数据科学基础大作业研究报告

学号	姓名	邮箱	Python 练习完成题目数量
181250089	刘一铭	181250089@smail.nju.edu.cn	199
181250129	檀潮	181250129@smail.nju.edu.cn	200
181250146	王宇博	181250146@smail.nju.edu.cn	200

一、研究问题

通过对学生在某类题目上的掌握程度进行分析，针对每个同学在编程中的弱项，个性化推荐相应的编程练习题目。

二、研究思路

首先，寻找一个合适的度量指标，将学生对题目的掌握程度进行量化。我们认为，针对一道题目，所有同学的掌握程度应当服从正态分布，即掌握的很差或很好的学生人数较少，中间范围内的人数较多，所以还需要对找出的度量指标所对应的数据的正态性进行检验，当满足正态分布时，认为找出的该度量指标是合理的，可以用其作为衡量学生对题目掌握程度的标准。

其次，我们对每位同学在每道题上的四个指标数据进行综合，针对不同题型下每位同学各自的数据进行正态性检验，当满足正态分布时，认为每种题型下的题目难度安排合理，方便之后根据学生已做题目上的掌握程度来推荐难度相近的适宜题目。

最后实现题目的推荐机制。

三、代码开源地址

https://github.com/Xassago/dataScience

四、研究方法

1、生成源数据集

在对 test_data.json 中原始数据的观察分析之后，我们决定按 case_id 对 test_data.json 中的数据进行分类，每个 case_id 下都包含了所有学生在该题上的各自的提交数据，以此生成研究第一阶段所需的数据集。（进行该数据处理过程的 Python 文件为 update_data.py，生成的数据集为 case_data.json）

同时，按“类型/caseId”对 test_data.json 中的数据进行分类,生成第三阶段所需的数据集。（进行该数据处理过程的 Python 文件为 update_data.py，生成的数据集为 classified_data.json）

2、计算度量指标

数据约定：

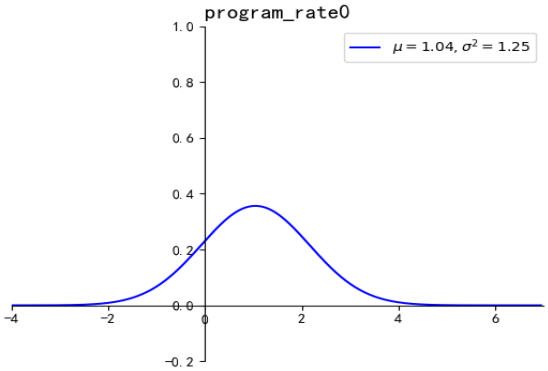
“完成度”：得分与满分（100）的比值

“提交时间间隔”：两次提交时间差

指标一：program_rate

计算方法：针对每一道题，筛选出提交次数四次及以上的学生的数据，对其中的每位学生，计算其在该题上的最大完成度和最大提交时间间隔的比值，然后将之代入 ln(x+1)得出结果，若最高分为 0 则将结果置 0，所得即为 program_rate 值。

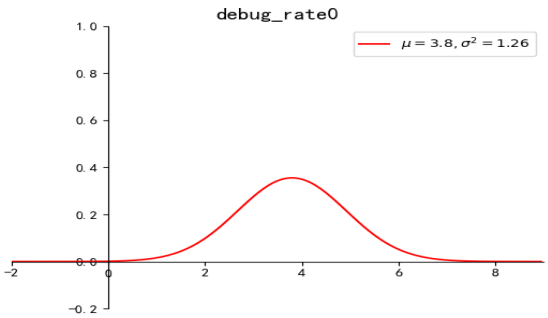
理由：我们认为最大完成度可以作为学生对题目的最终理解程度的一种度量，而提交时间的总跨度可以衡量学生的做题时间，包括 think、program 和 debug 的时间。代入 ln(x+1)不影响单调性且结果最小值为 0（在此指标计算中 x 最小值为 0），而最高分为 0 的学生按最小值 0 处理。



指标二：debug_rate

计算方法：针对每一道题，筛选出提交次数三次及以上的学生的数据，对其中的每位学生，筛选出其提交记录中相邻两次提交时间间隔小于一小时且完成度有提升的记录，计算其中每两次提交完成度之差的平方的平均数，然后对该值开方并将之代入 ln(x+1)得出结果，若最高分为 0 则将结果置 0，所得即为 debug_rate 值。

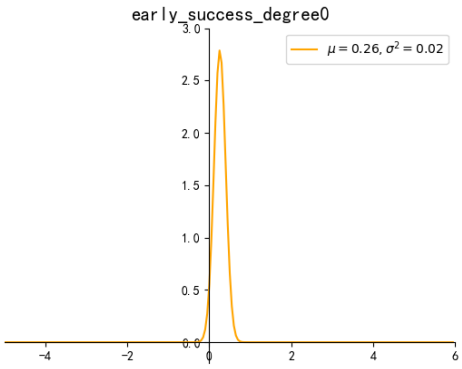
理由：限定只计算完成度提升的记录数据，以此期望所求的数值有更多可能是在原方法下攻克用例期间得到，而不是在试错或者转换方法期间得到，使得行为与 debug 更相符；限定计算平均数，使得每次分数提升都作为测量的一部分，与 program_rate 的测量标准相区别；限定只计算一小时内的分数提升，使得所测量的行为与 debug 更相符。代入 ln(x+1)不影响单调性且结果最小值为 0（在此指标计算中 x 最小值为 0），而最高分为 0 的学生按最小值 0 处理。



指标三：early_success_degree

计算方法：针对每一道题，筛选出提交次数四次及以上的学生的数据，对其中的每位学生，筛选出其提交记录中相邻两次提交完成度有提升的记录，计算记录中完成度的均值，将之代入 ln(x+1)得出结果，若最高分为 0 则将结果置 0，所得即为 early_success_degree 值。

理由：对于一道题，分数提高的越早，可以体现编程者对题目的理解速度越快，故用每次满分程度的均

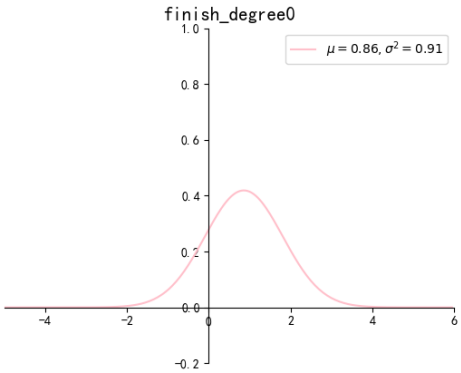


值来刻画此指标，使得分数提高越早时高分的权重越大、此指标的值越大；而对于分数下降的时候，我们认为这是编程者在尝试别的方法，其对题目的理解程度应保持与原来不变，所以仍采用之前分数的最大值计算。代入 ln(x+1)不影响单调性且结果最小值为 0（在此指标计算中 x 最小值为 0），而最高分为 0 的学生按最小值 0 处理。

指标四：finish_degree

计算方法：针对每一道题，筛选出提交次数四次及以上的学生的数据，对其中的每位学生，计算其每次提交时完成度（若本次提交完成度较之前下降，则按照之前提交记录中最高完成度计）与提交时间间隔（此处为首次提交时间与本次提交时间差）的比值的平均数，然后将之代入 ln(x+1)得出结果，若最高分为 0 则将结果置 0，所得即为 finish_degree 值。

理由：在该指标下，越快得到越高的分数，那么该指标的值就越大，但是如果提交并没有带来分数的提升，即分数没有改变或甚至下降，那么就可以认为在该题上可能遇到了困难，同时由于时间增加而分数未提升，该指标的值也会降低，以此来描述学生对题目的完成速度，间接反映其对题目的掌握程度。代入 ln(x+1)不影响单调性且结果最小值为 0（在此指标计算中 x 最小值为 0），而最高分为 0 的学生按最小值 0 处理。



3、生成中间数据集

依次计算每道题下每位学生做题数据的上述四个指标的具体数值，去除无效数据，并筛选出符合“含有有效指标个数为 15 及以上”条件的 case_id，然后将所得数据以四个指标进行分类，写入 intermediate_case_data.json 文件中，作为研究第一阶段的中间数据集。（具体代码见 Python 文件 calculate_potential_masterValue.py）

4、指标合理性的正态性检验

我们采用偏度、峰度检验的方法来检验中间数据集中数据的正态性。（偏度、峰度检验代码见 NDT_for_potential_masterValue.py 文件）

对于不同 caseId 的题的有效做题记录计算出上述四个指标，得到“指标类型/caseId”目录下“指标值”的集合，对每个集合进行如下验证：

计算步骤：

Ⅰ. 取显著性水平 α 为 0.004，则 $z_{\alpha/4} = z_{0.001} = 3.09$，n 为相应指标下的数据数量

Ⅱ. 使用如下公式计算 σ_1，σ_2，μ_2

$$\sigma_1 = \sqrt{\frac{6(n-2)}{(n+1)(n+3)}},\quad \sigma_2 = \sqrt{\frac{24n(n-2)(n-3)}{(n+1)^2(n+3)(n+5)}},\quad \mu_2 = 3 - \frac{6}{n+1}$$

Ⅲ. 计算样本中心距 B_2，B_3，B_4，利用以下关系式

$$B_2 = A_2 - A_1^2,\quad B3 = A_3 - 3A_2A_1 + 2A_1^3,\quad B_4 = A_4 - 4A_3A_1 + 6A_2A_1^2 - 3A_1^4$$

其中，$A_k = \frac{1}{n}\sum_{i=1}^{n} X_i^k\ (k = 1,2,3,4)$ 为 k 阶原点矩

Ⅳ. 计算偏度、峰度的观察值 g_1，g_2

$$g_1 = \frac{B_3}{B_2^{3/2}},\quad g_2 = \frac{B_4}{B_2^2}$$

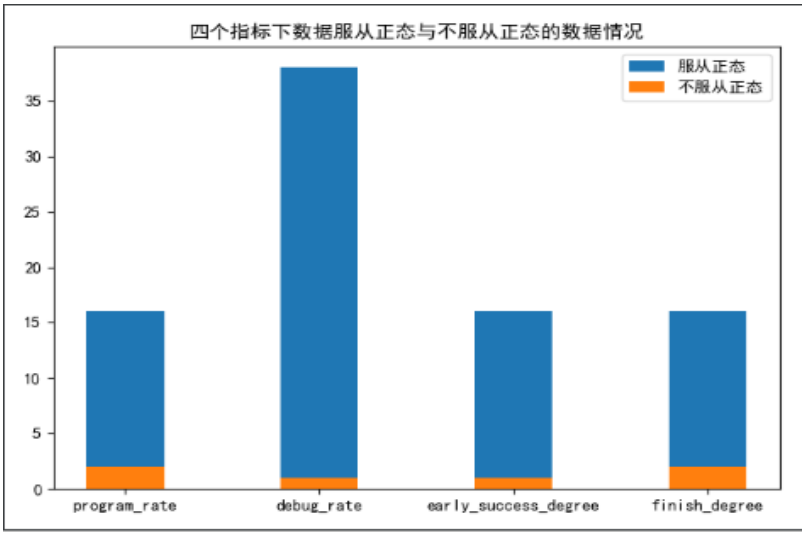
Ⅴ. 确定拒绝域

由于 $z_{\alpha/4} = z_{0.001} = 3.09$，则拒绝域为 $|u_1| = |g_1/\sigma_1| \geq 3.09$ 或 $|u_2| = |g_1 - \mu_2|/\sigma_2 \geq 3.09$

Ⅵ. 判定样本数据是否符合正态

假设算得 $|u_1| < 3.09$，$|u_2| < 3.09$，那么认为数据服从正态分布，否则认为不服从

验证结果：




```
各指标下数据服从正态分布的比例：
program_rate: 87.5%
debug_rate: 97.36842105263158%
early_success_degree: 93.75%
finish_degree: 87.5%
```

对于不同 caseId 的题的有效指标值进行经偏度、峰度检验，上述四个度量指标下的数据服从正态分布的比例均在 85%以上，能够有效区分学生的水平，由此我们认为所选的度量指标较为合理，可以作为刻画学生对于题目掌握程度的度量标准。

5、掌握值的计算

对于四种指标，计算所有有效做题记录下的均值与方差，由均值与方差对指标进行标准正态化，此时指标的均值为 0，值域为（$-\infty,+\infty$）。对于每个学生，计算其四个指标。无效的指标意味着按照做题记录难以提取的指标，并不是在该项指标下无成绩，故用均值替代，使之对学生掌握值排名的影响尽可能小。计算四个指标的均值为学生的掌握值。四个指标均无效时认为无法提取学生的有效做题信息，此时学生的掌握值无效。（代码见 calculate_masterValue.py 文件）

6、八大题型下学生成绩的正态性检验

我们仍采用偏度、峰度检验的方法来检验中间数据集中数据的正态性。（偏度、峰度检验代码见 NDT_for_student_masterValues.py 文件）

对于八大题型下每个学生的有效做题记录计算出上述掌握值，得到“题型”目录下“指标值”的用户有效掌握值集合的集合，筛去元素数量小于等于 5 的掌握值集合，对其中元素数量大于 5 的集合进行如下验证：

计算步骤：

Ⅰ. 取显著性水平 α 为 0.004，则 $z_{\alpha/4} = z_{0.001} = 3.09$，n 为相应指标下的数据数量

Ⅱ. 使用如下公式计算 σ_1，σ_2，μ_2

$$\sigma_1 = \sqrt{\frac{6(n-2)}{(n+1)(n+3)}},\quad \sigma_2 = \sqrt{\frac{24n(n-2)(n-3)}{(n+1)^2(n+3)(n+5)}},\quad \mu_2 = 3 - \frac{6}{n+1}$$

Ⅲ. 计算样本中心距 B_2，B_3，B_4，利用以下关系式

$$B_2 = A_2 - A_1^2,\quad B3 = A_3 - 3A_2A_1 + 2A_1^3,\quad B_4 = A_4 - 4A_3A_1 + 6A_2A_1^2 - 3A_1^4$$

其中，$A_k = \frac{1}{n}\sum_{i=1}^{n} X_i^k$ (k = 1,2,3,4) 为 k 阶原点矩

Ⅳ. 计算偏度、峰度的观察值 g_1，g_2

$$g_1 = \frac{B_3}{B_2^{3/2}},\quad g_2 = \frac{B_4}{B_2^2}$$

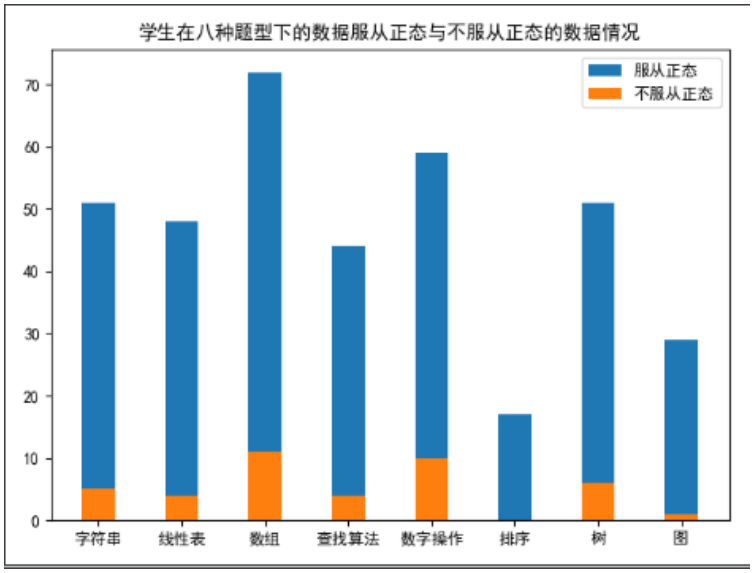
Ⅴ. 确定拒绝域

由于 $z_{\alpha/4} = z_{0.001} = 3.09$，则拒绝域为$|u_1|=|g_1/\sigma_1| \geqslant 3.09$ 或$|u_2|=|g_1-\mu_2|/\sigma_2 \geqslant 3.09$

Ⅵ. 判定样本数据是否符合正态

假设算得$|u_1|<3.09$，$|u_2|<3.09$，那么认为数据服从正态分布，否则认为不服从

验证结果：



```
各题型下学生对所做题掌握值服从正态分布的比例：
str: 90.19607843137256%
line: 91.66666666666666%
arr: 84.72222222222221%
find: 90.9090909090909%
num: 83.05084745762711%
sort: 100.0%
tree: 88.23529411764706%
gra: 96.55172413793103%

八种题型下学生对所做题掌握值服从正态比例的均值：90.6664904928197%
```

对每个题型下的有效掌握程度集合进行经偏度、峰度检验，八种题型下掌握值数据服从正态分布的学生比例（仅在掌握值集合大于 5 的学生中统计）均在 80%以上，平均在 90%以上。换言之，题目难度分布使得每个学生的在同一题型下的各题目分数分布近似符合正态。由此我们认为各题型下的难度设置较为合理。

7、题目推荐模型

根据已经完成的掌握值模型，提出题目推荐模型用于给学生推荐下一道题目。学生可选择在某一特定类型中获得推荐题目，如不选择便会默认采用该学生目前最弱类型题目进行推荐。首先，获得一个学生在各类别下的题目掌握值均值。然后根据其他学生的做题数据对所有题目的难度进行判断，判断依据为该所有已经做过该题的学生掌握值均值，得分越高说明题目越简单，反之越难。其次在学生选择的类型中找出他还未做的所有题目，将题目难度与学生在该类下的成绩最为接近的一题推荐给学生，至此完成题目推荐。（代码详见 recommendTest.py）


```
{
    "user_id": 3544,
    "cases": [
        {
            "case_id": "2908",
            "case_type": "字符串",
            "case_zip": "http://mooctest-site.oss-cn-shanghai.aliyuncs.com/target/单词分类_1581144899702.zip",
            "final_score": 40,
            "upload_records": [
                {
                    "upload_id": 236494,
                    "upload_time": 1582023290656,
                    "code_url": "http://mooctest-dev.oss-cn-shanghai.aliyuncs.com/data/answers/4238/3544/%E5%8D%95%E8%AF%8D%E5%88%86%E7%B1%BB_1582023289869.zip",
                    "score": 40.0
                },
                {
                    "upload_id": 252563,
                    "upload_time": 1582557830339,
                    "code_url": "http://mooctest-dev.oss-cn-shanghai.aliyuncs.com/data/answers/4249/3544/%E5%8D%95%E8%AF%8D%E5%88%86%E7%B1%BB_1582557829552.zip",
                    "score": 0.0
                },
                {
                    "upload_id": 252565,
                    "upload_time": 1582557836610,
                    "code_url": "http://mooctest-dev.oss-cn-shanghai.aliyuncs.com/data/answers/4249/3544/%E5%8D%95%E8%AF%8D%E5%88%86%E7%B1%BB_1582557835991.zip",
                    "score": 0.0
                },
                {
                    "upload_id": 252576,
                    "upload_time": 1582558130645,
                    "code_url": "http://mooctest-dev.oss-cn-shanghai.aliyuncs.com/data/answers/4249/3544/%E5%8D%95%E8%AF%8D%E5%88%86%E7%B1%BB_1582558130012.zip",
                    "score": 0.0
                },
                {
                    "upload_id": 252577,
                    "upload_time": 1582558139684,
                    "code_url": "http://mooctest-dev.oss-cn-shanghai.aliyuncs.com/data/answers/4249/3544/%E5%8D%95%E8%AF%8D%E5%88%86%E7%B1%BB_1582558139071.zip",
```


userCase 数据

```
用户ID: 3544 ;指定推荐题目类型: undefined;最终推荐题目类型: 线性表

userScore（用户在该题型上的掌握值）: -0.6943214321284034
testIds not tried [('2593', -0.17893974087346906), ('2683', -0.10728506203785301), ('2590', -0.09438563
testIds tried [('2908', -0.6943214321284034), ('2179', 0.0), ('2307', 0.0), ('2804', 0.0), ('2450', 0.0

[not duplicate]testIds not tried available in order: ['2593', '2683', '2590', '2686', '2610', '2562', '
[not duplicate]testIds tried available in order: ['2908', '2179', '2307', '2804', '2450', '2176', '2172
next recommended testId: 2593
```

最终输出示例

8、PCA 降维

在上述工作结束后，我们小组也额外进行了数据降维的尝试。经过对 PCA（主成分分析）、LDA（线性判别分析）、LLE（局部线性嵌入）三种降维方法的对比和对已有的掌握值数据的分析后，我们采用了 PCA 来对数据进行降维。（引用了 Python 的 sklearn.decomposition 库中的 PCA 模块进行处理，代码详见 PCA 部分）

1）数据处理：首先将之前基于每道题的掌握值数据进行处理，以某位学生的数据为例，我们将其在一种题型下的题目上的掌握值的均值作为其对该题型的掌握值，生成了初步的数据集（dataForPCA.json）

然后由于部分同学没有完成所有的题目，存在在某些题型上掌握值缺失的情况，所以我们在原有所有同学的掌握值数据基础上进行了筛选，留下了八种题型的掌握值均存在的学生的数据，形成了 PCA 降维所需的数据集（furtherDataForPCA.json）

2）降维：通过调用 sklearn.decomposition 库中的 PCA 模块，我们尝试保留源数据信息的 80%进行降维，最终将八维数据降为了五维，从投影后各特征维度的方差比例可以看出，前三个主成分分别占了 40.7%、17.3%、11.7%的方差比例，可以说该数据集的特征还是较为分散，并没有特别集中。

```
30  pca = PCA(0.8)  # 保留原数据的百分之多少
31  pca.fit(dataChart)  # 训练
32  print("将数据降到了" + str(pca.n_components_) + "维")
33  dataAfterPCA = pca.fit_transform(dataChart)  # 降维后的数据
34  print(list(pca.explained_variance_ratio_))  # 输出贡献率
35  # 输出降维后的数据
36  print("输出降维后的数据：")
37  for x in list(dataAfterPCA):
38      print(list(x))
```

```
将数据降到了5维
投影后各特征维度的方差比例如下：
[0.40714450718386724, 0.17352505487626468, 0.11688128176458623, 0.0995324252470594, 0.08237129767311173]
输出降维后的数据：
[-1.1892557495793 52, -0.03459266463405573, -0.2398734827414423, -0.20509620379655563, -0.31416705713460014]
[0.06853479720473171, 0.5311048066636639, 0.6838544527152772, 0.06261558360078362, 0.07211606417753638]
[1.264121163525966, -0.32555035401898674, 0.20273809353377711, -0.2505816790171723, -0.011245460658070224]
[-0.09804621909338522, -0.34428414441042576, 0.09417669826012531, -0.3036945576892953, 0.25696936853035324]
[-0.15875055137927627, -0.7410682330661731, 0.30130034696189806, 0.20604749634846095, -0.01191386507002765]
[-0.15354572407527098, -0.20006378661044763, -0.2561678653993459, 0.015411800660246378, 0.3002843959949066]
[-0.4109180314335584, -0.3317938833896747, 0.21017440402752524, 0.13707113592794937, 0.02422742944999598]
[0.16797271679322665, 1.0156232038142923, 0.1651350115941349, -0.20338127361655028, 0.7188270124777942]
[-0.291539566200257, -0.539735387238236, 0.13951385141961822, -0.13883413841145462, 0.18680567220780464]
[0.34646219624325547, 0.26911337844532346, 0.32707392839938526, 0.8564892899709824, -0.49724306586558736]
[1.0861521710222974, -0.795296146369705, -0.12490109192324234, -0.2501107913711899, -0.15820669988937436]
[-1.068759262305104, 0.7428884529494253, 0.3385731867206094, -0.1424595275186821, 0.11056262434046164]
```