浙大城市学院第十四届

大学生数学建模竞赛

2020 年 4 月 15 日－4 月 26 日

参赛队编号\_\_\_\_\_1108\_\_\_\_\_\_\_\_

(报名时网上自动产生的队号)

选题代码（请在括号内打“√”）A( ) B( √ )

所在组别（请在括号内打“√”）1( ) 2( √ ) 3( )

1：一年级组 2：高年级初级组 3：高年级高级组

参赛目的（请在括号内打“√”）志向高远( √ ) 仅混二课分( ) \_

|  |  |  |  |
| --- | --- | --- | --- |
|  | 队员1 | 队员2 | 队员3 |
| 姓名 | 王天宇 | 周鑫杰 | 王逸鹏 |
| 学号 | 31801317 | 31801319 | 31805307 |
| 分院 | 计算 | 计算 | 商学 |
| 专业 | 软件工程 | 软件工程 | 国际商务 |
| 年级班级 | 1801 | 1801 | 1802 |
| 联系电话 | 13566774836 | 13777398667 | 15267062833 |
| email | 31801317  @stu.zucc.edu.cn | 31801319  @stu.zucc.edu.cn | 31805307  @stu.zucc.edu.cn |

教务部

计算机与计算科学学院

数学建模实践基地

**2020浙大城市学院第十四届大学生数学建模竞赛**

**承 诺 书**

我们仔细阅读了浙大城市学院大学生数学建模竞赛的竞赛规则.

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

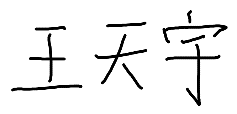
我们知道，抄袭别人的成果是违反竞赛规则的, 如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

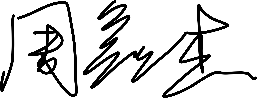
我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们将受到严肃处理。

我们的参赛队报名号为 1108

所属分院（若属不同分院要按照队员编号次序分别填写）：

计算机与计算科学学院 计算机与计算科学学院 商学院

参赛队员 (打印姓名并电子**签名**) ：1. 王天宇 

2. 周鑫杰 

3. 王逸鹏 

日期： 2020 年 4 月 26 日

基于DFM模型的学生消费行为分析

**[摘要]**本文主要通过对已有的国内某高校校园一卡通系统一个月的运行数据进行研究，最终建立了具有较高实用性的学生消费行为分析模型。

**针对问题一，**我们在对已知的数据进行预处理的基础上，对表二的消费记录进行深入的分析，我们的研究方向主要有三个：早午晚各食堂就餐人次占比的可视化，分析每个食堂的三餐就餐情况；工作日与休息日就餐人数与时间的关系的可视化，分析每个时段食堂的就餐高峰期，还有工作日和休息日的高峰期有什么不同；食堂餐费和食堂就餐人数的关系，从统计的角度出发，通过Pearson线性相关系数来评判两者之间的联系，得出了食堂餐费和食堂就餐人数之间存在比较紧密的负相关现象。

第一食堂三餐需要大幅改进，可以考虑降低价格、增加菜式、改善食堂就餐环境等措施；第二和第五食堂在三餐中都比较受欢迎，可以考虑增开窗口来缓解目前就餐压力；第三、第四食堂的早餐比较冷门，建议只开一个窗口或直接取消；工作日的9:40和16:20存在就餐高峰，食堂要提前做好准备；休息日食堂注意减少供应，以免造成浪费，休息日的早餐持续时间较长，食堂要注意进行保温和分批供应。

**针对问题二，**通过对学生的消费记录的进一步探究，提出了DFM模型，主要建立在月平均余额、消费频率和月总消费额三个指标上。通过Elbow Method法对K-Means算法的最佳K值进行了分析，然后采用了优化过后的K-Means++方法，对学生进行聚类，在此过程中发现了存在部分异常数据，比如一个月消费次数个位数的人群，经过了进一步清洗后再此聚类，发现结果有了比较大的提升。最终，分析了每个类型学生的消费特点。

通过聚类绘制三维散点图，将学生消费人群大致分成了6类，对低中高等的消费人群进行了初步的分类，简要推测了不同消费行为对应的消费人群。我们定义的DFM模型能初步根据消费特点对学生群体进行分类。

**针对问题三，**在问题二的模型基础上，我们选择了其中的需要补助的可能性最大的一类人进行了基于AHP的打分，最终得到的区间为1-5分，分数越低说明消费水平比较低，最后根据分数划分群体后，提出梯度补助的策略。

对分值1的人群为重点补助人群，分值2和3的人群为基本补助人群，分值为4的人群可以少量补助，分值为5的人群极有可能是上一个模型的漏网之鱼，不补助。综上可知，我们建立的结合AHP分析的DFM模型具有较高的实际价值。

**关键字**:DFM模型 K-Means++聚类 层次分析法

1. 问题重述

1.1问题的背景

校园一卡通是集身份认证、金融消费、数据共享等多项功能于一体的信息集成系统。在为师生提供优质、高效信息化服务的同时，系统自身也积累了大量的历史记录，其中蕴含着学生的消费行为以及学校食堂等各部门的运行状况等信息。很多高校基于校园一卡通系统进行“智慧校园”的相关建设，例如《扬子晚报》2016年1月27日的报道：《南理工给贫困生“暖心饭卡补助”》。 不用申请，不用审核，饭卡上竟然能悄悄多出几百元……记者昨天从南京理工大学独家了解到，南理工教育基金会正式启动了“暖心饭卡”项目，针对特困生的温饱问题进行“精准援助”。 项目专门针对贫困本科生的“温饱问题”进行援助。在学校一卡通中心，教育基金会的工作人员找来了全校一万六千余名在校本科生9月中旬到11月中旬的刷卡记录，对所有的记录进行了大数据分析。最终圈定了500余名“准援助对象”。 南理工教育基金会将拿出“种子基金”100万元作为启动资金，根据每位贫困学生的不同情况确定具体的补助金额，然后将这些钱“悄无声息”的打入学生的饭卡中，保证困难学生能够吃饱饭。

——《扬子晚报》2016年1月27日：南理工给贫困生“暖心饭卡补助”

1.2问题的相关信息

附件包含3张数据表，分别为data1.csv、data2.csv、data3.csv。可知如下数据：

表1为学生ID表，给出4341条记录，包含字段为序号、校园卡号、性别、专业名称和门禁卡号。

表2为消费记录表，给出519367条记录，包含字段为流水号、校园卡号、校园卡编号、消费时间、消费金额、存储金额、余额、消费次数、消费类型、消费项目的编码、消费项目的序列号、消费操作的编号、操作编码和消费地点。

表3为门禁记录表，给出43156条记录，包含字段为序号、门禁卡号、进出时间、进出地点、是否通过和描述。

1.3需解决的问题

本赛题提供国内某高校校园一卡通系统一个月的运行数据，请使用附件数据分析和建模，分析学生在校园内的学习生活行为，为改进学校服务并为相关部门的决策提供信息支持。

问题一：分析学生的消费行为和食堂的运营状况，为食堂运营提供建议。

问题二：根据学生的整体校园消费行为，选择合适的特征，构建模型，分析每一类学生群体的消费特点。

问题三：构建学生消费细分模型，为学校判定学生的经济状况提供参考意见。

1. 模型假设

（1）假设附件中所给的学生信息和消费记录数据都是真实可靠的；

（2）假设所有记录的消费是合理的，不存在恶意消费；

（3）假设一条消费记录代表一次消费，即不存在同一次消费产生多条记录的情况。

（4）假设一个月消费次数少于30次的都是不正常的消费现象。

（5）假设在月平均余额、消费频率和月总消费额三个指标中月总消费额对于评价学生的经济能力最重要，其次是消费频率，最后才是月平均余额。

（其余假设在文中另作说明）

1. 符号说明

|  |  |
| --- | --- |
| **符号** | **说明** |
|  | 月平均余额 |
|  | 消费频率 |
|  | 月消费总额 |
|  | DFM指标值 |
|  | 指标最大值 |
|  | 指标最小值 |
|  | 标准化的DFM指标值 |
|  | 聚类类别  聚类中心  距离平方和  总距离平方和 |
|  | 成对比较矩阵 |
|  | 判断矩阵 |
|  | 最大特征根 |
|  | 特征向量 |
|  | 一致性指标 |
|  | 平均随机一致性指标 |
|  | 一致性检验系数 |

（其余符号在文中另作说明）

1. 数据预处理

4.1处理数据类型

因为原Date字段是以<年/月/日 时:分>的形式给出的，我们为了统计不同天和不同时段的数据，所以将data2.csv中Date字段拆分为时间Time和日期Data两个字段。

4.2 处理重复数据

删除data2.csv中的重复数据项。考虑CardNo和CardCount，如果一张卡消费过，但是消费次数是一样的，说明记录是重复的，需要删除。

4.3 处理异常值和空值

data2.csv中，探索TermSerNo字段，发现基本为空值，但是有小部分数据是非空值的，其对应的时间却都在零点，因此判断此为异常数据，删除此字段不为空的数据；探索conOperNo字段，发现也有小部分的非空数据，其对应的类型为非消费类型，主题为消费，因此删除此字段不为空的数据。

4.4 筛选消费数据

data2.csv中，虽然在去空值的时候，有对一部分非消费类型进行处理，但是还有部分非消费类型没有被去除，筛选Type字段为“消费”的记录。

4.5 删除多余字段

data2.csv中，TermSerNo、conOperNo、FundMoney和Type这几个字段在以上处理中，清洗了部分数据，剩下的内容对后续分析没有作用，因此对这几个字段进行删除。Index和PeoNo字段因为是编号，没有实际的作用，这里也对其进行删除处理。

4.6 合并表格

之前处理后的data2.csv导出新表consume.csv，和data1.csv以CardNo字段为主键合并，导出新表grade18.csv。data3.csv在本题没有用到，因此这里就不做处理。

1. 问题一的分析与求解

5.1 问题分析

对于学生的消费行为和食堂的运营状况分析，我们主要从三个角度进行分析：早午晚各食堂就餐人次占比、工作日与休息日就餐时间曲线和学生消费的Pearson线性相关系数。因此我们需要对数据进行划分，做进一步的数据处理和提取，然后开始分析。

5.1.1 早午晚各食堂就餐人次占比

将时间分为早餐、午餐和晚餐三个时间段，选出消费地点Dept在包含“食堂”两个字的消费记录，然后把记录分到对应的时间段，如果不在时间段说明消费的不是正餐，不在记录分析的范围内。然后进行三餐各食堂就餐人次的饼图绘制。假设三餐的运营时间段如表5-1所示：

**表5-1 三餐运营时段表**

|  |  |
| --- | --- |
| **运营时段** | **时间** |
| 早餐 | 06:00–09:30 |
| 午餐 | 10:30–14:30 |
| 晚餐 | 16:00–22:00 |

5.1.2 工作日与休息日就餐人数与时间关系

采用工作日的数据处理包，按工作日与休息日对消费记录进行划分，过十分钟为一个时间段进行流量的统计，若某个时间段没有流量，则用0补充缺失时段。因为考虑到食堂在六点以前基本是不开门的，所以这里选取从06:00-24:00的数据。然后以时间为横轴，平均人数位纵轴绘制折线图。

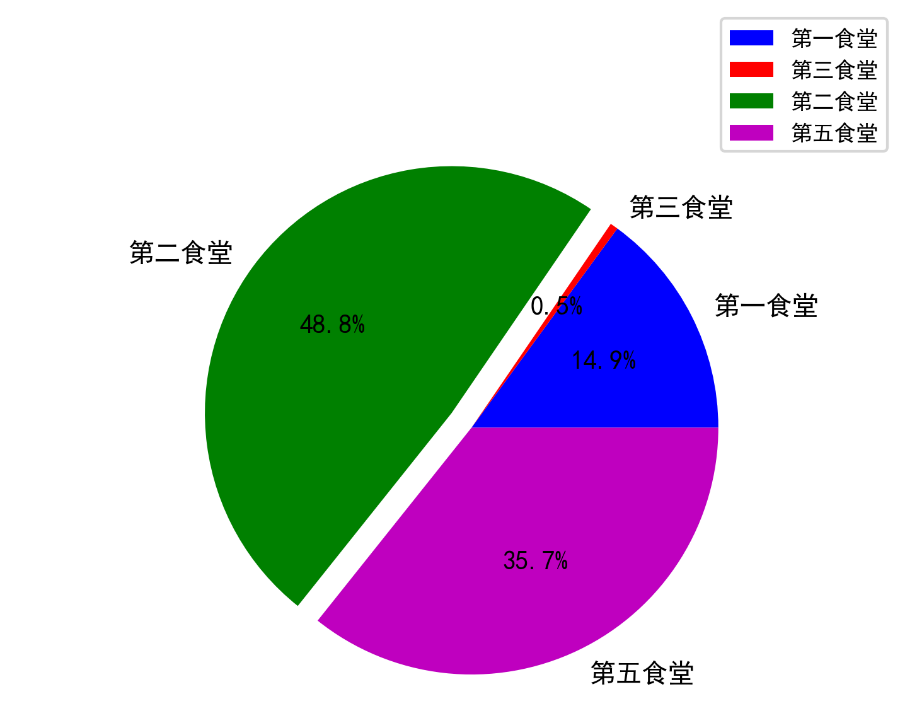
5.1.3 学生消费的Pearson线性相关系数

在5.1.1分析的基础之上，取出消费记录中的消费地点Dept、早/午/晚餐Meal、平均消费额AvgMoney和消费次数Count。利用Pearson积差相关系数计算平均消费额AvgMoney和消费次数Count两个字段的相似程度。皮尔森相关系数计算公式如下：

5.2 问题求解

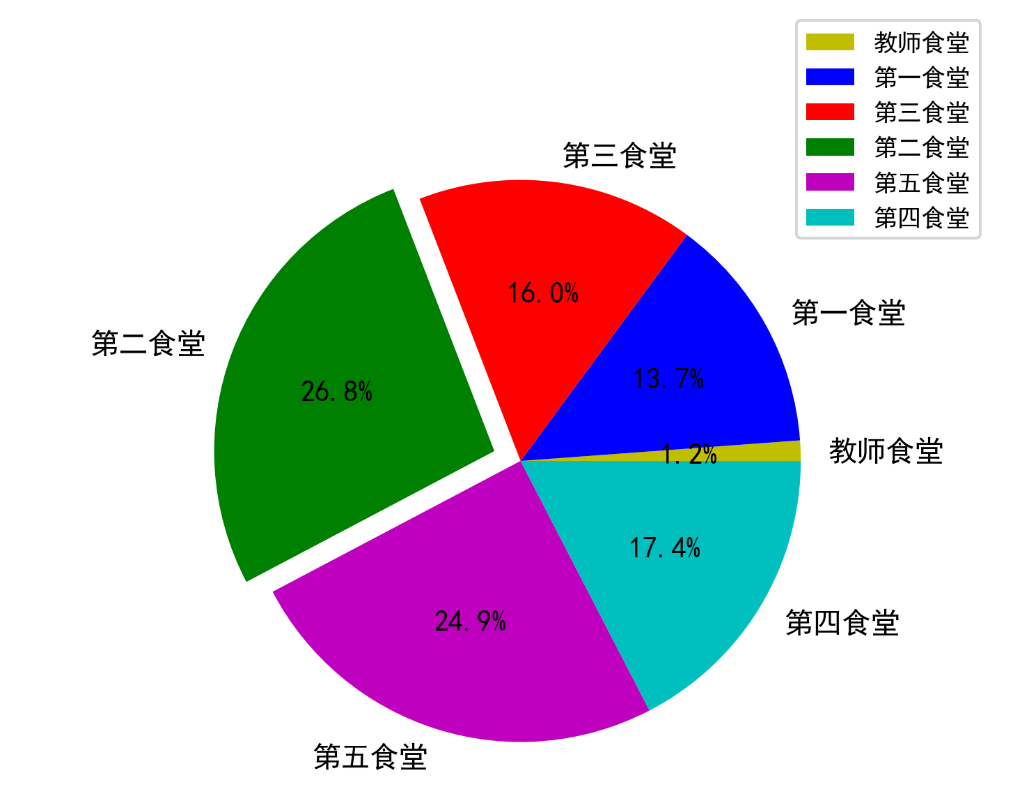
5.2.1 早午晚各食堂就餐人次占比饼图绘制

绘制三餐就餐人数饼图，具体如下：



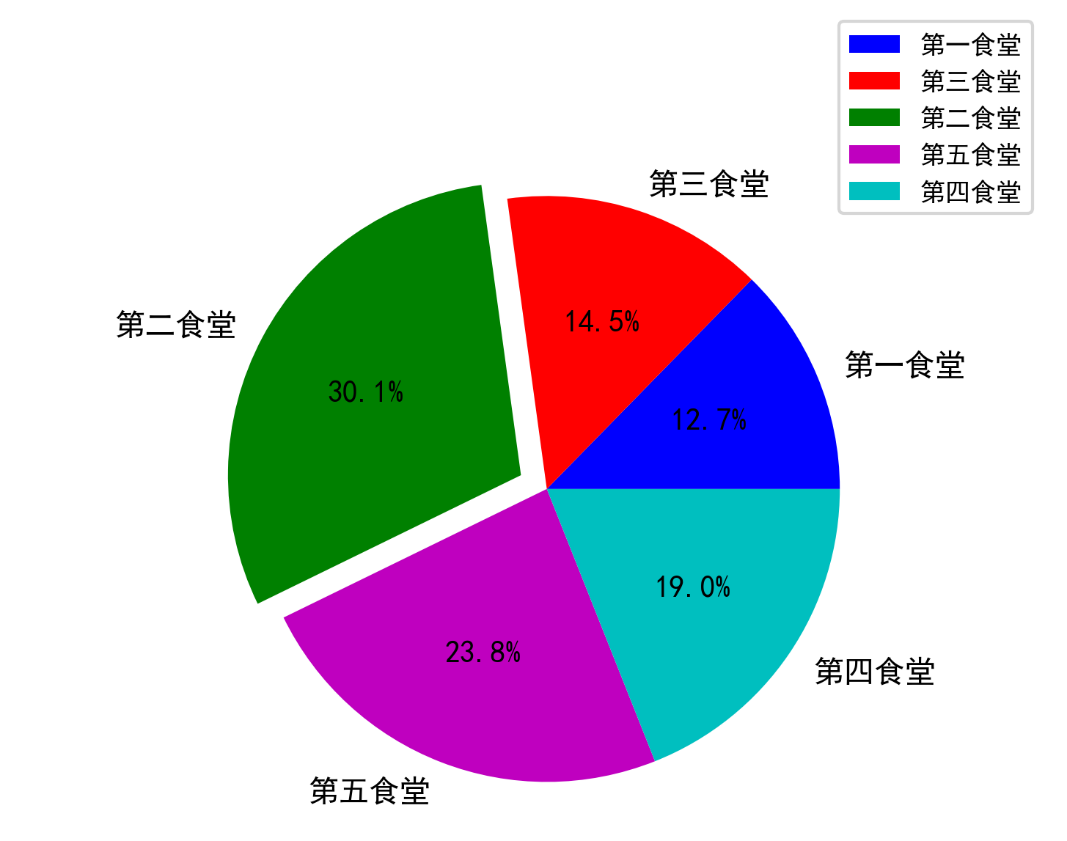
**图5-1(1)早餐就餐人数对比图**

从早餐的角度出发,在第二食堂就餐的人数最多,达到48.8%,几乎占了“半壁江山”,说明第二食堂的早饭最受欢迎,其次是第五食堂和第一食堂,分别达到35.7%和14.9%.两者相加也达到了50.6%,而第三和第四食堂的占比几乎为零。



**图5-1(2)午餐就餐人数对比图**

午餐除教师食堂以外,各食堂的人数占比相对比较均匀。第二食堂和第五食堂的就餐人数相对比较多,分别占比26.8%和24.9%,其余第四、第三食堂分别占比17.4%、16%，第一食堂相对比较落后，只有13.7%的就餐人数。教师食堂可能是因为主要提供教师就餐导致学生很少前往教师食堂就餐，情况比较特殊，只占1.2%。



**图5-1(3)晚餐就餐人数对比图**

晚餐的情况和午餐比较类似，但是晚餐比午餐的差距略大。同样是第二食堂就餐人数最多，占比30.1%，其次是第五食堂和第四食堂，分别占有23.8%和19%的市场份额，最后是第三食堂和第一食堂的情况相对不是非常乐观，只分别占比14.5%和12.7%。

基于以上分析，我们对食堂提出以下几点建议：

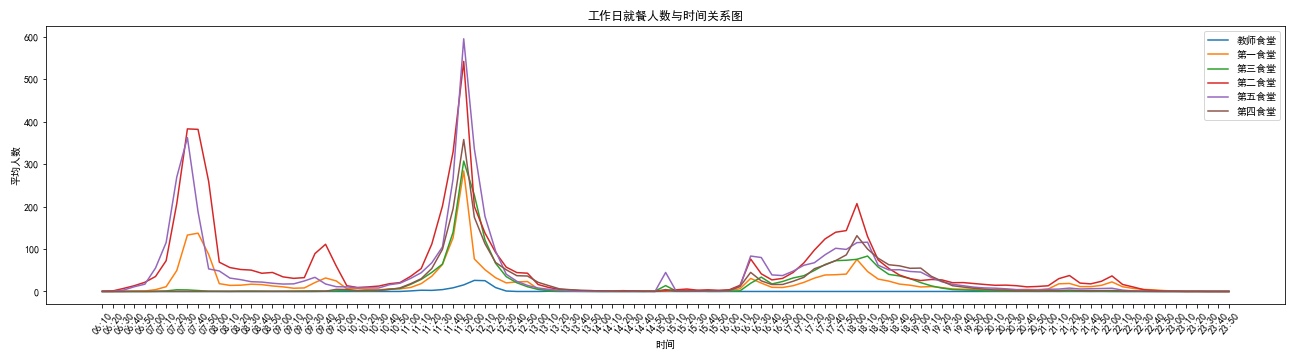
（1）第三、第四食堂的早餐几乎无市场份额，可以考虑只开一个窗口或直接取消早餐供应，节省成本；

（2）第一食堂在三餐中都逊色于其他食堂，可以考虑降低价格、增加菜式、改善食堂就餐环境等措施来吸引学生，增加市场竞争力；

（3）教师食堂只在午餐中有市场份额且份额较少，教师食堂可以只提供午餐且供应量满足教师需求即可；

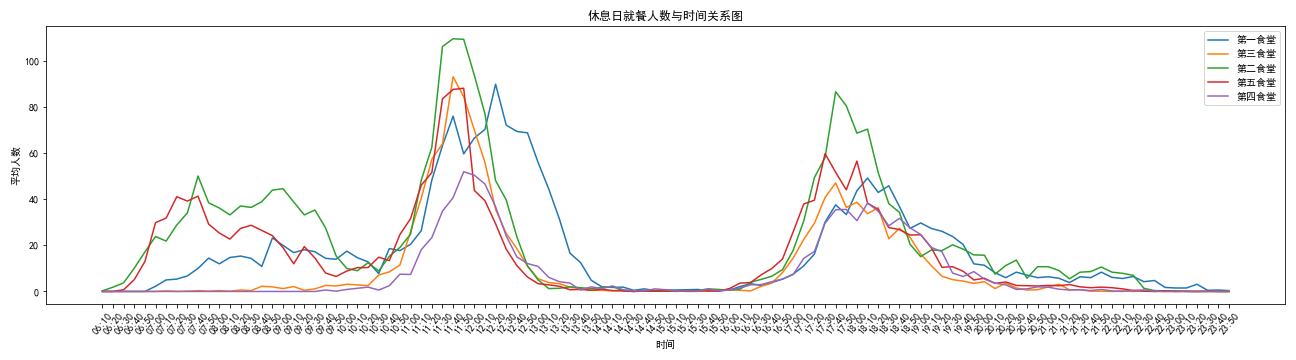
（4）第二和第五食堂在三餐中都比较受欢迎，就餐人数多，可以考虑扩大规模，增开窗口来缓解目前就餐压力；继续改善就餐环境，丰富菜品，吸引更多学生就餐。

5.2.2 工作日与休息日就餐人数与时间曲线绘制



**图5-2(1) 工作日就餐人数与时间关系图**

工作日的就餐时间曲线图显示出三个就餐高峰，分别对应早中晚三餐的时间且每个食堂都呈现相同的趋势。早餐高峰时段大致开始于7:20，结束于7:50，在7:35左右达到峰值；午餐高峰峰值出现于11：50左右，峰值前后的20分钟就餐人数较多；晚餐的高峰相对比较平缓，前后持续时间约达1.5小时，最高峰出现在18：00左右。除了三个大高峰以外，需要注意的是，图中在9:40，第二食堂会出现一个比较明显的小高峰，而16:20，第二和第五食堂也都出现比较明显的小高峰。



**图5-2(2) 休息日就餐人数与时间关系图**

休息日与工作日不同在于，休息日的早餐持续时间非常长，同时峰值也相对比较小，而且休息日不同食堂的就餐高峰存在一些差异。早餐就餐人数最多的三个食堂中，第一食堂和第五食堂的峰值分别出现在8:50和7:30，而第二食堂从7:40到9:10几乎都是高峰时段；午餐各个食堂的高峰都在11:30-12:00之间，但是第一食堂会在12:30左右迎来第二波高峰；晚餐的高峰相对午餐而言比较平缓，高峰时段大致位于17:00到18:30之间.总的来说，工作日和休息日相比，从就餐时间分析，午餐和晚餐高峰时间的差异并不明显，但是休息日的早餐高峰持续时间较长；从就餐人数分析，休息日的三餐就餐人数较工作日都明显减少，可能的原因在于休息日大家会选择外出就餐，而不考虑食堂。

基于以上分析，我们对食堂提出以下几点建议：

（1）对于以上数据反映出的三餐就餐高峰，各个食堂根据各自时间提前做好准备，特别是工作日的9:40，第二食堂要做好供应量和食品保温两方面工作，而在16:20晚餐小高峰，其中第二食堂和第五食堂要做好提前提供晚餐的准备；

（2）休息日的三餐就餐人数较工作日大幅减少，食堂注意减少供应，以免造成浪费；

（3）休息日的早餐持续时间较长，食堂要注意对食物进行保温或者分批持续供应。

5.2.3 学生消费的Pearson线性相关系数分析

**表5-2 Pearson线性相关系数表**

|  |  |  |
| --- | --- | --- |
|  | **时间段** | **线性相关系数** |
| 0 | 午餐 | -0.715571 |
| 1 | 早餐 | -0.903313 |
| 2 | 晚餐 | -0.346169 |

根据学生消费的Pearson线性相关系数表，我们发现三餐的平均消费金额与消费次数均呈现负的线性相关，且早餐的相关性最大，晚餐的相关性最小。

基于以上分析，我们提出以下相关建议：

（1）第一食堂的早餐消费热度远低于第二和第五食堂，可考虑适当降价来增加市场竞争力，从而提高就餐人数。

1. 问题二的模型建立与求解

6.1 问题分析

针对学生的整体校园消费行为，就需要考虑到学生的存储情况与消费情况，因此需要建立一个能有效刻画学生整体消费行为特征的模型。在众多的用户消费模型中，RFM模型是衡量客户消费行为的重要工具。但是考虑到本文的研究对象为学校的学生，最近一次消费这个指标在学校基本是天天消费，所以没有太大的参考价值。因为学生的消费频繁，可以增加一个储蓄情况的指标。故我们引入DFM模型，对学生的消费行为进行刻画，并基于DFM模型对学生进行K-Means++聚类。

6.2 模型建立

在RFM模型的基础上，对其进行改造，得到DFM模型，通过K-means聚类法对学生群体进行分类。

6.2.1 DFM模型介绍

在校园活动中，每个学生的消费行为不同，需要寻找一种工具来判定学生的经济状况。本文为学生的消费情况建立一个能够刻画学生经济状况的DFM数学模型。它以会员关系领域广泛用来衡量会员价值和描述会员行为的RFM模型为基础，拓展优化而成。DFM模型有四个指标，指标含义如下：

* D（Deposit）

D表示学生月平均余额。余额决定着学生能否继续进行消费，在评价学生的整体消费行为中是不可缺少的一项指标。某一时刻的余额高低只能说明某一时刻的经济状况，而学生的消费周期基本上是以月作为单位的，平均一个月的余额，可以大致掌握学生的经济状况。D指标主要刻画了学生的消费基础。

* F（Frequency）

F表示学生在一个月内的消费频率，消费频率越高的学生，一般认为经济状况不会太差消费频率低的学生会受到经济状况的影响而限制消费。F指标主要刻画了学生的消费热度。

* M（Monetary）

M表示学生当月消费的总额，消费总额是所有消费行为的支柱，直接反映了学生的消费能力。M指标主要刻画了学生的消费能力。

DFM模型以上述三个指标为替代变量，通过指标标准化来进行均值聚类分析，将学生分成不同的类别。

6.2.2 D、F、M值的标准化

对各属性进行规格化变换，规格化变换又称为极差正规比变换，是从数据矩阵中的每一个变量最大值和最小值，并用最大值减去最小值得出极差。然后用每一个原始数据减去该变量中的最小值，再除以极差，即得到规格化数据，标准化公式为：

其中,是标准化的D、F、M值，是原值， 和 分别是该指标的最大值和最小值。由于 D、F、M指标的影响是正向的，所以该公式适用。

6.2.3 K-Means的改进算法K-Means++

在标准化后的D、F、M值上，我们采用K-Means聚类方法对会员群体进行聚类分析，具体步骤如下所示[1]：

对于给定的一个包含n个d维数据点的数据集，以及要生成的数据子集的数目K，K-Means聚类算法将数据对象组织为K个划分。每个划分代表一个类，每个类有一个类别中心。选取欧氏距离作为相似性和距离判断准则，计算该类内各点到聚类中心的距离平方和：

聚类目标是使各类总的距离平方和最小。

其中，

显然，根据最小二乘法和拉格朗日原理，聚类中心应该取为类别类各数据点的平均值。K-Means 聚类算法从一个初始的 K 类别划分开始，然后将各数据点指派到各 类别中，以减小总距离平方和。因为K-means聚类算法中总的距离平方和随着类别个数 K 的增加而趋向于减小（当 K=n 时，J(C)=0）。因此,总的距离平方和只能在某个确定的类别个数K下,取得最小值。然而K个初始化的质心的位置选择对最后的聚类结果和运行时间都有很大的影响，因此需要选择合适的K个质心。如果仅仅是完全随机的选择，有可能导致算法收敛很慢。

K-Means++算法就是对K-Means随机初始化质心的方法的优化。具体步骤如下：

Step1：从输入的数据点集合中随机选择一个点作为第一个聚类中心；

Step2：对于数据集中的每一个点，计算它与已选择的聚类中心中最近聚类中心的距离；

Step3：选择一个新的数据点作为新的聚类中心，选择的原则是：较大的点，被选取作为聚类中心的概率较大；

Step4：重复Step2和Step3直到选择出K个聚类质心；

Step5：利用这K个质心作为初始化质心去运行标准的K-Means算法。

6.3 问题求解

6.3.1 创建特征数据集

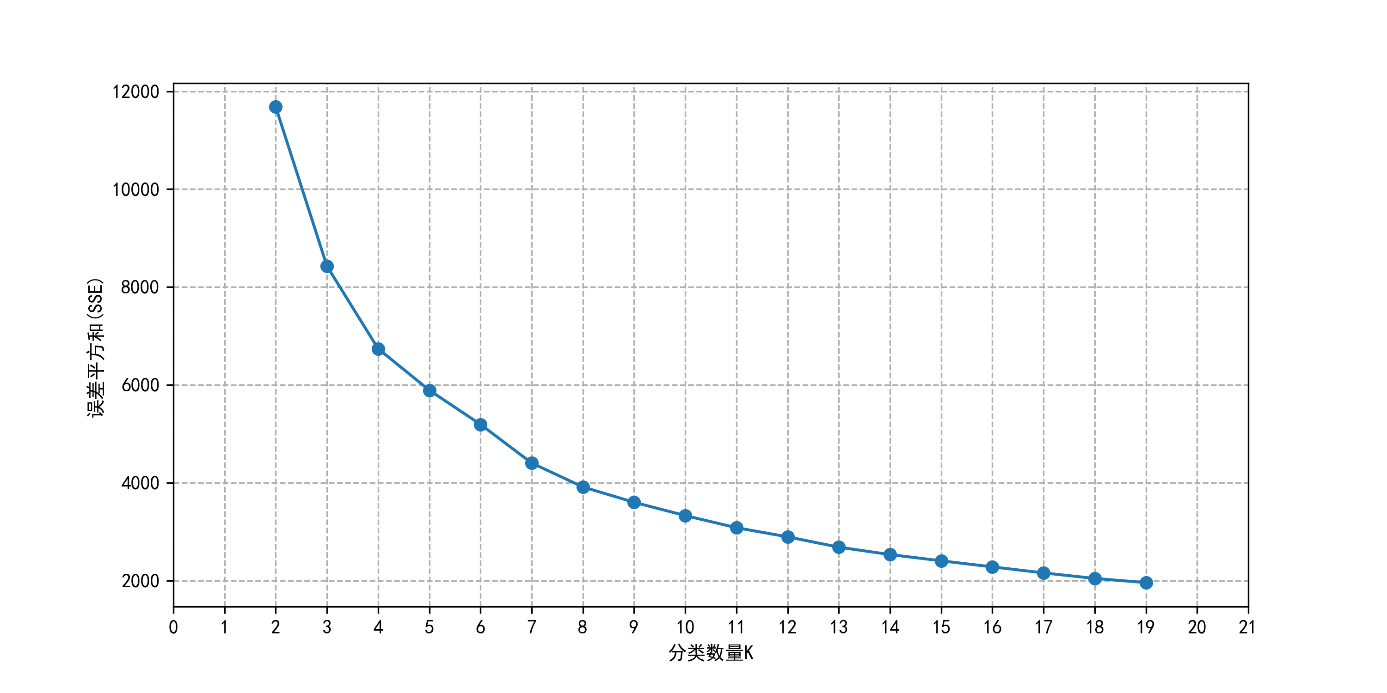
提取消费记录中非公共消费的记录，比如水电费之类的消费。这里选取了食堂和超市的消费，因为大学生的大部分个人消费地点都是在这两类地点。选取卡号、月平均余额、月总消费额和月消费频率四个字段，并将消费频率低于30次的数据作为异常数据删除。

**表6-1 消费特征表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | CardNo | AvgSurplus | TotalConsume | Freq |
| 0 | 9 | 52.56 | 110.90 | 45 |
| 4 | 58 | 179.18 | 119.90 | 37 |
| 12 | 4462 | 44.12 | 190.10 | 54 |
| … | … | … | … | … |
| 8334 | 186148 | 65.94 | 114.90 | 47 |
| 8335 | 186149 | 56.83 | 119.50 | 36 |
| 8336 | 186150 | 79.17 | 275.30 | 49 |

6.3.2 通过Elbow Method求K-Means++最佳k值

当k小于真实聚类数时，由于k的增大会大幅增加每个簇的聚合程度，故SSE的下降幅度会很大。而当k到达真实聚类数时，再增加k所得到的聚合程度回报会迅速变小，所以 SSE 的下降幅度会骤减，随着k值的继续增大而趋于平缓。

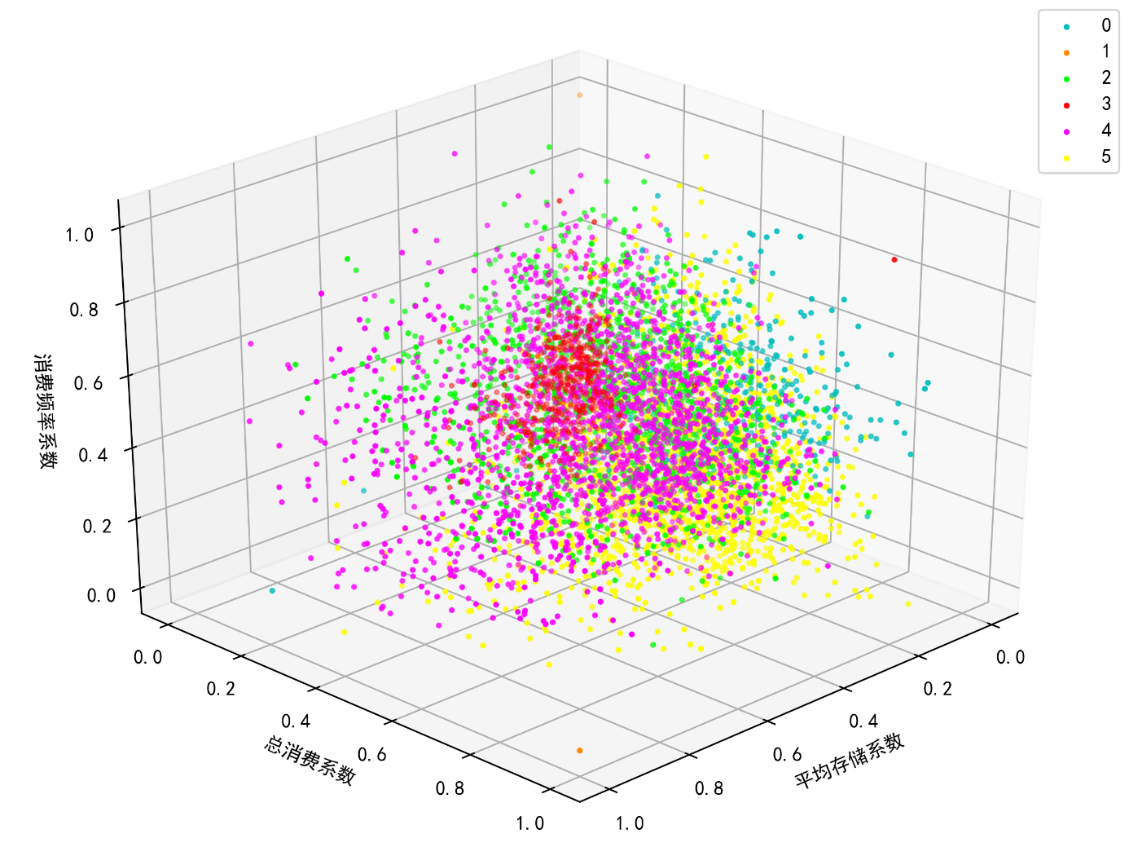


**图6-1 Elbow Method的SSE-K关系折线图**

将每个簇的质点与簇内样本点的平方距离误差和称为畸变程度，那么，对于一个簇，它的畸变程度越低，代表簇内成员越紧密，畸变程度越高，代表簇内结构越松散。畸变程度会随着类别的增加而降低，但对于有一定区分度的数据，在达到某个临界点时畸变程度会得到极大改善，之后缓慢下降，这个临界点就可以考虑为聚类性能较好的点。通过上图分析，最佳的K值在6左右。

6.3.3 数据标准化并聚类可视化

根据上述的分析，创建K-means++模型，将数据标准化，并以三维散点图的形式可视化模型的效果。



**图6-2 K-Means++聚类模型散点图**

如图6-2所示，根据三维散点图，我们对0-5类学生的消费特征进行分析与总结，总结如下：

（1）0类学生呈现出主要的特征是平均存储少，但是总消费和消费频率都较高，可初步判定为高消费能力人群；

（2）1类学生在图中占少数，可能是部分极端消费的人群；

（3）2类学生分布较分散，三项指标都在平均左右，因此推测是中低等消费人群；

（4）3类学生分布比较密集，总体呈现三低的特征，即平均存储少、总消费少、消费频率低，可初步判定为低消费能力人群；

（5）4类学生呈现出比较高的平均存储和较高的消费频率，总消费也较高，分布比较分散，可以初步判定为广大的中等消费人群；

（6）5类学生最主要特征为总消费非常多，消费频率比较低。

1. 问题三的模型建立与求解

7.1 问题分析

为了判断学生的经济状况，问题二使用的聚类方法虽然可以大致确定一个学生的消费水平，但是误差比较大，而且无法得出更多有用的信息，我们希望可以通过数据，来给学生下更加精细的划分，我们就以上题中选出的那最有可能包含需要补助学生的信息的数据集为输入数据，在DFM模型的基础上使用AHP算法进行打分，为学校判定经济状况提供更加可靠的数据。

7.2 模型建立

层次分析法(Analytic Hierarchy Process, AHP)是由美国运筹学家、匹兹堡大学教授Saaty T.L. 提出的一种将决策有关的元素分解成目标、准则、方案等层次，在此基础上进行定量和定性分析的决策方法。常被运用于多目标、多准则、多要素、多层次的非结构化的复杂决策问题上，特别是战略决策问题，可以较好地解决多要素相互关联、相互制约的复杂系统的评价。

7.2.1 定义成对比较矩阵

假设一共有n个因素，要比较每个因素的优劣，引入成对比较矩阵的概念。定义矩阵:

成对比较矩阵的特点:

AHP算法的发明者T. L. Saaty提供了一个关于的取值建议[2]，具体取值如表7-1所示。

表7-1 AHP算法中关于的取值建议

|  |  |  |
| --- | --- | --- |
|  | 相对重要程度 | 说明 |
| 1 | 等同重要 | 两者对目标的贡献相同 |
| 3 | 略为重要 | 根据经验一个比另一个评价稍微有利 |
| 5 | 基本重要 | 根据经验一个比另一个评价更为有利 |
| 7 | 确实重要 | 根据经验一个比另一个评价有利，且实践中证明 |
| 9 | 绝对重要 | 重要程度明显 |
| (2, 4, 6, 8) | 两相邻程度的中间值 | 按照需要折中使用 |

7.2.2 层次排序算法

层次单排序就是把本层所有各元素对上一层来说，排列出评价顺序，核心在于计算最大特征向量，常用的有和积法、方根法。

层次总排序是利用层次单排序的计算结果，进一步综合出对更上一层次的优劣顺序。计算方法和层次单排序一样，所以我们下面只介绍一遍算法的流程。

层次单排序和层次总排序两者加在一起就是层次排序算法。以和积法为例，介绍一下算法的流程[3]:

（1）求解特征向量近似解

根据我们上一步所得到的成对比较矩阵，每一列代表的就是一个判断标准，所以先把每一列的数据进行归一化：

将每一列经过了归一化处理后的判断矩阵按行相加：

对向量W再进行归一化处理：

转置之后得到一个特征向量的近似解：

（2）计算判断矩阵(成对比较矩阵)的最大特征根：

虽然已经有了一套计算方法，但是问题是我们并不能认为输入的判断矩阵全都能符合成对比较矩阵的定义，因此，我们引入一致性检验的概念。

7.2.3 一致性检验

虽然从理论上来说，只要矩阵A严格满足成对比较矩阵的特点，那么就有:

但是实际上要让所有的参与到算法中来的矩阵都满足这样的条件未免太严格了，所以我们可以允许矩阵不完全满足成比较矩阵的特点，但是也不是无底线的，需要一个指标来衡量什么样的矩阵是合法的。

计算一致性的指标 : ，其中的计算方法就和上题一样。针对层次单排序，给出随机一致性比例的定义如下:

其中可以查阅相关表格，部分的表格如下:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **阶数** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **RI** | 0 | 0 | 0.52 | 0.89 | 1.12 | 1.26 | 1.36 | 1.41 | 1.46 |

而层次总排序，我们也可以得到一个公式:

7.3问题求解

在我们上一题的DFM模型的基础上，我们认为M的优先级最高，其次是F，最后才是D，那么，我们就可以构建这样一个成对比较矩阵: 。模型分为准则层、方案层、目标层。准则层的分析比较简单，我们可以来手推一下。方案层和目标层的求解则交给程序来做就好了。

因为比较简单，所以一眼就可以看出它符合成对比较矩阵的定义，接下来我们根据算法来求出它的最大特征根。在此之前，先求特征向量:

检验一下结果，特征向量相加等于1，可以认为这就是我们的特征向量。

接下来判断矩阵一致性:

通过了一致性检验。

到此为止，准则层的验证已经做好了，接下来我们需要对方案层求解， 我们通过第二题的分析，我们选择相对来说三项指标都比较小的一类数据来分析，查看了一下，大概有2400+的候选人。把他们和他们的卫星数据一起输入到程序中，程序会对他们进行自动打分，当然，第一步是先生成方案，在生成方案的过程中去选择最优的方案。

我们对最后的数据进行按照分数升序排序，得到部分结果如下图所示:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CardNo | AvgSurplus | TotalConsume | Freq | Grade |
| 182989 | 23.87 | 21.0 | 41 | 1 |
| 163632 | 23.98 | 49.0 | 30 | 1 |
| 185568 | 28.45 | 57.8 | 30 | 1 |
| 184210 | 20.48 | 60.2 | 32 | 1 |
| 181878 | 23.15 | 62.3 | 33 | 1 |
| … | … | … | … | … |
| 182535 | 72.50 | 298.9 | 53 | 5 |
| 183062 | 75.95 | 299.5 | 52 | 5 |
| 180009 | 89.90 | 304.8 | 46 | 5 |
| 172455 | 57.17 | 327.1 | 46 | 5 |
| 186112 | 88.23 | 295.9 | 52 | 5 |

注: 2428 rows 4 columns

将所得结果的各分数人数绘制成饼图，如图7-1所示。:

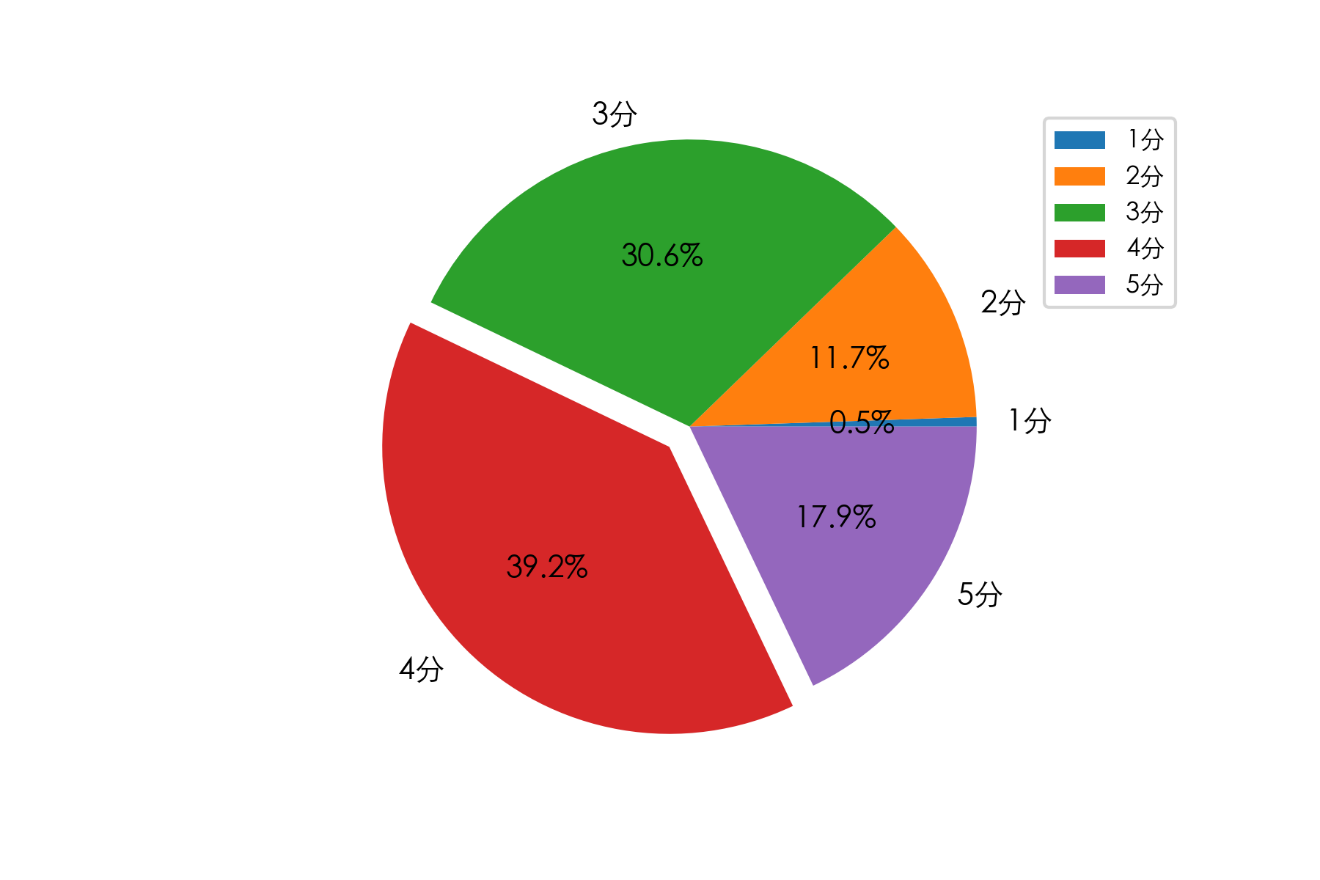


图7-1 各分数人数占比图

根据所得结果，我们对学校判定学生的经济状况并给予补助提供以下建议:

首先在总共大概有8000+的学生，其中相对贫困的人大概有2000+，在这2000+的人当中，根据评分结果， 5分的可以认为是K-Means模型的失误，被划分到了这个区间中，另外四分的比例也比较高，如果也算到最后的补助人口中，补助的比例过高，所以一个比较合理的建议是，选择评分为1、2、3分的进行补助。

梯度补助策略，对于不同分值的人，可以调整金额，比如，1分的人比较少，但是他们是最困难的学生，可以多给他们补助，而2分和3分的人比较多，相对来说他们也没1分的人更加需要这笔补助，所以可以相对来说拿的少一点。

1. 模型的评价及推广

8.1 模型的优点

（1）K-Means算法在分类中算是收敛速度比较快的，外加上我们使用的K-Means++由于对初始值做了优化，导致收敛速度更加快。

（2）无论是第二题的K-Means和第三题的AHP都相当简单，易于解释和理解。

（3）AHP每个层次中的每个因素对结果的影响程度都是量化的，非常清晰明确。

8.2 模型的缺点

（1）K-Means对噪声(异常值)比较铭感，所以我们花了很多时间去研究如何洗数据。

（2）K-Means需要提前确定K值，这非常麻烦，有时候还要在大篇幅的程序代码中寻找需要修改的地方

（3）K-Means的输出并不是很稳定，所以每次跑下来的结果都会有一点点差异。

（4）AHP有一个很致命的缺点，我们必须给定一个评价的策略才能让算法去求最优的结果，这导致了一旦策略不够优秀，AHP的结果不准确。也就是受到主观的影响很重。

（5）AHP中一旦一致性比较失败了，那么程序就会失败。

参考文献

[1]刘建平. K-Means聚类算法原理, cnblogs.com/pinard/p/6164214.html,2020.04

[2]刘敬辉.基于FTA-AHP的铁路安全风险综合评估方法[J].中国铁道科学,2017,38(2):140.DOI:10.3969/j.issn.1001-4632.2017.02.21.

[3]闫敏慧,姚秀萍,王蕾,张金峰.用层次分析法确定气象服务评价指标权重[J].应用气象学报,2014,25(04):470-475.

附录

**文件: init.py**

# %%

"""一些初始化的工作"""

import pandas as pd

import time

import os

# 创建必须的文件夹

if not os.path.exists('data'):

os.makedirs('data') # data文件夹下放置所有的csv文件

if not os.path.exists('img'):

os.makedirs('img') # img文件夹下放置所有的png文件

# 读入数据

amount = pd.read\_csv(os.path.join('data', 'data2.csv'), encoding='gb2312')

student = pd.read\_csv(os.path.join('data', 'data1.csv'), encoding='gb2312')

# 日期分割成两个字段

amount['Time'] = amount['Date'].apply(lambda date: time.strftime(

'%H:%M', (time.strptime(date, '%Y/%m/%d %H:%M'))))

amount['Date'] = amount['Date'].apply(lambda date: date.split(' ')[0])

# %%

"""数据清洗"""

del student['Index'] # data1的数据清洗

# data2.csv的清洗

# 如果卡号相同，但是消费次数没有变说明是相同的记录

# TerSerNo非空的数据为0点结算，作为异常数据排除

# conOperNo非空的数据是非消费数据，作为不需要的数据删除

amount.drop\_duplicates(subset=['CardNo', 'CardCount'])

amount = amount[amount.TermSerNo.isnull()]

amount = amount[amount.conOperNo.isnull()]

amount = amount[amount.Type == '消费'] # 以防万一，再加一条，显式删除非消费数据

# 删除这些被使用过，已经不再需要的字段

del amount['TermSerNo']

del amount['conOperNo']

del amount['FundMoney']

del amount['Type']

# 删除其他无效的数据

del amount['Index']

del amount['PeoNo']

# %%

"""数据整合"""

# consume.csv，保存了所有的消费记录

amount.to\_csv(os.path.join('data', 'consume.csv'))

# grade18.csv，只有18级的信息，但是有很多个人信息

data = pd.merge(student, amount, on=['CardNo'])

data.to\_csv(os.path.join('data', 'grade18.csv'))

**文件: analysis.py**

# %%

"""准备库函数、自定义函数、常量"""

import os

import pandas as pd

import matplotlib.pyplot as plt

import datetime

from collections import Counter

from sklearn.preprocessing import MinMaxScaler

# @github https://github.com/LKI/chinese-calendar

from chinese\_calendar import is\_workday

from IPython.display import display

# 解决matplotlib无法正常显示中文的问题

plt.rcParams['axes.unicode\_minus'] = False

plt.rcParams['font.sans-serif'] = ['STHeiti']

# 默认字体是mac用户的，windows使用下面的这条代码

# plt.rcParams['font.sans-serif'] = ['Simhei']

def time\_map(clock: str):

"""判断用餐时段的函数"""

time\_limit = [['06:00', '09:30'], ['10:30', '14:30'], ['16:00', '22:00']]

if time\_limit[0][0] <= clock < time\_limit[0][1]:

return '早餐'

elif time\_limit[1][0] <= clock < time\_limit[1][1]:

return '午餐'

elif time\_limit[2][0] <= clock < time\_limit[2][1]:

return '晚餐'

else:

return ''

def isWork(date: list):

"""判断是否工作"""

return is\_workday(datetime.date(date[0], date[1], date[2]))

# 制作需要用到的常量

color\_map = {

'第一食堂': 'b',

'第二食堂': 'g',

'第三食堂': 'r',

'第四食堂': 'c',

'第五食堂': 'm',

'教师食堂': 'y'

}

# 生成这一个月所有的工作日

counter = Counter(list(map(isWork, [[2019, 4, i] for i in range(1, 31)])))

# %%

"""数据预处理"""

# 创建食堂的数据

canteen = pd.read\_csv(os.path.join('data', 'consume.csv')) # 读取数据

canteen = canteen[canteen.Dept.str.contains('食堂')]

# 格式化时间，每十分钟为一个时间段

canteen['Time'] = canteen['Time'].apply(lambda time: time[:-1] + '0')

# 判断是否是工作日

canteen['isWork'] = canteen['Date'].apply(

lambda date: isWork(list(map(int, date.split('/')))))

# 创建用餐时段数据

have\_meal = canteen.copy()

have\_meal['Meal'] = have\_meal['Time'].apply(time\_map)

have\_meal = have\_meal[have\_meal.Meal != '']

display(have\_meal)

# %%

"""研究早中晚时段各个食堂就餐人次占比"""

for name, group in have\_meal.groupby(have\_meal['Meal']):

plt.figure(name)

count = []

labels = []

for name\_t, group\_t in group.groupby(group['Dept']):

if group\_t.shape[0] > 10:

count.append(group\_t.shape[0])

labels.append(name\_t)

explode = tuple(map(lambda x: 0.1 if x == max(count) else 0,

count)) # 创建explode向量，凸显出图中占比最大的部分

colors = list(map(lambda x: color\_map[x], labels)) # 固定每个食堂的颜色

plt.pie(count,

explode=explode,

labels=labels,

autopct='%3.1f%%',

colors=colors) # autopct是显示在标签上的数据格式，这里设置了百分比

plt.axis('equal') # 设置横纵坐标相等，使饼图变成圆形

plt.legend(fontsize='small',

bbox\_to\_anchor=(0, 1.02, 1, 0.2),

loc="upper right") # 设置图例的位置，防止与标签重合

plt.savefig(os.path.join('img', '1-1-' + name + '.png'),

bbox\_inches='tight',

dpi=300) # 保存图片

# %%

"""研究每个学生在各个食堂的月人均消费"""

dict\_list = []

for name, group in have\_meal.groupby(have\_meal['Dept']):

dict\_t = {'Dept': name}

for name\_t, group\_t in group.groupby(group['Meal']):

dict\_t[name\_t + '(人均月消费)'] = group\_t.groupby(

group\_t['CardNo'])['Money'].agg(['sum']).mean().values[0]

dict\_list.append(dict\_t)

display(pd.DataFrame(dict\_list[0:]))

# %%

"""研究工作日和非工作日食堂就餐时间峰值"""

# 统计出所有时段的客流量

dict\_list = []

for name, group in canteen.groupby(canteen['isWork']):

for name\_t, group\_t in group.groupby([group['Time'], group['Dept']]):

count = 0

for name\_tt, group\_tt in group\_t.groupby(group\_t['Date']):

count += group\_tt.shape[0]

count /= counter[name]

dict\_list.append({

'Type': '工作日' if name else '休息日',

'Dept': name\_t[1],

'Time': name\_t[0],

'Count': count

})

# 填补缺失的时段数据(如果不填补会导致后面画出来的图像的x轴混乱)

sequence = pd.DataFrame(dict\_list[0:])

for name, group in sequence.groupby([sequence['Type'], sequence['Dept']]):

for t in sequence['Time'].unique():

if t not in group['Time'].values:

sequence = sequence.append([{

'Type': name[0],

'Dept': name[1],

'Time': t,

'Count': 0

}])

# 筛选掉基本上无用的时间段

sequence = sequence[sequence['Time'] > '06:00']

# 画图

for name, group in sequence.groupby(sequence['Type']):

print(name)

group.sort\_values(by="Time", inplace=True)

plt.figure(name, figsize=(18, 5))

plt.xlabel('时间')

plt.ylabel('平均人数')

for name\_t, group\_t in group.groupby(group['Dept']):

group\_t.sort\_values(by="Time", inplace=True)

plt.plot(group\_t['Time'],

group\_t['Count'],

label=name\_t,

c=color\_map[name\_t])

plt.legend(prop={'size': 20})

# 选装x轴标签，防止重叠

plt.xticks(rotation=60) # 让x轴的标签旋转60度，避免标签重叠

plt.tight\_layout() # 让对坐标轴的操作生效

plt.savefig(os.path.join('img', '1-2-' + name + '.png'), dpi=300)

# %%

"""研究食堂物价与学生消费情况的关系"""

dict\_list = []

for name, group in have\_meal.groupby([have\_meal['Dept'], have\_meal['Meal']]):

dict\_t = {

'Dept': name[0],

'Meal': name[1],

'AvgMoney': group['Money'].agg(['sum'])[0] / group.shape[0],

'Count': group.shape[0]

}

dict\_list.append(dict\_t)

cost = pd.DataFrame(dict\_list[0:])

# print(cost)

# 通过Pearson线性相关系数验证物价越高，学生消费意愿越低

dict\_list = []

for name, group in cost.groupby('Meal'):

# 求解Pearson相关系数

std = MinMaxScaler(feature\_range=(0, 1)) # 线性处理

group['AvgMoney'] = std.fit\_transform(group['AvgMoney'].values.reshape(

-1, 1)) # 因为函数只接收纵向的向量，所以需要reshape变换

group['Count'] = std.fit\_transform(group['Count'].values.reshape(-1, 1))

# print(group)

dict\_t = {'时间段': name, '线性相关系数': group['AvgMoney'].corr(group['Count'])}

dict\_list.append(dict\_t)

relationship = pd.DataFrame(dict\_list[0:])

display(relationship)

**文件: model.py**

# %%

"""库函数导入、定义函数与类"""

import pandas as pd

from sklearn.pipeline import make\_pipeline

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import MinMaxScaler

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

import numpy as np

import warnings

import os

from IPython.display import display

# 解决matplotlib无法正常显示中文的问题

plt.rcParams['axes.unicode\_minus'] = False

plt.rcParams['font.sans-serif'] = ['STHeiti']

# 默认字体是mac用户的，windows使用下面的这条代码

# plt.rcParams['font.sans-serif'] = ['Simhei']

def comparision(vec: list):

"""

生成比较矩阵

CopyRight:

https://blog.csdn.net/aBIT\_Tu/article/details/84029849

"""

n = len(vec)

F = np.zeros([n, n])

for i in range(n):

for j in range(n):

if i == j:

F[i, j] = 1

else:

F[i, j] = vec[i] / vec[j]

return F

class AHP:

"""AHP类

提供AHP算法支持

CopyRight:

https://www.guofei.site/2020/01/05/ahp.html

"""

def \_\_init\_\_(self, criteria, b):

self.RI = (0, 0, 0.58, 0.9, 1.12, 1.24, 1.32, 1.41, 1.45, 1.49)

self.criteria = criteria

self.b = b

self.num\_criteria = criteria.shape[0]

self.num\_project = b[0].shape[0]

def cal\_weights(self, input\_matrix):

input\_matrix = np.array(input\_matrix)

n, n1 = input\_matrix.shape

assert n == n1, '不是一个方阵'

for i in range(n):

for j in range(n):

if np.abs(input\_matrix[i, j] \* input\_matrix[j, i] - 1) > 1e-7:

raise ValueError('不是反互对称矩阵')

eigenvalues, eigenvectors = np.linalg.eig(input\_matrix)

max\_idx = np.argmax(eigenvalues)

max\_eigen = eigenvalues[max\_idx].real

eigen = eigenvectors[:, max\_idx].real

eigen = eigen / eigen.sum()

if n > 9:

CR = None

warnings.warn('无法判断一致性')

else:

CI = (max\_eigen - n) / (n - 1)

CR = CI / self.RI[n]

return max\_eigen, CR, eigen

def run(self):

max\_eigen, CR, criteria\_eigen = self.cal\_weights(self.criteria)

print('准则层：最大特征值{:<5f},CR={:<5f},检验{}通过'.format(

max\_eigen, CR, '' if CR < 0.1 else '不'))

print('准则层权重={}\n'.format(criteria\_eigen))

max\_eigen\_list, CR\_list, eigen\_list = [], [], []

for i in self.b:

max\_eigen, CR, eigen = self.cal\_weights(i)

max\_eigen\_list.append(max\_eigen)

CR\_list.append(CR)

eigen\_list.append(eigen)

pd\_print = pd.DataFrame(

eigen\_list,

index=['准则' + str(i) for i in range(self.num\_criteria)],

columns=['方案' + str(i) for i in range(self.num\_project)],

)

pd\_print.loc[:, '最大特征值'] = max\_eigen\_list

pd\_print.loc[:, 'CR'] = CR\_list

pd\_print.loc[:, '一致性检验'] = pd\_print.loc[:, 'CR'] < 0.1

print('方案层')

print(pd\_print)

# 目标层

obj = np.dot(criteria\_eigen.reshape(1, -1), np.array(eigen\_list))

print('\n目标层', obj)

print('最优选择是方案{}'.format(np.argmax(obj)))

return obj

# %%

"""准备模型所需的特征数据集"""

amount = pd.read\_csv(os.path.join('data', 'consume.csv'))

amount = amount[amount.Dept.str.contains('食')

| amount.Dept.str.contains('超市')] # 筛选出所有的非公共消费，有部分消费是水电费等消费

# 创建模型评价所需的特征数据集

dict\_list = []

for name, group in amount.groupby(amount['CardNo']):

money = group['Money'].agg(['sum', 'mean', 'max'])

dict\_t = {

'CardNo': name,

'AvgSurplus': format(group['Surplus'].agg(['mean'])[0], '.2f'),

'TotalConsume': format(money[0], '.2f'),

'Freq': group.shape[0]

}

dict\_list.append(dict\_t)

# 导出成DataFrame

features = pd.DataFrame(dict\_list[0:])

features = features[features.Freq >= 30] # 筛选掉所有消费频率低于30次的数据，认为是不正常的

display(features)

# %%

"""通过Elbow Method求K-Means最佳k值"""

training\_data = features.loc[0:, ['AvgSurplus', 'TotalConsume',

'Freq']] # K-means训练集

# 当k小于真实聚类数时，由于k的增大会大幅增加每个簇的聚合程度，故 SSE 的下降幅度会很大。

# 而当k到达真实聚类数时，再增加k所得到的聚合程度回报会迅速变小，所以 SSE 的下降幅度会骤减，

# 随着k值的继续增大而趋于平缓。

# 将每个簇的质点与簇内样本点的平方距离误差和称为畸变程度(distortions)，

# 那么，对于一个簇，它的畸变程度越低，代表簇内成员越紧密，

# 畸变程度越高，代表簇内结构越松散。

# 畸变程度会随着类别的增加而降低，

# 但对于有一定区分度的数据，在达到某个临界点时畸变程度会得到极大改善，之后缓慢下降，

# 这个临界点就可以考虑为聚类性能较好的点。

SSE = [] # 存放每次结果的误差平方和

for k in range(2, 20):

km = KMeans(n\_clusters=k) # 构造聚类器

scaler = StandardScaler()

pipline = make\_pipeline(scaler, km)

pipline.fit(training\_data)

SSE.append(km.inertia\_) # inertia\_:样本到其最近聚类中心的平方距离之和

plt.figure(figsize=(10, 5))

plt.plot(range(2, 20), SSE, 'o-')

plt.xticks(range(0, 22, 1))

plt.grid(linestyle='--')

plt.xlabel("分类数量K")

plt.ylabel('误差平方和(SSE)')

plt.savefig(os.path.join('img', '2-1.png'), dpi=300)

# %%

"""创建K-means模型"""

# 由上图可以认为最佳的K值在5左右，我们选取n\_clusters为5。

km = KMeans(n\_clusters=6, algorithm='elkan')

# scaler = MinMaxScaler(feature\_range=(0, 1))

scaler = StandardScaler()

pipline = make\_pipeline(scaler, km) # 训练kmeans模型

pipline.fit(training\_data)

features['Labels'] = pipline.predict(training\_data) # 预测数据的Labels

display(features)

# %%

"""基于DFM(Deposit、Frequency、Monetary)模型的K-Means分类"""

fig = plt.figure(figsize=(8, 6)) # figsize-画布大小

ax = Axes3D(fig) # 创建3D图

colors = ['c', 'darkorange', 'lime', 'red', 'fuchsia', 'yellow']

for name, group in features.groupby(features['Labels']):

std = MinMaxScaler(feature\_range=(0, 1)) # 标准化，不然画出来比较难看

x = std.fit\_transform(group['AvgSurplus'].values.reshape(-1, 1))

y = std.fit\_transform(group['TotalConsume'].values.reshape(-1, 1))

z = std.fit\_transform(group['Freq'].values.reshape(-1, 1))

print('drawing label {}'.format(name))

ax.scatter3D(x, y, z, c=colors[name], label=name, s=4)

ax.view\_init(30, 45) # 旋转视图的角度

ax.set\_xlabel('平均存储系数')

ax.set\_ylabel('总消费系数')

ax.set\_zlabel('消费频率系数')

plt.legend()

plt.savefig(os.path.join('img', '2-2.png'), dpi=300)

# %%

"""对上图模型的验证"""

# 根据上图请自行调整，因为K-Means分类的label值不稳定，所以每次运行完前面的无比更改这个值

# 这个类是上图中所有项都最低的那一个类

group\_label = 2

deep\_model = features[features.Labels == group\_label]

deep\_model['AvgSurplus'] = deep\_model['AvgSurplus'].astype("float64")

deep\_model['TotalConsume'] = deep\_model['TotalConsume'].astype("float64")

display(deep\_model)

# %%

"""基于AHP算法细分模型"""

criteria = comparision([1, 5, 3])

b = [

comparision(deep\_model['AvgSurplus'].values),

comparision(deep\_model['TotalConsume'].values),

comparision(deep\_model['Freq'].values)

]

results = AHP(criteria, b).run()

# %%

""""""

std = MinMaxScaler(feature\_range=(0, 10)) # 标准化，不然画出来比较难看

results\_t = list(map(lambda x: x \* 10000, results[0]))

deep\_model['Rank'] = results\_t

deep\_model['Grade'] = deep\_model['Rank'].apply(int)

deep\_model = deep\_model.sort\_values(by='Rank')

display(deep\_model)

# %%

count = []

grade = []

for name, group in deep\_model.groupby(deep\_model['Grade']):

grade.append('{}分'.format(name))

count.append(group.shape[0])

explode = tuple(map(lambda x: 0.1 if x == max(count) else 0, count))

plt.pie(count, explode=explode, labels=grade, autopct='%3.1f%%')

plt.axis('equal')

plt.legend(fontsize='small',loc="upper right")

plt.savefig(os.path.join('img', '3-1.png'), dpi=300)

# %%