数据科学基础研究报告

1. 小组成员信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 姓名 | 学号 | 邮箱 | Python题目完成数量 | 职责 |
| 刘学卓 | 18125088 | [1246399584@qq.com](mailto:1246399584@qq.com) | 200 | 组长 |
|  |  |  |  |  |
| 陈子合 | 181250019 | 843731853@qq.com |  |  |

1. 研究问题

1. 问题简述。

对题目难度进行分类，将题目分成为容易，较易，中等，较难，困难五个等级。

2. 应用场景。

对于学生而言，同学们在慕码平台进行Python编程练习时，常常会有希望从易到难刷题的需求，让同学能在自己喜欢的刷题节奏中进行编程练习，提高编程练习效率。并且，对于不同能力水平的同学来说，有着不一样的编程练习的需求。对于能力较差的同学，希望先挑简单的题目来上手，待能力提升后，再去挑战更难的题目，对于能力较强的同学，刷简单的题目没有挑战性，对于自身的编程能力提升并没有太大的作用，他们往往喜欢挑较难的题目来进行编程练习，在研究复杂算法中提升自己。

对于老师而言，在布置作业时，老师可能希望根据各个学生的编程能力情况，给他们定制编程练习计划，如给低年级的同学布置难度较低的题目，给高年级的同学布置难度较高的题目。在设计机试题目时，老师可能希望慕码平台能自动推荐各种难度等级的题目给老师进行参考，而不需要老师每道题都看一遍，从而减轻工作量。

对于慕码平台而言，在慕码平台的习题集里，题目没有打上难度标签，而对于一个成熟的编程OJ平台来说，给题目打难度标签是十分普遍的，也是十分必要的。

综合上述场景，本研究通过获取助教给出的关于Python编程的源数据，以与题目难度相关的各种指标数据为基础，结合许多学者的研究成果，对Python编程习题进行难度评价，给出难度等级。本研究的结果对慕码平台的题目难度评价有一定的积极作用。

1. 代码开源地址

代码地址：https://github.com/lxzssfd/DataScience

代码主要分为三部分：第一部分是对课程所提供的json文件进行解析处理，得出我们所需要的数据生成新的json文件；第二部分对上一部分得出的数据进行标准化处理，利用熵权法得出各个数据对于最终结果的影响权重；第三部分对数据进行聚类处理，得出五种不同难度的分类。

1. 研究方法
2. 对于原始数据，求出每道题目平均最终得分，平均提交代码行数以及python代码占比。
3. 对于求出来三种数据，对数据进行标准化处理，以消除数据范围大小不同所带来的结果偏差。

由于每道题目的平均最终得分、平均提交代码行数以及Python代码占比的度量单位以及数量级的差别，存在不可共度性。所以，我们需要对三种数据进行数据标准化处理，以消除原始指标数据的差异带来的影响。

本次研究采用极差法的方法对三个指标（平均最终得分、平均提交代码行数、Python代码占比）进行无量纲化处理。三个指标中，平均最终得分以及Python代码占比是负向指标，即平均最终得分及Python代码占比越大，则该题的题目难度越低，题目难度度量值越小。而平均提交代码行数是正向指标，即平均提交代码行数越大，则该题目难度越大，题目难度度量值越大。

正向指标公式： (1)

负向指标公式： (2)

式(1)(2)中，i为第i个题目，j为第j个指标，为处理前的指标值，max()为第j项指标的最大值，min()为第j项指标的最小值,为处理后无量纲化的指标值。

1. 利用熵权法求出三种数据对于题目难度的影响权重。

（1）赋权法的选择。

我们在进行综合评价时，不能把各种指标同等对待，需要给指标赋予权重。确定权重有主观赋权法和客观赋权法。主观赋权法需要依靠研究人员自身的专业知识、经验，通过主观判断来确定各个指标的权重。考虑到我们对Python题目难度的研究不深，并且较难去联系相关领域的资深研究人员，以及主观赋权法难以避免的存在争议性，我们选择客观赋权法。而客观赋权法中有主成分分析赋权法以及熵权法等方法。基于我们目前的知识水平，我们较易理解的有主成分分析赋权法和熵权法。由于本次研究中的原始数据中，与题目难度相关的样本数据类较少，导致最终选择的评价指标数目较少，我们并不需要对数据进行降维处理，所以抛弃了主成分分析赋权法而使用熵权法。

（2）熵权法介绍。

熵权法的基本思路，是根据指标的离散程度大小来确定指标权重。

设有m个待评题目，n项评价指标，利用研究方法2中的极差法将数据标准化，形成标准数据矩阵，再通过该矩阵以及信息熵公式,求得各个指标的信息熵。对于某个指标，其信息熵越小，则指标的离散程度越大，该指标对综合评价的影响就越大，则其权重越大。反之，其信息熵越大，则指标的离散程度越小，该指标对综合评价的影响就越小，则其权重越小。

（3）算法实现步骤。

I.数据矩阵

其中为第i个题目第j个指标的标准化数值。

II.计算第j项指标下第i个题目占该指标的比重。

得到比重矩阵

III.计算第j项指标的熵值。

一般令，ln为自然对数，≥0。则0≤≤1.

IV.计算第j项指标的差异系数。

对于第j项指标，的差异越大，对题目难度评价的作用越大，熵值就越小。差异系数公式如下：

V.求权重。

通过上式即可求得n个评价指标的权重。

（4）加入主观因素，改良熵权法。

在专家讨论后，给出每个指标的主观权重，通过下列公式

计算出每个指标的主观权重占比，在(3)算法步骤中的数据矩阵中，令

(0≤≤1)

得到新的数据矩阵A，再进行接下来的计算。

通过上述步骤的改良，则能将经典的熵权法改良成带有主观因素的熵权法，令最终权重既有客观因素考量，又有主观因素考量，令计算权重的结果更有说服力。

1. 利用Kmeans聚类的方法讲题目根据难度大致分为五类。
2. Kmeans聚类算法的选择

Kmeans算法是无监督的聚类算法，实现起来比较简单，并且聚类效果

也不错，因此广泛应用。考虑到不同难度的题目占比不同，不可以用百分比一刀切的情况，结合所给数据的具体情况，我们选择用对平均行数、平均结果、python答案占比加权得到的结果进行聚类，这样得到的结果会大大减小用百分比划分题目难度导致的分类不完全的问题。

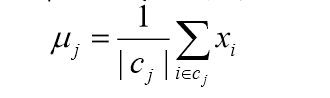
1. Kmeans算法大体步骤

I.选择初始的k个类别中心，u1, u2, ......, uk。

II.对于每个样本的xi，将其中标记为距离类别中心最近的类别，即：



III.将每个类别中心更新为隶属该类别的所有样本的均值。



V.重复II、III，直到类别中心变化小于某阈值。

1. 案例分析

根据课程所提供的数据，我们提出了三个能够衡量题目难易程度的指标，分别是题目平均得分，提交代码平均行数，提交代码python代码占比。在上学期的编程过程中我们发现，在完成部分困难的题目时，同学们会倾向于复制网上查找到的代码，且这些代码大部分都是java或是c++代码，所以可以推测某个题目越困难，其提交代码的python代码占比就会越小，平均得分就会越低，代码平均长度会越长。

根据所提供的数据test\_data.json，我们求出每道题目的平均得分，代码平均长度以及python代码占比，三个指标的原始数据如下表1所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 表1 部分原始数据 | | | |
| ID | pyPercent | countLine | mark |
| 2061 | 1 | 27.30851064 | 97.87234043 |
| 2063 | 1 | 4.170212766 | 100 |
| 2064 | 1 | 12.39361702 | 100 |
| 2065 | 1 | 11.85106383 | 100 |
| 2067 | 1 | 13.85869565 | 100 |
| 2068 | 1 | 7.893617021 | 100 |
| 2069 | 1 | 5.882978723 | 100 |
| 2070 | 1 | 5.206521739 | 100 |
| 2080 | 0.925 | 12.6 | 95.12195122 |
| 2081 | 1 | 5.066666667 | 100 |
| 2084 | 0.891891892 | 19.31081081 | 94.32432432 |
| 2085 | 0.916666667 | 24.61111111 | 94.44444444 |
| 2086 | 0.944444444 | 16.375 | 100 |
| 2087 | 0.911764706 | 18.94117647 | 89.30470588 |
| 2088 | 0.821428571 | 18.10714286 | 93.92857143 |
| 2089 | 0.851851852 | 16.42592593 | 97.03703704 |
| 2090 | 0.884615385 | 18.82692308 | 97.11538462 |
| 2091 | 0.807692308 | 16.21153846 | 90.38461538 |
| 2092 | 0.928571429 | 11.92857143 | 78.04214286 |
| 2093 | 1 | 9.542553191 | 100 |
| 2094 | 1 | 10.17777778 | 99.44444444 |
| … | … | … | … |

由于这三种数据的的范围不同，而且python代码占比与题目的难度成反比，平均代码长度与题目难度成正比，这样的情况对最后的结果会有影响，所以我们先使用了数据标准化的方法，将三组数据都变为值在0-1的区间内，且都与题目的难度成正比，方便了后续工作的进行，数据标准化部分结果如下表2所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 表2 数据标准化部分结果 | | | |
| ID | stdPyPercent | stdCountLine | stdMark |
| 2061 | 0 | 0.348203527 | 0.089834515 |
| 2063 | 0 | 0.033075194 | 0 |
| 2064 | 0 | 0.145072528 | 0 |
| 2065 | 0 | 0.137683312 | 0 |
| 2067 | 0 | 0.165025931 | 0 |
| 2068 | 0 | 0.0837855 | 0 |
| 2069 | 0 | 0.056401935 | 0 |
| 2070 | 0 | 0.047189037 | 0 |
| 2080 | 0.175 | 0.147883328 | 0.20596206 |
| 2081 | 0 | 0.045284304 | 0 |
| 2084 | 0.252252252 | 0.239280138 | 0.23963964 |
| 2085 | 0.194444444 | 0.311466727 | 0.234567901 |
| 2086 | 0.12962963 | 0.199296334 | 0 |
| 2087 | 0.205882353 | 0.234245963 | 0.451579085 |
| 2088 | 0.416666667 | 0.222886976 | 0.256349206 |
| 2089 | 0.345679012 | 0.199989912 | 0.125102881 |
| 2090 | 0.269230769 | 0.232689907 | 0.121794872 |
| 2091 | 0.448717949 | 0.197070096 | 0.405982906 |
| 2092 | 0.166666667 | 0.138738914 | 0.927109524 |
| 2093 | 0 | 0.106242922 | 0 |
| 2094 | 0 | 0.114894261 | 0.02345679 |
| … | … | … | … |

对于标准化后的数据，我们使用改良后的熵权法，对于每个指标的主观权重，我们暂且设置成相等，若后续有相关专家认为多个指标之间权重应有差异，则可再调整。我们先求出各项指标的信息熵，再根据信息熵计算各项指标的差异系数，再通过差异系数计算各项指标的权重。计算结果如下表3所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 表3 各项指标的主观权重、信息熵、差异系数和权重 | | | |
|  | pyPercent | CountLine | Mark |
| 主观权重wj | 1 | 1 | 1 |
| 信息熵ej | 0.99971874 | 0.999883335 | 0.999763462 |
| 差异系数gj | 0.00028126 | 0.000116665 | 0.000236538 |
| 权重Wj | 0.44330401 | 0.183880039 | 0.372815951 |

有了上面各项数据所占的比重以及各项数据标准化后的结果，我们就可以得到对于每道题目难度的数值评价，将这些数据都放在以为坐标上，根据它们之间的距离，也就是他们的相似程度，我们就可以使用聚类的处理方法，将相似的题目归为一类，也就可以由此将题目分为五种不同的类型。最终的结果为：最简单题目有606道，较简单有158道，中等难度有76道，较难有27道，极难有15道，详细结果请参考附录。对聚类结果进行模型效果指标评估，得到silhouette\_s:0.63240。silhouette\_s大于0.5，说明聚类质量较优。

1. 课程意见

由于慕测平台的一些局限性可能会导致本次作业中所使用可供用于题目难度分析的数据较少，得出的结果可能会跟实际有些小小的偏差，是否可以改进慕测系统方便以后提供更加全面的数据，例如：可以记录在网页上运行的记录而不只是提交记录、记录每道题应被完成的用户个数等，虽然更详尽地记录用户行为会耗费更多的系统资源，但是想要更加精确地分析Python题目的各种特征，这些代价我认为是不可避免的。

1. 附录