| | |
|---|---|
| 队伍编号 | dsa2101116 |
| 题号 | （A） |

# 硕士学位论文评价数据的统计与分析

## 摘 要

在学位论文抽检中，使用统计与分析的方法对硕士论文的真实水平进行挖掘对提高论文的评价效率和评价的科学性具有重要意义。

本文首先基于统计分析知识筛选出问题论文，其次统计计算了论文的各项得分数据并进行了分析，然后运用无监督学习的方法建立基于文本数据的论文评价模型，进而通过论文评价模型，结合论文的统计特征，构建了一套具有科学性的学位论文评价策略，最后根据统计学知识对问题论文与优秀论文进行对比分析。

问题 1：该问题涉及数据的统计计算，本文首先计算所有论文的总分平均分，再按学科门类计算该学科的后 5%总分平均分的分位数，免去了对论文的排序步骤，直接比较论文总分平均分与学科内的分位数的大小即可判断出问题论文。

问题 2：该问题涉及数据的统计计算与统计结果分析，本文选择 3 位评阅专家总分打分的标准差作为统计变量分析，利用总分标准差的平均值反应 3 个总分之间一致性和差异性，利用总分标准差的标准差反应不同论文之间打分一致性的差异；选择各学科各项得分平均值与标准差分析对比不同学科的论文水平。

问题 3：该问题涉及文本分析，本文首先对文本数据进行预处理（切词、去停词和词向量化），然后选择 K-means 聚类算法作为文本分类的模型，构建聚类分类与文本分析结果的对应关系，最后对文本分析结果与各项得分之间对比，得到一致性分析结果。

问题 4：该问题涉及文本分析与统计计算，本文将文本分析评价模型与论文的统计特征结合构建了论文综合评价机制，首先判断文本分析结果与论文原始得分是否一致，若不一致则根据原始得分标准差对原始得分进行调整，依据上述步骤分别对 3 位专家的评语与打分进行分析，最后求取调整后得分的平均值即为综合得分。

问题 5：该问题涉及统计分析，根据上述问题得到的附件 2 中的统计信息，提取其中工学学科的信息，根据“选题与综述”、“创新性及论文价值”、“科研能力与基础知识”、“论文规范性”、“总分”和“综合得分”等多个尺度的指标对问题论文与优秀论文进行对比分析。

关键词：文本分析、数据处理、聚类算法、统计计算

# 目 录

# 1 问题重述

## 问题背景

从翟天临学位论文涉嫌学术不端被查，到教育部和科技部推出的“破四唯”，让学位论文质量成为了研究生学位授予质量的重要评价参考。从培养单位的角度看，学位论文质量的高低是衡量研究生学术水平的重要手段。

严格把关论文质量是提高学生水平、促进学科进步的必要之举。但是随着研究生扩招，论文数目增多，严格审查论文需要投入大。如何利用数据统计分析方法进一步的提高对数据的利用效率，减轻人力负担是亟需解决的问题。

目前，许多学者针对相关问题展开了研究，文献[1]中研究表明“选题意义”“理论基础”“独立科研能力”这三项对综合评价得分具有显著性影响。文献[2]提出了博士学位论文质量多元多维度评价方法。文献[3]针对教育学专业学位论文展开研究，运用层次分析法确定了以“研究水平”“写作水平”为基本维度的教育硕士专业学位论文评价指标体系。文献[4]针对工程硕士学位论文展开研究，提出了工程硕士论文质量评价与保障策略。文献[5]对千余份学位论文盲审结果进行了分析，得出了以下结论：论文研究内容的创新性则和论文的总体评价息息相关；同学科的学位论文水平有明显差异；不同专家对论文合格线的把控差距较大，盲环节存在无法完全筛选出合格学位论文的问题。

从上述文献中可以看出，影响论文得分的因素有很多，并且论文在不同方面的表现都影响着最终得分。与此同时，不同专家的评价差异大，只凭借分数不能够完全反应论文水平。这些问题需要能够深入挖掘数据信息、深度探查论文水平的数据统计分析方法与论文评价策略。

文本分析是对文本的表示及其特征项进行选取，是文本挖掘、信息检索的一个基本问题，它把从文本中抽取出的特征词进行量化以表示文本信息。

文本分析一般主要由三步组成，解析数据，搜索检索，文本挖掘。解析数据主要是为了将非格式化的数据处理成格式化的数据以方便以后的分析，非结构化的数据主要有文本，日志，网页，xml，json 等；搜索检索主要是指对结构化的数据识别关键字，主题，以及相关性等；文本挖掘主要是根据识别的关键字，主题等找出其中的我们感兴趣的东西，并展示出来。

随着机器学习、数据挖掘、互联网技术等方面的快速发展，文本分析与文本挖掘理论方法的研究和应用已经成为前沿热点问题，在多个领域已经取得有价值的研究成果[6]。抛除对数值信息的挖掘，更重要也是更困难的是对文本信息的分析与挖掘，目前机器学习与深度学的方法被广泛的应用与文本分析，并且在其中达到了良好的效果[7]。因此，使用文本分析方法有利于提高论文评价的准确度与效率。

## 问题描述

问题 1：

请参赛者基于上述“末位后 5%淘汰制”，筛选出问题论文。填写附件 2。（该省教育厅采用了平均分和最低得分的方式对论文质量进行评价，提取 3 位评阅专家给出的总分最低分进行学科门类内排名，用“末位后 5%淘汰制”定出问题论文，用 3 位评阅专家给出的总分平均分反应论文的整体水平。）

问题 2：

计算每篇论文的各分项平均分和总分平均分。填写附件 2。按照学科门类，分别统计分析 3 位评阅专家给出的 3 个总分以及 1 个总分平均分之间的一致性和差异性，并分析对比各学科门类学位论文的水平

问题 3：

按照学科门类统计分析论文评阅评语，采用文本分析方法，建立评阅专家观点评价模型，并论证文本分析结果与其各分项得分间是否存在一致性。

问题 4：

应用问题 2 和问题 3 得到的结论对附件 1 中的每篇论文进行综合评价，给出综合得分。填写附件 2。

问题 5：

以学科门类工学（08）为例，对比问题论文（被淘汰的论文）与优秀论文（综合得分排名前 10%）的典型特征。

# 2 问题假设

## 问题 1

1.三位评阅专家给出的总分可以准确的反应学位论文的水平
2.末位 5%论文数目不是整数时，可以对其进行舍入

## 问题 2

1．每个学科门类的专家的打分指标都在同一个水平上，以保证不同学科门类的分数可以进行比较
2.三位评阅专家对每篇论文的各项打分与总分可以准确反应该学位论文在各个方面与整体的水平

## 问题 3

1.论文的评阅评语能够反映论文在各分项的水平
2.论文的各分项得分水平可以通过对评阅评语的无监督学习得到

## 问题 4

1.论文的评阅评语与打分都能一定程度的反应该论文的水平

## 符号说明

表 1　符号意义

| 符号 | 意义 |
| --- | --- |
| $Xi$ | 论文第 $i$ 分项得分平均分，$i$ 从 1 到 4 依次代表“选题与综述”、“创新性及论文价值”、“科研能力与基础知识”、“论文规范性” |
| $Xji$ | 论文第 $i$ 分项的第 $j$ 位专家打分 |
| $N$ | 专家数目，在本文中数值为 3 |
| $X$ | 论文总分平均分 |
| $Xj$ | 论文总分的第 $j$ 位专家打分 |
| $\sigma_X$ | 论文的总分标准差 |
| $\sigma_X^m$ | 所有论文的总分标准差的均值 |
| $\sigma_X^l$ | 第 $l$ 个论文的总分标准差 |
| $M$ | 论文数目 |
| $t$ | 表示 Tag，学科门类 |
| $X_t$ | 学科门类的论文总分的均值 |
| $\sigma_{Xt}$ | 学科门类的论文总分的标准差 |
| $Xi_t$ | 学科门类的子项的均值 |
| $\sigma i_t$ | 学科门类的子项的标准差 |
| $\sigma_{Xt}^m$ | 学科内论文中 3 位评阅专家打的 3 个总分的标准差的平均值 |
| $\sigma_\sigma$ | 3 位评阅专家打的 3 个总分的标准差的学科内论文的标准差。 |
| $Xj_r$ | 根据文本分析对第 $j$ 位专家对论文原始打分的修正值 |
| $X_{mt}$ | 学科门类内论文总分平均分 |
| $X_c$ | 论文综合得分 |
| $T$ | 论文文本分析结果 |

# 3 问题的分析与求解

## 问题 1 的分析

由问题描述可知，问题 1 需要对附件 1 中的数据按学科门类进行分类，并且需要对 3 位评阅专家出的总分计算平均值，再根据每篇论文的总分平均值进行学科门类内的排名，最后选取末位 5%被淘汰的论文。

本文采取的思路为首先利用 pandas 库读入数据并计算 3 位评阅专家的总分平均值，再从中筛选各学科门类，进而计算论文总分平均分的后 5%分位数对应的分数，筛选小于该分数的论文为淘汰的论文。

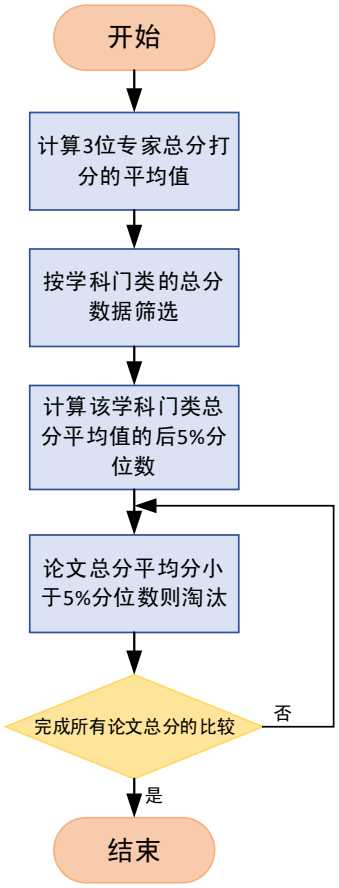


图 1　问题 1 程序流程图

## 问题 1 的求解

上述问题涉及到数据处理与分析，本文采用 Python 作为编程工具对数据进行处理，利用 pandas 与 numpy 库内置的函数可以提升数据处理与分析的效率。

首先，读取附件 1 的数据，

```
# 读取 xlsx 文件
data = pd.read_excel('附件 1.xlsx')
```

然后，获取样本数目，定义评阅专家数目和末尾淘汰的比例，

```
Num = len(data) # 样本数目
T_num = 3 # 评阅专家数目
percent = 5 # 筛选 末尾 5%
```

本文采取先对所有论文的总分求均值，再按学科门类进行分类，进而筛选问题论文的方式，定义求均值函数如下：

```
def get_ave(df1, T_num):
data = df1.values #  取 DataFrame 的值
ave_data = data.sum(axis = 1) / T_num #  求总分均值
return ave_data #  返回总分均值数据
```

先把数据中需要求均值的列筛选出来，

```
X = data[['X1','X2','X3']] #筛选出 X1,X2,X3  三个总分列
```

对筛选出来的总分列按行求均值，

```
X_ave = get_ave(X, T_num)#求均值
```

将求出的总分均值与学科门类标签赋值给 X，

```
X['ave'] = X_ave #赋值总分均值
X['Tag'] = data['Tag'] #赋值学科门类标签
```

按学科门类将数据分类，并且按学科门类计算总分均值的末尾 5%分位数，然后进行比较操作，即可得到需要被淘汰的论文。

```
for i in range(1,14):
    if i == 6 or i == 11: #  不存在的学科门类标签
    print('empty')
else:
    Tag = X.loc[X['Tag'] == i] #  学科分类
    percent_val = np.percentile(Tag['ave'], percent) #  计算分位数
    lose += list(Tag['ave'] < percent_val) #  比较论文是否为末位 5%
```

得出计算结果后，需要将计算结果写入到附件 2 中，那么首先读入附件 2：

```
data_get = pd.read_excel('附件 2.xlsx')
```

然后对附件 2 中的‘是否淘汰’列进行赋值操作：

```
data_get['是否淘汰'] = lose
```

再将三个评阅专家的总分的平均值写入附件 2：

```
data_get['论文总分平均分'] = X_ave
```

至此，问题 1 淘汰的论文已经筛选完毕并且写入附件 2。

### 问题 2 的分析

问题 2 分为三个方面，第一个方面是要对附件 1 中的数据信息进行统计运算，得出各个分项与总分的平均分并写入附件 2；第二方面，按学科门类对三位评阅专家给出的 3 个总分及一个平均分的一致性与差异性分析，那么由于平均分是三个总分的统计均值，因此 3 个总分的一致性与差异性就可以通过统计标准差来表述，标准差越大说明 3 个总分以平均分为中心越分散；最后一方面是分析各个学科门类的学位论文水平，那么该项可以通过各个学科门类的总分平均值的平均水平与标准差来分析，同时也可以通过各个子项的分数平均值来分析不同学科门类在不同分项上的得分差异。

首先是计算论文各分项平均分：

$$Xi = \frac{\sum_{j=1}^{N} Xji}{N}, i = 1,2,3,4 \tag{1}$$

式中， $Xi$ 为第 $i$ 子项评分的 3 位专家打分均值， $Xji$ 为第 $i$ 分项的第 $j$ 位专家打分， $N$ 为专家的数目。

其次，计算论文总分平均分：

$$X = \frac{\sum_{j=1}^{N} Xj}{N} \tag{2}$$

式中，$X$ 为论文总分的 3 位专家打分平均分，$Xj$ 为总分第 $j$ 位专家打分。

最后，需要计算 3 个总分的标准差：

$$\sigma_X = \sqrt{\frac{1}{N}\sum_{j=1}^{N}\left(Xj - X\right)^2} \tag{3}$$

式中，$\sigma_X$ 为 3 个总分的标准差，$N$ 为 3 是专家的数目，$Xj$ 为第 $j$ 位专家的总分打分，$X$ 为三位专家总分打分的均值。

由于论文数据众多，每篇论文都有其总分打分的标准差，需要对其求取均值反应整体水平的标准差：

$$\sigma_X^m = \sum_{l=1}^{M} \sigma_X^l \tag{4}$$

式中，$\sigma_X^m$ 为所有论文的总分标准差的均值，$\sigma_X^l$ 为第 $l$ 个论文的总分标准差，$M$ 为论文数目。

$X_t$、$\sigma_{Xt}$、$Xi_t$、$\sigma i_t$ 的计算与上述平均值与标准差计算同理。

问题 2 的求解

那么接续问题 1，可以从读取的附件 1 中获取三个评阅专家对每个项目的评分进而进行求均值运算，先分别获取每一分项的三个评阅专家评分 $Xji$：

```
Xk1 = data[['X11','X21','X31']] #  ‘选题与综述’的三个评阅专家评分
Xk2 = data[['X12','X22','X32']] #  ‘创新性与论文价值’的三个评阅专家评分
Xk3 = data[['X13','X23','X33']] #  ‘科研能力与基础知识’的三个评阅专家评分
Xk4 = data[['X14','X24','X34']] #  ‘论文规范性’的三个评阅专家评分
```

获取数据后则需要对其求得分均值 $Xi$：

```
Xk1_ave = get_ave(Xk1, T_num) # 对‘选题与综述’的三个评阅专家评分求均值
Xk2_ave = get_ave(Xk2, T_num) # 对‘创新性与论文价值’的三个评阅专家评分求均值
Xk3_ave = get_ave(Xk3, T_num) # 对‘科研能力与基础知识’的三个评阅专家评分求均值
Xk4_ave = get_ave(Xk4, T_num) # 对‘论文规范性’的三个评阅专家评分求均值
```

计算出各项分数的平均值 $Xi$ 后，需要将其赋值到附件 2 的数据对象中：

```
data_get['选题与综述平均分'] = Xk1_ave
data_get['创新性及论文价值平均分'] = Xk2_ave
data_get['科研能力与基础知识平均分'] = Xk3_ave
data_get['论文规范性平均分'] = Xk4_ave
```

最后需要输出附件 2 的文档：

```
data_get.to_excel('附件 2.xlsx', index=None)
```

其中 index = None 为防止将数据的第一列读作表格索引。

首先获取附件 1 中的除评语以外的数据：

```
Tag_all = data.drop(columns = ['R1','R2','R3'])
```

再将总分按学科门类进行划分：

```
Tag_all_dict = dict()
for i in range(1,14):
Tag_temp = Tag_all.loc[Tag_all['Tag'] == i]
Tag_all_dict.update({'Tag' + str(i):Tag_temp})
```

然后按学科门类对其中的所有总分评分数据提取并计算其平均值 $X_t$ 与标准差 $\sigma_{Xt}$，并在每个学科门类下按每个评分子项计算其平均值 $Xi_t$ 与标准差 $\sigma i_t$，并将其存入字典对象中：

```
Tag_all_mean_dict = dict()
for i in range(1,14):
    Tag_xk1 = Tag_all_dict.get('Tag' + str(i))[['X11','X21','X31']]
    Tag_xk2 = Tag_all_dict.get('Tag' + str(i))[['X12','X22','X32']]
    Tag_xk3 = Tag_all_dict.get('Tag' + str(i))[['X13','X23','X33']]
    Tag_xk4 = Tag_all_dict.get('Tag' + str(i))[['X14','X24','X34']]
    Tag_x = Tag_all_dict.get('Tag' + str(i))[['X1','X2','X3']]
    df1 = pd.DataFrame(index = ['mean','std'], columns = ['Xk1','Xk2','Xk3','Xk4','X'])
    df1['Xk1'] = [Tag_xk1.values.mean(), Tag_xk1.values.std()]
    df1['Xk2'] = [Tag_xk2.values.mean(), Tag_xk2.values.std()]
    df1['Xk3'] = [Tag_xk3.values.mean(), Tag_xk3.values.std()]
    df1['Xk4'] = [Tag_xk4.values.mean(), Tag_xk4.values.std()]
    df1['X'] = [Tag_x.values.mean(), Tag_x.values.std()]
    Tag_all_mean_dict.update({'Tag'+str(i):df1}) # 每个学科的分项与总项的均值与方差（不区分打分老师）
```

**问题 2 的结果分析**

表 2 中 Tagx 代表学科门类编码，$\sigma_{Xs}^{m}$ 代表学科内论文中 3 位评阅专家打的 3 个总分的标准差的平均值，$\sigma_{\sigma}$ 代表的是 3 位评阅专家打的 3 个总分的标准差 $\sigma$ 的标准差。

根据表 2 可以看出在同一篇论文中 3 位评阅专家打的总分一致性最高的是理学（Tag07），一致性最差的是经济学（Tag02），一致性高表明针对同一篇论文 3 位评阅专家的评分较为趋同，一致性差则反之。

表 2　不同学科门类的 3 位评阅专家给出的总分标准差的均值与标准差

| | Tag01 | Tag02 | Tag03 | Tag04 | Tag05 | Tag07 |
|---|---|---|---|---|---|---|
| $\sigma_{Xs}^{m}$ | 4.760461 | 5.103326 | 4.616701 | 4.142464 | 4.429407 | 2.637305 |
| $\sigma_{\sigma}$ | 2.660182 | 2.24231 | 2.544883 | 2.495154 | 2.55552 | 2.298368 |

续表

| | Tag08 | Tag09 | Tag10 | Tag12 | Tag13 |
|---|---|---|---|---|---|
| $\sigma_{Xs}^{m}$ | 4.282323 | 3.770609 | 2.792855 | 4.641842 | 4.608504 |
| $\sigma_{\sigma}$ | 2.667558 | 2.003784 | 2.403366 | 2.40216 | 2.465996 |

其次，还可以看出不同的学科门类内在每篇论文上的一致性变化不大，$\sigma_\sigma$最小的学科是农学（Tag09），最大的是工学（Tag08）。$\sigma_\sigma$数值大表明不同论文的总分标准差变化大，也就是不同的论文 3 位评阅专家的评分一致性变化较大，$\sigma_\sigma$数值小则反之。

表 3　哲学学科的各个子项与总分的统计均值与标准差

| | Xk1 | Xk2 | Xk3 | Xk4 | X |
|---|---|---|---|---|---|
| Mean | 20.31373 | 19.43137 | 19.92157 | 20.23529 | 80.13725 |
| Std | 2.539802 | 2.760043 | 2.565557 | 2.624083 | 9.218523 |

表 4　经济学学科的各个子项与总分的统计均值与标准差

| | Xk1 | Xk2 | Xk3 | Xk4 | X |
|---|---|---|---|---|---|
| Mean | 20.0719 | 19.54248 | 19.43137 | 19.30065 | 78.45752 |
| Std | 1.818944 | 2.108145 | 2.43548 | 2.289747 | 7.416628 |

表 5　法学学科的各个子项与总分的统计均值与标准差

| | Xk1 | Xk2 | Xk3 | Xk4 | X |
|---|---|---|---|---|---|
| Mean | 20.67296 | 20.08176 | 20.02516 | 20.2956 | 81.07547 |
| Std | 1.837748 | 2.231756 | 2.203341 | 2.266944 | 6.852027 |

表 6　教育学学科的各个子项与总分的统计均值与标准差

| | Xk1 | Xk2 | Xk3 | Xk4 | X |
|---|---|---|---|---|---|
| Mean | 20.06481 | 19.46296 | 19.38426 | 19.97685 | 78.88426 |
| Std | 2.161417 | 2.231616 | 2.354179 | 2.228689 | 7.785345 |

表 7　文学学科的各个子项与总分的统计均值与标准差

| | Xk1 | Xk2 | Xk3 | Xk4 | X |
|---|---|---|---|---|---|
| Mean | 20.63235 | 19.72059 | 19.7451 | 19.4902 | 79.64706 |
| Std | 2.050139 | 2.237244 | 2.366789 | 2.318912 | 7.229633 |

表 8　理学学科的各个子项与总分的统计均值与标准差

| | Xk1 | Xk2 | Xk3 | Xk4 | X |
|---|---|---|---|---|---|
| Mean | 20.75587 | 21.39906 | 21.07512 | 20.38498 | 82.29577 |
| Std | 1.922568 | 12.1081 | 6.770135 | 2.138878 | 6.101784 |

表 9　工学学科的各个子项与总分的统计均值与标准差

| | Xk1 | Xk2 | Xk3 | Xk4 | X |
|---|---|---|---|---|---|
| Mean | 20.46411 | 19.9936 | 20.1967 | 19.81752 | 80.2054 |
| Std | 2.214519 | 5.385829 | 5.449948 | 2.425355 | 7.522987 |

表 10　农学学科的各个子项与总分的统计均值与标准差

| | Xk1 | Xk2 | Xk3 | Xk4 | X |
|---|---|---|---|---|---|
| Mean | 20.60938 | 20.17188 | 19.95833 | 19.88021 | 80.60938 |
| Std | 2.214657 | 2.188188 | 2.272648 | 2.323168 | 7.504062 |

表 11　医学学科的各个子项与总分的统计均值与标准差

| | Xk1 | Xk2 | Xk3 | Xk4 | X |
|---|---|---|---|---|---|
| Mean | 19.92183 | 19.52065 | 19.27729 | 19.11357 | 77.85251 |
| Std | 2.460561 | 2.585759 | 2.624445 | 2.874528 | 8.154052 |

表 12　管理学学科的各个子项与总分的统计均值与标准差

| | Xk1 | Xk2 | Xk3 | Xk4 | X |
|---|---|---|---|---|---|
| Mean | 19.92412 | 19.23306 | 19.15447 | 19.04336 | 77.39295 |
| Std | 2.034842 | 2.095263 | 2.339826 | 2.401053 | 7.290715 |

表 13 艺术学学科的各个子项与总分的统计均值与标准差

| | Xk1 | Xk2 | Xk3 | Xk4 | X |
|---|---|---|---|---|---|
| Mean | 18.82292 | 18.39583 | 18.70833 | 18.42708 | 74.35417 |
| Std | 2.32735 | 2.094035 | 2.056072 | 2.154186 | 6.890481 |

从上述表格中可以看出，在选题与综述方面，法学学科的得分平均分最高，艺术学科最低；在创新性与论文价值方面，理学学科的得分平均分最高，艺术学科最低；在科研能力与基础知识方面，理学学科得分平均分最高，艺术学科最低；在论文规范性方面，理学学科得分平均分最高，艺术学最低；从总分角度看，同样是理学学科总分平均值最高，艺术学最低。

从标准差的角度看，理学与工学的分项标准差较大，其他学科的分项标准差较小且较为相近。分项标准差越大表明学科内在同一分项上不同论文的得分分布越分散，表明学科内论文在同一分项方面质量差异较大。总分上，哲学的标准差最大，表明学科内不同论文整体水平差异较大，理学的总分标准差最小，表明论文整体水平差异较小。

根据上述分析，可以得出结论理学学科的论文水平最高，艺术学学科的论文水平最低且与其他学科差距较大，医学、管理学、教育学和经济学论文水平处于中等，哲学、法学、文学、工学和农学论文水平处于上等。但是，理学与工学学科内论文子项得分水平差异最大，哲学学科的论文整体水平差异最大，理学学科整体水平差异最小。

### 问题 3 的分析

该问题首先需要进行数据预处理，按学科门类将论文的评阅评语提取，然后对提取的文本数据进行预处理，转化为机器可以处理的数据。因此需要对句子进行切词，对中文语句进行语句切词用到 jieba 库。由于评阅评语中有许多 jieba 库中不包含的词语，容易引起切词不准确，因此手动定义了一部分的词语库，引入 jieba 切词过程中。切词后由于存在诸多无用的词语，所以需要利用中文停词表去除其中的停词与标点符号。然后构建词袋空间，将词语向量化。

文本数据预处理结束后，分析问题中的描述，该问题并没有给出文本分析的结果标签，所以判断该问题为无监督学习问题，因此本文采取 K-means 聚类方法。通过对评语的分析，通常对“创新型及论文价值”和“科研能力与基础知识”两项的评语具有高度的耦合性，在进行无监督学习时，如果将两者进行区分分类容易造成误分类，因此本文将这两者看作耦合的分类项进行聚类。另外，本文将根据评语建立模型，判断论文在子项中的得分分为优和差两类，将优与差定义为该项的平均分以上或者以下。那么归结为三个分项的二分类问题，然后该聚类问题可以划分为 8 类，利用预处理后的数据，建立聚类模型对其进行分类，可得到文本分析结果。

K-means 算法是最常用的聚类算法之一。在 K-means 聚类算法中，当不同点到某一点的距离较近时则该点将只与离其最近的点被归为一类，通过该算法可以将样本进行聚类，具有

相似特征的样本聚为一类。

表 14　聚类标签与分项得分对应关系

| 聚类标签 | 选题与综述 | 创新性与论文价值、科研能力与基础知识 | 论文规范性 |
|---|---|---|---|
| 0 | 优 | 差 | 差 |
| 1 | 差 | 差 | 差 |
| 2 | 差 | 差 | 优 |
| 3 | 差 | 优 | 差 |
| 4 | 差 | 优 | 优 |
| 5 | 优 | 差 | 优 |
| 6 | 优 | 优 | 差 |
| 7 | 优 | 优 | 优 |

主要思想是：在给定 K 值和 K 个初始类簇中心点的情况下，把每个点(亦即数据记录)分到离其最近的类簇中心点所代表的类簇中，所有点分配完毕之后，根据一个类簇内的所有点重新计算该类簇的中心点(取平均值)，然后再迭代的进行分配点和更新类簇中心点的步骤，直至类簇中心点的变化很小，或者达到指定的迭代次数。

假设给定数据样本 $X$，包含了 $n$ 个对象 $X=\{X_1,X_2,X_3,...,X_n\}$，其中每个对象都具有 $m$ 个维度的属性。K-means 算法的目标是将 $n$ 个对象依据对象间的相似性聚集到指定的 $k$ 个类簇中，每个对象属于且仅属于一个其到类簇中心距离最小的类簇中。对于 K-means，首先需要初始化 $k$ 个聚类中心 $\{C_1,C_2,C_3,...,C_K\},1<k\le n$，然后通过计算每一个对象到每一个聚类中心的欧式距离，如下式所示

$$dis\left(X_i,C_j\right)=\sqrt{\sum_{t=1}^{m}\left(X_{it}-C_{jt}\right)^2} \tag{5}$$

式中 $X_i$ 表示第 $i$ 个对象，$1\le i\le n$；$C_j$ 表示第 $j$ 个聚类中心，$1\le j\le k$；$X_{ih}$ 表示第 $i$ 个对象的第 $h$ 个属性，$1\le h\le m$；$C_{jh}$ 表示第 $j$ 个聚类中心的第 $h$ 个属性。

依次比较每一个对象到每一个聚类中心的距离，将对象分配到距离最近的聚类中心的类簇中，得到 $k$ 个类簇 $\{S_1,S_2,S_3,...,S_k\}$

K-means 算法用中心定义了类簇的原型，类簇中心就是类簇内所有对象在各个维度的均值，其计算公式如下

$$C_l=\frac{\sum_{X_i\in S_i}X_i}{|S_l|} \tag{6}$$

式中 $C_l$ 表示第 $l$ 个聚类的中心，$1\le l\le k$；$|S_l|$ 表示第 $l$ 个类簇中对象的个数; $X_i$ 表示第 $l$ 个类簇中第 $i$ 个对象，$1\le i\le|S_l|$。

问题 3 的求解

首先提取评语的文本文件，读入附件 1，并提取其中的评阅评语文本：

```
import pandas as pd
# 读取 xlsx
data = pd.read_excel('附件 1.xlsx')
text_R = data[['R1','R2','R3']]
```

打开需要保存评阅评语的 txt 文件：

```
text=open("C:/Users/MYM/My_python_codes/DSA/text_words.txt",'w',encoding='GB2312',errors='ignore')
```

然后，依次将 excel 文件中的评语按行写入 txt 文件，跳过其中评语为空的以及评语填写错误（填写为数字的），并且有的评语中包含换行符，需要将其剔除：

```
for s in text_R['R1']:
    if pd.isnull(s) or type(s) == int:
        print('nan')
        count = count + 1
    else:
        s = s.replace("\n",',')
        text.write(s)
        text.write('\n')然后关闭 txt 文件：
text.close()
```

然后，引入 jieba 库，并引入自定义的词语字典：

```
import jieba
#加载自定义词语
jieba.load_userdict("C:/Users/MYM/My_python_codes/DSA/user_dict.txt")
```

打开上面提取评阅评语写入的 txt 文件以及需要写入切词后结果的 txt 文件。利用 jieba 对语句进行切词并将切词结果写入另一个 txt 文件：

```
#打开文件，文件在桌面上，可以自行修改路径
f1=open("C:/Users/MYM/My_python_codes/DSA/text_words.txt","r",encoding='GB2312',errors='ignore')
f2=open("C:/Users/MYM/My_python_codes/DSA/text_words_token.txt",'w',encoding='GB2312',errors='ignore')
for line in f1:
    seg_list = jieba.cut(line, cut_all = False)
    f2.write((" ".join(seg_list)).replace("\t\t\t","\t"))
f1.close()
f2.close()
```

然后读入停用词，停用词用来去除切词结果中无意义的词语：

定义读取停用词的函数为：

```
def get_custom_stopwords(stop_words_file):
    with open(stop_words_file, encoding='utf-8')as f:
    stopwords=f.read()
    stopwords_list=stopwords.split('\n')
    custom_stopwords_list=[i for i in stopwords_list]
    return custom_stopwords_list
```

读入停用词：

```
#停用词函数调用
stop_words_file= "C:/Users/MYM/My_python_codes/DSA/CNstopwords.txt"
stopwords = get_custom_stopwords(stop_words_file)
```

接下来需要转换词向量，并且去除其中的停用词：

```
titles=open("C:/Users/MYM/My_python_codes/DSA/text_words_token.txt",encoding='GB231
2', errors='ignore').read().split('\n') #  读取切词结果
#构建词向量，也就是把分好的次去除停词转化成 kmeans 可以接受的形式
from sklearn.feature_extraction.text import CountVectorizer
count_vec=CountVectorizer(stop_words = stopwords)
km_matrix= count_vec.fit_transform(titles)
```

这样数据的预处理就完成了，下一步需要建立聚类模型并将数据导入进行聚类计算：

```
#开始聚类
from sklearn.cluster import KMeans
num_clusters = 8 #聚为八类，可根据需要修改
km = KMeans(n_clusters=num_clusters)
km.fit(km_matrix)
clusters = km.labels_.tolist()
```

然后聚类计算结束后，将聚类的结果输出为 txt 文件保存：

```
#最后把聚类结果写在一个新的 txt 里面
f3=open("C:/Users/MYM/My_python_codes/DSA/cluster.txt",'w',encoding='GB2312',errors='i
gnore')
for i in clusters:
    f3.write(str(i))
    f3.write("\n")
f3.close()
```

为了论证文本分析与其各分项得分间是否存在一致性，本文对各分项的分数水平与聚类分类定义的分项分数水平进行比较，从而得出文本分析与分项得分的一致性如何。

通过分析确定每个聚类标签对应的每个分项应该取得平均分以上或者以下，首先取出问题 2 已经计算完成的各个学科子项平均分，根据聚类的标号判断该论文的子项得分是否符合该类定义的得分水平。由于聚类将“创新性与论文价值”和“科研能力与基础知识”进行了合并，所以在计算平均分时，将两者的平均分求均值，进行比较。

主要代码示例如下：

```
if pd.isnull(s) or type(s) == int: #  除去空的评语以及评语填写错误的
        print('nan')
 else:
        if data.loc[i,'X11'] >= mean_x1:
            if (data.loc[i,'X12'] + data.loc[i,'X13'])/2 >= mean_x23 :
                if data.loc[i,'X14'] >= mean_x4:      #三个分项都高于平均分
                    if clusters[i] == 7: # 111         #  聚类标号是否为 7
                        right+=1                       #  一致则计数加一
```

通过上述代码得到结果基本可以判断出一致性水平。

### 问题 3 的结果分析

从上述代码可以得到结果，论文评分与评语一致的论文数目为 232 篇，论文总数为 1246，去除评语为空以及评语填写错误的还有 1201 篇论文，从该结果可以看出，评语与论文得分的一致性较差。但是由于无监督学习，文本分析聚类方法的准确度较低，因此也可能是聚类分

析不准确导致的问题。

**问题 4 的分析**

该问题可以通过对结合学科内平均分、评语以及论文原本的打分对其进行综合评分，理论上三者都能一定程度的反应论文的水平。本文首先对论文的评阅评语进行文本分析，并进行聚类，使用二分类将论文的综合水平评为高或者低（分别代表分数应该在平均分以上或者以下）。如果聚类标签反应的论文水平与原始得分不一致，则根据聚类标签对原始得分进行一定的调整，调整的幅度定义为该学科门类总分标准差的二分之一。具体实施方法是，分别对 3 位专家的评阅评语进行分析，如果根据评语该论文的水平应该在学科内平均分以上，但是打分却在学科内平均分以下则将总分加上该学科内总分标准差的一半，反之则减去。最后将根据 3 位专家评语修正后的三个总分进行平均即可得到综合得分。

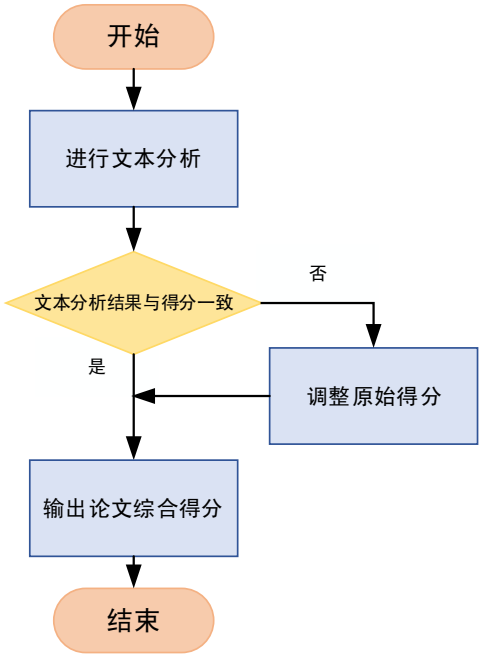


图 2　问题 4 程序流程图

设第 $j$ 位专家对论文原始打分总分为 $Xj$，根据文本分析对第 $j$ 位专家对论文原始打分的修正值为 $Xj_r$，学科门类内论文总分平均分 $X_{mt}$，第为论文综合得分为 $X_c$，论文文本分析结果为 $T$ 则有：

$$X_c = \sum_{j=1}^{N} Xj_r \tag{7}$$

其中，

如果 $T$ 为优秀，

$$Xj_r = \begin{cases} Xj, Xj \geq X_{mt} \\ Xj + \dfrac{\sigma}{2}, Xj \leq X_{mt} \end{cases} \tag{8}$$

如果 $T$ 为较差，

$$Xj_r = \begin{cases} Xj - \dfrac{\sigma}{2}, Xj \geq X_{mt} \\ Xj, Xj \leq X_{mt} \end{cases} \quad (9)$$

其中，$\sigma_{Xt}$ 代表学科门类内的原始总分的标准差。

**问题 4 的求解**

首先对专家评语进行二分类的聚类分析，并将结果输出为文件形式。其次，读取聚类结果与附件 1 数据。

```
#   读取聚类结果
    clusters = []
    f1=open("C:/Users/MYM/My_python_codes/DSA/cluster_q4.txt",'r',encoding='GB2312',errors
='ignore')   # 打开聚类结果文件
    for line in f1:
        clusters.append(eval(line))
```

将聚类结果读取并存储为列表形式后，考虑到需要将每篇论文的原始得分与学科内的平均分水平进行比较，并且需要根据标准差调整分数，因此读取问题 2 计算得到的学科内平均分与标准差信息。

```
#   每个学科的平均分与标准差
    tag_dict = dict()
    tag = (1,2,3,4,5,7,8,9,10,12,13)
    for i in tag:
        temp = pd.read_csv('Tag' + str(i) + '.csv')
        tag_dict.update({'Tag'+ str(i) : temp})
```

之后对每篇论文进行判断，判断总分水平是否与文本分析的结果一致，不一致则对原始总分进行调整，主要代码为：

```
    count = 0   #      聚类结果的序号
    for i in range(1246):
        Tag = data.loc[i,'Tag']   #   取每个论文所属的学科
        mean = tag_dict.get('Tag'+ str(Tag))   #   取学科内分数的平均值与标准差
        mean_x = mean.loc[0,'X']
        std_x = mean.loc[1,'X']
        s = data.loc[i,'R1'] #   取评语
        if pd.isnull(s) or type(s) == int:   #   跳过评语为空的
            print('nan')
        else:
            if clusters[count] == 0:   #   若文本分析论文水平高
                if all_score.loc[i,'X1'] <= mean_x:   # 如果论文得分在平均分以下
                    all_score.loc[i,'X1'] = all_score.loc[i,'X1'] + std_x/2#则对总分进行调整
            else:
                if all_score.loc[i,'X1'] >= mean_x:
                    all_score.loc[i,'X1'] = all_score.loc[i,'X1'] - std_x/2
```

根据上述代码对分数进行调整后，需要对 3 位专家修正后的总分进行平均计算：

```
    f_score = all_score.sum(axis = 1)/3
```

得到的 f_score 即为论文的综合得分，将其写入附件 2 中：

```
f_data['综合得分'] = f_score
f_data.to_excel('附件 2.xlsx')
```

### 问题 5 的分析

根据问题 5 的描述，需要对比优秀论文与被问题论文的特征，根据问题 1-4 所计算得到的信息，可以对优秀论文与问题论文在每项得分方面进行对比，由此可得优秀论文与问题论文在每个方面表现的差异。同样可以在原始得分的总分方面进行对比，以及对综合得分进行对比，由此可以得出两种论文在原始打分方面的差异以及结合评语修正后的打分间的差异。再对比两种论文原始得分与综合得分之间的变化，可以看出两种论文是否存在原始打分不准确的情况。

### 问题 5 的求解

首先，读取附件 2 的数据，然后提取其中的问题论文以及优秀论文的数据：

```
# 读取附件 2 数据
data_get = pd.read_excel('Pro_附件 2.xlsx')
#  提取问题论文
lose_paper = data_get.loc[data_get['是否淘汰'] == True]
#  提取优秀论文
percent = 90
percent_val = np.percentile(data_get['综合得分'], percent)
win_paper = data_get.loc[data_get['综合得分'] > percent_val]
```

然后，从优秀论文与问题论文中抽取工学学科的论文数据：

```
lose_paper = lose_paper[lose_paper['Tag'] == 8]
win_paper = win_paper[win_paper['Tag'] == 8]
```

最后，对这些论文中各个子项得分数据以及总分和综合分数数据进行统计均值以及标准差的计算：

```
lose_paper_val = lose_paper[['选题与综述平均分','创新性及论文价值平均分','科研能力与基础知识平均分','论文规范性平均分','论文总分平均分','综合得分']]
win_paper_val = win_paper[['选题与综述平均分','创新性及论文价值平均分','科研能力与基础知识平均分','论文规范性平均分','论文总分平均分','综合得分']]
print(lose_paper_val.sum(axis = 0)/len(lose_paper_val))  #  计算问题论文数据均值
print(lose_paper_val.std(axis = 0))    #  计算问题论文数据标准差
print(win_paper_val.sum(axis = 0)/len(win_paper_val)) #  计算优秀论文数据标准差
print(win_paper_val.std(axis = 0)) #  计算优秀论文数据标准差
```

至此得到计算数据，下面对这些数据进行分析。

### 问题 5 的结果分析

根据计算结果可以看出问题论文在子项与总分全面低于优秀论文。其中，问题论文中平均得分最高的项为“选题与综述”，平均得分最低的项为“科研能力与基础知识”；优秀论文中平均得分最高的项为“创新性及论文价值”，平均得分最低的项为“论文规范性”。从上述结果可以看出，问题论文在选题方面表现较好，但是科研能力与基础知识较差，优秀论文具有很好的创新型与价值，但是论文规范性相对较差。

另外，从得分的标准差来看，问题论文的每一项得分标准差普遍大于优秀论文，表明问题论文的每项得分差异较大，优秀论文的子项普遍高分较为集中。

对比两种论文的原始得分与综合得分，可以看出优秀论文的原始得分与综合得分差异不大，问题论文的综合得分相对原始得分有所提升，由此可以看出问题论文的专家评语中也存

在肯定该论文的点。

表 15　问题论文的各项数据均值与标准差

| | 选题与综述 | 创新性及论文价值 | 科研能力与基础知识 | 论文规范性 | 总分 | 综合得分 |
|---|---|---|---|---|---|---|
| 平均值 | 17.916667 | 16.444444 | 16.315972 | 16.628472 | 67.194444 | 70.955938 |
| 标准差 | 0.997884 | 1.218731 | 1.183619 | 1.294726 | 2.641030 | 2.641030 |

表 16　优秀论文的各项数据均值与标准差

| | 选题与综述 | 创新性及论文价值 | 科研能力与基础知识 | 论文规范性 | 总分 | 综合得分 |
|---|---|---|---|---|---|---|
| 平均值 | 22.448980 | 23.006803 | 22.244898 | 22.027211 | 88.482993 | 88.687700 |
| 标准差 | 0.660741 | 8.545830 | 0.756226 | 0.771563 | 1.640152 | 1.484068 |

## 4 模型评价

根据问题 3 的结果分析可以看出，根据文本分析建立的专家评阅评语模型达到的分类效果一般。由于该模型采用无监督学习，并且评语变化多样，不同评分项目之间在语言上存在耦合，并且对每个评分项目的评价词语间关联性强。如果利用带时序分析的有监督学习可以达到更好的效果，但是该问题并没有给出评阅评语的标签，因此只能采用无监督学习的聚类分析方法。

根据问题 4 与问题 5 的分析可以看出，当使用文本分析建立的专家评语模型用于对综合分数的评价时可以达到良好的效果。首先，利用文本数据聚类分析后的二分类结果与论文的原始得分具有基本的一致性，并且可以较为准确的修正原始得分。由于对评语的好与坏的二分类相对问题 3 简单，并且根据对评语的正负面分析可以得出较为准确的结果。所以该模型可以满足问题需求并且准确。

## 参考文献

[1]王庆华,程兰芳,石婧,胡荣涵.我国研究生学位论文质量的综合评价——基于某“211 工程”高校的论文抽检数据[J].大学教育,2021(05):180-183.
[2]王晶,甘阳,张丹.工科博士学位论文质量多元多维度评价研究[J].高教学刊,2021(01):16-19.
[3]朱晓民,张啾,王祎琪.我国教育硕士专业学位论文质量评价指标体系的构建[J].黑龙江高教研究,2020,38(11):84-89.
[4]黄志开,谢明祥,王振宁,张丽玲,刘克明.工程硕士学位论文质量评价与保障策略讨论——以南昌工程学院为例[J].江西科学,2020,38(05):793-796.
[5]赵文鹤,王斯一,何艺玲,赛江涛.什么影响了博士学位论文质量？——基于某高校 1874 份博士学位论文盲审结果的分析[J].学位与研究生教育,2020(07):70-74.
[6]姜维,文本分析与文本挖掘,北京:科学出版社,2018,pp:1-3.
[7]朱璐,陈世平.融合情感增强与注意力的文本情感分析模型[J/OL].小型微型计算机系统:1-8[2021-05-26].http://kns.cnki.net/kcms/detail/21.1106.TP.20210517.1547.015.html.

# 附录 A PYTHON 源程序

## 问题 1

```
# -*- coding: utf-8 -*-
"""
Created on Sat May 22 14:48:36 2021
@author: MYM
"""
import pandas as pd
import numpy as np
def get_ave(df1, T_num):
    data = df1.values
    ave_data = data.sum(axis = 1) / T_num
    return ave_data
# read xlsx
data = pd.read_excel('附件 1.xlsx')
data_get = pd.read_excel('附件 2.xlsx')
Num = len(data) #  样本数目
T_num = 3 # the number of teacher
percent = 5 #   筛选  末尾  5%
#  获取特定的列
X = data[['X1','X2','X3']]
Xk1 = data[['X11','X21','X31']]
Xk2 = data[['X12','X22','X32']]
Xk3 = data[['X13','X23','X33']]
Xk4 = data[['X14','X24','X34']]
X_ave = get_ave(X, T_num)
Xk1_ave = get_ave(Xk1, T_num)
Xk2_ave = get_ave(Xk2, T_num)
Xk3_ave = get_ave(Xk3, T_num)
Xk4_ave = get_ave(Xk4, T_num)
data_get['选题与综述平均分'] = Xk1_ave
data_get['创新性及论文价值平均分'] = Xk2_ave
data_get['科研能力与基础知识平均分'] = Xk3_ave
data_get['论文规范性平均分'] = Xk4_ave
data_get['论文总分平均分'] = X_ave
X['ave'] = X_ave
X['Tag'] = data['Tag']
lose = []
for i in range(1,14):
    if i == 6 or i == 11:
        print('empty')
    else:
        Tag = X.loc[X['Tag'] == i]
        percent_val = np.percentile(Tag['ave'], percent)
        lose += list(Tag['ave'] < percent_val)
```

```
data_get['是否淘汰'] = lose
data_get.to_excel('附件 2.xlsx', index=None)
```

问题 2

```
# -*- coding: utf-8 -*-
"""
Created on Sat May 22 16:37:22 2021

@author: MYM
"""
import pandas as pd
import numpy as np
def get_ave(df1, T_num):
    data = df1.values
    ave_data = data.sum(axis = 1) / T_num
    return ave_data
# read xlsx
data = pd.read_excel('附件 1.xlsx')
data_get = pd.read_excel('附件 2.xlsx')
Num = len(data) #  样本数目
T_num = 3 # the number of teacher
#  获取总分的列
X = data[['Tag','X1','X2','X3']]
X_ave = get_ave(X, T_num)
X['X_ave'] = X_ave
Sub_dict = dict()
for i in range(1,14):
    Tag = X.loc[X['Tag'] == i]
    Sub_dict.update({'Tag' + str(i):Tag})
Tag_std_dict = dict()
#  每个学科的三个总分的方差均值，与方差方差
Tag_std_mean = pd.DataFrame(index = ['mean','std'], columns = ['Tag1','Tag2','Tag3','Tag4','Tag5','Tag6','Tag7','Tag8','Tag9','Tag10','Tag11','Tag12','Tag13'])
for i in range(1,14):
    Tag_val = Sub_dict.get('Tag' + str(i))[['X1','X2','X3']]
    Tag_std = Tag_val.values.std(axis = 1)
    Tag_std_dict.update({'Tag' + str(i):Tag_std})
    Tag_std_mean.loc['mean','Tag'+str(i)] = Tag_std.mean()
    Tag_std_mean.loc['std','Tag'+str(i)] = Tag_std.std()
Tag_std_mean.to_csv('Total scores.csv')
#  计算每个学科的各个项目的得分，与总分平均分水平
#  获取非评语列
Tag_all = data.drop(columns = ['R1','R2','R3'])
Tag_all_dict = dict()
for i in range(1,14):
    Tag_temp = Tag_all.loc[Tag_all['Tag'] == i]
    Tag_all_dict.update({'Tag' + str(i):Tag_temp})
```

```
Tag_all_mean_dict = dict()
for i in range(1,14):
    Tag_xk1 = Tag_all_dict.get('Tag' + str(i))[['X11','X21','X31']]
    Tag_xk2 = Tag_all_dict.get('Tag' + str(i))[['X12','X22','X32']]
    Tag_xk3 = Tag_all_dict.get('Tag' + str(i))[['X13','X23','X33']]
    Tag_xk4 = Tag_all_dict.get('Tag' + str(i))[['X14','X24','X34']]
    Tag_x = Tag_all_dict.get('Tag' + str(i))[['X1','X2','X3']]
    df1 = pd.DataFrame(index = ['mean','std'], columns = ['Xk1','Xk2','Xk3','Xk4','X'])
    df1['Xk1'] = [Tag_xk1.values.mean(), Tag_xk1.values.std()]
    df1['Xk2'] = [Tag_xk2.values.mean(), Tag_xk2.values.std()]
    df1['Xk3'] = [Tag_xk3.values.mean(), Tag_xk3.values.std()]
    df1['Xk4'] = [Tag_xk4.values.mean(), Tag_xk4.values.std()]
    df1['X'] = [Tag_x.values.mean(), Tag_x.values.std()]
    Tag_all_mean_dict.update({'Tag'+str(i):df1})# 每个学科的分项与总项的均值与方差（不区分打分老师）
for i in range(1,14):
    if i == 6 or i == 11:
        print('skip')
    else:
        ex = Tag_all_mean_dict.get('Tag' + str(i))
        ex.to_csv('Tag' +str(i)+'.csv')
```

## 问题 3

```
# -*- coding: utf-8 -*-
"""
Created on Sun May 23 10:47:17 2021

@author: MYM
"""
import numpy as np
import pandas as pd
import jieba
import sklearn
from sklearn.feature_extraction.text import CountVectorizer
def get_custom_stopwords(stop_words_file):
    with open(stop_words_file, encoding='utf-8')as f:

        stopwords=f.read()
        stopwords_list=stopwords.split('\n')
        custom_stopwords_list=[i for i in stopwords_list]
    return custom_stopwords_list
#加载自定义词语
jieba.load_userdict("C:/Users/MYM/My_python_codes/DSA/user_dict.txt")
#打开文件，文件在桌面上，可以自行修改路径
f1 = open("C:/Users/MYM/My_python_codes/DSA/text_words.txt","r",encoding='GB2312',errors='ignore')
f2=open("C:/Users/MYM/My_python_codes/DSA/text_words_token.txt",'w',encoding='GB2312',errors='ignore')
for line in f1:
    seg_list = jieba.cut(line, cut_all = False)
```

```
        f2.write((" ".join(seg_list)).replace("\t\t\t","\t"))
        #print(w)
f1.close()
f2.close()
#  取需要分词的内容
titles = open("C:/Users/MYM/My_python_codes/DSA/text_words_token.txt", encoding='GB2312', errors='ignore').read().split('\n')
#查看内容，这里是一个 list, list 里面每个原素是分好的标题，查看下长度看有没有错误
#停用词函数调用
stop_words_file= "C:/Users/MYM/My_python_codes/DSA/CNstopwords.txt"
stopwords = get_custom_stopwords(stop_words_file)
#构建词向量，也就是把分好的次去除停词转化成 kmeans 可以接受的形式
from sklearn.feature_extraction.text import CountVectorizer
count_vec=CountVectorizer(stop_words = stopwords)
km_matrix= count_vec.fit_transform(titles)
print(km_matrix.shape)
#查看词向量
# print(km_matrix.toarray())
#开始聚类啦
from sklearn.cluster import KMeans

num_clusters = 8 #聚为八类，可根据需要修改
km = KMeans(n_clusters=num_clusters)
km.fit(km_matrix)
clusters = km.labels_.tolist()
#查看聚类的结果，是 list,这里省略，看看长度是不是和 title 一样就行啦
#len(clusters)
#最后把聚类结果写在一个新的 txt 里面
f3 =open("C:/Users/MYM/My_python_codes/DSA/cluster.txt", 'w',encoding='GB2312',errors='ignore')
for i in clusters:
        f3.write(str(i))
        f3.write("\n")
f3.close()



# -*- coding: utf-8 -*-
"""
Created on Wed May 26 10:07:12 2021

@author: MYM
"""
import numpy as np
import pandas as pd
data = pd.read_excel('附件 1.xlsx')
#  读取聚类结果
clusters = []
```

```
f1 = open("C:/Users/MYM/My_python_codes/DSA/cluster.txt", 'r',encoding='GB2312',errors='ignore')
for line in f1:
    clusters.append(eval(line))
tag_dict = dict()
tag = (1,2,3,4,5,7,8,9,10,12,13)
for i in tag:
    temp = pd.read_csv('Tag'+str(i)+'.csv')
    tag_dict.update({'Tag'+str(i):temp})
num = list()
for i in range(8):
    clusters_temp = [s == i for s in clusters]
    num.append(sum(clusters_temp))
right = 0
for i in range(1246):
    Tag = data.loc[i,'Tag']
    mean = tag_dict.get('Tag'+ str(Tag))
    mean_x1 = mean.loc[0,'Xk1']
    mean_x23 = (mean.loc[0,'Xk2'] + mean.loc[0,'Xk3'])/2
    mean_x4 = mean.loc[0,'Xk4']
    s = data.loc[i,'R1']
    if pd.isnull(s) or type(s) == int:
        print('nan')
    else:
        if data.loc[i,'X11'] >= mean_x1:
            if (data.loc[i,'X12'] + data.loc[i,'X13'])/2 >= mean_x23 :
                if data.loc[i,'X14'] >= mean_x4:
                    if clusters[i] == 7: # 111
                        right+=1
                else:
                    if clusters[i] == 6: # 110
                        right+=1
            else:
                if data.loc[i,'X14'] >= mean_x4:
                    if clusters[i] == 5:# 101
                        right+=1
                else:
                    if clusters[i] == 0: # 100
                        right+=1
        else:
            if (data.loc[i,'X12'] + data.loc[i,'X13'])/2 >= mean_x23 :
                if data.loc[i,'X14'] >= mean_x4:
                    if clusters[i] == 4: # 011
                        right+=1
                else:
                    if clusters[i] == 3: # 010
                        right+=1
```

```
            else:
                if data.loc[i,'X14'] >= mean_x4:
                    if clusters[i] == 2:# 001
                        right+=1
                else:
                    if clusters[i] == 1: # 000
                        right+=1
```

## 问题 4

```
# -*- coding: utf-8 -*-
"""
Created on Wed May 26 12:20:33 2021

@author: MYM
"""

import numpy as np
import pandas as pd
#   每个学科的平均分与标准差
tag_dict = dict()
tag = (1,2,3,4,5,7,8,9,10,12,13)
for i in tag:
    temp = pd.read_csv('Tag' + str(i) + '.csv')
    tag_dict.update({'Tag'+ str(i) : temp})
#   读取聚类结果
clusters = []
f1 = open("C:/Users/MYM/My_python_codes/DSA/cluster_q4.txt", 'r',encoding='GB2312',errors='ignore')
for line in f1:
    clusters.append(eval(line))
# 读取附件 1
data = pd.read_excel('附件 1.xlsx')
all_score = data[['X1','X2','X3']]
count = 0
for i in range(1246):
    Tag = data.loc[i,'Tag']
    mean = tag_dict.get('Tag'+ str(Tag))
    mean_x = mean.loc[0,'X']
    std_x = mean.loc[1,'X']
    s = data.loc[i,'R1']
    if pd.isnull(s) or type(s) == int:
        print('nan')
    else:
        if clusters[count] == 0:
            if all_score.loc[i,'X1'] <= mean_x:
                all_score.loc[i,'X1'] = all_score.loc[i,'X1'] + std_x/2
        else:
            if all_score.loc[i,'X1'] >= mean_x:
```

```
                    all_score.loc[i,'X1'] = all_score.loc[i,'X1'] - std_x/2
for i in range(1246):
    Tag = data.loc[i,'Tag']
    mean = tag_dict.get('Tag'+ str(Tag))
    mean_x = mean.loc[0,'X']
    std_x = mean.loc[1,'X']
    s = data.loc[i,'R2']
    if pd.isnull(s) or type(s) == int:
        print('nan')
    else:
        if clusters[count] == 0:
            if all_score.loc[i,'X2'] <= mean_x:
                all_score.loc[i,'X2'] = all_score.loc[i,'X2'] + std_x/2
        else:
            if all_score.loc[i,'X2'] >= mean_x:
                all_score.loc[i,'X2'] = all_score.loc[i,'X2'] - std_x/2
for i in range(1246):
    Tag = data.loc[i,'Tag']
    mean = tag_dict.get('Tag'+ str(Tag))
    mean_x = mean.loc[0,'X']
    std_x = mean.loc[1,'X']
    s = data.loc[i,'R3']
    if pd.isnull(s) or type(s) == int:
        print('nan')
    else:
        if clusters[count] == 0:
            if all_score.loc[i,'X3'] <= mean_x:
                all_score.loc[i,'X3'] = all_score.loc[i,'X3'] + std_x/2
        else:
            if all_score.loc[i,'X3'] >= mean_x:
                all_score.loc[i,'X3'] = all_score.loc[i,'X3'] - std_x/2
f_score = all_score.sum(axis = 1)/3
f_data = pd.read_excel('附件 2.xlsx')
f_data['综合得分'] = f_score
f_data.to_excel('附件 2.xlsx')
```

## 问题 5

```
# -*- coding: utf-8 -*-
"""
Created on Wed May 26 14:31:00 2021
@author: MYM
"""
import numpy as np
import pandas as pd
#  读取附件 2 数据
data_get = pd.read_excel('Pro_附件 2.xlsx')
```

```
#  提取问题论文
lose_paper = data_get.loc[data_get['是否淘汰'] == True]
#提取优秀论文
percent = 90
percent_val = np.percentile(data_get['综合得分'], percent)
win_paper = data_get.loc[data_get['综合得分'] > percent_val]
lose_paper = lose_paper[lose_paper['Tag'] == 8]
win_paper = win_paper[win_paper['Tag'] == 8]
lose_paper_val = lose_paper[['选题与综述平均分','创新性及论文价值平均分','科研能力与基础知识平均分','论文规范性平均分','论文总分平均分','综合得分']]
win_paper_val = win_paper[['选题与综述平均分','创新性及论文价值平均分','科研能力与基础知识平均分','论文规范性平均分','论文总分平均分','综合得分']]
lose_mean = lose_paper_val.sum(axis = 0)/len(lose_paper_val)
lose_std = lose_paper_val.std(axis = 0)
win_mean = win_paper_val.sum(axis = 0)/len(win_paper_val)
win_std = win_paper_val.std(axis = 0)
print(lose_paper_val.sum(axis = 0)/len(lose_paper_val))
print(lose_paper_val.std(axis = 0))
print(win_paper_val.sum(axis = 0)/len(win_paper_val))
print(win_paper_val.std(axis = 0))
```