“拍照赚钱”的任务定价

摘要

随着科技的发展，拍照赚钱成为了一种新兴的自助式服务模式。这种拍照模式既可以为企业提供图像识别、商业检查与信息搜索等服务，还可以节省调查成本、缩短调查周期并且很大程度地保证数据的真实性。而此软件最核心的因素是任务定价，从而给出合理的定价以及提高任务的完成比例是当前的主要任务。本文将根据题意制定出合理的定价模型。

针对问题一，为了解在广州、深圳、东莞和佛山四个城市的任务定价规律，本文研究并分析了任务完成情况及其影响因素。通过散点图、饼状图和地图工具进行多方面分析。其中，深圳的未完成任务明显较多，而东莞完成率最高。本文发现，广州和深圳的消费水平明显较高，而平均定价却相对较低，未能激起人们完成任务的意愿。通过指数拟合分析，发现任务点周围会员人数与定价呈负相关。深圳的会员人数与任务完成情况正相关，而广州和佛山则呈负相关。综上所述，本文认为定价偏低削弱了人们完成任务的积极性，并且在经济水平高的城市更为明显。

针对问题二，本文建立了一个针对任务定价的新模型。该模型综合考虑了地域、距离、会员人数以及会员的信誉分值等因素，以此来优化定价并提高任务完成率。这是一个多因素定价模型，可以反映出基础价格、会员因素及竞争情况的综合作用。并通过探讨新方案与原方案在定价和任务完成率方面的变化及其影响，得出结论：提高定价能够更好地激发人们的完成意愿，任务的完成率由原先的62.5%提高至69.1%，涨幅为6.6%，提升效果明显。

针对问题三，本文建立了基于DBSCAN算法的任务聚类模型，利用该算法对任务进行打包聚类，并将得到的聚类结果可视化。然后通过将新的定价信息替换掉旧的定价数据，结合历史数据，重新计算出每个聚类的任务完成率，再将预测完成率添加到数据表中如表2所示。最终新定价方式所对应的任务完成率预测值为72.1%，相比第二问涨幅为3.1%，可见聚类显著提高了任务的吸引力。

针对问题四，本文建立了一个新的定价模型，利用聚类结果来优化价格策略。采用第二问得出的新定价作为基准，结合聚类规模和任务点与聚类中心的距离，通过公式对每个任务的价格进行调整，显著降低了总成本与平均定价。在与先前模型的比较中，新方案展现出更高的经济效益，总成本由62619.1元降至60101.4元，任务点的平均定价也相应降低，由74.99元降为71.98元。

关键词：定价问题；DBSCAN聚类；随机森林；任务完成率；优化

一、 问题重述

在移动互联网时代，随着图片交易需求的不断递增，“拍照赚钱”成为了移动互联网下的一种新型服务模式用户通过下载APP，注册新用户，选择拍照任务（如去超市检查某种商品上架情况），赚取与任务对应的酬金。这种基于移动互联网的自助式劳务众包平台，能够快速为企业提供货架图像识别、商业检查与信息搜索。相比于传统市场的调查方式，这种人工智能技术可以节省调查成本、保证数据的真实性以及缩短了调查的周期。而决定这种新兴的人工智能技术，是否能够受到市场欢迎的核心因素则是APP 的任务定价。如果定价不合理，拍照任务无人问津，导致商品检查失败。

根据题目所提供的数据，结合附件一（已经结束的任务数据，包含了任务地点的经纬度和完成情况）、附件二（会员信息数据，包含了会员的位置、信誉值、开始预定时间以及预定限额，而原则上信誉值越高，预定时间越早，预定限额越高）、附件三（新的检查项目任务数据，只含有任务的位置信息），完成下面的问题解答：

问题一:研究附件一中项目的任务定价规律，分析任务未完成的原因;

问题二: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;

问题三:在实际情况下，由于多个任务可能因为位置比较集中，导致用户会争相选择，所以考虑到其中的一种情况是将这些任务联合在一起打包发布。而在这种考虑下，将如何修改前面的定价模型，对最终的任务完成情况又有什么影响;

问题四:对附件三中的新项目给出自己的任务定价方案，并评价该方案的实施效果。

二、 问题分析

2.1 问题一的分析

问题一的分析：问题一要求我们研究附件一中项目的任务定价规律。因此，本文根据附件一中的数据，绘制出任务定价与任务位置（经纬度）的散点图。通过对任务点的经纬度进行散点图和饼状图分析，初步了解到任务点的分布和完成情况。随后，使用地图工具将数据标注在地图上，发现任务点主要集中在上述四个城市。进一步的城市任务分布和完成度分析揭示，广州和深圳的未完成任务较多，而东莞的完成率最高。本文还通过研究各城市任务定价与完成率之间的关系，发现深圳的定价最低，完成率也最低，而东莞的平均定价最高，任务完成率达到100%，广州和佛山的定价和任务完成率都介于深圳和东莞之间。通过对四个城市的GDP数据研究，本文得出项目中的任务定价规律：任务的定价设置与任务点周围的会员人数因素有关，采用指数拟合得到

较好的效果，发现负相关效果明显，即任务点周围的会员人数越少，任务定价较高，这样设置定价才能使得在人员较稀疏的地区中，其任务点还有可能被会员执行。任务未完成的原因主要有以下几点：任务定价过低，无法激发人们的完成意愿；经济水平较高的城市反而定价更低，人们的完成意愿进一步降低；位置较偏远，周围会员人数较少。

2.2 问题二的分析

问题二的分析：问题二要求我们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。因此，在第一问的基础上，本文结合附件二的数据，建立了一个新的定价模型，根据任务点周围的会员人数，信誉分值，距离等因素来调整定价。对比 [6]的动态定价策略，该定价模型的目标是通过合理的定价策略提高任务的完成率。首先，基于不同城市的任务完成率数据，做出定价的基准决策，而后根据区域内的会员数据和任务点之间的具体位置分布来动态调整定价。

2.3 问题三的分析

问题三的分析：用问题三要求我们结合题中考虑，修改前面的定价模型，并指出修改后的模型对最终的任务完成情况又有什么影响。本文通过DBSCAN聚类算法对任务点进行打包，聚类目的为打包任务以减少单个会员需要完成的任务点数量，降低总距离，提高任务完成率，并通过历史完成率来重新计算聚类后任务的完成率，这样既考虑了任务的分布密度，又避免了任务过度集中带来的完成率下降问题。首先，通过设置聚类参数，保证任务点不会过于集中，影响用户执行意愿。然后进行任务完成率的计算，完成率大于等于50%的任务被视为可完成，从而有助于评估整体任务执行的可行性。最后，将该完成率与原先的任务完成率进行对比，发现，新的打包方案导致了任务完成率从69.1%提高至72.2%，增加了3.1%。这一提高证明了：通过聚类的方式，将任务点进行打包，使得任务点的定价更加合理，提高了任务的完成率。

2.4 问题四的分析

问题四的分析：问题四要求我们对附件三中的新项目给出自己的任务定价方案，并评价该方案的实施效果。因此，本文建立一个新的定价模型，利用聚类结果来优化价格策略。首先采用第二问中求解出的新定价作为基准值 b_0，再将根据聚类内任务数量和任务点到聚类中心的距离进行调整任务的价格。在价格调整后，根据定价公式计算优化后的新定价，生成新的价格表，展示了优化前后的定价变化。最后结合表3中的数据，对比第二问的优化方案，其总成本为71811.6元，平均任务点定价为86.00元。新

方案的总成本降低至60101.4元，平均任务点定价为71.98元。因此，本文得出结论：通过聚类处理，细分了任务定价策略，避免了极端定价因素，使得平均定价更加合理。

三、 模型假设

考虑到生活中的实际情况以及不确定因素，以及模型构建的合理性，本文做出以下假设：

1. 由于题目未给出各任务点以及会员位置之间的距离，本文用欧式距离来代替实际距离，如下式1和2所示：

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \times 111 \tag{1}$$

$$d_{ik} = \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2} \times 111 \tag{2}$$

由于附件中所给出的坐标都是经纬度数据，因此根据1度约等于111公里的公式，上式1和2都需要乘以111，将经纬度转化为实际距离。

2. 本文假设模型不受其他外在因素的影响，如天气因素，道路施工等。

3. 本文假设附件数据均真实可靠，无异常值的出现。

四、 符号说明

符号	表示含义
d_{ij}	任务点i和会员位置j之间的距离（km）
d_{ik}	任务点i和任务点k之间的距离（km）
x_i	任务点i的 x 坐标（经度）
y_i	任务点i的 y 坐标（纬度）
x_k	任务点k的 x 坐标（经度）
y_k	任务点k的 y 坐标（纬度）
m	任务点数量
n	会员数量
n_{i1}	任务点i周围1km的会员数量
b_0	基准值
$price_i$	任务点i的定价

$score_j$	会员位置j的信誉分数
cpt_{ik}	任务点k对任务点i的竞争因素影响值
cpt_i	周围任务点对任务点i的竞争因素总影响值
k	会员因素的权重系数
w	竞争因素的权重系数
o	聚类规模大小
d_{ci}	任务点i距离聚类中心点的距离（km）

五、 模型的建立与求解

5.1 问题一

5.1.1 模型的建立

首先对附件1中的所有任务点进行位置分析，画出如下散点图 1：

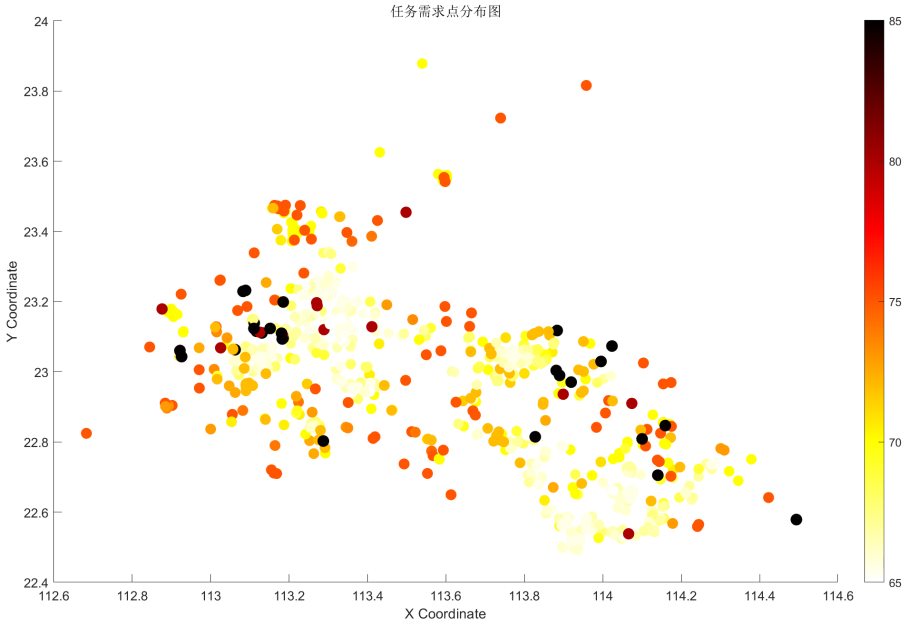

图 1: 任务点需求分布热力图

其中，任务点颜色的深浅代表着任务点的定价，颜色越深，定价越高。所有任务的定价差距较小，所有任务定价都在65-85元的区间内。

其次对任务的完成情况进行分析，统计后得到如下饼状图 2：

根据2可知：任务完成率为62.5%，有37.5%的任务未完成，可见完成率还有较大的

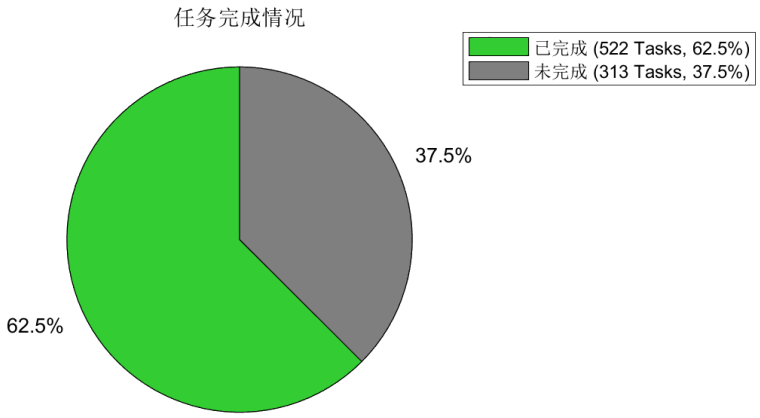

图 2: 任务完成情况

提升空间。

由于题目中提供了每个任务点真实的经纬度信息，因此我们利用地图慧工具，批量将数据点导入到地图中进行标注，得到的结果如下图 3 所示：

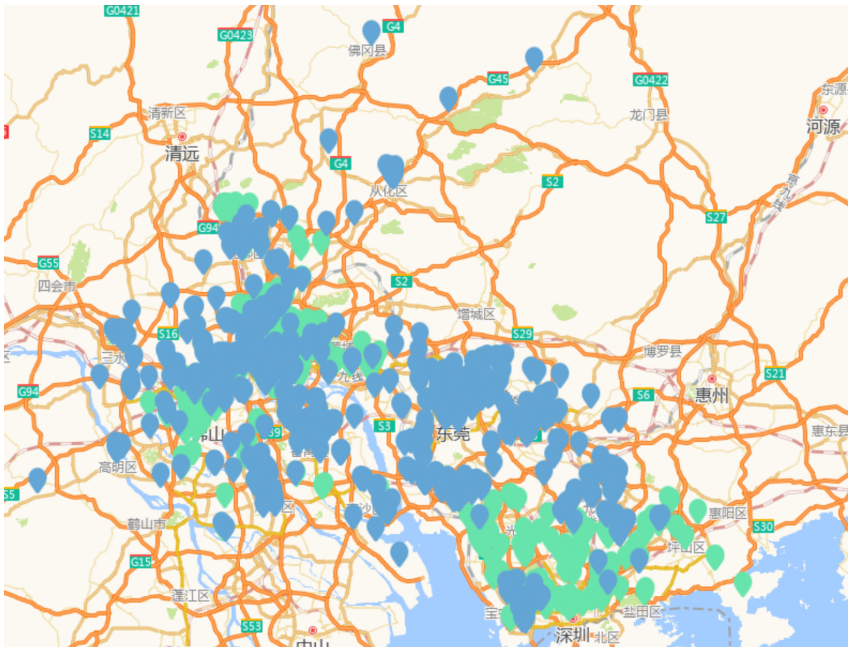

图 3: 四座城市任务完成情况

由上图3可见，这些任务点主要都集中在广州，深圳，东莞和佛山这四座城市，其中蓝色标记点代表任务已完成，绿色代表未完成。由于标记点过多，有较多重叠部分，可视化效果较为一般，得到的信息较少，因此分别对这四座城市的任务分布以及完成情况进行进一步的分析，探究不同地域对任务完成情况的影响大小。

先对任务分布进行分析，得到其城市分布情况，如下图 4 所示：

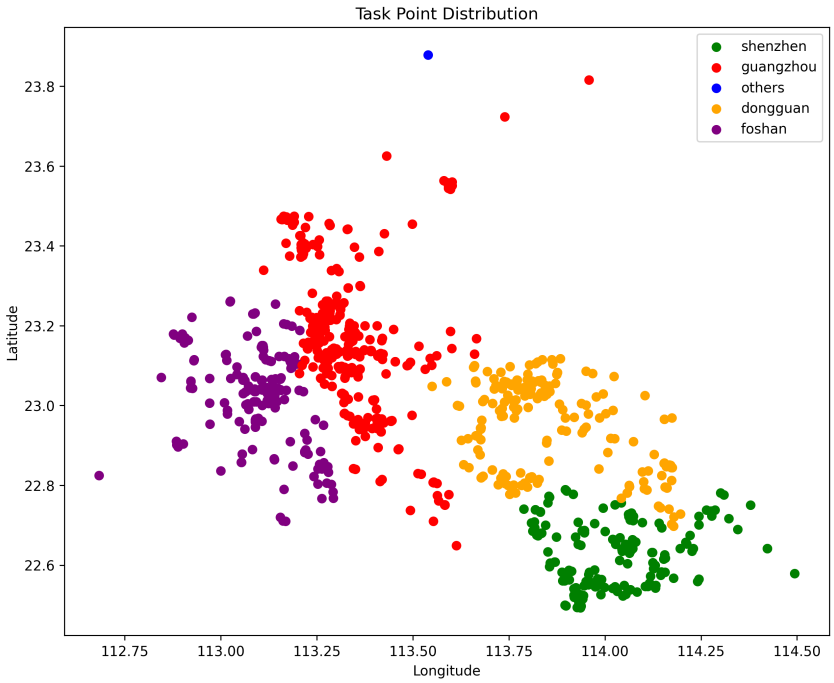

图 4: 任务点的城市分布情况

其中，红色点表示位于广州，绿色点表示位于深圳，紫色点表示位于东莞，黄色点表示位于佛山。

再对完成度进行分析，如下图 5 所示：

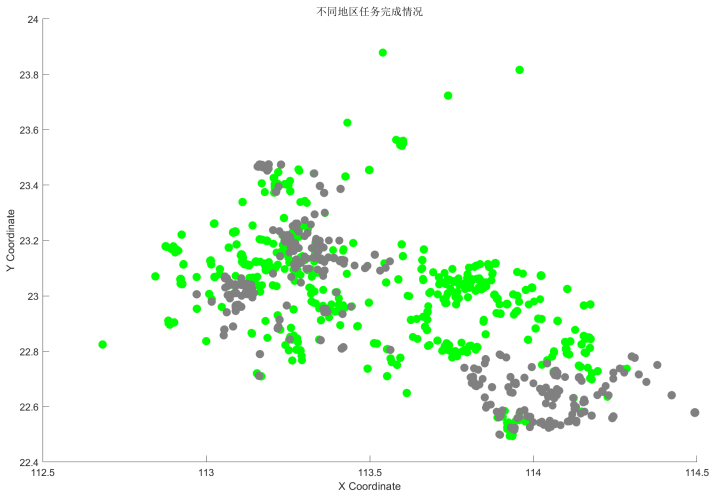

图 5: 四座城市任务完成情况

由上图4和5可见：未完成的任务点大部分集中在广州和深圳，还有部分集中在佛山，而东莞的任务点完成度较高。

为了进行具体的分析，本文将任务点按照城市进行分类，分别统计其任务定价以

及任务的完成率，得到如下图 6 所示：

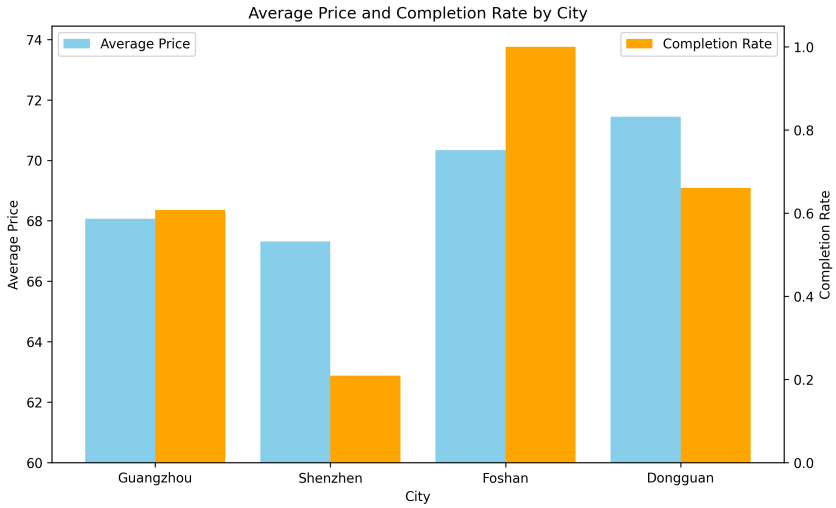

图 6: 四座城市任务情况对比

由上述对比图可知：广州和深圳的任务定价偏高，但完成率较低，而东莞和佛山的任务定价偏低，但完成率较高。从这一点可以看出：定价偏低的任务点，人们的接受意愿较低，因此完成率较低；而定价偏高的任务点，则能够吸引更多的人去完成，因此完成率更高。

此外，考虑到人们在不同地区的生活水平，消费理念等方面的不同，本文将还对不同城市近年来的GDP做了分析，得到4座城市的总GDP和人均GDP如下图 7 所示：

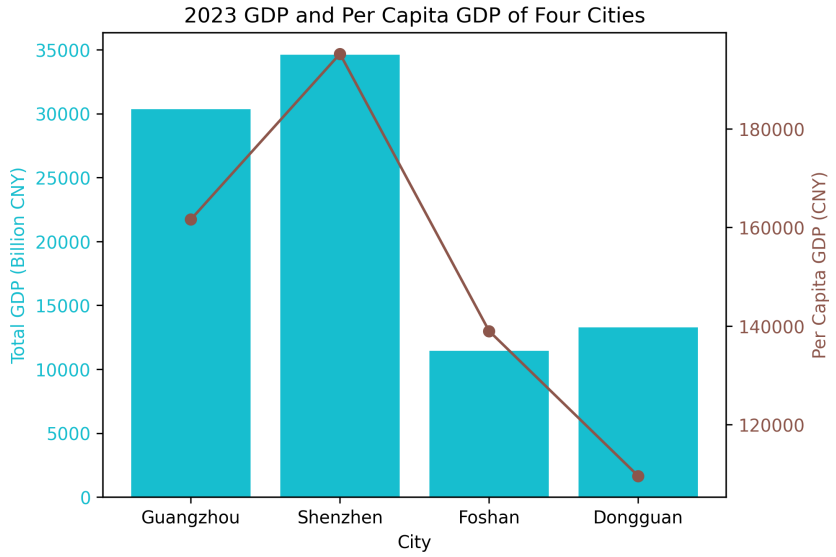

图 7: 四座城市近年来的GDP情况

由上图7可见：广州和深圳的GDP总量和人均GDP均较高，东莞和佛山的GDP总量和人均GDP较低。然而根据图6任务完成情况的对比，广州和深圳的任务定价却反而较低，在生活消费水平都较高的城市中，较低的定价显然无法有效激发人们去完成任务

的意愿，因此完成率较低。而在经济水平较低的东莞和佛山中，较低的任务定价能够吸引更多人去执行任务，因此完成率较高。

5.1.2 模型的求解

考虑到任务的定价设置与任务点周围的会员人数因素有关，本文统计了各个城市所有任务点方圆一公里的会员人数，并分析其与任务定价的关系，得到如下图 8 所示：

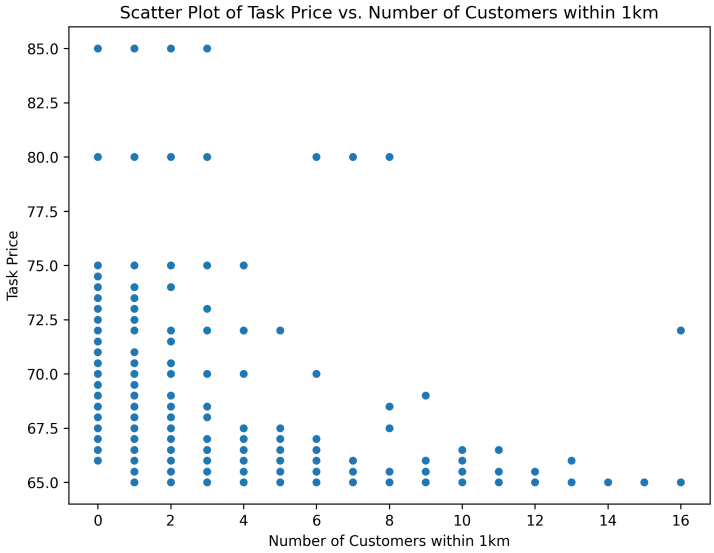

图 8: 会员人数与任务定价关系

由图8可知。会员人数和任务定价之间存在一定的负相关性，因此本文尝试用多项式，指数函数等方式来进行曲线拟合。根据 [9]中的拟合方法，最终采用指数拟合的方式，如下图 9 所示：

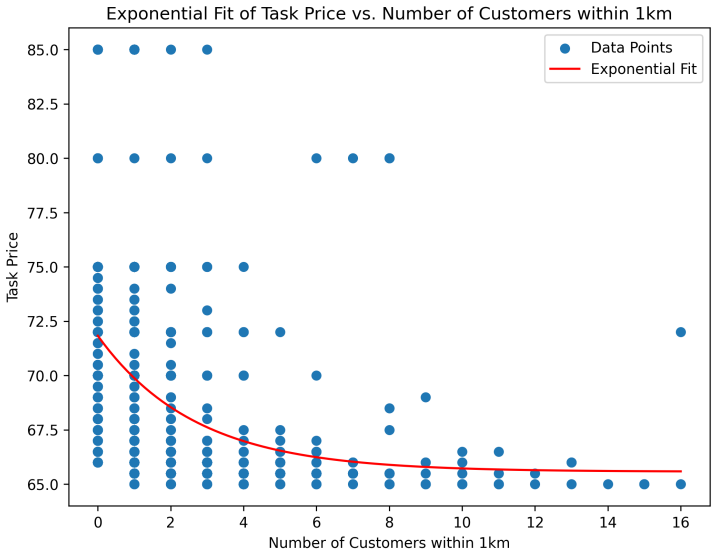

图 9: 指数拟合情况图

由图9可知，拟合效果较好，当周围会员数量较少，定价相对较高，这样才能吸引人们到更远处的任务点完成任务，符合定价规律。为了具体探究人们出行到任务点的距离远近方便程度与定价的关系，因此本文根据式1和3，统计了各个城市所有任务点方圆一公里的会员人数，并分析任务是否完成与会员人数的关系，得到如下图 10 所示：

$$n_{i1} = \sum_{j=1}^{n} \begin{cases} 1, & \text{if } \mathrm{dis}_{ij} \leq 1 \\ 0, & \text{otherwise} \end{cases}, \quad i = 1, \ldots, m \tag{3}$$

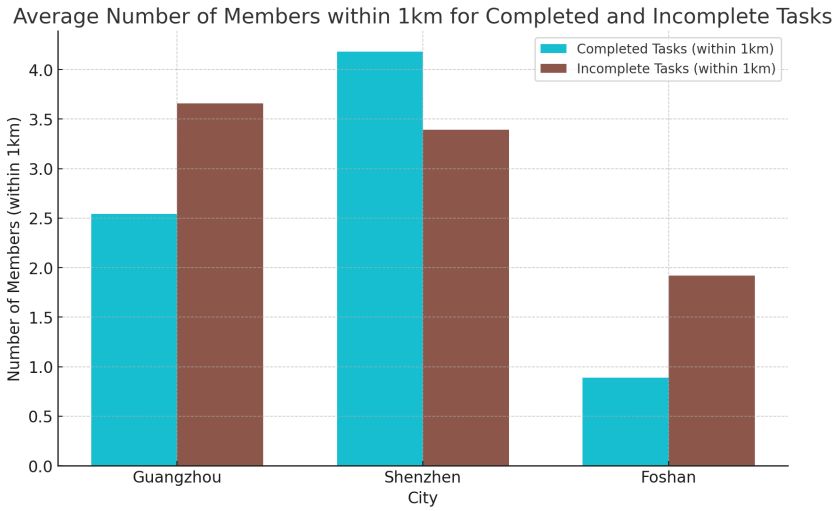

图 10: 会员人数与任务完成情况

由图10可见：深圳市的会员人数与任务完成情况呈正相关关系，即任务点周围的会员人数越多，任务完成情况越好。然而，广州市和佛山市的会员人数与任务完成情况呈负相关关系，即任务点周围的会员人数越多，任务完成情况越差。而东莞市所有的任务点均全部完成，因此不在讨论范围之内。综上所述，距离远近对任务的完成率有一些不确定性的影响，但并不是决定性因素。

综上所述，本文认为：

任务的定价设置与任务点周围的会员人数因素有关，采用指数拟合得到较好的效果，发现负相关效果明显，即任务点周围的会员人数越少，任务定价较高，这样才能吸引更多人去执行任务。

任务未完成的原因主要有以下几点：

1. 任务定价过低，无法激发人们的完成意愿；

2. 经济水平较高的城市反而定价更低，无法激发人们的完成意愿；

3. 与距离远近有关，周围会员人数较少。

5.2 问题二

5.2.1 模型的建立

首先，由第一问已知：定价的影响因素包括地域，距离，会员人数等。以地域角度作为出发点，根据不同城市的原方案任务完成率来看，由于东莞市完成率100%，因此本文不对其定价进行调整。其数据也不在本题讨论范围之内。此外，根据附件二的会员数据以及 [12]中的定价影响因素，可知信誉分值也是影响人们进行任务选择，从而决定任务完成率的重要因素。谢 [13]等人对任务定价方法进行分析，并考虑多种因素来进行曲线拟合。本文根据上述这些因素，建立了一个定价模型，根据任务点周围的会员人数，信誉分值，距离等因素来调整定价。具体定价模型的目标函数设定如下：

$$\text{定价} = \text{基准值} + k \cdot \sum(\text{信誉分值} \times \text{会员距离远近}) + \text{惩罚项} \tag{4}$$

其中各部分影响因素的详细分析如下：

1. 基准值

对附件一的数据进行分析可知：定价的最小值为65元，因此我们将65元作为基准值，即定价的最低基准为65元，如下式子5所示，否则价格过低可能导致用户流失。

$$b_0 = \text{基准值} = 65 \tag{5}$$

2. 会员因素

本文将信誉分值和会员到任务点的距离作为一个整体来考虑，信誉分值越高，会员距离任务点越近，定价越高。

$$k \cdot \sum_{j=1}^{n} \left(limit_j + \sqrt{score_j} \times (1 - dis_{ij}) \times 0.01 \right), \quad j = 1, \ldots, n \tag{6}$$

其中，k 为权重系数，用于调节信誉和距离对定价的影响。n 为任务点数量，$limit_j$ 为第 j 个会员的预定任务限额，$score_i$ 为第 i 个任务点的信誉分值，由于不同会员之间的信誉分值差距量级过大，因此进行开方操作弱化处理，减小影响程度。dis_{ij} 为第 j 个会员到任务点 i 的距离，其与定价的关系为负相关，即距离越远，定价偏向越低的趋势。由于计算出的值过大，因此需要×0.01进行范围缩小。

3. 竞争项

由于某些任务点周围的任务点较多，因此需要考虑竞争环境对定价的影响因素：任务点附近其他任务点数量及距离。当任务点越多，用户选择更多，相应地本任务点的竞争力会减弱，因此需要提高定价。此部分中，单个周围的任务点 k 影响本任务点 i

定价的公式如下7所示：

$$cpt_{ik} = \max\left(\sqrt{\frac{1}{d_{ik}}} \times 0.1, 5\right) \tag{7}$$

d_{ik} 表示到其他任务点的距离，距离为欧式距离，为防止正无穷，需要添加最大约束。由于计算出的值过大，因此需要×0.1进行范围缩小。

由于该任务点周围有多个任务点，其影响为叠加状态，因此该任务点总的竞争项为：

$$cpt_i = w \cdot \sum_{k=1}^{n} cpt_{ik}, \quad k = 1, \ldots, n \tag{8}$$

其中，w 为竞争系数，用于调节竞争环境对定价的影响。n 为周围1km范围内任务点数量。

4. 最终定价确定

综合上述式子，整合得到最终的定价模型，如下式9所示：

$$price_i = b_0 + k \cdot \sum_{j=1}^{n} \left(score_i \times (1 - dis_{ij})\right) + w \cdot \sum_{k=1}^{n} \min\left(\sqrt{\frac{1}{d_{ik}}}, 5\right), \quad i = 1, \ldots, n \tag{9}$$

5. 参数调整

在最终定价函数公式 9 中，k，w 的值是未知数，本文根据附件一得出的结果数据，对其采用随机森林进行拟合。

随机森林是一种集成学习方法，它通过构建多个决策树来进行分类或回归。Biau [1]等人对随机森林的发展历程以及该算法背后的数学原理进行了充分的论述。每棵树在训练时都从数据集中随机选取一部分样本和特征，这种随机性使得随机森林在处理高维数据和防止过拟合方面表现优异。最终，随机森林通过多数投票（分类）或平均值（回归）来生成最终预测。从 [10]中了解到随机森林方法的性能，它因其高准确性和稳健性，常用于各种机器学习任务。

本文使用随机森林方法并多次迭代调参，从而得到最优的k,w的值。最后再根据这些值，利用公式9来计算出最终的定价结果。

5.2.2 模型的求解

根据模型的建立过程，求解步骤如下：

step1: 数据预处理

由于本模型不采用东莞市任务点的数据，因此要根据第一问计算出来的东莞任务点的索引，将相应行数据删除。

step2: 相关因素项计算

根据式5，基准值为65元。

根据式6，考虑信誉分、会员距离远近这两个因素来计算会员因素，最终将其值作为新的一列”会员因子分数”，代入附件一表格中。

根据式7和8，计算竞争项的值，最终将其值作为新的一列”竞争因子分数”，代入附件一表格中。

step3: 最终定价确定

根据式9，计算最终的定价。最终将其值作为新的一列”预测合理定价”，代入附件一表格中。

step4: 参数调整

采用随机森林方法，先对任务是否完成进行预测，本质上是一个二分类问题。通过随机森林进行参数调整，以10为步长，从[10,200]范围内进行森林树木数量的选择。Oshiro [4]等人对随机森林中树的数量选择进行了充分的分析，其认为随机森林树的数量存在一个最优的临界点，超过该点后增加树的数量意义不大。Paul [5]等人也在树的数量最小化上进行了细致的研究，说明不能无谓增加树的数量，而需要根据实际情况调整选择。最终求得的图如下11所示：

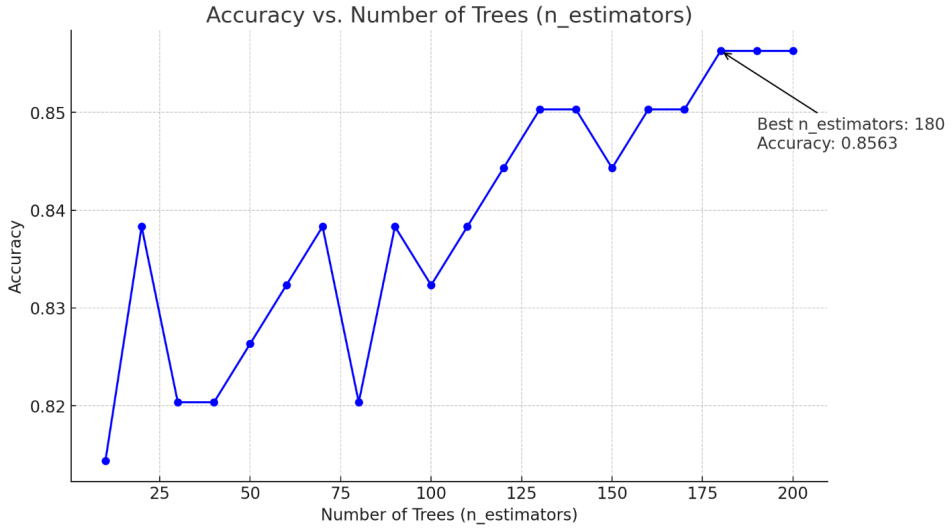

图 11: 随机森林参数调整

由上图可知：当树木数量达到180时，精确度达到最高，模型效果最好，因此选择树木数量为180的模型，对k，w的值进行拟合。首先根据选择的特征和标签，将数据集分为训练集和测试集，鉴于数据量较多，为了提高模型的效果，给训练集更大的样

本，因此训练集和测试集的划分比例确定为8:2，然后将k，w分别取[0,20]的范围，步长为0.2，进行总共约一万次的网格搜索，不断训练模型并迭代更新，最终得到最优的k，w的值。其中k为29.2，w为26.8。

step5: 任务完成率预测

根据算出来的k，w的值，代入公式9，计算出最终的定价。再将定价替换掉原先的定价数据，利用训练好的随机森林模型，对任务完成率进行预测。最后计算出的任务完成数量，加上东莞所有完成的任务数量，除以总任务数量，得到最终的任务完成率。将该完成率与原先的任务完成率进行对比，得到如下图 12 所示：

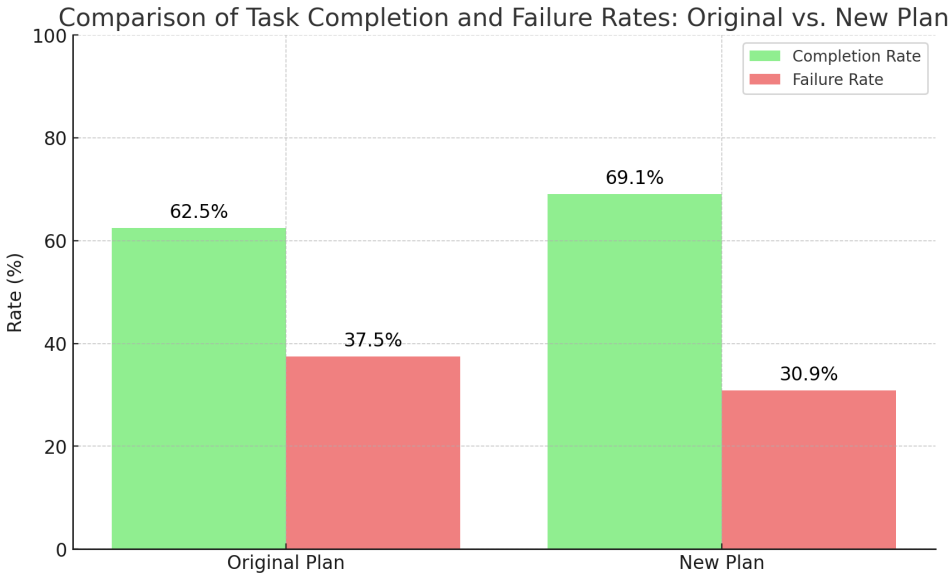

图 12: 新方案定价与原方案对比

由上图12可见：新方案的定价与原方案相比，有较明显的提升，任务完成率由原先的62.5%上升到69.1%，涨幅为6.6%，原方案的总成本为57707.5元，平均任务点定价为69.11元，新方案的总成本为71811.6元，平均任务点定价为86.00元，相比原来，定价也有较大的上升。这也进一步证明了：提高定价可以更好地激发人们的完成意愿，提高任务的完成率。

5.3 问题三

5.3.1 模型的建立

根据位置和任务的集中程度，将任务进行打包，可视作聚类问题，常见的聚类算法有K-means，DBSCAN，层次聚类等。本文采用DBSCAN聚类算法进行聚类处理。该算法是一种基于密度的聚类算法，能够发现任意形状的聚类，且能够处理噪声数据。其基本思想是：对于给定的数据集，通过计算每个点的密度，从而找出密度大于某一阈值

的点，然后以这些点为中心，找出其密度可达的点，从而形成一个聚类。Dehuri [2]等人对DBSCAN密度聚类进行了详细介绍，并与其他聚类算法(如K-means聚类)等进行了比较分析。Kim [3]等人在DBSCAN聚类过程中考虑了空间对象的影响。本文所考虑的聚类范围半径为1km，密度阈值为4，即最多有4个任务点打包在一起，防止任务点过于集中的情况下，一个会员所需完成的任务点数量过多，导致任务点总距离过大，影响人们的执行意愿，导致降低任务完成率。

此外，还需要预测聚类后的任务完成率，根据历史完成率，聚类后重新计算任务完成率，得到每次簇新的任务完成率。若完成率大于等于50%，则任务可以完成，否则无法完成。将所有可完成的任务点进行累加统计，除以总聚类数量，即可得到任务完成率，如下式10所示：

$$\text{任务完成率} = \frac{\text{可完成任务点数量}}{\text{总聚类数量}} \tag{10}$$

5.3.2 模型的求解

根据模型的建立过程，求解步骤如下：

step1: DBSCAN聚类

首先将任务进行打包聚类，根据DBSCAN算法以及 [7]中的聚类分析思想，设定半径为1km，密度阈值为4，进行聚类处理。最终得到的聚类结果如下图 13 所示：

其中，红色点代表着聚类的任务点，蓝色点由于密度不够，未被聚类，仍然单独作为一个任务点。

step2: 任务完成率计算

将新的定价替换掉原先的定价数据，再根据历史完成率，重新计算任务完成率。然后将预测完成率添加为表格中新的一列，如下表2所示：

表 2: 任务数据表

任务号码	任务gps纬度	任务gps经度	任务标价	任务执行情况	聚类情况	预测完成率
A0001	22.566142	113.980837	66.605203	1	0	0.25
A0002	22.686205	113.940525	67.486857	0	1	0.00
A0003	22.576512	113.957198	66.081525	0	2	0.00
A0004	22.564841	114.244571	65.580365	0	3	0.00
A0005	22.558888	113.950723	65.889314	0	4	0.00
...	...	...	...	...	...	...
A0807	23.039098	113.773178	65.500000	1	491	1.00
A0808	22.846704	114.159286	85.000000	1	385	1.00
A0828	23.012808	113.760312	66.000000	1	408	1.00
A0833	22.814676	113.827731	85.000000	1	492	1.00
A0834	23.063674	113.771188	65.500000	1	400	1.00

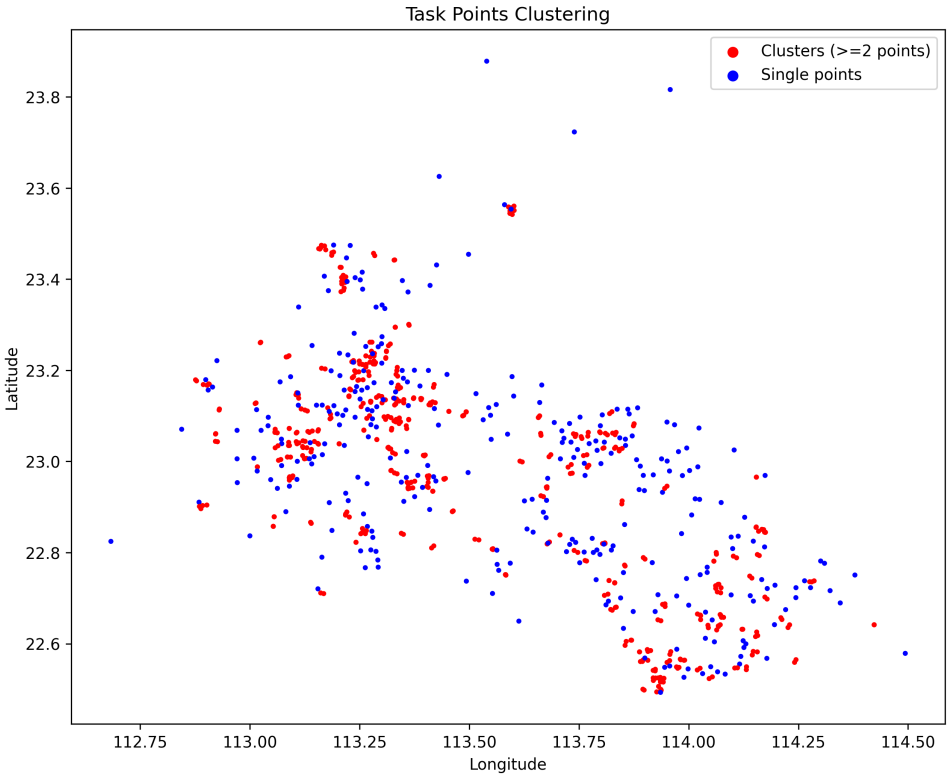

图 13: 任务点聚类结果

最后利用式10，用计算出的任务完成数量，除以总聚类数量，得到最终的任务完成率。将该完成率与原先的任务完成率以及优化后但为聚类的任务完成率进行对比，得到如下图 14 所示：

由上图14可见：新的打包方案的定价与第二问的优化方案相比，也有较明显的提升，任务完成率由原先的69.1%上升到72.2%，涨幅为3.1%。这也进一步证明了：通过DBSCAN聚类的方式，将任务点进行打包，使得任务点的定价更加合理，提高了任务的完成率。

5.4 问题四

5.4.1 模型的建立

聚类完成后需要对价格进行调整，价格基准值为第二问的新定价，此外，新定价还与聚类规模，同一个类中的任务间距离有关，结合多种情况构建出的新定价公式如下：

$$price_i = b_0 \times (1 + 0.05 \times (o - 1)) \times (1 - 0.05 \times d_{ci}), \quad i = 1, \ldots, m \tag{11}$$

其中 b_0 是以第二问求解出来的新定价作为的基准值，o 是聚类规模大小，d_{ci} 是任

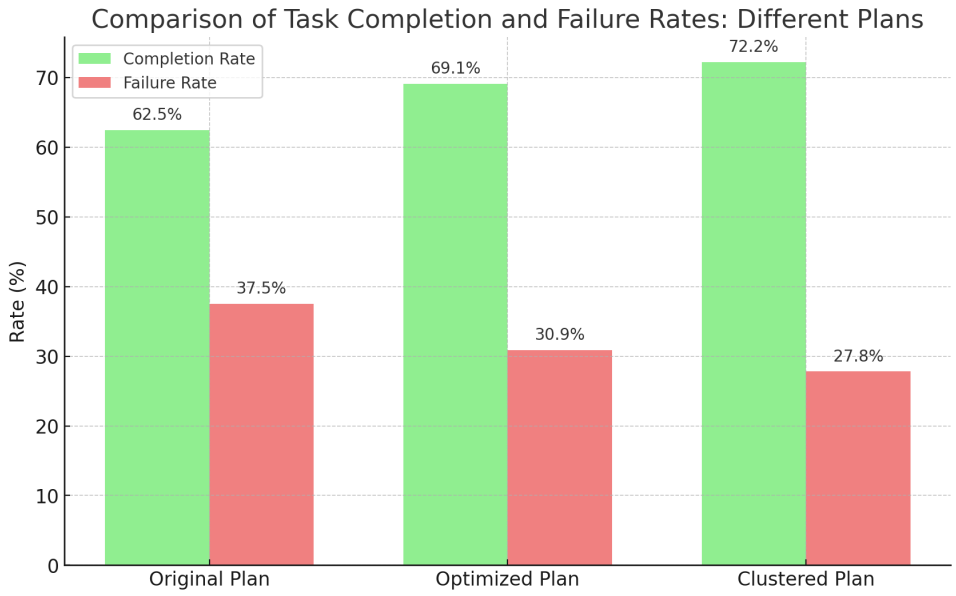

图 14: 新方案定价与原方案对比

务点 i 距离聚类中心点的距离。

5.4.2 模型的求解

step1: 价格调整

根据公式11，计算出新的定价。如下表3所示：

表 3: 新预测价格表

任务号码	任务gps纬度	任务gps经度	任务标价	优化后的任务标价	final_cluster
A0001	22.566142	113.980837	66.605203	76.576235	0
A0002	22.686205	113.940525	67.486857	74.221110	1
A0003	22.576512	113.957198	66.081525	69.374788	2
A0004	22.564841	114.244571	65.580365	68.847872	3
A0005	22.558888	113.950723	65.889314	69.171632	4
...	...	...	...	...	...
A0807	23.039098	113.773178	65.500000	65.500000	491
A0808	22.846704	114.159286	85.000000	89.214610	385
A0828	23.012808	113.760312	66.000000	69.286189	408
A0833	22.814676	113.827731	85.000000	85.000000	492
A0834	23.063674	113.771188	65.500000	72.039786	400

第二问优化方案的总成本为71811.6元，平均任务点定价为86.00元，新方案的总成本为60101.4元，平均任务点定价为71.98元，相比第二问，定价较低。这是因为第二问在单独处理定价的过程中，有些提高定价的任务点出现极端情况导致定价虚高，影响到的总体平均任务点定价。而在第三问中，通过聚类的方式，将任务点进行打包，使得任务点的定价更加合理，因此平均任务点定价较低，不过相比于原方案的平均定

价57.71元，还是有所提升。

六、 模型的评价与推广

6.1 模型的优点

1. 根据不同城市的经济水平等方面，对任务点的定价进行相应的划分，分情况讨论，在第二问的优化定价中，有较好的效果。

2. 在第三问中，通过聚类的方式，将任务点进行打包，使得任务点的定价更加合理，提高了任务的完成率。

3. 在第四问中，根据聚类规模大小，任务点距离聚类中心点的距离等因素，对价格进行调整，使得定价更加合理。

4. 在参数确定方面上，通过网格搜索的方式，不断训练模型并迭代更新，最终得到最优的k，w的值，使得参数选择更加合理，在模型三中结合了第二问的情况后仍有较好的效果，提高了模型的鲁棒性。

6.2 模型的缺点

1. 在第二问中，由于采用随机森林的方法来拟合k，w的值，需要大量的数据集，且需要多次迭代调参，耗时较长。此外，最终调参完成的任务是否完成的预测结果准确率为86.5%，不算特别高，会对后续价格调整后的任务完成预测产生一定的影响。

2. 会员的信誉分数以及预计任务限额与题目的任务定价均有一定的相关性，但是不同会员的信誉分数以及预计任务限额差距较大，尽管在计算过程中进行了开方弱化处理，但是仍会出现某些极端情况造成定价过高，尤其是第二问较为明显，为了提升任务完成率，所消耗的定价成本较大。

3. 在第三问中，模型未考虑可能影响任务完成意愿的外部社会经济因素以及环境变化，例如居民的参与意愿或任务执行的具体情况，这可能导致模型预测与实际情况存在偏差，且未对不同城市或任务类型进行针对性调优，可能导致统一的聚类标准无法适应不同特性的数据，影响聚类的有效性和任务的完成率。

4. 在第四问中，模型假设了价格调整仅与聚类规模和任务与聚类中心的距离有关，忽略了其他可能影响定价的重要因素，如市场需求、竞争对手的定价策略和用户的支付意愿等，使得定价公式过于单一化。

6.3 模型的推广

1. 对于问题二的定价模型，可以推广到其他类似的问题上，如城市交通与配送、公共服务等优化资源配置问题，在城运物流和外卖配送领域，可以利用该模型的定价机制优化配送路线，合理定价以提高驾驶员的积极性，降低配送耗时，提高用户满意度，在公共服务领域，可以通过该模型的定价机制，优化政府或公益机构在社区服务中的资源投入和服务定价，提高居民的参与度与服务质量。

2. 问题三中的模型通过采用DBSCAN聚类算法对任务进行打包，优化了任务点的定价，进而提高了任务的完成率。根据 [8]和 [11]的分析可知，该定价方法以及适用范围可推广到其他领域，如生态产品的定价等。在在线教育平台中，可以将任务点（如课程学习、测验）进行合理打包，优化学习路径，提高学生的学习完成率。

3. 本文中的基于数据驱动的定价模型，通过聚类分析，能够将相似特征的任务聚合在一起，根据任务特征和距离动态调整价格。这种数据驱动的定价方法使得价格更加透明和合理，避免了单纯定价带来的不公和误差。该模型可应用于于共享出行、共享单车等共享经济平台中，依据用户需求和区域特征优化任务定价，提高资源的利用率，同时吸引更多用户参与。

参考文献

[1] Gérard Biau and Erwan Scornet. A random forest guided tour. *Test*, 25:197–227, 2016.

[2] Satchidanandan Dehuri, Chinmay Mohapatra, Ashish Ghosh, and Rajib Mall. Comparative study of clustering algorithms. 2006.

[3] Byung-Cheol Kim. Design and development of clustering algorithm considering influences of spatial objects. *The Journal of the Korea Contents Association*, 6(12):113–120, 2006.

[4] Thais Mayumi Oshiro, Pedro Santoro Perez, and José Augusto Baranauskas. How many trees in a random forest? In *Machine Learning and Data Mining in Pattern Recognition: 8th International Conference, MLDM 2012, Berlin, Germany, July 13-20, 2012. Proceedings 8*, pages 154–168. Springer, 2012.

[5] Angshuman Paul, Dipti Prasad Mukherjee, Prasun Das, Abhinandan Gangopadhyay, Appa Rao Chintha, and Saurabh Kundu. Improved random forest for classification. *IEEE Transactions on Image Processing*, 27(8):4012–4024, 2018.

[6] 丁浩. 基于聚类分析的k航空公司亲子家庭旅客定价策略研究. Master's thesis, 云南大学, 2022.

[7] 冯振华. 基于dbscan聚类算法的研究与应用. Master's thesis, 江南大学, 2017.

[8] 张佳丽. 基于改进dbscan的聚类方法研究. Master's thesis, 吉林大学, 2024.

[9] 张帆. 基于windows-arima-svr混合模型的航运指数拟合与影响因素研究. Master's thesis, 大连海事大学, 2023.

[10] 武炜杰. 随机森林算法的应用与优化方法研究. Master's thesis, 江南大学, 2022.

[11] 汪劲松, 肖燚, and 石薇. 不同目的下生态产品的定价思路、方法选择及应用研究. *生态学报*, (16):1–17, 2024.

[12] 王笑勤. 分享经济平台产品的定价模型研究. Master's thesis, 浙江工商大学, 2022.

[13] 谢丽英, 房丽敏, and 陈强. 基于曲线拟合众包平台的任务发布和定价方法. *广东第二师范学院学报*, 39(05):70–73, 2019.

附录

问题一代码

```
import json
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.path import Path
# 打开并读取JSON文件
with open('../data广东省/.json', 'r', encoding='utf-8') as file:
    data = json.load(file)
    with open('../data广东省/.json', 'r', encoding='utf-8') as file:
    data = json.load(file)

guangzhou = []
shenzhen = []
foshan = []
dongguang = []

for feature in data['features']:
    if feature['properties']['name'] == '广州市':
        print(f"名称: {feature['properties']['name']}")
        guangzhou = feature['geometry']['coordinates'][0][0]
    if feature['properties']['name'] == '深圳市':
        print(f"名称: {feature['properties']['name']}")
        shenzhen = feature['geometry']['coordinates'][0][0]
    if feature['properties']['name'] == '佛山市':
        print(f"名称: {feature['properties']['name']}")
        foshan = feature['geometry']['coordinates'][0][0]
    if feature['properties']['name'] == '东莞市':
        print(f"名称: {feature['properties']['name']}")
        dongguang = feature['geometry']['coordinates'][0][0]

# 读取Excel文件
df = pd.read_excel('../data/Q1.xlsx')
# 从Excel中提取所有点的经纬度数据
lat_points = df['任务纬度gps'].tolist()
lon_points = df['任务经度gps'].tolist()
# 转换为 Path 对象
guangzhou_path = Path(guangzhou)
shenzhen_path = Path(shenzhen)
foshan_path = Path(foshan)
dongguan_path = Path(dongguang)

# 判断哪些点在各城市多边形内
points = list(zip(lon_points, lat_points))
is_inside_gz = guangzhou_path.contains_points(points)
is_inside_sz = shenzhen_path.contains_points(points)
is_inside_fs = foshan_path.contains_points(points)
is_inside_dg = dongguan_path.contains_points(points)

# 初始化标志变量，确保每个标签只出现一次
plotted_labels = {
    "guangzhou": False,
    "shenzhen": False,
    "foshan": False,
```

```
    "dongguan": False,
    "others": False
}

index1 = []
index2 = []
index3 = []
index4 = []


# 绘制所有点
plt.figure(figsize=(10, 8))
for i in range(len(lat_points)):
    if is_inside_gz[i]:
        label = 'guangzhou' if not plotted_labels['guangzhou'] else ""
        plotted_labels['guangzhou'] = True
        plt.scatter(lon_points[i], lat_points[i], color='red', label=
            label)
        index1.append(i)
    elif is_inside_sz[i]:
        label = 'shenzhen' if not plotted_labels['shenzhen'] else ""
        plotted_labels['shenzhen'] = True
        plt.scatter(lon_points[i], lat_points[i], color='green', label=
            label)
        index2.append(i)
    elif is_inside_fs[i]:
        label = 'foshan' if not plotted_labels['foshan'] else ""
        plotted_labels['foshan'] = True
        plt.scatter(lon_points[i], lat_points[i], color='purple', label
            =label)
        index3.append(i)
    elif is_inside_dg[i]:
        label = 'dongguan' if not plotted_labels['dongguan'] else ""
        plotted_labels['dongguan'] = True
        plt.scatter(lon_points[i], lat_points[i], color='orange', label
            =label)
        index4.append(i)
    else:
        label = 'others' if not plotted_labels['others'] else ""
        plotted_labels['others'] = True
        plt.scatter(lon_points[i], lat_points[i], color='blue', label=
            label)

# 添加图例
plt.legend(loc='best')
# 设置标题和轴标签
plt.title('Task Point Distribution')
plt.xlabel('Longitude')
plt.ylabel('Latitude')

# 显示图像
plt.show()


import matplotlib.pyplot as plt
import numpy as np

# 城市名称
```

```
cities = ['Guangzhou', 'Shenzhen', 'Foshan', 'Dongguan']

# 各城市的平均定价
average_prices = [68.067, 67.312, 70.343, 71.443]  # 替换为你的实际数据

# 各城市的订单完成率
completion_rates = [0.608, 0.209, 1.0, 0.661]  # 替换为你的实际数据

# 创建图形和子图
fig, ax1 = plt.subplots(figsize=(10, 6))

# 设置位置
x = np.arange(len(cities))
width = 0.4

# 绘制平均定价的柱状图
ax1.bar(x - width/2, average_prices, width, label='Average Price',
    color='skyblue')

# 设置平均定价的 y 轴范围，从 50 开始
ax1.set_ylim(60, max(average_prices) + 3)

# 创建第二个 y 轴来显示订单完成率
ax2 = ax1.twinx()
ax2.bar(x + width/2, completion_rates, width, label='Completion Rate',
    color='orange')

# 设置标题和轴标签
ax1.set_xlabel('City')
ax1.set_ylabel('Average Price')
ax2.set_ylabel('Completion Rate')
ax1.set_title('Average Price and Completion Rate by City')

# 设置 x 轴的刻度
ax1.set_xticks(x)
ax1.set_xticklabels(cities)

# 添加图例
ax1.legend(loc='upper left')
ax2.legend(loc='upper right')

# 显示图形
plt.show()
```

问题二代码

```
# 读取Excel文件
df_task = pd.read_excel('../data/Q1.xlsx')
df_member = pd.read_excel('../data/Q2.xlsx')
import math
def euclidean_distance(lat1, lon1, lat2, lon2):
    # 将纬度和经度转换为相同单位
    delta_lat = lat2 - lat1
    delta_lon = lon2 - lon1
    # 简单的欧几里得距离
```

```
    return math.sqrt(delta_lat ** 2 + delta_lon ** 2) * 111  # 1度大约
        为111公里

# 统计每个任务点1km内的顾客数量
result_member = []
for index, task in df_task.iterrows():
    task_lat, task_lon = task['任务纬度gps'], task['任务经度gps']
    count = 0
    score = 0
    members_within_1km = []  # 用于存储1km内的顾客信息

    # 遍历1km内的所有顾客
    for _, member in df_member.iterrows():
        member_lat, member_lon = map(float, member['会员位
            置(GPS)'].split())
        distance = euclidean_distance(task_lat, task_lon, member_lat,
            member_lon)

        if distance <= 1:
            count += 1
            members_within_1km.append({
                '会员ID': member['会员编号'],  # 可选，用于标识会员
                '信誉分': member['信誉值'],
                '距离': distance
            })
            # 分数累积
            score += (member['预订任务限额'] + math.sqrt(member['信誉
                值'])) * (1-distance) * 0.01

    result_member.append({
        '任务号码': task['任务号码'],
        '1内顾客数量km': count,
        '顾客信息': members_within_1km,  # 存储顾客的信誉分和距离
        '该任务点的会员因子分数': score       # 该任务点的会员因子分数
    })
# 转换为DataFrame并显示
df_result_member = pd.DataFrame(result_member)

# 统计每个任务点1km内的其他任务点数量
result_cpt = []
for index, task in df_task.iterrows():
    task_lat, task_lon = task['任务纬度gps'], task['任务经度gps']
    count = 0
    score = 0
    tasks_within_1km = []  # 用于存储1km内的顾客信息
    # 遍历1km内的所有其他任务点
    for _, oppotask in df_task.iterrows():
        oppotask_lat, oppotask_lon = oppotask['任务纬
            度gps'], oppotask['任务经度gps']
        distance = euclidean_distance(task_lat, task_lon, oppotask_lat,
            oppotask_lon)
        if distance <= 1 and distance != 0:
            count += 1
            tasks_within_1km.append({
                '任务号码': oppotask['任务号码'],  # 可选，用于标识会员
                '距离': distance
            })
            # 分数累积
            score += min(math.sqrt(1 / distance) * 0.1, 5)
```

```
    result_cpt.append({
        '其他任务ID': task['任务号码'],
        '1内其他任务点数量km': count,
        '其他任务点的信息': tasks_within_1km,  # 存储顾客的信誉分和距离
        '该任务点的竞争因子分数': score       # 该任务点的竞争因子分数
    })
# 转换为DataFrame并显示
df_result_cpt = pd.DataFrame(result_cpt)
# 首先将会员因子分数和竞争因子分数合并到 df_task 中
df_new_task = df_task.copy()  # 复制原始任务数据
# 添加会员因子分数列
df_new_task['会员因子分数'] = df_result_member['该任务点的会员因子分数']
# 添加竞争因子分数列
df_new_task['竞争因子分数'] = df_result_cpt['该任务点的竞争因子分数']
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# 1. 准备数据
X = df_new_task[['任务纬度gps', '任务经度gps', '任务标价', '会员因子分数', '竞争
    因子分数']]
y = df_new_task['任务执行情况']

# 2. 数据分割为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.2, random_state=42)

# 3. 调参
best_accuracy = 0
best_n_estimators = 0
accuracies = []

for n in range(10, 201, 10):
    model = RandomForestClassifier(n_estimators=n, random_state=42)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    accuracies.append((n, accuracy))

    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_n_estimators = n

# 输出最佳结果
print(f"Best n_estimators: {best_n_estimators} with Accuracy: {
    best_accuracy}")

# 打印每个 n_estimators 对应的准确率
for n, accuracy in accuracies:
    print(f"n_estimators: {n}, Accuracy: {accuracy}")

df_new_task_except = df_new_task.drop(index=eccept_index)
X_new_except = X_new.drop(index=eccept_index)
import numpy as np

# 初始化参数
best_k = 1
best_w = 1
```

```
best_res = 0


# 网格搜索 k 和 w 的值，步长为 0.2
for k in np.arange(1, 31, 0.2):  # np.arange(1, 11, 0.2) 代表从 1 到 10，步长为 0.2
    for w in np.arange(1, 31, 0.2):

        price = []  # 初始化价格列表

        for i in df_new_task_except.index:  # 使用 .index 确保按索引遍历
            # 获取会员因子分数和竞争因子分数
            m_factor = df_new_task_except.loc[i, '会员因子分数']
            c_factor = df_new_task_except.loc[i, '竞争因子分数']

            # 计算该任务点的价格
            task_price = 65 + k * m_factor + w * c_factor

            # 将计算的价格添加到 price 列表
            price.append(task_price)

        # 更新 '预测合理价格' 列
        X_new_except['任务标价'] = price

        # 使用训练好的模型进行预测
        y_new_pred = model.predict(X_new_except)

        # 将新的预测结果存储到 df_new_task 的 '预测任务执行情况' 列中
        df_new_task_except['预测任务执行情况'] = y_new_pred

        # 计算当前配置下预测的任务执行情况的总和（比如任务完成的数量）
        res = df_new_task_except['预测任务执行情况'].sum()

        # 更新最佳参数和结果
        if res > best_res:
            best_k = k
            best_w = w
            best_res = res

print('最好的 k 为: ', best_k, 'w 为: ', best_w, 'res 为: ', best_res)
X_new_except['任务标价'].mean()
X_new_except['任务标价'].sum()
```

问题三代码

```
# 读取Excel文件
df_member = pd.read_excel('../data/Q2.xlsx')
df_task = pd.read_excel('../data/combined_tasks.xlsx')
from sklearn.cluster import DBSCAN
import numpy as np

# 准备地理坐标数据
coords = df_task[['任务纬度gps', '任务经度gps']].values

# 定义1公里内的任务点为一个簇
kms_per_radian = 6371.0088  # 地球半径，单位为公里
```

```
epsilon = 1 / kms_per_radian  # 转换为弧度

# 初次聚类
db = DBSCAN(eps=epsilon, min_samples=1, algorithm='ball_tree', metric='
    haversine').fit(np.radians(coords))
df_task['initial_cluster'] = db.labels_

# 初始化 final_cluster 列
df_task['final_cluster'] = -1

# 初始化聚类标签
final_cluster_id = 0

# 处理初次聚类中点数超过4个的类
for cluster_id in np.unique(df_task['initial_cluster']):
    sub_cluster = df_task[df_task['initial_cluster'] == cluster_id]

    if len(sub_cluster) > 4:
        sub_coords = sub_cluster[['任务纬度gps', '任务经度gps']].values

        # 将超出4个点的聚类再次进行划分
        num_subclusters = int(np.ceil(len(sub_cluster) / 4))
        for i in range(num_subclusters):
            start_idx = i * 4
            end_idx = min(start_idx + 4, len(sub_cluster))
            df_task.loc[sub_cluster.index[start_idx:end_idx], '
                final_cluster'] = final_cluster_id
            final_cluster_id += 1
    else:
        df_task.loc[sub_cluster.index, 'final_cluster'] =
            final_cluster_id
        final_cluster_id += 1

# 检查每个聚类的大小
cluster_sizes = df_task['final_cluster'].value_counts()
print(cluster_sizes)
import matplotlib.pyplot as plt
import pandas as pd

# 统计每个聚类的数量
cluster_counts = df_task['final_cluster'].value_counts()

# 将每个任务点的颜色分配为红色或蓝色
colors = df_task['final_cluster'].apply(lambda x: 'red' if
    cluster_counts[x] > 1 else 'blue')

# 画散点图
plt.figure(figsize=(10, 8))
plt.scatter(df_task['任务经度gps'], df_task['任务纬
    度gps'], c=colors, s=5)  # 调整点的大小为50

# 添加图例
plt.scatter([], [], c='red', label='Clusters (>=2 points)')
plt.scatter([], [], c='blue', label='Single points')

plt.title('Task Points Clustering')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
```

```
plt.legend(loc='best')
plt.show()

def predict_completion_rate(cluster_id):
    tasks_in_cluster = df_task[df_task['final_cluster'] == cluster_id]
    completed_tasks = tasks_in_cluster['任务执行情况'].sum()
    total_tasks = len(tasks_in_cluster)
    completion_rate = completed_tasks / total_tasks
    return completion_rate

# 计算每个聚类的任务完成率
df_task['预测完成
    率'] = df_task['final_cluster'].apply(predict_completion_rate)

# 确保列名正确无误
if '预测完成率' in df_task.columns:
    # 将任务完成率进行二分类，> 0.5 视为已完成，<= 0.5 视为未完成
    df_task['任务是否完成'] = df_task['预测完成
        率'].apply(lambda x: 1 if x >= 0.5 else 0)

    # 统计已完成任务的数量
    completed_tasks_count = df_task['任务是否完成'].sum()

    # 查看统计结果
    print(f"已完成的任务数量: {completed_tasks_count}")
```

问题四代码

```
def optimize_price(base_price, cluster_size, distance_from_center):
    """基础价格:
    base_price聚类规模:
    cluster_size距离聚类中心的距离:
    distance_from_center
    """
    cluster_adjustment = 1 + 0.05 * (cluster_size - 1)  # 聚类规模调整因子
    distance_adjustment = 1 - 0.05 * distance_from_center  # 距离因素调整
        因子
    return base_price * cluster_adjustment * distance_adjustment

# 计算每个聚类的中心点
cluster_centers = df_task.groupby('final_cluster')[['任务纬度gps', '任务经
    度gps']].mean()

# 初始化调整后的价格列
df_task['优化后的任务标价'] = df_task['任务标价']  # 以原始标价作为基础

# 对每个任务点执行优化定价
for cluster_id, center in cluster_centers.iterrows():
    cluster_tasks = df_task[df_task['final_cluster'] == cluster_id]
    cluster_size = len(cluster_tasks)

    for idx, task in cluster_tasks.iterrows():
        # 计算任务点到聚类中心的距离
        distance_from_center = np.sqrt((task['任务纬度gps'] - center['任务
            纬度gps']) ** 2 +
                                       (task['任务经度gps'] - center['任务
                                           经度gps']) ** 2)
```

```
        # 优化价格
        optimized_price = optimize_price(task['任务标
            价'], cluster_size, distance_from_center)
        df_task.at[idx, '优化后的任务标价'] = optimized_price

# 查看优化后的数据
print(df_task[['任务纬度gps', '任务经度gps', '任务标价', '优化后的任务标
    价', 'final_cluster']])

df_task['优化后的任务标价'].mean()
df_task['优化后的任务标价'].sum()
```