在数据处理过程中主要分为三个阶段它们分别是数据获取、数据清理和数据整合。数据收集就是从招聘网站上运用Python scrapy框架抓取数据，并将获取到的数据存储到计算机本地的磁盘中。数据清理就是对获取到的数据进行缺省值处理以及异常值的处理。数据整合就是统一数据的格式便于接下来的数据分析。这三个步骤是环环相扣的，具体的操作步骤如UML图所示

Data Collection

Data Cleaning

Data Integration

Data Analysis

这里首先进行数据收集，然后对搜集到的数据进行清洗，接下来进行数据整合。这里可以看到数据整合与数据清理之间以及数据整合与数据分析之间都是双向箭头。这么设计的目的是在进行数据分析或者数据整合的时候发现数据并没有预想之中的结果那么就需要回过头来重新处理。

在本小节中我使用scrapy框架来进行数据搜集。所要搜集的数据来自两个招聘网站一个是中国的51job另一个是英国的jobtotal。虽然是两个不一样的网站但是搜集的方法是一样的。不一样的地是由于网站结构不同因此所使用的xpath是不一样的。此外，URL也是不一样的。因此在搜集的过程中仅仅需要更改这两个参数即可。

2.2 数据清洗

通过之前的运用scrapy框架对网上的招聘数据进行挖掘工作，获得了大量的招聘数据。在这一章节将对先前采集到的数据进行清洗，纠正采集到的错误数据。在清洗的过程中，将会使用Openrefine以及Python中的numpy和panda库来进行操作。

2.2.1 51job data clean

2.2.1

通过之前的信息挖掘总共获取到了10253条Java相关的职业信息以及1492条Python相关的岗位信息。首先，我将存储在本地数据库内的数据以csv格式的形式导出到电脑桌面。然后使用Openrefine工具读取该数据集。通过目测就不难发现，数据集出现的最大问题就在于教育背景与工作经验这两个数据记录存在颠倒行为。发生这一情况的一大原因是因为网站所显示的数据并没有一个严格统一的结构显示而在数据信息抓取的过程中也是使用通用的办法来获取数据。该两列脏数据的种类可以分为三类它们分别是数据缺失、数据记录相反以及完全不正确的数据内容。对于教育背景与工作经验存储位置相反的数据元素来说，我做的工作仅仅是将它们进行左右互换。

2.2.1.2 数据统一

在openrefine中使用text facet查看salary\_style和salary\_method，可以发现工资有三种方式来显示，它们分别是按天支付、按月支付以及按年支付。工资的结算单位有两种不同的方式分别是千和万。为了使得数据的统一以便于后续的分析，在这里数据将会统一为按年支付并且单位是’万’。具体操作如下，在Excel中使用筛选来查看salary\_style和salary\_method，此时忽略缺省值。对于工资支付方式为按月支付并且结算形式是‘万’的数据，我对这些数据同时乘以12再乘10000以达到按年支付这一结果。对于工资支付方式为按月支付并且结算形式是‘千’的数据，我对这些数据同时乘以12再乘以1000以达到按年支付这一结果。对于工资支付方式为按天支付的数据，我对这些数据同时乘以12再乘以22以达到按年支付这一结果。这里取22的原因是因为将每个月的工作天数按照22天来计算。对于按年支付的工资由于结算的单位都是‘万’因此仅仅将它们乘以10000来处理。除此之外，在openrefine中使用text facet来查看company\_scale和job\_industry，在这过程中可以发现它们分别有228和319个不同的分布。然而其中的数据有很多都是由于分类过于细致而导致的冗余现象，因此将这些在大类中细分的数据均统一为大类。最后在openrefine中用text facet来查看location的数据也发现了有许多城市划分得过于细致，例如有的工作地点显示为‘上海市-宝山区’，对此情况在openrefine中将它们修改为‘上海’。其他具有相似情况的城市名称都按照这一方式来处理。在使用openrefine处理location,company\_scale和job\_industry的时候同时对所有的数据集进行翻译，将它们从原先的中文翻译为英文。这一操作的原因是在使用weka软件进行数据分析的时候对于中文字符它并不能进行很好的编码解析。因而它会导致我的数据集在导入到weka中后所有的中文字符都是乱码的情况发生。

2.2.1.2 缺省值处理

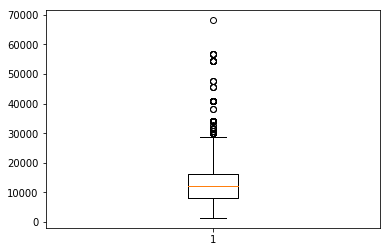
通过python中的panda库来查看所有的数据情况，可以发现无论是java的招聘数据还是python的招聘数据，工资部分都存在有缺省值的情况。其中java部分有124条数据缺失，python部分有32条数据缺失。除此之外在岗位信息发布时间以及职业详细描述也存在有缺省值的情况，但是对于这两部分暂时不做任何的处理。第一个原因是岗位信息的发布时间对于本次分析并无意义，其二，岗位描述具有独特性与唯一性因此也不做填充处理。接下来将使用openrefine工具也对缺失的数据进行填充。

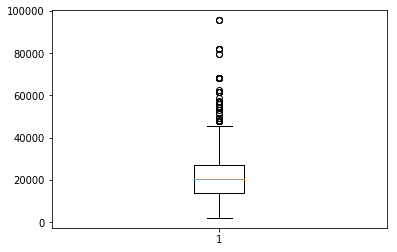


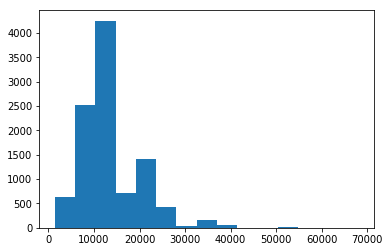


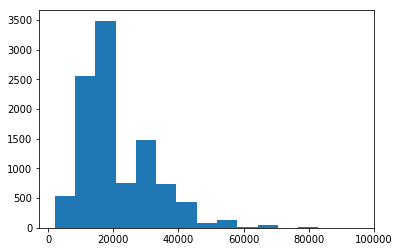
通过查看可以发现每一个地区都有缺少‘salary’的数据。对于本身数据量大的城市，对于缺省值的填充采用的方法是首先在openrefine中使用text facet来查看该城市的salary分布情况。例如，同时使用text facet来查看shanghai和salary\_max，可以看出总共有127条缺省的数据，由于本身上海的数据量比较大因此选择salary\_max中的众数1.5来填充缺省的数据，并使用同样的方式同时填充salary\_min的数据。对于缺省值大于10的城市，均选用上述的方式来填充缺省值。而对于缺省值小于10的城市，填充的方式则不一样了。由于这时该城市的总体样本数量就比较小而样本数据分较散，因此在这种情况下使用众数填充的意义就没有那么明显。除此之外，这一操作可能还会对最终的结果产生错误的指引。这时选择在Excel中来对数据进行操作。首先，在Excel中计算出所有但不包含缺省值的平均工资，然后使用这一平均值对剩余的城市的salary\_min与salary\_max进行填充。

2.2.1.2异常值的处理









通过在python中使用箱型图可以发现有一部分数值都被认为是异常值，然而幸运的是所有格的数值都没有出现小于0的情况发生。通过调用直方图我们更加直观地看到了工资的具体分布情况，因为是薪资分布所以在这里应该允许认为一个大于零的数值出现。因此，在这一部分不做其他的改动。

2.2.2 UK recruitment data clean

通过前期网络爬虫获取到2036条与java有关的岗位信息以及404条与python有关的岗位信息。由于该网站的数据结构化比较简单，仅仅在地址这一属性中有不一样的地方。因此当时在进行数据获取的时候，使用了两种不同的xpath来获取位置数据。

2.2.2.1 数据的统一

通过使用openrefine中的text facet来查看薪资支付类型，可以发现与中国的招聘数据类似都有三种支付方式，它们分别是按日支付、按月支付和按年支付。为了是数据形式进行统一以便于后续的分析工作，在此处我将所有的薪资都按照按年支付。其次，为了视觉上更好的观察数据，对于数据中所有还有k的薪资都将k替换为了’000’。对于按月支付的薪资，我在这里将它们都同时乘11来转换为年薪。这里乘以11而不是乘以12主要原因是我发现按月支付乘以12后整体薪资水平都比较高，从而这一现象会导致整体年薪被提高。对于按日支付的薪资，我在此先将它们乘以20表明一个月工作20天，然后再乘以11理由和之前的一致。在此我发现按日结算的工资换成年薪之后普遍提升了很大。这里我分析其中的主要一个原因是因为有部分项目仅仅是企业需求的短期劳务工。通过这种方式来填补该公司在部分业务上的短期需求而进行的一个招聘行为。此外，对于公司位置的统一。首先在Excel中job\_location与company\_location均合并为location。因为在数据获取的过程中网站对于位置的存储有且仅有这两种形式，因此在合并的过程中不会存在有数据被覆盖的情况。此外，我还发现位置信息大都以‘城市，城市所在的郡’来呈现。为了使数据不至于划分得过于细致在这个时候在Excel中将数据根据字符‘，’进行分离，从而达到仅保留郡名的目的。

2.2.2.2 缺省值处理

通过调用python来查看每一个属性的数据情况，可以看到在薪资、薪资类型以及岗位描述中存在缺省值。岗位描述由于前期的网站信息获取过程中就会因为信息结构化的不同意或者该公司原本就没有提供岗位信息而造成的缺损。因此，对于岗位描述以及薪资类型部分在这里暂时不做任何的调整。对于薪资这一属性的填充方式主要与之前的填充方法相类似。首先在openrefine中生成一个project来查看数据。然后使用text facet来查看整体数据集，选择数据空白的部分，再使用text facet来查看location属性。在这里我的策略与中国招聘网站的数据填充方式大致相同。对于数据量比较大的城市例如London，我使用最高薪资与最低薪资它们各自属性中数据的众数来对数据进行填充。而对于数据量较小的城市我在Excel中计算出整体不包括缺省值的数据的平均值。最后，使用这一平均值来对数据进行一个填充。同时我也发现在原始的工资属性中，很多企业是以薪资加福利的形式来呈现这一属性的数据的。因此在Excel中我将所有其中的数字与字母分开从而在一个新的属性中详细列举了公司福利。至此对于英国招聘网站的薪资缺省值也就填充完成了。

2.2.2.3 异常值处理

同样，通过在python绘制箱型图与直方图来观测是否有异常值的存在。主要的判断依据是所有薪资必须要大于零，因此所有非正整数的数据都是不可取的。幸运的是在本样本中没有数据是小于等于零的。对于过于大的数值还是可以接受的，毕竟有些行业所给出的薪资很高也并非在现实中是不存在的。

2.2.3 数据整合

为了满足不同的数据挖掘算法对于数据集的要求，本章节将整合已经完成清理工作的数据集。首先对于薪资部分，为了便于划分薪资等级我在Excel中对已获得的薪资部分进行进一步的处理。首先将薪资的最大值和最小值相加在除以二来获取每一个招聘信息的平均薪资。并根据平均薪资来划分薪资等级。划分薪资等级的目的就是为了在下一章节的分析过程中能够找到一个更好的规则来查看哪一类人可以获得较高的薪资而哪一类人的薪水会更低。对于两大数据集的薪资来说我都按照不同的平均值将它们划分为了‘高中低’。这一部分的操作都在openrefine中来完成。对于中国部分的招聘数据，将年收入低于10000英镑的岗位标记为‘low’，将年收入介于10000英镑至26000英镑之间的岗位标记为‘medium’，将年收入高于26000英镑的岗位标记为‘high’。与此同时对于英国的招聘数据，将年收入低于20000英镑的岗位同样标记为了‘low’，将年收入介于20000英镑至40000英镑之间的岗位标记为‘medium’，将年收入高于40000英镑的岗位标记为‘high’。至此，薪资等级的划分就完成了。为了进一步的缩减城市数量，对于中国的数据按照城市规模划分为一线城市、二线城市以及三线城市。而对于英国的数据则根据地理位置来进行划分。