# “拍照赚钱”的任务定价

## 摘 要

本文参考现有的数据资料，构建了符合问题的模型，利用曲面拟合、熵权法、k-中心聚类分析、机器学习等方法，对项目的定价方案以及方案实施效果评价等问题进行了研究。

对于问题一，其任务定价规律从确定最优中心点和确定价格与半径的关系两个步骤入手。首先通过数据可视化，绘制任务位置与会员位置的散点图像，定义区域组团。针对每一组团，利用 K-中心聚类法确定存在一定位差的任务位置中心与会员位置中心，构建中心点位置修正模型，获得最优中心。接着以此中心为圆心，多次拓展半径，对应不同圆区域，当圆内某价格的任务数与组团内该价格的任务数之比大于给定阈值时，获得该价格对应的半径。最后，以组团 1 为例，通过最小二乘法线性回归分析得到定价规律函数 $R = 0.0227869826156 \times price - 1.38302322646$。

为解释任务未完成的原因，可通过建立完成度分析模型，得到地理位置、会员密度、会员信誉度等因素与之相关。

对于问题二，首先，遴选出影响完成度的三个主要因素：任务、会员信息、社会经济，每个主要因素下包含若干次要因素。然后，将以上因素进行归一化处理，并建立熵权法综合评价打分模型，为每个任务点打分，用于量化学习结果。接下来，针对未完成任务点，向邻近已完成任务点学习，通过改变价格，优化未完成任务点得分接近已完成点得分，提高任务点完成的可能性。总结出基于已完成任务经验的定价方案。经检验，该定价方案完成度较问题 1 提高，方案更优。

对于问题三，首先构建任务打包模型，以集合内未分组的任务点为圆心，以阈值 T=0.0555 为半径，确定圆区域，区域内未分组的任务点与圆心任务点为同组，直至完成全部邻近任务的集合化处理，同组即打为一包。然后，统计包内任务数与可接包会员数，并利用加权平均方法确定此包的价格。接下来，利用曲面拟合包定价与任务数 x、可接单会员数 y 关系为：

$$price = -1.204 + 76.62x + 0.03657y - 0.517x^2 - 0.03157xy - 0.0001187y^2 + 0.007248x^3 - 0.002333x^2y + 0.0001074xy^2$$

最后，通过打包后的任务接单完成率高于附件 1 任务完成率，说明此定价方案使任务完成率提高。

对于问题四，首先，通过对附件 3 中数据可视化处理，发现新任务点更为集中，所以采取问题三的任务打包模型，更改阈值 T=0.02，完成打包。然后，用与问题三相同的方法统计包内任务数与可接包会员数。其次，遍历附件 3 与附件 1 的相同区域，学习附件 1 定价数据。接下来，利用曲面拟合包定价与任务数 x、可接单会员数 y 关系为：

$$price = 1.748 + 70.83x + 0.04777y - 0.04682x^2 - 0.07542xy - 0.0003519y^2 + 0.0001368x^3 - 0.003179x^2y + 0.0007037xy^2$$

最后，通过评价附件 3 中数据的接单成功率和完成度，说明此方案实施效果好。

本文的特色在于充分利用数据可视化手段，根据实际问题，建立模型，切实有效解决问题。

**关键词：K-中心聚类分析 数据拟合 熵权法 数据可视化**

# 一、问题重述

“拍照赚钱”是移动互联网下的一种自助式服务模式。用户下载 APP，注册成为 APP 的会员，然后从 APP 上领取需要拍照的任务（比如上超市去检查某种商品的上架情况），赚取 APP 对任务所标定的酬金。这种基于移动互联网的自助式劳务众包平台，为企业提供各种商业检查和信息搜集，相比传统的市场调查方式可以大大节省调查成本，而且有效地保证了调查数据真实性，缩短了调查的周期。因此 APP 成为该平台运行的核心，而 APP 中的任务定价又是其核心要素。如果定价不合理，有的任务就会无人问津，而导致商品检查的失败。

问题 1：研究附件一（一个已结束项目的任务数据，包含了每个任务的位置、定价和完成情况）中项目的任务定价规律，分析任务未完成的原因。

问题 2：为附件一中的项目设计新的任务定价方案，并和原方案进行比较。

问题 3：实际情况下，多个任务可能因为位置比较集中，导致用户会争相选择。若将这些任务联合在一起打包发布，给出修改后的定价模型，并研究对最终的任务完成情况的影响。

问题 4：对附件三（一个新的检查项目任务数据，包含任务的位置信息）中的新项目给出任务定价方案，并评价实施效果。

# 二、问题假设

1.假设会员只要接单，此任务就一定会达到完成状态
2.假设会员与任务之间信息对等，且执行任务的效率相同。
3.假设一个任务只能由一名会员来完成，同样一名会员在同一时刻，只能完成一个任务。
4.假设忽略 GPS 定位误差，题目所给位置即为准确信息。

# 三、符号说明

| 序号 | 符号 | 意义 |
|---|---|---|
| 1 | X | 组团数据集 |
| 2 | Z | 聚类中心 |
| 3 | K | 聚类中心数 |
| 4 | $\theta_N$ | 组团中样本数最小值 |
| 5 | $\theta_S$ | 组团中样本标准差阈值 |
| 6 | I | 允许迭代次数 |
| 7 | L | 迭代运算合并对数 |
| 8 | $\theta_C$ | 两组团最小中心距离 |

# 四、问题分析

## 4.1 问题一的分析

问题一要求根据附件 1 中数据，分析项目的任务定价规律，进一步分析得出任务未完成的原因。首先要明确影响任务定价和任务完成度的因素，分别是任务位置、会员位

置和任务定价、任务位置、会员信息（信誉度、密度）。对于较大的数据量处理分类后采取坐标可视化。通过建立价格与任务位置和价格与会员位置模型，在散点密集区构架城市区域组团，利用 K 中心聚类法找到两个存在一定位差的组团中心，构建中心位点修正模型，获得最优中心。最后以此中心为圆心多次拓展半径、与给定阈值（圆内某价格的任务数与组团内该价格的任务数之比）比较来确定价格半径区间，通过最小二乘法线性回归得到定价与半径间的规律函数。对于任务未完成的原因，建立完成度分析模型，从地理位置、会员密度、会员信誉度三方面分析。最终综合三方面因素，完成评价。

### 4.2 问题二的分析

问题二要求为附件 1 中的项目设计新的任务定价方案，并和原方案进行比较。首先分析得出任务、会员信息、社会经济等因素为影响完成度的三层指标，将其进行归一化处理成无量纲的量后，建立熵权法综合评价打分模型，衡量指标的影响程度。然后通过构建优秀经验学习模型，使未完成任务向已完成任务学习靠拢，改变未完成任务定价，使得新的定价优于附件一中原有定价。对比新旧方案的完成度可知新方案更优。

### 4.3 问题三的分析

问题三要求将任务联合在一起打包发布，给出修改后的定价模型，并研究对最终的任务完成情况的影响。首先建立任务打包模型将相邻一定距离内的任务集合化处理（“打包”处理）。然后以一组任务中心为圆心，一定距离为半径在会员位置可视化图像上确定圆区域，根据会员密度及预定限额，统计圆区域内任务数及可接单会员数。最后建立价格模型，寻找定价与任务数、会员数之间的关系，利用三维拟合方法得到修改后的定价模型。通过打包后的接单成功率与附件 1 任务完成率作比较，可分析评价出影响情况。

### 4.4 问题四的分析

问题四要求根据附件三新项目制定定价方案，并评价实施效果。在问题四中首先应该对附件 3 中给出的新的任务点的数据进行可视化。其次由于新任务点更为集中，需要将新任务点进行打包处理，得到相应的包信息。然后将处理好的包信息与附件 1 中的价格信息进行比拟，通过学习的方法得出新的打包的整体金额。再通过每个包中的相关数据：包内的任务个数、具有接单能力的会员个数以及任务的价格进行拟合，从而得出附件 3 情况下的定价模型的函数关系。在评价阶段，由题意可知：任务完成度的高低是评价定价方案的优劣最为关键的指标。在得出上述定价规律后，计算附件 3 中的数据的接单成功率和完成度，并与附件 1 的数据进行比较，进一步评价实施效果。

## 五、问题一模型的建立与求解

### 5.1 数据分析

#### 5.1.1 数据预处理与分类整理

首先，观察附件一数据构成，其包括任务编号、任务 GPS 经纬度位置、定价和完成情况。接着，遍历附件一数据，检测是否存在异常值，此时异常值包括具有明显偏差的值与空缺值。若存在异常值，则填补或删除空缺值，舍去明显偏差值。检测可得附件一数据，未存在异常值，可进行进一步分析。最后获得有关数据总表。

表 1：附件 1 数据统计总表

| 任务标价（元） | 已完成（件） | 未完成（件) | 总数（件） | 完成度 |
|---|---|---|---|---|
| 65 | 35 | 30 | 65 | 0.538 |
| 65.5 | 76 | 74 | 150 | 0.507 |
| 66 | 46 | 57 | 103 | 0.447 |

| | | | | |
|---|---|---|---|---|
| 66.5 | 35 | 28 | 63 | 0.556 |
| 67 | 18 | 20 | 38 | 0.474 |
| 67.5 | 17 | 6 | 23 | 0.739 |
| 68 | 25 | 5 | 30 | 0.833 |
| 68.5 | 6 | 5 | 11 | 0.545 |
| 69 | 12 | 7 | 19 | 0.632 |
| 69.5 | 6 | 2 | 9 | 0.667 |
| 70 | 77 | 19 | 96 | 0.802 |
| 70.5 | 9 | 2 | 11 | 0.818 |
| 71 | 3 | 1 | 4 | 0.75 |
| 71.5 | 4 | 1 | 5 | 0.8 |
| 72 | 42 | 18 | 60 | 0.7 |
| 72.5 | 7 | 2 | 9 | 0.778 |
| 73 | 7 | 3 | 10 | 0.7 |
| 73.5 | 3 | 2 | 5 | 0.6 |
| 74 | 1 | 4 | 5 | 0.2 |
| 74.5 | 1 | 1 | 2 | 0.5 |
| 75 | 59 | 19 | 78 | 0.641 |
| 80 | 9 | 4 | 13 | 0.692 |
| 85 | 24 | 3 | 27 | 0.889 |
| 总数（件） | 522 | 313 | 835 | |

由上表可知，附件一全部任务中，任务完成率为 62.51%。

### 5.1.2 数据坐标化与可视化

首先，进行 GPS 数据坐标化可行性分析。将经纬度坐标向平面二维坐标系投影时，会因纬线间隔由赤道向两极增大，产生角度和方向的变形。根据墨卡托投影定义，经度数据均匀变化，纬度数据在$(-85^{\circ}, 85^{\circ})$之间，且差距在$6^{\circ}$之内，没有区间转化，基本没有变形影响，可进行平面投影。

接着，以经度为 x 轴，以纬度为 y 轴，建立二维坐标系进行投影。并对附件 1、2 数据通过 python 进行图像可视化处理。

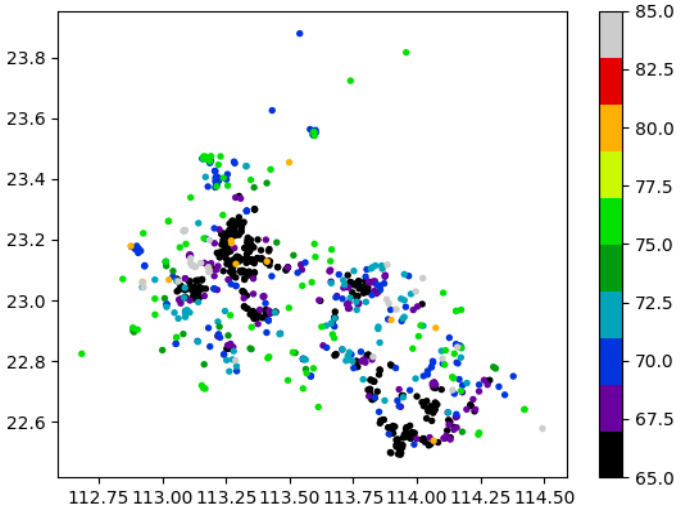


图 1：附件一可视化位置-价格结果

## 5.2 价格与任务地理位置模型与求解

由数据预处理阶段可知，当价格<=75 时，价格变动幅度为 0.5 元，当价格>75 时，价格变动幅度为 5 元。且价格<=75 的任务数占全体附件一任务数的 95.21%，因此将附

件一任务依照价格分为两类。且首先考虑附件 1 任务地理位置信息，并将附件 2 会员信息作为评价修正数据进行考虑。

首先，观察附件 1 可视化效果图，可得价格存在圈层辐射关系。考虑题目现实环境为某一实际大城市，结合城市发展、规划、分区等现实条件，一个大型城市之中拥有不同城市区域，且城市区域以圈层式向外拓展，定义这样的一个城市区域为一个组团，组团区域半径不同，且每个组团中都拥有各自区域中心。从区域向外扩散，距离越远，人员、资源密集度逐渐降低。此时，随着由区域中心向外围扩散，任务定价需提高价格，以吸引人们进行任务的领取。组团区域大小同此组团的经济发展水平、组团主导产业、发展历史等众多因素有关，因此不同组团的区域中心的影响半径不同，需分别考虑。

其次，针对以上分析，利用聚类方式找出组团区域中心点与半径显得尤为重要。结合可视化结果，将以上区域分为 3 个组团，通过 K 中心聚类分析，得到 3 个组团区域中心位置坐标（center1x,center1y），（center2x,center2y），（center3x,center3y）。

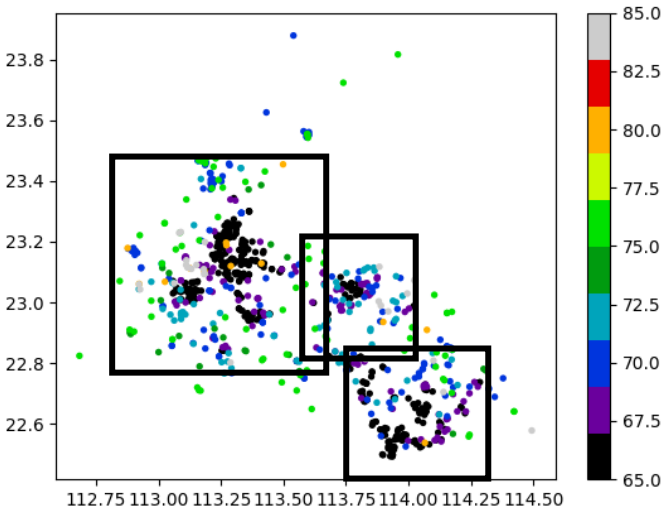


图 2：附件一组团划分

K 中心聚类分析的具体方法为[1]：任意一个组团数据集 X 中，随机选择一个数据对象作为初始的聚类中心，分别计算数据集中每个数据对象到此聚类中心的距离之和。遍历整个数据集，反复利用非中心点来代替中心点，以距离之和大小为代价，如果总代价为负的，那么实际的平方差将会减少，代表对象可被非代表对象代替，否则被认为可被接受。迭代至收敛，此点为最优中心点。公式如下：

$$d = min\sum_{k}\sqrt{\left(centerx - noncentralx\right)^{2} - \left(centery - noncentraly\right)^{2}} \quad (5\text{-}2\text{-}1)$$

为判定一个非中心点 n 是否为当前中心点 m 的好的替代，每一个非中心点对象 n，都有多种情况需要考虑。首先判断非中心点的中心点隶属问题，然后分支讨论与新中心点距离远近的情况，再研究分配问题。流程如图所示：

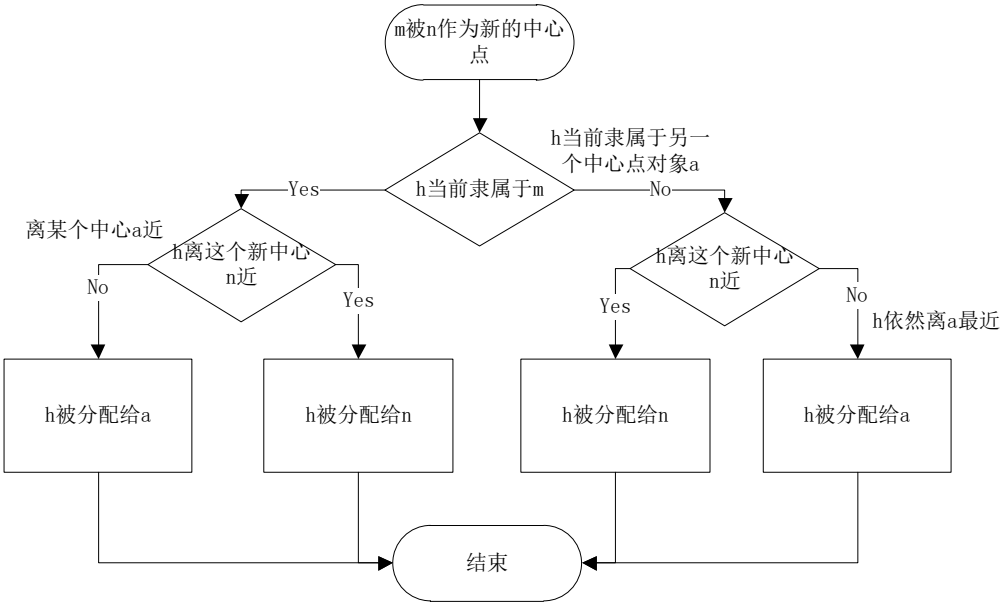


图 3：K 中心聚类分析判定流程图

以位于左侧面积最大的组团 1 为例进行以下分析，组团 2、3 同理即可得到，不再赘述。首先，依照上图组团位置，将组团 1 分离出来，并只考虑价格<=75 的点。进行可视化后效果如下图所示。

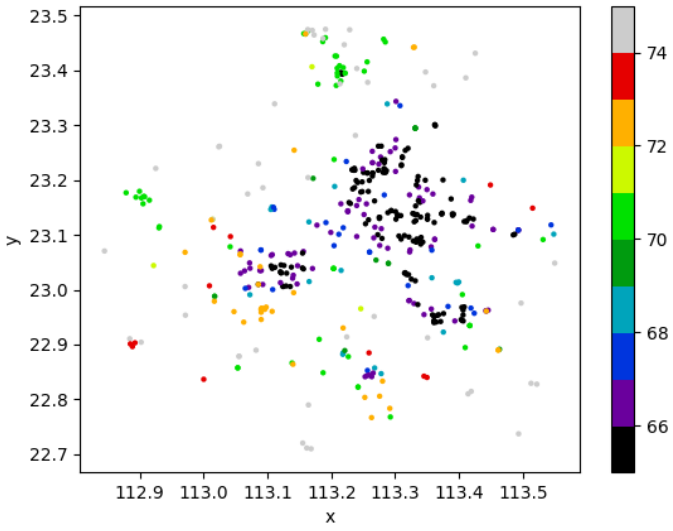


图 4：组团 1 位置-价格（价格<=75）对照结果

而后，利用上述 K 中心聚类方法，求出仅考虑任务位置因素下组团 1 区域中心，如下图中箭头指示黄色点位置。

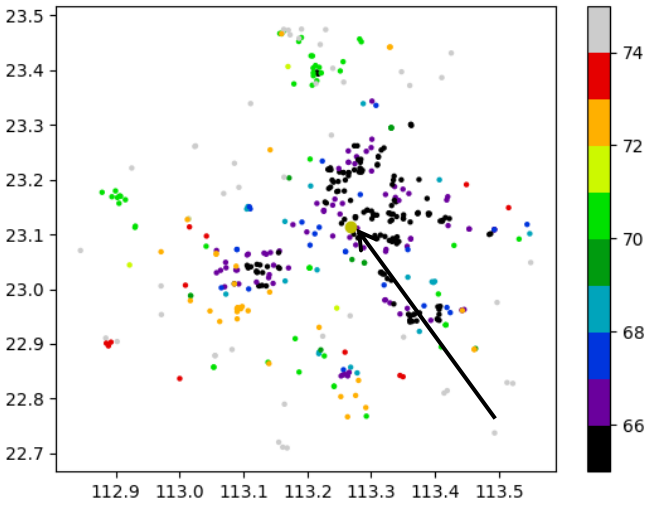


图 5：组团 1 任务位置导向区域中心

接下来，针对三个组团，分别确定其价格环半径区间。以区域中心为基准，以一定半径拓展进度 i 由区域中心向外拓展，每次拓展，计算圈层范围内同一价格的数量占该区域总任务中同一价格的数量的百分比，当某一价格百分比大于阈值 T 时，即确定该圈层为价格分界层，以此类推，得到组团与任务地理位置相关的定价标准。

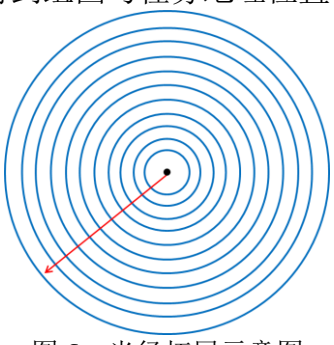

图 6：半径拓展示意图

以图 5 组团 1 任务位置导向区域中心为圆心向外拓展，以圆心与图片四个边界距离差的最大值为总拓展长度，为保证数据的可靠性，进行 1000 次拓展。因此半径拓展进度 i 由下列公式计算得到。

$$i = \max(centerx - left, right - centerx, up - centery, centery - bottom) / 1000 \quad (5\text{-}2\text{-}2)$$

以 T=0.4 为价格分阶层阈值效果良好，经计算得半径-价格关系如下图所示，近似满足线性关系。通过 python 依照最小二乘法进行线性回归。

$$R = k \times price + b \quad (5\text{-}2\text{-}3)$$

$$k = \frac{\sum price \times R - \frac{1}{N}\sum price \sum R}{\sum price^2 - \frac{1}{N}(\sum price)^2} \quad (5\text{-}2\text{-}4)$$

$$b = \overline{R} - a \times \overline{price} \quad (5\text{-}2\text{-}5)$$

得到半径-价格关系为：

$$R = 0.0220338248652 \times price - 1.33044963389 \quad (5\text{-}2\text{-}6)$$

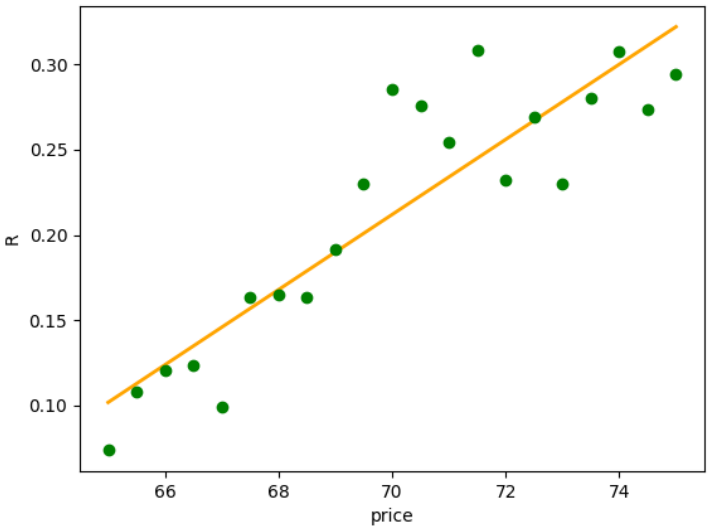


图 7：以任务位置中心拓展为导向的半径-价格关系

由于以上分析未考虑任务价格>75 元，因此对于价格>75 元的较为特殊任务单独分析。附件一中价格>75 元的任务共有 41 个，未完成率为 0.17，多集中于组团边缘，距离高质量会员集中地位置较远，操作便利程度较低而无法完成。在会员集中区的未完成任务可能因为操作难度和时间因素而无法完成。

### 5.3 建立价格与会员地理位置的模型与求解

首先，观察附件 2 可视化效果图，知其与 5.2 模型下的图像形状类似。由附件 1 数据可知信誉值高的会员，活跃度高，接单任务限额高，可保证价格合理地区的完成率，其地理位置对定价规律有很大影响。本文选取信誉度高于 100 的会员为高质量会员，任务限额总和 6279 件，人均 27.06 件，占接单的主导位置。通过 k 中心聚类分析，得到 3 个组团区域中心位置坐标（center1x’,center1y’），（center2x’,center2y’），（center3x’,center3y’）。

仍以组团 1 为例，经过 k 中心聚类分析，以高质量会员为导向的组团中心位置为下图绿色点所示。

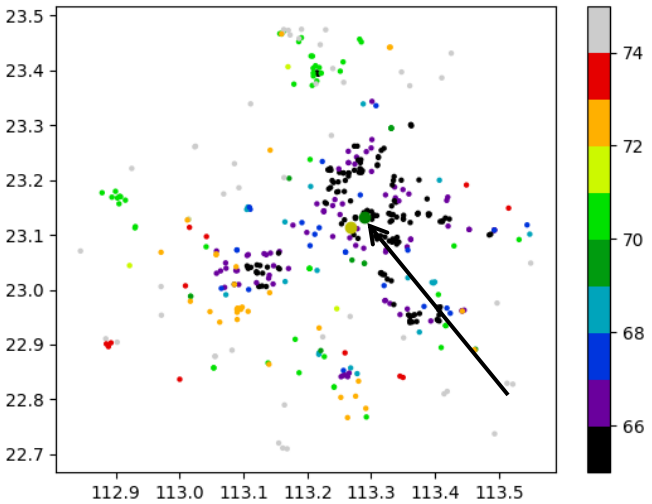


图 8：组团 1 高质量会员位置导向区域中心

经对比发现，两类情况下区域中心存在一定差别，因此进入中心点位置修正阶段。

### 5.4 中心点位置修正模型

价格会受到任务地理位置和会员地理位置的双重影响，可通过多次迭代的方式，将两种情况下的中心位置进行整合修正，从而获得最优中心。

Step1：选择任意两对应组团的两个聚类中心 $Z_1(1)$、 $Z_2(1)$，定义 k、$\theta_S$、$\theta_N$、$\theta_C$、I、L 等算法参数。分配 N 个样本到两个组团中，根据最近邻原则，模型如下：

$$\|x - z_i\| < \|x - z_j\| \quad i=1,2 \quad 且 i \neq j \tag{5-4-1}$$

可得出 $x \in X_i$。

Step2：对聚类中心进行修正，计算式如下：

$$z_i = \frac{1}{N_I} \sum_{x \in X_I} x \tag{5-4-2}$$

计算两聚类中心的平均距离，计算式如下：

$$\overline{d_i} = \frac{1}{N_i} \sum_{x \in X_i} \|x - z_i\| \tag{5-4-3}$$

Step3：计算标准差，通过比较与样本标准差阈值的大小关系，判断继续迭代与否，计算式如下：

$$\sigma_{ij} = \sqrt{\frac{1}{Ni} \sum_{x \in X_i} (x_{kj} - z_{ij})^2} \tag{5-4-4}$$

若 $\max \sigma_{ij} > \theta_s$，转回 Step1，继续进行迭代直到 $\max \sigma_{ij} < \theta_s$，后再修正聚类中心。此时 $d_i$=0，所得聚类中心为修正后的最优点。

依照中心点位置修正模型，以修正后所得点为新区域中心，以一定半径拓展进度 i 由区域中心向外拓展，每次拓展，计算圈层范围内同一价格的数量占该区域总任务中同一价格的数量的百分比，当某一价格百分比大于阈值 T 时，即确定该圈层为价格分界层，以此类推，得到组团与任务地理位置相关的定价标准。同样通过 python,经最小二乘法线性回归，得到此时线性回归结果为：

$$R = 0.0227869826156 \times price - 1.38302322646 \tag{5-4-5}$$

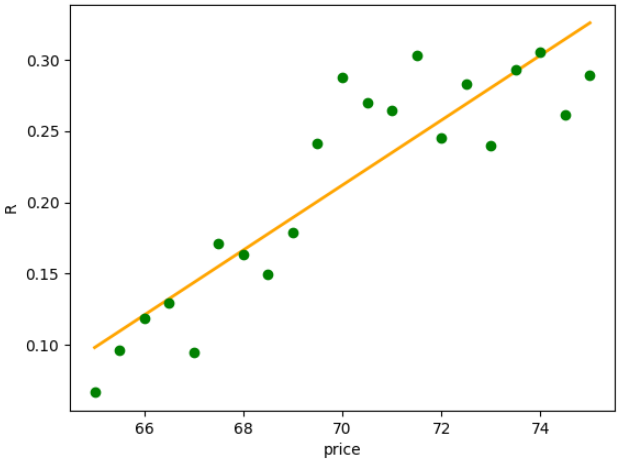


图 9：经高质量会员因素修正后的半径-价格关系

经检验修正后误差平方和较修正前减小，因此以组团 1 为例，确定修正后区域中心后，依照预测点与区域中心半径 R 结合式：

$$R = 0.0227869826156 \times price - 1.38302322646 \quad (5\text{-}4\text{-}6)$$

即可确定某点定价。组团 2、3 经同样分析过程可得定价规律，不再赘述。

## 5.5 未完成原因分析

### 5.5.1 任务标价与完成度分析

从表 1 中数据本文可以得出当价格越高的时候任务的完成度也会有一定的增加。根据表 1 进行具体的数学分析：确定阈值 $K_1$，用于判断完成度的优劣；当完成度 $n \le K_1$ 时，则定义为该价格情况下的任务完成度为不合格；当完成度 $n > K_1$ 时，则定义为该价格情况下的任务完成度为合格。

1.始阈值 $K_1$ 的确定：根据完成度的高低对价格进行排列，得到新的数据表格和顺序。在 835 个任务数据中，522 个为已完成；313 个为未完成，整体的完成度为：0.625。则将 0.625 作为 的初始阈值。

2.分析初始阈值情况下的完成情况：将 $K_1$ 与 23 组完成度 $n_i$（$i = 1、2、3.....23$）进行比较，得出 $n_i > K_1$ 的任务标价为：67.5、68、69、69.5、70、70.5、71、71.5、72、72.5、73、75、80、85。在这 14 组数据中，未完成的个数为 92、已完成的个数为 301、完成率为 0.766。因为 所以可得这 14 组数据的完成率要高于平均值。取数据中的价格最小值即 67.5，设为 $s_1$，判断价格 $s_i > s_1$ 时，计算完成率 $n$ 的值。此时 $n_0 = 0.748$。

3.阈值 K 的迭代计算与选取：将初始阈值得出的完成率进行比较，不断改变阈值 K 的大小，反复重复步骤 2，取出完成率组最高的阈值 $K_{\max}$。从而得出相应的任务标价。根据多次迭代计算可得：

$$n_{\max} = 0.747 \quad (5\text{-}5\text{-}1)$$

则相对应的任务金额为 70 元。即当任务金额 $n \ge 70$ 时（排除 80 和 85 的特殊情况）任务的完成度最高且为：0.747。

### 5.5.2 地理位置与完成度分析

如上文所述，本文已经根据任务密集程度，将城市划分为为三个组团，并通过 K 中心聚类的算法计算出了相应组团的中心点坐标（center1x,center1y），（center2x,center2y），（center3x,center3y）。接下来，以三个任务位置中心为原点建立向外环状辐射的数学模型用于求解任务标价与其辐射半径（距离密集中心距离）的函数关系 $R = 0.0220338248652 \times price - 1.33044963389$。

结合上一个步骤可知，当任务标价 $n \ge 70$ 时（排除 80 和 85 的特殊情况），任务的完成度较高，即当 $65 \le n < 70$ 时，任务的完成度较低。代入不同任务金额与地理位置信息的函数表达式可知：

$$R = 0.0227869826156 \times price - 1.38302322646 \quad (5\text{-}5\text{-}2)$$

所以有：

$$price = \frac{R + 1.38302322646}{0.0227869826156} \quad (5\text{-}5\text{-}3)$$

当 $n \ge 70$ 时，即上式大于 70 时，完成度较高。

### 5.5.3 会员信息与完成度分析

会员的密度和信誉度的高低与任务的标价具有函数的关系，且信誉度的评判标准之一为任务的完成情况，则可知会员的密度 $m$ 和其相应的信誉值 $a$ 都是完成度 $n$ 的影响因

素。

①会员密度的影响

将附件二中关于会员的数据信息进行可视化处理，通过 Python 编程得出相应的散点图。对比数据信息处理掉明显偏差值，即完成会员信息的预处理。

根据散点图进行观察，可得出会员的分布也具有明显的聚集性，亦可分为三个组团通过 K 中心点聚类算法寻找出会员地理信息中的聚集中心点（center1x’,center1y’），（center2x’,center2y’）,（center3x’,center3y’）。将会员地理信息的散点图与任务地理信息的散点图进行叠加，如下图所示：

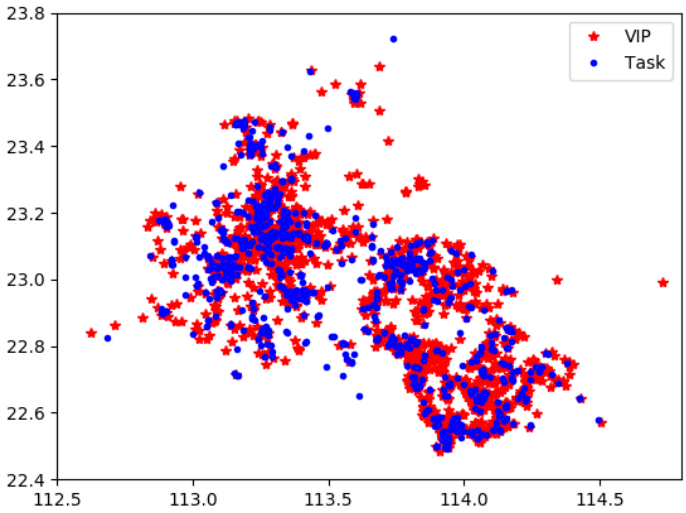


图 10：任务地理位置与会员地理位置叠加分布图

根据散点图观察可知，会员的散点分布与任务的散点分布具有重叠的影响。

②会员信誉度的影响

将会员的活动范围和影响力通过信誉值的高低来表示，即信誉值越高影响力越大、活动范围也就越大。为简化问题，本文通过对信誉值大小进行了筛选，确定了阈值 $K_h$，当信誉值大于 $K_h$ 时属于高级会员，同样，当信誉值小于 $K_h$ 时属于低级会员。根据数据筛选和迭代计算、本文可以得出：

$$K_h = 100 \qquad (5\text{-}5\text{-}4)$$

即可筛选出高级会员与低级会员，并根据两者之间的影响力与完成任务情况的不同来分析其对于任务完成度的影响。

根据题目中给出的判断信誉度的方法，本文可以得知高级会员的任务选择数量与完成情况均高于低级会员。通过 Python 编程，改变参数构建出低级会员的散点图和高级会员的散点图，如下图所示：

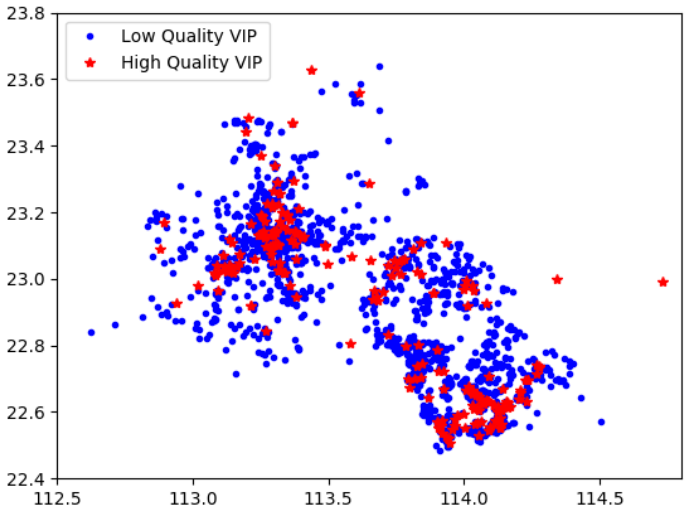


图 11：高低质量会员分布

### 5.5.4 综合分析

根据上述的三个影响条件进行综合性的考虑与分析可知：当任务难度系数一致时，价格越高，任务的完成度越高；当价格一致时，距离中心点越近且高质量会员的数目越多，则任务的完成度越高。即完成度与价格呈现正相关与距离和高质量会员数量呈现负相关。

## 六、问题二模型的建立与求解

### 6.1 数据预处理

针对附件一、二数据进行数据预处理，检查有无异常值，同问题一中异常值划分原理，经检验附件一中无异常值，附件二中具有明显偏差的会员位置信息，予以剔除。由此得到附件一任务已完成与未完成状况如下图所示：

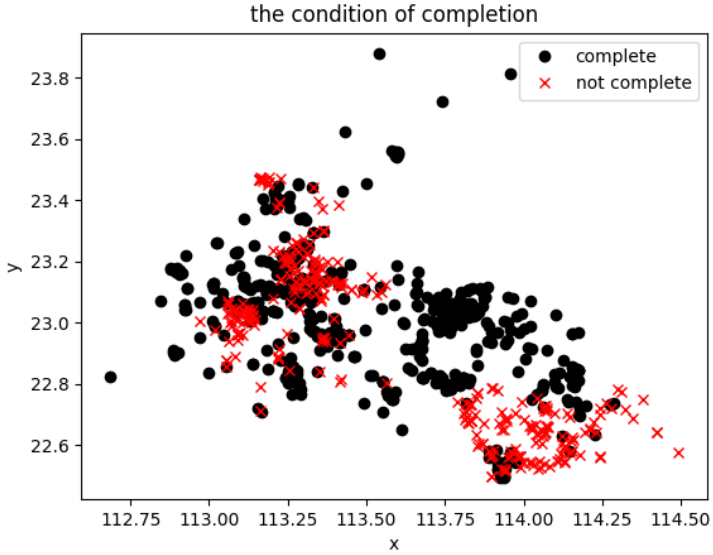


图 12：附件一任务位置与完成状况关系图

## 6.2 影响任务完成与未完成因素分析

任务完成与否受众多因素影响，在本问题中概括为 3 个一级指标，与 11 个二级指标。经定性分析，可知任务完成与否与以下因素有关：

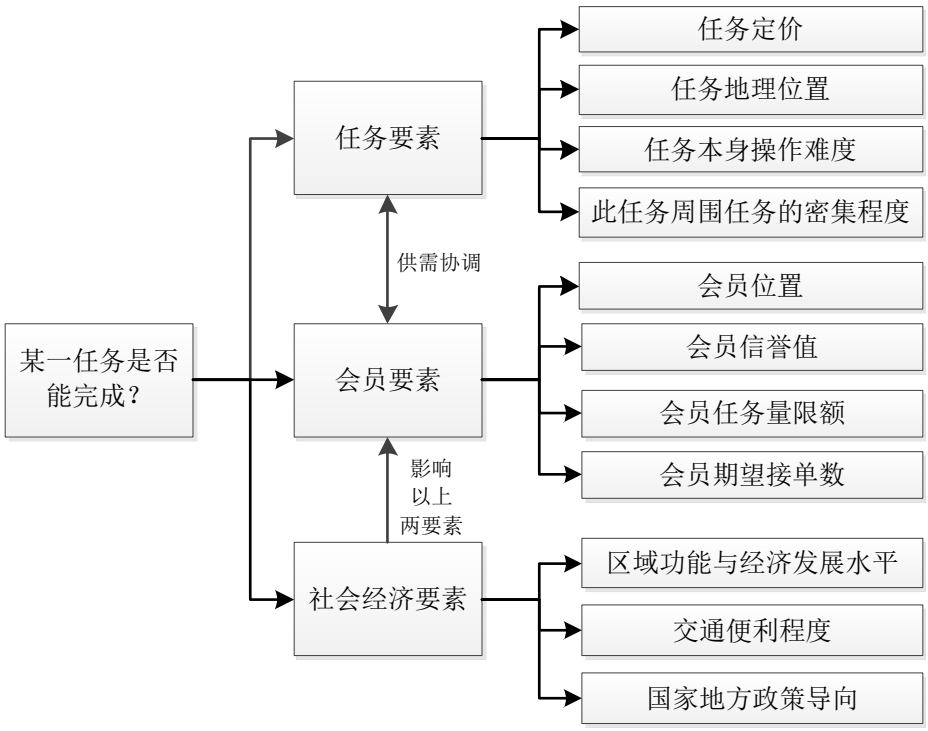


图 13：任务完成与否影响因素

## 6.3 影响因素标准化

经 6.2 定性分析知，以上因素对于任务完成与否均有影响，但由于不同影响因素的量纲不同，因此，需要对以上影响因素进行归一化处理，本文中均以极大—极小值法将不同影响因素处理成为无量纲的量。以便综合以上各项影响因素对各未完成任务点进行打分。

利用线性函数归一化公式，将数据转化为[0,1]范围内：

$$S_{normal} = \frac{S - S_{\min}}{S_{\max} - S_{\min}} \qquad (6\text{-}3\text{-}1)$$

其中：$S_{normal}$ 为各指标无量纲化后的数据，$S_{\min}$ 、$S_{\max}$ 分别为该指标中的极小、极大值通过此方法可以实现原有影响的等比例缩放，因此采用线性函数归一化公式。

## 6.4 基于熵权法的综合评价打分模型[2]

任务完成度与影响因素之间的关系难以量化确定。本文采用了一种相对评价法来衡量因素的影响情况。评价体系中各指标权重的确定一般包括主观赋权法和客观赋权法两种。主观赋权法主观随意性大，可靠性不足。因此，本文选择了主观性相对较小，能充分利用数据特征的熵值赋权法。可根据各项指标的变异度，利用信息熵计算出各指标权重的客观赋权法，信息熵越小，则表明指标的变异程度越大、重要程度越大。计算步骤如下：

Step1：计算标准化后的指标 $x'_{ij}$ 的比重 $R_{ij}$，其中：$R_{ij} = \frac{x'_{ij}}{\sum_{i=1}^{n} x'_{ij}}$。

Step2：计算第 $j$ 项指标的熵值 $e_{ij}$，其中：$e_{ij} = -\left(\frac{1}{\ln n}\right)\sum_{i=1}^{n} R_{ij} \ln R_{ij}$。

Step3：计算第 $j$ 项指标的差异性系数 $g_{jk}$，其中 $g_{jk} = 1 - e_{jk}$。当值 $g_j$ 越大，则指标 $x_j$ 在综合评价中的重要性越强。

Step4：计算指标 $x_j$ 的权数 $w_{jk}$，计算公式为 $w_{jk} = \frac{g_{jk}}{\sum_{i=1}^{n} g_{jk}}$，其中 $j = 1, 2, 3, \ldots$ $k = 1, 2, 3, \ldots$。

需要先确定各因素的权重并确定出各因素打分后的权数，进而利用熵权法确定出各层各指标的权重 $w_{jk}$。本文可以得到各因素情况的综合打分为：

$$Q_{\mathrm{i}} = \sum_{j \in J} \sum_{k \in K} x'_{ij} \cdot w_{ij} \cdot W_k \qquad (6\text{-}3\text{-}2)$$

式中：$x'_{ij}$ 为第 $i$ 点第 $j$ 项因素标准化后的得分；$w_{jk}$ 为第 $k$ 层下第 $j$ 个因素的权重系数；$w_k$ 为第 $k$ 层的权重系数。

至此，基于熵权法的综合评价打分模型建立完毕。

## 6.5 基于相似成功经验值价格学习定价模型

首先，提出假设。假设一：假设某未完成任务与，以该任务点为圆心，半径为 r 的小范围内的其他已完成任务在除价格以外的其他要素相似程度高。假设二：针对附件一数据，假设已完成的任务即定价合理，无需进行修改，未完成的任务定价存在不合理因素，需通过改变定价以提高任务完成的概率。任务未完成与已完成为二值状态，可以由 logistic 函数不断变形，以模拟未完成至已完成状态的改变。如下图所示：

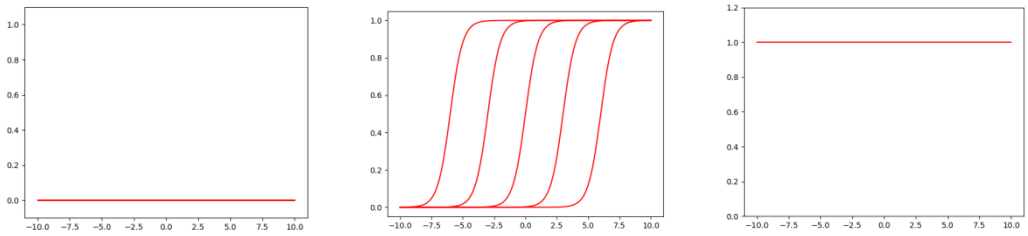

图 14：不同任务价格变动时完成状态变化曲线

以上变化过程原理为，部分任务在一定价格区间（[$price_{now}$，$price_{now}$+Scale]）其中 $price_{now}$ 现阶段定价，Scale 为允许价格提升范围）内，无论价格如何变化，都无法完成；部分任务可以通过调整价格，由未完成变为已完成；而其他任务在一定价格区间内均可以完成。因此验证以上假设二合理。

然后，基于以上两点假设，已完成的任务视为优秀经验，针对附件一中未完成的任务逐个分析：

Step1：选取 1 个未完成任务。

Step2：以该任务点为圆心，划定半径 r(此时 r 为一极小初始值)的圆形区域。

Step3：搜索该圆形区域内是否有已完成的任务视为优秀经验。

Step4：

Condition1：若划定区域内不存在已完成任务，则以一定拓展进度 i 拓展划定半径 r，即改变下一步划定半径为 $r := r + i$，并返回 Step2。设定划定半径 r 拓展的极限值 R，在 R 范围内假设一仍成立。若 r>R 时仍未寻得优秀经验，则不再学习，返回 Step1 寻找

下一未完成任务。

Condition2：若该圆形区域内存在 1 个已完成任务，则比较未完成任务与该已完成任务的评分差距，由于除价格外的其他因素差异较小且无法调整，此处不予考虑。然后，调整未完成任务价格使之评分接近与划定范围内已完成任务的评分。并返回 Step1 寻找下一未完成任务。

Condition3：若该圆形区域内存在 1 个以上已完成任务，由于这些任务点距离近且均为已完成状态，因此评分差距小，所以将范围内的多个已完成任务评分取平均值作为未完成任务的学习优秀经验。并进行 Condition2 后续步骤。

Step5：直至完成所有未完成任务的学习。

### 6.6 与原方案的比较

本文选取“完成度”这一指标对定价方案进行评价。

经以上分析，得到未完成任务的新定价。在此模型下，已完成任务价格没有改变，仍能够完成；未完成任务通过改变定价，完成的概率提高，部分未完成任务有可能通过此定价方案转化为已完成任务，项目中任务完成度较问题一定价情况下有所提高。因此通过以上模型确定新的定价优于附件一中原有定价。

对于经过此模型确定任务定价，可以将附件一提供的数据应用于此模型，得到新定价后进一步按照问题一思路确定定价规律即为定价方案。在实际 APP 运营过程中，可以依照当日以前一天至几天的数据进行分析，以确定当日定价，使定价更加科学、合理。

## 七、问题三模型的建立与求解

### 7.1 任务打包模型建立与求解

首先根据附件 1，将任务地理位置数据可视化。选择按照任务的经纬度分布进行打包合并工作。其次将全部任务点放入集合 $M_{\mathrm{i}}=\{M_1,M_2\cdots\}$ 中，选取其中一个任务点 $M_1$，以该任务点点为圆心，半径为适当阈值 t 画圆，将圆内任务存入集合 $N_1=\{M_a,M_b\cdots\}$ 中，并在全集 $M_i$ 中删除已打包的任务。最后遍历集合 $M_i$ 中剩余的任务点，形成存放多个任务的集合 $N_i$，完成打包过程。

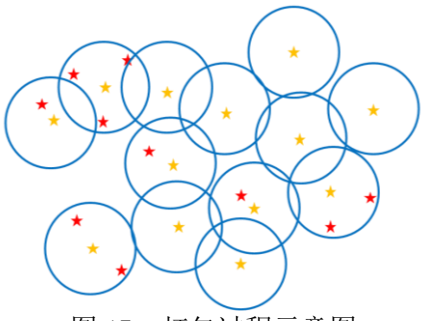

图 15：打包过程示意图

以上过程的算法描述为：

Step1：确定区域内所有需打包的任务点，并将所有任务点放入集合 $M_{\mathrm{i}}=\{M_1,M_2\cdots\}$，并初始化分组处理状态 condition 均为 0，初始化分组 group 为 0。

Step2：顺序选择分组处理状态 condition 为 0 的任务点，以该任务点点为圆心，半径为适当阈值 t 画圆，将圆内分组处理状态 condition 仍为 0 任务与该任务点分为一组，改变分组 group 为当前组数，改变分组处理状态 condition 为 1。并循环进行此步骤。

Step3：直至集合 $M_{\mathrm{i}} = \{M_1, M_2 \ldots\}$ 中全部任务点任务分组处理状态 condition 均为 1，结束循环。

根据附件 2 数据可知：

表 2：附件 2 描述性统计信息

| 会员数 | 预订任务限额均值 | 最小值 | 最大值 | 中位数 | 众数 |
|---|---|---|---|---|---|
| 1887 | 6.835375599 | 1 | 232 | 4 | 1 |

现需选择最优的阈值 t，使得所有包内的任务数均值与预订任务限额均值相近，提高打包效率和接单率，也使得更多的会员能够在限额内有接单能力。通过二分法反复验证，确定 t 值为 0.0555，打包的任务件数均值为 6.83606557377 件，共打包 122 组。t 为在平面二维坐标系下经度与纬度之间的距离，公式如下：

$$t = \sqrt{(x_1 - x_i)^2 + (y_1 - y_i)^2} \tag{7-1-1}$$

其中 $(x_1, y_1)$ 为圆区域中心点经纬度坐标，$(x_i, y_i)$ 为圆区域内其余点的经纬度坐标。

## 7.2 价格模型的建立与求解

首先选择第一个打包任务组 $N_1$，其中打包了 m 个任务，求得这 m 个位置的中心坐标（x,y），其中 $x = \frac{\sum_{i=1}^{m} x_i}{m}$，$y = \frac{\sum_{i=1}^{m} y_i}{m}$。然后在会员位置可视化图像上，以此中心为中点，t 为半径做圆，观察所包围的会员数目及其预定任务限额，判断在本任务组中是否存在能够接单的会员，并计算其个数。最后遍历所有任务组，统计其任务数及可接单会员数这两个数据。

分析表中数据，仅当包内任务数量为 39 件时，无接单会员。有接单会员的包数占总包数的比例超过了 97.30%。针对其余特殊情况，解释如下：剩余会员虽然有较高的限额，但因距离、交通、价格等因素而放弃接单。可就此特例减小 t 值，增多打包组数，减少同一组中任务件数，增大可接单会员人数百分比。

对于一个打包组，随着合并任务数量上升，可节省接单时间，提高接单效率，增加接单会员收益，即会员收益价格与打包任务数量呈正相关；随着区域内有接单能力的会员数量上升，竞争力增大，任务价格下降；即任务价格与会员数量呈负相关。

利用加权平均数，求得价格计算式如下：

$$price = \frac{\sum_{i=1}^{m} (\frac{\mathrm{i}^2}{n} \times priceper_i) + \sum_{j=1}^{o} (\frac{j^2}{n} \times priceper_j)}{n} \tag{7-2-1}$$

其中 m 是已完成任务个数，o 是未完成任务个数，n 是包中任务总个数。遍历所有包，完成价格（price）数据的统计工作。

如下表（部分，其余见附录 1）：

表 3：数据统计表

| 打包任务数/件 | 会员数/个 | 总定价/元 |
|---|---|---|
| 20 | 15 | 1315 |
| 14 | 30 | 924 |
| 15 | 30 | 1041.429 |
| 6 | 114 | 399 |
| 7 | 70 | 457.3333 |
| 2 | 360 | 144 |

| | | |
|---|---|---|
| 14 | 24 | 933.8 |

设一个包 $N_1$ 中任务数为 x，会员数为 y，其价格与任务数、会员数之间存在如下关系：

$$price = f(x)x + g(y)y + c \quad (7\text{-}2\text{-}2)$$

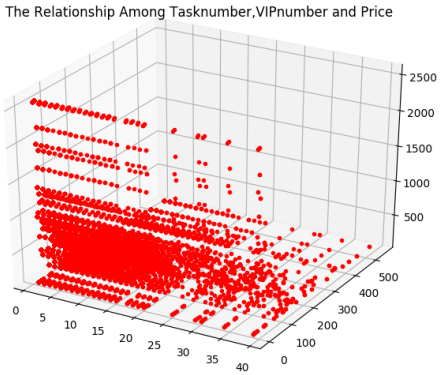


图 16：三维拟合离散点集

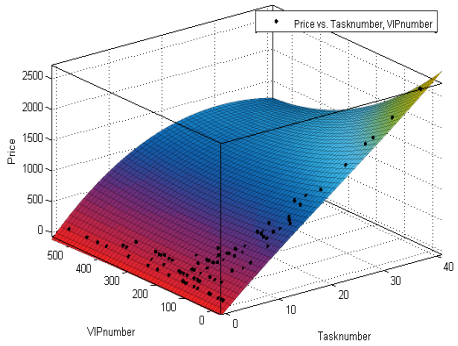


图 17：三维拟合光滑曲面

通过 MATLAB 三维拟合，将离散点拟合成符合多项式的光滑曲面，相关关系定量化可得：

$$price = p_{00} + p_{10}x + p_{01}y + p_{20}x^2 + p_{11}xy + p_{02}y^2 + p_{30}x^3 + p_{21}x^2y + p_{12}xy^2 \quad (7\text{-}2\text{-}3)$$

带入数值数据可得修改后的定价规律：

$$price = -1.204 + 76.62x + 0.03657y - 0.517x^2 - 0.03157xy - 0.0001187y^2 + 0.007248x^3 - 0.002333x^2y + 0.0001074xy^2 \quad (7\text{-}2\text{-}4)$$

表 5：拟合优度表

| 方差 | 决定系数 | 矫正后的决定系数 | 标准差 |
|---|---|---|---|
| 1.924e+04 | 0.9994 | 0.9993 | 14.23 |

由上表可知 $R^2$=0.9994，值较接近 1，回归图像对观测值的拟合程度较好。

### 7.3 新定价模式对任务完成的影响分析

由附录 1 数据知打包后的任务数、会员数、及总定价的统计数据。分析新定价模式对任务的影响程度可以转化为分析打包任务被接单的完成度。通过观察统计数据及分析会员接单心理，可规定一个打包任务在以 t=0.0555 为半径的辐射范围内，若存在 100 名及以上的有接单能力的会员，则此任务有接近 1 的完成概率。新定价模式下的完成度大于原定价模式，$\frac{92}{105}$=0.87619>0.6251，即新定价模式促使任务额完成度提高，呈现良性发展趋势。

## 八、问题四模型的建立与求解

### 8.1 新任务打包分组的模型建立与求解

根据附件 3 中给出的新任务的地理信息数据，对其进行可视化的处理工作。由可视化后得到的散点图可以发现：新任务的分布较为集中会导致会员争相选择。为解决这一问题，本文将采用与问题三类似的打包策略进行优化。首先本问题将任务的经纬度信息作为打包的基础。其次将附件 3 中全部的新任务点放入集合 $M_i=\{M_1,M_2...\}$ 中，任意选取其中一个任务点 $M_1$，并以该任务点的经纬度坐标作为圆心，取适当阈值 $k$ 作为半径画

圆。通过 Python 编程将圆内包含的任务进行编号并存入集合 $N_1 = \{M_a, M_b \ldots\}$ 中，并根据编号，在全集 $M_i$ 中删除已打包的任务。不断重复上述的步骤最即遍历集合 $M_i$ 中剩余的任务点，从而会形成存放多个任务的集合 $N_i$。此时，集合 $N_i$ 就为我们构建的打包分类的区域。

本过程算法与问题 3 相同此处不再赘述。

阈值 $k$ 的确定：

阈值 $k$ 是衡量打包过程中精度的重要指标，当 $k$ 值较大时，任务分组的数目 $n$ 会减小且内部包含的任务数量会增加；反之，当 $k$ 值较小时，任务分组的数目 $n$ 会增加但内部包含的任务数量会减少。

Step1：数据预处理。阈值 $k$ 的取值取决于会员的任务限额数量和任务的密集程度，所以需要对数据进行预处理：

表 6：会员任务限额统计表

| 会员数 | 预订任务限额均值 | 最小值 | 最大值 | 中位数 | 众数 |
|---|---|---|---|---|---|
| 1887 | 6.835375599 | 1 | 232 | 4 | 1 |

表 7：任务地理信息统计表

| 任务编号 | 任务 GPS 纬度 | 任务 GPS 经度 | 备注 |
|---|---|---|---|
| C0211 | 22.53470014 | 113.934485 | 纬度最小 |
| C1930 | 23.424401 | 113.3183627 | 纬度最大 |
| C1824 | 23.19821157 | 113.1471934 | 经度最小 |
| C1166 | 22.63892492 | 114.5130004 | 经度最大 |

根据公式：

$$\bar{x} = \frac{x_{\max} - x_{\min}}{n}, \quad \bar{y} = \frac{y_{\max} - y_{\min}}{n} \tag{8-1-1}$$

$$n = 2066$$

解得：$\bar{x} = 0.000430639332$，$\bar{y} = 0.000661087609$ 即平均每个相邻任务点之间的纬度差值为 0.000430639332，经度的差值为 0.000661087609。

根据会员的任务上限额度的平均值为 6.835375599，在 2066 个数据信息中使用二分法反复验证去寻求最合理的任务个数；经过多次模拟验证后，得出：当打包后每个包中的任务个数 N=22 时，使得打包效率和接单率更高。进一步推导可得：

$$k = 0.02$$

综上所述，阈值 $k$ 为 0.02。$k$ 直观表示为在平面二维坐标系下经度与纬度之间的距离，公式如下：

$$k = \sqrt{(x_1 - x_i)^2 + (y_1 - y_i)^2} \tag{8-1-2}$$

其中 $(x_1, y_1)$ 为圆区域中心点经纬度坐标，$(x_i, y_i)$ 为圆区域内其余点的经纬度坐标。

## 8.2 新任务价格的模型建立与求解

根据题意和前文数学模型可知，任务的价格受到任务的经纬度数据、会员的分布等因素影响。在本问题中，新数据的分布更为集中，其经纬度数据对于价格的影响函数发生了改变，单纯进行函数的修正并不能得出精准的函数关系和数学模型。为了使得新任务的价格更为准确，本文将建立相似学习的数学模型，以附件 1 中已知的数据作为基础，

让附件 3 中的任务通过学习，从而得出更为准确合理的价格。具体的建模步骤如下：

Step1：任务数据的预处理

①新数据的可视化处理。通过 Python 编程将新任务的经纬度数据转化为散点图等直观的表示形式。

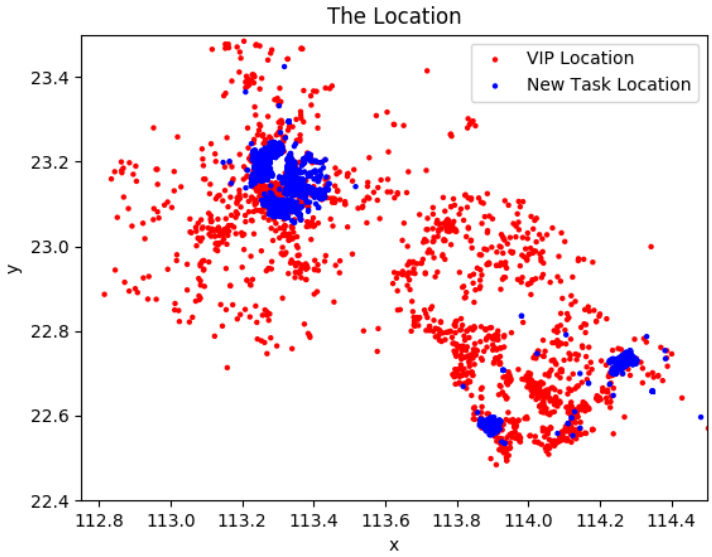


图 18：任务-会员分布图

②新数据的打包处理。将数据根据阈值 $k$ 的取值，进行集合化的处理即打包操作。具体步骤和算法见上文。

Step2：数据之间的比拟联系

将附件 1 的任务数据进行可视化的处理，通过 Python 编程将具体的经纬度数据转化为散点图等直观信息。将代表附件 1 的散点图与附件 3 的包分类后的散点图进行比拟。由于包为点的集合 $N$ ,则其内部包含多个点位，当与附件 1 比拟时，会将附件 1 的点划分为与包数量相同的点集。取其平均价格作为标准，则可得出每个包的总金额的数值。

具体的公式如下：

$$S = \pi \bullet k^2 \tag{8-2-1}$$

式中 S 表示每个包的面积，k 为打包时确定的阈值。该公式表明了每个包在散点图中的范围。

$$N = \{n_i \ldots n_o \ldots\} \tag{8-2-2}$$

式中 N 表示了附件 1 中点的部分集合，$n_i$、$n_o$ 等点包含于点集 N，且 $n_i$、$n_o$ 等点表示以某附件 3 中 S 范围内与附件 1 重合部分的点。

$$\overline{A} = \frac{a_n + a_{n+1} + \ldots + a_{n+m}}{m} \tag{8-2-3}$$

式中 $\overline{A}$ 表示在每个包内，附件 1 的点集的价格平均值。

$$A = \overline{A} \bullet n \tag{8-2-4}$$

式中 $A$ 表示该包的整体价格。$n$ 为该包中点的数量。

## 8.3 包内任务数量、具有完成能力的会员数和包价格之间的模型建立

Step1：计算包内能完成任务的会员数量

根据题意，首先应选择一个打包任务组表示为：$N_1$，设在 $N_1$ 中包含有 m 个任务。根据上述模型的描述和打包原理，可以求得这 m 个位置的中心坐标为（x,y）。

其中：$x = \frac{\sum_{i=1}^{m} x_i}{m}$ ，$y = \frac{\sum_{i=1}^{m} y_i}{m}$ （8-3-1）

然后在会员位置可视化图像上，以此中心为中点，阈值$k$为半径做圆，观察所包围的会员数目及其预定任务限额，判断在本任务组中是否存在能够接单的会员。最后遍历所有任务组，统计其任务数及可接单会员数这两个数据。

如下表（部分，其余见附录 2）

表 8：数据统计表

| 打包任务数/件 | 会员数/个 | 总定价/元 |
|---|---|---|
| 53 | 1 | 3711.472 |
| 50 | 1 | 3431.522 |
| 46 | 1 | 3209.486 |
| 58 | 1 | 4049.367 |
| 2 | 31 | 139.2308 |
| 21 | 5 | 1448.475 |
| 47 | 1 | 3281.383 |

根据题意和公司的运营情况可知，对于任意一个组团，随着其内部任务数量上升，可以使会员在接单时提高效率，增加接单的收益，即会员收益与打包任务数量呈正相关；随着区域内有接单能力的会员数量上升，竞争力增大，任务价格下降；即任务价格与接单会员的数量呈负相关。

由上一步骤可知，每个包的整体定价方式是通过学习模拟的方法来确定的，其价格表示为：$A$

学习目标中的价格平均化：

$$\overline{A} = \frac{a_n + a_{n+1} + ... + a_{n+m}}{m} \quad (8\text{-}3\text{-}2)$$

式中$\overline{A}$表示在每个包内，附件 1 的点集的价格平均值。

$$A = \overline{A} \bullet n \quad (8\text{-}3\text{-}3)$$

式中$A$表示该包的整体价格。$n$为该包中点的数量。

通过 Python 编程，便利所有包，得出每个包的任务数量、会员个数（有能力接单的）以及包的整体价格。具体数据如下表所示：

表 8：数据统计表

| 打包任务数/件 | 会员数/个 | 总定价/元 |
|---|---|---|
| 53 | 1 | 3711.472 |
| 50 | 1 | 3431.522 |
| 46 | 1 | 3209.486 |
| 58 | 1 | 4049.367 |
| 2 | 31 | 139.2308 |
| 21 | 5 | 1448.475 |
| 47 | 1 | 3281.383 |

假设一个包$N_1$，其中包含有$x$个任务，在包内能完成任务的会员数目为$y$；根据上述的关系，不妨设有：

$$\text{price} = f(x)x + g(y)y + c \quad (8\text{-}3\text{-}4)$$

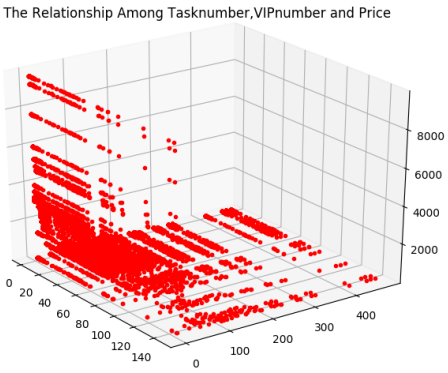


图 19：三维拟合离散点集

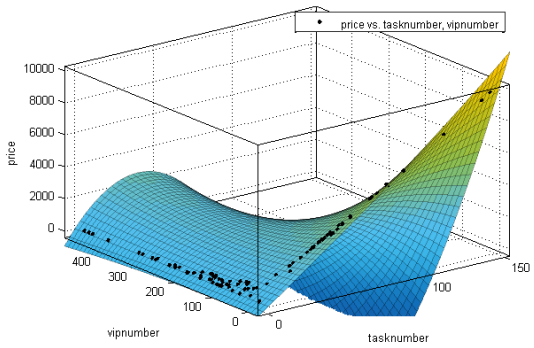


图 20：三维拟合光滑曲面

通过 MATLAB 三维拟合，将离散点拟合成光滑曲面，将三者关系进行定量量化分析：

$$price = p_{00} + p_{10}x + p_{01}y + p_{20}x^2 + p_{11}xy + p_{02}y^2 + p_{30}x^3 + p_{21}x^2y + p_{12}xy^2$$

（8-3-5）

代入上述过程中得出的数据，进行计算，从而得出定价规律为：

$$price = 1.748 + 70.83x + 0.04777y - 0.04682x^2 - 0.07542xy - 0.0003519y^2 + 0.0001368x^3 - 0.003179x^2y + 0.0007037xy^2$$

（8-3-6）

表 9：拟合优度表

| 方差 | 决定系数 | 矫正后的决定系数 | 标准差 |
|---|---|---|---|
| 1.3481e+04 | 1.0000 | 1.0000 | 12.7443 |

由上表可知 $R^2 = 1.0000$，所以回归图像对观测值的拟合程度较好。

### 8.4 新定价方案的实施效果

由附录 2 可得打包后的任务数、会员数、及总定价的统计数据。分析新定价模式对任务的影响程度可以转化为分析打包任务被接单的完成度。通过观察统计数据及分析会员接单心理，可规定一个打包任务在以 k=0.02 为半径的辐射范围内，若存在 40 名及以上的有接单能力的会员，则此任务有接近 1 的完成概率。新定价模式下的完成度大于原定价模式 $\frac{74}{93} = 0.7957 > 0.6251$，即新定价模式促使任务额完成度提高，呈现良性发展趋势。

## 九、模型的评价与推广

问题一建立了基于 K-中心聚类法的确定中心模型，该模型具有可靠性并且客观、全面的结合了影响定价的任务位置和会员位置因素，并构建了中心修正模型对得到的两个中心进行修正，确保后续的定价函数是基于此中心位置拟合的，提高了函数准确性和精度。

问题二建立熵权法综合评价打分模型，能够较为客观、量化地衡量影响完成度的三层指标，并判断因素的重要程度。建立优秀经验学习模型能够动态自主地改变为完成任务定价，使其完成概率趋近于 1，获得更优方案。

问题三建立任务打包模型，将距离范围内邻近的散点圈成多个“包”，保证了不遗漏、不重复计数。建立价格模型，寻找定价与任务数、会员数之间的关系，利用三维拟合方法得到定价函数，将定性问题转化为定量，完成模型的构架与求解过程。

问题四与问题三模型类似，此处不再赘述。

本文所建立的多种模型具有创新性，实际实用性强，充分利用了附件各数据及指标因素，从它们之间的性质与数量方面都做出了分析与评价，为不同的项目及操作方式提供了良好的定价方案，具有较广的适用范围。

## 十、参考文献

[1]谢娟英,郭文娟,谢维信. 基于邻域的 K 中心点聚类算法[J]. 陕西师范大学学报(自然科学版),2012,40(04):16-22. [2017-09-16]. DOI：10.15983/j.cnki.jsnu.2012.04.001

[2]姜君. 基于熵权与变异系数组合赋权法的模糊综合评价模型[D].首都师范大学,2011.

# 附 录

附录 1

| 打包任务数 | 会员数 | 总定价 |
|---|---|---|
| 20 | 15 | 1315 |
| 14 | 30 | 924 |
| 15 | 30 | 1041.429 |
| 6 | 114 | 399 |
| 7 | 70 | 457.3333 |
| 2 | 360 | 144 |
| 14 | 24 | 933.8 |
| 7 | 92 | 501.9 |
| 12 | 24 | 798 |
| 2 | 146 | 150 |
| 29 | 21 | 1904.333 |
| 2 | 159 | 150 |
| 5 | 112 | 355 |
| 9 | 60 | 600 |
| 34 | 19 | 2221.333 |
| 2 | 371 | 141 |
| 1 | 545 | 73 |
| 19 | 44 | 1254.633 |
| 11 | 80 | 720.5 |
| 25 | 16 | 1669.643 |
| 4 | 113 | 288 |
| 39 | 15 | 2584.636 |
| 9 | 72 | 612 |
| 13 | 63 | 915.0556 |
| 2 | 122 | 150 |
| 17 | 48 | 1216.563 |
| 12 | 71 | 786 |
| 7 | 16 | 509.5 |
| 3 | 117 | 213.5 |
| 6 | 22 | 424 |
| 20 | 11 | 1402.941 |
| 1 | 44 | 75 |
| 14 | 10 | 1050 |
| 2 | 271 | 134.5 |
| 2 | 102 | 140.5 |
| 1 | 192 | 75 |
| 5 | 170 | 343.125 |
| 12 | 63 | 802.6667 |
| 14 | 7 | 1000 |
| 1 | 18 | 70 |
| 1 | 0 | 70 |
| 1 | 8 | 75 |
| 1 | 0 | 75 |
| 6 | 92 | 413.5 |
| 1 | 103 | 75 |

| | | |
|---|---|---|
| 2 | 108 | 146 |
| 1 | 335 | 75 |
| 1 | 51 | 80 |
| 39 | 10 | 2574 |
| 30 | 13 | 2004.5 |
| 13 | 23 | 900.5 |
| 12 | 29 | 867 |
| 11 | 45 | 774.5 |
| 11 | 33 | 773.5 |
| 3 | 229 | 217 |
| 8 | 85 | 549.5 |
| 8 | 62 | 546.5 |
| 7 | 42 | 466 |
| 3 | 257 | 216.5 |
| 2 | 168 | 133.5 |
| 8 | 89 | 564 |
| 7 | 98 | 486 |
| 6 | 110 | 461.5 |
| 17 | 27 | 1193.967 |
| 10 | 11 | 715 |
| 7 | 131 | 506 |
| 2 | 109 | 150 |
| 1 | 50 | 75 |
| 5 | 44 | 369.5 |
| 11 | 67 | 809.6 |
| 5 | 96 | 394 |
| 1 | 90 | 73 |
| 3 | 140 | 220.5 |
| 1 | 133 | 75 |
| 11 | 53 | 810.7 |
| 1 | 8 | 75 |
| 7 | 73 | 500 |
| 2 | 167 | 142 |
| 3 | 217 | 213 |
| 1 | 413 | 72 |
| 2 | 228 | 140.5 |
| 3 | 277 | 230 |
| 4 | 146 | 297 |
| 3 | 85 | 225 |
| 5 | 126 | 356.5 |
| 5 | 108 | 352.5 |
| 5 | 132 | 354.1667 |
| 3 | 264 | 213.75 |
| 2 | 373 | 150 |
| 3 | 44 | 225 |
| 3 | 67 | 214.5 |
| 3 | 147 | 218 |
| 1 | 444 | 72.5 |
| 3 | 245 | 245 |
| 3 | 150 | 205.5 |
| 1 | 129 | 75 |

| | | |
|---|---|---|
| 1 | 482 | 75 |
| 2 | 176 | 157 |
| 1 | 266 | 75 |
| 1 | 143 | 85 |
| 2 | 100 | 140 |
| 3 | 346 | 238 |
| 1 | 233 | 80 |

附录 2

| 打包任务数 | 会员数 | 总定价 |
|---|---|---|
| 53 | 1 | 3711.472 |
| 50 | 1 | 3431.522 |
| 46 | 1 | 3209.486 |
| 58 | 1 | 4049.367 |
| 2 | 31 | 139.2308 |
| 21 | 5 | 1448.475 |
| 47 | 1 | 3281.383 |
| 5 | 50 | 349.9242 |
| 4 | 55 | 279.2414 |
| 1 | 55 | 68.5 |
| 1 | 291 | 68.52667 |
| 141 | 1 | 9367.304 |
| 82 | 2 | 5448.176 |
| 1 | 291 | 66.72131 |
| 3 | 110 | 199.3261 |
| 5 | 78 | 337.7692 |
| 6 | 39 | 401.6557 |
| 1 | 198 | 66.36111 |
| 7 | 35 | 464.9545 |
| 2 | 165 | 132.825 |
| 3 | 94 | 199.3154 |
| 1 | 229 | 68.89474 |
| 3 | 99 | 199.4516 |
| 92 | 2 | 6110.418 |
| 11 | 24 | 727.6276 |
| 76 | 5 | 5039.914 |
| 26 | 18 | 1720.985 |
| 48 | 12 | 3180.686 |
| 36 | 13 | 2380.815 |
| 82 | 6 | 5426.77 |
| 24 | 20 | 1591.217 |
| 53 | 11 | 3508.902 |
| 41 | 14 | 2714.542 |
| 4 | 139 | 265.3871 |
| 43 | 14 | 2846.19 |
| 7 | 80 | 467.1494 |
| 4 | 116 | 264.2692 |
| 2 | 141 | 132.45 |
| 1 | 410 | 66.27778 |
| 5 | 50 | 331.0065 |

| | | |
|---|---|---|
| 4 | 124 | 265.6757 |
| 46 | 11 | 3078.39 |
| 41 | 12 | 2731.92 |
| 34 | 14 | 2252.053 |
| 24 | 18 | 1585.899 |
| 4 | 169 | 265.0072 |
| 117 | 1 | 7811.213 |
| 73 | 6 | 4842.493 |
| 8 | 67 | 533.0514 |
| 17 | 26 | 1136.276 |
| 5 | 99 | 331 |
| 2 | 280 | 133.8013 |
| 5 | 151 | 338.4593 |
| 37 | 13 | 2502.693 |
| 44 | 11 | 2945.72 |
| 146 | 0 | 9773.631 |
| 43 | 11 | 2888.847 |
| 43 | 8 | 2879.656 |
| 73 | 7 | 4839.227 |
| 30 | 15 | 2014.557 |
| 72 | 7 | 4786.917 |
| 5 | 147 | 335.4265 |
| 8 | 61 | 529.7662 |
| 5 | 137 | 338.602 |
| 38 | 13 | 2520.542 |
| 2 | 182 | 132.7265 |
| 7 | 61 | 468.9727 |
| 1 | 282 | 68.66554 |
| 2 | 233 | 133.5259 |
| 2 | 234 | 136.3467 |
| 2 | 221 | 132.7581 |
| 1 | 75 | 70.4375 |
| 1 | 327 | 66.43798 |
| 1 | 117 | 68.30303 |
| 1 | 474 | 66.71186 |
| 1 | 105 | 69.71875 |
| 2 | 184 | 136.4848 |
| 1 | 161 | 69.4186 |
| 1 | 317 | 67.92029 |
| 1 | 245 | 67.90196 |
| 1 | 110 | 71.06863 |
| 1 | 244 | 67.34314 |
| 1 | 3 | 78.33333 |
| 1 | 162 | 69.20513 |
| 1 | 2 | 78.33333 |
| 1 | 194 | 67.02041 |
| 2 | 89 | 142.6471 |
| 4 | 134 | 265.4872 |
| 1 | 452 | 66.84731 |
| 1 | 462 | 66.88068 |
| 2 | 138 | 138.8485 |

| | | |
|---|---|---|
| 1 | 128 | 67.85542 |

## 附录 3 问题一代码

```
###支撑材料文件名：第一问代码.py
########第一问代码########
#以下代码在 jupyter notebook 交互式环境下得到
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
data=pd.read_excel('attach1.xls')

#data1 为一团块数据且价格<=75
data1=data[((data.xlong)<=113.55)&((data.xlong)>=112.75)&((data.yla)>=22.7)&((data.yla)<=23.5)&((dat
a.price)<=75)]
'''
plt.scatter(data1.loc[:, 'xlong'], data1.loc[:, 'yla'],s=10,
c=data1.loc[:,'price'], edgecolor='none', alpha=1,
cmap=plt.cm.get_cmap('nipy_spectral', 10))
plt.colorbar();
plt.xlabel('x')
plt.ylabel('y')
plt.show()
'''
#任务位置导向中心
a65=data1[data1.price==65].shape[0]
a65b=data1[data1.price==65.5].shape[0]
a66=data1[data1.price==66].shape[0]
a66b=data1[data1.price==66.5].shape[0]
a67=data1[data1.price==67].shape[0]
a67b=data1[data1.price==67.5].shape[0]
a68=data1[data1.price==68].shape[0]
a68b=data1[data1.price==68.5].shape[0]
a69=data1[data1.price==69].shape[0]
a69b=data1[data1.price==69.5].shape[0]
a70=data1[data1.price==70].shape[0]
a70b=data1[data1.price==70.5].shape[0]
a71=data1[data1.price==71].shape[0]
a71b=data1[data1.price==71.5].shape[0]
a72=data1[data1.price==72].shape[0]
a72b=data1[data1.price==72.5].shape[0]
a73=data1[data1.price==73].shape[0]
a73b=data1[data1.price==73.5].shape[0]
a74=data1[data1.price==74].shape[0]
a74b=data1[data1.price==74.5].shape[0]
a75=data1[data1.price==75].shape[0]
xlong=np.array(data1.loc[:,'xlong'])
yla=np.array(data1.loc[:,'yla'])
price=np.array(data1.loc[:,'price'])
alldistance=[]
for i,j in zip(xlong,yla):
    distance=0
    for p,q in zip(xlong,yla):
        distance1=np.sqrt((i-p)*(i-p)+(j-q)*(j-q))
        distance=distance+distance1
    alldistance.append(distance)
```

```
alldistance=np.array(alldistance)
df2=pd.DataFrame({'number':np.arange(alldistance.size),
                  'xlong':xlong,
                  'yla':yla,
                  'distance':alldistance})
df2=df2.sort_values(by='distance')
'''
plt.scatter(data1.loc[:, 'xlong'], data1.loc[:, 'yla'],s=10,
c=data1.loc[:,'price'], edgecolor='none', alpha=1,
cmap=plt.cm.get_cmap('nipy_spectral', 10))
plt.colorbar();
'''
center1x=113.268162#df2 首行数据粘贴而来
center1y=23.114054#df2 首行数据粘贴而来
'''
plt.plot(center1x,center1y,'o',c='y')
plt.show()
'''
#任务导向圈层扩散
l=112.75
r=113.55
d=22.7
u=23.5
jrl=[]
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
    c=0
    for p,q,m in zip(xlong,yla,price):
        if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==65:
            c=c+1
    if c/a65>0.4:
        break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
    c=0
    for p,q,m in zip(xlong,yla,price):
        if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==65.5:
            c=c+1
    if c/a65b>0.4:
        break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
    c=0
    for p,q,m in zip(xlong,yla,price):
        if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==66:
            c=c+1
    if c/a66>0.4:
        break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
    c=0
    for p,q,m in zip(xlong,yla,price):
        if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==66.5:
            c=c+1
    if c/a66b>0.4:
        break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
```

```
    c=0
    for p,q,m in zip(xlong,yla,price):
        if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==67:
            c=c+1
    if c/a67>0.4:
        break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
    c=0
    for p,q,m in zip(xlong,yla,price):
        if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==67.5:
            c=c+1
    if c/a67b>0.4:
        break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
    c=0
    for p,q,m in zip(xlong,yla,price):
        if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==68:
            c=c+1
    if c/a68>0.4:
        break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
    c=0
    for p,q,m in zip(xlong,yla,price):
        if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==68.5:
            c=c+1
    if c/a68b>0.4:
        break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
    c=0
    for p,q,m in zip(xlong,yla,price):
        if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==69:
            c=c+1
    if c/a69>0.4:
        break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
    c=0
    for p,q,m in zip(xlong,yla,price):
        if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==69.5:
            c=c+1
    if c/a69b>0.4:
        break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
    c=0
    for p,q,m in zip(xlong,yla,price):
        if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==70:
            c=c+1
    if c/a70>0.4:
        break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
    c=0
```

```
    for p,q,m in zip(xlong,yla,price):
        if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==70.5:
            c=c+1
    if c/a70b>0.4:
        break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
    c=0
    for p,q,m in zip(xlong,yla,price):
        if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==71:
            c=c+1
    if c/a71>0.4:
        break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
    c=0
    for p,q,m in zip(xlong,yla,price):
        if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==71.5:
            c=c+1
    if c/a71b>0.4:
        break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
    c=0
    for p,q,m in zip(xlong,yla,price):
        if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==72:
            c=c+1
    if c/a72>0.4:
        break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
    c=0
    for p,q,m in zip(xlong,yla,price):
        if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==72.5:
            c=c+1
    if c/a72b>0.4:
        break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
    c=0
    for p,q,m in zip(xlong,yla,price):
        if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==73:
            c=c+1
    if c/a73>0.4:
        break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
    c=0
    for p,q,m in zip(xlong,yla,price):
        if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==73.5:
            c=c+1
    if c/a73b>0.4:
        break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
    c=0
    for p,q,m in zip(xlong,yla,price):
```

```
            if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==74:
                c=c+1
        if c/a74>0.4:
            break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
        c=0
        for p,q,m in zip(xlong,yla,price):
            if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==74.5:
                c=c+1
        if c/a74b>0.4:
            break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
        c=0
        for p,q,m in zip(xlong,yla,price):
            if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==75:
                c=c+1
        if c/a75>0.4:
            break
jrl.append(i)
#拟合与回归
jrlx=np.arange(65,75.5,0.5)
from scipy.optimize import leastsq
def func(p,x):
        k,b=p
        return k*x+b

def error(p,x,y,s):
        print (s)
        return func(p,x)-y

#TEST
p0=[100,2]
#print( error(p0,Xi,Yi) )


s="Test the number of iteration"
Para=leastsq(error,p0,args=(jrlx,jrl,s))
k,b=Para[0]
print("k=",k,'\n',"b=",b)

y=k*jrlx+b
'''
plt.plot(jrlx,y,color="orange",label="Fitting Line",linewidth=2)

plt.plot(jrlx,jrl,'o',c='g')
plt.xlabel("price")
plt.ylabel("R")
plt.show()
'''
#高质量会员导向中心选择
iddata=pd.read_excel('attach2.xlsx')
iddata1=iddata[((iddata.id_x)<=113.55)&((iddata.id_x)>=112.75)&((iddata.id_y)>=22.7)&((iddata.id_y)<=
23.5)&((iddata.count2>100))]
#一组团范围内的会员
id_x=np.array(iddata1.loc[:,'id_x'])
```

```
id_y=np.array(iddata1.loc[:,'id_y'])
count2=np.array(iddata1.loc[:,'count2'])
alliddistance=[]
for i,j in zip(id_x,id_y):
    iddistance=0
    for p,q in zip(id_x,id_y):
        iddistance=iddistance+np.sqrt((i-p)*(i-p)+(j-q)*(j-q))
    alliddistance.append(iddistance)
alliddistance=np.array(alliddistance)
df3=pd.DataFrame({'number':np.arange(alliddistance.size),
                  'id_x':id_x,
                  'id_y':id_y,
                  'iddistance':alliddistance})
df3=df3.sort_values(by='iddistance')
'''
plt.scatter(data1.loc[:, 'xlong'], data1.loc[:, 'yla'],s=10,
c=data1.loc[:,'price'], edgecolor='none', alpha=1,
cmap=plt.cm.get_cmap('nipy_spectral', 10))
plt.colorbar();
'''
center1x=113.268162#df2 首行数据粘贴而来
center1y=23.114054#df2 首行数据粘贴而来
idcenter1x=113.288213#df3 首行数据粘贴而来
idcenter1y=23.133346#df3 首行数据粘贴而来
'''
plt.plot(center1x,center1y,'o',c='y')
plt.plot(idcenter1x,idcenter1y,'o',c='g')
plt.show()
'''
l=112.75
r=113.55
d=22.7
u=23.5
jrl=[]
#中心修正
center1x=(center1x+idcenter1x)/2
center1y=(center1y+idcenter1y)/2
#修正中心圈层外扩
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
    c=0
    for p,q,m in zip(xlong,yla,price):
        if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==65:
            c=c+1
    if c/a65>0.4:
        break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
    c=0
    for p,q,m in zip(xlong,yla,price):
        if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==65.5:
            c=c+1
    if c/a65b>0.4:
        break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
    c=0
    for p,q,m in zip(xlong,yla,price):
```

```
            if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==66:
                c=c+1
        if c/a66>0.4:
            break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
        c=0
        for p,q,m in zip(xlong,yla,price):
            if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==66.5:
                c=c+1
        if c/a66b>0.4:
            break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
        c=0
        for p,q,m in zip(xlong,yla,price):
            if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==67:
                c=c+1
        if c/a67>0.4:
            break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
        c=0
        for p,q,m in zip(xlong,yla,price):
            if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==67.5:
                c=c+1
        if c/a67b>0.4:
            break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
        c=0
        for p,q,m in zip(xlong,yla,price):
            if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==68:
                c=c+1
        if c/a68>0.4:
            break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
        c=0
        for p,q,m in zip(xlong,yla,price):
            if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==68.5:
                c=c+1
        if c/a68b>0.4:
            break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
        c=0
        for p,q,m in zip(xlong,yla,price):
            if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==69:
                c=c+1
        if c/a69>0.4:
            break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
        c=0
        for p,q,m in zip(xlong,yla,price):
            if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==69.5:
```

```
                c=c+1
        if c/a69b>0.4:
            break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
        c=0
        for p,q,m in zip(xlong,yla,price):
            if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==70:
                c=c+1
        if c/a70>0.4:
            break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
        c=0
        for p,q,m in zip(xlong,yla,price):
            if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==70.5:
                c=c+1
        if c/a70b>0.4:
            break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
        c=0
        for p,q,m in zip(xlong,yla,price):
            if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==71:
                c=c+1
        if c/a71>0.4:
            break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
        c=0
        for p,q,m in zip(xlong,yla,price):
            if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==71.5:
                c=c+1
        if c/a71b>0.4:
            break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
        c=0
        for p,q,m in zip(xlong,yla,price):
            if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==72:
                c=c+1
        if c/a72>0.4:
            break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
        c=0
        for p,q,m in zip(xlong,yla,price):
            if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==72.5:
                c=c+1
        if c/a72b>0.4:
            break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
        c=0
        for p,q,m in zip(xlong,yla,price):
            if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==73:
                c=c+1
```

```
        if c/a73>0.4:
            break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
        c=0
        for p,q,m in zip(xlong,yla,price):
            if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==73.5:
                c=c+1
        if c/a73b>0.4:
            break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
        c=0
        for p,q,m in zip(xlong,yla,price):
            if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==74:
                c=c+1
        if c/a74>0.4:
            break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
        c=0
        for p,q,m in zip(xlong,yla,price):
            if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==74.5:
                c=c+1
        if c/a74b>0.4:
            break
jrl.append(i)
for i in (np.linspace(0,max(r-center1x,center1x-l,u-center1y,center1y-d),1000)):
        c=0
        for p,q,m in zip(xlong,yla,price):
            if np.sqrt((p-center1x)*(p-center1x)+(q-center1y)*(q-center1y))<i and m==75:
                c=c+1
        if c/a75>0.4:
            break
jrl.append(i)

jrlx=np.arange(65,75.5,0.5)
from scipy.optimize import leastsq
def func(p,x):
        k,b=p
        return k*x+b

def error(p,x,y,s):
        print (s)
        return func(p,x)-y

#TEST
p0=[100,2]
#print( error(p0,Xi,Yi) )

###主函数从此开始###
s="Test the number of iteration"
Para=leastsq(error,p0,args=(jrlx,jrl,s))
k,b=Para[0]
print("k=",k,'\n',"b=",b)

y=k*jrlx+b
```

```
plt.plot(jrlx,y,color="orange",label="Fitting Line",linewidth=2)

plt.plot(jrlx,jrl,'o',c='g')
plt.xlabel("price")
plt.ylabel("R")
plt.show()
```

## 附录 4 问题一部分绘图代码

```
####支撑材料文件名：第一问部分绘图代码.py
#########第一问绘图代码###########
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
data=pd.read_excel('attach1.xls')
yla=data.loc[:,'yla']
xlong=data.loc[:,"xlong"]
price=data.loc[:,'price']
data2=data.sort_values(by='price')
pricesort=data2.loc[:,'price']
pricesort=set(pricesort)
iddata=pd.read_excel("attach2.xlsx")
'''
iddata2=iddata[(iddata.count2>=0)]
plt.plot(iddata2.loc[:,'id_x'],iddata2.loc[:,'id_y'],'*',c='r',label='VIP')
data3=data2[(data2.price>=65)]
plt.plot(data3.iloc[:,2],data3.iloc[:,1],'.',c='b',label='Task')
plt.xlim((112.5,114.8))
plt.ylim((22.4,23.8))
plt.legend()
plt.show()
'''
iddata2=iddata[(iddata.count2>=100)]

iddata3=iddata[(iddata.count2<100)]
plt.plot(iddata3.loc[:,'id_x'],iddata3.loc[:,'id_y'],'.',c='b',label='Low Quality VIP')
plt.plot(iddata2.loc[:,'id_x'],iddata2.loc[:,'id_y'],'*',c='r',label='High Quality VIP')
plt.xlim((112.5,114.8))
plt.ylim((22.4,23.8))
plt.legend()
plt.show()
```

## 附录 5 问题三代码

```
#####支撑材料文件名：第三问代码.py
#########第三问代码#########
#最佳半径判断与分组
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
centerx=[]
centery=[]
people=[]
pr=[]
data=pd.read_excel('attach1.xls')
xlong=np.array(data.xlong)
yla=np.array(data.yla)
price=np.array(data.price)
```

```
number=np.array(np.arange(835))
m=0
for i in range(835):
    if data.iloc[i,6]==0:
        data.iloc[i,5]=m
        data.iloc[i,6]=1
        for j in range(835):
            Td=0.0555
            if i==j:
                distance=100000000
            else:
                distance=np.sqrt((xlong[j]-xlong[i])*(xlong[j]-xlong[i])+(yla[j]-yla[i])*(yla[j]-yla[i]))
            if distance<=Td and data.iloc[j,6]==0:
                data.iloc[j,6]=1
                data.iloc[j,5]=m
        m=m+1
    else:
        continue
data=data.sort_values(by='group')
print(data)
number=[]
#统计每个包内任务数、会员数以及依据附件一得到的包的总价格
for i in range (max(data.group)):
    numbernow=data.group[data['group']==i].count()
    number.append(numbernow)
#number=np.array(number)
print(number)
#print('mean=',number.mean())
#print('max=',number.max())
for i in range (len(number)):
    centerx.append(data.xlong[data.group==i].mean())
    centery.append(data.yla[data.group==i].mean())
iddata=pd.read_excel('attach2.xlsx')
for i in range (len(number)):
    qdata=iddata[iddata.limit>=number[i]]
    q2data=qdata[((qdata.id_x-centerx[i])**2+(qdata.id_y-centery[i])**2)<=Td]
    peoplenow=q2data.id_x.count()
    people.append(peoplenow)
print("people_number=",people)

for i in range (len(number)):
    #if data.group[(data.group==i)&(data.complete==1)].count!=0:
    #pr.append((data.price[(data.group==i)&(data.complete==1)].mean())*number[i])
    pr.append((data.price[(data.group==i)].mean())*number[i])
    #else:
        #pr.append(-1)
print('group_all_price=',pr)

prnew=[]
numbernew=[]
peoplenew=[]
positionofselect=[]
for i in range(len(pr)):
    if pr[i] >0:
        positionofselect.append(i)
for j in positionofselect:
    prnew.append(pr[j])
```

```
        numbernew.append(number[j])
        peoplenew.append(people[j])

#三维图

from mpl_toolkits.mplot3d import Axes3D
fig=plt.figure()
ax=Axes3D(fig)
X=np.array(numbernew)
Y=np.array(peoplenew)
Z=np.array(prnew)
X, Y = np.meshgrid(X, Y)
ax.plot(X,Y,Z,'.',c='r')
ax.set_xlim=(0,40)
ax.set_title('The Relationship Among Tasknumber,VIPnumber and Price')
ax.set_xlabel=('Tasknumber')
ax.set_ylabel=('VIPnumber')
ax.set_zlabel=('Price')
plt.show()
print("Tasknumber=",numbernew)
print("VIPnumber=",peoplenew)
print("Price=",prnew)
```

## 附录 6 问题四代码

```
####支撑材料文件名：第四问代码.py
########第四问代码########
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
newdata=pd.read_excel('attach3.xls')
data=pd.read_excel('attach1.xls')

xnlong=np.array(newdata.xnlong)
ynla=np.array(newdata.ynla)
number=np.array(np.arange(2066))
m=0
for i in range(2066):
    if newdata.iloc[i,4]==0:
        newdata.iloc[i,3]=m
        newdata.iloc[i,4]=1
        for j in range(2066):
            Td=0.02
            if i==j:
                distance=100000000
            else:

distance=np.sqrt((xnlong[j]-xnlong[i])*(xnlong[j]-xnlong[i])+(ynla[j]-ynla[i])*(ynla[j]-ynla[i]))
            if distance<=Td and newdata.iloc[j,4]==0:
                newdata.iloc[j,4]=1
                newdata.iloc[j,3]=m
        m=m+1
    else:
        continue
newdata=newdata.sort_values(by='group3')
#print(newdata)

number=[]
```

```
#统计每个包内任务数、会员数以及依据附件一学习价格
for i in range (max(newdata.group3)):
    numbernow=newdata.group3[newdata['group3']==i].count()
    number.append(numbernow)
number=np.array(number)
print(number)
print('mean=',number.mean())
print('max=',number.max())
centerx=[]
centery=[]
people=[]
pr=[]
for i in range (len(number)):
    centerx.append(newdata.xnlong[newdata.group3==i].mean())
    centery.append(newdata.ynla[newdata.group3==i].mean())
iddata=pd.read_excel('attach2.xlsx')
for i in range (len(number)):
    qdata=iddata[iddata.limit>=number[i]]
    q2data=qdata[((qdata.id_x-centerx[i])**2+(qdata.id_y-centery[i])**2)<=Td]
    peoplenow=q2data.id_x.count()
    people.append(peoplenow)
print("people_number=",people)

for i in range (len(number)):
    data2=data[((data.xlong-centerx[i])**2+(data.yla-centery[i])**2)<=Td]
    pr.append(data2.price.mean()*number[i])
print('group_all_price=',pr)

prnew=[]
numbernew=[]
peoplenew=[]
positionofselect=[]
for i in range(len(pr)):
    if pr[i] >0:
        positionofselect.append(i)
for j in positionofselect:
    prnew.append(pr[j])
    numbernew.append(number[j])
    peoplenew.append(people[j])

from mpl_toolkits.mplot3d import Axes3D
fig=plt.figure()
ax=Axes3D(fig)
X=np.array(numbernew)
Y=np.array(peoplenew)
Z=np.array(prnew)
X, Y = np.meshgrid(X, Y)
ax.plot(X,Y,Z,'.',c='r')
ax.set_xlim=(0,40)
ax.set_title('The Relationship Among Tasknumber,VIPnumber and Price')
ax.set_xlabel=('Tasknumber')
ax.set_ylabel=('VIPnumber')
ax.set_zlabel=('Price')
plt.show()
print("Tasknumber=",numbernew)
print("VIPnumber=",peoplenew)
print("Price=",prnew)
```

## 附录 7 问题四部分绘图代码

```
###支撑材料文件名：第四问部分绘图代码.py
########第四问绘图代码########
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
newdata=pd.read_excel('attach3.xls')
iddata=pd.read_excel('attach2.xlsx')
plt.scatter(iddata.loc[:, 'id_x'], iddata.loc[:, 'id_y'],s=10,
  edgecolor='none', alpha=1,c='r',label='VIP Location')

plt.scatter(newdata.loc[:, 'xnlong'], newdata.loc[:, 'ynla'],s=10,
  edgecolor='none', alpha=1,c='b',label='New Task Location')

plt.xlabel('x')
plt.ylabel('y')
plt.title('The Location')
plt.xlim((112.75,114.5))
plt.ylim((22.4,23.5))
plt.legend(loc='best')
plt.show()
```