)
    elif re.compile("50-150").search(tmp):
        newframe.append(100)
    elif re.compile("150-500").search(tmp):
        newframe.append(325)
    elif re.compile("500-1000").search(tmp):
        newframe.append(750)
    elif re.compile("1000-5000").search(tmp):
        newframe.append(3000)
    elif re.compile("5000-10000").search(tmp):
        newframe.append(7500)
    elif re.compile('10000人以上').search(tmp):
        newframe.append(10000)
    else:
        newframe.append(25)
df = df.drop("公司规模",axis=1)
df.insert(loc=len(df.columns), column="公司规模", value=newframe)
print(list(df["公司规模"].unique()))
df.to_csv('./all_guimo.csv', encoding='utf_8_sig')
```

岗位地区预处理

```
df = pd.read_csv("./all_guimo.csv")
newframe = []
print(list(df["岗位地区"].unique()))

onehot_leibie = pd.get_dummies(df["岗位地区"])
print(onehot_leibie.head(5))
df = pd.concat([df,onehot_leibie],axis=1)
df = df.drop("岗位地区", axis=1)

df.to_csv('./all_diqu.csv', encoding='utf_8_sig')
```

岗位行业预处理

```
df = pd.read_csv("./all_diqu.csv")
print(list(df["分类"].unique()))


onehot_leibie = pd.get_dummies(df["分类"])
print(onehot_leibie.head(5))
df = pd.concat([df,onehot_leibie],axis=1)
df = df.drop("分类", axis=1)

df.to_csv('./all_fenlei.csv', encoding='utf_8_sig')
```

公司福利预处理

```
df = pd.read_csv("./all_fenlei.csv")
s = {}
t = {}
for i in range(len(df['公司福利'])):
    tmp = df["公司福利"][i]
    words = str(tmp).split(' ')
    for j in words:
        if j in s.keys():
            s[j]=s[j]+1
        else:
            s[j]=1
for i in s.keys():
    if s[i]>2000:
        t[i] = s[i]
print(t)
del t["nan"]
print(t)
df["公司福利"] = df["公司福利"].fillna(value=0)
for i in t.keys():
    newframe = []
    for j in range(len(df['公司福利'])):
        tmp = df["公司福利"][j]
        if re.compile(i).search(str(tmp)):
            newframe.append(1)
        else:
            newframe.append(0)
    df.insert(loc=len(df.columns), column=i, value=newframe)
print(df.info())
df.to_csv('./all_fuli.csv', encoding='utf_8_sig')
```

职位预处理

```
def get_all_substrings(input_string):
  length = len(input_string)
  return [input_string[i:j + 1] for i in range(length) for j in range(i,length)]


df = pd.read_csv("./all_fuli.csv")
```

```
s = {}
t = {}
for i in range(len(df['职位'])):
    tmp = df["职位"][i]
    words = get_all_substrings(str(tmp))
    for j in words:
        if j in s.keys():
            s[j]=s[j]+1
        else:
            s[j]=1
for i in s.keys():
    if s[i]>2000:
        t[i] = s[i]
print(t)
for i in t.keys():
    newframe = []
    for j in range(len(df['职位'])):
        tmp = df["职位"][j]
        if re.compile(i).search(str(tmp)):
            newframe.append(1)
        else:
            newframe.append(0)
    df.insert(loc=len(df.columns), column=i, value=newframe)
print(df.info())
df.to_csv('./all_final.csv', encoding='utf_8_sig')
```

职位薪酬预测模型

```
def provider(filepath="../data/all_final.csv",
             is_regression=False,
             salary_pred="high",
             all_data=False):
    # read data from the csv file
    df = pd.read_csv(filepath, header=0, encoding="utf-8")
    # print df.head(5)

    # preprocess the data [city, quality, company, job, welfare]
    df_city = df.apply(lambda s: np.argmax(list(s[18:34])), axis=1)
    df_quality = df.apply(lambda s: np.argmax(list(s[1: 11])), axis=1)
    df_com = df.apply(lambda s: np.argmax(list(s[35: 50])), axis=1)
    df_job = df.apply(lambda s: np.argmax(list(s[75: 90])), axis=1)
    df_welfare = df.apply(lambda s: np.sum(list(s[51: 74])), axis=1)

    # merge the properties
    if is_regression:
        data = df.iloc[:, 15:18]
        if salary_pred == "high":
            labels = df.iloc[:, 12]
        elif salary_pred == "low":
            labels = df.iloc[:, 13]
        else:
            labels = df.iloc[:, 12:14]
    else:
        data = df.iloc[:, 12:18]
    # print data.head(5)
    # print labels.head(5)
```

```
    data['city'] = df_city.values
    data['quality'] = df_quality.values
    data['company'] = df_com.values
    if is_regression:
        data['job'] = df_job.values
    else:
        labels = df_job.values
    data['welfare'] = df_welfare.values
    print(data.head(2))

    # split the data
    X_train, X_test, Y_train, Y_test = train_test_split(data.values,
labels.values, test_size=0.2, random_state=42)
    print(X_train.shape, X_test.shape)
    print(Y_train.shape, Y_test.shape)
    if all_data:
        return data.values, labels.values
    return X_train, X_test, Y_train, Y_test
```

```
def salary_provider(preprocessing="None"):
    X_train, X_test, Y_train, Y_test = provider(is_regression=True)

    # Normalize the label [No sense!!]
    # salary_max, salary_min = np.max(Y_train), np.min(Y_train)
    # Y_train = (Y_train - salary_min) / float(salary_max - salary_min)
    # Y_test = (Y_test - salary_min) / float(salary_max - salary_min)
    if preprocessing == "normalize":
        normalizer = Normalizer()
        X_train = normalizer.fit_transform(X_train)
        X_test = normalizer.fit_transform(X_test)
    elif preprocessing == "minmax":
        minmaxscaler = MinMaxScaler()
        X_train = minmaxscaler.fit_transform(X_train)
        X_test = minmaxscaler.fit_transform(X_test)
    elif preprocessing == "standard":
        standardscale = StandardScaler()
        X_train = standardscale.fit_transform(X_train)
        X_test = standardscale.fit_transform(X_test)
    else:
        pass

    print(Y_test)
    print(Y_train)
    return X_train, X_test, Y_train, Y_test
```

```
def train():
    X_train, X_test, Y_train, Y_test = salary_provider()
    ada = AdaBoostRegressor()
    rf = RandomForestRegressor()
    bagging = BaggingRegressor()
    grad = GradientBoostingRegressor()
    svr = SVR()
    bayes_ridge = BayesianRidge()
    elastic_net = ElasticNet()
    mlp = MLPRegressor(hidden_layer_sizes=(64, 128, 64), max_iter=1000)
```

```
    regressors = [ada, rf, bagging, grad, svr, bayes_ridge, elastic_net, mlp]
    regressor_names = ["AdaBoost", "Random Forest", "Bagging",
                       "Gradient Boost", "SVR", "Bayesian Ridge",
                       "Elastic Net", "MLPRegressor"]

    # regressors = [mlp]
    # regressor_names = ["MLP"]

    for regressor, regressor_name in zip(regressors, regressor_names):
        intime = time.time()
        regressor.fit(X_train, Y_train)
        Y_pred = regressor.predict(X_test)
        print(X_train.shape, Y_train.shape)
        print("-------------------------------------")
        print(time.time() - intime)
        print("For Regressor : ", regressor_name)
        print("Mean Absolute Error : ", metrics.mean_absolute_error(Y_test,
Y_pred))
        # print "Median Absolute Error : ",metrics.median_absolute_error(Y_test,
Y_pred)
        # print "Mean Squared Error : ",metrics.mean_squared_error(Y_test,
Y_pred)
        print("R2 Score : ", metrics.r2_score(Y_test, Y_pred))
        print("---------------------------------\n")

        for i in range(5):
            print(Y_pred[i], Y_test[i], X_test[i])
```

职位类型预测模型

```
def provider(filepath="../data/all_final.csv",
             is_regression=False,
             salary_pred="high",
             all_data=False):
    # read data from the csv file
    df = pd.read_csv(filepath, header=0, encoding="utf-8")
    # print df.head(5)

    # preprocess the data [city, quality, company, job, welfare]
    df_city = df.apply(lambda s: np.argmax(list(s[18:34])), axis=1)
    df_quality = df.apply(lambda s: np.argmax(list(s[1: 11])), axis=1)
    df_com = df.apply(lambda s: np.argmax(list(s[35: 50])), axis=1)
    df_job = df.apply(lambda s: np.argmax(list(s[75: 90])), axis=1)
    df_welfare = df.apply(lambda s: np.sum(list(s[51: 74])), axis=1)

    # merge the properties
    if is_regression:
        data = df.iloc[:, 15:18]
        if salary_pred == "high":
            labels = df.iloc[:, 12]
        elif salary_pred == "low":
            labels = df.iloc[:, 13]
        else:
            labels = df.iloc[:, 12:14]
    else:
```

```
        data = df.iloc[:, 12:18]
    # print data.head(5)
    # print labels.head(5)

    data['city'] = df_city.values
    data['quality'] = df_quality.values
    data['company'] = df_com.values
    if is_regression:
        data['job'] = df_job.values
    else:
        labels = df_job.values
    data['welfare'] = df_welfare.values
    print(data.head(2))

    # split the data
    X_train, X_test, Y_train, Y_test = train_test_split(data.values,
labels.values, test_size=0.2, random_state=42)
    print(X_train.shape, X_test.shape)
    print(Y_train.shape, Y_test.shape)
    if all_data:
        return data.values, labels.values
    return X_train, X_test, Y_train, Y_test
```

```
def job_provider(preprocessing="None"):
    X_train, X_test, Y_train, Y_test = provider(is_regression=True)
    print(X_train.shape, Y_train.shape)

    if preprocessing == "normalize":
        normalizer = Normalizer()
        X_train = normalizer.fit_transform(X_train)
        X_test = normalizer.fit_transform(X_test)
    elif preprocessing == "minmax":
        minmaxscaler = MinMaxScaler()
        X_train = minmaxscaler.fit_transform(X_train)
        X_test = minmaxscaler.fit_transform(X_test)
    elif preprocessing == "standard":
        standardscale = StandardScaler()
        X_train = standardscale.fit_transform(X_train)
        X_test = standardscale.fit_transform(X_test)
    else:
        pass

    return X_train, X_test, Y_train, Y_test
```

```
def train():
    X_train, X_test, Y_train, Y_test = job_provider()
    svm = SVC()
    perceptron = Perceptron()
    gnb = GaussianNB()
    knn = KNeighborsClassifier()
    rf = RandomForestClassifier()
    xg = XGBClassifier()
    mlp = MLPClassifier()

    classifiers = [svm, perceptron, gnb, knn, rf, xg, mlp]
    classifier_names = ["SVM", "LP", "GNB", "KNN", "RF", "XGB", "MLP"]
```

```
    for classifier, classifier_name in zip(classifiers, classifier_names):
        print("-------------------------------------")
        print(classifier)
        intime = time.time()
        classifier.fit(X_train, Y_train)
        Y_pred = classifier.predict(X_test)
        print(time.time() - intime)
        print("Accuracy for ", classifier_name, " : ",
metrics.accuracy_score(Y_test, Y_pred))
        print("-------------------------------------\n")
```